UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

BRENDA SEARA BARCELOS
RENATO RICARDI

DESENVOLVIMENTO DE UNIDADE INSTRUCIONAL PARA ENSINAR COMPUTAÇÃO EXTRACLASSE

FLORIANÓPOLIS
2016

Brenda Seara Barcelos
Renato Ricardi

DESENVOLVIMENTO DE UNIDADE INSTRUCIONAL PARA ENSINAR COMPUTAÇÃO EXTRACLASSE

Trabalho de Conclusão de Curso apresentado ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharel em Sistemas de Informação.

Orientador: Elder Rizzon Santos

FLORIANÓPOLIS

2016

Brenda Seara Barcelos
Renato Ricardi

DESENVOLVIMENTO DE UNIDADE INSTRUCIONAL PARA ENSINAR COMPUTAÇÃO EXTRACLASSE

Trabalho de conclusão de curso submetido ao Departamento de Informática e Estatística da Universidade Federal de Santa Catarina para a obtenção do Grau de Bacharelado em Sistemas de Informação.

Orientador:

_____

Prof. Elder Rizzon Santos
UFSC

Banca Examinadora:

_____

Prof. Dr. Jean Carlo Rossa Hauck
UFSC

_____

Profa. Dra. Juliana Eyng
UFSC

AGRADECIMENTOS

Deixo aqui meus agradecimentos:

A nossas famílias e amigos, por sempre fornecerem apoio em todos os momentos necessários.

Ao Professor Elder pela orientação, atenção, carinho e paciência durante todo o período de desenvolvimento deste trabalho.

Ao Professor Diógenes Becker, por toda a ajuda e interesse em contribuir com a avaliação da unidade instrucional.

RESUMO

A tecnologia avança significativamente a cada dia que passa em escala mundial. Ela invadiu lares, empresas, instituições, tornando a sociedade mais informatizada, tornando assim a familiaridade com tecnologias e computação competências básicas de um cidadão do século XXI (SOARES; ALVES, 2012). Contudo sua importância não é vista como um fator primordial no aprendizado de crianças e jovens. A computação é tida como uma área de estudo que deve ser abordada somente em instituições de ensino superior. Devido a ausência da computação na educação de crianças e jovens existem iniciativas com o âmbito de ensinar algo nesta área. Para incentivar o interesse desse público é proposto um modelo de unidade instrucional para ensinar computação extraclasse para crianças e jovens com faixa etária de 8 à 14 anos. O objetivo do presente trabalho é desenvolver um design instrucional com base no estudo de metodologias e modelos de desenvolvimento de unidades instrucionais alinhado ao currículo CSTA K-12. Este currículo define um conjunto de normas, projetadas para fornecer a base para um currículo completo de ciências da computação. Para o desenvolvimento da unidade instrucional do presente trabalho foi desenvolvido um modelo de design instrucional apropriado para o ensino à distância. É esperado que após a realização desta unidade que os jovens adquiram interesse pela aprendizagem de computação e desenvolvam cada vez mais o raciocínio lógico e criativo, podendo assim criar um interesse ainda maior pela área de computação.

Palavras-chave: unidade instrucional, computação física, programação, Scratch, crianças e jovens, arduíno.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ACM – *Association for Computing Machinery*

ADDIE – *Analyze, Design, Develop, Implement, Evaluate*

API - *Application Programming Interface*

ARM - *Advanced RISC Machine*

BYOB - *Build Your Own Blocks*

CSTA – *Computer Science Teachers Association*

EAD - Ensino à distância

GUI - *Graphical User Interface*

IDE - *Integrated Development Environment*

IoT - *Internet of Things*

ISD – *Instructional System Development*

I/O - *Input/Output*

LED - *Light Emitting Diode*

MDF - *Medium-Density Fiberboard*

MIT – *Massachusetts Institute of Technology*

OBI - Olimpíada Brasileira de Informática

TI - Tecnologia da Informação

URL - *Uniform Resource Locator*

USB - *Universal Serial Bus*

SUMÁRIO

1. INTRODUÇÃO

Uma das grandes necessidades urgentes no século XXI é o aprimoramento do nível de compreensão da computação tanto no campo profissional quanto no campo acadêmico (CSTA, 2011). É uma necessidade também incluir outros conceitos tais como Gestão de Sistemas de Informação, Tecnologia da Informação, Matemática e outras ciências exatas (CSTA, 2011). Para que isso seja realmente aprimorado na sociedade, cada um deve compreender, pelo menos, os princípios básicos de computação (CSTA, 2011). Este conceito só tem sido mostrado nos níveis de graduação e subsequentes (CSTA, 2011). Ele pode ser definido como o estudo de computadores e dos processos algorítmicos, incluindo seus princípios básicos, impactos na sociedade e suas aplicações (CSTA, 2011).

No Brasil e em outros países subdesenvolvidos, o cenário em relação a computação é bem preocupante pois ela é vista somente quando o indivíduo está cursando algum curso superior na área, o que acaba acarretando em uma falta de conhecimento na área (DIOGOMNZ, 2015). Como resultado dessa falta, a sociedade acaba não sendo bem capacitada sobre o conceito de computação, a ponto de que a mesma sofre uma grave escassez de cientistas da computação de todos os níveis, o que é provável que continue no futuro (CSTA, 2011). Além disso, poucas escolas têm o conhecimento de disponibilizar computadores para turmas e ainda de contratar profissionais da área para oferecer um ensino de qualidade na área de computação. Também são fatores cruciais, a falta de motivação, interesse e conscientização da importância da computação nos dias de hoje (PINTO; ROCHA; VILARIM, 2016)

Com o intuito de melhorar este problema, todo indivíduo deve estar por dentro desta área de atuação, não somente para crescer ou se tornar competitivo, mas também para ser produtivo. O indivíduo que não está incluso neste meio pode se prejudicar bastante ou até então perder muito tempo em uma tarefa que poderia ser concluída em muito pouco tempo. Para ser cidadãos bem instruídos em um mundo de computação intensiva e estar preparado para as carreiras do século XXI, o indivíduo deve compreender claramente os princípios e práticas de computação (CSTA, 2011).

Uma solução para contornar esse tipo de problema sobre a falta de profissionais qualificados na área de computação, seria inserir o ensino de computação já no Ensino Fundamental. Deveria ser de responsabilidade das escolas incluir na sua grade curricular matérias vinculadas à computação, contudo não é o que ocorre na maioria das escolas,

principalmente no Brasil (NOGUEIRA, 2011). Os governos da Austrália, Inglaterra e outros países desenvolvidos, por exemplo, têm incentivado as escolas a aumentar o enfoque das aulas em áreas de ciência, tecnologia, engenharia e matemática. O ensino de computação no Ensino Fundamental deve abordar várias áreas de conhecimento tais como pensamento computacional, colaboração, impactos éticos, globais e na comunidade, computadores e dispositivos de comunicação e programação. Essas áreas devem compor um ciclo, onde o aluno consiga englobar todos os conceitos de uma forma para ser utilizada não só pelo mesmo no futuro mas também como cidadão (CSTA, 2011). O fundamental entendimento por computação permite os alunos não só serem consumidores de tecnologia mas também criadores e inovadores, capazes de projetar sistemas computacionais para melhorar a qualidade de vida de todas as pessoas (CSTA, 2011).

Quando se tem a oportunidade de aprender algo novo, os jovens estudantes desfrutam da sensação de domínio e mágica que a programação fornece (CSTA, 2011). Os estudantes mais velhos são atraídos pelas combinações de arte, narrativa, design, programação e puro prazer que vem da criação de seus próprios mundos virtuais (CSTA, 2011). Misturando computação com outros interesses é possível também desenvolver ricas oportunidades para a aprendizagem (CSTA, 2011). Estudantes com um interesse na música, por exemplo, podem aprender sobre música e áudio digital (CSTA, 2011). Este campo integra eletrônica, matemática, teoria musical, programação de computadores, e um ouvido afiado para o que soa bonito, harmonioso, ou o que simplesmente é interessante (CSTA, 2011). Além desse aprendizado todo, através da computação é possível resolver problemas urgentes com o intuito de ajudar a sociedade como por exemplo, a prevenção ou cura de doenças (KOKAY, 2015). Ela também amplia a nossa compreensão de nós mesmos como sistemas biológicos e de nossa relação com o mundo que nos rodeia (CSTA, 2011).

No Brasil e no mundo inteiro já existem muitas iniciativas para ensinar a computação focando em algo que o aluno possa deixar de lado aquele conceito de matéria obrigatória e começar a se envolver no mundo real, em um mundo em que as coisas possam acontecer (HERINGER, 2015). Uma dessas iniciativas é o Code Club Brasil, que tem como objetivo fazer com que cada criança tenha a oportunidade de aprender a programar (CODECLUBRASIL, 2016; HERINGER, 2015) Para realizar esta tarefa, a iniciativa possui um material de ensino e uma estrutura de voluntariados que apoia a realização de atividades extracurriculares ligadas à programação de computadores (CODECLUBRASIL, 2016). Uma outra iniciativa que pode ser citada é a Computação na Escola que é uma iniciativa dedicada a

aumentar o ensino de computação no Ensino Fundamental e Médio (HERINGER, 2015; COMPUTACAONAESCOLA, 2016). Ela tem como um de seus objetivos a criação de unidades instrucionais voltadas à computação física utilizando tecnologias de *software* de fácil entendimento e manipulação e *hardware* de baixo custo (COMPUTACAONAESCOLA, 2016).

A computação física tem como conceito explorar computacionalmente artefatos e ambientes da realidade (CAMARATA, 2003). Através do seu estudo é possível utilizar a realidade para sintetizar as ideias e fornecer uma base para olhar para o futuro da computação em nosso ambiente físico (CAMARATA, 2003). Além disso, a computação física inclui o uso de computadores, controladores e *softwares* que ligados a sensores e atuadores, permitem construir sistemas e aparelhos autômatos que percebem a realidade e respondem com ações físicas (CAMARATA, 2003). Resumidamente falando é a interação entre o homem e a máquina através de dispositivos que permitem a captação e manipulação de dados referentes a objetos e espaços reais por meio de sistemas mecânicos e/ou eletrônicos (CAMARATA, 2003).

Existem diversas iniciativas para ensinar computação por meio da computação física, utilizando por exemplo *Lego Mindstorm* (LEGO, 2015) que oferece um conjunto de peças tradicionais, bem conhecidas por todos, juntamente com sensores de toque, de intensidade luminosa e de temperatura, controlados por um sensor programável (LEGO, 2015). Porém, essa iniciativa não é tão acessível a qualquer pessoa que tenha interesse de adquiri-la. O custo para se obter este brinquedo de educação tecnológica é muito alto e nem sempre é uma opção de quem está interessado em aprimorar o ensino (RICHETTI, 2014; BHATTACHARYA, 2014). Outras alternativas então são utilizadas para tentar despertar este interesse nas crianças e nos jovens. Tais como *Pico Cricket*[1], *Botball*[2], *ModKit*[3], *GoGoBoard*[4] que possuem preços mais acessíveis porém não são tão completos e robustos quanto o *Lego Mindstorm*, no entanto podem ser início do despertar da criança por este assunto. Existem diversas dessas tecnologias de baixo custo que podem ser exploradas nos dias de hoje. Um exemplo delas é o Arduino, que é uma plataforma eletrônica de código aberto baseado em *hardware* e *software*

1 http://www.picocricket.com/whatisit.html

2 http://www.botball.org/what-is-botball

3 http://www.modkit.com/

4 http://www.gogoboardpi.org/

de fácil utilização (ARDUINO, 2015). É destinado para qualquer um que queira fazer projetos interativos (ARDUINO, 2015). Com o intuito de unir o conceito de computação física, existem também ferramentas para auxiliar na programação como por exemplo, a linguagem de programação Scratch (SCRATCH, 2015), que basicamente é uma linguagem em que é possível programar suas próprias histórias interativas e jogos (SCRATCH, 2015). O propósito de uma linguagem como Scratch é tornar mais fácil o aprendizado em programação, auxiliando os programadores iniciantes a superar barreiras de sintaxe. Através de ferramentas de programação como Scratch e da utilização de plataformas eletrônicas de *hardware* como o Arduino, é possível reproduzir funções e ações através de um boneco. Com a utilização destes recursos é possível fazer com que um boneco possa se movimentar, emitir sons e até mesmo consiga perceber algo chegando perto dele, o que torna a experiência com a tecnologia muito mais rica e interessante.

Para as crianças que possuem um conhecimento prévio destas tecnologias ou até mesmo tem a vontade de aprendê-las sem depender de algum curso ou instrutor da área, foi pensado em disponibilizar algo em que possa ser prático e fácil o engajamento nesta área. É esperado que através deste aprendizado, a criança possa, em casa criar novos projetos e realizar diversas outras tarefas. Para atingir o propósito deste aprendizado extraclasse, o objetivo do trabalho é desenvolver uma atividade instrucional com diferentes conteúdos para que desperte interesse em todas as idades e gostos. Para poder desenvolver esse tipo de atividade, o participante deve possuir alguns equipamentos básicos necessários tais como computador que possua a configuração mínima desejada, acesso à internet, algum kit compondo arduíno e outros componentes eletrônicos e o mais importante de tudo, muita criatividade e imaginação.

Assim pode-se observar que, existe aqui uma oportunidade de mostrar que o ensino da computação e a sua vivência é de grande importância, isto através da prática da programação em diversas atividades, popularizando cada vez mais a computação. Assim, a pergunta para a análise do presente trabalho consiste em: “Como desenvolver atividades extraclasse que mantenham o interesse de crianças (8-14 anos) na aprendizagem de computação”?

1.1 Objetivos

OBJETIVO GERAL

O objetivo do presente trabalho é o desenvolvimento e avaliação de uma unidade instrucional para o ensino de computação física e programação para crianças de 8 a 14 anos utilizando modelos de design instrucional apropriados.

OBJETIVOS ESPECÍFICOS

1. Analisar a fundamentação teórica sobre ensino, aprendizagem, modelos de design instrucional, computação física e a linguagens apropriadas para o ensino de programação para crianças;
2. Elaborar um modelo adequado para definir uma unidade instrucional extraclasse para crianças de 8 a 14 anos com intuito de ensinar programação e computação física;
3. Propor uma estrutura de computação física adequada para o público-alvo e com baixo custo;
4. Elaborar uma unidade instrucional para testar o modelo de design instrucional e a estrutura de computação física e de programação;
5. Avaliar a unidade instrucional elaborada.

DELIMITAÇÕES DO TRABALHO

A unidade desenvolvida deve ser realizada extraclasse, não possuindo metodologia para o ensino em sala de aula. A unidade é desenvolvida exclusivamente na linguagem Scratch. O presente trabalho será realizado de acordo com o regulamento vigente do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina em relação aos Trabalhos de Conclusão de Curso.

1.2 Metodologia

A metodologia de pesquisa utilizada neste presente trabalho é dividida em quatro etapas:

Etapa 1 – Análise e síntese da literatura referente ao ensino e aprendizagem, computação física e a linguagens de programação com foco educacional. Esta etapa tem como objetivo apresentar um embasamento teórico sobre o presente trabalho. Esta primeira etapa é dividida em:

A1.1 – Análise da teoria de computação física.

A1.2 – Análise da teoria de ensino e de aprendizagem.

A1.3 – Análise da teoria e aplicação das linguagens de programação com foco educacional.

Etapa 2 – Revisão sistemática de unidades instrucionais e estratégias de ensino atualmente utilizadas e voltadas ao ensino de computação física. Nesta etapa será realizada uma revisão sistemática da literatura seguindo o processo proposto por (KITCHENHAM, 1994). Essa etapa está dividida em:

A2.1 – Definição da revisão sistemática.

A2.2 – Execução da busca.

A2.3 – Extração e análise da informação.

Etapa 3 – Desenvolvimento de uma unidade instrucional para o ensino de computação física e programação extraclasse seguindo uma metodologia de design instrucional. Essa etapa está dividida em análise e desenvolvimento:

A3.1 – Análise do contexto das unidade instrucional e definição do objetivo de aprendizagem. Esta unidade instrucional vai conter materiais instrucionais que serão implementados e avaliados posteriormente. A implementação desta unidade fornece *feedbacks* acerca da aprendizagem de conceitos computação.

A3.2 - Desenvolvimento de um modelo de design instrucional para o ensino de computação física à distância.

A3.3 - Desenvolvimento da unidade instrucional

Etapa 4 – Aplicação e avaliação da unidade instrucional desenvolvida. Nesta etapa a unidade instrucional desenvolvida é aplicada a uma criança dentro dos requisitos do público alvo estudado e a um professor do ensino fundamental e médio, assim obtendo um *feedback*. Essa etapa está dividida em:

A4.1 – Aplicação da unidade.

A4.2 – Execução da avaliação.

A4.3 – Análise dos dados coletados.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo abrange conceitos referentes ao processo de ensino e aprendizagem, design e unidade instrucional. Também são apresentados tópicos como definição de

computação física, *hardware, software,* conceito e tipos de Arduino e uma visão geral sobre o ambiente e linguagem de programação Scratch.

2.1 Computação física

Computação física é concebida em sistemas digitais, incluindo computadores, controladores e *softwares* que ligados a sensores e atuadores, permitem construir sistemas e aparelhos autômatos que percebem a realidade e respondem com ações físicas a esta realidade (CAMARATA, 2003). Ou seja, é a interação entre o homem e a máquina através de dispositivos que permitem a captação e manipulação de dados referentes a objetos e espaços reais por meio de sistemas mecânicos e/ou eletrônicos (CAMARATA, 2003). A figura 1 ilustra o funcionamento geral da computação física.

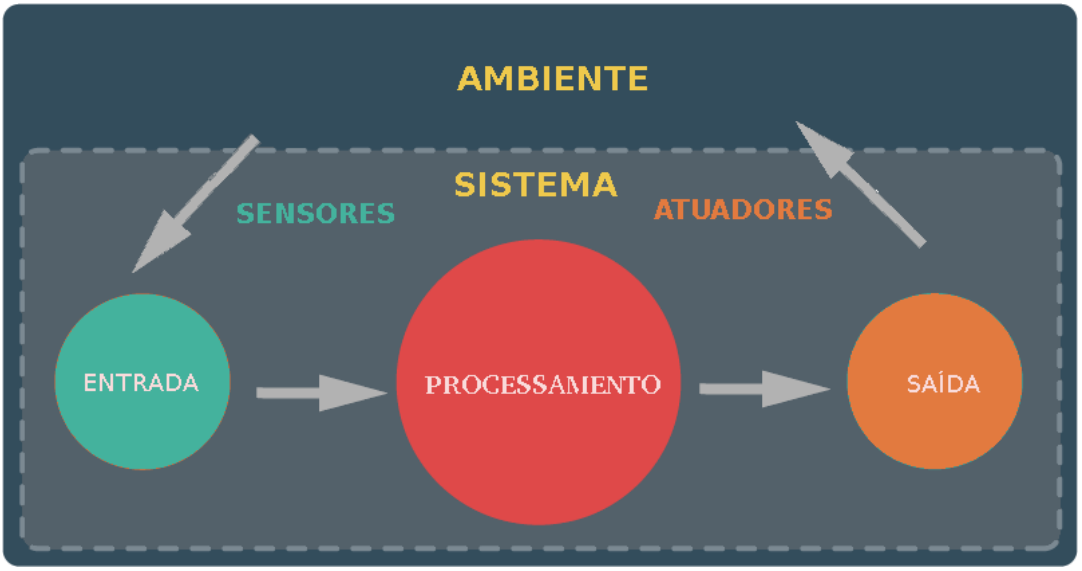


Figura 1 - Funcionamento lógico de computação física (Fonte: Elaborada pelo(a) autor(a) à partir de COTE, 2015)

O ambiente interage com o sistema por meio da manipulação dos sensores e atuadores. O processamento representa a lógica do sistema que decide o que fazer com as entradas (informações transmitidas aos sensores) e saídas (informações transmitidas aos atuadores).

Por meio da computação física, pessoas podem interagir com os objetos do mundo real através da implantação de computadores minúsculos e alguns circuitos eletrônicos (KATO, 2010). Tudo dependerá mais da criatividade do que limitações técnicas pois a computação física nos deixa criar diversos tipos de aplicações, desde diversão passando pela

arte, automação residencial a até ajudar outras pessoas, isso permite que todo mundo seja criador de tecnologia (LEMOS, 2014). As figuras 2.1 e 2.2 mostram alguns projetos de interação *hardware* e *software* juntamente com a computação física.

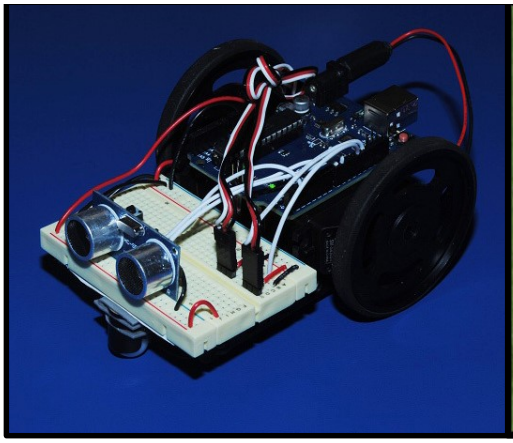


Figura 2.1 - Carro feito com Arduino
(Fonte: INTOROBOTICS, 2014)

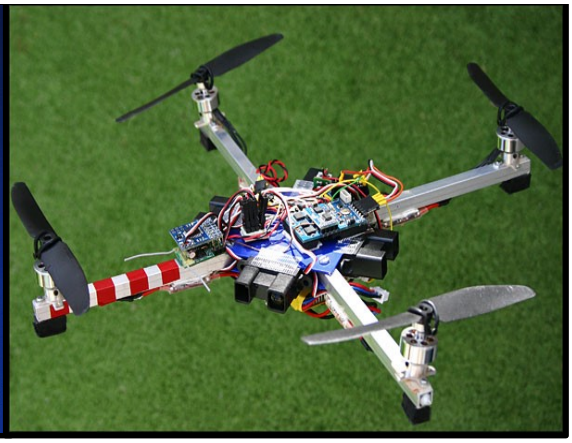


Figura 2.2 - *Drone* feito com Arduino
(Fonte: INTOROBOTICS, 2014)

A imagem 2.1 apresenta um projeto feito com Arduino de um robô com rodas com um sensor de ultrassom anexado na frente e programado para detectar qualquer obstáculo que possa interferir o caminho do robô. Já a imagem 2.2 é um projeto de *drone* também feito com Arduino que automaticamente evita obstáculos com base nas informações recebidas de sensores de distância e 4 sensores de infravermelho com alcance de 1,5 metros.

Os diversos projetos elaborados com Arduino podem ser construídos com a utilização de microcontroladores e podem ser controlados através de um plataforma de *software.* O campo da computação física engloba todas as áreas que permitem construir equipamentos digitais de computação que interagem com, e respondem à, realidade física analógica que os rodeia, usando *software* e *hardware* para este fim (DREAMFEEL, 2009). O desenvolvimento de ambos vem se tornando cada vez mais facilitado devido ao surgimento e popularização dos kits educacionais. Estes kits permitem a implementação de *hardware* e/ou *software* de forma facilitada (COMPUTACAOFISICABR, 2014). A presença de vários tipos de dispositivos de computação física em ambientes educacionais é atribuível a muitas iniciativas dos últimos anos de pesquisa (BLIKSTEIN, 2013).

2.1.1 Hardware

Hardware engloba todos os dispositivos físicos e equipamentos utilizados no processo de informações (MARTINS, 2007). Estes dispositivos físicos e equipamentos são compostos

de circuitos de fios e luz, placas, utensílios, correntes, e qualquer outro material em estado físico, que seja necessário ao funcionamento de um computador (CHICOLI, 2008). Os elementos de *hardware* envolvidos tipicamente na computação física são microcontroladores, sensores e atuadores.

O microcontrolador é o *chip* de computador que controla a maioria dos dispositivos eletrônicos complexos que as pessoas utilizam todos os dias (IEEE, 2015). Os microcontroladores são pequenos sistemas de computação utilizados para fins de pouca memória e de baixa potência. A composição de um microcontrolador é basicamente *Central Processing Unit* (CPU), memória e dispositivos de entrada e saída. O esquema da composição do microcontrolador é mostrada na figura 3.

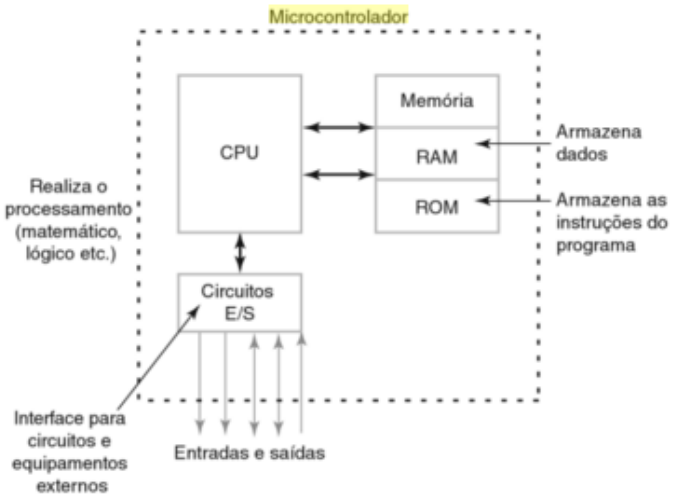


Figura 3 - Principais componentes de um microcontrolador (Fonte: FRENZEL JR, 2016)

A CPU tem como função realizar o processamento dos dados reais. Através de um conjunto de comandos ou instruções é possível torná-la programável, ou seja, dizer ao processador o que ele deve fazer. Um programa é formado por este conjunto de instruções à partir do momento em que elas são colocadas em sequência, com o objetivo de realizar alguma operação desejada. Estas instruções são representadas através de códigos que são armazenadas em uma memória (FRENZEL JR, 2016). A memória principal é externa as instruções e dados da CPU. Ela pode ser dividida em dois tipos: *random access memory* (RAM) e *read-only memory* (ROM). A memória RAM é muito mais proeminente, ela mantém o conteúdo desde que se forneça energia para o computador e perde o conteúdo quando a energia é desligada. A RAM também é chamada de memória temporária ou memória volátil. A memória ROM, por outro lado, retém o conteúdo mesmo depois que a

energia do computador é desligada. Portanto, ela é chamada de memória permanente ou memória não volátil (AKSOY; DENARDIS, 2008). Os dispositivos de entrada e saída fazem a ponte de comunicação para a CPU e a memória (FRENZEL JR, 2016). Além do microcontrolador, existem também outras partes tipicamente utilizadas na computação física a serem citadas tais como sensores e atuadores.

Sensor. Um sensor é definido como um dispositivo que recebe e responde a um estímulo ou um sinal. Sensores são necessários para medir sinais desconhecidos e parâmetros de um sistema de engenharia e seu ambiente. Essencialmente, os sensores são utilizados para monitorar e aprender sobre determinado sistema (SILVA, 2016). Além disso, estes dispositivos são sensíveis à alguma forma de energia do ambiente que pode ser luminosa, térmica, mensurada tais como temperatura, velocidade, posição, dentre diversas outras (COMPUTACAONAESCOLA, 2016).

Na computação física, os principais tipos de sensores utilizados são o de ultrassom, luz e temperatura (COMPUTACAONAESCOLA, 2016). O sensor de ultrassom mostrado na figura 4, por exemplo, é um dispositivo que funciona como um radar que utiliza a transmissão de ondas sonoras para identificar a distância entre o sensor e um objeto qualquer (COMPUTACAONAESCOLA, 2016).

Figura 4 - Ilustração de um sensor de ultrassom (Fonte: MODMYPI, 2014)

A composição do sensor inclui um emissor e um receptor de ultrassom (CALIN, 2014). A parte do emissor transmite ondas na frequência do ultrassom. Essas ondas ou parte delas são refletidas em algum objeto que estiver no campo de abrangência do sensor e retornam com ondas transmitidas ao receptor, conforme mostrado na figura 5.

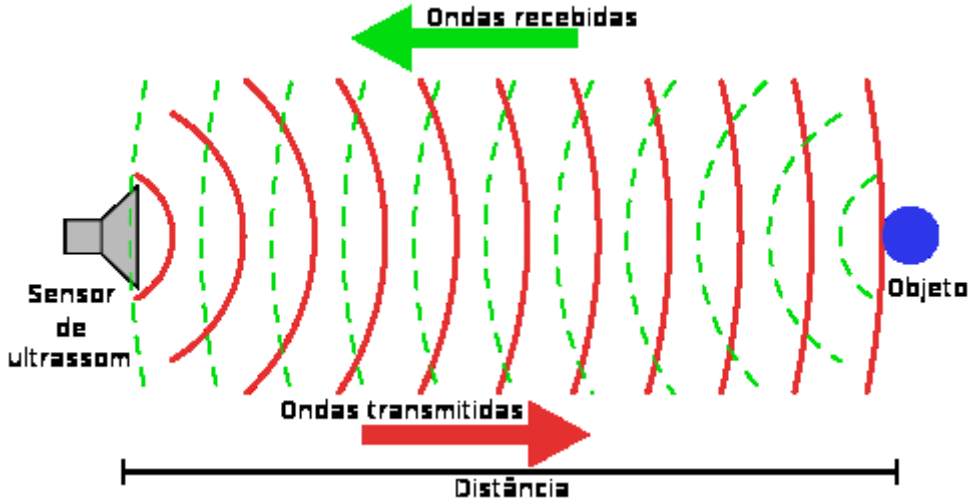


Figura 5 - Princípio de um ultrassom ativo (Fonte: Elaborada pelo(a) autor(a) à partir de CALIN, 2014)

O Arduino se comunica com o sensor de ultrassom para emitir pequenos pulsos e medir o tempo de resposta. Através do tempo resposta é possível saber a que distância algum objeto está do sensor (COMPUTACAONAESCOLA, 2016). O sinal emissor, ao colidir em algum obstáculo, é refletido na direção do sensor. Durante este processo, podemos saber o tempo que o sinal leva desde a sua emissão até a sua volta. Como sabemos a velocidade que o som trafega no ar, de posse do tempo que o sinal leva para fazer esse processo, conseguimos calcular a distância entre o sensor e o objeto em questão (ARAUJO, 2014).

Atuador. Um atuador é um elemento que produz movimento, atendendo a comandos que podem ser manuais, elétricos ou mecânicos. Atuadores são dispositivos controlados por um computador ou microcontrolador que permitem construir coisas automatizadas que atuam sobre o mundo real, modificando-o ou fazendo coisas reais acontecerem. Os motores são os atuadores mais comuns. Um servomotor é um motor controlado pelo Arduino e pode ser utilizado para a movimentação (COMPUTACAONAESCOLA, 2016). O servomotor é composto de um circuito de controle, um motor, uma combinação de engrenagens e um potenciômetro que é definido como um componente eletrônico que possui resistência elétrica ajustável (IGOE, 2004). A figura 6 ilustra o componente atuador.

Figura 6 - Ilustração de um atuador Micro Servo 9g SG90 TowerPro (FONTE: THOMSE, 2013)

Existem servomotores analógicos e digitais que possuem exatamente a mesma aparência porém se diferenciam na forma como sinalizam e processam informações (COKER, 2016). Os servomotores digitais usam um pequeno microprocessador para receber pulsos de voltagem de alta frequência (COKER, 2016). O servo digital envia 300 pulsos por segundo, onde o analógico só funciona em 50 pulsos por segundo. Esses pulsos mais rápidos provêm um torque consistente para tempos de resposta mais rápidos e suaves. Este é um grande benefício para os servos digitais, porém consomem muito mais energia comparados ao analógico (COKER, 2016).

Os movimentos executados pelos atuadores são bem variados, podendo realizar movimentos angulares, entre 0º e 180º graus (SUZUMORI, 2004). Tipicamente, essa variação angular pode ser posicionada através de um sinal de entrada que serve para a movimentação do eixo (COKER, 2016). Após aplicado o sinal, o servo manterá a posição angular do seu eixo sem mudanças. Se o sinal mudar, então a posição angular do eixo também muda (COKER, 2016). A figura 7 ilustra a angulação aplicada pelo servomotor.

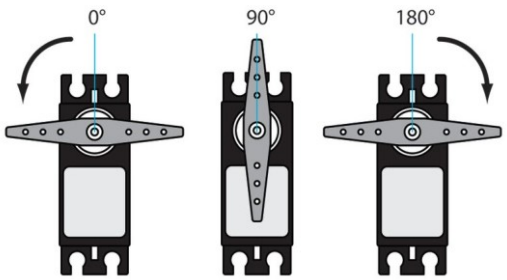


Figura 7 - Movimentação angular de um servomotor (PANDALAB, 2010)

Para ilustrar melhor o funcionamento de todo os conceitos comentados anteriormente, a Figura 8 mostra a arquitetura genérica do funcionamento conjunto entre microcontroladores, sensores e atuadores.

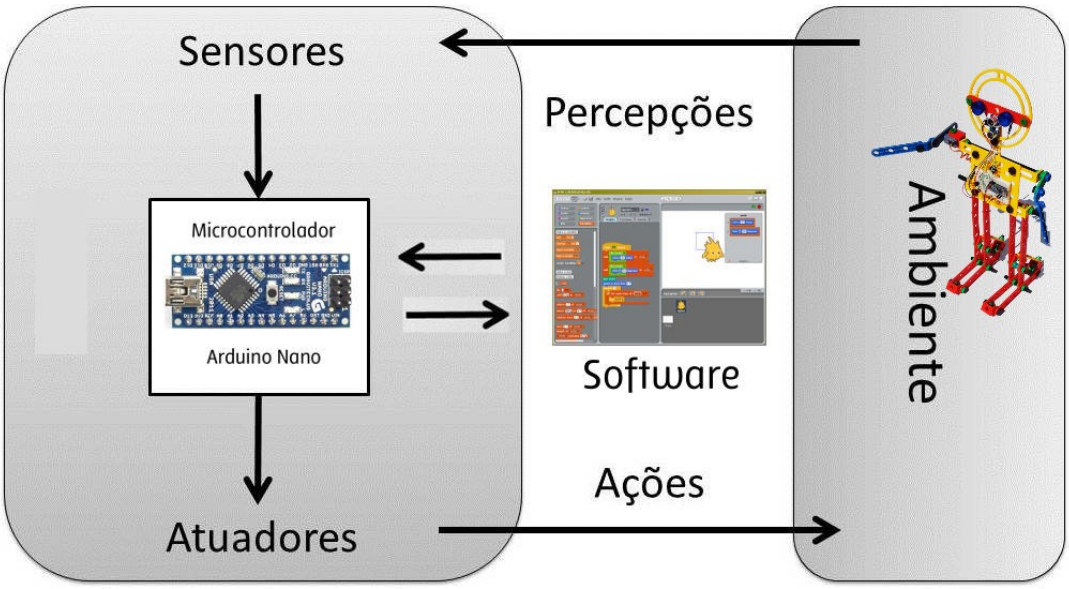


Figura 8 - Funcionamento conjunto de um microcontrolador, sensor e atuador através de uma interface de software (Fonte: Elaborada pelo(a) autor(a) à partir de JULIAN, 2016; ARDUINO, 2016; BERKELEY, 2016; COMPUTACAONAESCOLA, 2016)

Como pode ser observado na figura 8, os sensores realizam a coleta de dados do ambiente. Já os atuadores realizam ações de interação com o ambiente e o microcontrolador recebe comandos e instruções de uma plataforma forma de *software* para realizar determinada tarefa. Para dar continuidade aos conceitos abordados, o capítulo 2.1.1.1 descreve com mais delhates o microcontrolador Arduino.

2.1.1.1 Arduino

Atualmente existem diversos microcontroladores tais como Arduino (ARDUINO, 2016) e Raspberry PI (RASPBERRYPI, 2016) que estão disponíveis para qualquer pessoa a um preço favorável. Através destes microcontroladores é possível construir sistemas práticos e simples (MITROVIC et al., 2013). Ambas as plataformas nos dias de hoje são amplamente utilizadas, tornando a computação física bastante atraente para a educação (KATO, 2010). A tabela 1 faz um comparativo geral entre estas plataformas.

Tabela 1 - Comparativo entre plataformas Arduino e Raspberry Pi (Fonte: CODEDUINO, 2014; BELLAMY, 2013; ORSINI, 2014)

Imagem		
Plataforma	Arduino Uno	Raspberry Pi model B
Adequado para	Hardware	Software
Pinos de I/O digital	14	8
Internet	Utilizando um *shield*	Sim
Processador	ATmega328P	BCM2835 (ARM)
Memória RAM	2KB	256MB
Velocidade do Clock	16MHz	700MHz
Rede embutida	Não	Sim
Voltagem	7V a 12V	5V
USB	1	2
Flash	32KB	SD (2GB a 16GB)
Preço (abril, 2016)	R$83,00	R$143,00

O Raspberry Pi é um computador totalmente funcional, enquanto o Arduino é um microcontrolador, que é apenas um único componente de um computador (ORSINI, 2014). O Raspberry Pi é aproximadamente 40 vezes mais rápido do que um Arduino quando se trata de velocidade de *clock*. Além do mais, o Raspberry Pi tem 128.000 vezes mais memória RAM do que o Arduino e é considerado um computador independente que pode rodar um sistema operacional real em Linux (ORSINI, 2014). Também pode suportar duas portas USB, conexão sem fio à Internet e foi desenhado para atuar em alto nível (ORSINI, 2014; BELLAMY, 2013). No entanto, a plataforma Arduino possui um microcontrolador que consome pouca energia e que dá ao usuário completo controle de seu *hardware*. Através de uma *Integrated Development Environment* (IDE) própria, é possível escrever pequenos programas que fazem interface com vários tipos de dispositivos como sensores, atuadores, LCDs e outros microcontroladores (ALLAN, 2013; ORSINI, 2014; BELLAMY, 2013).

Assim, pode-se notar que cada uma dessas plataformas buscam atingir diferentes necessidades. O Raspberry Pi tem por objetivo ser uma plataforma de computação extremamente barata onde as pessoas possam programar em alto nível. O Arduino, por sua vez, lê e controla outros dispositivos de hardware. Certamente o Raspberry Pi possui maior capacidade computacional, entretanto o Arduino proporciona um fácil aprendizado de eletrônica e computação física. Além do mais, o microcontrolador Arduino para vários níveis de ensino, pode ajudar a melhorar o interesse da construção/concepção das coisas.

Por se tratar de uma plataforma de prototipagem eletrônica de hardware livre, o Arduino é amplamente utilizado em aplicações de robótica moderna devido ao seu baixo custo e facilidade de programação e recursos de prototipagem rápida (BANZI, 2012; CHENG; LI; WEST, 2015). Além disso, com a utilização de uma IDE pode-se escrever código com o intuito de manipular o *hardware*. A sua linguagem é baseada no C/C++ (KATO, 2010), que é uma linguagem que suporta múltiplos paradigmas e agrega o conceito de classes e de orientação à objetos. Sensores e atuadores podem ser facilmente conectados aos pinos digitais e analógicos de entrada e saída de um dispositivo Arduino, que possui um microcontrolador programado através de uma *Application Programming Interface* (API) própria (CHENG; LI; WEST, 2015).

A crescente complexidade das aplicações de computação física já levou a uma série de dispositivos compatíveis ao Arduino com processadores mais rápidos, maior armazenamento de memória *flash*, memórias com capacidades maiores e arquiteturas de entrada e saída mais complicadas (CHENG; LI; WEST, 2015). Projetos feitos com Arduino podem ser operados independentemente de outros *hardwares* e *softwares* ou também podem se comunicar com um *software* executado em um computador (ARDUINO, 2015). Os variados tipos de hardware Arduino podem ser comprados a um preço relativamente acessível, dependendo dos tipos de projetos (HERGER, 2015; BODARKY, 2015).

Tabela 2 - Comparativo entre alguns dos diversos tipos de Arduino

Imagem	Placa	Microcontrolador	Pinos de I/O Digital	Memória Flash	SRAM	EEPROM	Velocidade do Clock	Preço (R$)	Referência
	Uno	ATmega328P	14/6	32 KB	2 KB	1 KB	16 MHz	83	ARDUINO, 2016
	101	Intel Curie	14/4	196 KB	24 KB	-	32 MHz	119	ARDUINO, 2016
	Due	AT91SAM3X8E	54/12	512 KB	96 KB	-	84 MHz	150	ARDUINO, 2016
	Esplora	ATmega32u4	-	32 KB	2.5 KB	1 KB	16 MHz	359	ARDUINO, 2016; MULTILOGICA-SHOP, 2016
	Leonardo	ATmega32u4	20/7	32 KB	2.5 KB	1 KB	16 MHz	145	ARDUINO, 2016; MULTILOGICA-SHOP, 2016
	ADK	ATmega2560	54/15	256 KB	8 KB	4 KB	16 MHz	448	ARDUINO, 2016; MULTILOGICA-SHOP, 2016
	Nano	Atmel ATmega168 ou ATmega328	14/6	16 KB (ATmega168) ou 32KB (ATmega328)	1 KB (ATmega168) ou 2 KB (ATmega328)	512 bytes (ATmega168) ou 1 KB (ATmega328)	16 MHz	179	ARDUINO, 2016; MULTILOGICA-SHOP, 2016
	Zero	ATSAMD21G18	14/10	256 KB	32 KB	-	48 MHz	176	ARDUINO, 2016;
	Yun	ATmega32U4	20/7	32 KB	2.5 KB	1 KB	16 MHz	217	ARDUINO, 2016

A tabela 2 mostra um comparativo geral de alguns dos principais tipos de Arduino que existem hoje no mercado. As diferenças entre os tipos apresentados são basicamente a capacidade de processamento, voltagem de funcionamento, memória, número de conexões e preço. Todas elas podem ser conectadas a computadores e a grande maioria possui entrada USB. As placas podem ser modificadas livremente de acordo com a necessidade de cada projeto, mas é possível adquirir extensões chamadas *shields* que nada mais são do que placas que podem ser conectados no Arduino estendendo suas capacidades e os transformando (ARDUINO, 2015). Os diferentes *shields* seguem a mesma filosofia do manual original do Arduino, são fáceis de montar e baratos de produzir (ARDUINO, 2015). Existem *shields* para conexão à internet com cabo de rede, Wi-Fi, para controle de motores, para aumentar o número de portas USB, entre outras (SOUSA, 2015).

Dentre os tipos de Arduino citados na tabela 2, o Nano se destaca por ser uma placa pequena, poderosa e a mais completa das placas Arduino (MULTILOGICA-SHOP, 2016). Ela é composta por 14 pinos digitais, que podem ser usados tanto para entrada quanto para saída de dados. A voltagem dos pinos trabalham com 5V, e podem fornecer ou receber apenas 40mA de corrente máxima em cada um deles. O Arduino Nano pode ser alimentado por uma conexão mini-B USB, por uma fonte externa não regulada de 6 a 20 volts, ou por uma fonte externa regulada de 5V. A fonte de alimentação selecionada automaticamente é a de maior voltagem. Além disso, o Arduino Nano possui uma série de facilidades para se comunicar com um computador, outro Arduino ou outros microcontroladores (ARDUINO, 2016). Fisicamente o Arduino Nano pode ser usado diretamente na *protoboard* (MULTILOGICA-SHOP, 2016). A figura 9 mostra um Arduino nano juntamente com um *protoboard*.

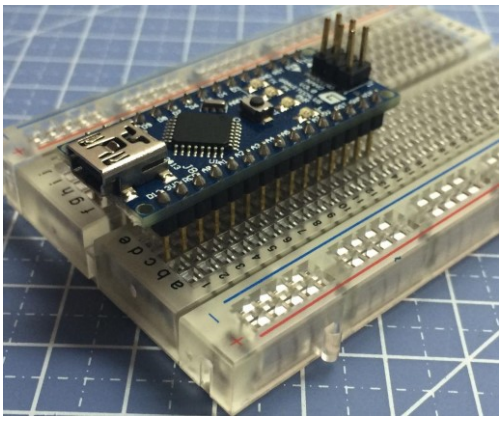


Figura 9 - *Protoboard* juntamente com o Arduino Nano (FONTE: ARAUJO, 2014)

A utilização da *protoboard* facilita a montagem pois oferece a praticidade de conexões que dispensam a parte dos *jumpers* e auxiliam na manipulação de circuitos eletrônicos (MULTILOGICA-SHOP, 2016). Porém a manipulação dessas *protoboards* exigem um conhecimento básico de circuitos pois necessitam a compreensão de termos da lógica matricial de *protoboard*, além de possuírem uma boa habilidade motora para posicionar corretamente todos os conectores necessários e componentes eletrônicos sobre a *protoboard* (COMPUTACAONAESCOLA, 2016).

2.1.1.2 Kits educacionais de computação física

Para tornar o ensino da computação física mais atrativa é ideal criar um processo que seja mais rápido e divertido, onde a necessidade de possuir habilidades técnicas avançadas não seja um impedimento ao aprendizado (COMPUTACAONAESCOLA, 2016). Com esse intuito de ajudar no aprendizado de uma forma mais simples, existem kits que facilitam bastante a manipulação de componentes complexos. Podem ser citados a iniciativa Computação na Escola, Grove Starter Kit (INTEL, 2016) e Lego Mindstorm (LEGO, 2016).

Na iniciativa Computação na Escola é utilizado o *Scratchboard* mostrado na Figura 10 que é uma placa de circuito impresso com um soquete para circuito integrado, no qual um microcontrolador Arduino Nano pode ser posicionado, e oito tomadas de telefone de 4 vias, onde dispositivos de automação podem ser conectados. Estes dispositivos podem ser tanto analógicos quanto digitais, de atuadores a sensores até dispositivos analógicos feitos em casa como interruptores de papel alumínio (COMPUTACAONAESCOLA, 2016).

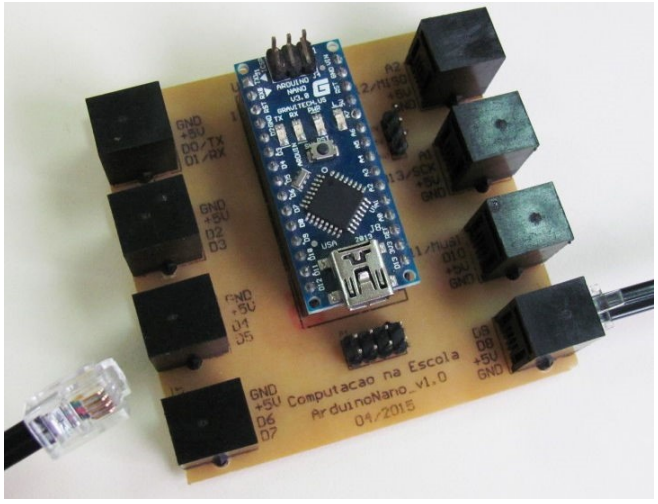


Figura 10 - Placa *Scratchboard* (Fonte: COMPUTACAONAESCOLA, 2016)

O kit disponibilizado pela iniciativa Computação na Escola utiliza além do *Scratchboard* uma lista materiais para a composição de um boneco (COMPUTACAONAESCOLA, 2016). Para a construção do kit é necessário o *Scratchboard* juntamente de uma placa Arduino Nano. O kit também inclui sensor de ultrassom para medir a distância a que se encontram objetos à frente do boneco, servomotor para realizar os movimentos, *Light Emitting Diode* (LED) para executar diferentes tipos de sinalização e resistores para evitar que o LED queime.(COMPUTACAONAESCOLA, 2016).

O Lego Mindstorm é um kit educacional muito conhecido para ensinar computação por meio da computação física. Ele oferece um conjunto de peças tradicionais, bem conhecidas por todos, juntamente com sensores de toque, de intensidade luminosa e de temperatura, controlados por um sensor programável (BAGNALL, 2002). Alguns destes componentes podem ser visualizados na figura 11.

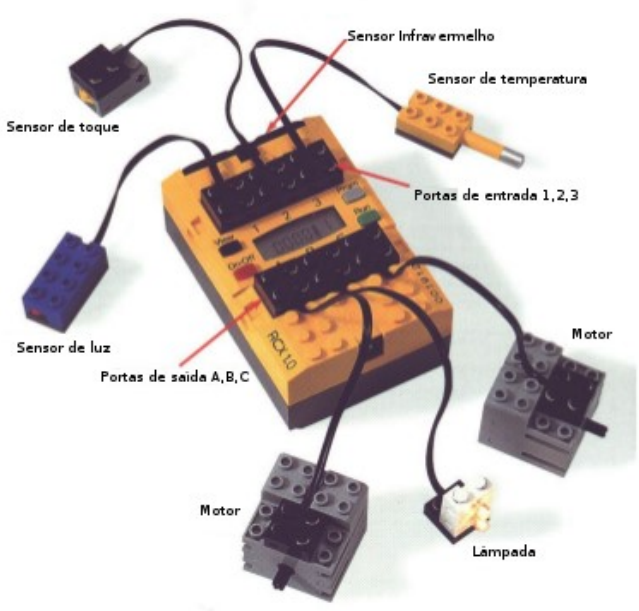


Figura 11 - Principais componentes presentes no Lego Mindstorm (Fonte: Elaborada pelo(a) autor(a) à partir de KISS, 2010)

Além disso, o Lego Mindstorm é amplamente utilizado em todos os níveis de ensino vários países e é disponibilizado através de um kit com o elemento de Lego e com uma unidade programável (KISS, 2010). A interface de programação é composta de blocos que são arrastados em uma linha lembrando um pouco as peças de montagem do Lego. Todo o

conteúdo contido nesta linha será compilado e o programa executará na ordem definida (KISS, 2010).

O *Grove Starter Kit* que consiste em um kit completo de desenvolvimento que é formado por um conjunto de sensores, atuadores e *shields* (INTEL, 2016). É uma solução tanto de *hardware* quanto de *software* para auxiliar na criação de projetos e explorar a área de *Internet of Things* (IoT) que tem o objetivo de tornar a realização de tarefas mais fáceis para os usuários utilizando dispositivos automatizados e sensores (BARASKAR, 2015; CHAOUCHI, 2010). As principais vantagens desse kit estão na utilização de sensores e do *Grove shield* que possuem 4 pinos conectores que permitem ligar outros dispositivos com facilidade em vez de gerenciar circuitos e sensores que utilizam placa de ensaio. Também há uma enorme variedade de módulos de sensores disponíveis para serem utilizados (BARASKAR, 2015). O *Grove Starter Kit* pode ser utilizado juntamente com uma placa Intel Galileo como mostrado na figura 12. Esta é a primeira placa na arquitetura Intel compatível com o Arduino e tem o mesmo objetivo que o Lego e o Arduino para computação física, em vez de perder tempo com fios e outros componentes eletrônicos, simplesmente é necessário plugar os componentes e sair utilizando (INTEL, 2016).

Figura 12 - *Grove Starter Kit* e placa Intel Galileo (Fonte: BARASKAR, 2015)

As plataformas citadas anteriormente possuem algumas diferenças importantes a serem mencionadas. O Arduino Nano disponibiliza de uma CPU de 16 MHz, o mesmo poder de processamento da CPU do Lego Mindstorm RCX. Já o Galileo tem um processador Intel muito poderoso, tendo 400 MHz disponíveis. Além disso o Galileo tem 512 KB de *Static Random Access Memory* (SRAM), o que também se torna vantajoso em relação ao

Nano e o RCX que possuem geralmente em torno 2KB e 32KB respectivamente (ARDUINO, 2016; JULIANO; RENNER; JAUREGUI, 2004). Tanto o Arduino quanto o Galileo são microcontroladores e executam o mesmo *software*. Assim podemos observar que as plataformas Galileo e RCX são mais rápidas comparadas ao Nano. No entanto, se o objetivo é realizar algum projeto mais simples o Nano é muito mais vantajoso pelo custo/benefício. O kit oferecido pela iniciativa Computação na Escola tem um preço menos elevado, custa em torno de R$160,00 comparado com o *Grove starter kit* que custa aproximadamente R$700,00 juntamente da placa Galileo e o Kit do Lego Mindstorm que custam de R$1.200,00 a R$3.000,00 dependendo do local de compra e de qual o kit tenha sido adquirido (ARDUINO, 2016; AMAZON; MERCADOLIVRE, 2016). Um dos pontos negativos do kit Computação na Escola é que ele ainda não está disponível para venda, porém ainda sim se torna uma alternativa mais em conta do que o Grove Starter Kit e o Lego Mindstorm que possuem produtos protegidos e de alto custo de aquisição (BAGNALL, 2002). Os preços comentados anteriormente referentes aos kits foram extraídos no mês de abril de 2016. Além disso, podemos ter alternativas na construção da parte da estrutura física dos kits, este assunto será abordado mais detalhadamente no subcapítulo 2.3 onde são mostrados diversos tipos de materiais para a montagem desta estrutura.

2.1.2 Software

Software é um conjunto de componentes lógicos ou sistemas de processamento de dados utilizados para operar um computador. Também pode ser descrito como programas que auxiliam o funcionamento e execução de um computador (UFPA, 2010). Com isso, *software* se define como uma sequência de instruções ou comandos a serem executadas e interpretadas por um processador (FERNANDES, 2002; UFPA, 2010).

Um ambiente de desenvolvimento de *software* é comumente chamado de IDE que consiste em um programa de computador normalmente usado para aumentar o grau de produtividade para o desenvolvimento de *software* (DUJMOVIC; NAGASHIMA, 2006). Com a utilização de uma IDE é possível que sequências de instruções sejam convertidas em ações. No contexto do presente trabalho, este recurso vai ser utilizado para comandar o funcionamento de um robô. O Arduino IDE é uma aplicação multiplataforma escrita em Java, uma linguagem de programação interpretada orientada a objetos que permite o desenvolvimento de aplicações (ARDUINO, 2016; JAVA, 2016). O Arduino IDE contém um editor de texto para escrita de código, área de mensagem, console de texto, além de outras funcionalidades. Ele se conecta ao *hardware* Arduino para carregar programas e realizar a

comunicação entre eles. O ambiente de desenvolvimento Arduino torna mais fácil a escrita do código e a comunicação com a placa (ARDUINO, 2016). Além disso, esta aplicação consegue utilizar bibliotecas que oferecem funcionalidades extras para elaboração de funcionalidades. Estas bibliotecas possuem pacotes e através da adição desses pacotes, é possível por exemplo, trabalhar com *hardwares* ou manipular dados (ARDUINO, 2016). Um *hardware* que pode ser gerenciado através da interface do Arduino é a placa Intel Galileo, sendo somente necessária a instalação de seu pacote, conforme mostra a figura 13.

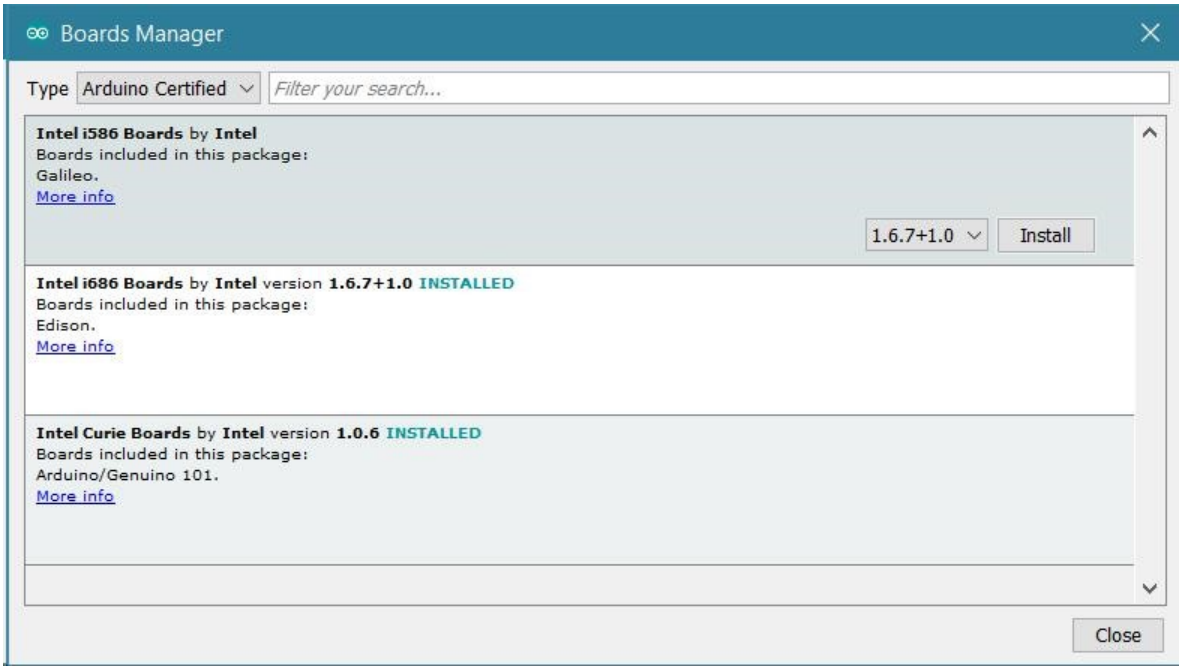


Figura 13 - Interface de gerenciamento de placas do Arduino IDE (Fonte: ARDUINO, 2016)

Após a instalação dos pacotes, é possível rodar programas para realizar uma diversidade de funcionalidades, desde a comunicação com os microcontroladores de software em um computador até ativação de um LED, que está ilustrado na figura 14. Os programas escritos utilizando o ambiente Arduino são chamados de *sketches* (ARDUINO, 2016). Além da codificação no editor de texto através da IDE, estes *sketches* podem ser carregados na placa Arduino utilizando esta interface (ARDUINO, 2016).

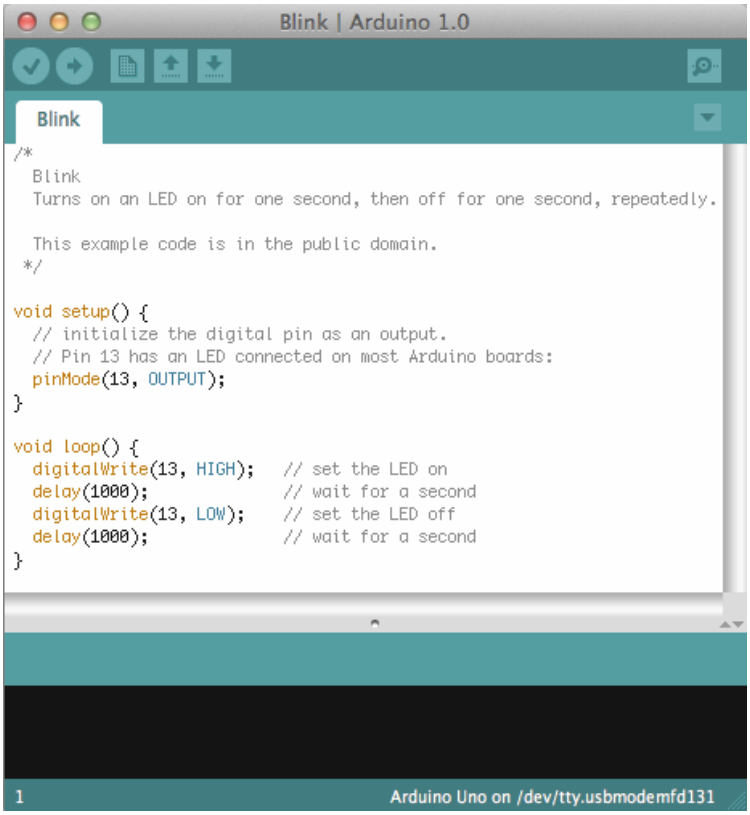


Figura 14 - Código ilustrado no Arduino IDE que ativa e desativa um LED a cada segundo (Fonte: ARDUINO, 2016)

Mesmo utilizando alguns recursos disponíveis, as tarefas de desenvolvimento ainda não se tornam algo de execução natural e tranquila (SANDOVAL-REYES et al., 2011). Tendo em vista o público alvo deste trabalho e com o intuito de facilitar a programar na IDE do Arduino com, foram utilizadas algumas linguagens de programação visual e de fácil aprendizado, tais como Scratch (SCRATCH, 2016), Snap! (BERKELEY, 2016), Alice (ALICE, 2016) e Blockly (GOOGLE, 2016).

Uma linguagem de programação visual é qualquer linguagem de programação que permite aos usuários criar programas manipulando elementos graficamente ao invés de especificá-los textualmente (JOST et al., 2014). Linguagens visuais partem do princípio de que imagens são mais facilmente entendidas do que código (SHU, 1988). Com isso, um programa é definido através de um conjunto de elementos visuais ou outros recursos gráficos que podem auxiliar a programação tornando-a mais fácil (SHU, 1988). Por exemplo, muitas linguagens de programação visuais baseiam-se na ideia de "caixas e flechas", onde caixas ou outros objetos da tela são tratados como entidades, ligadas por setas, linhas ou arcos que representam as relações (JOST et al., 2014). Através destas linguagens é possível ensinar

programação de forma intuitiva (DZHENZHER, 2014). A abordagem dessas linguagens é considerada de simples manuseio e de rápido aprendizado (COMPUTACAONAESCOLA, 2016).

Essas linguagens permitem ainda que alunos adquiram conhecimentos mais complexos, tais como programação orientada a objetos e desenvolvimento do pensamento computacional (DZHENZHER, 2014). Também auxiliam na compreensão da linguagem textual e possibilitam o desenvolvimento de um bom estilo de programação (DZHENZHER, 2014).

Dentre uma vasta lista de linguagens de programação com abordagem educacional para crianças e jovens são detalhadas as mais utilizadas em projetos de pesquisa na área educacional.

2.1.2.2 Scratch

Scratch é uma linguagem de programação livre baseada em blocos para crianças (SCRATCH, 2016). Esta linguagem permite a aprendizagem sobre conceitos de programação através da utilização da programação visual para controlar histórias interativas, jogos e animações (SCRATCH, 2016). No Scratch o IDE é a linguagem de programação (SANDOVAL-REYES et al., 2011). A programação no Scratch é feita pela junção de blocos de comando. Estes comandos representam declarações, expressões e estruturas de controle (MALONEY et al., 2010). A figura 18 representa a interface de criação de projetos no Scratch. É possível utilizar o ambiente de programação do Arduino para utilizar as mesmas estruturas da linguagem Scratch, o que facilita o desenvolvimento do projeto de quem está iniciando no mundo da computação (CANTU; SANTOS, 2013) .

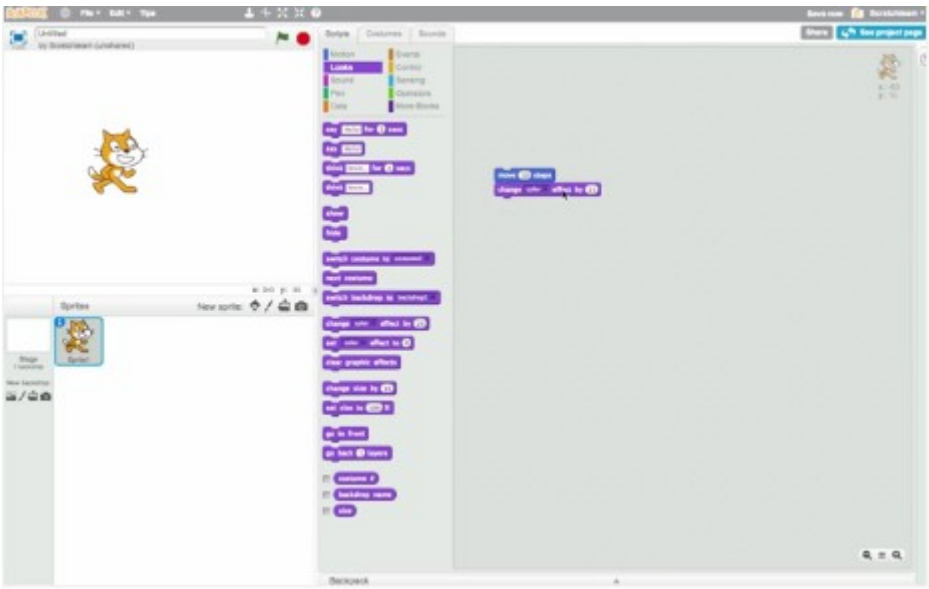


Figura 15 - Ambiente de programação Scratch (Fonte: SCRATCH, 2016)

Para realizar a comunicação entre o Arduino e o Scratch é necessário acrescentar alguns blocos de código específicos, conforme exemplo na figura 16. Através destes blocos é possível receber e enviar dados ao Arduino utilizando a leitura/escrita dos pinos digitais e analógicos do Arduino. Esses blocos são carregados via interface Scratch utilizando uma API do Arduino ou uma interface de aplicação (COMPUTACAONAESCOLA, 2016). Pinos digitais são utilizados para detectarem ou transmitirem níveis lógicos digitais (ARDUINO, 2016). Já os pinos analógicos são usados para leitura de sinais analógicos de sensores conectados ao Arduino (ARDUINO, 2016). Além de assumir valores de entrada e saída, os pinos também podem reconhecer valores de 0 e 1. Os pinos do Arduino usam lógica digital (COMPUTACAONAESCOLA, 2016). Lógica digital é uma forma de representar os sinais binários por níveis de tensão (0 – desligado / 1 – ligado). A figura 17 mostra um exemplo de bloco de código utilizando valores de entrada e saída e também a manipulando os valores de 0 e 1.

Figura 16 - Blocos de comunicação Arduino e Scratch (Fonte: Elaborado pelo(a) autor(a))

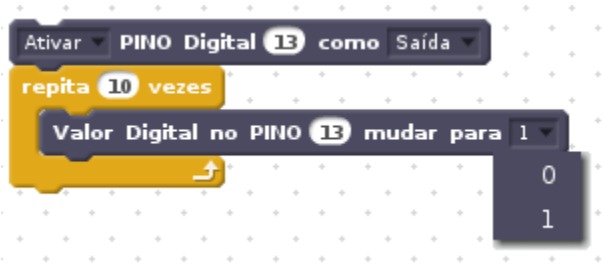


Figura 17 - Bloco de código representando valores de entrada/saída e valores binários (Fonte: COMPUTACAONAESCOLA, 2016).

Além das funcionalidades de entrada e saída de dados também é possível gerar sons através da manipulação da frequência, operar com servomotores para realizar um movimento dada uma angulação e até depurar o código desenvolvido.

2.1.2.4 Snap!

O Snap! (BERKELEY, 2016) anteriormente conhecido como *Build Your Own Blocks* (BYOB) é uma linguagem de programação educacional de código aberto baseada na tecnologia *drag-and-drop* que permite aos usuários criar animações interativas, jogos e muito mais, ao mesmo tempo aprender sobre conceitos matemáticos e computacionais (BERKELEY, 2016). É uma re-implementação estendida do Scratch que lhe permite construir seus próprios blocos (BERKELEY, 2016). Também tem como alvo os alunos iniciantes e mais avançados, porém com o intuito de incluir e expandir recursos do Scratch. O seu funcionamento é feito através de um navegador. Ao contrário do Scratch que foi escrito em Adobe Flash, o Snap! é implementado usando *JavaScript*, permitindo assim que os

desenvolvedores transformem código *JavaScript* arbitrário em blocos gráficos (BERKELEY, 2016). Para facilitar a utilização e ser uma plataforma independente, a IDE do Snap! e o seu interpretador são também totalmente implementados em *JavaScript* onde sua execução é feita direta no navegador sem que haja necessidade de qualquer tipo de processo de instalação (BERKELEY, 2016). A figura 18 apresenta a interface do Snap!.

Para realizar a integração do Snap! com o Arduino, Intel Galileo ou então com outro tipo de *hardware* compatível é necessário acrescentar os mesmos blocos de comunicação utilizados no Scratch (COMPUTACAONAESCOLA, 2016). Através da utilização desses blocos também é possível receber e enviar dados para o *hardware* em questão (COMPUTACAONAESCOLA, 2016).

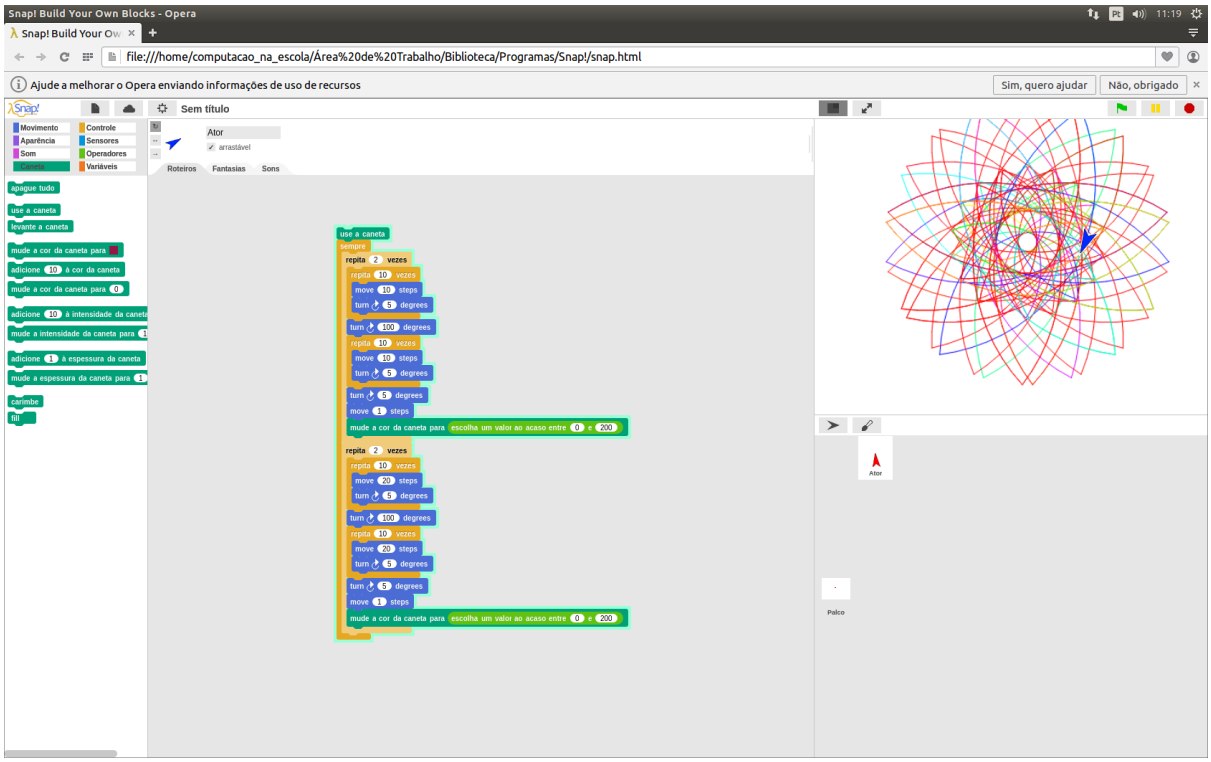


Figura 18 - Ambiente de programação Snap! (Fonte: Elaborado(a) pelo autor(a))

2.1.2.5 Alice

Alice é uma ferramenta de ensino gratuita projetada para ser o primeiro contato de uma criança com a programação orientada a objetos (ALICE, 2016). Através dessa linguagem de programação educativa é possível criar animações para contar histórias, jogar jogos interativos ou até mesmo compartilhar vídeos na web (ALICE, 2016). Contudo, não foi encontrado nenhum tipo de suporte para a computação física utilizando a linguagem de

programação Alice. Na interface interativa da Alice, conforme mostrado na figura 19, as crianças arrastam e largam blocos gráficos para criarem um programa, onde as instruções correspondem a declarações padrões em uma linguagem de programação orientada a objetos (ALICE, 2016). Alice permite às crianças verem imediatamente como seus programas de animação estão sendo construídos, permitindo-lhes compreender facilmente a relação entre as instruções de programação e o comportamento dos objetos em sua animação (ALICE, 2016). Ao manipular os objetos em seu mundo virtual, as crianças ganham experiência com todas os comandos de programação comumente ensinados em um curso de programação introdutório (ALICE, 2016).

Figura 19 - Interface de programação Alice (ALICE, 2016).

2.1.2.6 Blockly

Blockly é um projeto de código livre da Google que serve para a construção de editores de programação visual (GOOGLE, 2016). Ele é executado em um navegador web e se assemelha ao Scratch, conforme mostrado na figura 20. Blockly utiliza blocos que ligados em conjunto tornam o código escrito mais fácil, seu código pode ser transformado em código JavaScript, Python e outros (GOOGLE, 2016). Os desenvolvedores podem integrar o editor

Blockly em suas próprias aplicações web para criar uma grande interface com o usuário para usuários iniciantes (GOOGLE, 2016).

Para realizar a integração do Blockly com o Arduino, existe um editor de programação visual chamado BlocklyDuino (GASOLIN, 2012). O BlocklyDuino possui suporte aos *hardwares* da Intel com blocos específicos do Grove Starter Kit (GASOLIN, 2012). Esses blocos abrangem todos os componentes existentes no kit, que é composto por sensores de temperatura, sensores de luz, atuadores, entre outros (GASOLIN, 2016). O processo de integração do Blockly com o Arduino acaba sendo um pouco mais custoso pois é necessário acessar interface do BlocklyDuino, montar os blocos e em seguida converter este código desenvolvido em blocos para a linguagem interpretada pela interface do Arduino.

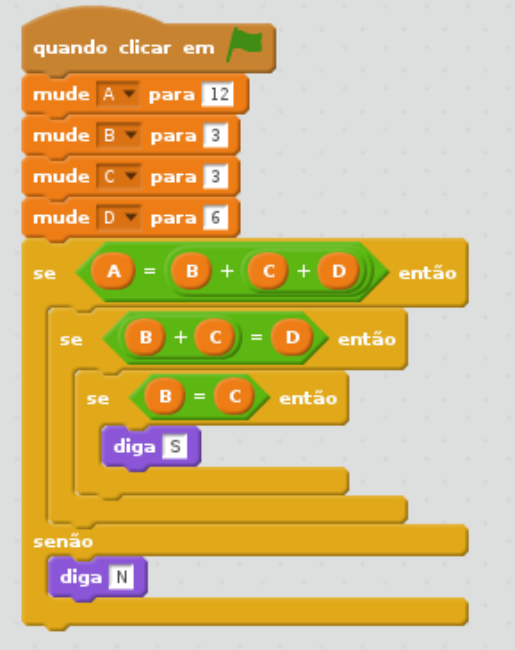
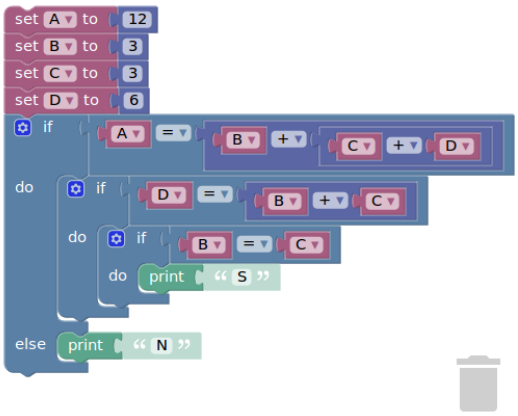
Figura 20 - Ambiente de programação Blockly (Fonte: GOOGLE, 2016)

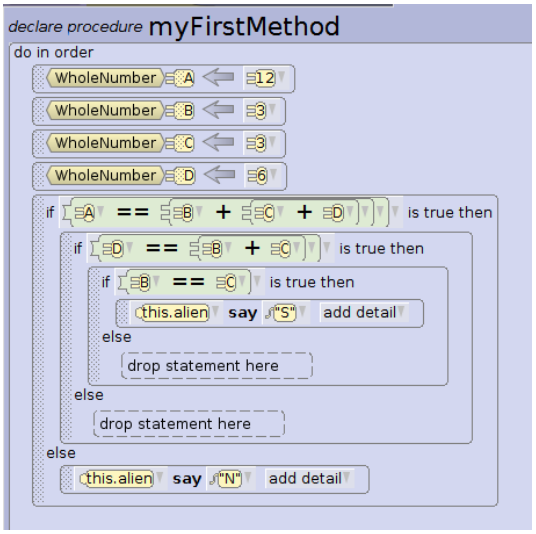
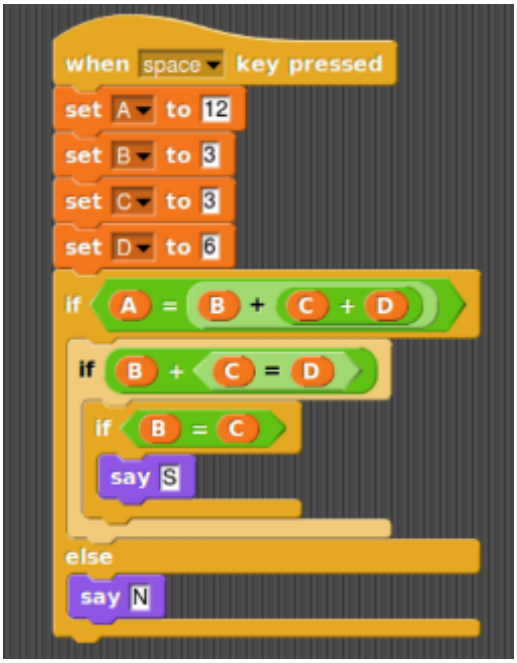
2.1.2.7 Análise das linguagens

Para fins de comparação das linguagens foi feita a implementação de um mesmo algoritmo para todas as linguagens citadas anteriormente. O algoritmo implementado foi retirado da Olimpíada Brasileira de Informática (OBI). A OBI é uma competição baseada em outras olimpíadas científicas brasileiras onde o objetivo é despertar no aluno o interesse por computação, através de atividades que envolvem desafios de lógica (UNICAMP, 2016). O intuito dessa comparação foi definir qual é a linguagem de mais fácil manipulação e entendimento considerando o público-alvo do presente trabalho. O algoritmo escolhido foi retirado de uma tarefa da OBI do ano de 2015 fase 1 - nível júnior, esse nível é voltado para

os alunos do ensino fundamental, público-alvo deste trabalho. A tabela 3 mostra a implementação desse algoritmo em cada uma das linguagens citadas.

Tabela 3 - Implementação do mesmo algoritmo para distintas linguagens de programação de bloco (Fonte: Elaborado(a) pelo autor(a))

Linguagem	Algoritmo	Pontos Positivos	Pontos Negativos
Scratch		Interface amigável. Opção de idioma português. Desenhos de animação interativos. Não necessita instalação para utilizar. Pode se utilizar via navegador web. Abordagem *drag-and-drop*. Possibilidade de integração com computação física.	
Blockly		Exportável para outras linguagens de programação. Não necessita instalação para utilizar. Pode se utilizar via navegador web. Abordagem *drag-and-drop*. Possibilidade de integração com computação física.	Não possui desenhos de animação interativos.

Alice		Desenhos de animação interativos em 3D. Abordagem *drag-and-drop*.	Interface não muito intuitiva. Necessita instalação para utilizar. Sem integração com computação física.
Snap!		Interface amigável. Desenhos de animação interativos. Não necessita instalação para utilizar. Pode se utilizar via navegador web. Abordagem *drag-and-drop*. Implementa conceitos de programação mais complexos. Possibilidade de integração com computação física.	

Dentre as linguagens analisadas foi observada uma grande similaridade entre as sintaxes. Todas elas suportam os comandos necessários para implementação da atividade exemplo da OBI. Porém algumas diferenças entre elas podem ser consideradas, como por exemplo a Alice que não possui uma interface tão visível e de fácil compreensão comparada com as outras. Com exceção do Blocky todas as demais linguagens analisadas possuem interação entre as instruções de programação e o comportamento de objetos de animação. O Snap! por se tratar de uma derivação do Scratch não apresenta basicamente nenhuma

diferença em relação ao Scratch no nível de dificuldade do algoritmo analisado. A diferença entre Scratch e Snap! seria notória somente na implementação de algo um pouco mais complexo e que fosse voltado para alguns conceitos específicos de programação que o Snap! tem suporte.

2.1.2.8 Comunicação entre Arduino e Scratch

Através de um *software* e de uma extensão de *hardware* é possível realizar a comunicação entre o Arduino e o Scratch. Uma das formas de realizar esta comunicação é através de uma extensão conhecida como s2a_fm (YORINKS, 2014). Esta extensão é escrita em Python e permite que o Scratch e o Arduino se comuniquem sem que haja problemas (YORINKS, 2014). O s2a_fm utiliza o protocolo de abstração de *hardware* Firmata para controlar o Arduino (YORINKS, 2014). Caso seja necessário o suporte a som e/ou sensores de ultrassom é necessária a utilização do protocolo Firmata Plus (COMPUTACAONAESCOLA, 2016). O protocolo Firmata serve para se comunicar com o *software* no *host* do computador e será necessário ser carregado pelo menos uma vez no Arduino para o que o mesmo possa se comunicar com o Scratch (ARDUINO, 2016; COMPUTACAONAESCOLA, 2016). Isso permite que seja possível a escrita de um *firmware* personalizado, ou seja, que seja criado instruções operacionais programadas diretamente no *hardware,* sem que haja a necessidade de criar o seu próprio protocolo e objetos para o ambiente de programação que está sendo utilizado (ARDUINO, 2016). Para a interação do s2a_fm também é necessária a instalação do PyMata que é uma biblioteca do Python para comunicação serial com o Firmata no Arduino (YORINKS, 2014). A explicação da comunicação entre Scratch e Arduino utilizando s2a_fm pode ser vista na figura 21.

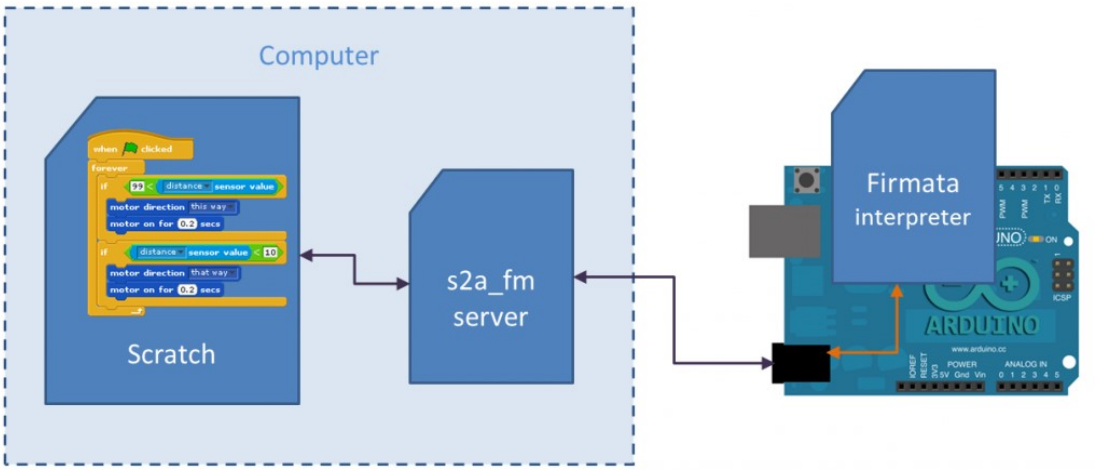


Figura 21 - Funcionamento da comunicação Scratch e Arduino usando s2a_fm (FONTE: COMPUTACAONAESCOLA, 2016)

Para rodar o servidor s2a_fm é necessário saber qual a porta de comunicação serial virtual o computador conectou o Arduino através da conexão USB estabelecida (COMPUTACAONAESCOLA, 2016). Após isso é necessário executar um comando para o servidor de comunicação Scratch/Arduino (COMPUTACAONAESCOLA, 2016). Esse comando reproduz uma mensagem onde é mostrado o status da conexão, por exemplo, total de pinos detectados e status do servidor (COMPUTACAONAESCOLA, 2016). Esta comunicação não se restringe apenas ao Arduino, pode ser realizada também com outros tipos de placas, como por exemplo à Intel Galileo, Intel Edison, dentre outras (ARDUINO, 2016).

Uma outra alternativa de *software* de comunicação é o *Scratchduino* que é um servidor de comunicação multiplataforma que controla robôs e dispositivos de computação física através da utilização de um Arduino (COMPUTACAONAESCOLA, 2016). O *Scratchduino* é adequado para a utilização de crianças por ser um *software* com uma interface gráfica amigável. Ele substitui o tradicional servidor de comunicação s2a_fm, implementando os mesmos protocolos de comunicação utilizados pelo s2a_fm (COMPUTACAONAESCOLA, 2016). A figura 22 ilustra o modo de funcionamento do *Scratchduino*.

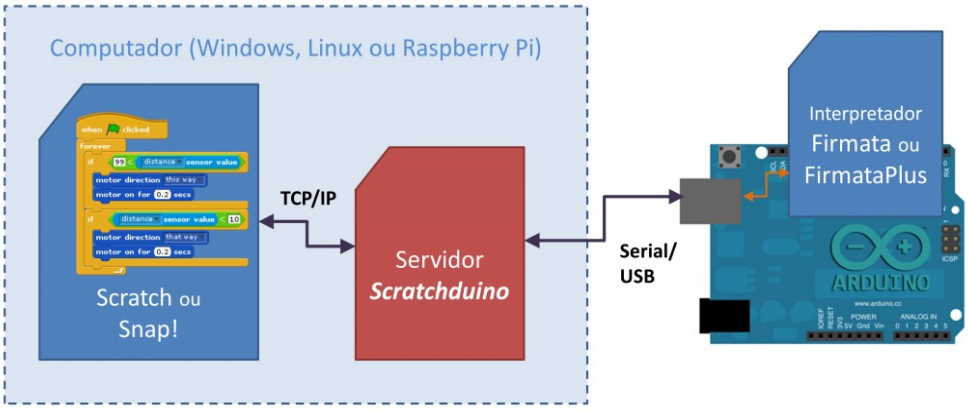


Figura 22 - *Scratchduino* funcionando como servidor de comunicação entre Scratch e Arduino (Fonte: COMPUTACAONAESCOLA, 2016)

Assim como no servidor s2a_fm, para realizar a comunicação entre o *Scratchduino* e o microcontrolador é necessário haver uma comunicação serial sendo feita através de um dispositivo USB (COMPUTACAONAESCOLA, 2016). Para funcionar, o microcontrolador tem de estar conectado ao computador e estar executando um *firmware* interpretador do protocolo firmata (COMPUTACAONAESCOLA, 2016).

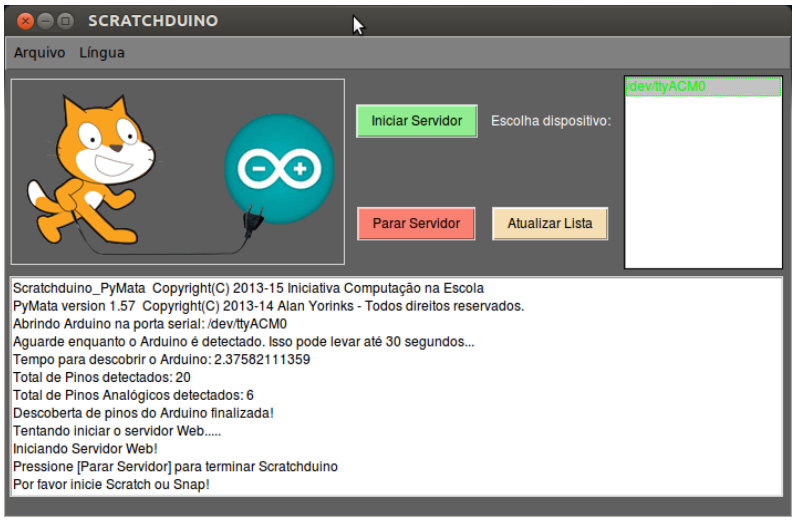


Figura 23 - Interface gráfica de usuário (GUI) de *Scratchduino* (Fonte: COMPUTACAONAESCOLA, 2016)

A interface do *Scratchduino*, mostrado na figura 23, listará todos os dispositivos USB conectados ao computador, basta selecionar o dispositivo desejado e então iniciar o servidor. Após isso, é necessário somente rodar o Scratch para iniciar o desenvolvimento. O *Scratchduino* está disponível nas versões português e inglês (COMPUTACAONAESCOLA, 2016).

2.1.3 Estrutura física

A estrutura física, além do *hardware* e *software,* é também muito importante para a composição da ideia de computação física tendo em vista que ela é fundamental para o desenvolvimento de estruturas elaboradas e para a construção de formatos mais confeccionados. É possível criar estruturas físicas a partir de uma grande variedade de materiais. O material reciclável é um tipo de material que pode ser utilizado para a montagem dessas estruturas físicas, utilizando objetos tais como garrafas *pet*, cano, papelão, dentre outros. Também existe a possibilidade de criar essas estruturas com materiais artesanais, como por exemplo o feltro que é um tecido basicamente feito de lã e pêlos de animais. Nas figuras 24-32 são apresentados alguns projetos e atividades com a utilização destes materiais.

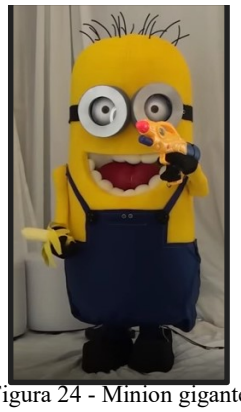


Figura 24 - Minion gigante controlado por Arduino e Scratch (FONTE: COMPUTACAONAESCOLA, 2016)

Figura 25 - Lixeirinho elaborado com material reciclável (FONTE: ROBOTICAIFAL, 2010)

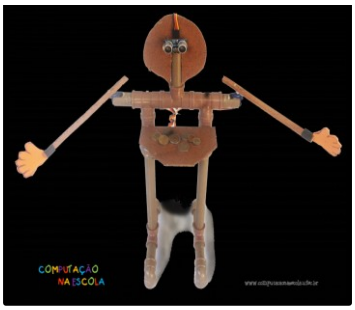


Figura 26 - Robô que protege o seu tesouro elaborado com Arduino (FONTE: COMPUTACAONAESCOLA, 2014)

Figura 27 - Elefante felpudo (FONTE: INSTRUCTABLES, 2012)

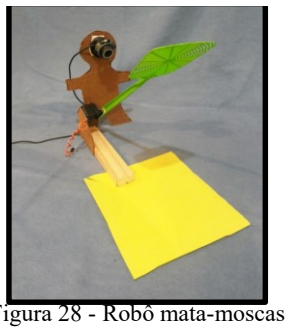


Figura 28 - Robô mata-moscas (FONTE: COMPUTACAONAESCOLA, 2014)

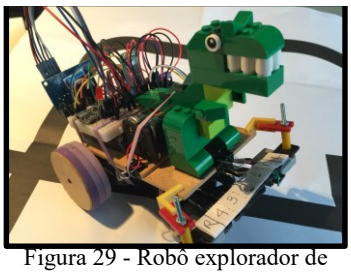


Figura 29 - Robô explorador de labirintos (FONTE: ROVAI, 2016)

Figura 30 - Coruja de feltro feito com arduino lilypad (FONTE: MEDIAMIT, 2016)

Figura 31 - Robô Andarilho de Cabide de Arame (COMPUTACAONAESCOLA, 2014)

Figura 32 - Colar feito com arduino lilypad (FONTE: ALPHAONELABS, 2010)

Além destes materiais utilizados para a criação de estruturas em computação física também é possível utilizar kits educacionais para este tipo de construção tais como Lego e o Atto educacional (ATTOEDUCACIONAL, 2016).

O Atto educacional é formado por um conjunto de peças de fácil utilização e permite a criação e execução de diversos tipos de atividades de maneira simples e ágil (ATTOEDUCACIONAL, 2016). Além disso, o Atto educacional está associado diretamente a educação, oferecendo condições estimulantes e adequadas para auxiliar no aprendizado do interessado (ATTOEDUCACIONAL, 2016). Dentre as áreas de atuação se destaca a robótica que proporciona atividades integradas com a utilização de sensores, atuadores e *softwares* livres (ATTOEDUCACIONAL, 2016). A figura 33 mostra alguma das peças que compõem o kit Atto Educacional.

Figura 33 - Peças Atto (FONTE: ATTOEDUCACIONAL, 2016)

Todos estes materiais comentados anteriormente auxiliam na construção de atividades e projetos. Os materiais recicláveis são acessíveis pois possuem um custo mínimo e são de fácil aquisição. Garrafas *pet*, madeira, cano e papelão são exemplos de materiais que podem ser encontrados em diversos locais, como na própria casa da pessoa. São também materiais de fácil manipulação, podendo ser utilizado métodos como corte, cola e pintura sem que a pessoa tenha muita experiência com trabalho manual. Um cano com cerca de 3 metros de comprimento, por exemplo, pode ser encontrado em uma loja de materiais de construção pelo valor aproximado de R$7,00 (LEROYMERLIN, 2016). O cano pode ser utilizado para construir estruturas mais rígidas ou até simular o braço de um boneco. O feltro, outro material bastante utilizado para enfeitar estruturas físicas, custa em torno de R$12,00 o metro e pode ser encontrado em qualquer loja de tecido (PONTOCHEIO, 2016).

Já os materiais de kits educacionais comentados anteriormente tem um custo um pouco maior dependendo de qual kit a pessoa queira adquirir. Os preços desses kits detalhados não estão disponíveis no site do fabricante por isso não foram incluídos momentaneamente.

Existem diversas possibilidades para a montagem da estrutura física, cada qual com seus benefícios e dificuldades. A existência de kits com peças já prontas tais como Atto e Lego facilita bastante a montagem desta estrutura, porém a utilização de outros recursos e materiais também podem ter suas vantagens. Tudo dependerá da força de vontade e principalmente a imaginação da pessoa para fazer algo criativo.

2.2 Processo de ensino e aprendizagem

2.2.1 Aprendizagem e ensino

A aprendizagem implica nas mudanças de comportamento e competências adquiridas a partir de diferentes experiências (LEFRANÇOIS, 2008, p. 6). O ensino, por sua vez, pode ser definido como a forma, estratégias e métodos envolvidos na transmissão dessas competências (LINO, 2007; CAMARGO, 2013; SANTOS, 2001). Pode-se mostrar como funciona o fluxo do ensino através da figura 34.

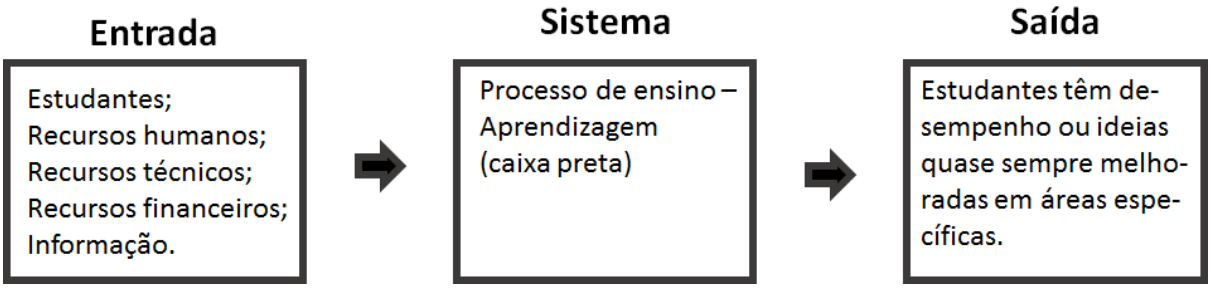


Figura 34 - Fluxo do ensino (FONTE: PERCIVAL; ELLINGTON & RACE, 1993).

Aprendizagem e ensino são processos intimamente ligados entre si (NETTO, 1987). O ensino está relacionado à transmissão de conhecimentos e a aprendizagem à aquisição de conhecimentos de forma sistemática ou institucionalizada (SPRINTHALL, 1993). O ensino pode ocorrer de duas maneiras, presencial ou à distância (XANTHOPOYLOS, 2013). Ensino presencial é o termo utilizado para caracterizar o ensino tradicional, onde o conteúdo é transmitido pelo professor através de aulas expositivas, para seus alunos, num local físico como a sala de aula (MENEZES, 2001). Já o ensino à distância (EAD) é o termo utilizado para caracterizar ensino que ocorre quando o professor e o aluno estão separados em relação

ao tempo e o espaço (PUERTA; AMARAL, 2008). A mediação didático pedagógica nos processos de ensino e aprendizagem ocorre com a utilização de meios e tecnologias de informação e comunicação, com estudantes e professores desenvolvendo atividades educativas em lugares ou tempos diversos. A unidade instrucional desenvolvida no presente trabalho é aplicada conforme o métodos do ensino à distância utilizando atividades extraclasse. Atividades extraclasse apoiam e complementam o processo de ensino e aprendizagem, desenvolvido no ensino presencial, ampliando as possibilidades de aprofundamento de um determinado conteúdo (JEREISSATI, 2012).

O ensino pode ser realizado utilizando unidades instrucionais. Uma unidade instrucional pode ser uma aula, um jogo, uma disciplina, um curso, um evento de aprendizado ou qualquer outra unidade possível (GONÇALVES, 1993). A definição dessas unidades instrucionais, incluindo atividades, estratégias, sistemas de avaliação, métodos e materiais instrucionais é abordado pelo design instrucional (GONÇALVES, 1993).

Design instrucional (DI) ou engenharia pedagógica trata de métodos, técnicas e recursos utilizados em processos de ensino-aprendizagem (PAQUETTE, 2002). O design instrucional corresponde à "ação intencional e sistemática de ensino, que envolve o planejamento, o desenvolvimento e a utilização de métodos, técnicas, atividades, materiais, eventos e produtos educacionais em situações didáticas específicas, a fim de facilitar a aprendizagem humana a partir dos princípios de aprendizagem e instrução conhecidos" (FILATRO, 2004, p. 65). Como processo, o DI é associado ao *Instructional System Design (ISD*) - que compreende uma série de atividades a serem executadas para desenvolver um unidade instrucional. O design instrucional pode ser fixo, aberto ou contextualizado, já que existem diferentes contextos de utilização do mesmo. É necessário ter um modelo de design adequado às diferentes realidades educacionais (FREIRE, 2009). Existem diversos modelos que representam as etapas do design instrucional, tais como Dick, Carey & Carey (DICK; W. CAREY; L. CAREY, 2000), IDOL (SIRAGUSA; DIXON; DIXON, 2007) e ADDIE (BRANCH, 2009). O modelo ADDIE é um dos mais utilizados (MARUIAMA, 2016). Para o desenvolvimento do presente trabalho serão utilizados os modelos ADDIE por ser o mais tradicional e utilizado, e o modelo IDOL por possuir dimensões que tratam do ensino à distância.

2.2.1.1 Addie

Um dos principais modelos de ISD é o modelo ADDIE[5], processo iterativo que envolve ciclos de feedback onde atividades são realizadas simultaneamente conforme pode ser visto no diagrama apresentado na figura 35 (BRANCH, 2009).

Figura 35 - Fases do Modelo ADDIE (Fonte: Elaborada pelo(a) autor(a) à partir de (GRAFINGER, 1988)

Dentro de cada uma destas fases ilustradas na figura 35, existem instruções claramente definidas sobre o que é cada item. Essas instruções são explicadas mais detalhadamente a seguir (BRANCH, 2009):

1. Análise: a fase de análise tem como objetivo analisar os componentes utilizados para orientar o desenvolvimento da unidade instrucional. Deve-se analisar o público-alvo da unidade instrucional, identificando o perfil dos alunos, o nível de conhecimento e seus interesses. Faz-se necessária também a análise do ambiente onde a unidade instrucional vai ser inserida, definindo as possibilidades e limitações oferecidas pela infraestrutura. A partir desta contextualização são definidos os objetivos de aprendizagem a serem alcançados (AHMAD, 2013; ABOUT E-LEARNING).

Os **objetivos de aprendizagem** definem as competências que os alunos devem possuir ao final da execução da unidade instrucional. Os objetivos de aprendizagem podem

5 Em inglês a sigla significa Analysis (Análise), Design (Desenho), Development (Desenvolvimento), Implementation (Implementação) e Evaluation (Avaliação).

ser definidos utilizando-se a Taxonomia de Bloom (BLOOM, 1956). A Taxonomia de Bloom é uma forma de organização hierárquica de objetivos de aprendizagem fazendo a distinção entre os aspectos fundamentais no contexto da educação. Segundo Bloom (1956) a aprendizagem de competências ocorre em três domínios: cognitivo, afetivo e psicomotor. Esses domínios por sua vez são divididos em níveis que possuem uma hierarquia entre eles:

- **Cognitivo:** Abrange a aprendizagem intelectual. As habilidades no domínio cognitivo tratam de conhecimento, compreensão e o pensar sobre um problema ou fato. Esse domínio é dividido em 6 níveis suscetíveis conforme tabela 4.

Tabela 4 - Níveis do domínio cognitivo da Taxonomia de Bloom (BLOOM, 1956)

1. Conhecimento	Memorizar informações, termos, padrões e conceitos aprendidos através de fatos.
2. Compreensão	Traduzir, compreender e interpretar informações com base em conhecimento prévio.
3. Aplicação	Aplicar novos conhecimentos para conseguir completar problemas ou tarefas de novas situações por meio da aplicação de um novo conhecimento.
4. Análise	Examinar, classificar, dividir e relacionar informações identificando causas. Fazer inferência e encontrar provas.
5. Síntese	Integrar e combinar informações de maneira nova.
6. Avaliação	Apresentar sua opinião e avalia informações e ideias.

- **Afetivo:** Abrange aspectos sociais e de valores. Enfatizam o sentimento, emoção ou grau de aceitação ou rejeição. Esse domínio é dividido em 5 níveis suscetíveis conforme tabela 5.

Tabela 5 - Níveis do domínio afetivo da Taxonomia de Bloom (BLOOM, 1956)

1. Receber	Participar com atenção passiva e disposição para assistir a estímulos (aula por exemplo)
2. Responder	Participar ativamente e possuir disponibilidade para reagir de alguma forma, responder, expressar sua vontade (motivação).
3. Valorizar	Atribuir valor para objetivos, fenômenos, informações ou comportamentos. Isso varia com grau de comprometimento do aluno.

4. Organizar	Reunir diferentes valores, começando a construir e organizar um sistema próprio de valores distintos e resolvendo o conflito entre eles
5. Caracterizar	Apresentar um determinado valor ou grupo de valores que resulta em um comportamento consistente, característica pessoal.

- **Psicomotor:** Abrange aspectos de habilidade motoras como manipulação de ferramentas ou objetos. No caso do presente trabalho esse domínio vai ser explorado de forma da comunicação e trabalho em equipe. Esse domínio possui 7 subníveis propostos por Simpson conforme tabela 6 (SIMPSON, 1972).

Tabela 6 - Níveis do domínio psicomotor por Simpson (SIMPSON, 1972)

1. Percepção	A habilidade de usar estímulos sensoriais para guiar atividades motoras.
2. Prontidão	Prontidão para agir, incluindo prontidão mental, física e emocional.
3. Resposta guiada	Fase inicial na aprendizagem de uma habilidade complexa, que inclui imitação, tentativa e erro. A adequação ao desempenho é alcançada por meio da prática.
4. Mecanismo	Fase intermediária na aprendizagem de uma habilidade complexa. Respostas aprendidas se tornam habituais e os movimentos com confiança e proficiência.
5. Resposta complexa	Desempenho habilidoso da realização dos atos que envolvem padrões de movimentos complexos e sem hesitação possuindo assim desempenho automático.
6. Adaptação	As habilidades são bem desenvolvidas e o aluno pode modificar os padrões de movimento para atender as necessidades especiais.
7. Originalidade	Criação de novos padrões de movimentos podendo atender a uma situação ou problema específicos. Os resultados da aprendizagem enfatizam a criatividade com base em habilidades altamente desenvolvidas.

2. Design: a fase de design é a fase em que detalha-se o projeto da unidade instrucional. São definidos os conteúdos, o tempo atribuído para cada um deles e sequência que os conceitos serão transmitidos aos alunos. Escolhem-se os métodos e estratégias de ensino, material didático, recursos e ferramentas utilizados para auxiliar no processo de aprendizagem. A estratégia instrucional é modo em que o conteúdo será tratado e exposto, podendo produzir experiências diferentes de aprendizado (NITZKE, 2013). Os tipos e exemplos de estratégias instrucionais podem ser vistos na tabela 7. É também na fase de

design que são definidas as formas de avaliação da própria unidade. O resultado da fase de design é um plano de ensino que deve contemplar as estratégias instrucionais, objetivos de aprendizagem, e as formas de avaliação de desempenho.

Tabela 7 - Estratégias Instrucionais (KEESEE, 2015)

Estratégias Instrucionais		
Instrução Direto	É o ensino mais comum onde o professor ensina o aluno através de um processo de aprendizagem em um horário específico, conhecido como aula.	Exemplos: aula expositiva, palestra, vídeo, apostila, painel.
Instrução Indireto	É o ensino onde o aluno busca o conhecimento por conta própria e o professor serve como um facilitador que provê o suporte.	Exemplos: descoberta, análise de caso, solução de problemas.
Estudo Independente	É o estudo por conta própria, através de duplas ou pequenos grupos onde não existe a figura do professor.	Exemplos: leitura, preparação de relatórios, exercícios.
Instrução Interativa	É o estudo feito em grupos de discussão.	Exemplos: debates, discussões, laboratório, *brainstorming*.
Aprendizagem Experimental	É o ensino orientado à atividades. A pessoa aprende através de experiências. Entre as técnicas de aprendizagem experimental existem os jogos educativos.	Exemplos: jogos, observação no campo, experiência na prática, estudos de caso.

3. Desenvolvimento: é a fase em que desenvolve-se o que foi definido no plano de ensino elaborado na fase anterior. Durante esta fase é feita a geração de conteúdo para o material didático, preparação de recursos e ferramentas necessárias para implementar as atividades instrucionais. São utilizadas várias ferramentas (papel, caneta, editores de texto, *software* de programação, etc) para o desenvolver os materiais didáticos selecionados, podendo ser slides, folhas de exercícios, rubricas, etc.

4. Implementação: esta fase consiste na aplicação da unidade instrucional desenvolvida na fase anterior. É nesta fase em que o material didático com o conteúdo desenvolvido é disponibilizado de forma acessível aos alunos. Durante esta fase em que o aluno fará uso dos materiais e atividades instrucionais criados. Em atividades extraclasse,

como é o caso do presente trabalho, são utilizados métodos de interação de ensino a distância. Neste caso a fase de implementação consiste no momento em que o aluno realiza as atividades da unidade instrucional e tem contato com o material desenvolvido.

5. Avaliação: esta fase destina-se à avaliação da qualidade das unidades instrucionais e processos. Após a conclusão desta fase deve-se identificar os sucessos e recomendar melhorias, avaliando se os objetivos de aprendizagem foram alcançados e também identificar o que deve ser feito para melhorar a unidade instrucional (EDUCATIONALTECHNOLOGY, 2014). Com isso a unidade instrucional é revisada e refinada.

2.2.1.2 Modelo IDOL

O modelo de design instrucional para aprendizagem *on-line* (IDOL) é uma adaptação de Reeves e Reeves (1997) (SIRAGUSA, 2007). Este modelo é projetado para trabalhar em conjunto a outros modelos de design instrucional (SIRAGUSA, 2007). O modelo IDOL possui 24 dimensões que são compatíveis com as diversas necessidades pedagógicas dos alunos e diversos estilos de cursos e aulas (BAHMAN, 2014). Estes 24 dimensões, conforme mostra a figura 36 são agrupadas em três aspectos principais (análise, estratégia e avaliação) que são utilizados com base na formação de análise educacional, concebida para receber e avaliar os ambientes de aprendizagem eletrônica para otimização da qualidade pedagógica (BAHMAN, 2014).

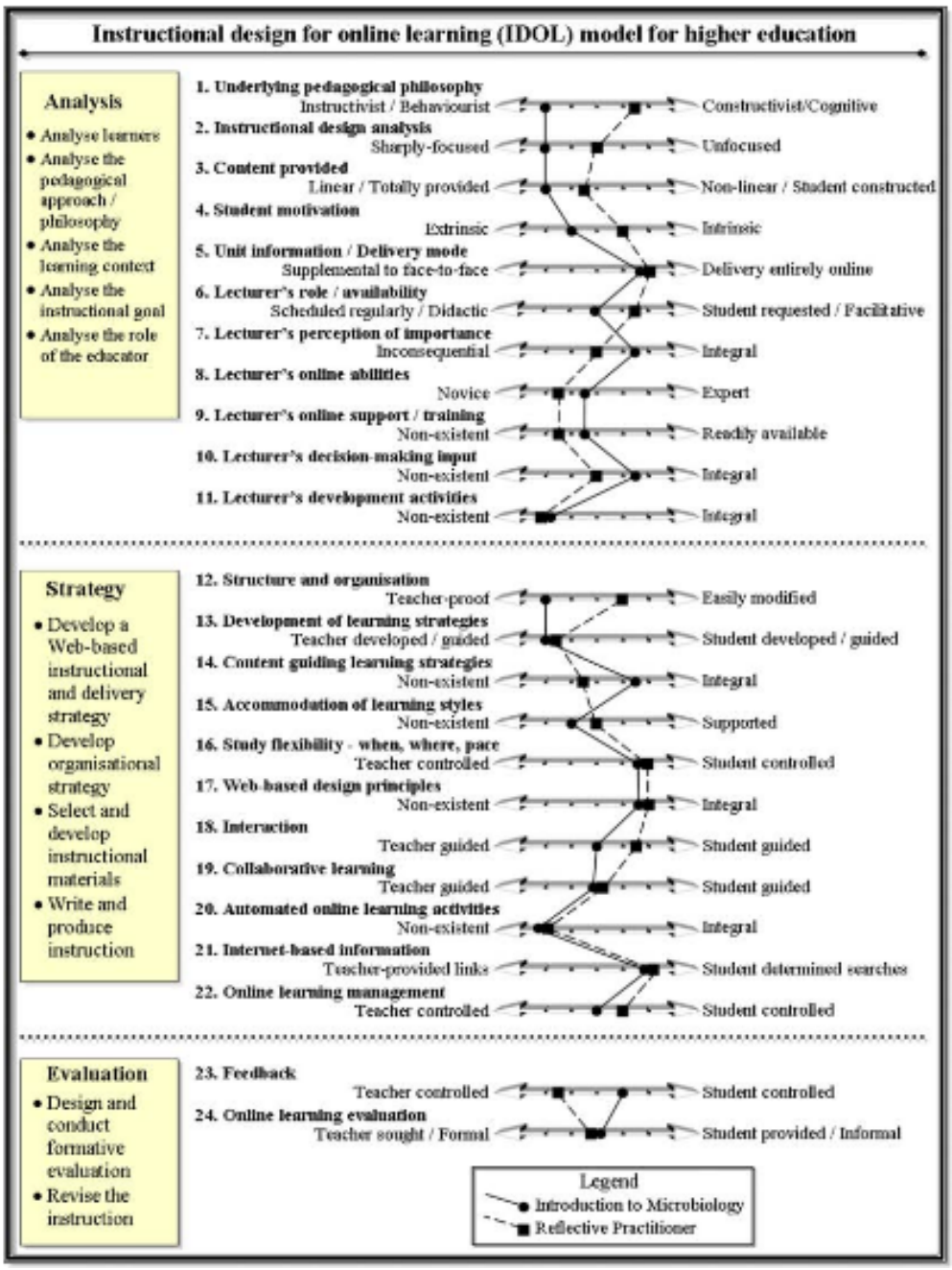


Figura 36 - Design instrucional para ensino à distância - IDOL (Fonte: SIRAGUSA; DIXON; DIXON, 2007)

Análise

O primeiro aspecto educacional é a análise, que tem como objetivo analisar os alunos, a abordagem pedagógica, o contexto de aprendizagem, o objetivo instrucional e o papel do educador dentro de 11 dimensões descritas a seguir.

1. Filosofia Pedagógica Fundamental

A fim de promover o ensino pensando através de ambientes de aprendizagem baseados em tecnologia e estratégias de ensino que promovem os alunos a fazer conexões com novas informações o educador pode definir dois tipos de abordagens. A abordagem construtivista de ensino, onde os alunos são incentivados a construir o seu próprio significado do conteúdo através de suas experiências anteriores ou a abordagem comportamental. Para isso deve-se analisar os requisitos dos alunos, o domínio da aprendizagem e necessidades especiais dos alunos.

2. Análise do Design Instrucional

O desenvolvimento de ambientes de aprendizagem on-line precisa definir sobre o vasto campo de conhecimentos relacionados com modelos de Design Instrucional (por exemplo, Dick, Carey & Carey, 2005; Gagné, Briggs & Wager, 1992) para a análise da instrução, os alunos (fundo, conhecimento prévio, motivação, etc.), o contexto de aprendizagem, o desenvolvimento de uma estratégia de ensino e avaliação. O professor pode desenvolver avaliações formativas e / ou somativa para identificar como melhorar a unidade e para determinar a sua eficácia.

3. Fornecimento de conteúdo

O detalhe e extensão do conteúdo fornecido aos estudantes pode variar dependendo das necessidades pedagógicas dos alunos. Estudantes que estudam totalmente *on-line* devem ter acesso a todo o conteúdo da unidade, incluindo os resultados da aprendizagem, requisitos de atribuição e recursos relevantes. Os alunos que frequentam as aulas presenciais podem receber o conteúdo em sala de aula e conteúdo adicional no site da suplementação de classe.

4. Motivação do Aluno

Alunos que estudam na modalidade a distância precisam sentir são parte de um grupo de alunos e são capazes de obter assistência com as exigências da unidade e dificuldades técnicas. Os designers de aprendizagem online deve usar estratégias de motivação intrínseca ou extrínseca baseado no perfil do aluno.

5. Informações da Unidade / Modo de disponibilidade

Se uma unidade é para ser entregue totalmente on-line deve incluir todas as informações necessárias para a conclusão bem sucedida dos alunos da unidade, incluindo o conteúdo apropriadamente detalhado, atividades de aprendizagem, requisitos de atribuição e

materiais de apoio. Se o material *on-line* é para ser complementar às aulas presenciais professor deve determinar quais informações serão fornecidas on-line e quais as informações que serão distribuídas durante as aulas.

6. Função da Aula / Disponibilidade

Um professor pode ter de assumir um papel didático para orientar a aprendizagem dos alunos. Este professor precisa estar disponível em horários regulares para ajudar os alunos com as atividades de aprendizagem e para clarificar conceitos. Professores devem rotineiramente verificar os meios de comunicação *on-line* para novos lançamentos e fornecer respostas rápidas e adequadas para as perguntas dos alunos.

7. Percepção da Importância da aula

Como professores percebem a importância da aprendizagem *on-line* pode influenciar a forma como a aprendizagem on-line é utilizada e integrada em suas práticas de ensino.

8. Habilidades online do professor

Os conhecimentos e habilidades de tecnologias de aprendizagem *on-line* do professor pode influenciar a forma como eles utilizam os meios para melhorar a aprendizagem dos alunos.

9. Suporte online do professor / Treinamento

Professores com conhecimentos avançados de práticas de desenvolvimento de aprendizagem *on-line* podem ser aplicadas formas mais eficientes de apresentar os mesmos materiais de aprendizagem.

10. Contribuição da aula para tomada de decisões

Professores que mostram interesse nos aspectos de desenvolvimento e tomada de decisão de aprendizagem *on-line* são muitas vezes envolvidos em soluções inovadoras para a aprendizagem on-line dentro da sua área de ensino (McMurray & Dunlop, 1999).

11. Atividades de desenvolvimento da aula

Professores envolvidos no projeto de aprendizagem *on-line* são mais propensos a empregar alguma forma de processo de design instrucional, a fim de analisar e acomodar as necessidades específicas de aprendizagem de seus alunos.

Estratégia

O segundo aspecto educacional é a estratégia, que tem como objetivo desenvolver uma estratégia instrucional de entrega *on-line*, desenvolver a estratégia organizacional, selecionar e desenvolver materiais didáticos e escrever e produzir informações.

12. Estrutura e organização

A estrutura pode ser rígida para que os alunos possam seguir apenas um caminho de aprendizagem linear. As informações podem ser fornecidas com flexibilidade para desenvolver a estrutura, conforme necessário (Siragusa & Dixon, 2005). A estrutura, incluindo navegação, deve ser auto-intuitiva. A flexibilidade pode ser fornecida para o desenvolvimento da estrutura, conforme necessário.

13. Desenvolvimento de estratégias de aprendizagem

Decisões de design instrucional podem influenciar e encorajar diferentes estratégias de aprendizagem que podem ser usados pelos alunos (Bula et al, 1998; Smith e Ragan, 2005). O desenvolvimento de conteúdos para a aprendizagem on-line podem incluir estratégias de aprendizagem específicos para a construção de novos conhecimentos sobre o conhecimento aprendido anteriormente. Os estudantes também podem ser incentivados a partilhar os seus pensamentos sobre o conteúdo e atribuições através de meios de comunicação.

14. Estratégias de aprendizagem para orientação de conteúdo

O conteúdo colocado disponibilizado *on-line* pode ajudar com guiando particulares estratégias de aprendizagem para promover a compreensão profunda de um determinado assunto (Miller & Miller, 2000). Para facilitar essas estratégias, o conteúdo pode incluir exemplos da vida real, materiais de enriquecimento e *links* para sites relevantes.

15. Concepção de estilos de aprendizagem

Professores envolvidos no desenvolvimento da aprendizagem *on-line* precisam considerar como a concepção de materiais *on-line* podem acomodar estilos de aprendizagem dos alunos e facilitar a abordagem profunda à aprendizagem através do envolvimento ativo dos materiais *on-line* (Weigel, 2002).

16. Estudo da flexibilidade - quando, onde, ritmo

O projeto de um ambiente de aprendizagem *on-line* pode facilitar se os alunos são capazes de estudar quando, onde, em que sequência e em que ritmo que escolherem. Essa escolha pode ser feita pelo professor, definindo previamente, ou pelo aluno conforme for seu interesse.

17. Princípios de design baseados na *web*

Ao desenvolver um ambiente de aprendizagem *on-line*, tais como *web design* adequado para o público-alvo, alguns detalhes precisam ser definidos, incluindo a navegação auto-intuitiva, *layouts* de páginas, o uso de texto, cores de fundo e texturas, a compatibilidade com várias configurações de computador (Lynch; Horton, 2002).

18. Interação

Estudantes e professores precisam estar familiarizados com os recursos de comunicação disponíveis. Facilidades de comunicação *on-line*, como bate-papo *on-line* e *whiteboard on-line* também podem ser utilizados. Sessões de bate-papo *on-line* podem ser programadas para a compartilhamento de ideias.

19. Aprendizado colaborativo

A aprendizagem centrada no aluno deve ser encorajada através de estratégias tais como a aprendizagem cooperativa (Ralph, 1998). Estudantes que estudam em locais remotos podem ser incentivados a comunicar uns com os outros conforme necessário.

20. Atividades automatizadas de aprendizagem *on-line*

Atividades automatizadas de aprendizagem *on-line* devem apoiar a prática e *feedback* do aluno proporcionando condições ideais para receber e processar informações. As atividades podem começar com uma introdução suportado com gráficos e outros meios de comunicação do conceito ou problema a ser examinado, uma demonstração de como o problema pode ser resolvido, seguido por uma atividade que permite ao aluno para tentar um problema semelhante.

21. Informações baseadas na internet

Incentivar os alunos a procurar informações específicas com base na Internet para promover uma compreensão mais profunda do assunto. Estudantes e professores podem

postar URLs úteis para sites relevantes que têm encontrado para outros estudantes para acessar.

22. Gestão de aprendizagem online

Um professor pode utilizar um recurso de progresso que permite que os alunos tenham acesso ao seu crescimento durante toda a duração da unidade. Os estudantes podem ser encorajados a disponibilizar os seus trabalhos *on-line* para os outros alunos a avaliarem.

Avaliação

23. Feedback

Os professores podem ajudar os alunos com a sua aprendizagem através da prestação de apoio e *feedback* adequado aos estudantes durante seus estudos *on-line* para melhorar a sua aprendizagem. *Feedback* pode enriquecer as experiências de aprendizagem *on-line*, a quantidade e o tipo de *feedback* que os estudantes exigem irá variar dependendo necessidade dos alunos e nível de envolvimento com os materiais de aprendizagem.

24. Avaliação da aprendizagem online

O ciclo de desenvolvimento contínuo de um ambiente de *e-learning*, como acontece com todos os outros ambientes de aprendizagem, deve incluir um processo de avaliação para determinar e manter a eficácia do sistema. O contínuo desenvolvimento de ambientes de aprendizagem *on-line* pode se beneficiar de comentários de avaliação dos alunos sobre as suas experiências com cursos anteriores.

2.2.2 Ensino de computação

Ciência da computação é uma disciplina definida como o estudo de computadores e processos algorítmicos, incluindo os seus princípios, hardware e design de software, suas aplicações e seu impacto na sociedade (CSTA, 2011). As diretrizes do currículo CSTA K-12 dizem que computação e tecnologia, independentemente do campo de estudo de um jovem, são os assuntos que mais abrem portas no século XXI (CSTA, 2011). A aprendizagem de computação envolve 5 áreas essenciais para os níveis de aprendizagem, conforme Figura 37.

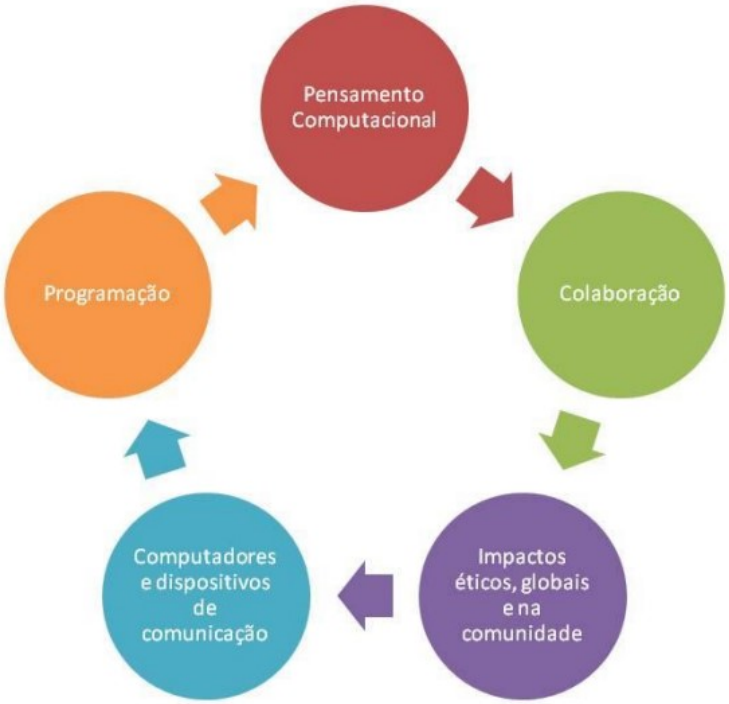


Figura 37 - Áreas das diretrizes para ciência da computação (CSTA, 2011).

Pensamento computacional: é uma abordagem para resolver problemas de uma forma que pode ser implementada por um computador. O que faz com que os alunos não se tornem apenas utilizadores de ferramentas, mas construtores de ferramentas. Assim os alunos podem utilizar um conceitos, tais como abstração, recursão e iteração, para processar e analisar dados e criar artefatos reais e virtuais. O pensamento computacional é um método de resolução de problemas de diversas áreas, que pode ser transferido e aplicado entre os alunos.

Colaboração: a ciência da computação é uma disciplina intrinsecamente colaborativa. Raramente grandes progressos são realizados por uma pessoa que trabalha sozinha. Os projetos de computação envolvem grandes equipes de profissionais de computação que trabalham em conjunto para projetar código, testar, depurar, descrever e manter o software ao longo do tempo. Portanto é de grande importância desenvolver habilidades como trabalhar em equipe, comunicar-se de forma eficaz e criticar construtivamente.

Programação/Prática da computação: trata-se de uma parte essencial da computação, desenvolvendo a competência e habilidade de criar programas de software. A prática da computação a nível K-12 deve, portanto, incluir a capacidade de explorar o uso de

programação na resolução de problemas, escolher formatos apropriados de arquivos e banco de dados para um problema computacional particular, e utilizar a aplicação adequada Interfaces de Programas, ferramentas de software e bibliotecas para ajudar a resolver problemas algorítmicos e computacionais. Os alunos devem compreender algoritmos e a sua aplicação prática.

Computadores e dispositivos de comunicação: Os alunos em todos os níveis devem compreender os elementos de dispositivos e redes modernas de informática e comunicação. Os alunos também devem usar a terminologia apropriada e precisa ao se comunicar sobre tecnologias. Sendo assim devem entender as facilidades da internet como comunicação global e como praticar boa cidadania.

Impactos éticos globais e na comunidade: O uso ético de computadores e redes é um aspecto fundamental da informática e deve ser visto como um elemento essencial da aprendizagem e prática. Os alunos devem ter conhecimento dos princípios de privacidade pessoal, segurança de rede, licenças de software e direitos autorais, a fim de que estejam para se tornarem cidadãos responsáveis no mundo moderno. Os alunos também devem ser capazes de avaliar a confiabilidade e a precisão das informações que recebem a partir da Internet. Os alunos devem estar preparados para avaliar os diversos impactos positivos e negativos de computadores na sociedade e identificar até que ponto os problemas de acesso impactam nossas vidas.

O ensino de computação do infantil ao ensino médio é baseado em um modelo de três níveis, conforme figura 38. O nível 1 fornece os padrões de aprendizagem para os estudantes do ensino infantil até ao sexto ano. O nível 2 fornece os padrões de aprendizagem para os estudantes do sexto até o nono ano. O nível 3 fornece os padrões de aprendizagem para os alunos do ensino médio (CSTA, 2011).

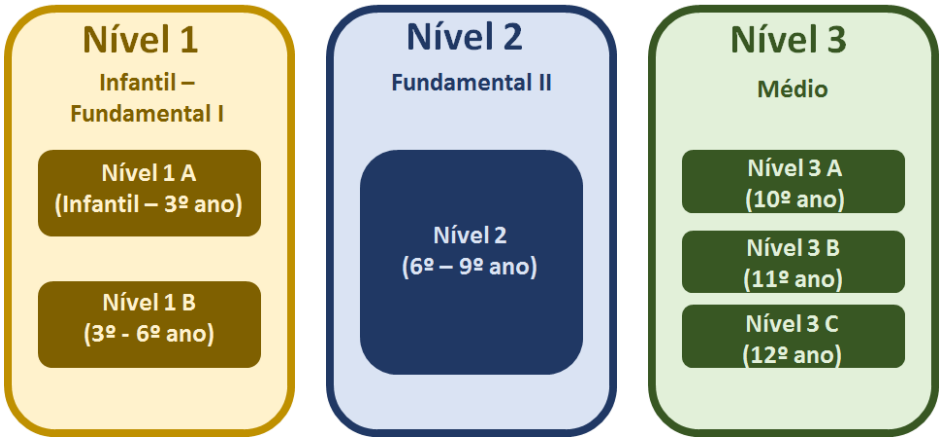


Figura 38 - Níveis de ensino de computação (Fonte: Elaborada pelo(a) autor(a) à partir de CSTA, 2011).

Este trabalho é voltado ao nível 2 das diretrizes CSTA, ensino de computação para crianças de 8 a 14 anos. Neste nível os alunos devem começar a usar o pensamento computacional como uma ferramenta de resolução de problemas e resolução de questões relevantes, e não apenas para eles, mas para o mundo em torno deles. As experiências de aprendizagem criadas a partir dessas normas deve ser relevante para os alunos e promover a sua percepção de si mesmos como solucionadores pró-ativos de problemas. A tabela 8 apresenta os objetivos de aprendizagem nível 2 (CSTA, 2011):

Tabela 8 - Objetivos para o nível 2 (CSTA, 2011)

Objetivos de aprendizagem nível 2 (CSTA, 2011)	
Pensamento computacional	[PC1] Usar os passos básicos de algoritmos para a resolução de problemas ao projetar soluções [PC2] Definir o processo de paralelização na forma que se refere à resolução de problemas. [PC3] Definir um algoritmo, como sendo uma sequência de instruções que podem ser processadas por um computador. [PC4] Avaliar formas em que algoritmos diferentes podem ser utilizados para resolver o mesmo problema. [PC5] Pesquisar algoritmos de busca e ordenação. [PC6] Descrever e analisar uma sequência de instruções a ser seguida (p.ex., descrever o comportamento de um personagem em um videogame, dirigido por regras e algoritmos). [PC7] Representar dados em maneiras variadas, incluindo texto, sons, imagens e números. [PC8] Usar representações visuais de estados de problema, estruturas, e dados (p.ex., gráficos, tabelas, diagramas de rede, fluxogramas). [PC9] Interagir com modelos específicos de conteúdo e simulações para apoiar a aprendizagem e pesquisa. [PC10] Avaliar que tipos de problemas podem ser resolvidos usando modelagem e simulação. [PC11] Analisar o grau em que um modelo computacional representa o mundo real. [PC12] Fazer uso da abstração para decompor um problema em subproblemas. [PC13] Compreender a noção de hierarquia e abstração em computação, incluindo linguagens de alto-nível, conjunto de instruções e circuitos lógicos. [PC14] Examinar conexões entre elementos matemáticos e ciência da computação, incluindo números binários, lógica, conjuntos e funções. [PC15] Fornecer exemplos de aplicações interdisciplinares do pensamento computacional.
Colaboração	[C1] Aplicar ferramentas e periféricos de produtividade para colaboração em grupo e para apoiar a aprendizagem ao longo do currículo. [C2] Colaborativamente projetar, desenvolver, publicar e apresentar produtos, utilizando recursos tecnológicos que demonstram e comunicam conceitos do currículo. [C3] Colaborar com colegas, especialistas e outros utilizando práticas colaborativas como programação em pares, trabalho em equipes e participação em atividades de aprendizagem ativa em grupo. [C4] Exibir disposições necessárias para colaboração: fornecer feedback útil, compreender e aceitar múltiplas perspectivas, socialização.
Programação/Prática da computação	[P1] Selecionar ferramentas e recursos tecnológicos apropriados para realizar tarefas variadas e resolver problemas. [P2] Usar uma variedade de ferramentas e periféricos de multimídia para apoiar a produtividade e aprendizagem pessoal ao longo de todo o currículo. [P3] Conceber, desenvolver, publicar e apresentar produtos (p.ex., páginas web, aplicações móveis, animações) usando recursos de tecnologia que demonstram e comunicam os conceitos do currículo. [P4] Demonstrar uma compreensão de algoritmos e a sua aplicação prática. [P5] Implementar soluções de problema utilizando uma linguagem de programação, incluindo: o comportamento de laços (sequências de instruções que se repetem), instruções condicionais, lógica, expressões, variáveis e funções. [P6] Demonstrar boas práticas na segurança da informação pessoal, usando senhas, encriptação e transações seguras. [P7] Identificar carreiras interdisciplinares que são abrangidas pela ciência da computação. [P8] Demonstrar disposição para resolver e programar problemas indeterminados. [P9] Coletar e analisar dados emitidos da saída de múltiplas execuções de um programa de computador.

Computadores e dispositivos de comunicação	[CDC1] Reconhecer que os computadores são dispositivos que executam programas. [CDC2] Identificar uma variedade de dispositivos eletrônicos que contêm processadores computacionais. [CDC3] Demonstrar compreensão sobre a relação entre hardware e software. [CDC 4] Usar terminologias adequadas e precisas ao desenvolvimento na comunicação sobre tecnologia. [CDC5] Aplicar estratégias para identificar e resolver problemas de rotina de hardware que ocorrem no uso de computadores diariamente. [CDC6] Descrever os principais componentes e funções de sistemas de computadores e redes. [CDC7] Descrever o que distingue os seres humanos de máquinas, dando um enfoque na inteligência humana contra a inteligência de máquina e, formas que podemos nos comunicar. [CDC8] Descrever maneiras em que os computadores usam modelos de comportamento inteligente (e.ex., movimento de robô, fala e compreensão da linguagem e, visão computacional).
Impactos éticos, globais e na comunidade	[IEG1] Exibir comportamentos legais e éticos no uso de informação e tecnologia e, discutir as consequências do uso indevido. [IEG2] Demonstrar conhecimento das mudanças nas tecnologias de informação ao longo do tempo e os efeitos destas mudanças na educação, no local de trabalho e na sociedade. [IEG3] Analisar os impactos positivos e negativos da computação na cultura humana. [IEG4] Avaliar a precisão, relevância, adequação, abrangência, e viés de fontes de informação eletrônicas referentes a problemas do mundo real. [IEG5] Avaliar a precisão, relevância, adequação, abrangência, e viés de fontes de informação eletrônicas referentes a problemas do mundo real. [IEG6] Discutir como a distribuição desigual de recursos de computação em uma economia global levanta questões de equidade, acesso e poder.

3. LEVANTAMENTO DE UNIDADES INSTRUCIONAIS PARA COMPUTAÇÃO FÍSICA

Este capítulo abrange uma visão geral do estado da arte para unidades instrucionais para o ensino de computação física. Para isto, foi realizada uma revisão sistemática da literatura (KITCHENHAM, 2004).

3.1 Definição da revisão

Esta revisão tem como objetivo identificar e avaliar trabalhos e pesquisas existentes através de critérios de qualificação bem definidos em relação ao tema do presente trabalho (KITCHENHAM, 2004). Para este trabalho, busca-se responder a seguinte pergunta de pesquisa através da revisão: **Existem unidades instrucionais para ensinar computação física para crianças e adolescentes extraclasse?**

Por intermédio desta pergunta, faz-se os seguintes questionamentos de pesquisa em relação a unidades existentes:

- Quais são os objetivos de aprendizagem?
- Quais conteúdos são ministrados nestas unidades instrucionais?
- Qual a estratégia utilizada para o ensino?
- Quais tipos de atividades extraclasse são utilizadas?
- Qual o tipo de material didático utilizado?

- Quais são os kits existentes que auxiliam no ensino da computação física?
- Quais tecnologias de *hardware* são utilizadas?
- Quais linguagens de programação são utilizadas para ensinar conceitos de lógica para crianças e adolescentes?
- Quais tipos de estrutura física estão disponíveis para o auxílio da construção de um projeto para computação física?
- Qual o custo relacionado às atividades realizadas para as atividades?
- Como é feita a avaliação?
- Em quais locais foram aplicados? Em média quantas pessoas participaram?

Fontes de dados: As ferramentas de busca utilizadas para responder aos questionamentos de pesquisa são o Google Scholar[6] para consulta de artigos e livros, o Google[7] para abordar soluções mais comerciais e o portal Scratched[8] para relatos e propostas de educadores falando sobre suas experiências em relação a utilização da ferramenta como material de ensino.

Critérios de inclusão e exclusão: Para responder os questionamentos de pesquisa, os seguintes itens devem ser definidos para consideração dos trabalhos disponíveis:

- Deve ter como objetivo de aprendizagem conceitos de computação física, sendo necessário incluir conceitos de *hardware*, *software* e programação;
- Devem ser atividades realizadas principalmente com o intuito de ensino à distância. Porém por se tratar de um conceito muito recente e não muito comum para este tipo de atividade, os presenciais não foram excluídos;
- Deve focar em alunos com a idade entre 8 e 14 anos. Porém se houver uma leve variação na faixa etária, os mesmos não foram excluídos;
- Foram excluídos os trabalhos que tratam de pessoas com algum tipo de deficiência pois não focam diretamente em nossos objetivos de análise;
- As iniciativas que focarem somente em alunos do jardim de infância também não foram considerados.

6 http://scholar.google.com.br/
7 http://google.com.br
8 http://scratched.gse.harvard.edu

Critérios de qualidade: Pesquisas e estudos que não possuírem informações fundamentais sobre a forma em que foram realizados, ou seja, aqueles que apenas apresentarem informações superficiais e não completas sobre o assunto, não são relevantes para a definição da revisão.

Termos de busca: Foram definidos os termos de busca com base nos critérios de inclusão. Na tabela 9 estão os termos com seus respectivos sinônimos em português e suas traduções para inglês.

Tabela 9 - Sinônimos e traduções dos termos de busca

Termos	Sinônimos	Tradução (Inglês)
computação	ciência da computação	*computer science, computing*
programação	-	*programming, code*
computação física	robótica	*physical computing, robotics*
crianças	-	*children*
ensino	educação, aprendizagem, ensino à distância	*teaching, learning, e-learning*
unidade instrucional	-	*instructional unit, teaching unit, workshop, course, class, school*

3.2 Execução da busca

A execução da busca foi realizada em junho de 2016 utilizando os termos de busca em inglês apresentados na tabela 10. Não foi estipulado um intervalo de data para a busca dos dados pois se trata de um assunto não muito comum. A análise foi feita em três iterações e foram analisados diversos tipos de informações sobre o ensino de computação.

Tabela 10 - Termos utilizados na execução da busca

Iteração	Termos de Busca
1	(“physical computing” OR “computer science” OR “robotics” OR “computer science” OR “computing") AND (“programming” OR “computer programming”) AND (“children”) AND (“teaching” OR “learning” OR “instructional unit” OR “e-learning”)
2	(“physical computing” OR “computer science” OR “robotics” OR “computer science” OR

	“computing") AND (“programming” OR “computer programming”) AND (“children”) AND (“teaching” OR “learning” OR “instructional unit”)
3	(“physical computing” OR “computer science” OR “robotics” OR “computer science” OR “computing") AND (“programming” OR “computer programming” OR “code”) AND (“children”) AND (“teaching” OR “learning” OR “instructional unit” OR “e-learning” OR “workshop” OR “course” OR “class” OR “school”)

Primeira iteração: A primeira iteração da busca foi feita com a utilização de todos os termos de busca em inglês apresentados na tabela 10, exceto com os acrescentados na terceira iteração. Todas as bases que retornaram resultados tiveram a análise dos 100 primeiros resultados. Pela ferramenta Google Scholar a busca retornou cerca de 165 mil resultados e pela ferramenta SCRATCHED não foi encontrado nenhum resultado. Após a análise dos resultados, foi possível verificar que a grande maioria dos resultados encontrados não atendia a todos os critérios de inclusão definidos.

Na ferramenta Google foram retornados diversos resultados que não atendiam muito aos critérios definidos. Porém foi encontrado um resultado que aplica o ensino da computação à distância, “Thinking Tap Robotics” (THINKINGTAP, 2016).

Segunda iteração: A segunda iteração foi feita pois não se encontraram resultados relevantes para os questionamentos de pesquisa. Por esse motivo, o termo “*e-learning*” foi retirado da busca. No geral, as fontes de busca apresentaram alguns resultados relevantes porém a grande maioria não foi considerado. Uma parte dos resultados obtidos pela ferramenta Google Scholar era composta de uma abordagem mais voltada ao ensino da computação somente através do *software*. Estes trabalhos não foram incluídos na revisão pois não atendem aos critérios de inclusão.

No portal SCRATCHED foram encontrados poucos relatos porém todos da iniciativa Computação na Escola e por este motivo não foram incluídos na extração. Já através da ferramenta Google foram encontradas aproximadamente 24.000 resultados e foram analisados os 100 primeiros. A grande maioria desses 100 resultados analisados eram relevantes, dentre eles pode-se citar o “Indian Startup Avishkaar Box Launches Robots to Teach Programming to Children” (SINGH, 2015).

Terceira iteração: Na terceira iteração foram acrescentados os termos “*workshop”, “course”, “class”, “school”, “code”* por ainda não se ter obtido um número considerável para a extração. O portal SCRATCHED não retornou nenhum resultado novo, somente os já tinham sido encontrados anteriormente.

Analisando os 100 primeiros resultados da ferramenta Google Scholar foram identificados praticamente os mesmos resultados da busca anterior, porém ainda sim pode-se considerar um resultado como relevante, “Introductory Programming Workshop for Children Using Robotics” (BALTES; ANDERSON, 2016). Já na ferramenta Google foram encontrados novos resultados, “Webinar – Programming robots in Scratch” (ROBBO, 2016) e “PROGRAMITRA” (PROGRAMITRA, 2016).

3.3 Extração de informação

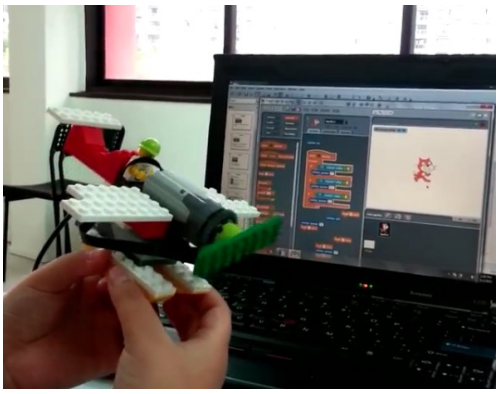
Esta seção apresenta a extração da informação a partir dos resultados encontrados com base nas buscas realizadas e descritas na seção 3.2. A tabela 11 mostra os itens a serem analisados referentes aos resultados encontrados.

Tabela 11 - Descrição detalhada da unidade instrucional

Contexto	
ID	Número de identificação da unidade
Nome	Unidade instrucional
Foto	Ilustração da unidade
Descrição	Descrição detalhada do unidade
Referência	Local de onde foram extraídas as informações
Unidade instrucional	
Objetivos de aprendizagem	Quais objetivos são definidos para cada unidade
Conteúdo	Quais conteúdos são ministrados nas unidades
Estratégia de ensino	Qual a estratégia de ensino aplicada
Idade participantes	Faixa etária dos participantes
Tipo de atividade	Tipos de atividades executada extraclasse
Material didático	Qual o material didático utilizado para ministrar as atividades
Custo aplicado	O custo necessário para realizar e/ou participar da unidade
Locais aplicados	Em que localidades foram aplicadas as atividades instrucionais
Presencial ou à distância?	Pode ser realizada presencial ou não?
Estrutura do kit	

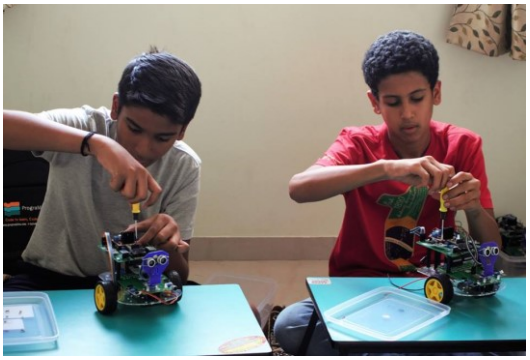
Tecnologia de hardware	Que tipo de tecnologia de *hardware* é utilizado
Linguagem de programação	Que linguagem de programação foi utilizada pelos alunos para as atividades de programação
Estrutura física	Quais tipos de materiais foram utilizados para construir a estrutura física
Avaliação	
Objetivo da avaliação	Qual principal objetivo da avaliação: Avaliar a motivação dos alunos antes e depois da aplicação sobre a computação? Avaliar a eficácia da unidade instrucional?
Pontos fortes	Pontos fortes da unidade levantados pelos autores deste presente trabalho
Pontos fracos	Pontos fracos da unidade levantadas pelos autores deste presente trabalho
Número de participantes	Qual o números de participantes da atividades, tanto para o número de crianças quanto para o número de monitores ajudantes

Tabela 12 - Extração de informações “Thinking Tap Robotics”

Contexto	
ID	1
Nome	Thinking Tap Robotics
Foto	
Descrição	A iniciativa oferece um ensino de robótica para auxiliar crianças a estimular o seu processo de pensamento, habilidades de programação, matemática, engenharia e informática.
Referência	http://sg.thinkingtap.com.sg/
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Explorar o potencial interno de cada criança.

	O2. Criar uma experiência de aprendizagem agradável. O3. Desenvolver habilidades sociais através da colaboração e comunicação social.
Conteúdo	O conteúdo ministrado é dividido em níveis, onde em cada nível a criança aprende sobre conceitos tanto de programação quanto de robótica. **Foundation - Level 2** Construir e programar modelos robótica simples, com diferentes sensores. **Progressive - Level 3** Introduzir uma metodologia de programação modular e conceito de máquinas simples. **Intermediate - Level 4** Introduzir conceito de programação (programação comportamental) **Advanced - Level 5** Aprender programação avançada (aplicação de algoritmo e lógica)
Estratégia de ensino	São aplicadas diversas atividades que possuem níveis de dificuldades que vão desde aprender conceitos básicos de programação até realizar competições entre alunos para eles desenvolverem suas próprias soluções
Idade participantes	7-11 anos
Tipo de atividade	Exercícios online
Material didático	São disponibilizados 6 lições online no site da iniciativa com 2 horas de duração para cada atividade.
Custo aplicado	Os valores variam de R$1.000,00 - R$1.300,00 dependendo da atividade
Locais aplicados	Singapura
Presencial ou à distância?	À distância
Estrutura do kit	
Tecnologia de hardware	Lego Mindstorms EV3
Linguagem de programação	Scratch e linguagem de programação visual nativa do Lego
Estrutura física	Peças de Lego convencionais e Makey Makey
Avaliação	
Objetivo da avaliação	Não informado
Pontos fortes	- Possuem atividades extraclasse com foco em diversos nives educacionais - Realizam atividades em diversos locais em seu país
Pontos fracos	- O valor para participação das atividades juntamente da aquisição do kit - Não descrevem objetivos de avaliação
Número de participantes	Não informado

Tabela 13 - Extração de informações “Programitra”

ID	2
Nome	Programitra
Foto	
Descrição	Instituto de educação que visa desenvolver a resolução de problemas e habilidades do pensamento lógico em escolas através de programação de computadores.
Referência	http://www.programitra.com/
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Ajudar crianças a desenvolver e melhorar suas habilidades de resolução de problemas e raciocínio lógico. O2. Aprender e praticar programação de computadores.
Conteúdo	Introdução à programação de computadores, noções básicas de lógica, a importância da sequência e precisão das instruções por meio de atividades de grupo.
Estratégia de ensino	É um *workshop* onde as crianças aprendem sobre o conteúdo ministrado durante 5 dias, 2 horas por dia.
Idade participantes	8-13 anos
Tipo de atividade	Não informado
Material didático	Não informado
Custo aplicado	R$250,00
Locais aplicados	Índia
Presencial ou à distância?	Presencial
Estrutura do kit	
Tecnologia de hardware	Arduino

Linguagem de programação	Blockly, Scratch
Estrutura física	Não informado
Avaliação	
Objetivo da avaliação	Não informado
Pontos fortes	- Possuem diversos tipos de atividades com diversas tecnologias utilizadas - Os materiais utilizados são de baixo custo
Pontos fracos	- Não são especificados os tipos de atividades e nem os materiais didáticos realizadas nesta unidade - Não descrevem objetivos de avaliação
Número de participantes	Não informado

Tabela 14 - Extração de informações “Introductory Programming Workshop for Children Using Robotics”

ID	3
Nome	Introductory Programming Workshop for Children Using Robotics
Foto	
Descrição	O artigo dá uma visão geral de técnicas para que através de um workshop seja possível introduzir as crianças conceitos de programação usando a robótica. A tarefa em torno do qual o *workshop* foi estruturado foi a construção e programação de um robô que é capaz de encontrar uma linha no meio ambiente e, em seguida, acompanhar esta linha.
Referência	https://www.researchgate.net/profile/Jacky_Baltes/publication/228748799_Introductory_programming_workshop_for_children_using_robotics/links/00b7d5142864ef2c1a000000.pdf
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Combinar ideias para usar a robótica para ensinar às crianças os conceitos elementares envolvidas na programação. O2. Prover um ensino que seria tanto esclarecedor quanto agradável.
Conteú	As atividades foram divididas em três segmentos de conteúdo.

do	**Construção e "Olá mundo!"** - Realizar um processo de compilação e carregamento de um programa - Construção de um kit de robótica e entendimento do mesmo através de manipulação com sensores **Sequência: Curvas e linhas retas** - Conceito de sequência - Declaração de variáveis **Condicionais, iterações e Variáveis** - Aprendizado de conceitos como: condições, laços e utilização de variáveis - Funcionamento de sensores
Estratégia de ensino	É um *workshop* onde as crianças foram colocadas em grupos de dois, e um adulto foi atribuído a cada grupo como um mentor. As tarefas foram divididas em etapas e a partir da passagem de cada etapa foram sendo introduzidos os conceitos necessários.
Idade participantes	8-16 anos
Tipo de atividade	Exercícios
Material didático	Não informado
Custo aplicado	Não informado
Locais aplicados	Nova Zelândia
Presencial ou à distância	Presencial
Estrutura do kit	
Tecnologia de hardware	Lego Mindstorms RCX
Linguagem de programação	Not Quite C (NQC)
Estrutura física	Peças de Lego convencionais
Avaliação	

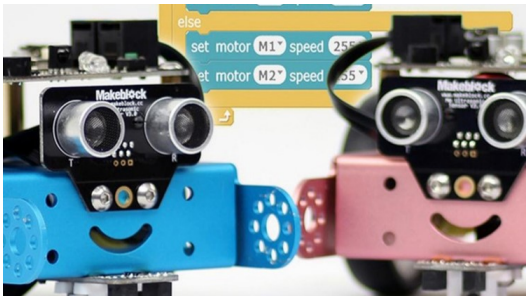
Objetivo da avaliação	Avaliar o interesse dos alunos em relação ao ensino sobre robótica e programação sem que haja uma sobrecarga do assunto sobre a criança
Pontos fortes	- O conteúdo de programação ensinado é bem abrangente
Pontos fracos	- Não foi informado o custo nem o material utilizado para este *workshop* - A linguagem de programação utilizada não é muito didática em relação às existentes
Número de participantes	Não informado

Tabela 15 - Extração de informações "Indian Startup Avishkaar Box Launches Robots to Teach Programming to Children"

ID	4
Nome	Indian Startup Avishkaar Box Launches Robots to Teach Programming to Children
Foto	
Descrição	Iniciativa para incentivar crianças a aprenderem programação através de robôs interativos, Robby e Bonny, desenvolvidos exclusivamente para isso. Os robôs ensinam programação e construção lógica interagindo e brincando com as crianças. Está disponível para utilização através do aplicativo de *smartphones*, o Robo B, e também pode ser utilizado do computador pessoal através do *software* Robo G utilizando blocos de programação.
Referência	http://www.iamwire.com/2015/03/indian-startup-avishkaar-box-launches-robots-teach-programming-children/111491 http://www.avishkaarbox.com/robby-and-bonny#faq
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Ensinar programação e construção de lógica para crianças. O2. Introduzir as crianças no mundo da ciência, tecnologia e programação.

Conteúdo	O conteúdo ministrado é dividido em níveis, onde em cada nível a criança aprende sobre conceitos tanto de programação quanto de robótica. **Programação em tempo real:** (faixa etária 5-8) crianças podem escrever vários tipos de programas por um simples arrastar e soltar de blocos. Desenvolvimento de programas mais básicos. **Programação avançada:** (faixa etária 8-10) estes robôs podem ser programados através de um software de programação Robo G. Este software segue o básico de qualquer linguagem de programação estruturada e recomenda-se que a criança tenha realizado o primeiro nível acima mencionado. Desenvolvimento de programas com *loops* e estruturas *if-then-else*. **Programação expansível:** (faixa etária 10-12) com o tempo, as crianças podem aumentar ainda mais peças de hardware adicionais compatíveis como LED, telas de LCD e motores. Desenvolvimento de programas mais focados na parte de eletrônica.
Estratégia de ensino	São aplicadas diversas atividades que possuem níveis de dificuldades que vão desde aprender conceitos básicos de programação até realizar competições entre alunos para eles desenvolverem suas próprias soluções
Idade participantes	5-12 anos
Tipo de atividade	Atividades realizadas no aplicativo, como desenvolver seu próprio código de programação.
Material didático	Não informado.
Custo aplicado	R$100,00
Locais aplicados	Países asiáticos.
Presencial ou à distância?	Presencial
Estrutura do kit	
Tecnologia de hardware	Não informado.
Linguagem de programação	Não informado. Similar com Scratch.
Estrutura física	Robôs de material MDF.
Avaliação	
Objetivo da avaliação	Não informado
Pontos fortes	- É uma iniciativa com um custo bastante acessível - Possuem atividades para faixas etárias bem distintas - Utilizam um tipo de estrutura bem distinta de outras iniciativas
Pontos fracos	- Não especificam a tecnologia de *hardware* utilizada - Não descrevem objetivos de avaliação
Número de participantes	Não informado

Tabela 16 - Extração de informações “Code Academy 1 2 3”

ID	5
Nome	Code Academy 1 2 3
Foto	
Descrição	Um projeto que abrange vários programas para ensinar programação para crianças. Um desses programas envolve robótica e Scratch.
Referência	http://codeacademy123-cn.com/content/our-programs
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Aprender a tecnologia e ter um impacto favorável sobre o seu futuro. O.2 Promover a criatividade, raciocínio e resolução de problemas habilidades. O3. Aprender conceitos de matemática e ciência. O4. Aumentar o vocabulário em inglês da criança e habilidades de comunicação. O5. Divertir-se de aprendizagem codificação com jogos e robótica e compartilhar seus próprios jogos com os amigos.
Conteúdo	O conteúdo abrange os conceitos principais e programação como expressões booleanas, condições, *loops*, variáveis, eventos, jogos.
Estratégia de ensino	São aulas expositivas que ocorrem em acampamentos de verão para aprender a programar.
Idade participantes	7-14 anos
Tipo de atividade	Programação de robôs Makeblock com Scratch.
Material didático	Não informado
Custo aplicado	Não Informado.
Locais aplicados	Shangai
Presencial ou à distância?	Presencial
Estrutura do kit	
Tecnologia de hardware	Makeblock
Linguagem de programação	Scratch

Estrutura física	Makeblock
Avaliação	
Objetivo da avaliação	Não informado
Pontos fortes	- Possuem diversas atividades incentivando a programação para crianças - Incentivam os pais a criarem o interesse na programação para seus filhos
Pontos fracos	- Não informam o custo e nem o material didático utilizado - Não descrevem objetivos de avaliação
Número de participantes	Não informado

Tabela 17 -“IVNTR”

ID	6
Nome	IVNTR
Foto	
Descrição	Empresa que ministra aulas para crianças com intuito de ensinar programação e robótica.
Referência	http://www.invntr.net/
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Aprender a trabalhar em equipe, comunicando-se e colaborando. O2. Aprender a resolver problemas com sua própria criatividade. O3. Aprender sobre ciência, engenharia e robótica. O4. Promover a criatividade e pensamento crítico
Conteúdo	Os conteúdos são divididos conforme as classes que são divididas por faixa etária. **Máquinas simples (idades 6-9):** crianças fazem máquinas simples usando LEGO. Eles aprendem a ser criativos e resolver problemas de

	construção de máquinas que incorporam as engrenagens, polias, correias e motores. **Robótica introdutória (idades 7 – 10):** as crianças que já passaram pela fase anterior, etapa onde pode se adicionar elementos robóticos para seu modelo, usando LEGO WeDo. Seus modelos ganham sentidos e movimentos através de sensores e motores. **Computação Criativa usando Scratch (idades 9-13):** crianças criam animações, histórias e jogos utilizando Scratch. Assim as crianças estão se divertindo, sendo criativas e ao mesmo tempo aprendendo conceitos de computação importantes como *loops*, If/then e paralelismo.
Estratégia de ensino	Aulas expositivas.
Idade participantes	6-13 anos
Tipo de atividade	Atividades de montagem da estrutura e desenvolvimento de programas lógicos.
Material didático	Não informado
Custo aplicado	Em torno de R$850,00 a mensalidade.
Locais aplicados	Nova Iorque
Presencial ou à distância?	Presencial
Estrutura do kit	
Tecnologia de hardware	LEGO Mindstorm EV3, Júnior FIRST LEGO League, LEGO WeDo
Linguagem de programação	Scratch
Estrutura física	LEGO Mindstorm EV3, Júnior FIRST LEGO League, LEGO WeDo
Avaliação	
Objetivo da avaliação	Não informado
Pontos fortes	- Possuem atividades para diversos tipos de faixas etárias
Pontos fracos	- O valor para participar não é muito acessível - Não descrevem objetivos de avaliação
Número de participantes	Não informado

Tabela 18 -“ROBBO”

ID	7
Nome	ROBBO

Foto	
Descrição	São ferramentas educacionais para aprender conceitos básicos de tecnologia, engenharia e programação.
Referência	http://robbo.world/
Unidade instrucional	
Objetivos de aprendizagem	**Objetivos Gerais** O1. Incentivar as crianças a desenvolver habilidades de resolução de problemas e tornar a aprendizagem de programação uma experiência emocionante.
Conteúdo	Conceitos de tecnologia, programação e robótica.
Estratégia de ensino	Não informado
Idade participantes	Não informado
Tipo de atividade	Atividades de desenvolvimento de programas computacionais para robôs.
Material didático	Não informado
Custo aplicado	ROBBO Kit + Lab R$1.400,00
Locais aplicados	Finlândia
Presencial ou à distância?	Presencial
Estrutura do kit	
Tecnologia de hardware	Arduino UNO
Linguagem de programação	Scratch
Estrutura física	Não informado
Avaliação	
Objetivo da avaliação	Não informado
Pontos fortes	- O *design* do material utilizado foi muito bem elaborado

Pontos fracos	- Pouca informação sobre atividades e estratégias de ensino - Possuem um alto custo de aquisição do kit - Não descrevem objetivos de avaliação
Número de participantes	Não informado

Além das iniciativas e unidades instrucionais citadas anteriormente, também vale ressaltar outros resultados encontrados na busca. Por não atenderem a todas informações necessárias para realizar a extração, foram extraídos somente algumas informações relevantes. Uma dessas iniciativas é o "*Root Robot"* que é um pequeno robô hexagonal que utiliza um ambiente de programação chamado "Square" (FUTURITE, 2016; ROOTROBOT, 2016). Este ambiente de programação é acessado através do uso de um *iPad* (FUTURITE, 2016; ROOTROBOT, 2016). O *Root Robot* é para iniciantes que querem obter um conhecimento maior com as maravilhas da programação (FUTURITE, 2016; ROOTROBOT, 2016). Este robô pode ser adquirido pelo valor aproximado de R$700,00 (ROOTROBOT, 2016). Na figura 39 podemos ver o robô sendo controlado por um dispositivo móvel.

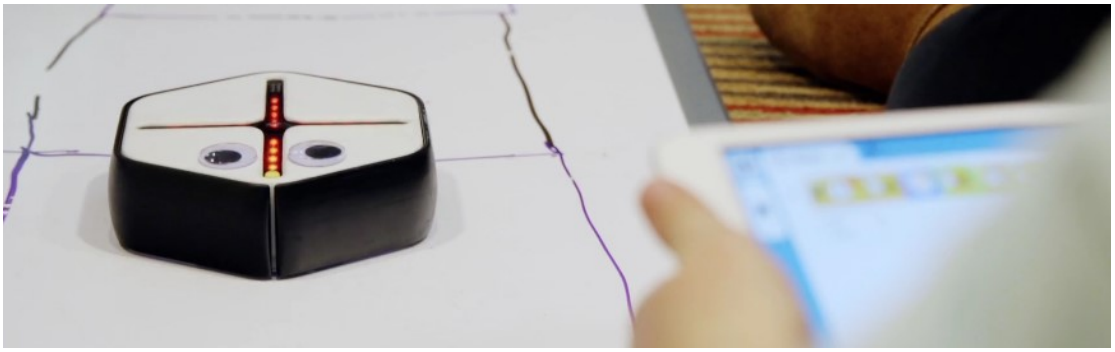


Figura 39 - Root Robot (ROOTROBOT, 2016)

Uma outra iniciativa que pode ser comentada é a "RoboCamp" que é uma plataforma online que contém conteúdos para realizar aulas de robótica com Lego WeDo ou Lego Mindstorms (ROBOCAMP, 2016). Além disso, também existem aulas de ensino de robótica e programação para crianças utilizando *Scratch* (ROBOCAMP, 2016). São disponibilizados materiais de aula que incluem um plano de ensino, uma introdução para exercícios, um passo a passo com instruções para a construção de um robô e instruções de programação detalhada, além de jogos e outros benefícios (ROBOCAMP, 2016). Os preços de utilização desta ferramenta online variam de acordo com o pacote escolhido que vão desde materiais gratuitos, pacotes mensais até pacotes anuais ou até mesmo lições customizadas (ROBOCAMP, 2016). O pacote anual, por exemplo, possui um valor de R$1.200,00 e poderá

ser usufruído nesta data (ROBOCAMP, 2016). Além disso, este pacote também engloba atividades para 3 professores com 12 lições (ROBOCAMP, 2016). A figura 40 mostra uma criança realizando algumas tarefas com o Lego Wedo.

Figura 40 - Criança realizando atividades com Lego Wedo (FONTE: ROBOCAMP, 2016)

Ainda sim existem diversas outras iniciativas que poderiam ser citadas porém nem todas possuem informações suficientes para se extrair informações. Com a extração realizada pode-se ter uma noção de como está o desenvolvimento desses tipos de atividades e as tecnologias que estão sendo usadas.

3.4 Discussão

Percebeu-se que são poucos ou praticamente inexistentes os trabalhos voltados ao ensino de computação física através de atividades extraclasse. Foi encontrado somente a iniciativa representada pelo ID 1 (THINKINGTAP, 2016) que tinha como foco o ensino à distância. No entanto, não se falava nada sobre o custo e sobre os objetivos de avaliação desta iniciativa, o que acabou prejudicando a informação sobre a efetividade da mesma. Infelizmente, essa falta de informação sobre os objetivos de avaliação também ocorreu para praticamente todos os outros casos extraídos.

Notou-se também que grande parte das iniciativas voltadas ao ensino da programação através da robótica varia pouco na utilização de materiais alternativos para a construção da estrutura física. A iniciativa representada pelo ID 4 (SINGH, 2015) apresentou um material um pouco diferente das outras. Foi utilizado para a construção da estrutura física um material conhecido como *Medium-Density Fiberboard* (MDF) que é um material derivado da madeira que é frequentemente usado como componente para partes que requerem montagem. (BAYLOR, 2015). Outra iniciativa que se diferenciou na montagem da estrutura física, foi a

iniciativa representada pelo ID 5 (CODEACADEMY123, 2016) que utilizou o Makeblock para construção do robô. O Makeblock é uma plataforma aberta de construção de robô, onde as partes mecânicas são feitas de alumínio, podendo trabalhar com outras peças do mesmo material perfeitamente (MAKEBLOCK, 2016). Para o restante das iniciativas encontradas, utilizou-se peças tradicionais do Lego.

Foi observado também que o custo das iniciativas varia bastante, podendo ser encontradas iniciativas tanto com preço acessível como com preço um pouco elevado. A iniciativa representado pelo ID 2 (PROGRAMITRA, 2016), apresentou um custo bem reduzido. Por utilizar uma tecnologia de baixo custo como o Arduino, pode ser adquirido mais facilmente. Outra iniciativa que possui um custo ainda menor e acessível é a representada pelo ID 4, já comentada anteriormente. O restante das iniciativas analisadas por utilizarem como principal recurso o Lego, acabam se tornando menos acessíveis por possuírem um alto custo de aquisição.

3.4.1 Ameaças à validade

Existem alguns fatores que podem ameaçar a validade da revisão sistemática da literatura. Essas ameaças podem ser tanto na possibilidade de não ter sido encontrado alguma unidade instrucional extraclasse quanto alguma imprecisão no momento da extração da informação. Com o intuito de minimizar o risco de não ser encontrado algum resultado relevante, foram utilizadas ferramentas de busca bastante abrangentes. Além da utilização destas ferramentas buscou-se a aplicação de palavras chaves mais próximas ao contexto do presente trabalho juntamente com seus sinônimos e traduções para a língua inglesa. Dessa maneira assume-se a redução do risco de não ter-se obtido algum resultado importante voltado ao foco da pesquisa. Em relação ao procedimento de extração da informação, houve o cuidado na análise da mesma tentando mostrar de forma mais correta possível as informações fornecidas.

4. PROPOSTA DA UNIDADE INSTRUCIONAL

Este capítulo consiste na apresentação da unidade instrucional e na descrição dos métodos adotados para o desenvolvimento da mesma. É também apresentado o plano de ensino e o sequenciamento de atividades para a unidade instrucional.

Os objetivos de aprendizagem são avaliados através de uma unidade instrucional para ensinar computação física de uma forma extraclasse. Uma unidade instrucional foi aplicada em um aluno do Ensino Fundamental e além disso foi feita a avaliação por parte de um

professor que ministra o assunto de computação física para alunos também do Ensino Fundamental. Após esta unidade, o aluno deverá ser capaz de aplicar conceitos básicos de programação e computação física, além de exercer o pensamento computacional.

4.1 Requisitos da unidade instrucional

O objetivo principal da unidade instrucional é auxiliar no ensino de conceitos básicos de computação física e programação mantendo o interesse em computação de crianças que já tiveram algum contato com programação e/ou computação física. Conforme o design instrucional analisa-se o público-alvo e o ambiente em que a unidade é inserida e em seguida definem-se os objetivos de aprendizagem.

Análise do público-alvo: O público alvo do presente trabalho abrange alunos na faixa etária dos 8 aos 14 anos que na sua maioria já possui um conhecimento prévio sobre as tecnologias e ferramentas utilizadas. Espera-se que a maioria dos alunos já tenham um conhecimento básico na utilização de computadores devido ao uso do mesmo para trabalhos escolares, utilização de redes sociais e acesso à internet, conforme mostrado na figura 41 através de dados extraídos do IBGE. É importante salientar também que é esperado que o aluno já possua algum conhecimento prévio de programação e/ou já tenha participado de algum curso ou oficina de computação física.

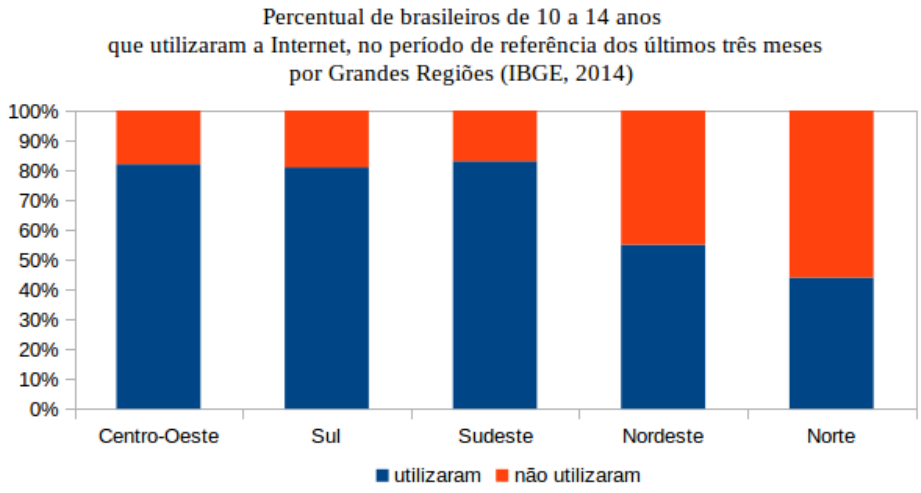


Figura 41 - Percentual de brasileiros de 10 a 14 anos que utilizaram a Internet, no período de referência dos últimos três meses, por Grandes Regiões - 2014 (IBGE, 2014).

Em torno de 81% das crianças e adolescentes brasileiros que possuem acesso à rede têm como hábito comum utilizar a internet todos os dias (SETIC, 2014). Além disso, o conhecimento adquirido pelas crianças também são oriundas de outros tipos de tecnologias, tais como *vídeo-games* e dispositivos móveis, como *smartphones* e *tablets* (SETIC, 2014). A figura 42 mostra a utilização destes diferentes tipos de tecnologias no ano 2014 por crianças e adolescentes.

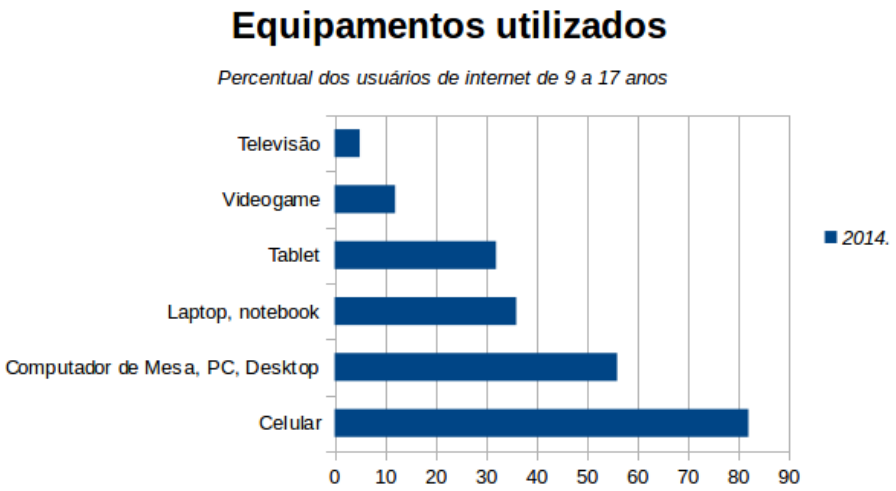


Figura 42 - Percentual de equipamentos utilizados por crianças e adolescentes em 2014
(FONTE: Elaborada pelo(a) autor(a) à partir de SETIC, 2014)

No que se refere aos interesses pessoais dos alunos sabe-se que existe elevada presença de jogos na vida das crianças, seja no videogame ou jogos na Internet (DIARIOCATARINENSE, 2012). A maior diferença entre os gêneros está no tema das brincadeiras, a preferência feminina é por jogos de culinária, memória, vestir e trocar roupas, maquiagem e manicure, fazendinha e bichinhos. Já a preferência masculina é diversões *on-line* com temas ligados a corridas, lutas, ação e aventura, futebol, tiro e perseguição, terror e nave espacial (DIARIOCATARINENSE, 2012). Tipicamente faz parte da rotina das crianças dessa faixa etária assistir programas televisivos, filmes e desenhos animados (INNOVARE, 2015). Em meados do ano 2016 foi lançado o jogo Pokemon GO que virou uma febre mundial. No Brasil não foi diferente, atraindo crianças e jovens o jogo se espalhou por todos os cantos do Brasil tirando muitos caçadores de pokémons de suas casas (PEREIRA, 2016).

Análise do ambiente: A unidade instrucional desenvolvida é projetada para ser realizadas à distância. Para isso é necessário que a unidade seja realizada num ambiente que disponha de

um computador com acesso à internet. Cerca de 49,5% da população brasileira tem computador com acesso à internet na sua própria casa, segundo indica pesquisa divulgada pelo IBGE no ano de 2013 (GRANDA, 2014). É necessário ter um recurso financeiro mínimo de R$156,00 para iniciar a montagem da estrutura inicial da unidade que é composta o kit básico (Protoboard, Arduino, sensor de ultrassom, servomotor, *LED*, resistor e jumpers), conforme capítulo 2.1.1. É importante salientar o baixo custo desta unidade visto que os preços de outros kits de computação física existentes no mercado são mais elevados, como por exemplo, o kit da Intel e o Lego Mindstorm que custam R$700,00 e de R1.200,00 a R$3.000,0 respectivamente (ARDUINO, 2016; AMAZON; MERCADOLIVRE, 2016). A estrutura física é feita utilizando materiais simples e recicláveis como folha A4, palitos de churrasco, fita adesiva e isopor.

A unidade poderá ser realizada com ou sem o auxílio dos pais. O intuito é que a atividade seja inserida num momento de lazer e entretenimento das crianças, para que elas possam absorver o conteúdo enquanto se divertem, sem um padrão formal de ensino. Tendo uma duração estimada de 4 a 5 horas.

Objetivos de aprendizagem: O objetivo principal da unidade instrucional é retomar conceitos básicos da computação física e programação de acordo com as diretrizes do currículo de referência da ACM CSTA K-12 os conteúdos de programação definidos pela OBI. Transmitir ao aluno uma abordagem de resolução de problemas que pode ser desenvolvida num computador envolvendo conceitos de programação tais como estruturas de dados, chamadas de funções, orientação a objetos, recursão e iteração juntamente do conhecimento de computação física, sabendo diferenciar no contexto da unidade instrucional a utilização de hardware e software. Os objetivos de aprendizagem específicos foram definidos conforme os domínios da Taxonomia de Bloom discutida na seção 2.2.1.1. As aulas conceituais e as práticas de programação estão dentro do **domínio cognitivo**, o tema utilizado na atividade final envolve o **domínio afetivo** e a montagem da estrutura física foca no **domínio psicomotor.** Os objetivos específicos estão descritos na tabela 19:

Tabela 19 - Objetivos de aprendizagem

	Objetivos de aprendizagem específicos	**Relação com o currículo de referência (CSTA, 2011)**	**Referência com os domínios da Taxonomia de Bloom**

Pensamento computacional	[O1] Aprender os conceitos básicos de linguagens de programação. [O2] Usar abstração para decompor um problema em subproblemas.	[PC1] Usar os passos básicos de algoritmos para a resolução de problemas ao projetar soluções [PC2] Definir o processo de paralelização na forma que se refere à resolução de problemas. [PC3] Definir um algoritmo, como sendo uma sequência de instruções que podem ser processadas por um computador. [PC10] Avaliar que tipos de problemas podem ser resolvidos usando modelagem e simulação. [PC12] Fazer uso da abstração para decompor um problema em subproblemas.	Domínio Cognitivo
Programação / Prática da computação	[O3] Conseguir criar/programar um algoritmo usando o ambiente para solucionar problemas. [O4] Compreender o funcionamento das atividades desenvolvidas.	[P4] Demonstrar uma compreensão de algoritmos e a sua aplicação prática. [P5] Implementar soluções de problema utilizando uma linguagem de programação, incluindo: o comportamento de laços (sequências de instruções que se repetem), instruções condicionais, lógica, expressões, variáveis e funções. [P8] Demonstrar disposição para resolver e programar problemas indeterminados. [P9] Coletar e analisar dados emitidos da saída de múltiplas execuções de um programa de computador.	Domínio Cognitivo
Computadores e dispositivos de comunicação:	[O5] Compreender os conceitos de hardware e software e saber identificá-los nas atividades desenvolvidas.	[CDC3] Demonstrar compreensão sobre a relação entre hardware e software.	Domínio Cognitivo e Psicomotor

4.2 Proposta de modelo de design instrucional

Conforme discutido na seção 2.2.1 os modelos de design instrucional, ADDIE e IDOL, para o desenvolvimento desta unidade instrucional serão seguidos ambos os modelos se complementando. Desenvolveu-se um modelo híbrido que engloba todas as fases do modelo ADDIE e algumas dimensões do IDOL considerando os objetivos da unidade instrucional apresentados na seção anterior.

Do modelo IDOL foram adotadas as seguintes dimensões:

Análise:

1. Filosofia Pedagógica Fundamental
2. Análise do Design Instrucional
3. Fornecimento de conteúdo
4. Motivação do Aluno
5. Informações da Unidade / Modo de disponibilidade
10. Contribuição da aula para tomada de decisões
11. Atividades de desenvolvimento da aula

Estratégia:

12. Estrutura e organização
13. Desenvolvimento de estratégias de aprendizagem
14. Estratégias de aprendizagem para orientação de conteúdo
15. Concepção de estilos de aprendizagem
16. Estudo da flexibilidade - quando, onde, ritmo

Avaliação:

24. Avaliação da aprendizagem online

As fases do modelo ADDIE foram seguidas na ordem proposta. As dimensões do modelo IDOL citadas acima complementam as fases do ADDIE dando diretrizes para o ensino a distância. Juntamente das fase de análise e design do modelo ADDIE foram utilizadas as dimensões da fase de análise do modelo IDOL. Essas dimensões acrescentaram orientações na análise do público alvo e no projeto do modelo da unidade instrucional. Para as fases de desenvolvimento e implementação do ADDIE foram utilizadas as dimensões da fase de estratégia do modelo IDOL. Elas auxiliaram na elaboração dos materiais, seus tipos, formas e conteúdos e também na forma de fornecê-los. Por fim as fases de avaliação do IDOL e do ADDIE foram utilizadas se complementando na etapa de realizar a avaliação da unidade instrucional.

As dimensões citadas acima auxiliam em fatores essenciais como a organização dos conteúdos nas aulas, a forma de disponibilizar a matéria, a necessidade de todas as atividades possuírem gabaritos acessíveis e a forma linear de fornecer as o material.

4.3 Proposta de unidade instrucional

Neste trabalho é desenvolvida uma unidade instrucional extraclasse para o ensino de computação física e programação para crianças abordando conceitos em meio à montagem de uma estrutura que remete aos gostos dos alunos, como desenhos, jogos e etc. Essa unidade será realizada com a utilização da linguagem de programação Scratch e da construção de um boneco com a plataforma de prototipagem eletrônica Arduino. Além disso, também será utilizado materiais recicláveis para a montagem da estrutura física do boneco.

Por meio da análise de contexto e dos objetivos de aprendizagem definidos na tabela 19 da seção 4.1 é proposta uma atividade com os seguintes conteúdos: conceitos de *hardware*, *software* e programação (operações lógicas, variáveis, condicionais, função e

laços). Com base nisto é apresentado o plano de ensino para a atividade em questão que compreende cada um dos conceitos definidos para aprendizagem. A tabela 20 descreve o plano de ensino e as tabela 21 e 22, o cronograma de atividades e conteúdo programático. As tabelas 20 e 21 se diferenciam na estimativa de duração da atividade instrucional para um aluno com perfil ideal e um aluno sem um conhecimento prévio necessário.

Tabela 20 - Plano de ensino

Plano de ensino
Identificação
Curso para ensinar conceitos de computação física juntamente de conceitos de programação. Inclui a prática de programação, na qual os alunos entram no mundo da computação aprendendo a programar o hardware com tema Pokemon. Os alunos, ao longo das aulas, aprenderão conceitos fundamentais de programação usando Scratch, uma linguagem de progra mação gráfica desenvolvida no *Massachusetts Institute of Technology* (MIT) Media Lab, e alguns conceitos de hardware (Arduino, protoboard, atuadores, LED e sensores).
Cursos
• Programação • Computação Física
Requisitos
Noções básicas de programação e\ou computação física.
Ementa
Conceitos básicos de computação. Diferença de Hardware e Software. Algoritmo, Conceitos de Linguagem de programação e Ambiente de programação (Scratch). Conceitos básicos de linguagem de programação (variáveis, estruturas de controle, estruturas de repetição, condicionais e funções). Conceitos básicos de Computação Física (Arduino, Protoboard, Sensor, Atuador, LEDs, Jumper)
Objetivos
Geral: O Objetivo é retomar conceitos básicos da computação física e programação de acordo com as diretrizes do currículo de referência da ACM CSTA K-12. Transmitir ao aluno uma abordagem de resolução de problemas que pode ser desenvolvida num computador envolvendo conceitos de programação tais como estruturas de dados, chamadas de funções, juntamente do conhecimento de

computação física, sabendo diferenciar no contexto da unidade instrucional a utilização de hardware e software.

Específicos:

- Aprender os conceitos básicos de linguagens de programação.
- Usar abstração para decompor um problema em subproblemas.
- Conseguir criar/programar um algoritmo usando o ambiente para solucionar problemas.
- Compreender o funcionamento das atividades desenvolvidas.
- Compreender os conceitos de hardware e software e saber identificá-los nas atividades desenvolvidas.

Conteúdo Programático

1) Apresentação do curso

Objetivos do curso e apresentação ferramentas utilizadas

2) Apresentação da atividade a ser desenvolvida

Descrição da atividade de captura do Pikachu e suas funcionalidades

3) Introdução à computação e programação

Explicar o que é computação, linguagem de programação, conceitos de programação: variáveis, operadores lógicos e aritméticos, estruturas condicionais, estruturas de repetição, funções

4) Introdução ao Scratch

Instalação do ambiente Scratch, mudança de idioma e desenvolvimento de um programa exemplo com comandos básicos

5) Introdução aos conceitos de computação física

Conceituar computação física e explicar sobre os componentes que serão utilizados (arduino, protoboard, LED, atuador e sensor)

6) Instalação e configuração do ambiente

Instalação e manipulação do Arduino IDE, instalação de drivers, execução de códigos básicos de funcionamento do Arduino, instalação do servidor de comunicação para o Scratch, instalação do Python e algumas de suas bibliotecas, blocos de comunicação do Scratch

7) Teste do ambiente

Programa exemplo para testar a comunicação entre o Scratch e Arduino

8) Montagem da estrutura inicial da atividade

Elaboração da estrutura básica inicial para os componentes utilizados na atividade, montagem do origami

9) Montagem da pokebola

Manipulação da estrutura da pokebola, manipulação do LED

10) Programação da pokebola

Implementação do código responsável pela emissão da luz emitida pela pokebola para a captura do pikachu

11) Trabalhando com a movimentação

Manipulação dos componentes responsáveis pela movimentação do Pikachu

12) Programação dos movimentos

Implementação do código responsável pela movimentação do Pikachu

13) Trabalhando com os sensores

Manipulação dos componentes responsáveis por percepção do Pikachu

14) Programação sensores

Implementação do código responsável pela percepção do Pikachu

15) Melhorando o código

Melhoria do código para deixá-lo mais organizado e legível com o intuito de ensinar outros conceitos de programação

1**6) Desafios**

Desafios com o intuito de fixar melhor o conteúdo e instigar o interesse pelo assunto

Metodologia

Por se tratar de EaD (Ensino à Distância) o conteúdo é transmitido através do sequenciamento de aulas em slides, chamados de guias, com seus respectivos manuais manuais. Os guias são responsáveis pela transmissão do conteúdo teórico. Os manuais tratam da parte prática, instalação de ambientes, atividades práticas de programação e montagem da estrutura física. Após a conclusão do conteúdo teórico existe a atividade final misturando os conceitos vistos de programação e computação física, juntamente de alguns desafios extras.

Cronograma

1) Aula 1 - Introdução

- Guia 1

2) Aula 2 - Conceitos de Programação

- Guia 2
- Manual 1

3) Aula 3 - Conceitos de Computação Física

- Guia 3
- Manual 2 e 3

4) Aula 4 - Desenvolvimento da Atividade Captura Pokemon

- Guia 4
- Manual 4

5) Questionário pós-unidade

Tabela 21 - Cronograma de atividades e conteúdo programático (aluno ideal)

Sequência das atividades	Duração (min)	Estratégia instrucional	Recursos didáticos
Atividade Captura do Pikachu			
1. Apresentação da unidade instrucional Objetivos da unidade e das ferramentas utilizadas	10	Atividade conceitual	Guia 1 - Aula 1 - Introdução
2. Apresentação da atividade Descrição da atividade de captura do Pikachu e suas funcionalidades	5	Atividade conceitual	Guia 1 - Aula 1 - Introdução
3. Introdução à computação e programação Explicar o que é computação, linguagem de programação, conceitos de programação: variáveis, operadores lógicos e aritméticos, estruturas condicionais, estruturas de repetição, funções	25	Atividade conceitual	Guia 2 - Aula 2 - Conceitos de programação
4. Introdução ao Scratch Instalação do ambiente Scratch, mudança de idioma e desenvolvimento de um programa exemplo com comandos básicos	25	Atividade conceitual e prática através de tutorial	Manual 1 - Instalação e desenvolvimento do primeiro código no Scratch Ambiente de desenvolvimento Scratch
5. Introdução aos conceitos de computação física Conceituar computação física e explicar sobre os componentes que serão utilizados (arduino, protoboard, LED, atuador e sensor)	25	Atividade conceitual e prática através de um tutorial	Guia 3 - Aula 3 – Conceitos de Computação Física Kit de *hardware* (Arduino Nano, protoboard, LEDs, resistor, atuador, jumper macho-macho, jumper macho-fêmea e sensor de ultrassom)
6. Instalação e configuração do ambiente Instalação e manipulação do Arduino IDE, instalação de drivers, execução de códigos básicos de funcionamento do Arduino, instalação do	45	Atividade prática através de um tutorial	Guia 3 - Aula 3 - Conceitos de Computação Física Arduino IDE Instaladores (arquivos *.bat*)

servidor de comunicação para o Scratch, instalação do Python e algumas de suas bibliotecas, blocos de comunicação do Scratch			Tutorial 2 - Instalação e configuração do ambiente Python e alguma de suas bibliotecas Blocos de comunicação Scratch
7. Teste do ambiente Programa exemplo para testar a comunicação entre o Scratch e Arduino	15	Atividade prática através do desenvolvimento de um programa teste	Guia 3 - Aula 3 - Conceitos de Computação Física Manual 3 - Testando o ambiente Arduino/Scratch Ambiente de desenvolvimento Scratch
8. Montagem da estrutura inicial da atividade Elaboração da estrutura básica inicial para os componentes utilizados na atividade, montagem do origami	15	Atividade prática através de um tutorial	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Kit de *hardware* Tutorial 4 - Manipulando os componentes da atividade
9. Montagem da pokebola Manipulação da estrutura da pokebola, manipulação do LED	20	Atividade prática através de um tutorial	Slides Aula 4 – Desenvolvimento da Atividade Captura Pokemon Tutorial 4 - Manipulando os componentes da atividade
10. Programação da pokebola Implementação do código responsável pela emissão da luz emitida pela pokebola para a captura do pikachu	10	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
11. Trabalhando com a movimentação Manipulação dos componentes responsáveis pela movimentação do Pikachu	25	Atividade prática através de um tutorial	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Manual 4 - Manipulando os componentes da atividade
12. Programação dos movimentos Implementação do código responsável pela movimentação do Pikachu	15	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
13. Trabalhando com os sensores Manipulação dos componentes responsáveis por percepção do Pikachu	15	Atividade prática através de um tutorial	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Manual 4 - Manipulando os componentes da atividade
14. Programação sensores Implementação do código responsável pela percepção do	15	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon

Pikachu			Ambiente de desenvolvimento Scratch
15. Melhorando o código Melhoria do código para deixá-lo mais organizado e legível com o intuito de ensinar outros conceitos de programação	10	Atividade prática através de um tutorial Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
16. Desafios Desafios com o intuito de fixar melhor o conteúdo e instigar o interesse pelo assunto	-	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
Avaliação da unidade	-	-	Questionário pós-unidade

Tabela 22 - Cronograma de atividades e conteúdo programático (aluno sem conhecimento prévio)

Sequência das atividades	Duração (min)	Estratégia instrucional	Recursos didáticos
Atividade Captura do Pikachu			
1. Apresentação da unidade instrucional Objetivos da unidade e das ferramentas utilizadas	20	Atividade conceitual	Guia 1 - Aula 1 - Introdução
2. Apresentação da atividade Descrição da atividade de captura do Pikachu e suas funcionalidades	10	Atividade conceitual	Guia 1 - Aula 1 - Introdução
3. Introdução à computação e programação Explicar o que é computação, linguagem de programação, conceitos de programação: variáveis, operadores lógicos e aritméticos, estruturas condicionais, estruturas de repetição, funções	50	Atividade conceitual	Guia 2 - Aula 2 - Conceitos de programação
4. Introdução ao Scratch Instalação do ambiente Scratch, mudança de idioma e desenvolvimento de um programa exemplo com comandos básicos	50	Atividade conceitual e prática através de tutorial	Manual 1 - Instalação e desenvolvimento do primeiro código no Scratch Ambiente de desenvolvimento Scratch

5. Introdução aos conceitos de computação física Conceituar computação física e explicar sobre os componentes que serão utilizados (arduino, protoboard, LED, atuador e sensor)	50	Atividade conceitual e prática através de um tutorial	Guia 3 - Aula 3 – Conceitos de Computação Física Kit de *hardware* (Arduino Nano, protoboard, LEDs, resistor, atuador, jumper macho-macho, jumper macho-fêmea e sensor de ultrassom)
6. Instalação e configuração do ambiente Instalação e manipulação do Arduino IDE, instalação de drivers, execução de códigos básicos de funcionamento do Arduino, instalação do servidor de comunicação para o Scratch, instalação do Python e algumas de suas bibliotecas, blocos de comunicação do Scratch	90	Atividade prática através de um tutorial	Guia 3 - Aula 3 - Conceitos de Computação Física Arduino IDE Instaladores (arquivos *.bat*) Tutorial 2 - Instalação e configuração do ambiente Python e alguma de suas bibliotecas Blocos de comunicação Scratch
7. Teste do ambiente Programa exemplo para testar a comunicação entre o Scratch e Arduino	30	Atividade prática através do desenvolvimento de um programa teste	Guia 3 - Aula 3 - Conceitos de Computação Física Manual 3 - Testando o ambiente Arduino/Scratch Ambiente de desenvolvimento Scratch
8. Montagem da estrutura inicial da atividade Elaboração da estrutura básica inicial para os componentes utilizados na atividade, montagem do origami	30	Atividade prática através de um tutorial	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Kit de *hardware* Tutorial 4 - Manipulando os componentes da atividade
9. Montagem da pokebola Manipulação da estrutura da pokebola, manipulação do LED	40	Atividade prática através de um tutorial	Slides Aula 4 – Desenvolvimento da Atividade Captura Pokemon Tutorial 4 - Manipulando os componentes da atividade
10. Programação da pokebola Implementação do código responsável pela emissão da luz emitida pela pokebola para a captura do pikachu	20	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
11. Trabalhando com a movimentação Manipulação dos componentes responsáveis pela movimentação do Pikachu	50	Atividade prática através de um tutorial	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Manual 4 - Manipulando os componentes da atividade

12. Programação dos movimentos Implementação do código responsável pela movimentação do Pikachu	30	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
13. Trabalhando com os sensores Manipulação dos componentes responsáveis por percepção do Pikachu	30	Atividade prática através de um tutorial	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Manual 4 - Manipulando os componentes da atividade
14. Programação sensores Implementação do código responsável pela percepção do Pikachu	30	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
15. Melhorando o código Melhoria do código para deixá-lo mais organizado e legível com o intuito de ensinar outros conceitos de programação	20	Atividade prática através de um tutorial Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
16. Desafios Desafios com o intuito de fixar melhor o conteúdo e instigar o interesse pelo assunto	-	Atividade prática através de um desenvolvimento de um programa	Guia 4 - Aula 4 – Desenvolvimento da Atividade Captura Pokemon Ambiente de desenvolvimento Scratch
Avaliação da unidade	-	-	Questionário pós-unidade

Para que a aplicação destas atividades e desta unidade sejam efetivas, é necessário atender alguns pré-requisitos de *hardware*, *software* e outros tipos de recursos, os quais podem ser vistos na tabela 23.

Tabela 23 - Pré-requisitos para aplicação da unidade instrucional

Pré-requisitos necessários para a unidade instrucional	
Software	Navegador *Web Mozilla Firefox* ou *Google Chrome* - para baixar o ambiente de programação Scratch
	Sistema Operacional Windows (XP ou superior)
	Arduino IDE para carregar alguns programas para o funcionamento do Arduino

	Instalador para integração Arduino e Scratch (disponibilizado pelos autores)
Hardware	Computador *Desktop* com monitor, teclado e mouse ou *Notebook*
	Acesso à internet no computador ou *notebook* - utilizado para desenvolvimento através do Scratch
	Kit dos componentes eletrônicos - utilizado para realizar as funcionalidades do boneco
Estrutura física	Materiais reciclados - utilizando qualquer material reciclado como por exemplo papel, cano, madeira, garrafa pet, meias, rolo de papel higiênico, dentre outros

4.4 Estrutura física

Tendo em vista o objetivo do presente trabalho em desenvolver uma unidade instrucional de baixo custo definiu-se a utilização do Arduino Nano. Juntamente do mesmo foi escolhido a utilização de outros componentes que podem ser adquiridos também por um baixo custo tais como LEDs, sensores, atuadores, dentre outros. Além disso para a estrutura física optou-se por trabalhar com materiais que também fossem de fácil aquisição e de baixo custo. Na atividade desenvolvida foram utilizados folha de papel, bola de isopor, fita adesiva, lápis e palitinho de churrasco para a composição da estrutura física. Todos os componentes eletrônicos foram adquiridos *on-line* em sites voltados para este tipo de comércio, já os materiais para estrutura física, foram adquiridos em papelarias.

Para a atividade da unidade instrucional foi elaborado algo que pudesse compor os requisitos necessários de aprendizado de computação física e que estivesse dentro do gosto das crianças da faixa etária de 8 a 14 anos. Por isso optou-se por criar uma atividade de captura do Pikachu, conforme ilustra a figura 43. A atividade funciona da seguinte maneira, a pokébola se aproxima do Pikachu e à partir de uma determinada distância, a pokebola emite uma luz. Esta luz pode variar na quantidade de vezes que piscar, simulando a tentativa de captura do Pikachu. Caso a captura do Pikachu venha a acontecer, o mesmo cai no chão. Porém dependendo do nível de força do Pikachu ele pode resistir e não ser capturado pela pokebola, voltando a se “levantar”.

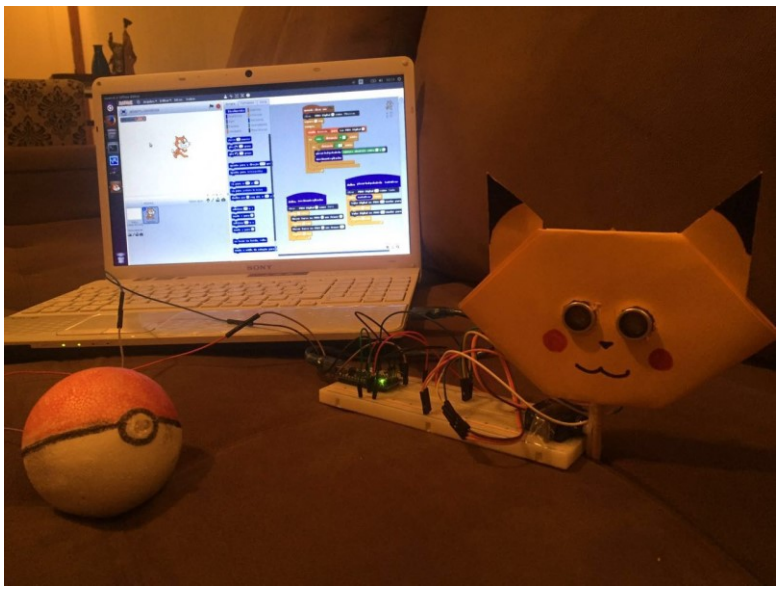


Figura 43 - Atividade para capturar um Pikachu (Fonte: Elaborada pelo(a) autor(a))

4.5 Estrutura para ensino-aprendizagem de programação

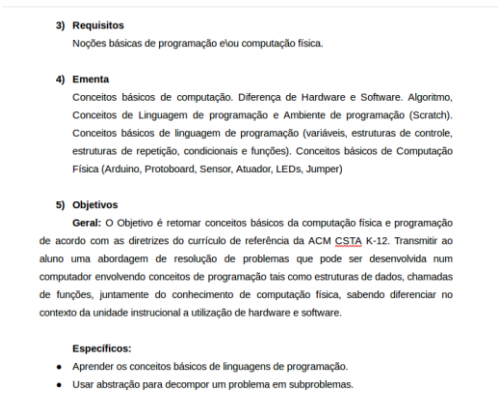
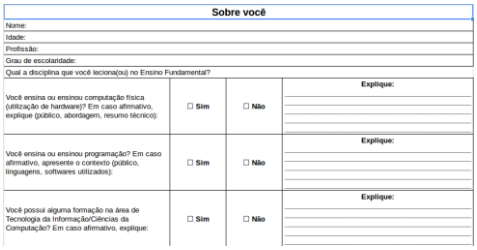
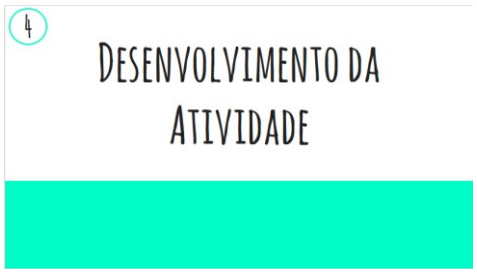
Após a comparação de diversos aspectos mostrados na tabela 3 da seção 2.1.2.5 que compõem as linguagens voltadas ao ensino de programação para crianças, optou-se pela utilização do Scratch. Dentre as linguagens analisadas, o Scratch é a que possui a interface mais amigável e de fácil utilização para as crianças. Também levou-se em consideração na escolha da linguagem, os conceitos de programação que seriam passados na unidade instrucional. No caso da linguagem Scratch, ela possuía todos os conceitos necessários de uma forma simples e didática. Um outro ponto muito forte para a escolha do Scratch foi a sua documentação em relação aos projetos já existentes. Além disso, o Scratch também possui integração com o Arduino e diversos componentes eletrônicos que são utilizados nas atividades de computação física. Existem algumas APIs que realizam essa comunicação entre Scratch/Arduino e são bem simples de utilizá-las. Dentre as disponíveis, optou-se por utilizar a extensão s2a_fm por ser bem utilizada e fácil de manipular.

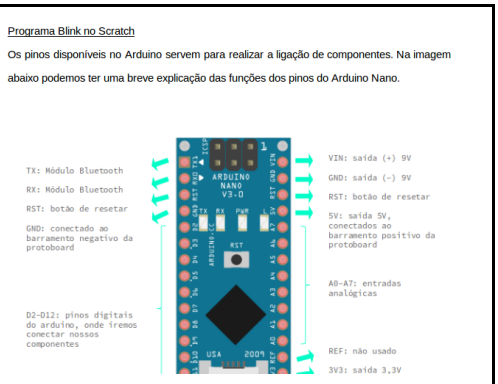
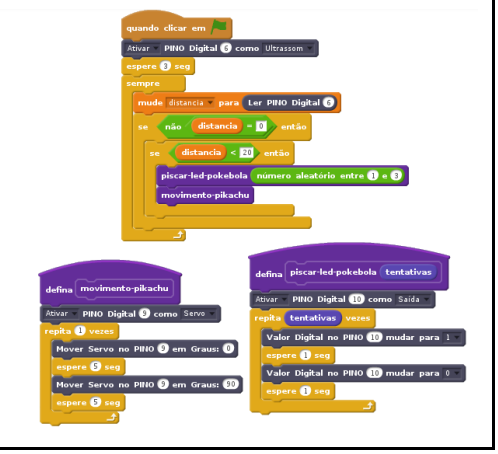
Para estimular o aprendizado de programação através da linguagem Scratch foi elaborado um tutorial mostrando detalhadamente as ações que seriam executadas através da programação pelo boneco do Pikachu durante a atividade. Com o intuito de avaliar o aprendizado da unidade e dos conceitos de programação passados, foi proposto dois desafios no final da atividade. O primeiro desafio foi basicamente melhorar o código desenvolvido e ao mesmo tempo ensinar novos conceitos de programação. Já o segundo desafio era mais voltado ao raciocínio lógico e poder de resolução de problemas.

4.6 Estudo da unidade instrucional

A unidade instrucional, o plano de ensino, os materiais elaborados e a forma de disponibilização dos mesmos foi feita conforme os modelos de design instrucional estudados na Seção 4.2. Os materiais desenvolvidos englobam slides guia, atividades, manuais e gabaritos conforme mostra a tabela 24. O cronograma e conteúdo programático desses materiais podem ser encontrados na seção 4.3 nas tabelas 21 e 22. Os materiais podem ser acessados na íntegra neste
link:
https://drive.google.com/drive/folders/0B5w23bj7pkKmMURWTjN6b2RmdG8?usp=sharing

Tabela 24 - Materiais desenvolvidos para a unidade instrucional

Material	Descrição	Imagem demonstrativa
Plano de ensino	Documento que mostra o planejamento de conteúdo da unidade instrucional apresenta objetivos e sequenciamento de atividades para as aulas	 3) **Requisitos** Noções básicas de programação e\ou computação física. 4) **Ementa** Conceitos básicos de computação. Diferença de Hardware e Software. Algoritmo, Conceitos de Linguagem de programação e Ambiente de programação (Scratch). Conceitos básicos de linguagem de programação (variáveis, estruturas de controle, estruturas de repetição, condicionais e funções). Conceitos básicos de Computação Física (Arduino, Protoboard, Sensor, Atuador, LEDs, Jumper) 5) **Objetivos** **Geral:** O Objetivo é retomar conceitos básicos da computação física e programação de acordo com as diretrizes do currículo de referência da ACM CSTA K-12. Transmitir ao aluno uma abordagem de resolução de problemas que pode ser desenvolvida num computador envolvendo conceitos de programação tais como estruturas de dados, chamadas de funções, juntamente do conhecimento de computação física, sabendo diferenciar no contexto da unidade instrucional a utilização de hardware e software. **Específicos:** • Aprender os conceitos básicos de linguagens de programação. • Usar abstração para decompor um problema em subproblemas.
Questionários	Documentos para serem aplicados em crianças e professores para avaliação da unidade instrucional	 **Sobre você** Nome: Idade: Profissão: Grau de escolaridade: Qual a disciplina que você leciona(ou) no Ensino Fundamental? Você ensina ou ensinou computação física (utilização de hardware)? Em caso afirmativo, explique (público, abordagem, resumo técnico): ☐ Sim ☐ Não **Explique:** Você ensina ou ensinou programação? Em caso afirmativo, apresente o contexto (público, linguagens, softwares utilizados): ☐ Sim ☐ Não **Explique:** Você possui alguma formação na área de Tecnologia da Informação/Ciências da Computação? Em caso afirmativo, explique: ☐ Sim ☐ Não **Explique:**
Guias	Documentos para apresentação dos conceitos que serão abordados durante a unidade instrucional	

Manuais	Documentos que auxiliam em diversas etapas da unidade instrucional tanto para aprendizado quanto para desenvolvimento da atividade e configuração de ambiente	
Gabaritos	Documento para auxiliar a verificar se a atividade ou desafio foi desenvolvido corretamente.	

4.7 Avaliação da unidade

Este capítulo apresenta a execução da avaliação, mostrando as partes envolvidas no processo, bem como suas avaliações. É apresentado também um detalhamento maior em relação ao dados coletados.

4.7.1 Execução da avaliação

O foco principal da avaliação consiste em compreender se através da implementação de uma unidade instrucional de computação física é possível despertar um interesse maior por parte das crianças pela área de computação. A avaliação da unidade instrucional foi feita de duas maneiras. Foi feita a avaliação com um aluno ideal, ou seja, que se encaixa em todos os quesitos do perfil do público alvo estudado e por um professor que ministra aulas de computação física.

A unidade instrucional foi aplicada com um aluno do Ensino Fundamental, do Colégio Geração, durante o segundo semestre do ano de 2016, escolhido por proximidade de parentesco de um dos presentes autores. Os autores estavam presentes na execução da unidade com o intuito de encontrar possíveis melhorias para aplicação da unidade. A aplicação da atividade instrucional pode ser vista na figura 44.

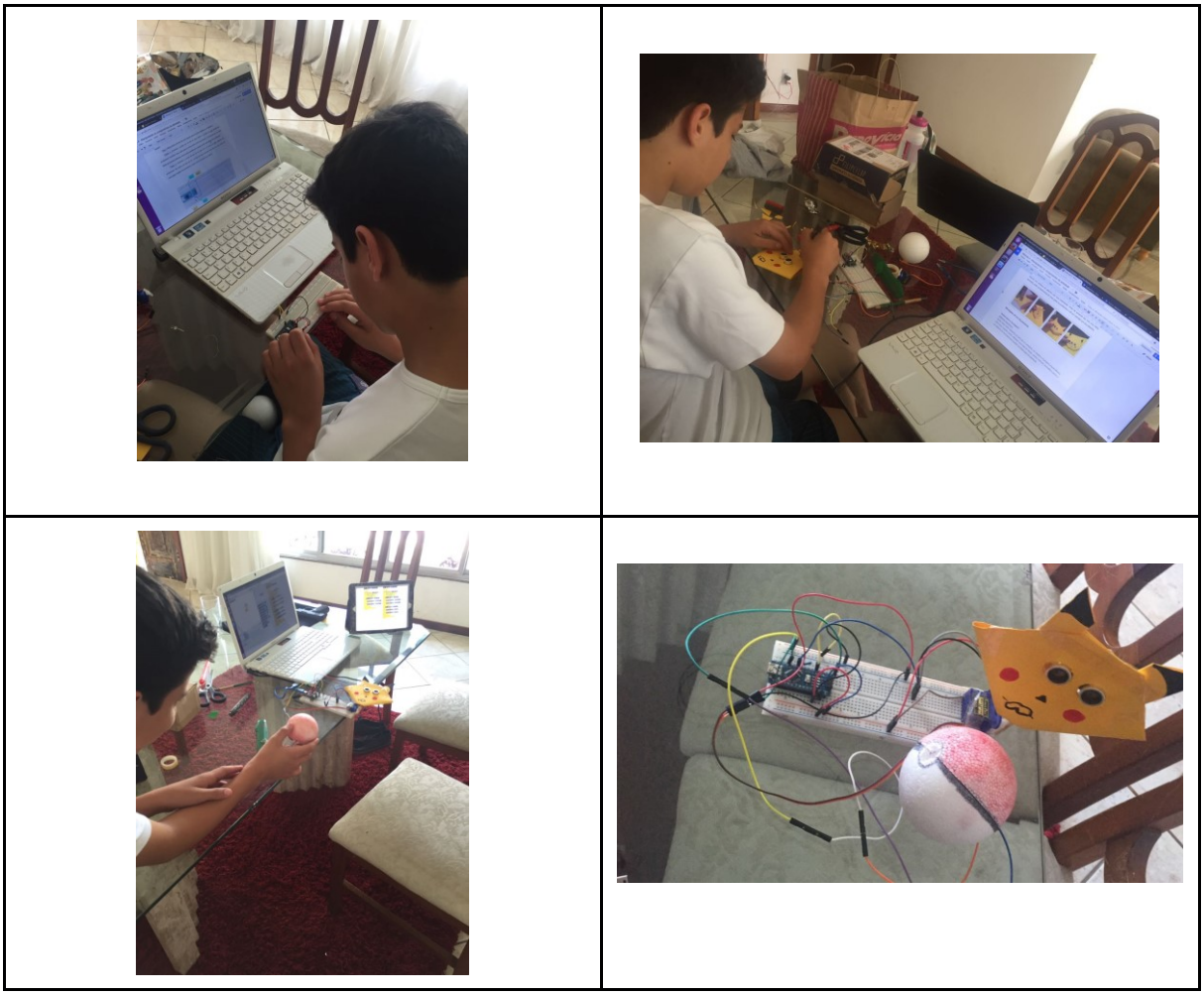


Figura 44 - Execução da atividade no aluno (Fonte: Elaborado pelo(a) autor(a))

Além da aplicação da unidade instrucional com um aluno do Ensino Fundamental, também foi feita uma avaliação da mesma com um professor que ministra aulas de computação física para a mesma faixa etária. O professor foi escolhido por indicação do aluno que realizou a atividade e também avaliou a atividade durante o segundo semestre do ano de 2016. O professor que realizou a avaliação dá aulas de física e robótica há mais de 10 anos. Já atuou em diversos colégios em atividades com o Lego Mindstorms e robótica com componentes da base Arduino. Possui uma boa experiência em relação ao ensino de programação para diversas faixas etárias com diferentes linguagens e abordagens.

A aplicação da unidade instrucional foi feita seguindo os sequenciamentos das atividades do plano de ensino contidas na seção 4.3, abrangendo todos os itens listados na tabela 24. O aluno realizou a atividade em dois momentos, primeiro montando a atividade em um computador com o sistema operacional Linux com o ambiente todo instalado e configurado. Em um segundo momento, o aluno foi submetido a realizar a instalação e

configuração do ambiente em um computador com sistema operacional Windows e após isso desenvolver o mesmo código feito anteriormente, realizando somente neste momento a etapa de desafios propostas na atividade. Nos dois sistemas operacionais foram utilizados o Scratch *off-line* na versão 2.0. Dentre os recursos que poderiam ser utilizados na atividade, não foi possível realizar a comunicação entre o Scratch e o Arduino utilizando o programa disponibilizado pela iniciativa Computação na Escola. O impedimento em relação à utilização deste recurso foi a questão da versão em que o comunicador se encontrava em um de suas determinadas bibliotecas. Para contornar este problema, foi criado um arquivo configurável e disponibilizado um tutorial passo-a-passo no ambiente do sistema operacional Windows. A instalação e configuração do ambiente Linux não foi documentada pois foi feito somente para testes e desenvolvimento.

4.7.2 Análise dos dados coletados

Os dados foram coletados através de questionários que podem ser visualizados na seção de anexos do presente trabalho feito tanto para o aluno quanto para o professor após a aplicação e/ou avaliação da unidade instrucional. Para o aluno também levou-se em consideração, a observação durante a aplicação da unidade.

4.7.2.1 Dados coletados por parte do aluno

Em relação ao atingimento dos objetivos de aprendizagem da unidade foi possível verificar que o aluno sabia acompanhar e a analisar a sequência de instruções a ser seguida. Porém nem todos os conceitos que foram passados demonstraram ser entendidos. A principal dificuldade foi entender os conceitos que eram utilizados na elaboração da configuração do ambiente por possuir termos que estavam totalmente fora do conhecimento e aprendizado do aluno.

De acordo com o questionário aplicado, na pergunta "**O que você aprendeu sobre computação física?**", foi comentado o aprendizado de conceitos como protoboard, jumpers e seus diferentes tipos utilizados na atividade e a possibilidade de programar através do Scratch. Em relação a pergunta "**O que você aprendeu sobre programação?**" foi comentado o aprendizado de conceitos como variáveis, blocos condicionais e criação de blocos para a manipulação de funções.

A partir da percepção em relação ao grau de facilidade de aprendizagem da unidade passada pelo aluno, foi notado que em alguns casos não era possível entender totalmente o conteúdo textual dos tutoriais. Porém após a visualização das imagens indicando o que era

para ser o feito, o mesmo conseguia se desenvolver sozinho. Além disso, analisando as respostas no questionário do aluno, ficou evidente de que ele não encontrou sérias dificuldades na elaboração da atividade. Ele considerou a atividade em um nível médio e mesmo assim considerou o seu aprendizado tanto na parte de programação quanto na parte de computação bastante efetivo. Durante a elaboração dos desafios, o aluno mencionou que se sentiu um pouco perdido em relação à lógica condicional do comando "se, então" pois era difícil de entender.

Durante a aplicação da atividade, com o decorrer do tempo, foi possível perceber que o aluno se sentia cada confiante em realizar as etapas da atividade. Também observou-se que quando o aluno era submetido a manipular alguns componentes que ainda não conhecia, mostrava estar um pouco desconfortável e com dúvidas de como fazê-lo. Ao fim da atividade, o aluno se mostrava cansado porém ainda sim queria continuar a fazer algo a mais. No questionário foi possível avaliar que o aluno gostou bastante de ter feito a atividade e que gostaria de dar continuidade na realização de mais tarefas de programação e computação física.

O aluno também foi avaliado em relação a apontar pontos positivos e negativos da atividade. Como ponto positivo foi ressaltado um conteúdo completo disponibilizado para a parte de programação. O aluno informou que acredita que ao se basear pelo material disponibilizado poderá aprender melhor os conceitos de programação. Já como ponto negativo a questão da manipulação dos jumpers na protoboard. O aluno se sentiu um pouco desconfortável e inseguro em relação a conexão dos jumpers nos furos da protoboard.

Os dados coletados foram classificados em um número quantitativo, variando em uma escala de 1 a 3, onde o valor 1 representa "Pouco", o valor 2 representa "Mais ou Menos" e o valor 3 representa "Muito". Estes dados são mostrados na figura 45.

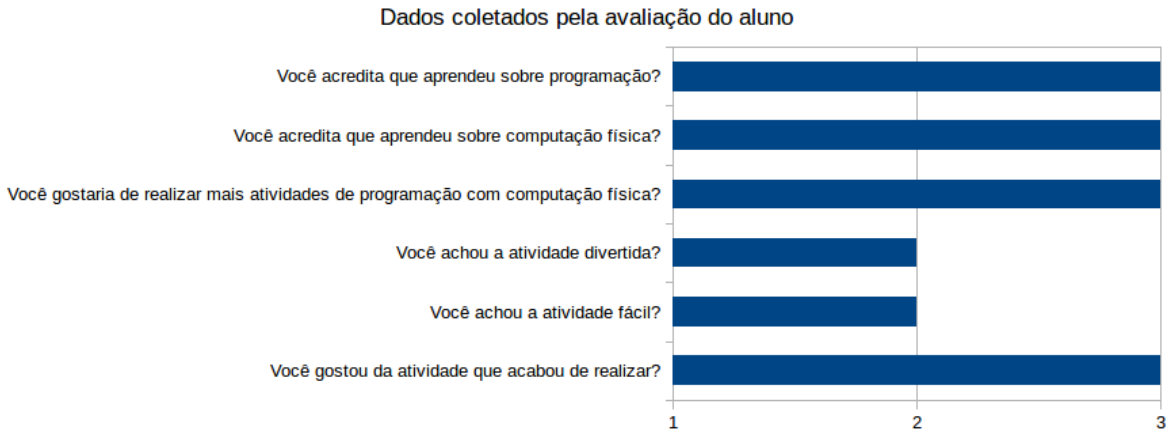


Figura 45 - Dados coletados pela avaliação do aluno na unidade instrucional (Fonte: Elaborado pelo(a) autor(a))

4.7.2.2 Dados coletados por parte do professor

Em relação aos objetivos de ensino de computação física, o professor avaliou a atividade em uma abordagem clara e limpa, onde o embasamento didático está bem definido em todas as etapas da unidade. Ele acredita que o aluno que completar a atividade estará apto a interagir com os elementos básicos e essenciais da computação. Além disso, o plano de ensino e os guias de apresentação do conteúdo mostram-se estruturados e planejados em detalhes. Na parte de avaliação do ensino de programação, avaliou-se a unidade como ideal para introduzir os conceitos básicos de programação. Também foi comentado a boa escolha pela utilização do Scratch, que segundo o mesmo já utiliza há bastante tempo para introduzir conceitos de programação.

No questionário foi feito o questionamento se a avaliação em relação a apresentação e contextualização das etapas de desenvolvimento propostas eram apropriadas para o público alvo. Como resposta foi enfatizado o sequenciamento lógico adequado e os objetivos explícitos de uma maneira geral.

Em relação a elaboração do material didático da atividade, o professor acredita que o tema envolve o interesse dos alunos desta faixa etária. É desenvolvido também um procedimento que explora o tema de forma lúdica e bem planejada.

A atividade desenvolvida para ser aplicada em alunos que já possuem conhecimento prévio sobre o conteúdo foi avaliada como interessante. Utilizando o material desenvolvido, o aluno poderá explorar de uma maneira mais completa as etapas do trabalho, por já conhecer alguns elementos que compõem a execução da atividade.

Como avaliação do tema da atividade para a faixa etária em questão foi avaliado que a atividade é bastante adequada e com um pouco de criatividade pode ser aperfeiçoada ou então até mesmo substituída por algo na mesma vertente sendo executado por outros alunos interessados.

A atividade desenvolvida foi avaliada de forma extremamente eficiente para o quesito baixo custo. O próprio professor ministra um curso, onde os alunos utilizam como material, componentes elaborados com uma impressora 3D. Porém essa solução acaba tendo um custo altíssimo. A solução apresentada de baixo custo trouxe algumas reflexões para o mesmo sobre os objetivos de ensino e a facilidade de sua aplicabilidade.

O professor também foi avaliado em relação a apontar pontos positivos e negativos da atividade. Como pontos positivos foi ressaltado a boa estrutura da atividade em geral e a abordagem efetiva da mesma. Como pontos negativos, não foi citado nada que possa ser considerado. Só foi ressaltado problemas recorrentes para a área de ensino em relação ao conhecimento prévio e facilidade de aprendizado dos alunos.

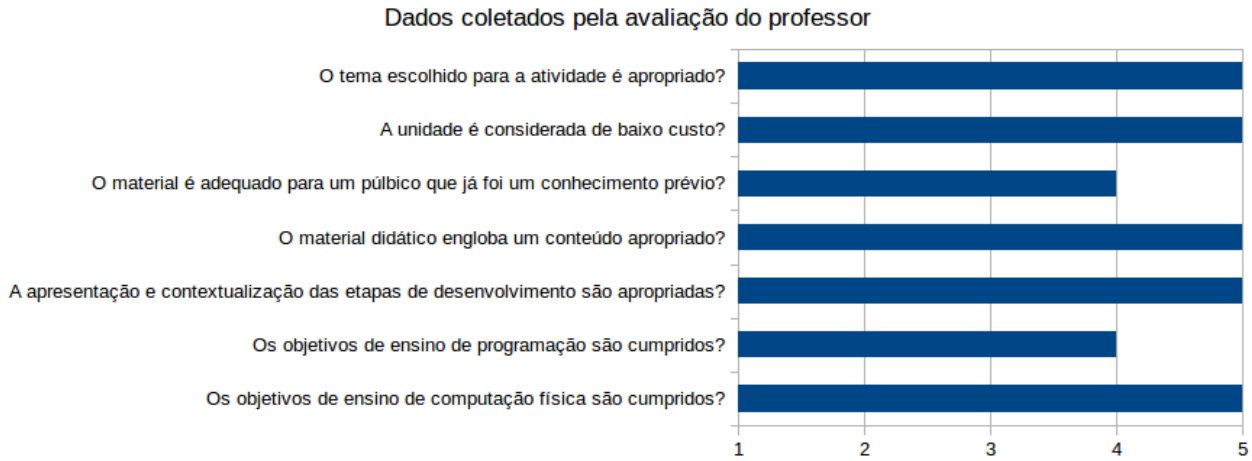


Figura 46 - Dados coletados pela avaliação do professor na unidade instrucional (Fonte: Elaborado pelo(a) autor(a))

Os dados coletados mostrados na figura 46, foram classificados em um número quantitativo, variando em uma escala de 1 a 5, onde quanto maior o valor, maior proximidade ele tem com a pergunta em questão.

5. CONCLUSÃO

O objetivo do presente trabalho foi o desenvolvimento de uma unidade instrucional extraclasse para ensinar computação física e programação para crianças de 8 a 14 anos que já tiveram algum contato com esses assuntos. Nesse contexto, foi apresentada uma síntese sobre a fundamentação teórica de computação física, *hardware* e *software*, aprendizagem e ensino de computação. Além disso, foi realizado uma revisão do estado da arte referente ao ensino de computação física e/ou programação para crianças e jovens. Como primeiro passo para o desenvolvimento da unidade instrucional, foi feito uma análise do contexto e do público-alvo. Com base nesta análise, foram definidos os objetivos de aprendizado a serem abordados pela unidade instrucional. Seguindo o processo de design instrucional foi projetado a unidade instrucional e desenvolvidos todos os materiais didáticos necessários. A unidade foi aplicada em um aluno que se encaixava no público alvo e em um professor que leciona computação física. Foi realizada a avaliação da qualidade da unidade instrucional. De uma maneira em geral os questionamentos feitos sobre a unidade tiveram *feedbacks* positivos. Tanto o professor quanto o aluno apontaram pontos positivos e negativos em relação a aplicação da unidade. Um ponto interessante apontado na avaliação do professor, foi a questão da utilização de materiais de baixo custo para realizar atividades de computação física. Em suas aulas, o professor utiliza recursos que são de altíssimo custo e após a avaliação da unidade instrucional pensa em alterar a estrutura física de suas aulas. A solução apresentada pela atividade lhes trouxe reflexões sobre os objetivos de ensino e a facilidade na sua aplicabilidade. Com esse *feedback* pode-se concluir que a unidade instrucional pode atingir o seu objetivo de aprendizagem. Contribuindo, dessa forma com o ensino de computação por meio de uma experiência agradável, motivadora e satisfatória.

Como resultado do presente trabalho, é disponibilizado uma unidade instrucional para ensinar computação física e programação. Essa unidade pode ser aplicada em crianças de 8 a 14 anos preferencialmente com experiência prévia em computação, mas isso não impossibilita que inexperientes consigam realizá-la. O objetivo da unidade instrucional é reforçar ao aluno conceitos básicos da computação física e programação, mantendo o interesse e a prática de computação sem ter que sair de casa.

Também é resultado do presente trabalho o modelo de design instrucional utilizado. Após terem sido estudados os modelos ADDIE e IDOL, foi desenvolvido um modelo híbrido que faz uso das fases do ADDIE juntamente de algumas dimensões do IDOL. As dimensões

trabalhadas possuem como foco o ensino à distância e na ausência de comunicação com algum professor. Esse modelo pode ser utilizado em quaisquer outras unidades instrucionais, inclusive de áreas distintas de computação.

Como conteúdo para trabalhos futuros, é possível fazer um estudo de caso para um público alvo diferente, como por exemplo crianças totalmente leigas, que nunca tiveram contato com conteúdos de computação. Sendo assim a unidade deve abordar os conceitos de maneira mais aprofundada. Por via de comparação pode-se fazer uma análise estatística do cumprimento dos objetivos de aprendizagem dessa nova unidade. Aprimorar a unidade instrucional dividindo-a em sub-unidades e também incrementando os materiais didáticos disponibilizando um vídeo de passo-a-passo. Além disso, pode-se realizar a unidade instrucional com outros tipos de hardware, como por exemplo o Galileo da intel. É possível empregar uma nova abordagem para a unidade instrucional utilizando realidade aumentada, abandonando a parte de computação física e dificuldades de instalação, dessa forma focando apenas no ensino de programação. Para facilitar a parte de instalação de ambiente seria interessante elaborar um instalador que ficasse responsável por toda a configuração de ambiente. Aprimorar o meio de avaliação para que possa ser possível validar se todos os objetivos foram alcançados. Com base no *feedback* do aluno foi notado a necessidade da criação de outras unidades para ensinar os conceitos de programação de uma forma mais detalhada. E por fim melhorar a abordagem de propor desafios de programação.

6. REFERÊNCIAS

The CSTA Standards Task Force. **CSTA K–12 Computer Science Standards** – Revised 2011,ACM, New York/USA, 2011.

CAMARATA, K. et al. **A Physical Computing Studio: Exploring Computational Artifacts and Environments**. International Journal of Architectural Computing, 1(2): 169-190, 2003.

BAGNALL, B. ***Core LEGO Mindstorms***. Prentice-Hall PTR. 2002.

IGOE, Tom. and O'SULLIVAN, Dan. **Physical Computing – Sensing and Controlling the Physical World with Computers**, Boston, MA: Thomson Course Technology. 2004.

IFRAH, Georges. ***The Universal History of Computing*** **- New York: John Wiley and Sons**. p. 121-122. 2001.

SOARES, Cristiane da Silva; ALVES, Thays de Souza. **Sociedade da informação no Brasil: Inclusão Digital e a importância do profissional de TI.** Rio de Janeiro: Brasil Escola, 2012.

MARTINS, Leandro. **Curso Profissional de Hardware** - São Paulo: Digerati Books. 2007.

CHICOLI, Milton. **Guia de Manutenção de PCs e notebooks** - São Paulo: Digerati Books. 2008.

FRENZEL JR., L. E. **Eletrônica moderna: fundamentos, dispositivos, circuitos e sistemas.** Porto Alegre: AMGH, 2016.

MITROVIC, N. et al. **Physical computing and Android in Robotics**. Mediterranean Conference on Embedded Computing MECD, Budva, Montenegro. 2013.

KATO, Y. **Splish: A Visual Programming Environment for Arduino to Accelerate Physical Computing Experiences**. 2010.

LEMOS, Manoel. **Arduino: Conheça esta plataforma de hardware livre e suas aplicações**. 2014. Disponível em: <http://imasters.com.br/desenvolvimento/arduino-conheca-esta-plataforma-de-hardware-livre-e-suas-aplicacoes/>. Acesso em: 03 abr. 2016.

COMPUTACAOFISICABR. **Sobre Computação Física**. 2014. Disponível em: <https://computacaofisicabr.wordpress.com/o-que-e-computacao-fisica/>. Acesso em: 04 abr. 2016.

DREAMFELL. **Computação Física**. 2009. Disponível em: <https://dreamfeel.wordpress.com/2009/03/07/computacao-fisica/>. Acesso em: 04 abr. 2016.

EDUCATIONALTECHNOLOGY. **The ADDIE Model: Instructional Design**. 2014. Disponível em: <http://educationaltechnology.net/the-addie-model-instructional-design/>. Acesso em: 04 abr. 2016.

NETTO, Samuel Pfromm. **Psicologia da aprendizagem e do ensino**. São Paulo. EPU, 1987.

CHENG, Zhuoqun; LI, Ye; WEST, Richard. **Qduino: A Multithreaded Arduino System for Embedded Computing.** 2015. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7383583>. Acesso em: 04 abr. 2016.

BLIKSTEIN, Paulo. **Gears of Our Childhood: Constructionist Toolkits, Robotics, and Physical Computing, Past and Future.** 2013. Disponível em: <https://www.researchgate.net/profile/Paulo_Blikstein2/publication/262201733_Gears_of_our_childhood_constructionist_toolkits_robotics_and_physical_computing_past_and_future/links/55eb629e08ae21d099c5e87a.pdf>. Acesso em: 04 abr. 2016.

HERGET, L. M. and BODARKY, M. Bodarky. **Engaging Students with Open Source Technologies and Arduino.** Research Integrated Solutions (RIS), IBM TJ Watson Research Center. 2015.

IEEE, **Ieee global history network: Microcontroller.** Disponível em: <http://www.ieeeghn.org/wiki/index.php/Microcontroller> Acesso em: 06 abr. 2016.

ARDUINO, **Introduction: What is Arduino?** Disponível em: <http://www.arduino.cc/en/Guide/Introduction> Acesso em: 06 abr. 2016.

ARDUINO, **Arduino - ArduinoBoardNano.** Disponível em: <https://www.arduino.cc/en/Main/ArduinoBoardNano> Acesso em: 06 abr. 2016.

BANZI, M. **Primeiros Passos com o Arduino**. São Paulo: Novatec, 2011. 152 p.

KISS, G. **Using the Lego-Mindstorm kit in German Computer Science Education.** 8th IEEE International Symposium on Applied Machine Intelligence and Informatics, Herl'any, Slovakia. 2010.

SOUSA, Diego. **Arduino ou Raspberry Pi? Saiba qual micro PC é melhor para seu projeto**. Disponível em: <http://www.techtudo.com.br/noticias/noticia/2015/04/arduino-ou-raspberry-pi-saiba-qual-micro-pc-e-melhor-para-seu-projeto.html> Acesso em: 06 abr. 2016

RASPBERRYPI, **Whats is a Raspberry PI?** Disponível em: <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/> Acesso em: 07 abr. 2016

JAMIESON, Peter and HERDTNER, Jeff. **More Missing the Boat - Arduino, Raspberry Pi, and Small Prototyping Boards and Engineering Education Needs Them.** Frontiers in Education Conference (FIE) - IEEE, El Paso, TX, 2015.

ORSINI, L. **Arduino Vs. Raspberry Pi: Which Is The Right DIY Platform For You?** Disponível em: <http://readwrite.com/2014/05/07/arduino-vs-raspberry-pi-projects-diy-platform/> Acesso em: 10 abr. 2016.

PINTO, Carmem Lúcia Quintana; ROCHA, Célia Regina Cruz da; VILARIM, Gilvan. **Desafios da Prática da Interdisciplinaridade em Cursos de Ciência da Computação: a Experiência do UNIFESO**. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/wei/2010/0021.pdf>. Acesso em: 10 abr. 2016.

BELLAMY, B. **Arduino vs. Raspberry Pi vs. CubieBoard vs. Gooseberry vs. APC Rock vs. OLinuXino vs. Hackberry A10**. Disponível em: <http://techwatch.keeward.com/geeks-and-nerds/arduino-vs-raspberry-pi-vs-cubieboard-vs-gooseberry-vs-apc-rock-vs-olinuxino-vs-hackberry-a10/>. Acesso em: 11 abr. 2016.

ALLAN, A. **Arduino Uno vs BeagleBone vs Raspberry Pi.** 2013. Disponível em: <http://makezine.com/2013/04/15/arduino-uno-vs-beaglebone-vs-raspberry-pi/>. Acesso em: 11 abr. 2016.

AKSOY, Pelin; DENARDIS, Laura. **Information Technology in Theory.** Canada: Thomson Course Technology, 2008.

SILVA, Clarence W de. **Sensors and Actuadors: Engineering System Instrumentation.** 2. ed. New York: Crc Press Taylor & Francis Group, 2016.

COMPUTACAONAESCOLA. **Passo 5: Dê sentidos ao seu boneco: sensores.** 2016. Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=189> Acesso em: 25 abr. 2016.

PANDALAB, 2010. P&A LAB: Servo_TechLow_Light_test03 (SoundSensor + Servo + Dimmer Light). Disponível em: <http://pandalabccc.blogspot.com.br/2010/04/servotechlowlighttest03-soundsensor.html> Acesso em: 26 abr. 2016.

COMPUTACAONAESCOLA. **Passo 4: Faça o boneco abanar o braço.** Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=324> Acesso em: 25 abr. 2016.

SIMPSON, E. J.. "**The classification of educational objectives: Psychomotor domain**". 1972. Illinois Journal of Home Economics 10 (4): 110–144.

KEESEE, G. S. **Teaching and learning resources / Instructional Approaches**. 2015. Disponível em: <http://teachinglearningresources.pbworks.com/w/page/19919560/Instructional%20Approaches> Acesso em: 27 abr. 2016.

FERNANDES, J. **O que é um Programa (Software)?** 2002. Disponível em: <http://cic.unb.br/~jhcf/MyBooks/iess/Software/oqueehsoftware.html>. Acesso em: 27 abr. 2016.

UFPA. **Programas - Função e tipos.** 2010. Disponível em: <http://www.ufpa.br/dicas/progra/protipos.htm>. Acesso em: 27 abr. 2016.

JAVA, **O que é Java?** 2016. Disponível em: <https://www.java.com/pt_BR/about/whatis_java.jsp>. Acesso em: 27 abr. 2016.

J.J. Dujmovic and H. Nagashima. **LSP Method and its use for evaluation of Java IDEs.** International Journal of Approximation Reasoning, 41 :3-22, 2006.

SANDOVAL-REYES, Sergio. **Visual Learning Environments for Computer Programming**. Centre For Comput. Res., Nat. Polytech. Inst., Mexico City, Mexico: Electronics, Robotics And Automotive Mechanics Conference (cerma), IEEE, 2011.

SHU, N. C. **A visual programming language designed for automatic programming**. International Conference on System Sciences, Kailua-Kona, HI, USA, 1988.

COMPUTACAONAESCOLA. **SCRATCHBOARD – Uma plataforma simples e fácil de usar para computação física com Scratch e Arduino**. 2016. Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=75>. Acesso em: 28 abr. 2016.

DZHENZHER, V. O. **Computer Simulation at School: Scratch and Programming Language Choosing Criteria**. Global Engineering Education Conference (EDUCON), Military Museum and Cultural Center, Harbiye, Istanbul, Turkey, p. 727 – 732, 2014.

BERKELEY. **Snap! (Build Your Own Blocks) 4.0**. 2016. Disponível em: <http://snap.berkeley.edu/>. Acesso em 28 abr. 2016.

MODMYPI. **HC-SR04 Ultrasonic Range Sensor on the Raspberry Pi**. 2014. Disponível em: <http://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi>. Acesso em em: 1 mai. 2016.

CALIN, D. Ty**pes of sensors for target detection and tracking**. 2014. Disponível em: <http://www.intorobotics.com/types-sensors-target-detection-tracking/>. Acesso em: 1 mai. 2016.

COTE, J. P. **Getting started with physical computing in JavaScript**. 2015. Disponível em: <http://tangiblejs.com/posts/getting-started-with-physical-computing-in-javascript>. Acesso em: 1 mai. 2016.

THOMSEN, A. **Micro Servo Motor 9g SG90 com Arduino Uno**. 2013. Disponível em: <http://blog.filipeflop.com/motores-e-servos/micro-servo-motor-9g-sg90-com-arduino-uno.html> Acesso em: 1 mai. 2016.

JULIAN, R. **Engenharia de Software Orientada a Agentes**. 2016. Disponível em: <http://www.devmedia.com.br/engenharia-de-software-orientada-a-agentes/29238> Acesso em: 1 mai. 2016.

ARAUJO, T. **Arduino Nano 3.0 – Conheça este pequeno e poderoso membro da família Arduino**. 2014. Disponível em: <http://blog.fazedores.com/arduino-nano-3-0/> Acesso em: 1 mai. 2016.

ARAUJO, T. **Sensor Ultrassônico com Arduino**. 2014. Disponível em: <http://blog.fazedores.com/sensor-ultrassonico-com-arduino/> Acesso em: 1 mai. 2016.

B.A. JULIANO, R.S. RENNER, F. JAUREGUI. **LEGO Mindstorms - Hitachi H8-based RCX brick**. California: Chico Intelligent Systems Laboratory, 2004. Disponível em: <http://www.docfoc.com/lego-mindstorms-hitachi-h8-based-rcx-brick-ba-juliano-rs-renner-f-jauregui>. Acesso em: 1 maio 2016.

BARASKAR, P. **Grove Starter Kit with Intel Galileo Gen 2 – Getting Started**. 2015. Disponível em: <https://software.intel.com/en-us/blogs/2015/05/29/grove-starter-kit-with-intel-galileo-gen-2-getting-started-0> Acesso em: 4 mai. 2016.

CHAOUCHI, Hakima. **The Internet of Things.** United States: British Library Cataloguing-in-publication Data, 2010.

INTEL. **Getting Started with the Grove Starter Kit for the Intel Galileo**. 2016. Disponível em: <http://innovationtoolbox.intel.com.au/wp-content/uploads/2015/08/Getting-

Started-with-the-Intel-Galileo-a-possible-classroom-continuum.pdf> Acesso em: 4 mai. 2016.

BATTISTELLA, P. **Ensinar com Jogos**. 2012. Disponível em: <http://pt.slideshare.net/pet-computacao/ensinar-com-jogos> Acesso em 4 mai. 2016.

SCRATCH. **Scratch - Image, Program and Share**. 2016. Disponível em: <https://scratch.mit.edu/> Acesso em 6 de mai. 2016.

SCRATCH. **Scratch - About**. 2016. Disponível em: <https://scratch.mit.edu/about> Acesso em 6 de mai. 2016.

GOOGLE. **FAQ | Blockly | Google Developers**. 2016. Disponível em: <https://developers.google.com/blockly/about/faq> Acesso em 6 de mai. 2016.

JOST, Beate et al. **Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One?** Schloss Birlinghoven, Fraunhofer Iais, St. Augustin, Germany: Multimedia (ism), 2014 Ieee International Symposium On, 2014.

MALONEY, J. et al. **The Scratch Programming Language and Environment**. ACM Transactions on Computing Education, v. 10, n. 4, p. 15, 2010.

COMPUTACAONAESCOLA. **Ajuda para Scratchduino**. Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=53> Acesso em: 6 mai. 2016.

ARDUINO. **Arduino - Firmata Library.** Disponível em: <https://www.arduino.cc/en/reference/firmata> Acesso em: 07 mai. 2016.

COMPUTACAONAESCOLA. **Instalando a Comunicação Scratch/Arduino usando s2a_fm**. Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=61> Acesso em: 7 mai. 2016.

LEGO. **LEGO United States - Home of the toy building brick**. Disponível em: <http://www.lego.com/> Acesso em: 10 mai. 2016.

KITCHENHAM, B. 2004. **"Procedures for Performing Systematic Reviews"**. 2004. Joint Technical Report, TR/SE-0401 and NICTA 0400011T.1, Keele University. Disponível em <http://csnotes.upm.edu.my/kelasmaya/pgkm20910.nsf/0/715071a8011d4c2f482577a700386d3a/$FILE/10.1.1.122.3308%5B1%5D.pdf> Acesso em: 10 mai. 2016.

COMPUTACAONAESCOLA. **Instalando a Comunicação Scratch/Arduino usando s2a_fm**. Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=183> Acesso em: 15 mai. 2016.

MDN. **Introduction to Object-Oriented JavaScript**. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript> Acesso em: 15 mai. 2016.

MDN. **Closures**. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Closures> Acesso em: 15 mai. 2016.

ANU. **Programming - It's a Snap!**. Disponível em: <https://cs.anu.edu.au/research/student-research-projects/programming-its-snap> Acesso em: 15 mai. 2016.

STUMM, V. J. **Funções como objetos de primeira classe**. 2014 Disponível em: <https://pythonhelp.wordpress.com/2014/05/11/funcoes-como-objetos-de-primeira-classe/> Acesso em: 16 mai. 2016.

CANTU, Evandro; SANTOS, Luciano Marcos dos. **Usando a linguagem Scratch e a plataforma Arduino para implementar uma abordagem metodológica baseada em aprender fazendo.** Foz do Iguaçu, Pr, Brasil: Nuevas Ideas En Informática Educativa Tise, 2013.

COKER, E. **Understanding Types of Servo Motors and How They Work.** 2016. Disponível em: <http://makezine.com/2016/05/13/understanding-types-of-servo-motors-and-how-they-work/> Acesso em: 20 mai. 2016.

COMPUTACAONAESCOLA. **Lista de Eletrônica para um Boneco Computação na Escola usando Protoboard.** Disponível em: <http://www.computacaonaescola.ufsc.br/?page_id=497> Acesso em: 22 mai. 2016.

COMPUTACAONAESCOLA. **Fazendo Bonecos Animados: Um Robô que Protege o seu Tesouro**. 2014. Disponível em: <https://computacaonaescola.wordpress.com/2014/04/17/fazendo-bonecos-animados-um-robo-que-protege-o-seu-tesouro/> Acesso em 23 mai. 2016.

ROBOTICAIFAL, 2010. **RECICLABOT: Um Robô Feito de Lixo para Reciclar Lixo**. Disponível em: <http://roboticaifal.webnode.com.br/prototipos/> Acesso em: 23 mai. 2016.

INSTRUCTABLES. **Furry Elephant.** 2012. Disponível em: <http://www.instructables.com/id/Furry-Elephant/> Acesso em: 23 mai. 2016.

COMPUTACAONAESCOLA. **Integrando a Webcam com Arduino: O Minion Mata-Moscas Automático.** 2014. <https://computacaonaescola.wordpress.com/2014/04/16/integrando-a-webcam-com-arduino-o-minion-mata-moscas-automatico/> Acesso em: 23 mai. 2016.

ROVAI, M. **Robô explorador de labirintos, utilizando Inteligência Artificial com Arduino.** 2016. Disponível em: <http://labdegaragem.com/profiles/blogs/rob-explorador-de-labirintos-utilizando-intelig-ncia-artificial> Acesso em: 23 mai. 2016.

OSKAI, W. **Bristlebot: A tiny directional vibrobot.** 2007. Disponível em: <http://www.evilmadscientist.com/2007/bristlebot-a-tiny-directional-vibrobot/> Acesso em: 23 mai. 2016.

COMPUTACAONAESCOLA. **Kleerhangerbeest – o Robô Andarilho de Cabide de Arame.** 2014. Disponível em: <https://computacaonaescola.wordpress.com/2014/06/23/kleerhangerbeest-o-robo-andarilho-de-cabide-de-arame/> Acesso em: 23 mai. 2016.

ATTOEDUCACIONAL. **Atto Educacional.** 2016. Disponível em: <http://attoeducacional.com.br/> Acesso em: 23 mai. 2016.

ATTOEDUCACIONAL. **Atto Educacional.** 2016. Disponível em: <http://jfwdual.wix.com/atto/> Acesso em: 23 mai. 2016.

LEROYMERLIN. **Cano Marrom PVC Soldável 20mm ou 1/2" 3.00m Tigre**. Disponível em: <http://www.leroymerlin.com.br/cano-marrom-pvc-soldavel-20mm-ou-1-2-3-00m-tigre_85949871> Acesso em: 23 mai. 2016.

PONTOCHEIO. **Feltro por Metro - Santa Fé**. Disponível em: <http://www.pontocheio.com.br/feltro-por-metro--santa-fe.1924.html> Acesso em: 23 jun. 2016.

ROOTROBOT. **The robot that brings code to life.** 2016. Disponível em: <http://www.rootrobot.io/> Acesso em: 26 jun. 2016.

FUTURITE. **Root, a novel bot capable of teaching code to children.** 2016. Disponível em: <http://www.futurite.in/blog/root-novel-bot-capable-teaching-code-children> Acesso em: 26 jun. 2016.

ROBOCAMP. **Lego Mindstorms | Lego WeDo robotics lesson plans**. 2016. Disponível em: <https://www.robocamp.eu/> Acesso em: 26 jun. 2016.

SETIC. **Cresce frequência de uso da Internet por crianças e adolescentes, aponta Cetic.br**. 2014. Disponível em: <http://www.cetic.br/noticia/cresce-frequencia-de-uso-da-internet-por-criancas-e-adolescentes-aponta-cetic-br/> Acesso em 27 jun. 2016.

BAYLOR, C.. **Woodworking with Medium-Density Fiberboard (MDF).** 2015. Disponível em: <http://woodworking.about.com/od/typesofwood/p/Woodworking-With-Medium-Density-Fiberboard-Mdf.htm> Acesso em 4 jul. 2016.

MAKEBLOCK, 2016. **Open-source Arduino Robot Building Platform**. 2016. Disponível em: <http://www.makeblock.cc> Acesso em: 4 jul. 2016.

ALPHAONELABS. **SparkL Motion**. 2010. Disponível em: <http://blog.alphaonelabs.com/post/74552956882/sparkl-motion> Acesso em 5 jul. 2016.

MEDIAMIT. **MIT Media Lab**. 2016. Disponível em: <http://www.media.mit.edu/> Acesso em: 5 jul. 2016.

NOGUEIRA, F. **Apenas 4% das escolas públicas têm computador em classe, diz pesquisa**. 2011. Disponível em: <http://g1.globo.com/educacao/noticia/2011/08/apenas-4-das-escolas-publicas-tem-computador-em-classe-diz-pesquisa.html> Acesso em: 06 mai. 2016.

FREIRE, Karine Xavier. **Design Instrucional: Aplicabilidade dos desenhos pedagógicos na EAD on-line**. Brasília: Secretaria de Estado de Educação do Distrito Federal, 2009. Disponível em: <http://www.abed.org.br/congresso2009/CD/trabalhos/1352009130007.pdf>. Acesso em: 08 jun. 2016.

DIOGOMNZ. **Sobre a falta de engenheiros e profissionais de TI no mercado.** 2015. Disponível em: <https://conhecimentoeconomico.wordpress.com/2015/07/12/sobre-a-falta-de-engenheiros-e-profissionais-de-ti-no-mercado/> Acesso em: 6 mai. 2016.

MARUIAMA, Willian. **Design Instrucional: O que é e como ele pode revolucionar sua empresa**. Disponível em: <http://blog.tuttolabs.com.br/design-instrucional/>. Acesso em: 15 jun. 2016.

KOKAY, E. **Computação cognitiva: sete maneiras de mudar, de vez, a vida das pessoas**. 2015. Disponível em: <http://epoca.globo.com/vida/noticia/2015/06/sete-maneiras-de-computacao-cognitiva-mudar-vida-das-pessoas.html> Acesso em: 7 jul. 2016.

HERINGER, V. **Conheça alguns projetos que incentivam o ensino de programação para crianças e adolescentes**. 2015. Disponível em: <https://www.institutoclaro.org.br/blog/conheca-alguns-projetos-que-incentivam-o-ensino-de-programacao-para-criancas-e-adolescentes-/> Acesso em: 7 jul. 2016.

CODECLUBRASIL. **Code Club Brasil.** 2016. Disponível em: <http://codeclubbrasil.org/> Acesso em 7 jul. 2016.

RICHETTI, J. **Implementação do módulo simulador de mundos virtuais para o SimBot - Simulador de Robôs para Lego NXT**. 2014. Disponível em: <http://www.inf.unioeste.br/gpa/ProjAtuais/Juliano1.pdf> Acesso em: 7 jul. 2016.

BHATTACHARYA, Riya et al. **Dancing Puppets - An Innovative approach to Learning Programming.** Hindi: Iit Kanpur, 2014. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.6955&rep=rep1&type=pdf> Acesso em: 07 jul. 2016.

XANTHOPOYLOS, S. P. **Mito ou verdade: Especialistas esclarecem 11 dúvidas sobre EAD**. 2013. Disponível em: <http://educacao.uol.com.br/noticias/2013/03/20/especialistas-esclarecem-11-mitos-que-rondam-o-ensino-a-distancia.htm> Acesso em: 8 de jul. 2016.

MENEZES, E. B. **Ensino Presencial.** 2001. Disponível em: <http://www.educabrasil.com.br/ensino-presencial/> Acesso em: 8 jul. 2016.

PUERTA, A. A.; AMARAL R. M. **Comparação da educação presencial com a educação á distância através de uma pesquisa aplicada.** 2008. Disponível em: <http://www.sbu.unicamp.br/snbu2008/anais/site/pdfs/2866.pdf> Acesso em: 8 de jul. 2016.

JEREISSATI, J. **Atividades extraclasse - Conceito e objetivos.** 2012. Disponível em: <http://jereissati-jardim.comunidades.net/atividades-extraclasse-conceito-e-objetivos> Acesso em: 8 de jul. 2016.

DIARIOCATARINENSE. **Pesquisa indica hábitos de crianças e pré-adolescentes de oito a 12 anos**. 2012. Disponível em: <http://dc.clicrbs.com.br/sc/estilo-de-vida/noticia/2012/10/pesquisa-indica-habitos-de-criancas-e-pre-adolescentes-de-oito-a-12-anos-3920532.html> Acesso em: 8 de jul. 2016.

INNOVARE. **O entretenimento preferido das crianças**. 2015. Disponível em: <http://www.innovarepesquisa.com.br/blog/o-entretenimento-preferido-das-criancas/> Acesso em: 8 de jul. 2016.

GUIADASEMANA. **Confira os desenhos preferidos pela garotada.** 2011. Disponível em: <http://www.guiadasemana.com.br/filhos/noticia/confira-os-desenhos-preferidos-pela-garotada> Acesso em: 8 de jul. 2016.

GRANDA, A. **Quase metade das casas brasileiras tem computador**. 2014. Disponível em: <http://exame.abril.com.br/brasil/noticias/quase-metade-das-casas-brasileiras-tem-computador> Acesso em: 8 de jul. 2016.

GRESSE VON WANGENHEIM, C.; VON WANGENHEIM, A. **Teaching Game Programming in Family Workshops**. IEEE Computer Society, v.47, p. 84-87, 2014.

BLOOM, B. S.; Engelhart, M. D.; Furst, E. J.; Hill, W. H.; Krathwohl, D. R. (1956). "**Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain**". New York: David McKay Company.

BRANCH, R. M. 2009. "**Instructional Design: The ADDIE Approach**". ISBN 978-0-387-09505-9. Springer Science+Business Media, LLC 2009.

FILATRO, Andrea. "**Design instrucional contextualizado: educação e tecnologia**". São Paulo: SENAC, 2004.

UNICAMP. **OBI 2016 - Geral**. 2016. Disponível em: <http://olimpiada.ic.unicamp.br/> Acesso em: 16 ago. 2016.

GASOLIN. **BlocklyDuino is a web-based visual programming editor for arduino.** 2012. Disponível em: <https://github.com/BlocklyDuino/BlocklyDuino> Acesso em: 20 ago. 2016.

SIRAGUSA, Lou; DIXON, Kathryn C.; DIXON, Robert. **Designing quality e-learning environments in higher education.** Singapore: Ascilite, 2007. Disponível em:

<http://www.ascilite.org/conferences/singapore07/procs/siragusa.pdf>. Acesso em: 01 out. 2016.

PEREIRA, Roger. **“Pokémon Go” chega ao Brasil e já é febre**. Disponível em: <http://paranaportal.uol.com.br/geral/pokemon-go-chega-ao-brasil-e-ja-e-febre/>. Acesso em: 18 out. 2016.

Dick, W., Carey, L. (2000). **The Systematic Design of Instruction.** Glenview, IL: Scott, Foresman, and Company.

APÊNDICE A - ARTIGO PRODUZIDO COM BASE NO TCC

Desenvolvimento de uma unidade instrucional para ensinar computação física extraclasse

Brenda Seara Barcelos[1], Renato Ricardi[1]

[1]Departamento de Informática e Estatística (INE) / Universidade Federal de Santa Catarina

renato.ricardi@grad.ufsc.br, brenda.barcelos@grad.ufsc.br

Abstract. *Computing significantly advances everyday worldwide. However its importance is not seen as a primary factor in the learning of children and young people. Computing is seen as an area of study that must be addressed only in higher education institutions. In the context of this project we propose a model of instructional unit to teach extracurricular computing for children and young people aged 8 to 14 years. The results of the application of instructional unit show that the unit is suitable for meeting the learning objectives and arouses the interest of the children on the computing.*

Resumo*. A computação avança significativamente a cada dia que passa em escala mundial. Contudo sua importância não é vista como um fator primordial no aprendizado de crianças e jovens. A computação é tida como uma área de estudo que deve ser abordada somente em instituições de ensino superior. No contexto deste trabalho é proposto um modelo de unidade instrucional para ensinar computação extraclasse para crianças e jovens com faixa etária de 8 à 14 anos. Os resultados obtidos da aplicação da unidade instrucional demonstram que a unidade é adequada pois cumpre os objetivos de aprendizagem e desperta o interesse as crianças sobre a computação.*

1. Introdução

Uma das grandes necessidades urgentes no século XXI é o aprimoramento do nível de compreensão da computação tanto no campo profissional quanto no campo acadêmico [9]. Para que isso seja realmente aprimorado na sociedade, cada um deve compreender, pelo menos, os princípios básicos de computação [9]. No Brasil e em outros países subdesenvolvidos, o cenário em relação a computação é bem preocupante pois ela é vista

somente quando o indivíduo está cursando algum curso superior na área, o que acaba acarretando em uma alta taxa de falta de conhecimento [3].

Uma solução para contornar esse tipo de problema na área de computação, seria inserir o ensino de computação já no Ensino Fundamental. Deveria ser de responsabilidade das escolas incluir na sua grade curricular matérias vinculadas à computação, contudo não é o que ocorre na maioria das escolas, principalmente no Brasil [6]. No Brasil e no mundo inteiro já existem muitas iniciativas de ensinar a computação focando em algo que o aluno possa deixar de lado aquele conceito de matéria obrigatória e começar a se envolver no mundo real, em um mundo em que as coisas possam acontecer [4].

Existem diversas iniciativas para ensinar computação por meio da computação física, utilizando por exemplo Lego Mindstorm [5] que oferece um conjunto de peças tradicionais, bem conhecidas por todos, juntamente com sensores de toque, de intensidade luminosa e de temperatura, controlados por um sensor programável [5]. Porém, essa iniciativa não é tão acessível a qualquer pessoa que tenha interesse de adquiri-la. O custo para se obter este brinquedo de educação tecnológica é muito alto e nem sempre é uma opção de quem está interessado em aprimorar o ensino [7]. Existem tecnologias que são de baixo custo e podem ser exploradas nos dias de hoje. Um exemplo delas é o Arduino, que é uma plataforma eletrônica de código aberto baseado em hardware e software de fácil utilização [1]. Com o intuito de unir o conceito de computação física, existem também ferramentas para auxiliar na programação como por exemplo, a linguagem de programação Scratch [8], que basicamente é uma linguagem em que você pode programar suas próprias histórias interativas e jogos [8]. O propósito de uma linguagem como Scratch é tornar mais fácil o aprendizado em programação, auxiliando os programadores iniciantes a superar barreiras de sintaxe. Através de ferramentas de programação como Scratch e da utilização de plataformas eletrônicas de hardware como o Arduino, é possível reproduzir funções e ações através de um boneco.

Para as crianças que possuem um conhecimento prévio destas tecnologias ou até mesmo tem a vontade de aprendê-las sem depender de algum curso ou instrutor da área, foi pensado em disponibilizar algo em que possa ser prático e fácil o engajamento nesta área. É esperado que através deste aprendizado, a criança possa, em casa criar novos projetos e realizar diversas outras tarefas. Para atingir o propósito deste aprendizado extraclasse, o objetivo do trabalho é desenvolver uma atividade instrucional com diferentes conteúdos para que desperte interesse em todas as idades e gostos.

2. Contexto da Unidade Instrucional

O propósito desta experiência consiste no auxílio do ensino de conceitos básicos de computação física e programação mantendo o interesse em computação de crianças que já tiveram algum contato com programação e/ou computação física. O público alvo que compõe este trabalho são alunos do Ensino Fundamental com idade entre 8 e 14 anos.

Os objetivos de aprendizagem são basicamente retomar conceitos básicos da computação física e programação de acordo com as diretrizes do currículo de referência da ACM CSTA K-12 com os conteúdos de programação definidos pela OBI. Transmitir ao aluno uma abordagem de resolução de problemas que pode ser desenvolvida num computador envolvendo conceitos de programação tais como estruturas de dados, chamadas de funções, orientação a objetos, recursão e iteração juntamente do conhecimento de computação física, sabendo diferenciar no contexto da unidade instrucional a utilização de hardware e software.

A unidade instrucional inclui a utilização do ambiente de programação Scratch; elaboração e programação de uma atividade do Pikachu; manipulação de componentes eletrônicos para computação física (Arduíno, sensor, atuador, LED, protoboard). Todo o material utilizado no desenvolvimento da unidade instrucional pode ser encontrado neste link: https://drive.google.com/drive/u/0/folders/0B5w23bj7pkKmMURWTjN6b2RmdG8.

3. Avaliação

A unidade instrucional foi aplicada com um aluno do Ensino Fundamental, do Colégio Geração em Florianópolis - Santa Catarina, no ano de 2016 (Fig. 1). Além disso, a unidade instrucional também foi avaliada por um professor que ministra aulas de computação física para alunos do Ensino Fundamental, sendo avaliada no mesmo período.

Figura 1: Aplicação da unidade instrucional

Os dados foram coletados através de questionários tanto para o aluno quanto para o professor após a aplicação e/ou avaliação da unidade instrucional. Foi possível demonstrar que a execução da atividade utilizando Scratch, juntamente do conceito de computação física, foi um sucesso.

Avaliação da unidade instrucional pelo aluno

Em relação ao atingimento dos objetivos de aprendizagem da unidade foi possível verificar que o aluno sabia acompanhar e a analisar a sequência de instruções a ser seguida. De acordo com o questionário aplicado, na pergunta "**O que você aprendeu sobre computação física?**", foi comentado o aprendizado de conceitos como protoboard, jumpers e seus diferentes tipos utilizados na atividade e a possibilidade de programar através do Scratch. Em relação a pergunta "**O que você aprendeu sobre programação?**" foi comentado o aprendizado de conceitos como variáveis, blocos condicionais e criação de blocos para a manipulação de funções.

A partir da percepção em relação ao grau de facilidade de aprendizagem da unidade passada pelo aluno, foi notado que em alguns casos não era possível entender totalmente o conteúdo textual dos tutoriais. Porém após a visualização das imagens indicando o que era para ser o feito, o mesmo conseguia se desenvolver sozinho. Além disso, analisando as respostas no questionário do aluno, ficou evidente de que ele não encontrou sérias dificuldades na elaboração da atividade. Ele considerou a atividade em um nível médio e mesmo assim considerou o seu aprendizado tanto na parte de programação quanto na parte de computação bastante efetivo. Durante a elaboração dos desafios, o aluno mencionou que se

sentiu um pouco perdido em relação à lógica condicional do comando "se, então" pois era difícil de entender.

Durante a aplicação da atividade, com o decorrer do tempo, foi possível perceber que o aluno se sentia cada confiante em realizar as etapas da atividade. Também observou-se que quando o aluno era submetido a manipular alguns componentes que ainda não conhecia, mostrava estar um pouco desconfortável e com dúvidas de como fazê-lo. No questionário foi possível avaliar que o aluno gostou bastante de ter feito a atividade e que gostaria de dar continuidade na realização de mais tarefas de programação e computação física.

O aluno também foi avaliado em relação a apontar pontos positivos e negativos da atividade. Como ponto positivo foi ressaltado um conteúdo completo disponibilizado para a parte de programação. O aluno informou que acredita que ao se basear pelo material disponibilizado poderá aprender melhor os conceitos de programação. Já como ponto negativo a questão da manipulação dos jumpers na protoboard. O aluno se sentiu um pouco desconfortável e inseguro em relação a conexão dos jumpers nos furos da protoboard.

Os dados coletados foram classificados em um número quantitativo, variando em uma escala de 1 a 3, onde o valor 1 representa "Pouco", o valor 2 representa "Mais ou Menos" e o valor 3 representa "Muito" (fig. 2).

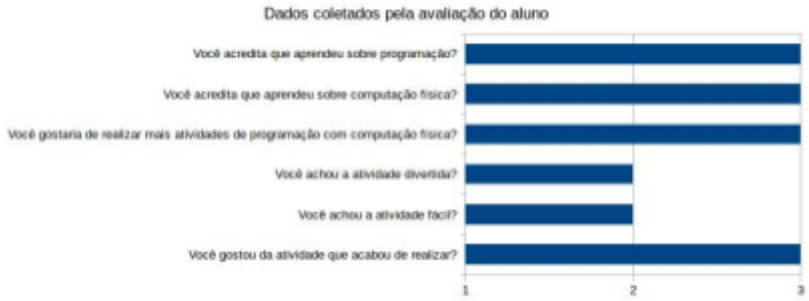


Figura 2: Dados coletados pela avaliação do aluno na unidade instrucional

Avaliação da unidade instrucional pelo professor

Em relação aos objetivos de ensino de computação física, o professor avaliou a atividade em uma abordagem clara e limpa, onde o embasamento didático está bem definido em todas as

etapas da unidade. Ele acredita que o aluno que completar a atividade estará apto a interagir com os elementos básicos e essenciais da computação. Além disso, o plano de ensino e os guias de apresentação do conteúdo mostram-se estruturados e planejados em detalhes. Na parte de avaliação do ensino de programação, avaliou-se a unidade como ideal para introduzir os conceitos básicos de programação. Também foi comentado a boa escolha pela utilização do Scratch, que segundo o mesmo já utiliza há bastante tempo para introduzir conceitos de programação.

No questionário foi feito o questionamento se a avaliação em relação a apresentação e contextualização das etapas de desenvolvimento propostas eram apropriadas para o público alvo. Como resposta foi enfatizado o sequenciamento lógico adequado e os objetivos explícitos de uma maneira geral.

Em relação a elaboração do material didático da atividade, o professor acredita que o tema envolve o interesse dos alunos desta faixa etária. É desenvolvido também um procedimento que explora o tema de forma lúdica e bem planejada.

A atividade desenvolvida para ser aplicada em alunos que já possuem conhecimento prévio sobre o conteúdo foi avaliada como interessante. Utilizando o material desenvolvido, o aluno poderá explorar de uma maneira mais completa as etapas do trabalho, por já conhecer alguns elementos que compõem a execução da atividade.

Como avaliação do tema da atividade para a faixa etária em questão foi avaliado que a atividade é bastante adequada e com um pouco de criatividade pode ser aperfeiçoada ou então até mesmo substituída por algo na mesma vertente sendo executado por outros alunos interessados.

A atividade desenvolvida foi avaliada de forma extremamente eficiente para o quesito baixo custo. O próprio professor ministra um curso, onde os alunos utilizam como material, componentes elaborados com uma impressora 3D. Porém essa solução acaba tendo um custo altíssimo. A solução apresentada de baixo custo trouxe algumas reflexões para o mesmo sobre os objetivos de ensino e a facilidade de sua aplicabilidade.

O professor também foi avaliado em relação a apontar pontos positivos e negativos da atividade. Como pontos positivos foi ressaltado a boa estrutura da atividade em geral e a abordagem efetiva da mesma. Como pontos negativos, não foi citado nada que possa ser considerado. Só foi ressaltado problemas recorrentes para a área de ensino em relação ao conhecimento prévio e facilidade de aprendizado dos alunos.

Os dados coletados (fig. 3), foram classificados em um número quantitativo, variando em uma escala de 1 a 5, onde quanto maior o valor, maior proximidade ele tem com a pergunta em questão.

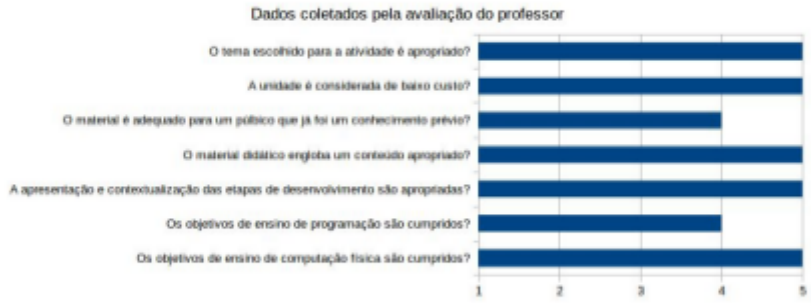


Figura 3: Dados coletados pela avaliação do professor na unidade instrucional

4. Conclusão

O objetivo do presente trabalho foi o desenvolvimento de uma unidade instrucional extraclasse para ensinar computação física e programação para crianças de 8 a 14 anos que já tiveram algum contato com esses assuntos. A unidade foi aplicada em um aluno que se encaixava no público alvo e em um professor que leciona computação física. Foi realizada a avaliação da qualidade da unidade instrucional. De uma maneira em geral os questionamentos feitos sobre a unidade tiveram *feedbacks* positivos. Tanto o professor quanto o aluno apontaram pontos positivos e negativos em relação a aplicação da unidade. Um ponto interessante apontado na avaliação do professor, foi a questão da utilização de materiais de baixo custo para realizar atividades de computação física. Em suas aulas, o professor utiliza recursos que são de altíssimo custo e após a avaliação da unidade instrucional pensa em alterar suas aulas. A solução apresentada pela atividade, trouxe ao professor reflexões sobre os objetivos de ensino e a facilidade na sua aplicabilidade.

Como resultado do presente trabalho, é disponibilizado uma unidade instrucional para ensinar computação física e programação. Essa unidade pode ser aplicada em crianças de 8 a 14 anos preferencialmente com experiência prévia em computação, mas isso não impossibilita que inexperientes consigam realizá-la. O objetivo da unidade instrucional é reforçar ao aluno

conceitos básicos da computação física e programação, mantendo o interesse e a prática de computação sem ter que sair de casa.

Também é resultado do presente trabalho o modelo de design instrucional utilizado. Após terem sido estudados os modelos ADDIE e IDOL, foi desenvolvido um modelo híbrido que faz uso das fases do ADDIE juntamente de algumas dimensões do IDOL. As dimensões trabalhadas possuem como foco o ensino à distância e na ausência de comunicação com algum professor. Esse modelo pode ser utilizado em quaisquer outras unidades instrucionais, inclusive de áreas distintas de computação.

Com esses feedbacks pode-se concluir que a unidade instrucional pode atingir o seu objetivo educacional. Contribuindo, dessa forma com o ensino de computação por meio de uma experiência agradável, motivadora e satisfatória.

5. Referências

[1] Arduino, Introduction: What is Arduino? Disponível: http://www.arduino.cc/en/Guide/Introduction Acesso: abr. 2016.

[2] Bhattacharya, Riya et al. Dancing Puppets - An Innovative approach to Learning Programming. Hindi: Iit Kanpur, 2014.

[3] DiogoMnz. Sobre a falta de engenheiros e profissionais de TI no mercado. 2015. Disponível: https://conhecimentoeconomico.wordpress.com/2015/07/12/sobre-a-falta-de-engenheiros-e-profissionais-de-ti-no-mercado/> Acesso: mai. 2016.

[4] Heringer, V. Conheça alguns projetos que incentivam o ensino de programação para crianças e adolescentes. 2015. Disponível: https://www.institutoclaro.org.br/blog/conheca-alguns-projetos-que-incentivam-o-ensino-de-programacao-para-criancas-e-adolescentes-/ Acesso: jul. 2016.

[5] Lego. LEGO United States - Home of the toy building brick. Disponível: http://www.lego.com Acesso: mai. 2016.

[6] Nogueira, F. Apenas 4% das escolas públicas têm computador em classe, diz pesquisa. 2011. Disponível: http://g1.globo.com/educacao/noticia/2011/08/apenas-4-das-escolas-publicas-tem-computador-em-classe-diz-pesquisa.html Acesso: mai. 2016.

[7] Richetti, J. Implementação do módulo simulador de mundos virtuais para o SimBot - Simulador de Robôs para Lego NXT. 2014. Disponível: http://www.inf.unioeste.br/gpa/ProjAtuais/Juliano1.pdf Acesso: jul. 2016.

[8] Scratch. Scratch - Image, Program and Share. 2016. Disponível: https://scratch.mit.edu/ Acesso: mai. 2016.

[9] The CSTA Standards Task Force. CSTA K–12 Computer Science Standards – Revised 2011,ACM, New York/USA, 2011.

APÊNDICE B - MATERIAIS PARA A UNIDADE INSTRUCIONAL

Guia 1 – Aula 1 – Introdução

1

Introdução

Sobre o curso

O nosso curso vai te ajudar a entender melhor o que tem por trás da telinha do seu computador e relembrar alguns conceitos.

- Como computadores funcionam?
- O que é um programa de computador?
- Hardware? Software? O que é isso?

Vamos descobrir!

Sobre a unidade instrucional

O objetivo do curso é aprender conceitos de computação e conseguir solucionar problemas escrevendo programas de computador.

Mas vamos com calma, o curso está dividido em 4 aulas, com 4 tutoriais e atividades para você aprender tudinho.

E ao final de tudo vamos desenvolver um programa para capturar um pokemon.

Vamos nessa?

Apresentação das aulas

Aula 1
Introdução ao que vamos aprender

Computação, Hardware, Software

Manual 1

Aula 2
Vamos aprender a programar?

Estruturas de repetição, variáveis, condicionais e etc

Manual 2

Manual 3

Aula 3
Vamos aprender o que é computação física?

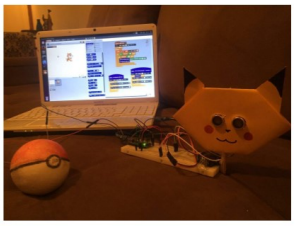
Arduino, sensores, atuadores, led's e protoboard

Manual 4

Aula 4
Hora de desenvolver um programa para capturar esse pokemon!

O que vou desenvolver a final de contas?

Um sistema físico automatizado para capturar um Pikachu feito por você!

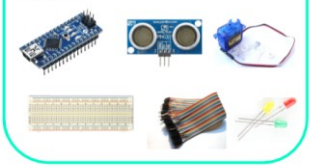


O que vou precisar?

1. Um computador pessoal com sistema operacional Windows com acesso à internet.

2. Folha A4 amarela, fita adesiva, bolinha de isopor, estilete e palito de churrasco.

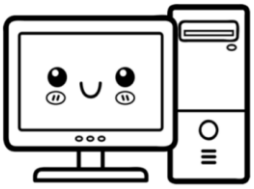
2. Kit de Hardware (Arduino Nano protoboard, LEDs, resistor, atuador, jumper macho-macho, jumper macho-fêmea e sensor de ultrassom)



Primeiro vamos ver alguns conceitos básicos sobre computação e programação

Mas o que é computação?

Computação pode ser definida como a busca de uma solução para um problema, a partir de entradas de informações e pelo uso de algoritmos.

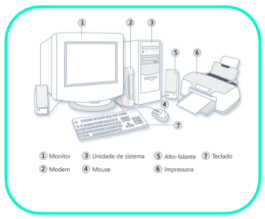


Hardware?

Hardware engloba todas as partes físicas e equipamentos do seu computador.

Você pode ver seu monitor, teclado, mouse e etc.

Os elementos de hardware envolvidos na computação física (que vamos ver mais pra frente) são microcontroladores, sensores e atuadores.



Software?

Software é um conjunto de programas de computador utilizados para conversar com o hardware.

Exemplos de software que você usa são:

- O Word que você usa pra fazer os trabalhos escolares
- O navegador que você utiliza pra acessar as redes sociais

Algoritmo?

Um algoritmo é um procedimento composto por uma sequência de instruções (comandos).

A partir de uma entrada (conjunto de dados) produz uma saída (outro conjunto de dados, que é o resultado da execução do algoritmo).

Programa de computador?

Um programa é um algoritmo escrito em uma linguagem que um computador entende (linguagem de programação).

Um programa de computador é feito de um conjunto de comandos. Os comandos dizem ao computador quais coisas que ele tem de fazer.

Linguagem de programação?

Uma linguagem de programação define comandos que podem ser usados para construir programas.

Cada linguagem tem um conjunto específico de comandos e de regras de construção, mas não é preciso conhecer todos os comandos e todas as regras para começar a programar.

Ambiente de programação?

Um ambiente de programação é um programa de computador que nos permite criar, editar e executar nossos próprios programas de computador.

Neste curso vamos utilizar o ambiente de programação do Scratch (aula 2) e do Arduino (aula 3).

Agora que você está por dentro dos conceitos principais, vamos começar?

Bora pra aula 2!

Guia 2 – Aula 2 – Conceitos de Programação

2

Conceitos de Programação

O que vamos ver?

- Retomando o que é linguagem de programação
- Revisão de conceitos de linguagem de programação
  - Variável
  - Operadores lógicos, relacionais e aritméticos
  - Estrutura de controle
  - Estruturas condicionais
  - Estruturas de repetição
  - Função
- O Scratch

Linguagem de Programação

Como já vimos anteriormente linguagem de programação é a linguagem que nos permite conversar com o computador.

Scratch

Scratch é uma Linguagem de Programação Visual.

Isso significa que, no Scratch, os comandos que dizem ao computador o que ele tem de fazer são figuras que você pode arrastar e juntar como blocos de Lego para fazer os seus programas!

Vamos começar então? Acesse o **Como instalar o Scratch** (pág. 1) no **Tutorial 1** e mão na massa!

Manual 1

Conceitos - Variáveis

Variável é local reservado na memória do computador que pode armazenar qualquer valor.

Uma variável é composta por:

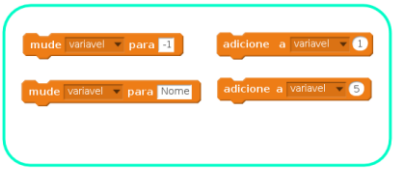
- um nome *(que você escolhe)*
- um tipo *(texto, número)*
- um valor *(o valor que ela vai armazenar)*

Para ver quais blocos de variáveis existem no Scratch, clique na aba "Variáveis" e crie uma variável.

Conceitos - Variáveis

Comando de Atribuição

O comando de atribuição é utilizado para armazenar um valor em uma variável.

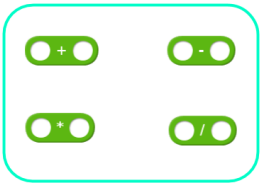


Conceitos - Operadores aritméticos

Operadores aritméticos: servem para somar, subtrair, multiplicar e dividir.

Para ver no Scratch onde estão os operadores clique na aba "Operadores"

\+ - / *

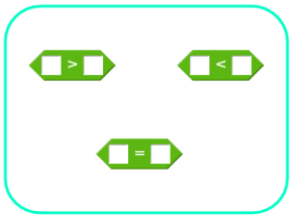


Conceitos - Operadores relacionais

Operadores relacionais: servem para comparar valores numéricos.

Para ver no Scratch onde estão os operadores clique na aba "Operadores"

\> = <

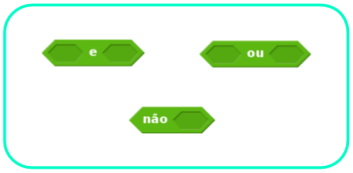


Conceitos - Operadores lógicos

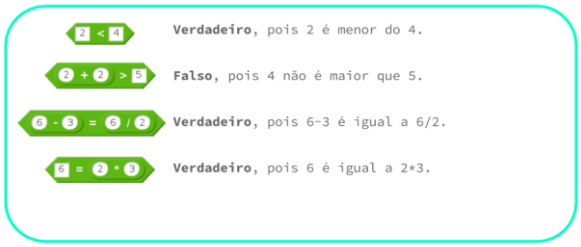
Operadores lógicos: servem para comparar valores booleanos.

Para ver no Scratch onde estão os operadores clique na aba "Operadores"

E OU NÃO



Exemplos - Operadores relacionais e aritméticos



Conceitos - Estruturas de controle

Para uma programa fazer coisas com comandos, como por exemplo, repeti-los, executar determinados fluxos, nós usamos controles.

Para ver quais controles existem no Scratch, clique na aba "Controle".

Conceitos - Estruturas de repetição

A estrutura de controle que faz comandos repetirem são as **estruturas de repetição.**

O comando de repetição serve para executar repetidamente, um bloco de comandos uma quantidade controlada de vezes.

Para ver quais estruturas de repetição existem no Scratch, clique na aba "Controle"

Exemplos - Estruturas de repetição

O que acontece com o código?

Por padrão toda variável que é um número é iniciada com o valor 0.

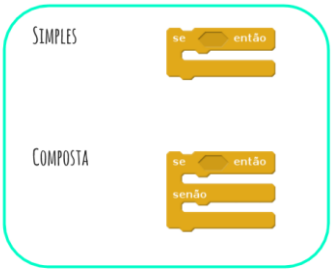
O código adiciona o valor 1 três vezes na variável.

No final o valor mostrado é igual a 3.

Conceitos - Estruturas condicionais

Estrutura condicional: serve para executar blocos diferentes de comandos, dependendo do resultado de uma expressão lógica (uma expressão cujo resultado é Verdadeiro ou Falso).

Para ver quais estruturas condicionais existem no Scratch, clique na aba "Controle".



Exemplos - Estruturas Condicionais

O que acontece com o código?

Primeiramente é atrelado o valor 0 na variável.

É feita uma verificação se 4×2 que tem como resultado o valor 8 é maior do 10.

Esta sentença é falsa, então é adicionado o valor 2 na variável.

No final o valor mostrado é igual a 2.

Conceitos - Função

Função: serve para que um conjunto de comandos que são utilizados muitas vezes no código, sejam representados em um só comando.

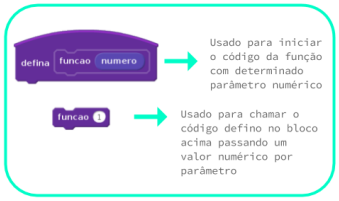
Vantagem: Não é necessário copiar o código todas as vezes que precisar executar determinada operação, além de também o código mais legível e intuitivo.

Para ver quais blocos de variáveis existem no Scratch, clique na aba "Mais Blocos" e crie uma bloco.

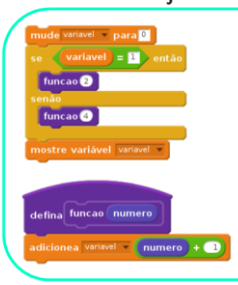
Conceitos - Função Com Parâmetro

Parâmetro: é um valor proveniente de uma variável ou de uma expressão, que pode ser passado para uma função. A função utiliza os valores atribuídos aos parâmetros para alterar o seu comportamento em determinados casos.

Para ver quais blocos de variáveis existem no Scratch, clique na aba "Mais Blocos" e crie uma bloco.



Exemplos - Função



O que acontece com o código?

Primeiramente é atrelado o valor 0 na variável.

É feita a verificação se a variável tem o valor igual a 1.

Esta condição é falsa, então ele executa a função, passando o valor de parâmetro igual 4.

Dentro da função, a variável recebe a adição do valor 1 mais valor do parâmetro passado, que no caso é 4.

No final o valor mostrado é igual a 5.

Atividade

Vamos programar?

Esta atividade tem o intuito de fazer você começar a dar os primeiros passos com Scratch.

Ela é bastante simples e servirá para você se acostumar em como mexer nesta interface tão amigável. O nosso programa fará com o que o gato do Scratch ande alguns passos e logo após isso apareça e desapareça em sua tela misteriosamente.

Vamos começar então? Acesse o **Aprendendo Scratch** (pág. 2) no **Tutorial 1** e mão na massa!

MANUAL 1

Agora que você já sabe programar, vamos saber do que se trata computação física!

Bora pra aula 3!

Guia 3 – Aula 3 – Conceitos de Computação Física

3

Conceitos de Computação Física

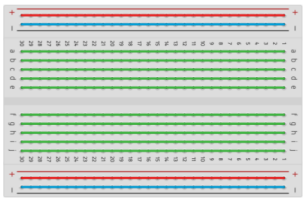
O que vamos ver?

- O que é computação física?
- Protoboard
- Jumper
- Arduino
- Atuador
- Sensor
- Resistor
- LED
- Instalação e configuração do ambiente

O que é computação física?

Computação física compõe tudo que permita construir equipamentos digitais de computação que interagem à realidade que nos rodeia através da utilização de software e hardware.

Protoboard



Protoboard é uma placa com ligações elétricas e com furos onde componentes podem ser conectados mais facilmente.

A protoboard é composta da **área central** e de seus pólos **positivos** e **negativos**.

Protoboard - tecnicamente...

A **área central** são furos que estão todos ligados entre si. Caso algum componente seja ligado em um furo, todos os furos da mesma coluna estarão ligados juntos.

Os pólos **positivos** e **negativos** são barramentos de energia. Nestas colunas todos os furos também estão ligados juntos. Esses barramentos servem para ligar componentes do Arduino que trabalham com energia positiva e negativa, como por exemplo o pino GND e 5V. Desta forma é possível expandir essas conexões para um número mais considerável.

Jumpers

Jumpers são ligações móveis utilizadas entre dois pontos dentro de um circuito eletrônico. São fios metálicos responsáveis pela condução de eletricidade em Protoboards.

Jumpers Macho-Macho

As duas extremidades possuem pontas de metais

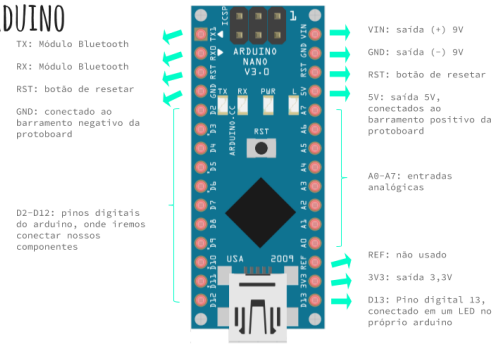
Uma extremidade é de metal e a outra é uma extremidade plugável

Jumpers Macho-Fêmea

Arduino

Arduino é uma plataforma de prototipagem eletrônica que funciona como um mini computador. Ele possui praticamente todos os recursos que um computador oferece porém de uma forma reduzida e com menos poder.

Pinos do Arduino



Atuador

Um **atuador**, também conhecido como servomotor, é um motor controlado pelo Arduino e pode ser utilizado para a movimentação.

O servomotor possui de 3 fios:

- GND (Preto ou marrom)
- 5V (Vermelho)
- Pino Digital (Laranja ou amarelo)

Sensor

Um **sensor** é um dispositivo que recebe e responde a um estímulo ou um sinal.

O sensor de ultrassom é composto de 4 pinos:

- VCC: alimentação de 5V
- TRIG: pino de gatilho
- ECHO: pino de eco
- GND: pino para fio terra

Sensor

O funcionamento de um sensor é ilustrado abaixo:

Resistor

Um **resistor** tem a função limitar o fluxo de corrente elétrica que passa por ele. É bastante usado para evitar que certos componentes queimem, como por exemplo um LED.

LED

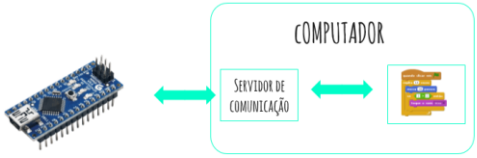
LED é um condutor elétrico que emite luz através de energia elétrica. O LED possui um pólo positivo e outro negativo. O pólo **positivo** é representado pela perna maior, já o pólo **negativo** pela perna menor.

IDE - Ambiente de Programação

Uma **IDE** é um programa de computador que auxilia no desenvolvimento de software de uma forma mais ágil, como já vimos na aula 1.

Com a utilização de uma IDE, é possível escrever pequenos programas que se comunicam com vários tipos de dispositivos como sensores, atuadores, LCDs e outros microcontroladores. Exemplos de IDE: Scratch, Arduino IDE, dentre outros.

Servidor de comunicação



Um **servidor de comunicação** faz a ponte entre o Arduino e o Scratch.

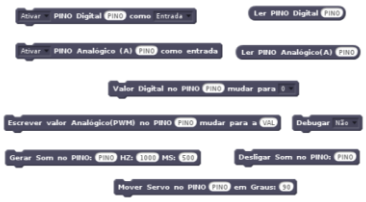
Vamos deixar nosso ambiente prontinho?

Agora que você já possui todo o embasamento teórico para esta atividade, vamos configurar nosso ambiente para podermos ir para o momento mais esperado. Vamos começar então?

Acesse o **Instalação e configuração do ambiente** no **Tutorial 2** e mão na massa!

MANUAL 2

Blocos de comunicação Arduino + Scratch



Um servidor de comunicação conhecido como **s2a_fm** disponibiliza alguns blocos de comunicação para conversar com o Scratch de forma fácil.

Blocos de comunicação Arduino + Scratch

Testando o ambiente Arduino + Scratch

Você lembra aquele programa que utilizamos anteriormente no Arduino IDE chamado Blink? Então, vamos reproduzi-los agora no Scratch. Através da elaboração deste programa, vamos garantir que a comunicação Scratch + Arduino está funcionando perfeitamente.

Acesse o **Testando o seu Arduino com Scratch** no **Tutorial 3** e mão na massa!

MANUAL 3

Agora que você já sabe programar, sabe tudo de computação física e possui o ambiente configurado e rodando, vamos para nossa atividade!

Bora pra aula 4!

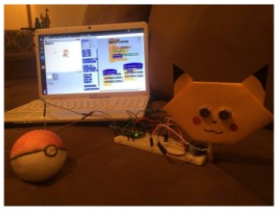
Guia 4 – Aula 4 – Desenvolvimento da Atividade Captura Pokemon

4

Desenvolvimento da Atividade

O que vou desenvolver?

Vamos desenvolver um programa de computador para capturar um Pikachu como na imagem abaixo.



A ATIVIDADE

Esta atividade descreve a montagem de um boneco do Pikachu utilizando Arduino e a linguagem de programação Scratch.

Na construção do Pikachu serão utilizados componentes de hardware que serão capazes de não só comandar ações para o boneco mas também através de entradas sensoriais, executar comandos oriundos do programa construído no Scratch.

MONTAGEM DE UM PIKACHU UTILIZANDO ARDUINO E SCRATCH

A estrutura física do Pikachu será elaborada com a montagem de um origami utilizando folhas de papel. Já a pokebola será elaborada utilizando uma de bola de isopor e alguns outros materiais para uma melhor finalização da mesma.

FUNCIONAMENTO

A atividade vai funcionar da seguinte maneira, a pokebola irá se aproximar do Pikachu e à partir de uma determinada distância, a pokebola emitirá uma luz.

Esta luz poderá variar na quantidade de vezes que piscar, simulando a tentativa de captura do Pikachu.

Caso o Pikachu seja capturado, o mesmo "cairá" no chão. Porém dependendo da força do Pikachu ele poderá resistir e não ser capturado pela pokebola, voltando a se "levantar".

PASSO-A-PASSO

MANUAL 1

PASSO 1
PIKACHU
VAMOS FAZER NOSSO ORIGAMI DE PIKACHU

PASSO 2
POKEBOLA
VAMOS FAZER NOSSA POKEBOLA PISCAR

PASSO 3
MOVIMENTAÇÃO DO PIKACHU
VAMOS POSICIONAR NOSSO PIKACHU E O ATUADOR

PASSO 4
PERCEPÇÃO DO PIKACHU
VAMOS FAZER NOSSO PIKACHU PERCEBER A DISTÂNCIA DA POKEBOLA

PASSO 5
DESENVOLVIMENTO DO PROGRAMA
VAMOS FAZER TUDO FUNCIONAR DIREITINHO

PASSO 6
TESTE
HORA DE TENTAR CAPTURAR O PIKACHU

DESAFIOS

MANUAL 1

DESAFIO1
MELHORANDO O CÓDIGO
VAMOS MELHORAR NOSSO CÓDIGO APRENDENDO NOVOS CONCEITOS

DESAFIO 2
AUMENTAR A DIFICULDADE
VAMOS DIFICULTAR UM POUCO A CAPTURA DO NOSSO PIKACHU

E AÍ GOSTOU?

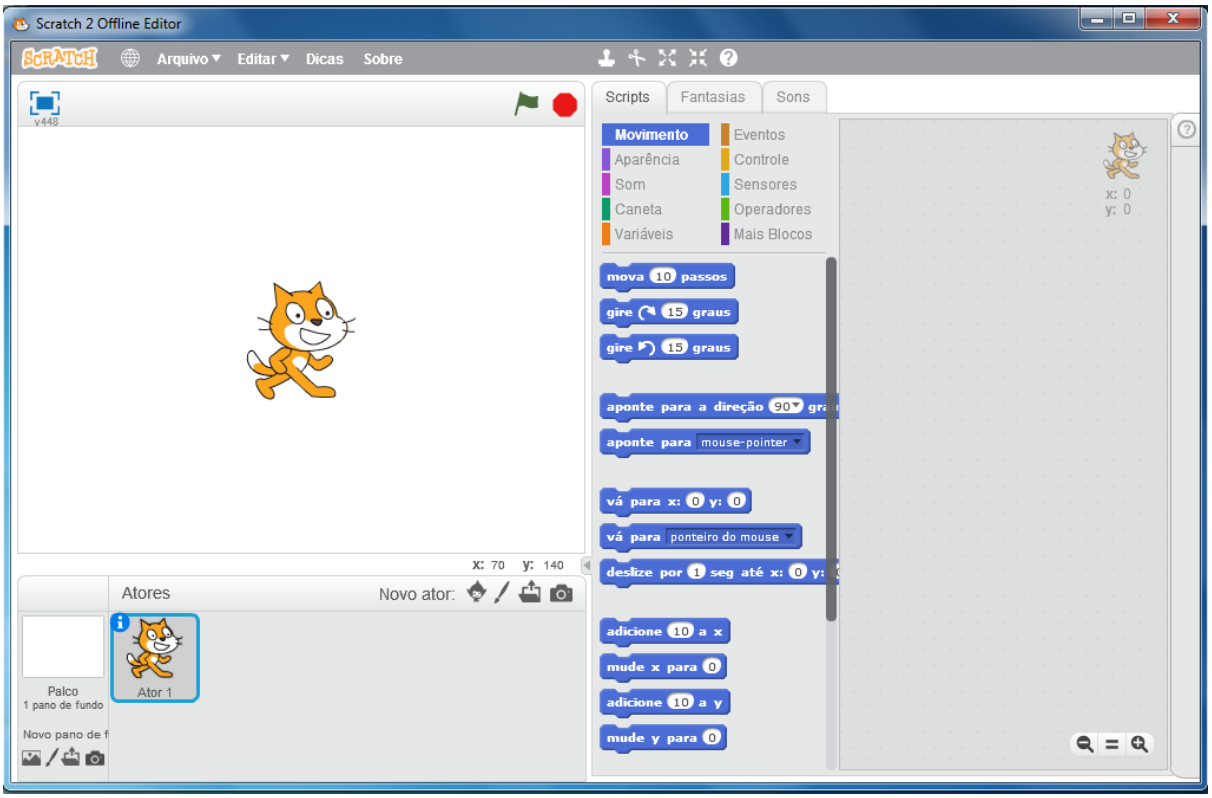
QUESTIONÁRIO FINAL

Manual 1 - Instalação e desenvolvimento do primeiro código no Scratch

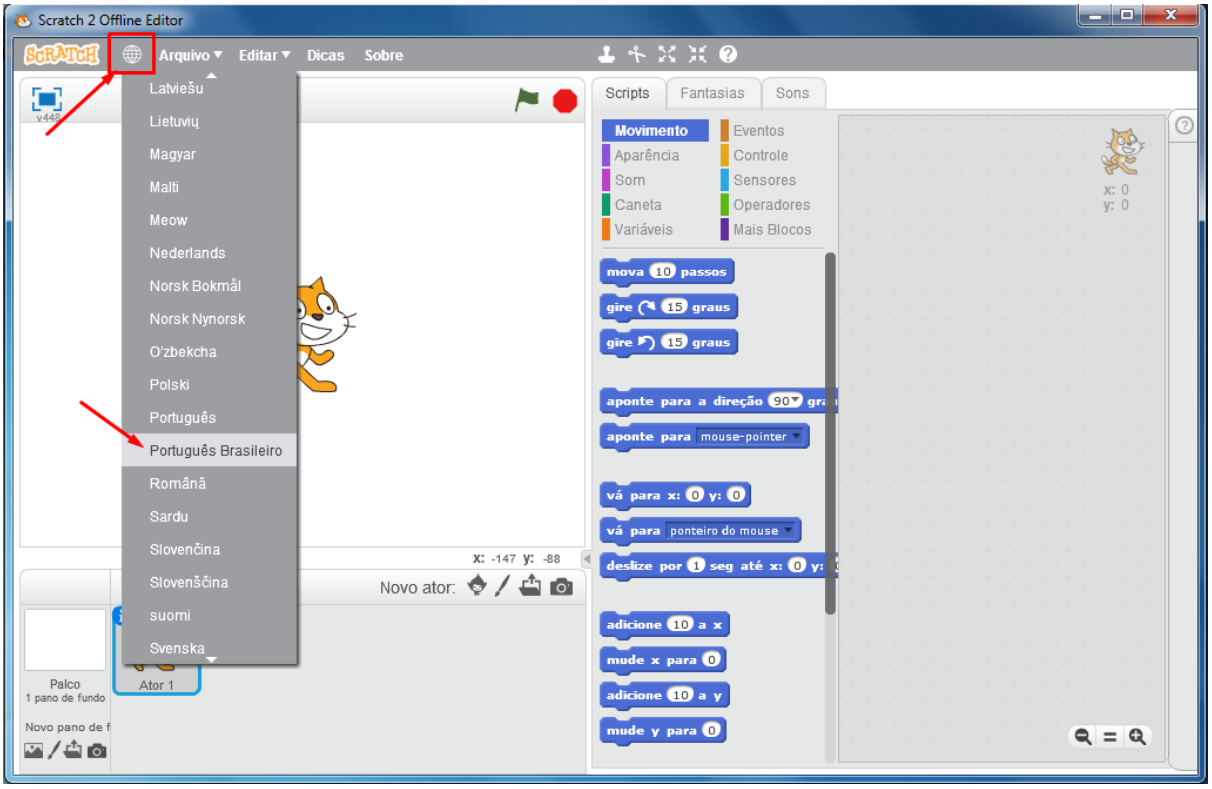
Como instalar o Scratch?

O Scratch pode ser acessado e utilizado através de um navegador porém para nossa atividade é necessário baixá-lo em seu computador. A instalação do Scratch é simples, basta acessar o site do Scratch ou executar o arquivo **Adobe AIR** (faça-o primeiramente) e o arquivo **Scratch** (faça-o após instalar o Adobe Air) que estão na pasta programas. O Scratch Offline Editor foi desenvolvido em Adobe Flash e para ser executado precisa do Adobe Air. Feita a instalação do Scratch Offline Editor será possível acessar sua interface,

conforme ilustrado na imagem abaixo.



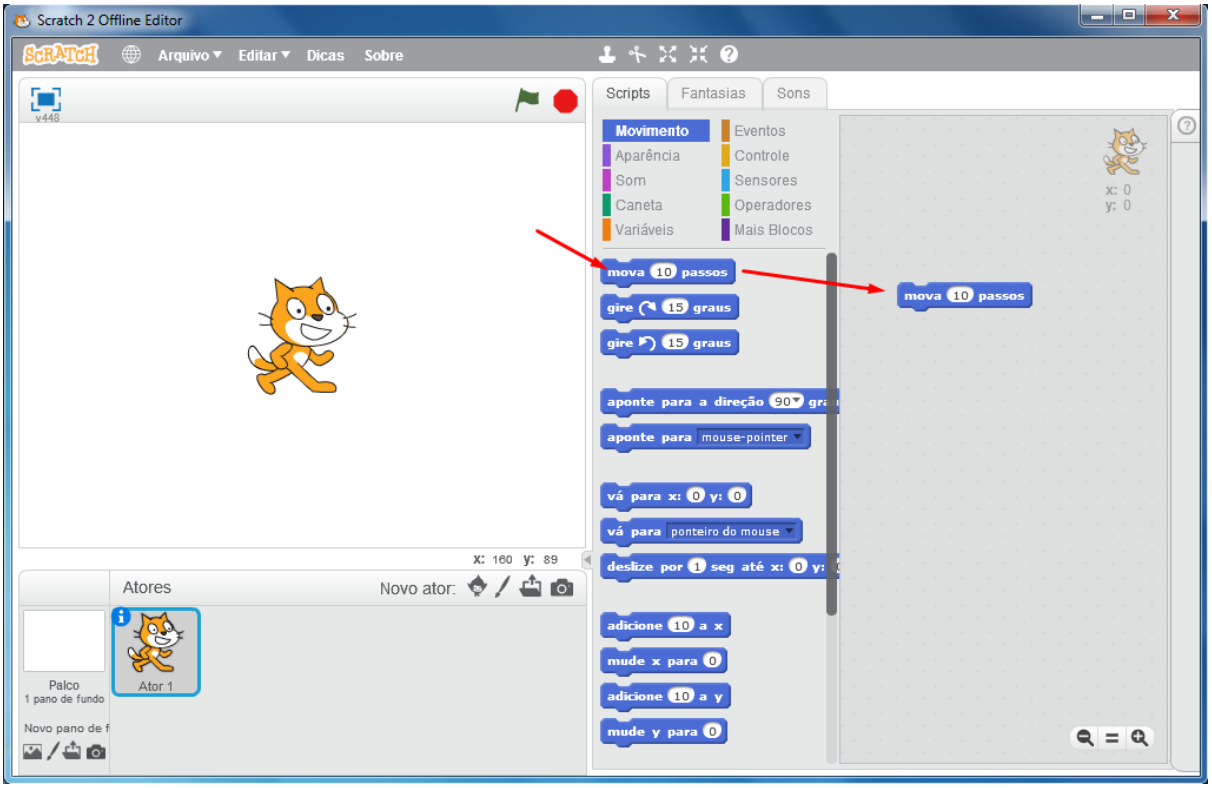
Ao acessar a interface do Scratch é possível mudar seu idioma para a versão em Português ou até mesmo para outro idioma de sua preferência. Para mudar o Scratch para Português basta selecionar o ícone de um globo no menu, conforme mostra a imagem abaixo.



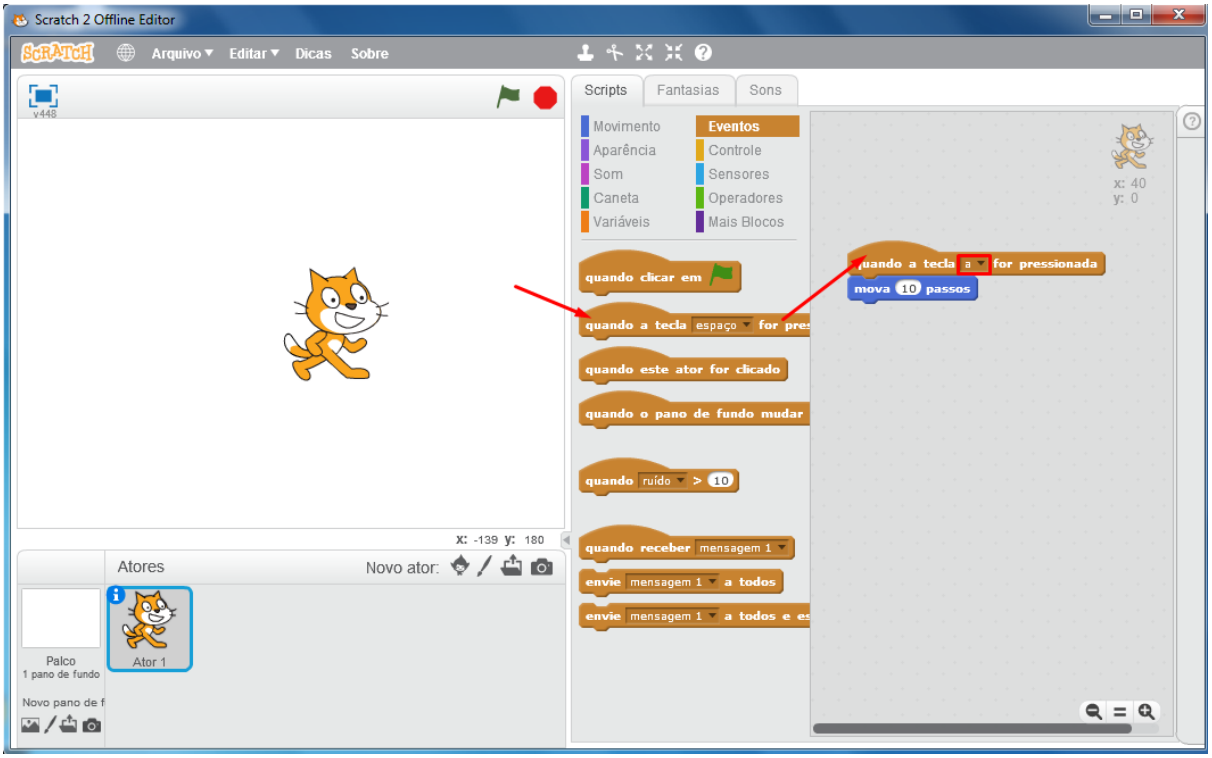
Agora já é possível passar para a próxima etapa e relembrar sobre os funcionamentos do Scratch.

Aprendendo Scratch

O Scratch é uma linguagem de programação visual muito simples de se aprender. Não é necessário conhecimento prévio de outras linguagens de programação e é ideal para quem está começando a engajar neste novo mundo. Antes de iniciarmos nossa atividade, é interessante fazermos uma pequena revisão sobre o funcionamento da linguagem Scratch. Primeiro vamos criar um programa exemplo para podermos lembrar como é o funcionamento do Scratch. Este programa fará com o que o gato do Scratch ande alguns passos, e logo após isso apareça e desapareça em sua tela misteriosamente. Isso só é possível pois você está dando comandos ao gato para que ele realize alguma ação. No Scratch, esses comandos são executados em ordem que colocamos no programa, lembram né? Vamos começar arrastando nosso primeiro bloco para nossa área de construção do programa à direita, exatamente conforme ilustra a imagem abaixo.

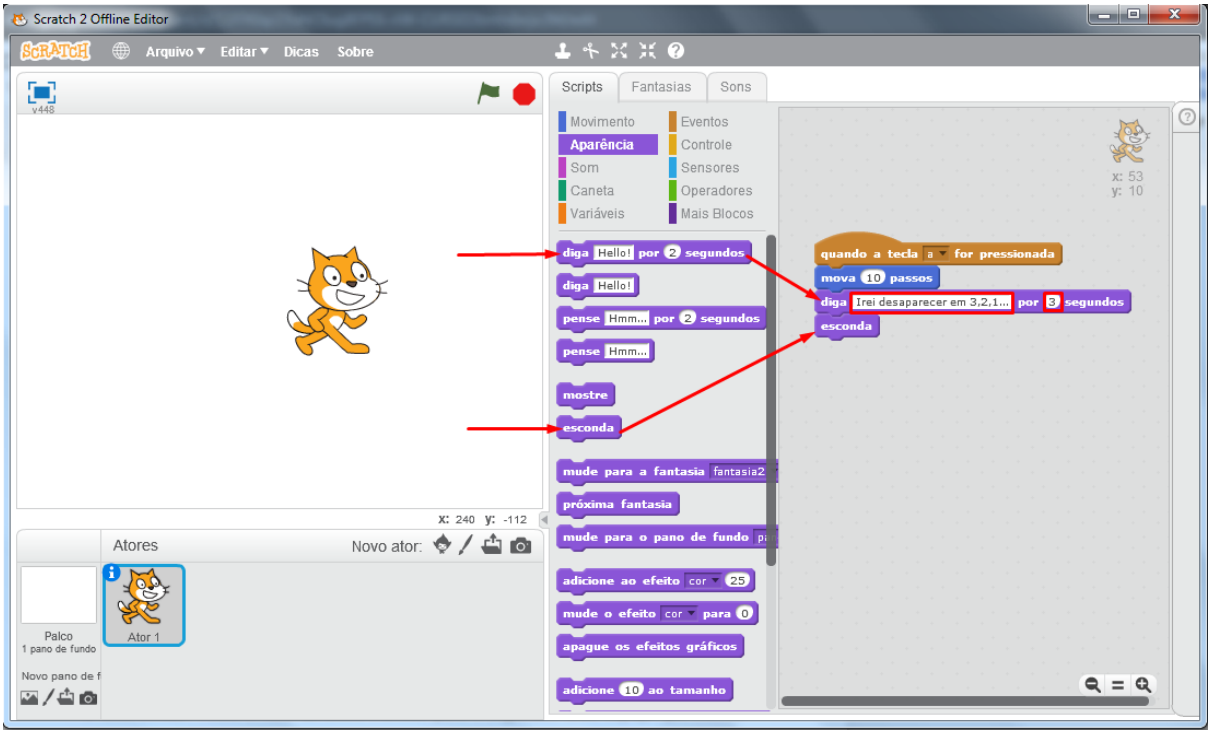


Agora se você clicar em cima deste bloco, veja o acontece. Você viu que o gato andou um pouco? Agora clique repetidas vezes para que o gato ande um pouco mais. Legal né? Pois é, tirando a parte que você vai ter que clicar diversas vezes para fazer o gato andar, tudo ótimo. Porém vamos resolver o seu problema, vá na aba “Eventos” e escolha o bloco indicado na imagem abaixo. Este bloco permite que após determinada tecla for pressionada execute algum código. Vamos escolher para que quando apertarmos a letra ‘a’ no teclado, o gato do Scratch se movimente.

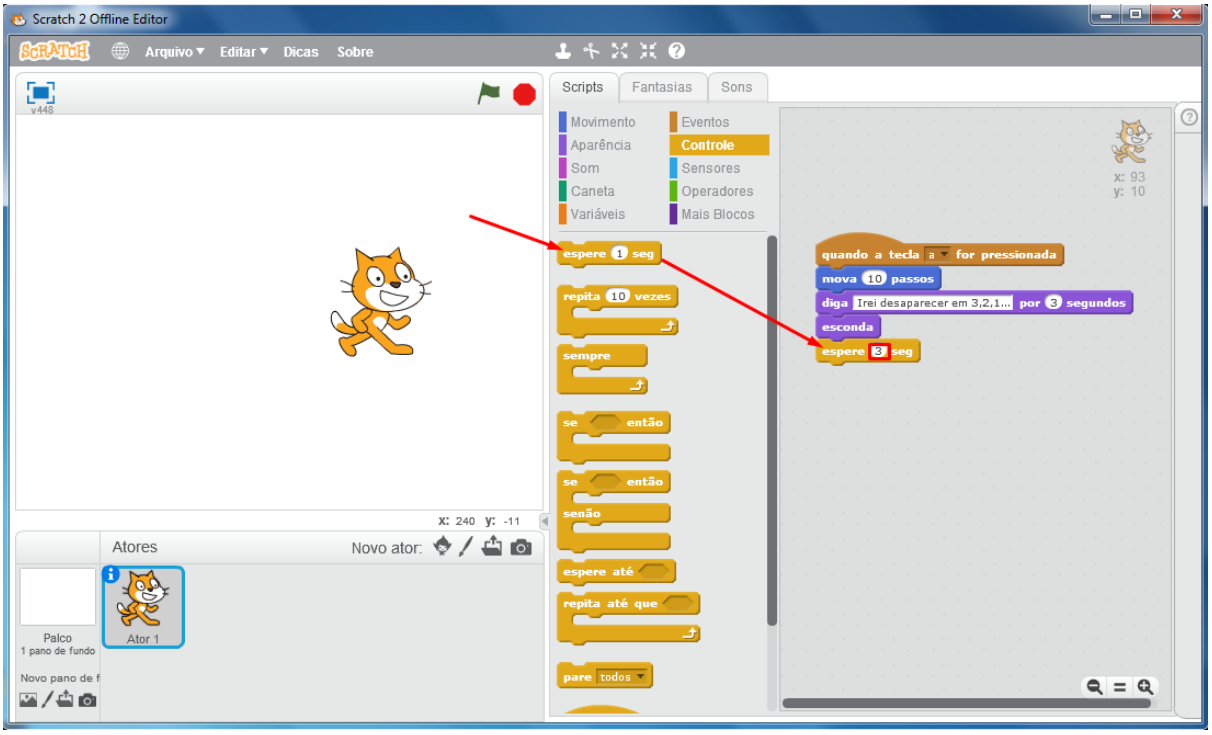


Aperte a letra ‘a’ para ver o que acontece. Bem melhor né? Isso aconteceu pois acabamos de realizar um evento. Eventos servem para que algo inicie no Scratch, ou seja, quando você quiser determinar o início de seu programa você poderá escolher algumas opções disponíveis nesta aba Eventos.

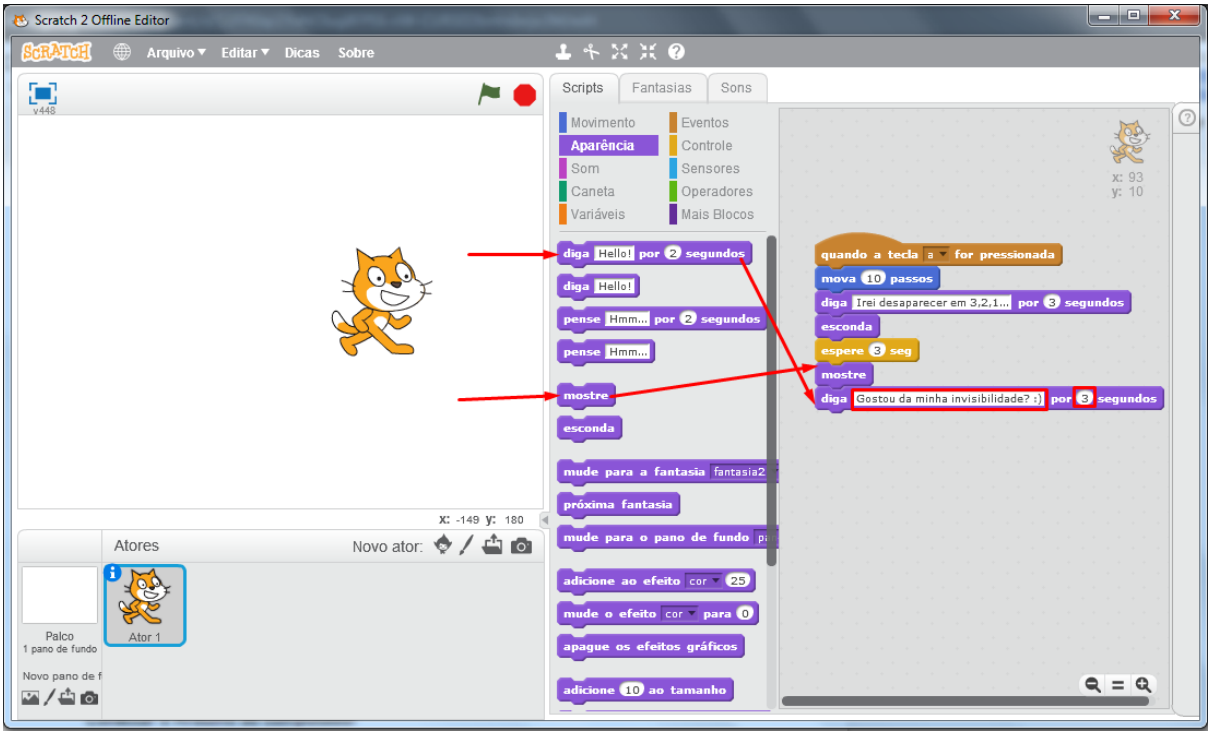
Agora vamos para a parte em que o gato some misteriosamente, que tal? Mas antes disso vamos fazer com que ele nos avise que está desaparecendo. Vá na aba “Aparência” e escolha o bloco indicado na imagem abaixo. Este bloco mostrará uma mensagem por determinados segundos avisando que o gato desaparecerá. Mude a mensagem “Hello” deste bloco para “Irei desaparecer em 3,2,1...” e troque o tempo para 3 segundos e logo em seguida escolha o bloco “Esconda”, que está na mesma aba. A mudança da mensagem e do valor dos segundos é conhecido como parâmetro e serve basicamente para inserir um comando de uma forma que ele aconteça de uma maneira diferente, ou seja, ele irá assumir o valor que você passar para ele.



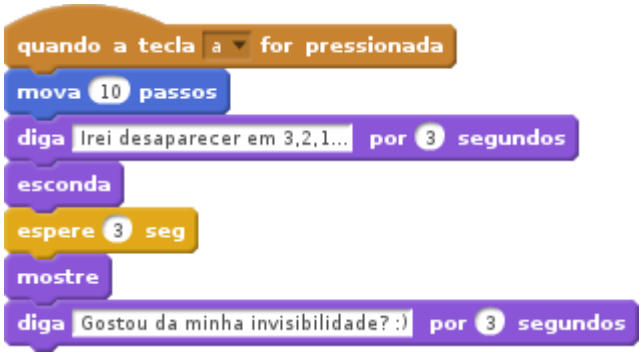
Aperte a tecla ‘a’ e veja o que aconteceu agora. Misteriosamente o gato sumiu, será que ele não volta mais? :(Que tal nós fazermos ele sumir somente por alguns segundos e depois voltar? Para isso vá na aba “Controle” e selecione o bloco indicado na imagem abaixo. Os comandos da aba controle são blocos que usaremos bastante em nossa atividade do Pikachu. Neste momento não vamos entrar tanto em detalhes mas já fica a dica, beleza? Mude o valor do bloco indicado para 3 segundos.



Quase íamos esquecendo de fazê-lo voltar né? Vá novamente na aba "Aparência" e escolha o bloco "Mostre". Para ficar mais interativo vamos adicionar uma outra mensagem no final da execução do nosso programa.



O código final é o mostrado na imagem abaixo, muito simples né?



Agora que relembramos o funcionamento do Scratch, que tal darmos os primeiros passos em nossa atividade? Acho que era o que você mais queria né? Então vamos lá, mão na massa!

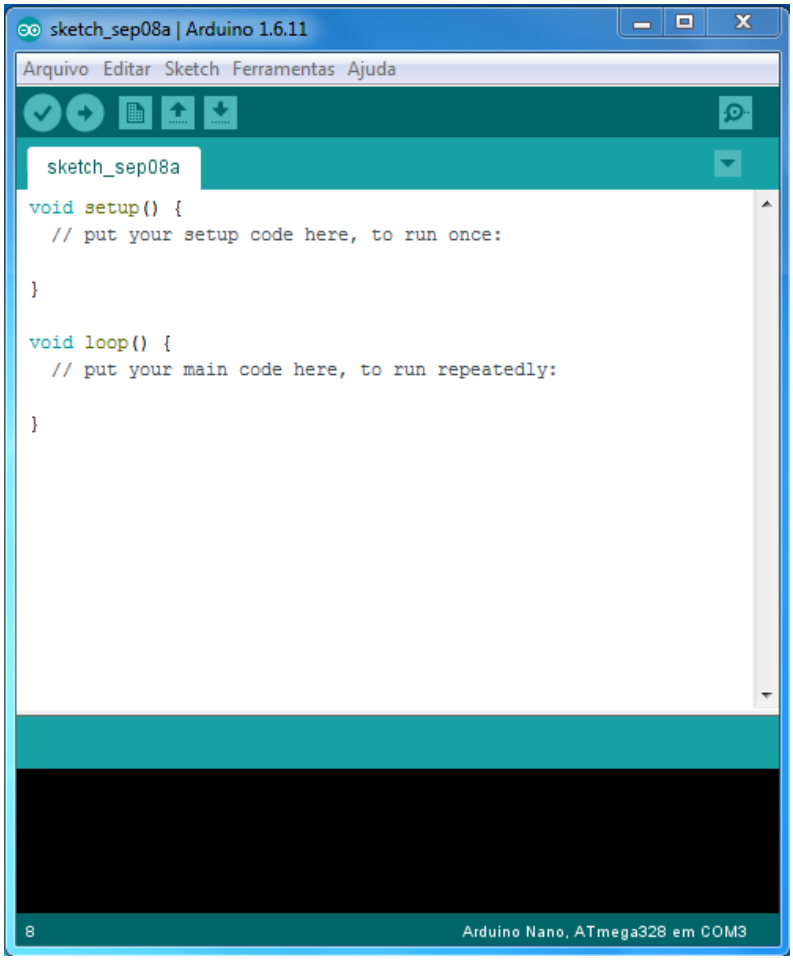
Manual 2 - Instalação e configuração do ambiente

Instalação e configuração do ambiente

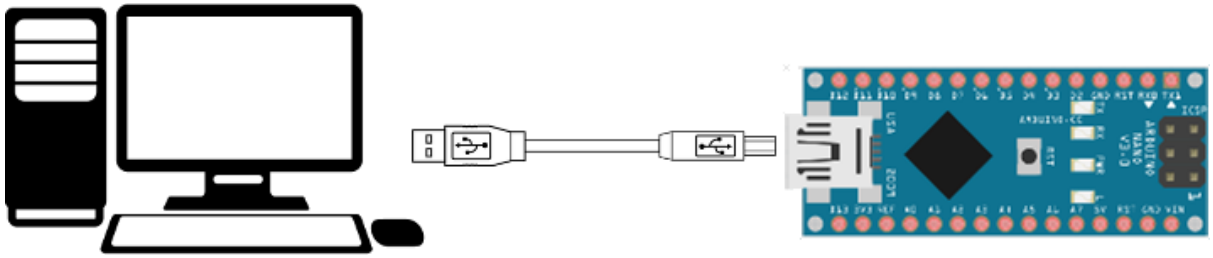
Para quem está iniciando as atividades com Arduino é necessária algumas configurações antes de iniciar alguma coisa. Então basicamente vamos ter que realizar alguns passos antes de garantir total funcionamento do nosso Pikachu. Vamos lá então?

Rodar o servidor de comunicação

Para realizar a comunicação do Scratch com o Arduino é necessária a instalação do **Arduino Software (IDE)**. o Arduino IDE é uma interface que torna mais fácil a escrita de código para que o Arduino possa realizar determinadas tarefas. Este programa pode ser encontrado no site oficial do Arduino, neste link https://www.Arduino.cc/en/Main/Software. Faça o download do programa escolhendo o seu sistema operacional. Após a conclusão do download instale o programa e todos os drivers solicitados durante a instalação do mesmo. Ao executar o programa instalado, uma tela semelhante a mostrada na imagem abaixo será exibida.

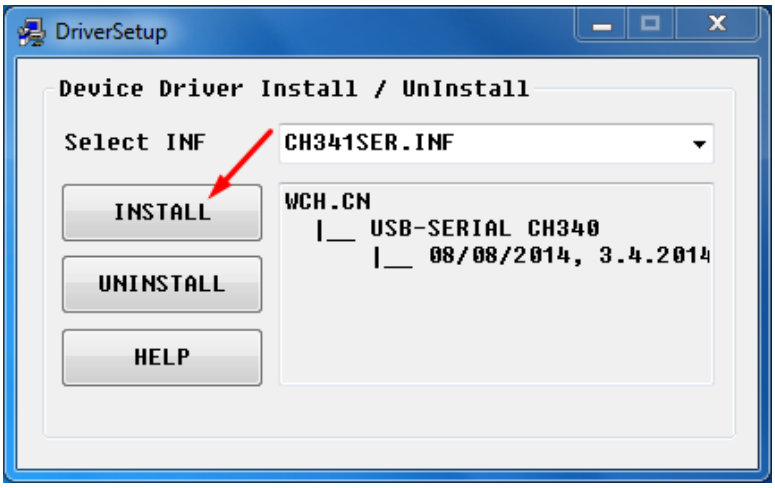


Para o computador reconhecer a porta USB do Arduino é necessário a instalação de um driver específico do Arduino. Primeiro conecte o Arduíno em uma USB de seu computador e depois faça o download para o seu respectivo sistema operacional neste link https://drive.google.com/drive/folders/0B5Ty0xKzazEVQ2tCT09rTUh6eDg.



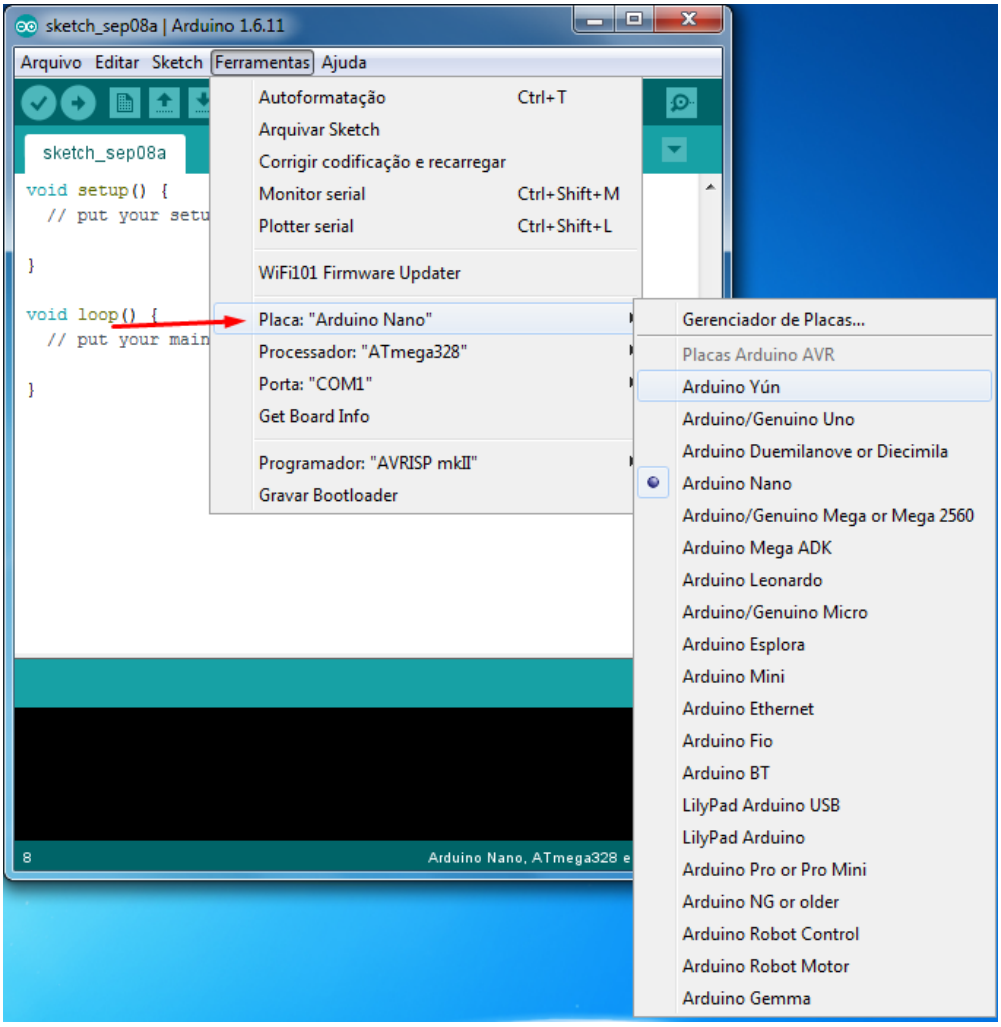
Após o término do download, faça a extração do arquivo baixado, execute o arquivo “Setup.exe” e clique

no botão “Install” conforme mostra imagem abaixo.

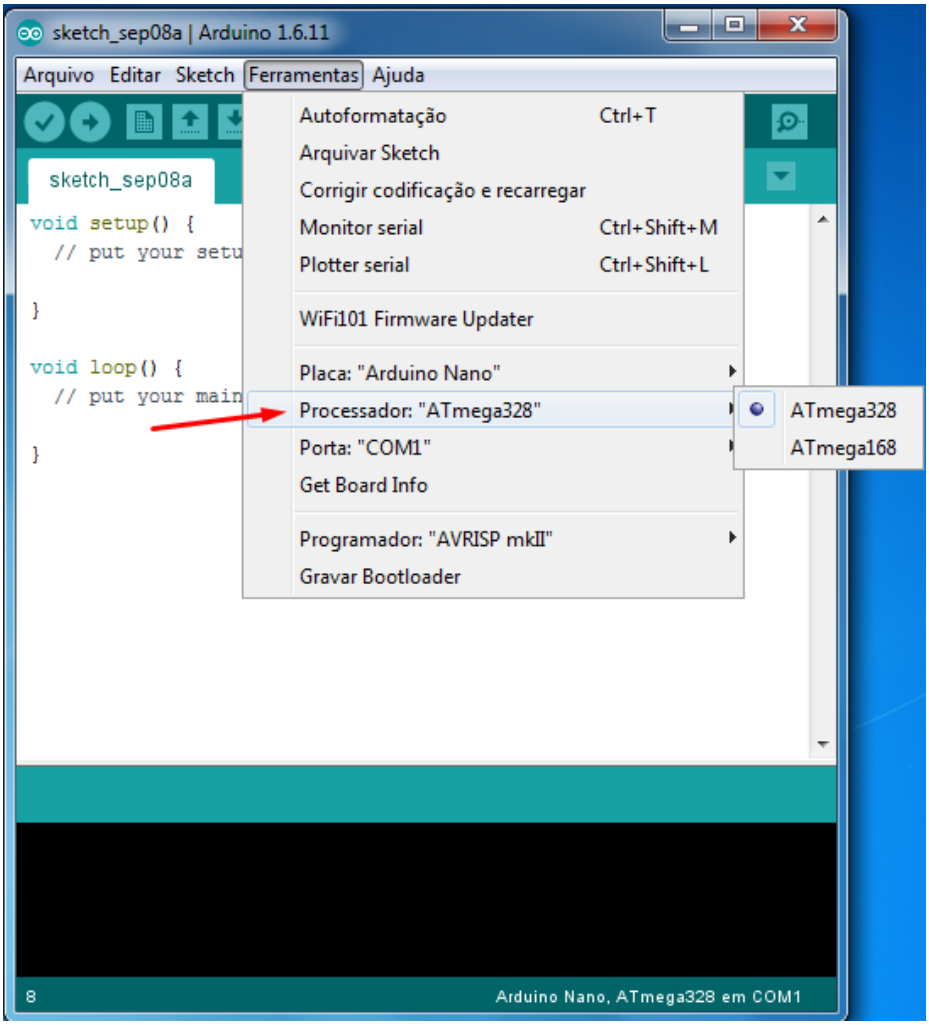


Se tudo correr bem, seu computador já é capaz de identificar a porta USB do Arduino. Agora iremos voltar para interface do Arduino IDE para realizar alguns ajustes necessários com o intuito de fazermos nossas primeiras ações. Esses ajustes são necessários para que o Arduino IDE identifique as configurações corretas de seu modelo de Arduino. Com o Arduino conectado em seu computador, execute os seguintes passos:

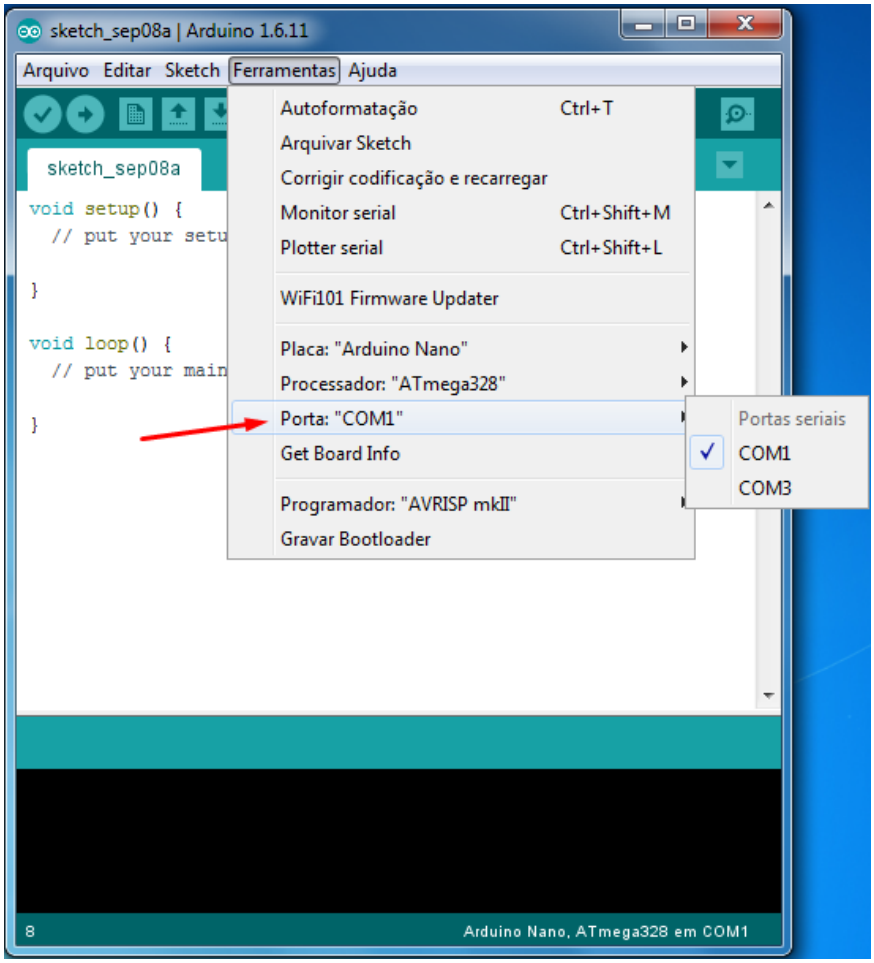
1. Vá em **Ferramentas > Placa** e selecione o modelo do Arduino Nano.



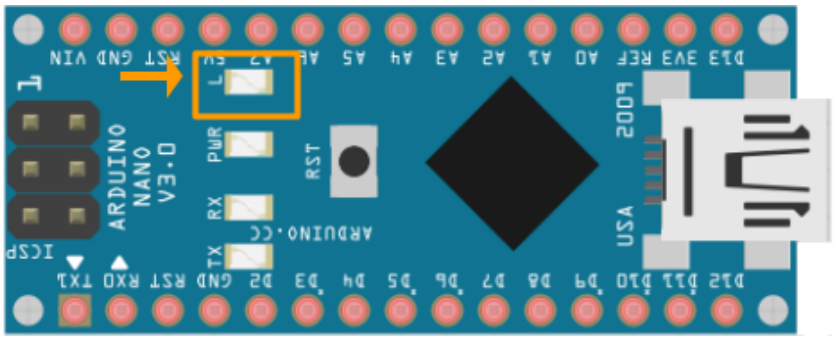
2. Vá em **Ferramentas > Processador** e selecione o microcontrolador ATmega328, certifique-se que este é o modelo que você adquiriu. Para adquirir esta informação veja nas especificações de compra do Arduino Nano ou então próprio microcontrolador esta informação estará disponível.



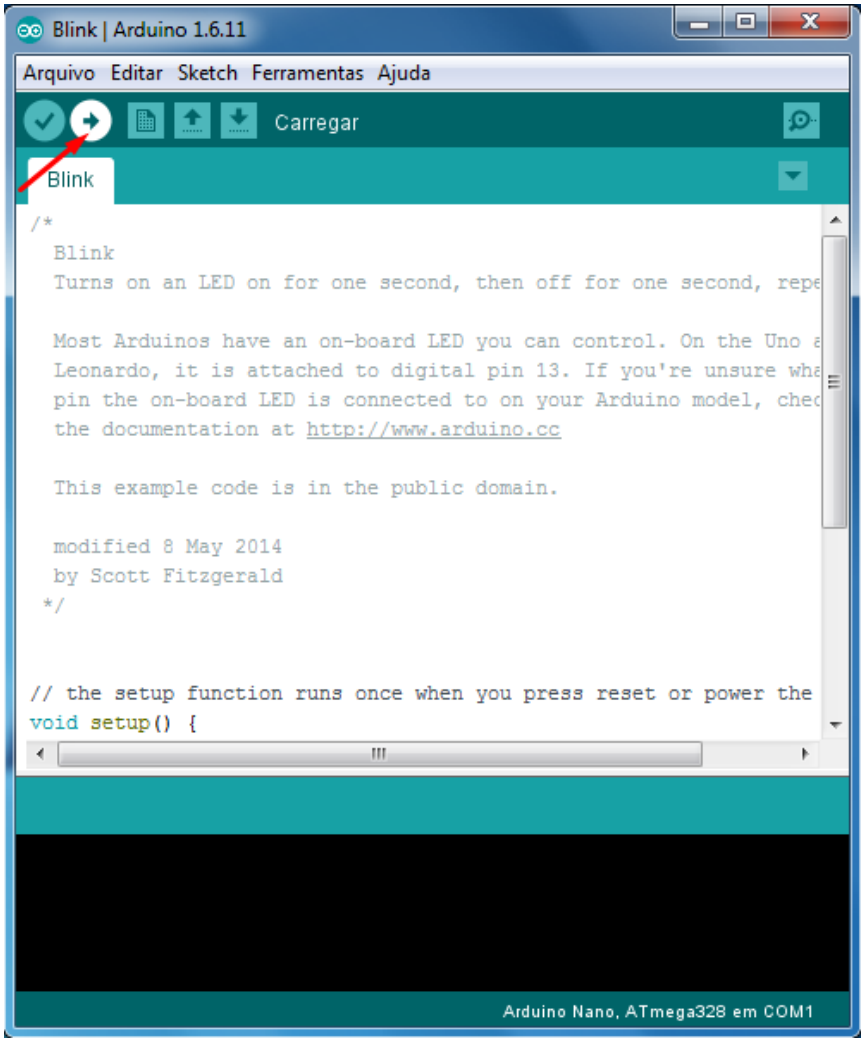
3. Vá em **Ferramentas > Porta** e selecione a porta USB em que seu Arduino Nano está conectado. Para saber qual é a porta correta caso apareça mais de uma disponível, acesse a interface do Arduino IDE sem a USB estar plugada no computador e veja quais estão aparecendo, logo em seguida conecte a USB do Arduino em seu computador e veja qual a porta que apareceu agora.



Agora que já temos esta configuração feita, vamos carregar um programa para que possamos ter certeza que o Arduino está funcionamento normalmente e apto para seguir em frente para nossa atividade. O programa que iremos carregar é chamado de Blink, este programa é bastante familiar para quem está iniciando o desenvolvimento com o Arduino. Este programa possui alguns comandos que mandam o Arduino piscar constantemente um LED (cor laranja) em sua placa, como mostra imagem abaixo.



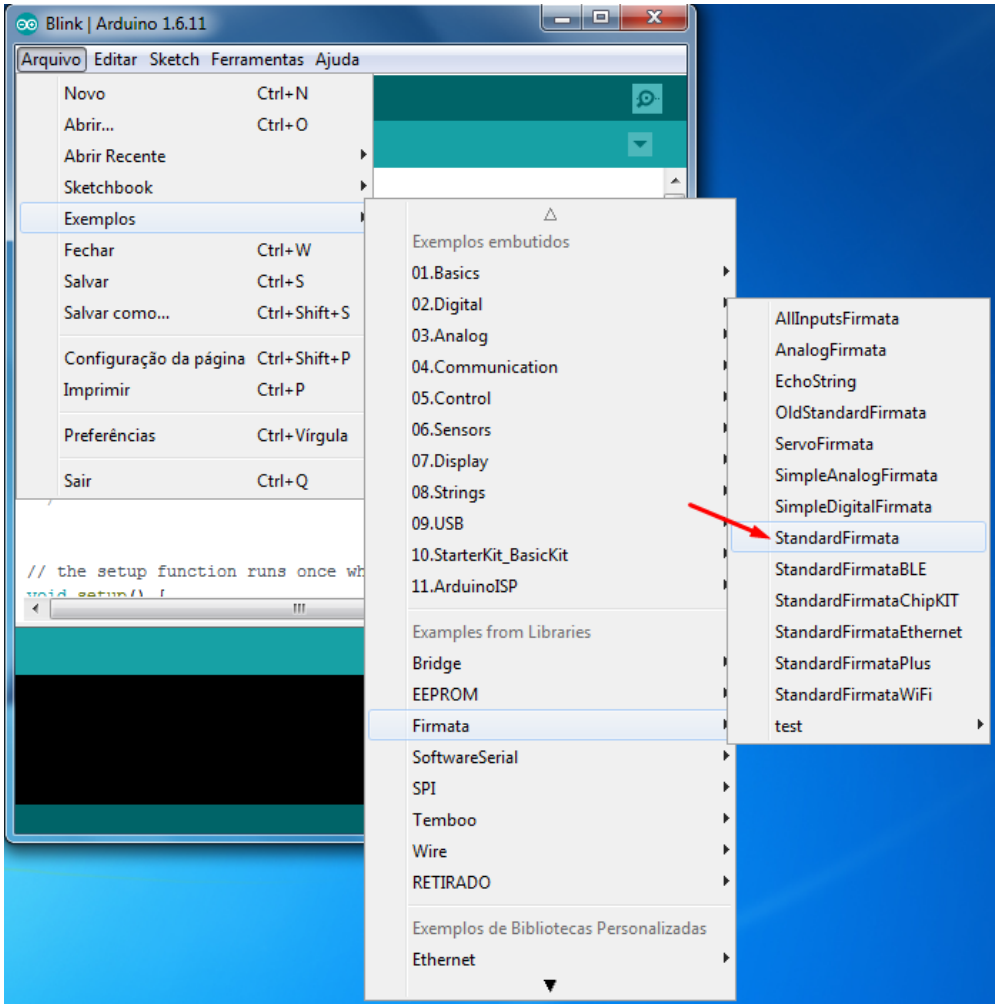
Para abrir o programa você terá que ir em **Arquivo > Exemplos > 01. Basics > Blink**. Se você conseguiu acessar este caminho no menu, deve estar vendo algo parecido com a imagem abaixo. Aperte no botão "Carregar" conforme também indicado na imagem abaixo e veja o que aconteceu.



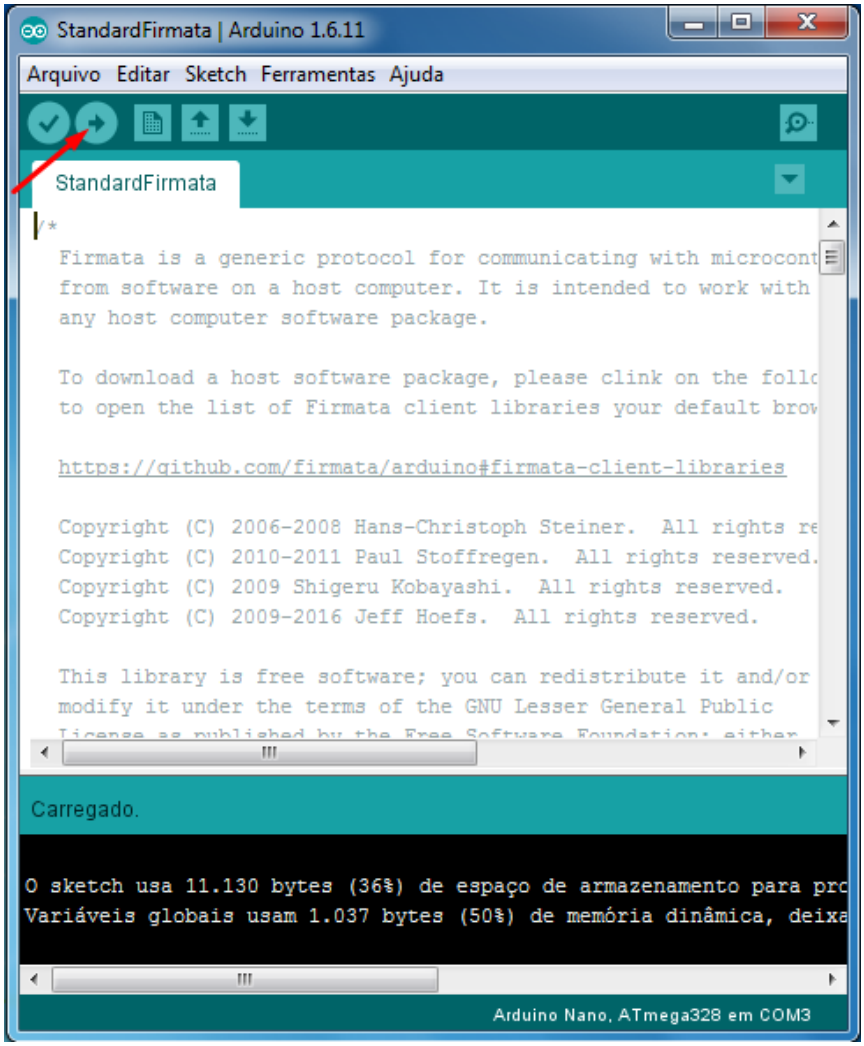
Está piscando uma luz laranja constantemente em seu Arduino, certo? Caso não esteja acontecendo isso, algo foi configurado errado. Visualize a mensagem na parte inferior de cor preta no Arduino IDE para ter uma ideia do que possa ter ocorrido.

A partir de agora já estamos bem avançados na configuração de nosso ambiente para montagem da atividade. Nesta etapa iremos realizar o carregamento do código chamado de Standard Firmata no

Arduino. O Firmata é um protocolo de comunicação para que o Arduino possa dar suporte a alguns recursos disponibilizados por outros componentes. Por exemplo, para fazermos o LED (emissor de luz da pokebola) da nossa atividade funcionar, precisamos ter este protocolo carregado em nosso Arduino. Para carregá-lo você terá que ir em **Arquivo > Exemplos > Firmata > StandardFirmata**, assim como mostra a imagem abaixo.

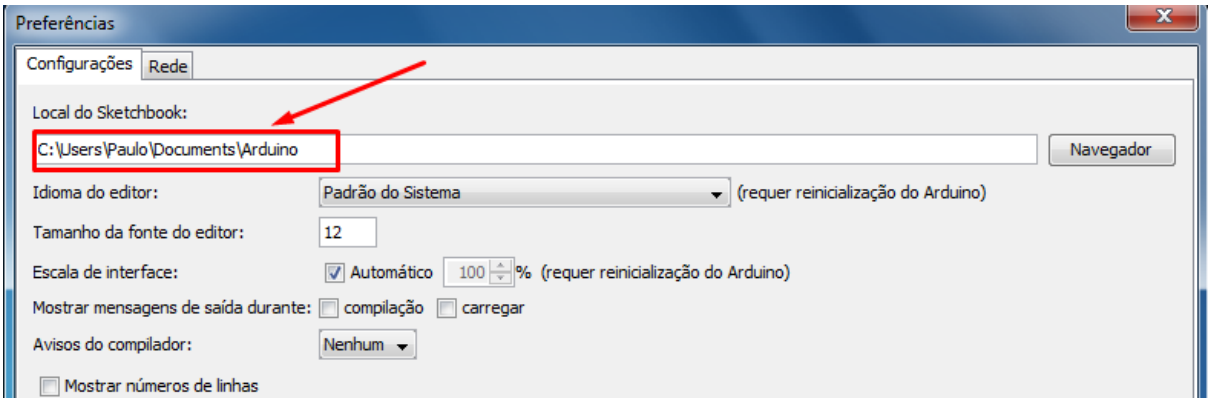


Se tudo deu certo você deve ter aberto uma tela igual à mostrada na imagem abaixo. Para executar o código basta seguir as instruções marcadas em vermelho na imagem, assim como foi feito com o código Blink anteriormente. Caso o código tenha sido carregado corretamente no Arduino, você verá uma mensagem semelhante a que está na parte inferior (parte de cor preta) no Arduino IDE e um status de "Carregado", um pouco acima, assim como mostra também na imagem abaixo.

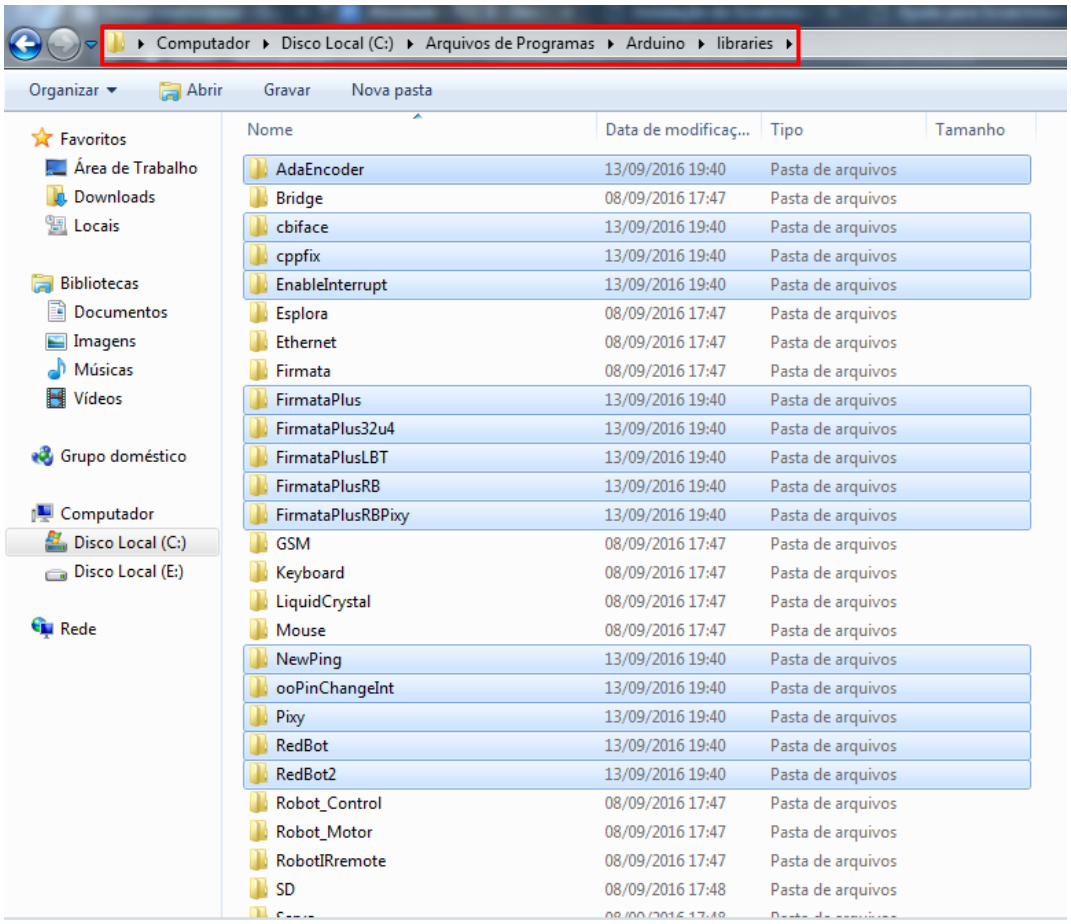


Agora que você tem o Firmata carregado em seu Arduino, você consegue utilizar alguns recursos disponíveis (LED, servomotor). Porém para a nossa atividade nós iremos um pouco além disso. Como iremos utilizar sensor de ultrassom, é necessário a utilização de um protocolo um pouco mais completo, o Firmata Plus. Este protocolo tem as mesmas funcionalidades do Firmata porém tem suporte a outros componentes tais como som e sensor de ultrassom. Para instalar o Firmata Plus você deve realizar o download do arquivo disponível neste link https://github.com/MrYsLab/pymata-aio/blob/master/FirmataPlus/libraries.zip. Extraia o arquivo libraries.zip e cole o conteúdo extraído na pasta padrão do *skecthbook*. Esta pasta é onde estão diversas bibliotecas e programas disponíveis no Arduino IDE. Para saber onde é o diretório correto do *sketchbook*, vá em **Arquivo > Preferências** e veja o campo "Local do *Sketchbook*", conforme mostra imagem abaixo. Logo após localizar o diretório, feche o

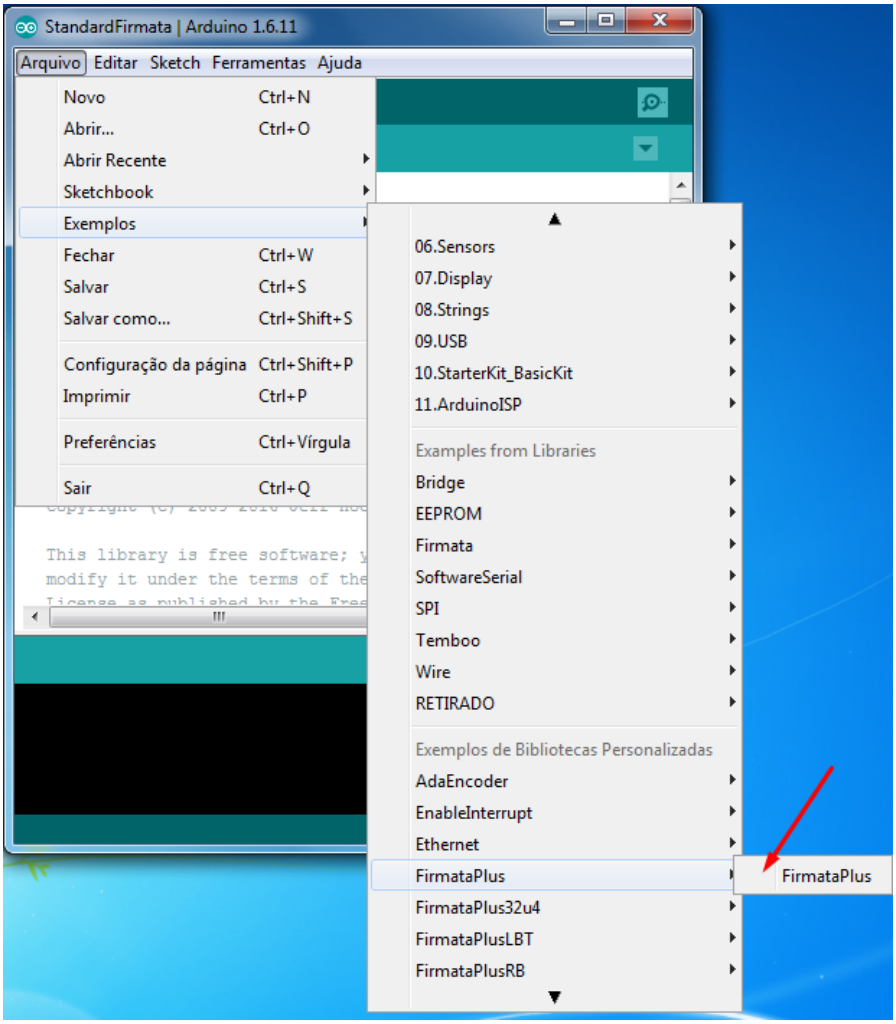
Arduino IDE.



Dentro do diretório “Arduino” existe uma pasta chamada libraries, é lá dentro que você irá colar o conteúdo extraído, exatamente como mostra imagem abaixo. Para nossa atividade iremos utilizar somente a biblioteca “Firmata Plus”, porém se você quiser se aprofundar mais e utilizar outros recursos, eles já estarão disponíveis em seu Arduino IDE.



Agora abra novamente o Arduino IDE e as bibliotecas já devem estar disponíveis no menu. Para carregar o Firmata Plus em seu Arduino, vá em **Arquivo > Exemplos > FirmataPlus > FirmataPlus**, conforme indicado na imagem abaixo. Não esqueça de conferir se o seu Arduino está conectado na USB e se as configurações da aba "Ferramentas" que foram feitas anteriormente estão corretas conforme o Arduino que você está utilizando.



Agora que já temos o Firmata Plus carregado em nosso Arduino, precisamos realizar as últimas configurações do servidor de comunicação. Em nossa atividade, iremos realizar a comunicação Scratch/Arduino através de um servidor conhecido como s2a_fm. O s2a_fm é uma extensão de hardware escrita em Python[9] que permite o Scratch se comunicar com Arduino sem problemas. Para utilizá-lo, é

9 Python é uma linguagem de programação bastante utilizadas nos dias de hoje

necessário a instalação do Python e algumas bibliotecas disponíveis nesta linguagem de programação. Acesse este link https://www.python.org/downloads/ e faça o download da versão 2.7.12. Feito o download, faça a instalação do mesmo. A partir desta etapa nós iremos somente realizar o download de alguns recursos e depois com um instalador, faremos todas as configurações de uma forma bem mais fácil. Após a instalação do Python, você deve instalar uma de suas bibliotecas chamada pip que nos permite instalar outras bibliotecas apenas com um comando simples sem precisar baixar um arquivo, descompactar e instalar. Para isso, acesse o link https://github.com/pypa/pip, faça o download e faça a extração do arquivo baixado na pasta Downloads. Não faça nada com os arquivos desta pasta, pois como comentado anteriormente só vamos precisar chamar um instalador que o resto será tudo feito automaticamente, vamos partir para próxima etapa.

Carregar os blocos de comunicação

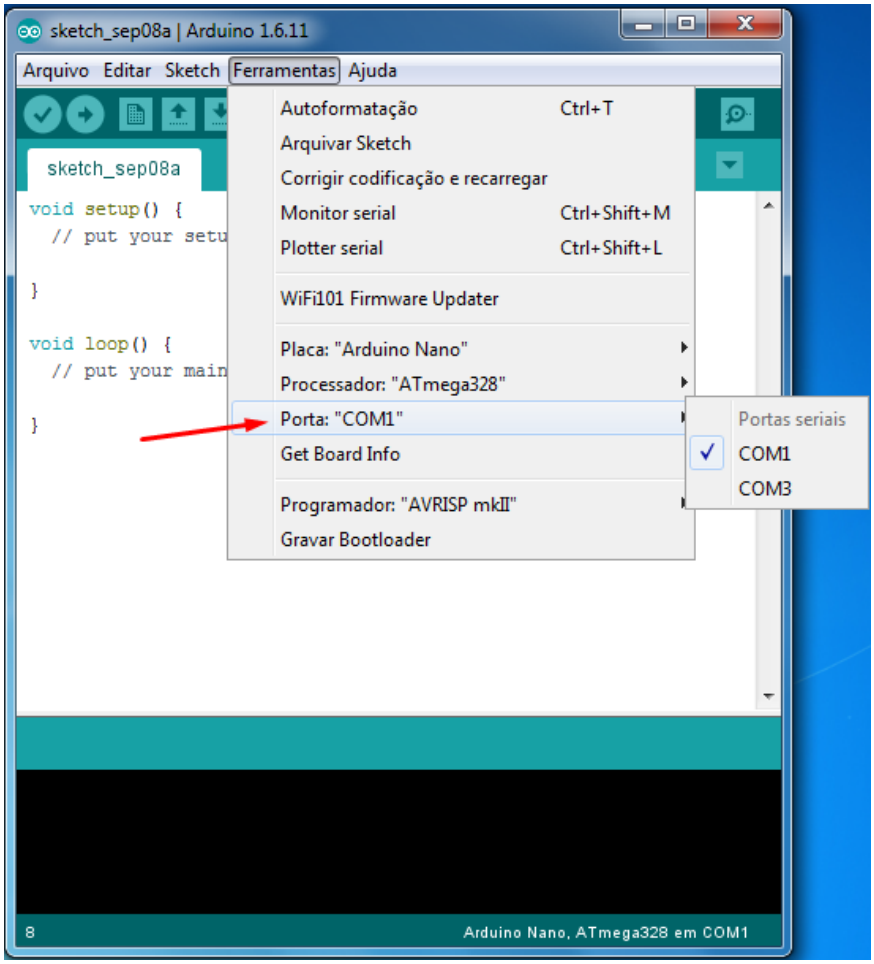
Bastante informação até agora né? Mas você quer uma notícia boa? Esta é última etapa para deixarmos tudo pronto para nossa atividade, vamos lá. Para que o Scratch possa se comunicar com o Arduino, ele precisa de novos blocos. Esses blocos possuem comandos para cada uma das funcionalidades do Arduino. Para isso, nós teremos que carregar uma API[1] que disponibiliza esses recursos. Os blocos estão disponíveis neste link https://github.com/MrYsLab/s2a_fm. Após realizar o download, você deve extrair o arquivo baixado na pasta Downloads. Agora é o momento que usaremos o instalador mágico, ele fará a configuração de diversas etapas pra gente de uma força muito simples. Para conseguir este instalador faça o download neste link https://github.com/rricardi/configuracao_scratch_arduino. Feito o download dos arquivos, faça a extração da pasta e siga os seguintes passos:

1. Clique em **paths.bat** para executar o arquivo. Ao clicar neste arquivo abrirá uma tela preta, após o término da configuração aperte qualquer tecla para finalizar esta etapa.

Para prosseguirmos para a próxima etapa é necessário que seu Arduino esteja conectado ao computador. Precisamos saber qual o nome de nossa porta USB o Arduino se encontra. É possível visualizar o nome da

USB na interface do Arduino IDE no menu **Ferramentas > Porta**. Em nosso exemplo o nome da USB é chamada de COM1, assim como mostra a imagem abaixo. Tendo essa informação, podemos prosseguir para o próximo passo.

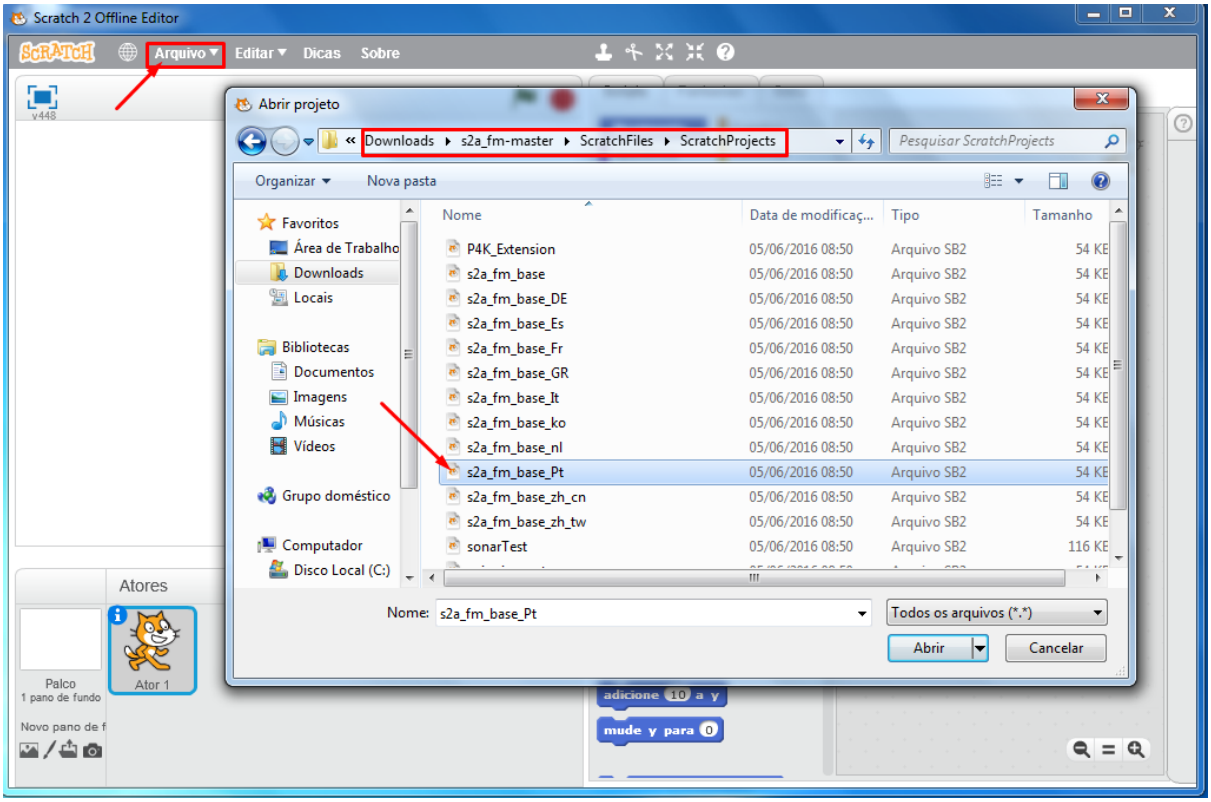
[1] API é uma interface de aplicação responsável por possuir comandos prontos para que você possa agilizar seu desenvolvimento.



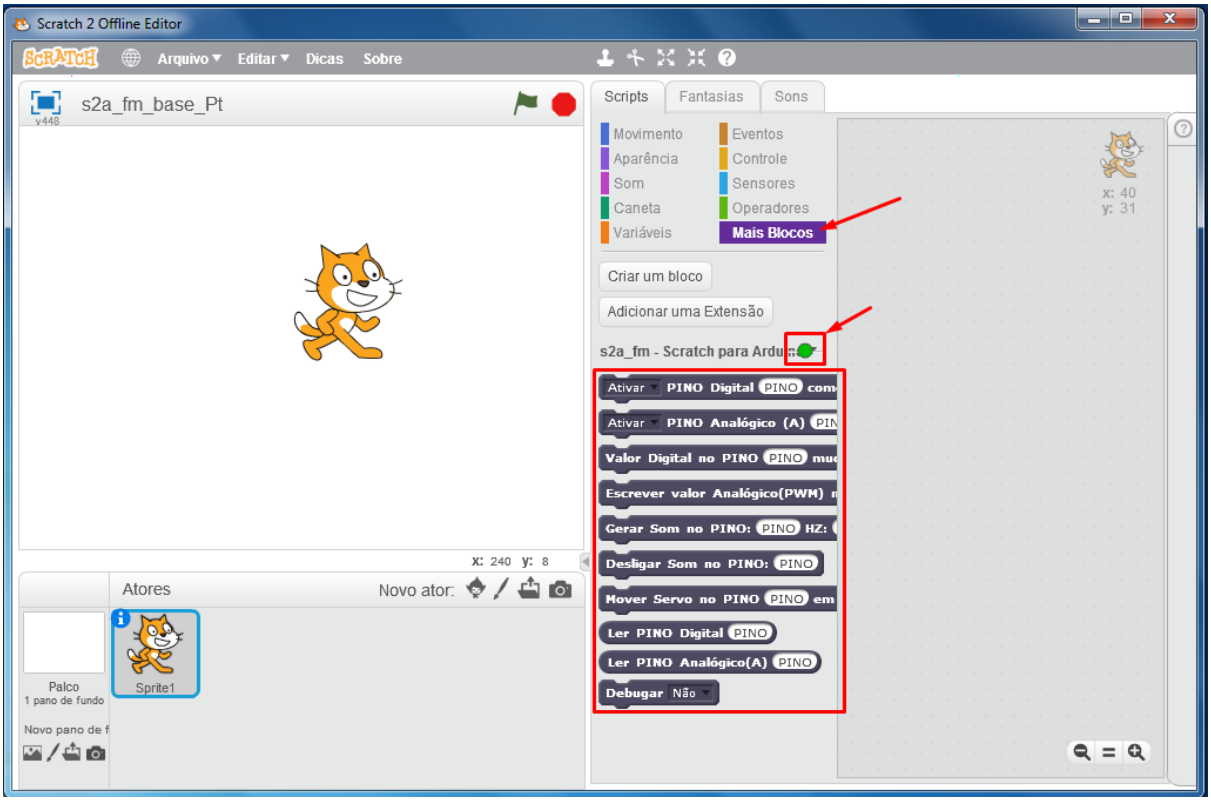
2. Clique em **instalador.bat** para executar o arquivo. Ao clicar neste arquivo abrirá uma tela preta, após o término da configuração será necessário digitar a porta em que o Arduíno está conectado. Ex: COM1

Se você estiver visualizando a mensagem “Please start Scratch or Snap! Scratch detected! Ready to rock and roll...”, é porque ocorreu tudo corretamente e está quase tudo pronto pra gente finalizar a configuração

de nosso ambiente. Estas duas etapas que realizamos anteriormente, é necessário ser feita somente uma vez. Toda vez que você quiser fazer alguma atividade com Scratch/Arduino, a partir de agora você deve rodar somente o arquivo **s2afm.bat** e no final de sua execução será necessário informar a porta USB que o Arduino está conectado. Bom, agora você deve acessar a interface do Scratch e clicar em **Arquivo > Abrir** e ir até o diretório onde você fez o download dos blocos (arquivo chamado s2a_fm-master). O arquivo que você deve carregar está no seguinte diretório **s2a_fm-master > ScratchFiles > ScratchProjects > s2a_fm_base_Pt.sb2.**



Conseguiu encontrar? Agora os blocos da API do Arduino estão carregadas e estão prontas para serem utilizados. Para visualizá-los você deve clicar na aba “Mais Blocos” na interface do Scratch.



Se tudo deu certo, você deve estar visualizando diversos blocos da cor preta. Esses blocos de cor preta significam algo externo ao próprio Scratch, ou seja, através deles é possível enviar comandos ao Arduino. Logo acima dos blocos pretos existe uma luz da cor verde que indica que a comunicação Scratch/Arduino está funcionamento corretamente. Caso contrário aparecerá uma luz vermelha, indicando que temos algum problema de comunicação.

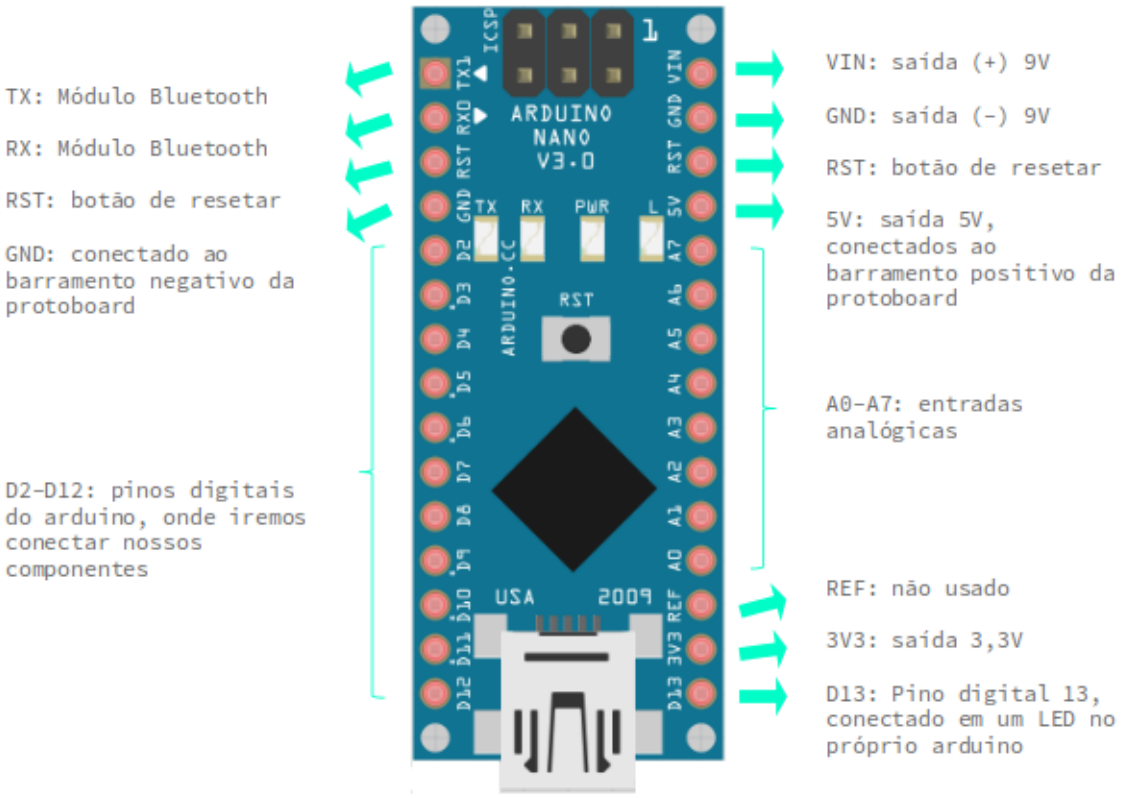
Manual 3 - Testando o ambiente Arduino/Scratch

Testando o seu Arduino com Scratch

Agora que já temos todo o nosso ambiente configurado, vamos fazer um programa teste para ver se está tudo rodando corretamente? Você lembra aquele programa que utilizamos anteriormente no Arduino IDE chamado Blink? Então, vamos reproduzi-los agora no Scratch.

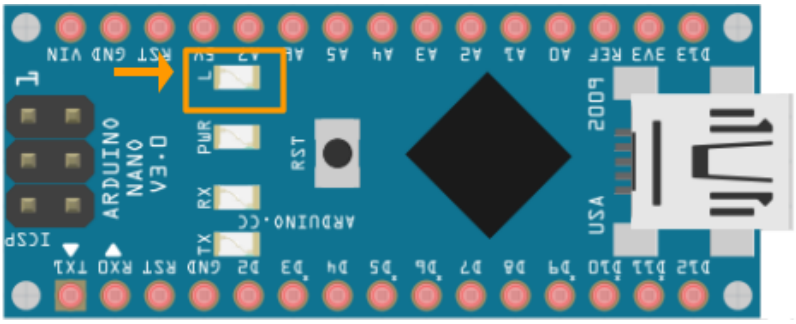
Programa Blink no Scratch

Os pinos disponíveis no Arduino servem para realizar a ligação de componentes. Na imagem abaixo podemos ter uma breve explicação das funções dos pinos do Arduino Nano.

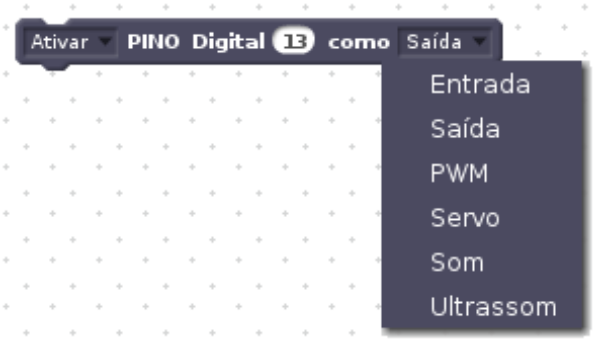


Caso o pino tenha a função de controlar algo, ele é chamado de pino de saída. Em nosso programa teste nós iremos utilizar um bloco que ativa um determinado pino com a função de saída. Sempre quando formos utilizar algum pino do Arduino temos que ativá-lo com alguma função. Vá na aba “Mais Blocos” e arraste o bloco “Ativar pino digital” para área de programação.

O Arduino possui um pino em que um LED já está conectado na placa. Este pino é 13 e está conectado ao LED indicado na imagem abaixo.



Então para darmos a funcionalidade esperada para este pino, precisamos ativá-lo como um pino de saída. Substitua o valor de “PINO” por “13” (o pino que já está ligado no LED) e escolha a opção “Saída”, conforme mostra imagem abaixo.

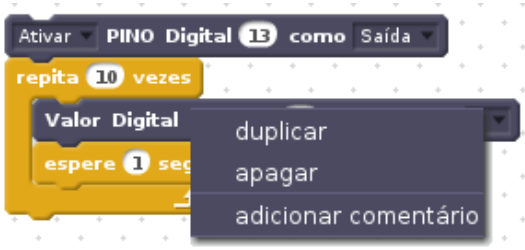


Bom, agora que já temos nosso pino ativado, vamos colocar um bloco em que determinada funcionalidade se repita algumas vezes. No nosso caso, iremos fazer com que o LED acenda e apague 10 vezes seguidas. Para isso, utilizaremos um bloco que é conhecido como estrutura de repetição ou laço. Ele tem a função de executar determinados comandos em um intervalo de vezes. Vá na aba Controle e arraste o bloco “Repita 10 vezes” para baixo do bloco de ativação do pino 13, conforme mostrado na imagem abaixo.

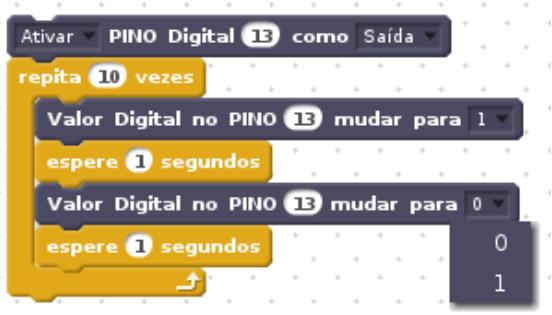
Agora para fazermos a luz apagar e acender 10 vezes, colocamos um bloco dentro da estrutura de repetição. Este bloco muda o valor de saída do pino para determinada função. Primeiramente queremos

acender o LED e depois apagá-lo, certo?. Então para acendê-lo escolhemos bloco “Valor Digital no Pino” que está na aba “Mais Blocos” e mudamos o valor do pino 13 para 1. O valor 1 significa que o LED acenderá, isso acontece pois o computador trabalha com valores 0 ou 1. Esta forma de trabalhar é conhecida como Lógica Digital, e é como o computador sabe que funcionalidades ou que ações deve exercer. Após piscar o LED, vamos deixá-lo aceso durante 1 segundo. Para isso, vamos na aba “Controle” e fazemos como imagem abaixo.

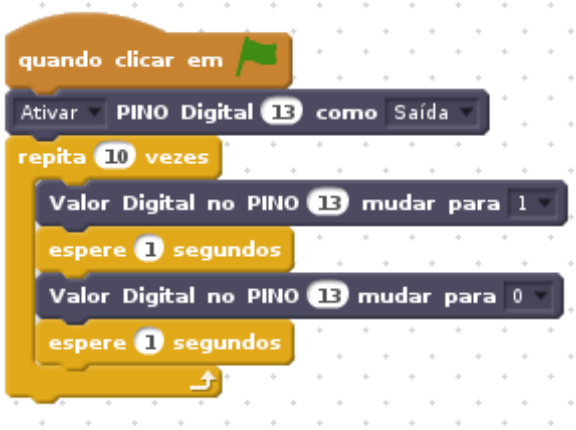
Agora para que o LED apague durante um tempo, nós iremos utilizar os mesmos blocos utilizados anteriormente, porém só iremos modificar a função dele. Para facilitar nossa vida, podemos duplicar os comandos já utilizados. Clique com o botão direito em cima do bloco que queira utilizar novamente e escolha a opção “Duplicar”.



Foi feita uma cópia dos blocos duplicados, certo?. Mude agora o valor do pino 13 para 0 em vez de 1, conforme mostrado na imagem abaixo.



Neste momento nosso código já está apto para rodar normalmente, só iremos colocar um bloco que inicialize nosso código. Para isso, vá na aba “Eventos” e escolha o bloco da bandeirinha verde “Quando clicar em...”, conforme mostra imagem abaixo.



Pronto, agora nosso código Blink está prontinho para ser executado, clique na bandeira verde acima do gatinho do Scratch e vejo o LED do Arduino piscar. Parabéns, você acaba de criar seu primeiro projeto com Arduino e Scratch. Guarde este código pois utilizaremos ele para nossa atividade do Pikachu.

Manual 4 - Manipulando os componentes da atividade

Atividade Pikachu

A partir de agora iremos iniciar a montagem da nossa atividade do pikachu. Que tal começarmos pela luz que saíra da pokebola simulando a captura do pokemon? Vamos lá então.

Lista de materiais

- Componentes eletrônicos
  - Arduino Nano
  - Protoboard 830 furos
  - Servo motor 9g Tower Pro
  - Sensor de Ultrassom HC-SR04
  - Jumpers macho-macho
  - Jumpers macho-fêmea
  - LED
  - Resistor
- Materiais para estrutura física
  - Folha de papel A4 amarela
  - Bola de isopor 75mm
  - Fita durex
  - Caneta hidrocor de vermelha e de cor preta
  - Palitinho para churrasco
  - Tesoura
  - Estilete
  - Lápis

Montagem do Arduino

Vamos para a montagem?

Materiais necessários:

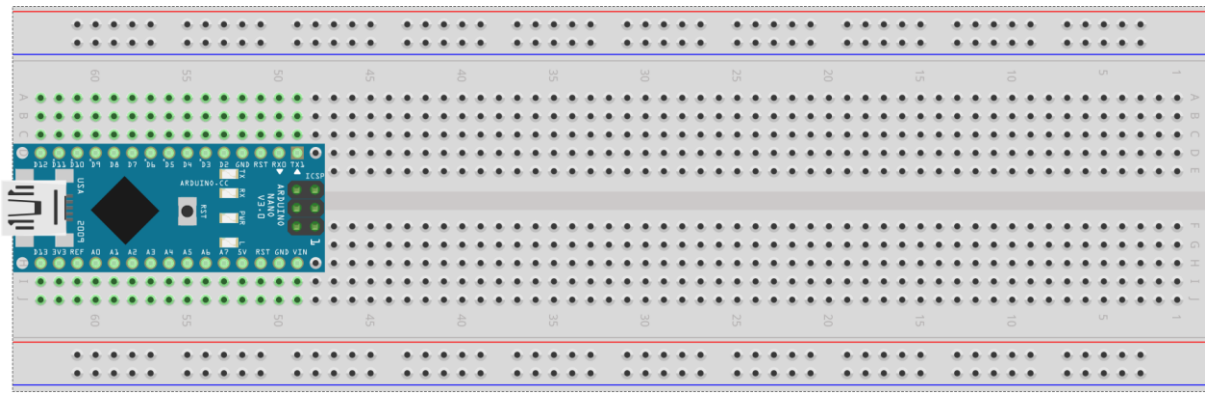
Nesta etapa serão utilizados um Arduino Nano e uma protoboard.

Montagem

Para esta etapa siga a sequência de passos:

Dica: A extremidade em que o Arduino será colocado não importa, só é importante deixar a entrada USB virada para o lado de fora da protoboard para um melhor manuseio.

Passo 1: Encaixar o Arduino Nano na protoboard, conforme ilustrado na abaixo.



Montagem da estrutura inicial da atividade

Vamos para a montagem?

Materiais necessários

Nesta etapa serão utilizados 4 jumpers macho-macho. Além disso também serão utilizados papel, tesoura e estilete.

Montagem

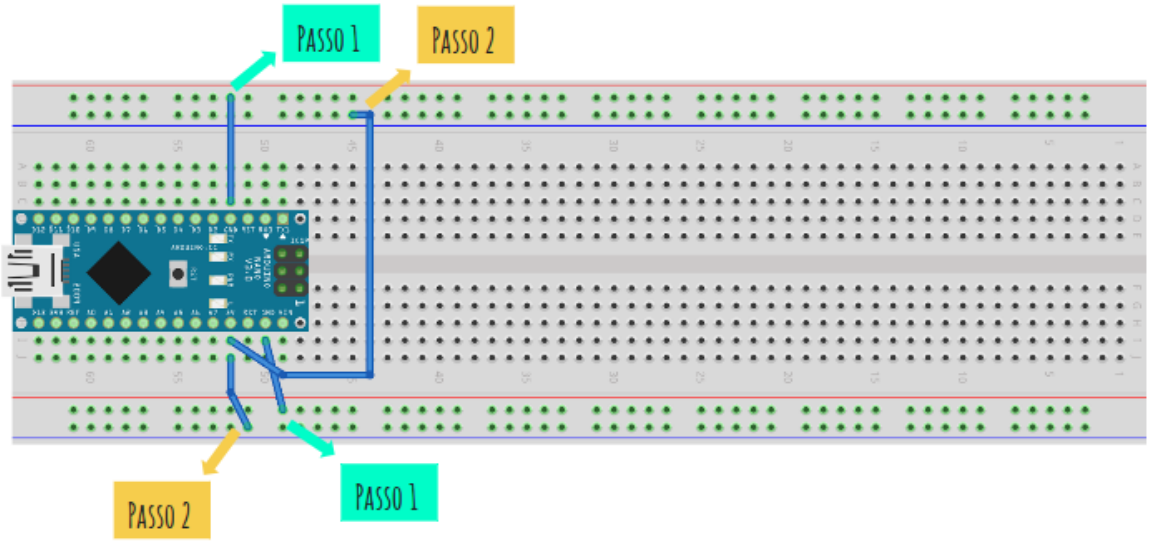
- Componentes eletrônicos

Esta etapa é necessária para expandir alguns pinos do Arduino (5V e GND) para que possam ser usados diversas vezes. O pino 5V fornece uma tensão de 5 volts para alimentação de circuitos externos e o GND é o terra que serve para recolher elétrons “fugitivos”. O Arduino Nano possui 1 pino 5V e 2 pinos GND, com a montagem dos passos desta etapa será possível expandir esse número de pinos. Para isso realize os seguintes passos:

Dica: Utilizar pares de fios de cores iguais para os polos positivos e negativos.

Passo 1: Conectar a ponta de um fio no pino GND e a outra no polo positivo da protoboard. Realizar essa ação para os dois pinos GND do Arduino. Lembrando que no caso desta protoboard utilizada, temos polos positivos e negativos dos dois lados da protoboard.

Passo 2: Conectar a ponta de um fio no pino 5V e a outra no polo negativo da protoboard. Na coluna do pino de 5V, no furo ao lado conectar outro fio e a outra ponta no outro polo negativo da protoboard, assim como mostrado na imagem.



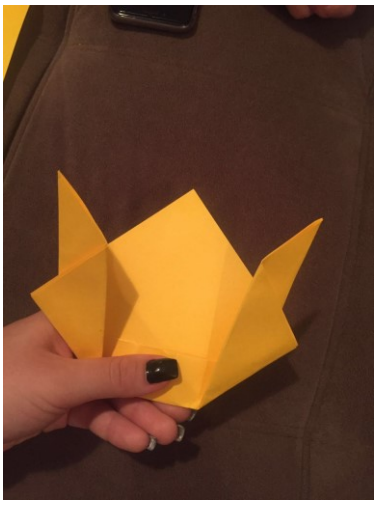
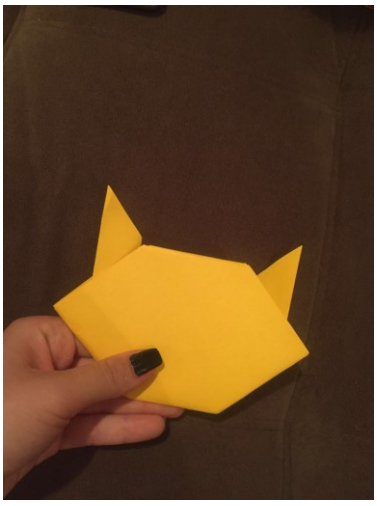
- Estrutura física

Para montar nosso pikachu vamos utilizar uma folha A4 e fazer dobraduras conforme imagem abaixo.

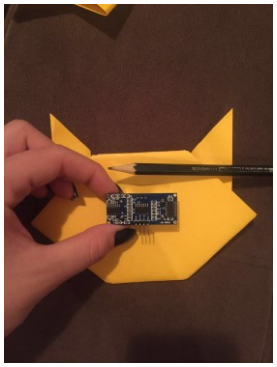
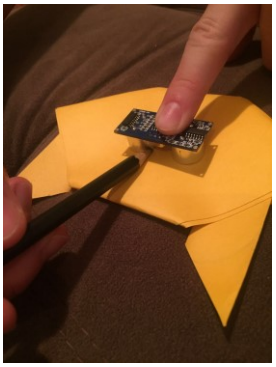

PiKOCHU
Origami
1.
2.
3.
4.
5.
6.



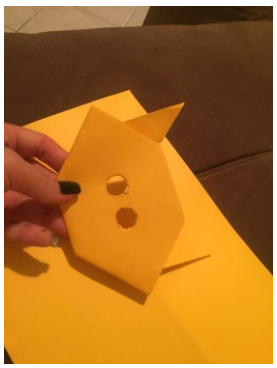


Os olhinhos do pikachu serão o nosso sensor. Após o origami do pikachu estar pronto meça o tamanho das bolinhas do sensor fazendo uma marcação de lápis no papel.


Depois de fazer as marcações, corte ao redor das bolinhas com estilete (com muito cuidado!!!). Faça os ajustes necessários para encaixar o sensor no lugar dos olhos.
Quanto aos enfeites do pikachu use sua criatividade. Pinte de amarelo (ou use uma folha amarela), pinte as bochechas de rosa, pinte as orelhinhas e dê um sorrisão pro seu pikachu.

Montagem da luz da pokebola

Vamos para a montagem?

Materiais necessários

Nesta etapa serão utilizados um LED da cor vermelha, um resistor e 6 jumpers macho-fêmea. Além disso também serão utilizados uma bola de isopor, caneta hidrocor vermelha e preta e um palitinho de churrasco para auxiliar na elaboração do pokebola.

Montagem

- Componentes eletrônicos

O LED possui um polo positivo e outro negativo. O polo positivo é representado pela perna maior, já o fio negativo pela perna menor. Para esta etapa siga a sequência de passos:

Dica: Para facilitar o manuseio dos LEDs na protoboard e da pokebola na atividade em geral, pegue três pares de cores diferentes, como por exemplo azul e branco.

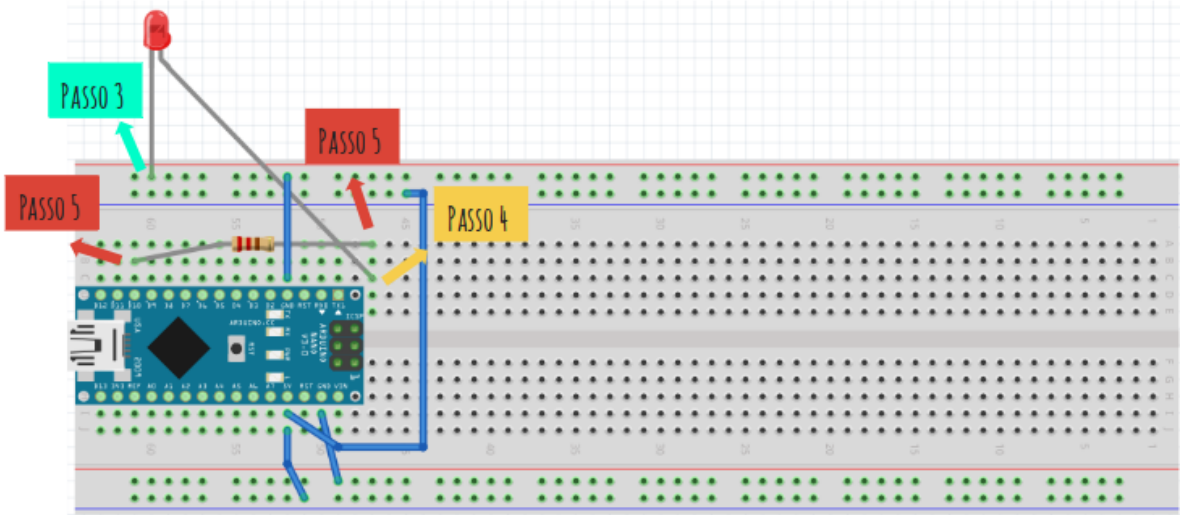
Passo 1: Conectar a perna positiva do LED a uma tomada fêmea do jumper. Podemos utilizar o fio de cor branca, conecte os três jumpers macho-fêmea juntos.

Passo 2: Conectar a perna negativa do LED a uma tomada fêmea do jumper. Podemos utilizar o fio de cor azul, conecte os três jumpers macho-fêmea juntos.

Passo 3: Conectar o lado negativo (fio de cor azul) ao barramento positivo da protoboard.

Passo 4: Conectar o lado positivo (fio de cor branca) à um furo livre da protoboard que não esteja conectada ao Arduino.

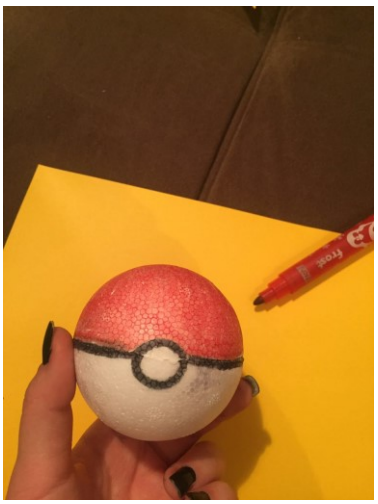
Passo 5: Conecte uma perna do resistor ao barramento de furos que passa pelo pino digital 10 do Arduino e a outra perna à coluna de fios positivos que foram usados nos LEDs.



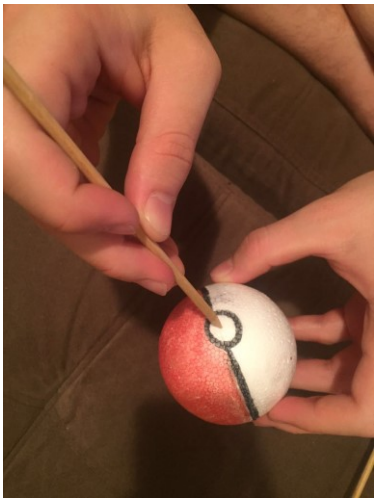
- Estrutura física

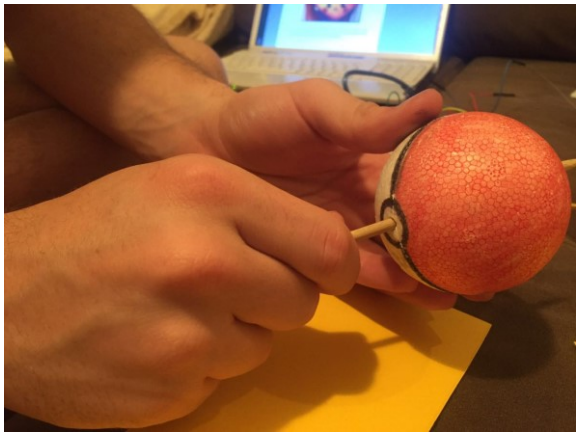
Chegamos a etapa de montar a pokebola. Antes de tudo decida como você vai pintá-la, com tinta

guache ou hidrocor, nessa etapa ela deve estar sequinha. Você pode seguir as cores vermelha, branca e preta conforme a pokebola original. Ou transformar sua pokebola numa “Great ball” ou até “Ultra ball”.

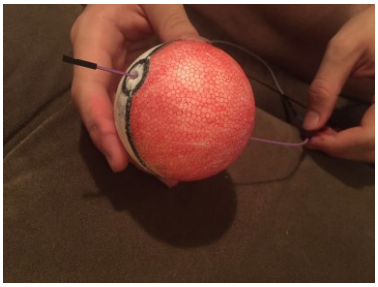


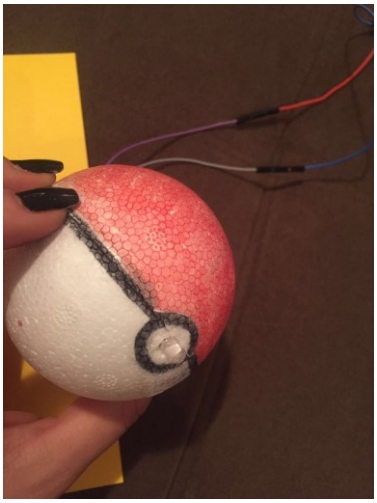
Use o palito de churrasco (com cuidado!!!) para perfurar a pokebola fora a fora abrindo espaço para passar o LED e os fios.





Em seguida coloque o LED que deve encaixar certinho nesse espaço.





Vamos para a programação?

Anteriormente nós criamos um código que fazia com que a luz no Arduino piscasse por diversas vezes, nesta etapa vamos utilizar este código novamente. Porém agora a luz que piscará será a do LED da pokebola. Para isso abra o código desenvolvido anteriormente e somente mude o pino do Arduino que está configurado como saída para o valor 10 e faça isso se repetir somente duas vezes. Além disso também vamos iniciar o programa de uma diferente. Para isso vá na aba Eventos e pegue o bloco que funciona através de uma tecla e mude-a para letra “L”, assim como na imagem abaixo. Desta forma, quando pressionarmos a tecla “L” o LED começará a piscar.

Agora encaixe este bloco na código desenvolvido anteriormente e você terá um algo semelhante a imagem abaixo.

A explicação deste trecho de código é exatamente a comentada no código desenvolvido anteriormente. A única mudança é no pino do Arduino que vamos utilizar e na quantidade de vezes que o LED piscará. Bem tranquila essa parte né? Agora você já está preparado para capturar um pokemon, só precisamos agora encontrá-lo.

Montagem do movimento do Pikachu

Vamos para a montagem?

Materiais necessários

Nesta etapa serão utilizados um servo motor e 3 jumpers macho-macho. Além disso também serão utilizados um palitinho de churrasco e durex.

Montagem

- Componentes eletrônicos

A estrutura de funcionamento do servomotor é composta de 3 fios: (preto ou marrom), (vermelho) e (laranja ou amarelo).

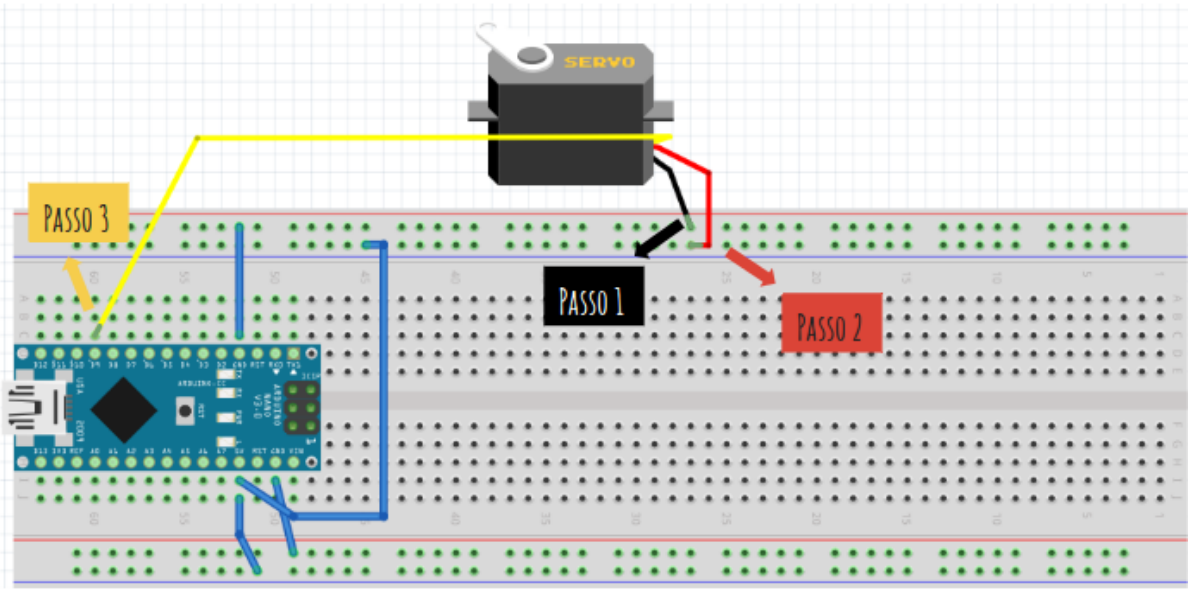
Dica: Utilize fios de cores iguais ou parecidas com as que são utilizadas na conexão do servo motor. Isto deixará o seu circuito mais visível caso seja necessária alguma modificação.

Passo 1: Conectar o fio preto ou marrom que representa o negativo (GND) ao Pino GND do

Arduino.

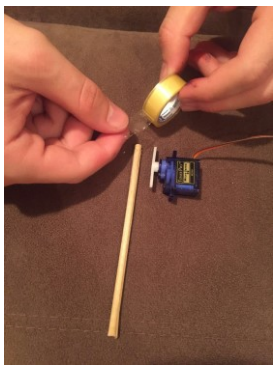
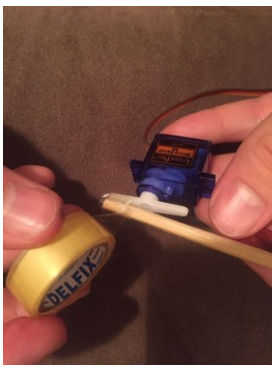
Passo 2: Conectar o fio vermelho que é o positivo, por onde a energia passa, ao pino 5V do Arduino.

Passo 3: Conectar o fio amarelo ou laranja que é o fio de controle por onde o Arduino enviará comandos ao pino digital 9 do Arduino.

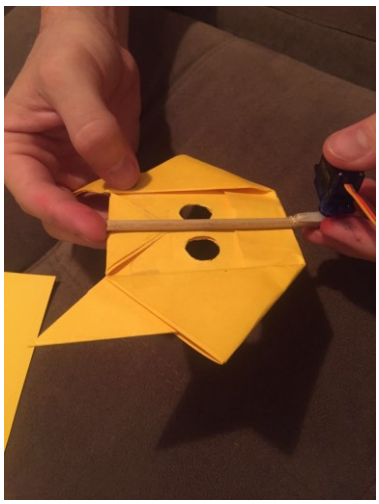
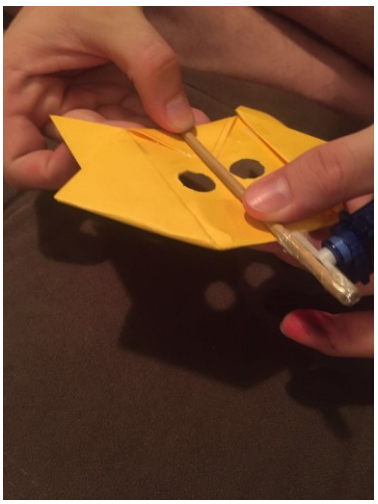


- Estrutura física

Agora iremos montar a parte que dá a movimentação ao Pikachu. Para isso, encaixe a peça que possuem duas pontas no servomotor. Agora você deve pegar o palitinho de churrasco e cortá-lo pela metade para que não fique tão grande. Pegue o palitinho que você acabou de cortar e junte-o com o atuador passando fita adesiva em volta.

Agora junte o palitinho de churrasco que está no atuador com o origami do Pikachu. Para isso encaixe o palitinho na parte de trás do origami, passe fita durex para deixar a estrutura mais fixa.


Vamos para a programação?

Agora vamos fazer com que o nosso pikachu tenha um movimento. Nesta etapa iremos criar um código separado para que possamos rodar este trecho separadamente porém ainda no mesmo projeto do Scratch. No final de todas as etapas iremos juntar os códigos e transformá-lo em um só. Vamos iniciar nosso programa indo na aba Eventos e escolhendo o bloco de tecla e depois mude para a letra "M". Desta forma quando pressionarmos a tecla "M" o pikachu realizará um movimento.

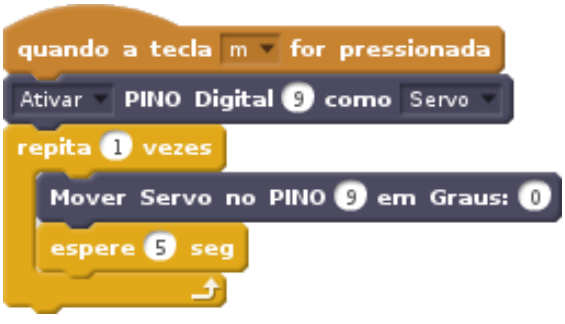
O próximo passo é escolher um bloco para que o Arduino possa entender no que pino 9, ele terá a função de "Servo", ou seja, de movimento. Para isso vá na aba "Mais Blocos" e escolha o bloco de ativar pinos e troque para a função "Servo".

Neste momento queremos que o Pikachu realize o movimento de ser capturado somente uma vez. Para isso vá na aba "Controle" e escolha o bloco que tem a função de um laço e troque o parâmetro para 1,

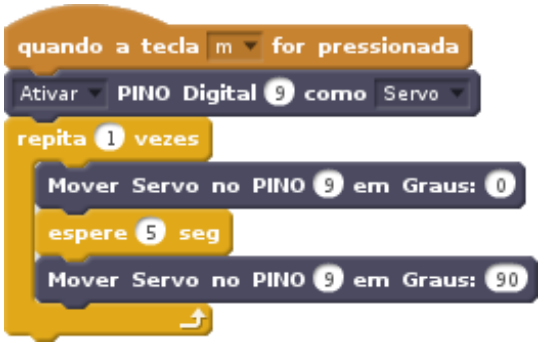
assim como na imagem abaixo.

Para que o Pikachu possa realizar um movimento, ele precisa mover determinada angulação. Para isso iremos escolher um bloco que mexe na angulação, vá em “Mais Blocos” e mude seus parâmetros para que mova no pino 9 e em 0 graus, assim como na imagem abaixo. Este é o momento em que você pode considerar que o Pikachu fora capturado pois ele fará um movimento para o chão.

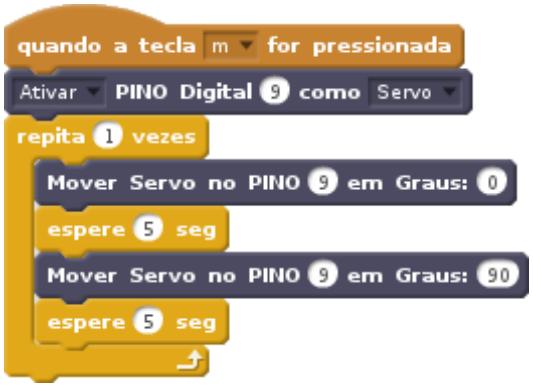
Após esta etapa, vamos fazer com que a captura fique um pouco mais real deixando o Pikachu um intervalo de tempo deitado no chão. Para isso vá na aba “Controle” e escolha o bloco para esperar alguns segundos e mude o valor do parâmetro para 5.



Como queremos que o Pikachu retorne à posição original e que possa ser capturado novamente, nós precisamos fazer ele voltar a ficar em pé. Para isso vá escolha novamente a movimentação do “Servo” e troque para o pino e mude a angulação para 90 graus.



Faça que ele espere alguns segundos novamente e você pode iniciar a sua movimentação mais uma vez. O código final desta etapa é o mostrado na imagem abaixo, vamos testar agora?



Montagem da percepção do Pikachu

Vamos para a montagem?

Materiais necessários

Nesta etapa serão utilizados um sensor de ultrassom, 4 jumpers macho-fêmea e 1 jumper macho-macho. Além disso também serão utilizados o origami do Pikachu e durex.

Montagem

- Componentes eletrônicos

O sensor de ultrassom é composto de 4 pinos: (VCC, TRIG, ECHO, GND). Nesta etapa serão utilizados 4 jumpers macho-fêmea e 1 macho-macho, além do sensor de ultrassom.

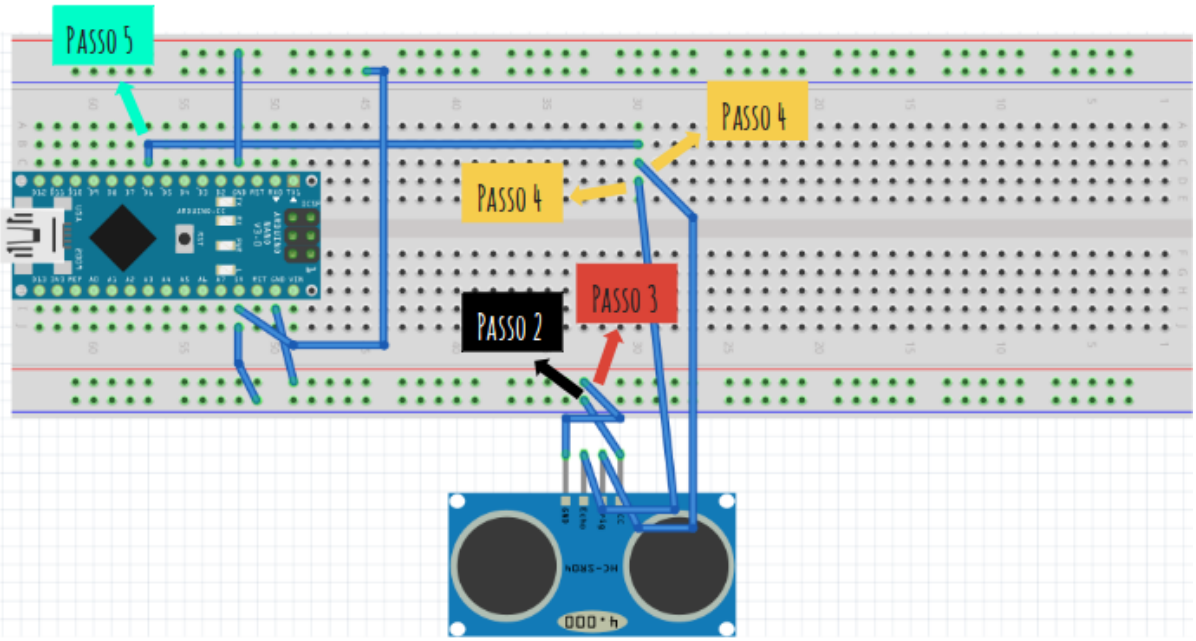
Passo 1: Conectar os jumpers macho-fêmea em cada um dos pinos presentes no sensor.

Passo 2: Conectar o fio do pino VCC do sensor ao pino 5V do Arduino.

Passo 3: Conectar o fio do pino GND do sensor ao pino GND do Arduino.

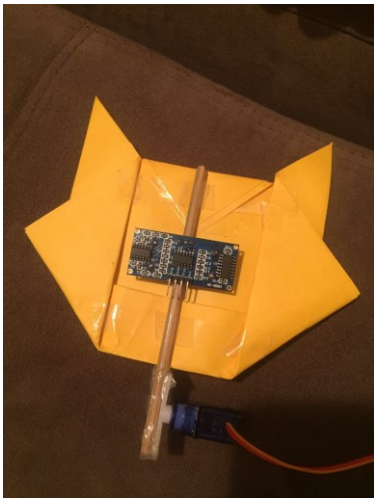
Passo 4: Conectar os pinos TRIG e ECHO em uma coluna livre da protoboard, conforme mostra figura 3.

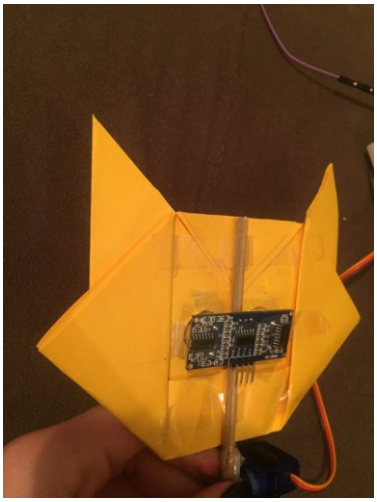
Passo 5: Conectar uma extremidade do jumper macho-macho em outro furo na mesma coluna dos pinos TRIG e ECHO à outra extremidade no pino digital 6 do Arduino.



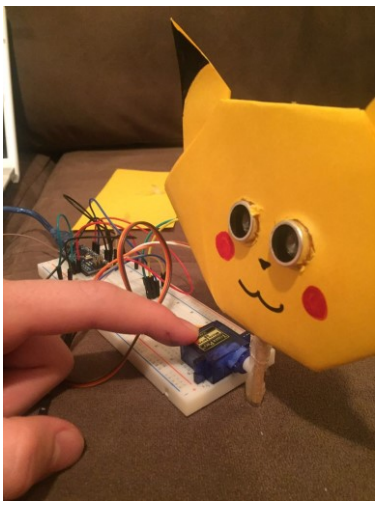
- Estrutura física

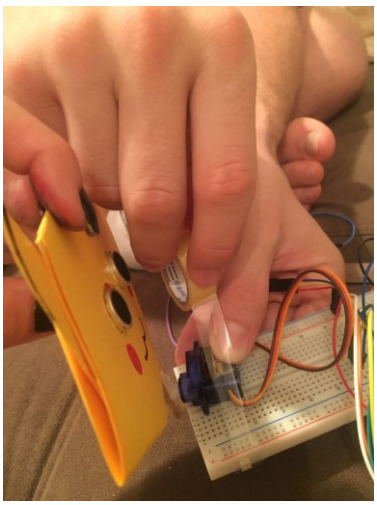
Pegue o origami e encaixe o sensor nos furos dos olhinhos do Pikachu. Passe uma fita durex atrás para fixar melhor o sensor.

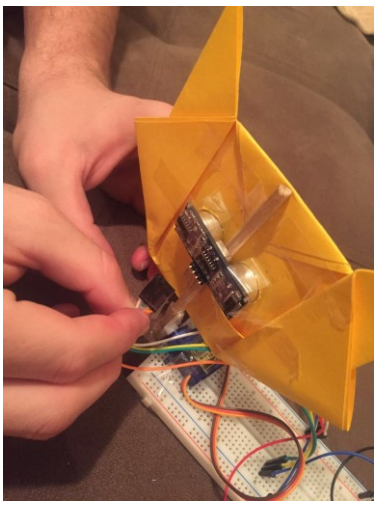




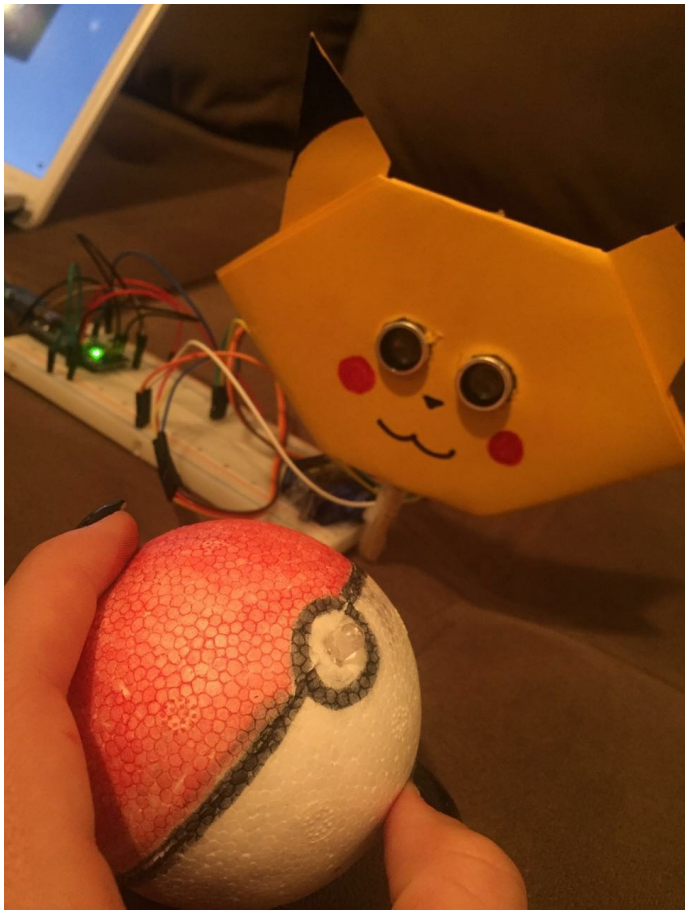
Agora pegue o atuador e coloque-o fixo na protoboard e passe durex em volta dele também. Lembrando que a posição do atuador já deve ser pensada para realizar o movimento de queda do Pikachu. Não esqueça de conectar os pinos do sensor, conforme mostra a terceira imagem abaixo.







A estrutura final deve ficar semelhante a imagem abaixo, legal né? Agora você já está com a estrutura toda pronta, vamos terminar a parte de programação para finalizarmos a atividade.



Vamos para a programação?

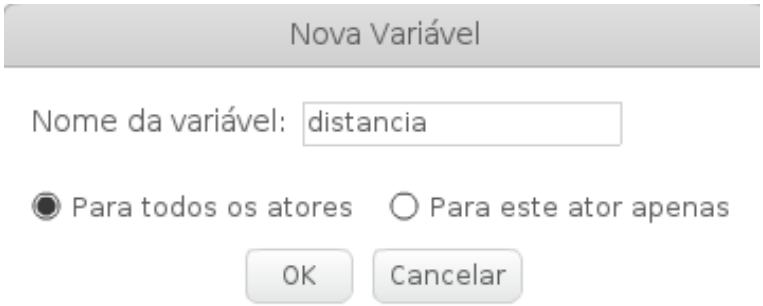
Agora vamos fazer com que o Pikachu perceba a pokebola se aproximando e que reaja com essa aproximação. Dependendo da distância entre a pokebola e o Pikachu, esse caíra simulando a sua captura. Vamos começar indo na aba “Eventos” e escolhendo o bloco para clicar na bandeira verde e iniciar o código.

O próximo passo é escolher um bloco para que o Arduino possa entender no que pino 6, ele terá a função de “Ultrassom”, ou seja, de perceber a chegada de algum objeto próximo a ele. Para isso vá na aba “Mais Blocos” e escolha o bloco de ativar pinos e troque para a função “Ultrassom”.

Agora precisamos esperar um tempo para que o Arduino e o sensor de ultrassom possam se estabilizar e responder com a informação correta. Para isso vá na aba “Controle” e escolha o bloco para esperar alguns segundos e troque-o para 3, tempo necessário para que esta comunicação seja estabilizada.

Nesta parte nós iremos criar um local para guardar a distância que o sensor irá medir. Este local chamamos de variável. Variável é onde armazenamos os dados ou informações de um programa por um determinado espaço de tempo. Para isso vá na aba “Variáveis” e clique “Criar uma variável”.

Preenche o espaço em branco com o nome da variável, no nosso caso como estamos querendo guardar a distância da pokebola até o Pikachu vamos chamá-la de “distancia”. Após isso clique em “Ok”.



Podemos observar que foram criados novos blocos com determinadas funcionalidades. Esses novos blocos agora estão disponíveis para serem usados em nosso programa.

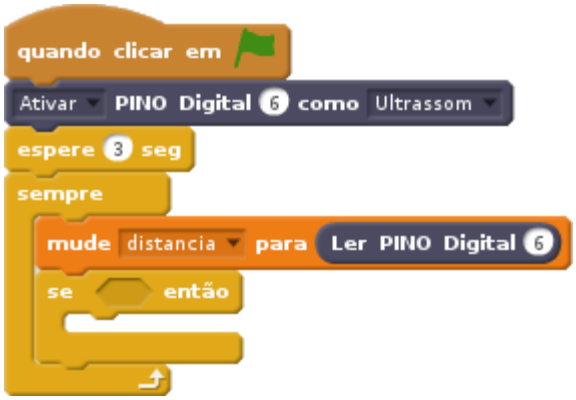
Bom, agora que já temos a variável distância criada, podemos utilizá-la em nosso código. Nesta etapa precisamos fazer com o sensor sempre leia o valor da distância entre os objetos. Para isso vá na aba "Controle" e escolha o bloco de sempre. Este é um bloco de repetição, com o mesmo intuito do que utilizamos anteriormente porém não repetindo um determinado número de vezes mas sim repetindo sempre.

Iremos colocar agora dentro de nosso laço algo que faça ler as informações de distância emita pelo nosso sensor. Precisamos de um bloco que mude o valor da distância conforme chegamos mais próximo ou nos distanciamos dos objetos. Para isso vá na aba “Variáveis” e pegue o bloco para mudar a distância.

Agora precisamos informar de onde queremos pegar esta distância. Neste caso queremos pegar do sensor de ultrassom que está passando informação no pino 6. Para isso vá na aba “Mais blocos” e pegue o bloco que lê o valor do pino digital, mude-o para 6 e coloque este bloco no parâmetro do bloco que muda a distância, assim como na imagem abaixo.

À partir de agora é possível medir a distância entre os objetos. Vamos garantir que está tudo funcionando, antes de continuarmos nossa atividade. Clique na bandeira e verde e observe o valor que está sendo passado na variável distância (canto superior esquerdo da interface do Scratch). Aproximando algo do sensor você pode ver o valor desta distância mudando, legal né?

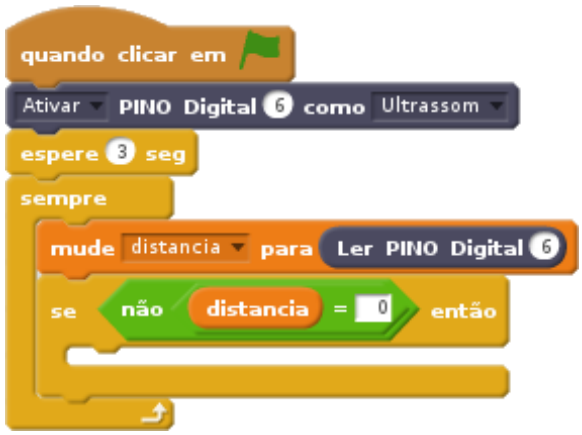
Agora que já sabemos que a distância está sendo capturada pelo sensor, vamos prosseguir com a atividade. Nesta etapa vamos ter que limitar o valor da distância para que o Pikachu realize determinada ação, que neste caso seria "cair" por alguns instantes. Em alguns casos, o sensor acaba exibindo o valor de distância zero antes mesmo de realmente de estar a esta distância do objeto. Para que isso não atrapalhe o funcionamento da atividade, precisamos garantir que nenhuma ação seja realizada quando ele estiver com distância zero.



1. Vá na aba "Controle" e escolha o bloco condicional.

2. Vá na aba "Operadores" e escolha o bloco de negação.

3. Vá na aba "Operadores" e escolha o bloco de igualdade.



4. Vá na aba "Variáveis", pegue a variável distância e coloque no primeiro parâmetro do bloco de igualdade, depois preencha o outro parâmetro com o valor 0.

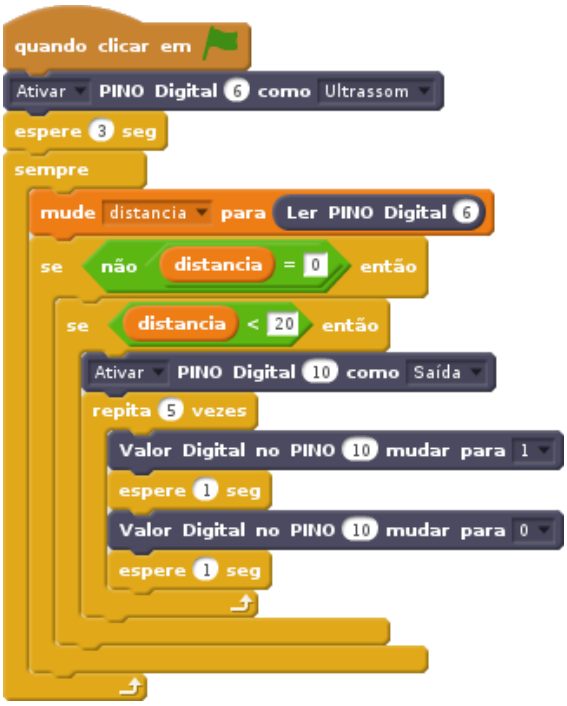
Bom, agora que garantimos que não haverá problemas quando a distância for igual a zero, iremos fazer outra condição. Essa outra condição fará com que um objeto a uma distância igual a 20 realize uma ação, ou seja, fará com que o Pikachu caia no chão.

1. Vá na aba “Controle” e escolha o bloco condicional.

2. Vá na aba “Operadores” e escolha o bloco “menor do que”.

3. Vá na aba “Variáveis”, pegue a variável distância e coloque no primeiro parâmetro do bloco de igualdade, depois preencha o outro parâmetro com o valor 20.

Agora nesta etapa vamos juntar nossos códigos desenvolvidos para que todos possam interagir e fazer nossa atividade funcionar corretamente. Primeiro pegue o código que faz a pokebola piscar, retire o bloco que o inicia e encaixe logo abaixo da verificação da distância menor do 20. Depois faça o mesmo para o código que realiza o movimento do Pikachu, essas etapas são mostradas nas imagens abaixo.

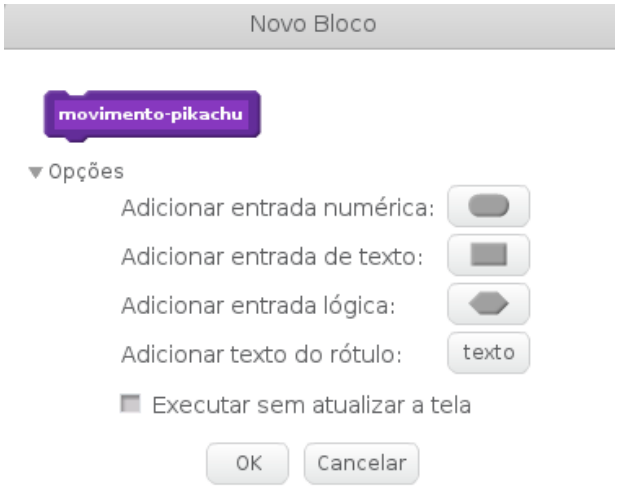


Bom, neste momento nós já temos a nossa atividade montada e funcionando. Mas que tal nos aprofundarmos mais nos conceitos de programação e deixarmos nosso código mais organizado? Vamos lá então!!

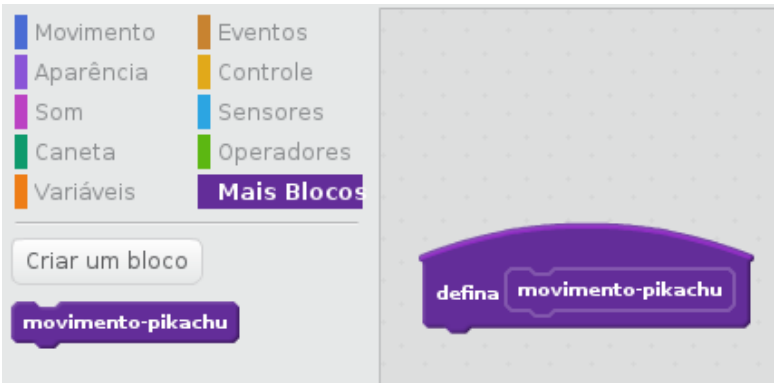
Melhorando o código

Nesta etapa iremos melhorar o nosso código para que ele possa ficar mais legível. Imagine que no meio de nossa atividade nós tivéssemos que fazer o pikachu se movimentar em diversos momentos. Teríamos que copiar o mesmo código várias vezes para que fosse possível essa execução. Para facilitar nossa vida podemos criar um bloco que somente ele no meio do código realize a função de movimentar o pikachu. Para criar esse bloco especial nós vamos na aba “Mais Blocos” e clicamos no botão “Criar um bloco”.

Ao clicar no botão abrirá uma janela onde você digitará o nome do bloco. Neste caso chamamos o bloco de “movimento-pikachu” porém fique livre em escolher qualquer nome. Além disso também é possível passar parâmetros nesse bloco. Os parâmetros podem receber números, textos e outras coisas. Para este momento nós não utilizaremos nenhuma entrada de parâmetro mas já fica a dica pro desafio que está por vir. Após digitar o nome do bloco clique em “Ok”.

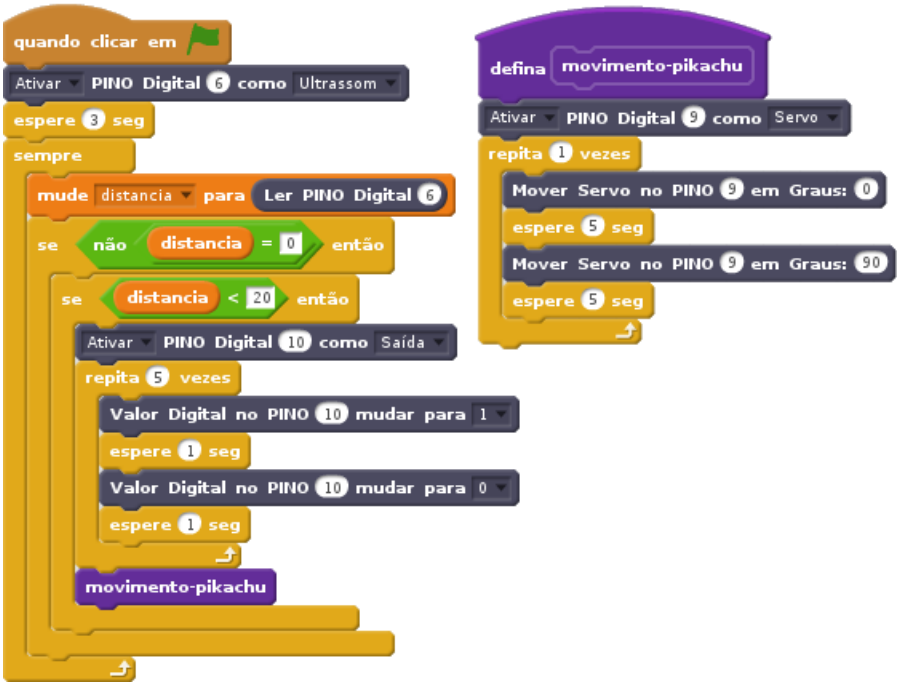


Após a criação do bloco, aparecerá duas opções para utilizarmos. A primeira opção (bloco do lado esquerdo conforme a imagem abaixo) é o nosso bloco criado, onde poderemos chamá-lo sempre que necessário. Já a segunda opção (bloco do lado direito da imagem) é o que representa o nosso bloco criado, ou seja, tudo que for encaixado nele, fará parte da execução do bloco.



Para que o nosso bloco possa executar o movimento do Pikachu, precisamos dar continuidade a programação dele. Para isso pegue a parte do nosso código que dá movimento ao pikachu e encaixe logo abaixo do novo bloco criado, assim como mostrado na imagem abaixo.

Agora que temos o código de nosso bloco “movimento-pikachu” desenvolvido, podemos chamá-lo em nosso código da atividade.



Prontinho, agora temos um código que tem a mesma função de anteriormente porém de uma forma mais organizada. Um bom programador sempre deixa o seu código legível e bem organizado.

Desafios

Agora que você já está craque em programação Scratch, vamos fazer alguns desafios para podermos ver se você realmente pegou bem o conteúdo.

Desafio 1

Este desafio é bem tranquilo, você continuará melhorando nosso código e transformará a parte do código de piscar a pokebola em um novo bloco, assim como foi feito com o movimento do Pikachu. Você seguirá os mesmo passos da criação do bloco “movimento-pikachu”, só que agora teremos alguns pontos a mais a serem considerados. Em vez de fazermos a pokebola piscar 5 vezes, vamos fazer com que ela escolha um intervalo de tentativas para piscar, pode ser entre 1 e 3, por exemplo. Para isso você terá que usar um bloco que gera números aleatórios. Além disso, esse número aleatório terá que ser passado por parâmetro em nosso novo bloco. No final, vamos deixar nosso código organizado e o piscar da pokebola será de forma aleatória.

Desafio 2

Este desafio exigirá um pouco mais de raciocínio lógico para ser escrito, porém também não será difícil.

Vamos acrescentar um nível de dificuldade ao nosso Pokémon, onde dependendo desta dificuldade o Pokémon não será tão fácil de ser capturado. As regras das dificuldades serão as seguintes:

- **Nível 1**: Para a primeira tentativa de captura, o Pókemon já deve "cair", ou seja, ser capturado.
- **Nível 2**: Para um pokémon deste nível de dificuldade ser capturado, ele deve "cair" e levantar uma vez.
- **Nível 3**: Para um pokémon deste nível de dificuldade ser capturado, ele deve "cair" e levantar duas vezes consecutivas.

ANEXO I - QUESTIONÁRIO PÓS-REALIZAÇÃO DA UNIDADE INSTRUCIONAL (PROFESSOR)

Sobre você			
Questionário Professor			
Idade:34			
Profissão:Professor de Física, Empresário			
Grau de escolaridade:Superior Completo			
Qual a disciplina que você leciona(ou) no Ensino Fundamental? CIências(FSC/QMC)			
Você ensina ou ensinou computação física (utilização de hardware)? Em caso afirmativo, explique (público, abordagem, resumo técnico):	X Sim	☐ Não	Sou professor de física e robótica com mais de 10 anos de atuação. Já trabalhei com vários colégios particulares em atividades com o Lego Mindstorms e robótica com componetes da base Arduino.
Você ensina ou ensinou programação? Em caso afirmativo, apresente o contexto (público, linguagens, softwares utilizados):	X Sim	☐ Não	Ensinei programação para todas as faixas etárias com diferentes linguagens e abordagens. Linguagens Logo, C, Java, Python etc... em plataformas Windows, Linux e Unix.
Você possui alguma formação na área de Tecnologia da Informação/Ciências da Computação? Em caso afirmativo, explique:	X Sim	☐ Não	Sou físico licenciado pela UFSC. Programo computadores desde os 13 anos de idade. Comecei com a linguagem visual basic e depois passei por muitas outras, algumas mais antigas como Pascal ou Fortrann e outras mais modernas com Python. Fiz iniciação científica em astrofísica programando um telescópio robótico na UFSC. Ensino robótica educacional por mais de uma década em colégios tradicionais da Grande Florianópolis.
Sobre a unidade instrucional			
Em uma escala de 1 a 5 o quanto você considera que a unidade intrucional cumpre com o objetivo de ensinar computação física para crianças e adolescentes? Justifique.	☐ 1 ☐ 2 ☐ 3 ☐ 4 X 5		A unidade instrucional engloba as várias etapas da aprendizagem, com uma abordagem clara e limpa. O aluno que completar a atividade estrá apto a interagir com elementos básicos e essências de computação. Desde as guias de apresentação até o plano de ensino a atividade mostra-se estruturada e planejada em detalhes. O embasamento didático e pedagógico está muito bem definido em todo a unidade.
Em uma escala de 1 a 5 o quanto voce considera cumprido o objetivo de ensinar programação para crianças e adolescentes? Justifique.	☐ 1 ☐ 2 ☐ 3 X 4 ☐ 5		As linguangens de programação possuem aspectos diferenciados e muitas vezes otimizados para certas atividades de nicho. Como por exemplo utilizar PHP para bancos de dados na Web e C++ para jogos eletrônicos. Apesar das diferenças entre as linguagens, encontramos semelhanças na maneira que o código poderá ser escrito, com utilização de controladores de fluxo, variáveis, padrões lógicos. A unidade aqui apresentada me parece ideial para introduzir esses aspectos essênciais da programação.
Em uma escala de 1 a 5 o quanto voce considera a apresentação e contextualização das etapas de desenvolvimento propostas apropriadas para o público alvo? Justifique.	☐ 1 ☐ 2 ☐ 3 ☐ 4 X 5		As etapas foram apresentadas numa sequência lógica, com objetivos explícitos.
Em uma escala de 1 a 5 o quanto voce considera que o material didático engloba um conteúdo apropriado para a faixa etária? Justifique.	☐ 1 ☐ 2 ☐ 3 ☐ 4 X 5		O trabalho envolve um tema de interesse para os alunos dessa faixa etária. E desenvolve todo um procedimento explorando o tema de forma lúdica. Acredito que o material foi muito bem planejado em sua apresentação.
Você acredita que a linguagem de programação escolhida (Scratch) é apropriada? Justifique.	XSim	☐ Não	Como diz o ditado: "Sou suspeito em falar". Utilizo o Scratch desde o começo. O software tem suas raízes no MIT e sabemos que essa instituição possui uma equipe de pesquisadores extremamente bem qualificada.
Em uma escala de 1 a 5 o quanto voce considera a adequação do conteúdo a um público que já foi introduzido ao conteúdo/assunto (já participaram de uma oficina)? Justifique.	☐ 1 ☐ 2 ☐ 3 X 4 ☐ 5		Interessante. O aluno poderá explorar de maneira mais completa as etapas do trabalho, por conhecer alguns elementos que coordenam a execução da unidade.
Em uma escala de 1 a 5 o quanto voce considera que o objetivo de oferecer uma unidade instrucional de baixo custo é cumprido? Justifique.	☐ 1 ☐ 2 ☐ 3 ☐ 4 X 5		Extremamente eficiente. Trabalho com impressoras 3D, e apesar de ter um resultado interessante na robótica educacional, o custo é altíssimo. A solução aqui apresentada me trouxe reflexões sobre os objetivos de ensino e a facilidade na sua aplicabilidade.
Em uma escala de 1 a 5 o quanto voce considera que o tema escolhido é apropriado para a faixa etária? Justifique.	☐ 1 ☐ 2 ☐ 3 ☐ 4 X 5		O tema parece bastante adequado e talvez possa ser substituído e adequado por outro de interesse dos alunos em diferentes situações .
COMENTÁRIOS: O trabalho está muito bem estruturado e fico feliz em ver alunos saindo de sua formação com este nível de engajamento com o conhecimento e a pedagogia. Como profissional da área posso dizer que a unidade está condizente com o que se espera de uma abordagem que terá uma abordagem efetiva. Acredito que os problemas que poderão surgir no desenvolvimento da atividade não são problemas implícitos na atividade, mas sim, situações que acontecem em outras atividades de ensino. Um exemplo desse tipo de "problema" seria o desnível de conhecimento entre os alunos. Alguns alunos podem já ter conhecimentos prévios avançados de programação e computação física, e outros nunca ouviram falar em Aduino.			

ANEXO II - QUESTIONÁRIO PÓS-REALIZAÇÃO DA UNIDADE INSTRUCIONAL (ALUNO)

Sobre você			
Questionário aluno			
Idade: 14			
Você gosta de mexer em computadores?	x Muito	☐ Mais ou menos	☐ Pouco
Você já realizou alguma atividade de computação antes? Se sim, qual?	x Sim ☐ Não	Eu faço um curso no colégio geração, sobre impressora 3d, ai fizemos um robo e nele tivemos que botar um arduino, foi ai que eu fiz.	
Sobre a unidade instrucional			
Você gostou da atividade que acabou de realizar?	x Muito	☐ Mais ou menos	☐ Pouco
Você achou a atividade fácil?	☐ Muito	x Mais ou menos	☐ Pouco
Você achou a atividade divertida?	☐ Muito	x Mais ou menos	☐ Pouco
Você gostaria de realizar mais atividades de programação com computação física?	x Muito	☐ Mais ou menos	☐ Pouco
Você acredita que aprendeu sobre computação física?	x Muito	☐ Mais ou menos	☐ Pouco
Você acredita que aprendeu sobre programação?	x Muito	☐ Mais ou menos	☐ Pouco
O que você mais gostou na atividade?	R: Eu gostei mais da parte da programção		
O que você menos gostou na atividade?	R: Da parte de botar os fios		
O que você aprendeu sobre computação física?	R: Eu aprendi o que é protoboard, o que é a jump femea e o jump macho, aprendi que da para programar pelo scratch e outras coisas		
O que você aprendeu sobre programação?	R: Eu aprendi as variaveis, aprendi como criar um novo bloco, aprendi fazer o gato andar, fazer barulho, girar, aprendi a ver a distancia pelo sensor e os blocos condicionais		