



FUTURO DO TRABALHO, TRABALHO DO FUTURO

PROGRAMAÇÃO DE SOFTWARE EMBARCADO EM IOT

Apostila do aluno



MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÕES



Ricardo Franco – Arquiteto de Soluções
Bruno Agrofoglio Ferreira – Analista de Capacitação Técnica
Vinicius de Souza – Analista de Capacitação Técnica
Autores da apostila

Larissa Jessica Alves – Analista de Suporte Pedagógico Sr.
Revisão da apostila

FIT Instituto de Tecnologia

Sorocaba, maio de 2022.

Autores



Ricardo Franco tem formação em Técnico em Eletrônica, Tecnologia em Automação Industrial, Engenharia de Produção e atualmente é mestrando em Sistemas Eletrônicos.

Possui experiência de mais de 15 anos em desenvolvimento de produtos eletrônicos com sistemas embarcados.



Professor Bruno Agrofoglio Ferreira é formado em Física pela UNICAMP. Já atuou como professor de Física, Ciências, Astronomia, Programação e Robótica nas redes públicas e privada. Autor de obras didáticas colaborativas em Ciências para o Ensino Fundamental I e Física para o Ensino Médio. Membro do Laboratório Hacker de Campinas. Atualmente é

Analista de Capacitação Técnica no Flextronics Instituto de Tecnologia (FIT).



Vinicius de Souza é formado em engenharia elétrica pela Universidade de Sorocaba, onde também realizou uma iniciação científica com o foco em IoT. Trabalha como Analista de capacitação técnica no Flextronics Instituto de Tecnologia (FIT), onde desenvolve conteúdo técnico-científico para capacitação em Internet das coisas. Além disso, é engenheiro eletrônico e desenvolvedor de software em uma empresa de carregadores para veículos

elétricos. Possui experiência em desenvolvimento web, desenvolvimento de software para embarcados e desenvolvimento em nuvem.

APRESENTAÇÃO

A presente apostila é um instrumento teórico que complementa o curso de capacitação de **Programação de Software Embarcado em IoT**, executado pelo FIT Instituto de Tecnologia. Nela, veremos uma introdução sobre os seguintes tópicos:

- Introdução a Lógica de Programação;
- Variáveis e tipos de variáveis;
- Operadores e Expressões Aritméticas;
- Expressões Lógicas;
- Estrutura de Controle;

Este material é baseado em artigos científicos, periódicos, revistas científicas e livros científicos. É extremamente recomendável ao aluno que, ao final da leitura de cada seção, realize os exercícios propostos e acesse os materiais indicados nas referências bibliográficas para aprofundar a leitura desse material e complementar o que foi aprendido no curso.

A apostila está dividida em duas seções, teoria e prática, iniciando com o conteúdo teórico, que tem por objetivo ambientar o leitor ao ambiente de desenvolvimento, e a parte prática, que tem por finalidade demonstrar os conteúdos abordados na teoria.

Desejo a você, prezado aluno, que tenha um excelente curso!!

Boa Leitura !!

INDICAÇÃO DE ÍCONES



Saiba mais: *oferece novas informações que enriquecem o assunto ou "curiosidades" e notícias recentes relacionadas ao tema estudado.*



Exemplos: *descreve exemplos sobre o assunto que está sendo abordado.*



Atenção: *indica pontos de maior relevância no texto.*



Avisos: *oferece avisos referente ao assunto..*

SUMÁRIO

| | |
|--|----|
| Segurança do usuário..... | 8 |
| Teoria- Orientações | 9 |
| 1.5 Termos e definições básicas | 10 |
| 1.6 Introdução a arquitetura de sistemas embarcados..... | 12 |
| 1.7 Introdução a Lógica de Programação | 15 |
| 1.8 Algoritmo | 15 |
| 1.9 Linguagem de programação..... | 18 |
| 1.10 Tabela ASCII | 19 |
| 1.11 Compiladores | 20 |
| 1.12 Variáveis, tipos de variáveis e escopo | 22 |
| Prática com Kit Didático..... | 25 |
| 1.13 Objetivos de Aprendizagem | 25 |
| 1.14 Arduino..... | 26 |
| 1.15 Convenções para facilitar as montagens | 27 |
| Prática com Simulador Autodesk Tinkercad | 29 |
| 1.16 Primeiro acesso: | 29 |
| Objetivos de Aprendizagem..... | 38 |
| Desafio 1: Escrita digital: | 38 |
| Desafio 2: Leitura digital: | 39 |
| Desafio 3: Leitura analógica: | 40 |
| Desafio 4: Escrita analógica: | 41 |
| Desafio 5: Semáforo inteligente:..... | 42 |
| Conclusão..... | 44 |
| Referências..... | 46 |
| Apêndice | 49 |

| | |
|---|----|
| CONTROLE DE REVISÃO DO DOCUMENTO / <i>DOCUMENT REVISION</i> | |
| <i>CONTROL</i> | 63 |



Segurança do usuário

Aviso importante!

Antes de iniciar os experimentos descritos nesta apostila é extremamente importante que alguns cuidados com a segurança sejam tomados:

- Usar somente o material fornecido no kit.
- Montar o kit sobre uma superfície limpa e isolada eletricamente. Mesas e bancadas metálicas não podem ser utilizadas.
- Retirar anéis, pulseiras, correntes e outros objetos condutores ao manusear os componentes eletrônicos.
- Não utilizar nenhuma fonte de alimentação externa para nenhum experimento.

Em caso de aquecimento de algum componente ou liberação de fumaça, desligue a alimentação do circuito retirando o cabo USB do PC imediatamente.

Teoria- Orientações

O objetivo deste curso é possibilitar ao aluno o conhecimento básico para operar um sistema embarcado mínimo, com leitura de sensores, acionamento de saídas e envio de informações através de uma interface sem fio. Apesar de não se aprofundar em nenhum tópico, com o conhecimento obtido o aluno ao final do curso deve ser capaz de entender como um sistema embarcado funciona e modificar os exemplos para criar as próprias aplicações.

Todas as experiências sugeridas tem a intenção de demonstrar para o aluno como um sistema microcontrolado pode ser utilizado em situações cotidianas, utilizando exemplos práticos e criando desafios lógicos para que a curiosidade e criatividade sejam utilizados para resoluções de problemas reais. Não são exigidos conhecimentos específicos de eletrônica para realizar nenhuma das atividades.

O material on-line deve ser utilizado para auxílio das atividades, mas buscas na web podem facilitar a resolução de dúvidas rápidas.

O Arduino foi o escolhido para a ser a placa base deste curso devido a vasta documentação disponível, interface simples com diversos sensores e vários exemplos de utilização.

Em caso de dúvidas utilize a plataforma on-line e seu instrutor lhe auxiliará.

1.5 Termos e definições básicas

Alguns termos e definições serão utilizados de forma recorrente nesta apostila, para facilitar o entendimento do texto, estão listados abaixo os principais termos e seu significado dentro do contexto deste documento.

| Termo | Definição |
|---------------------------------------|--|
| <i>dispositivo/device</i> | Aparelho ou mecanismo destinado à obtenção de certo fim. |
| <i>sistema</i> | Combinação de partes reunidas para concorrerem para um resultado, ou de modo a formarem um conjunto. |
| <i>ferramenta</i> | Conjunto de instrumentos e utensílios empregados em um ofício. |
| <i>rádio</i> | Aparelho transmissor-receptor usado para comunicações. |
| <i>IDE</i> | Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. |
| <i>protoboard - Breadboard</i> | Placa perfurada feita de plástico, que possuem matrizes de contatos elétricos para ligação de componentes eletrônicos. |
| <i>Entrada Digital</i> | Circuito eletrônico que é utilizado para entrada de dados do sistema. Baseado em uma tensão elétrica informa o sistema se a entrada é 1 ou 0 lógico. |
| <i>Entrada Analógica</i> | Circuito eletrônico que é utilizado para entrada de dados do sistema. Baseado em uma tensão elétrica informa o sistema se a entrada esta entre os níveis 0 até 255. |
| <i>Saída digital</i> | Circuito eletrônico que é utilizado para saída de dados do sistema. Pode ser 0 ou 1, ou 0V e 5V. Controlada pelo microcontrolador. |

| | |
|------------------------|---|
| Saída analógica | Circuito eletrônico que é utilizado para saída de dados do sistema. Utilizada em conjunto com PWM, oferece 255 níveis de tensões, que possuem resolução de $\pm 20\text{mV}$ para cada nível, que são somatórias dos níveis anteriores. Controlada pelo microcontrolador. |
| PWM | Modulação por largura de pulso. |
| Duty Cycle | Porcentagem de tempo do sinal em nível 1 em relação a duração total do sinal. |
| frequência | Número de ocorrência de uma onda em um segundo. Medida em Hz. |
| potenciômetro | Resistor variável com um knob para ajuste da sua resistência. |
| led | Diodo emissor de luz. Semi-condutor que quando polarizado emite luz. |
| resistor | Dispositivo eletrônico que limita a passagem da corrente elétrica. Medido em Ohms. |
| chave | Dispositivo utilizado para interromper/conectar duas conexões elétricas. |
| bluetooth | Rádio transmissor/receptor que opera em 2,4 GHz e que segue as regras estabelecidas por protocolo de mesmo nome. |
| IoT | Internet das Coisas. Termo utilizado como adjetivo de algum equipamento que possui algum link de comunicação com a Internet ou rede interna. |
| Arduino | Placa eletrônica utilizada para programação embarcada. É um hardware simples, que possibilita o entendimento de conceitos de programação, eletrônica, controle e física. |
| Jumper | Pedaço de fio que serve para conectar dois pontos no protoboard. |

| | |
|-------------------------|--|
| <i>pull-up</i> | Resistor com um dos terminais conectado ao VCC. |
| <i>loopback</i> | Sinal que volta para sua origem sem nenhuma modificação. |
| <i>Pull-down</i> | Resistor com um dos terminais conectado ao GND. |

Tabela 1- Termos e definições básicas utilizadas na apostila. Fonte: autoria própria.

1.6 Introdução a arquitetura de sistemas embarcados

Os computadores atuais, assim como smartphones e smartwatches, são baseados na arquitetura proposta por John Von Neumann em 1936. Nesta arquitetura, os computadores são divididos em blocos funcionais fundamentais. São eles:

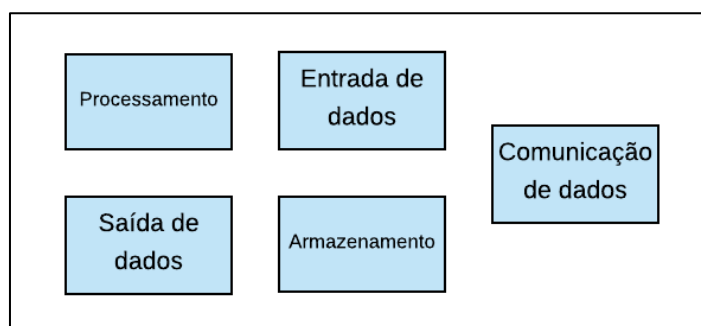


Figura 1- Blocos de arquitetura de um computador genérico. Fonte: autoria própria.

Os componentes necessários para operação do computador, como entrada, saída e armazenamento, são interligados por vias de dados à Unidade Central de Processamento (CPU) ou processador.

Essa estrutura é padrão para dispositivos que rodam sistemas operacionais comuns e também sistemas embarcados.

Já os microcontroladores utilizados em sistemas embarcados, em sua maioria, possuem arquitetura Harvard, uma evolução da arquitetura de Von Neumann, onde pode-se ler o programa e também os dados simultaneamente, deixando o microcontrolador mais rápido durante a execução.

Mas a final, o que é um sistema embarcado?

Um sistema embarcado pode ser definido como ***um computador construído especificamente para uma finalidade***. A definição independe do *hardware* do processador, que poder ser um processador de 64-bits ou um microcontrolador de 8 bits.

Segundo o portal Embarcados, o primeiro sistema embarcado de conhecimento público é o ACG (Apollo Guidance Computer). Ele era um microcomputador de 16 bits, 15 de dados e 1 de paridade, que operava a 1,024 MHz e que possuía 2048 words de memória RAM e 36864 words de memória de programa. Foi desenvolvido em um laboratório de instrumentação do MIT, nos Estados Unidos e fabricado pela Raytheon a partir de 1966. Ele operava em 1,024 MHz e era responsável pelo controle das espaçonaves Apollo, que levaram o homem à Lua nas décadas de 60 e 70. O AGC, no entanto, não possuía processador, era todo feito com portas NOR.



Figura 2 - ACG case and keyboard. Fonte: NASA.

A escolha de se utilizar portas NOR deve-se ao fato de serem conhecidas como porta universal, a partir dela podemos montar qualquer circuito lógico.

Um sistema embarcado pode rodar em computadores comuns, porém geralmente roda em máquina com *hardware* dedicado, pois atua com um número limitado de tarefas se comparado a um computador genérico.

Sistemas embarcados podem ser encontrados em equipamentos simples, como em termômetros de mesas e lâmpadas inteligentes, mas não são limitados a estas aplicações, pois estão presentes em sistemas de alta complexidade, como cockpits de carros de fórmula 1, foguetes e satélites.

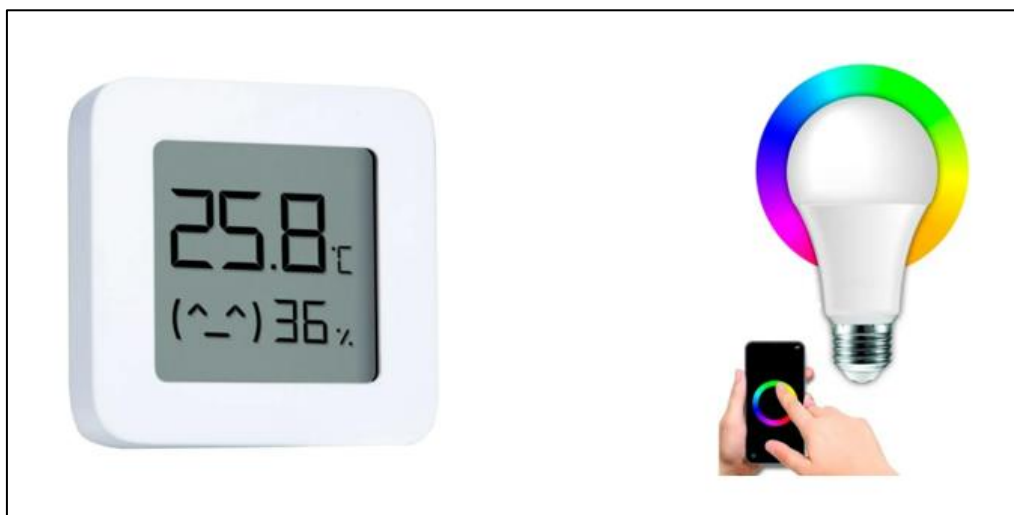


Figura 3 - Sistemas Embarcados simples. Fonte: adaptado de Midobrasil e Amazon.

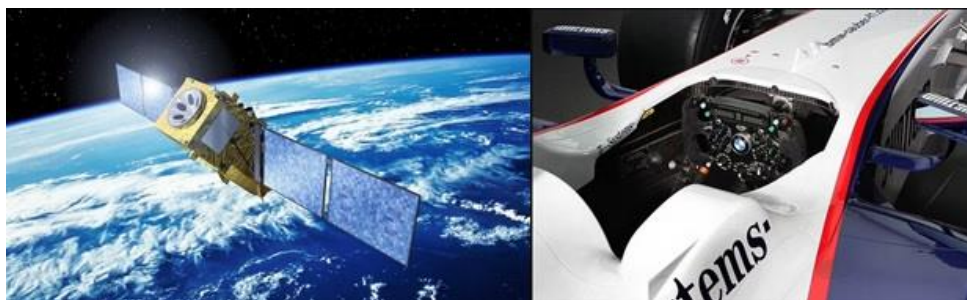


Figura 4 - Sistemas Embarcados complexos. Fonte: adaptado de Inpe.

1.7 Introdução a Lógica de Programação

De acordo com Schildt e Mayer (2008), uma definição simples de Lógica de Programação pode ser dada por:

Escrever um programa de computador que execute um determinado roteiro (algoritmo) utilizando uma linguagem de programação que seja adequada a máquina ou dispositivo que será programado.

Apesar de ser uma definição simples, ela deixa explícita duas informações:

- a) O programador deve ter em mente um roteiro (algoritmo) que seu código deve executar;
- b) Deve utilizar uma linguagem de programação adequada para a máquina.

1.8 Algoritmo

O roteiro, ou algoritmo, nada mais é do que uma lista de instruções que devem ser executadas para que determinada tarefa seja realizada pelo programa.

Exemplo abaixo é de um algoritmo explicando o processo de passar manteiga em um pão:

:

```

/*****/

Algoritmo para passar manteiga em um pão:

/*****/

Início:
Abrir o pote de manteiga.
Pegar uma faca.
Pegar um pão.
Cortar o pão.
Passar manteiga no pão.
Fim

/*****/

```

Figura 5 - Algoritmo simples. Fonte: autoria própria.

Quando um fluxograma fica complexo o bastante para não ser viável descrever as ações em poucas linhas, é feita a utilização de uma estrutura de imagens para auxiliar o entendimento. A esta estrutura damos o nome de Fluxograma.

Exemplo de fluxograma explicando o processo de passar manteiga em um pão:

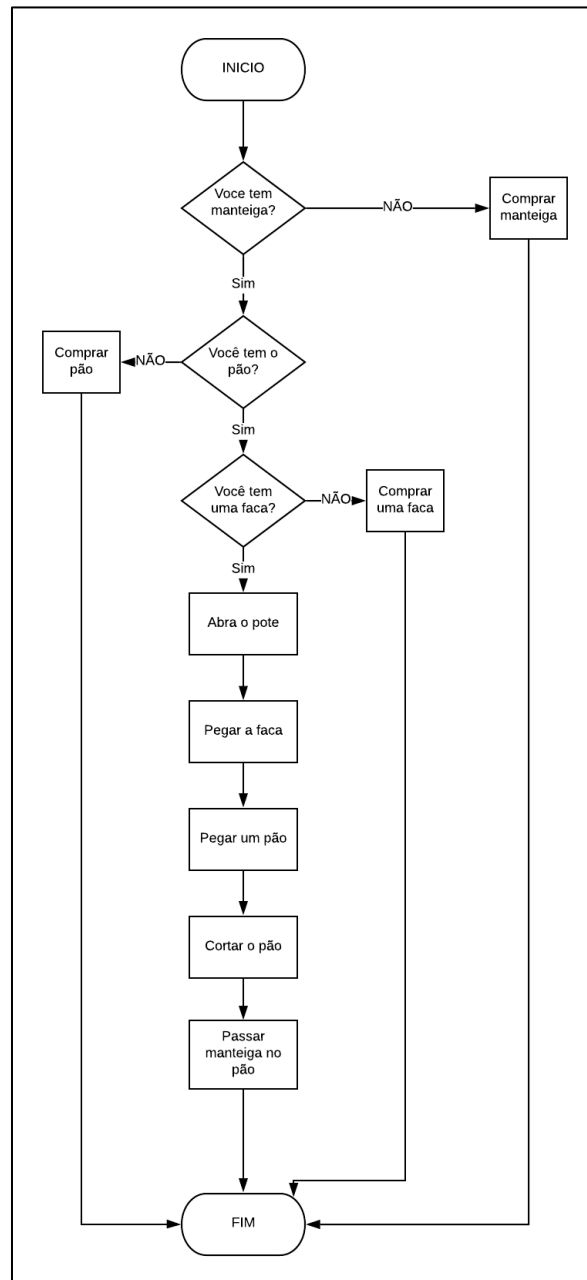


Figura 6 - Fluxograma simples. Fonte: autoria própria.

No fluxograma acima foi adicionado um novo elemento, a tomada de decisão. Caso o usuário não tenha a manteiga, o pão ou a faca, o programa não é executado. O exemplo apresentado utiliza uma lógica programada no algoritmo para tomar uma decisão baseada nas entradas do sistema.

1.9 Linguagem de programação

De acordo com a Wikipedia, temos que a “linguagem de programação é um método padronizado, formado por um conjunto de regras sintáticas e semânticas, de implementação de um código fonte - que pode ser compilado e transformado em um programa de computador,^[1] ou usado como script interpretado - que informará instruções de processamento ao computador”. Atualmente temos cerca de 9000 linguagens de programação no mundo, sendo as mais “conhecidas” cerca de 50 destas linguagens.

Exemplos de Linguagem de programação:

| | |
|--|--|
| <ul style="list-style-type: none"> • ASP • ActionScript • C/C++ • C# • Pascal/Object Pascal • Euphoria • Java • JavaScript | <ul style="list-style-type: none"> • Lua • MATLAB • PHP • Python • R • Ruby • Tcl • Basic/Visual Basic |
|--|--|

Figura 7 - Exemplos de linguagens de programação. Fonte: autoria própria.

Ok, mas o que isso quer dizer?

Você já parou para pensar quantos idiomas são falados ao redor do planeta? Segundo a revista Ethnologue, temos cerca de 34 idiomas listados contendo mais de 45 milhões de falantes no mundo. O ser humano utiliza a linguagem para se comunicar, podendo ser oralizada, visual, por gestos, sinais, entre outras.

Assim a linguagem de programação nada mais é do que um meio desenvolvido para que o homem possa se comunicar com uma máquina! Computadores e seus derivados são máquinas digitais que utilizam o sistema binário para realizar todas as suas operações. De forma geral isto significa que um computador sempre escreve ou lê 0 ou 1.

Portanto foi necessário criar uma forma de converter a linguagem que fosse inteligível por seres humanos para zeros e uns que seriam executados pelas máquinas.

1.10 Tabela ASCII

Antes de 1960 cada computador utilizava uma regra diferente para representar estes caracteres. Segundo o site Techtudo, isso mudou em 1960. Robert W. Bemer propôs um código que seria uma solução para unificar a representação de caracteres alfanuméricos em computadores. Ele tabelou estes valores e batizou de ASCII.

O código ASCII nasceu para se tornar comum entre todas as máquinas. Seu nome- ASCII vem do inglês "American Standard Code for Information Interchange" ou "Código Padrão Americano para o Intercâmbio de Informação". Ele é baseado no alfabeto romano e sua função é padronizar a forma como os computadores representam letras, números, acentos, sinais diversos e alguns códigos de controle.

No ASCII existem apenas 95 caracteres que podem ser impressos, eles são numerados de 32 a 126 sendo os caracteres de 0 a 31 reservados para funções de controle, ou seja, funções de computador. Alguns caracteres acabaram caindo em desuso pois eram funções específicas para computadores da época como o Teletype (máquinas de escrever eletro-mecânicas), fitas de papel perfurado e impressoras de cilindro.

Assim, a letra "A" para um computador é vista da seguinte forma:

| <u>Bin</u> | <u>Oct</u> | <u>Dec</u> | <u>Hex</u> | <u>Sinal</u> |
|------------|------------|------------|------------|--------------|
| 0100 0001 | 101 | 65 | 41 | A |

Figura 8 - Carácter "A" de acordo com a tabela ASCII. Fonte: ASCII.



O site abaixo é um conversor de código ASCII para binário. Escreva uma frase simples e verifique como o compilador enxerga o seu texto em linguagem de máquina!
<http://www.unit-conversion.info/texttools/convert-text-to-binary/>

1.11 Compiladores

Um compilador é um programa de sistema, que traduz um programa descrito em uma linguagem de alto nível para um programa equivalente em código de máquina para um processador. Em geral, um compilador não produz diretamente o código de máquina, mas sim um programa em linguagem simbólica (assembly) semanticamente equivalente ao programa em linguagem de alto nível. O programa em linguagem simbólica é então traduzido para o programa em linguagem de máquina através de montadores.

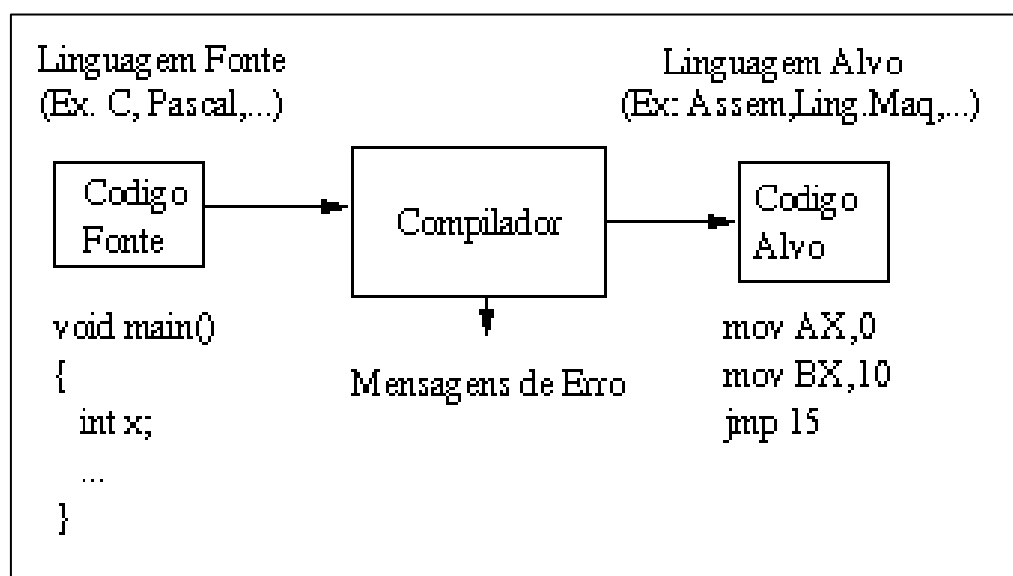


Figura 9 - Blocos de um código sendo compilado. Fonte: GPEC.

Para desempenhar suas tarefas, um compilador deve executar duas atividades básicas. A primeira atividade é a análise do código fonte,

onde a estrutura e o significado do programa de alto nível são reconhecidos. A segunda atividade é a síntese, que traduz o programa a seu equivalente em linguagem simbólica. Embora conceitualmente seja possível executar toda a análise e apenas então iniciar a síntese, em geral as duas atividades ocorrem praticamente em paralelo.

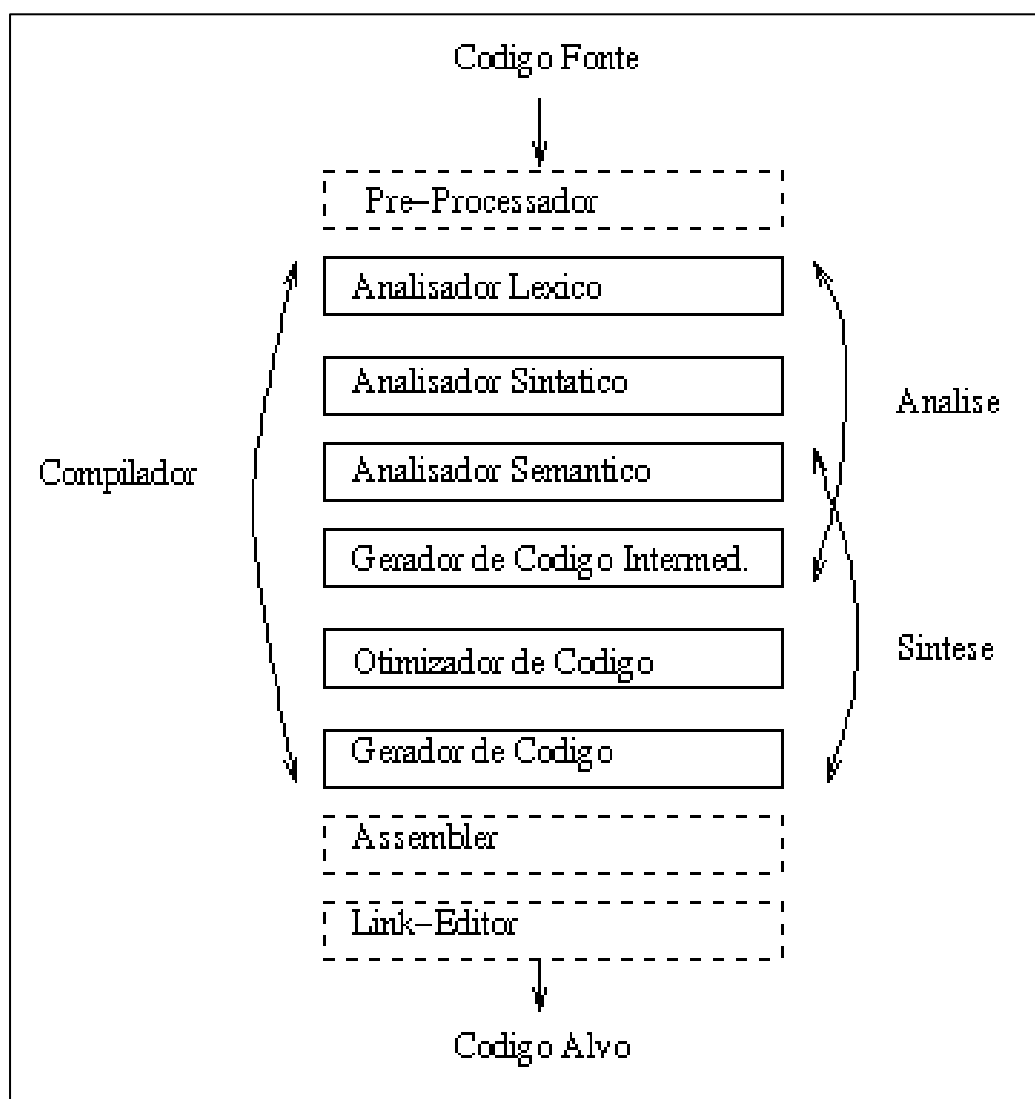


Figura 10 - Estados de um compilador. Fonte: GPEC.

1.12 Variáveis, tipos de variáveis e escopo

1.12.1 Definição de uma variável

De acordo com Schildt, uma variável é um local nomeado na memória que é usado para manter um valor que pode ser modificado pelo programa. Todas as variáveis devem ser declaradas antes de serem usadas. A forma geral de uma declaração é vista abaixo:

type variable list;

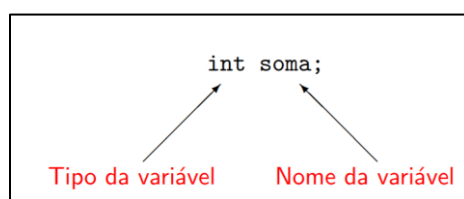


Figura 11 - Exemplo da formação de uma variável. Fonte: autoria própria.

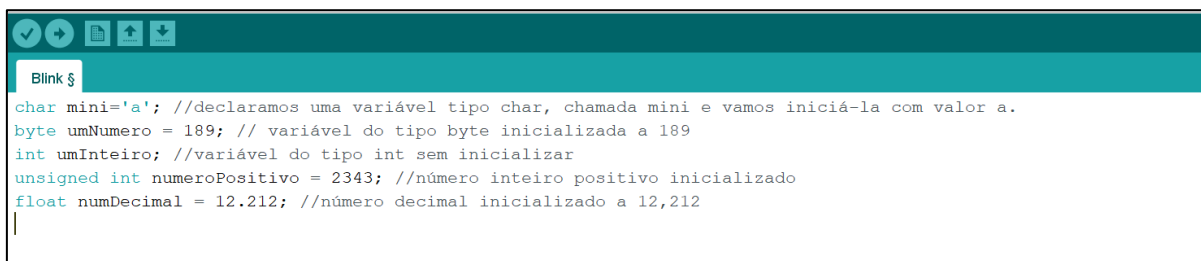
1.12.2 Tipos de variáveis

No Arduino, as variáveis suportadas pelo compilador são:

| Variáveis | Características |
|------------------------------|---|
| char | São utilizados para armazenar caracteres e ocupam um byte. |
| byte | Podem armazenar um número entre 0 e 255. |
| int | Ocupam 2 bytes (16 bits) e tanto armazenam um número entre 2-15 e 2 ¹⁵ -1, ou seja, entre -32,768 e 32,767. |
| unsigned int | Também ocupam 2 bytes, mas como não possuem sinal, podem tomar valores entre 0 e 2 ¹⁶ -1, ou seja, entre 0 e 65,535. |
| long | Ocupa 32 bits (4 bytes), de -2,147,483,648 até 2,147,483,647. |
| unsigned long e float | Números decimais que ocupam 32 bits (4 bytes). Podem tomar valores entre -3.4028235E+38 e +3.4028235E+38. |
| double | Também armazena números decimais, mas possuem 8-bytes (64 bit). |

Tabela 2 - Tipos de variáveis utilizadas na IDE do Arduino: Fonte: adaptado de Diwo.

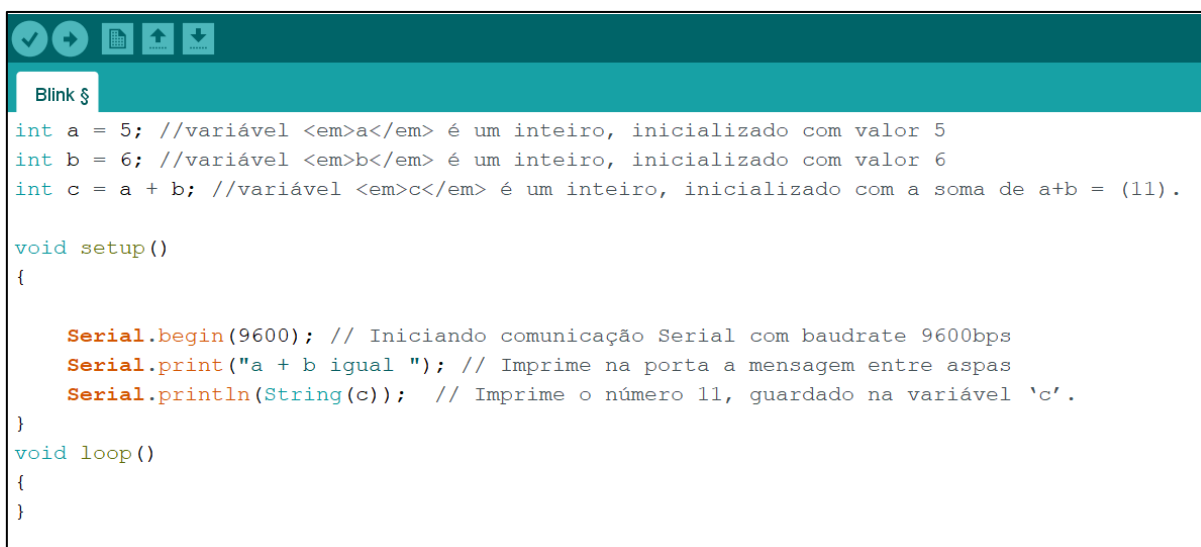
Exemplo de variáveis declaradas no Arduino:



```
char mini='a'; //declaramos uma variável tipo char, chamada mini e vamos iniciá-la com valor a.
byte umNumero = 189; // variável do tipo byte inicializada a 189
int umInteiro; //variável do tipo int sem inicializar
unsigned int numeroPositivo = 2343; //número inteiro positivo inicializado
float numDecimal = 12.212; //número decimal inicializado a 12,212
```

Figura 12 - Exemplo de declaração de variáveis no Arduino. Fonte: adaptado de Diwo.

Exemplo de um programa que soma dois números:



```
int a = 5; //variável <em>a</em> é um inteiro, inicializado com valor 5
int b = 6; //variável <em>b</em> é um inteiro, inicializado com valor 6
int c = a + b; //variável <em>c</em> é um inteiro, inicializado com a soma de a+b = (11).

void setup()
{
    Serial.begin(9600); // Iniciando comunicação Serial com baudrate 9600bps
    Serial.print("a + b igual "); // Imprime na porta a mensagem entre aspas
    Serial.println(String(c)); // Imprime o número 11, guardado na variável 'c'.
}

void loop()
{
}
```

Figura 13 - Programa que soma dois números e exibe o resultado na tela. Fonte adaptado de Diwo.

2.8.3 Escopo

De acordo com a página da IDE Arduino, escopo é uma propriedade da linguagem C++. Esta propriedade diz ao compilador se a variável é do tipo global ou do tipo local.

Uma **variável global** pode ser vista por todas as funções de um programa. Diferentemente das **variáveis locais** que são visíveis apenas nas funções nas quais são declaradas. No ambiente Arduino, qualquer variável declarada fora de uma função (ex. `setup()`, `loop()`, etc.), é uma variável `_global_`.

Quando programas começam a ficar muito longos e complexos, variáveis locais são uma forma útil de garantir que apenas uma função tenha acesso as suas próprias variáveis. Isso previne erros de programação quando uma função inadvertidamente modifica variáveis usadas por outra função. Às vezes também é vantajoso declarar e inicializar uma variável dentro de um loop `for`. Isso cria uma **variável local**, que pode ser acessada apenas dentro do próprio bloco do loop `for`.

Prática com Kit Didático

No caso de você possuir um kit didático com Arduino, você poderá seguir as instruções a seguir, realizando a montagem física com seu microcontrolador. Não se preocupe caso não possua a parte física do curso com você, utilizaremos para todos os próximos exercícios simuladores digitais, incluindo o simulador online da Autodesk, o Tinkercad. Neste caso, vá para o capítulo 4 desta apostila.



Figura 14 – Prática com kit físico. Fonte: Getting Started with Arduino.

1.13 Objetivos de Aprendizagem

- Utilizar o hardware Arduino;
- Utilizar a IDE Arduino;
- Escrever, carregar e executar programas básicos;
- Realizar montagens eletrônicas com o protoboard.

1.14 Arduino

A placa do Arduino Uno será utilizada em todas as montagens. Abaixo uma imagem da placa do Arduino.

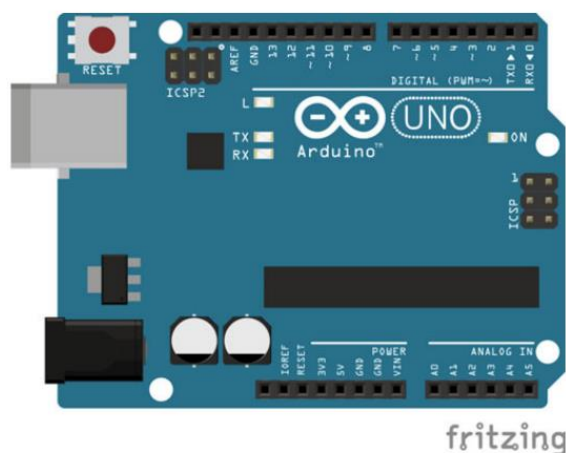


Figura 15 - Placa Arduino Uno. Fonte: autoria própria.

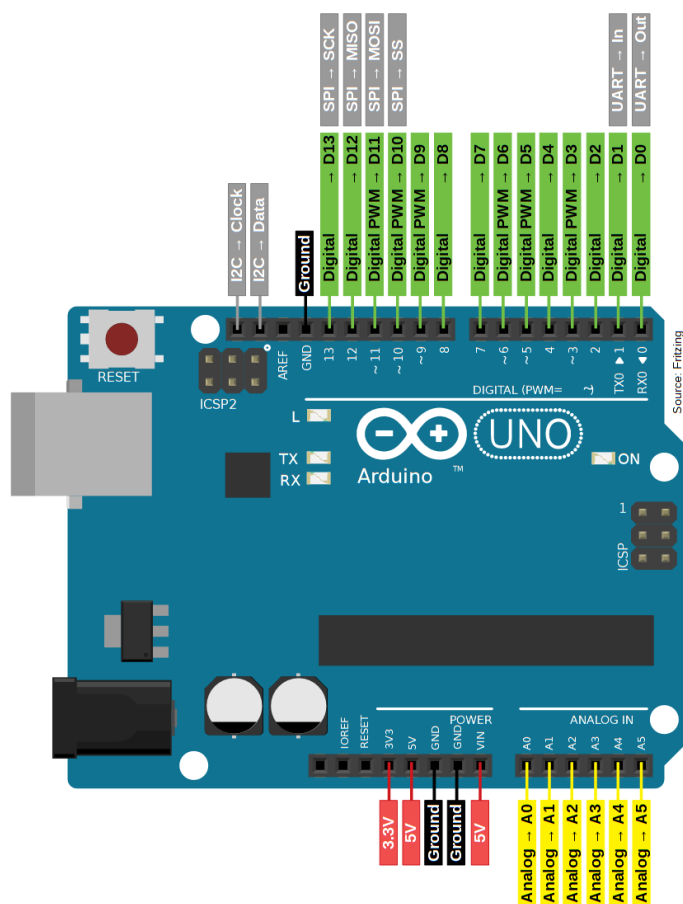


Figura 16 - Descrição dos pinos do Arduino. Fonte: Medium.

A placa Arduino possui as seguintes especificações técnicas:

| | |
|-------------------------------|---|
| Microcontrolador | ATmega328P |
| Tensão de Operação | 5V |
| Tensão de alimentação externa | 7-12V |
| Tensão máxima DC (Vmax) | 6-20V |
| Digital I/Os | 14 (com 6 podendo ser PWM output) |
| PWM Digital I/O Pins | 6 |
| Analog Input Pins | 6 |
| Corrente DC por I/O Pin | 20 mA |
| Corrente DC para pino de 3V3 | 50 mA |
| Flash Memory | 32 KB (ATmega328P) of which 0.5 KB used by bootloader |
| SRAM | 2 KB (ATmega328P) |
| EEPROM | 1 KB (ATmega328P) |
| Clock Speed | 16 MHz |
| LED_BUILTIN | 13 |
| Comprimento | 68.6 mm |
| Largura | 53.4 mm |
| Peso | 25 g |

Tabela 3 - Especificação técnica Arduino. Fonte: autoria própria.

1.15 Convenções para facilitar as montagens

- **Fio Preto:** GND – Ligado ao barramento de alimentação da protoboard
- **Fio Vermelho:** +5V – Ligado ao periférico alimentado
- Todas as outras cores podem ser utilizadas para qualquer outro tipo de conexão.
- As cores utilizadas nas figuras são ilustrativas, cada kit pode vir com jumpers com cores distintas das indicadas nas figuras.
- Não é necessário desmontar o circuito do bluetooth, pois ele será utilizado várias vezes.

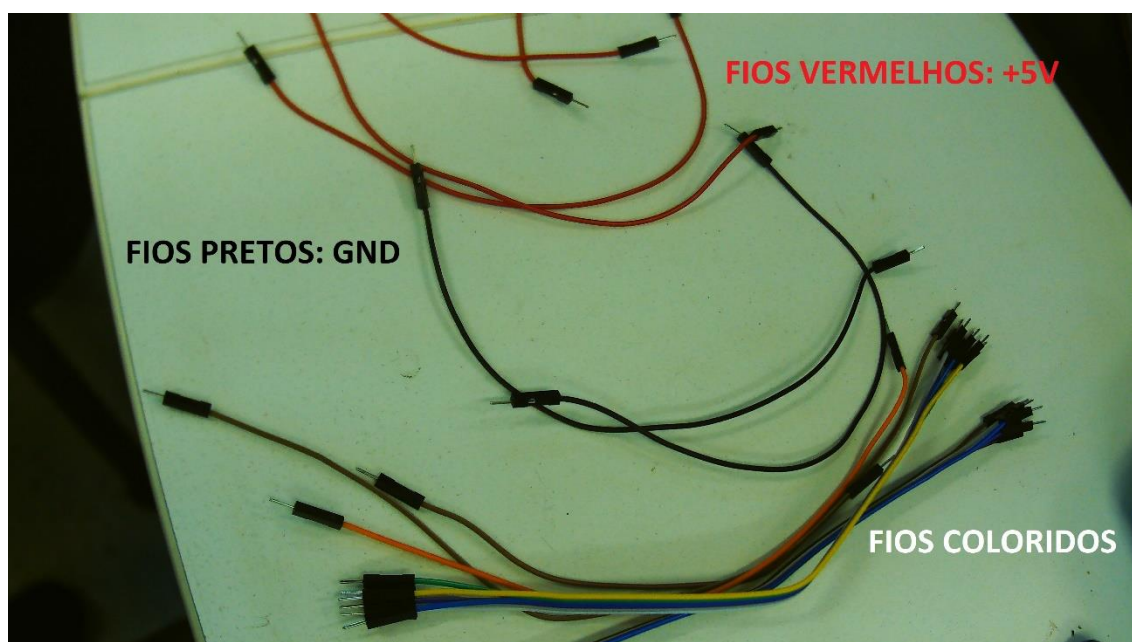


Figura 17 - Fios para +5V, GND e outros sinais. Fonte: autoria própria.

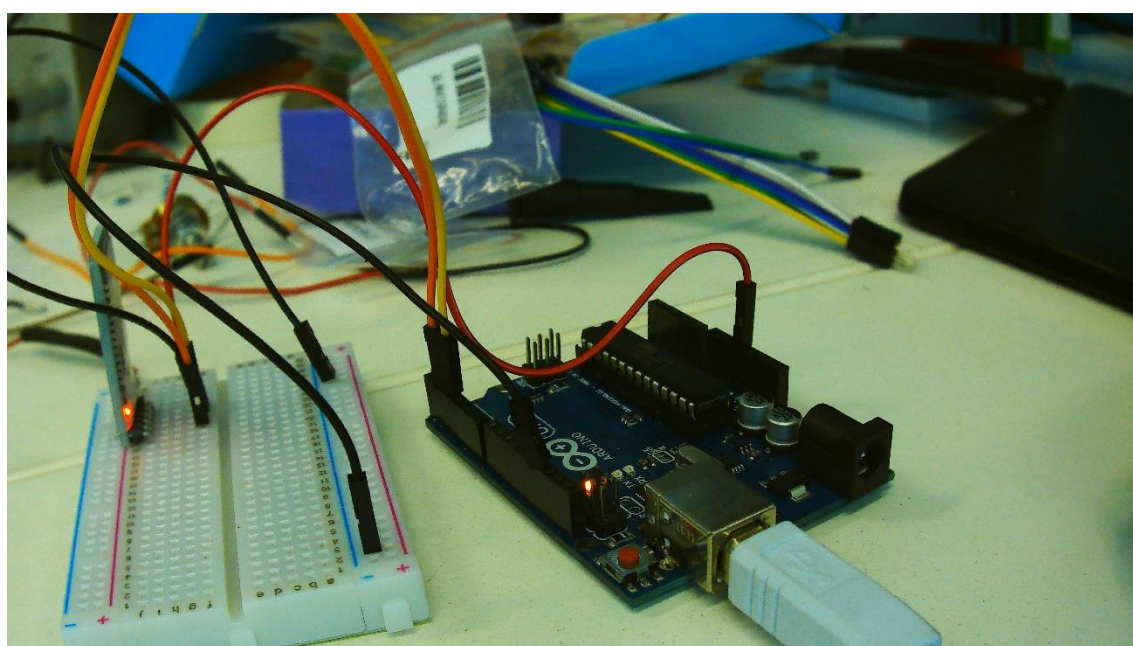


Figura 18 - Exemplo de montagem utilizando os jumpers do kit. Fonte: autoria própria.

Prática com Simulador Autodesk Tinkercad

O simulador Tinkercad permite simular montagens e códigos desenvolvidos para Arduino em uma plataforma Web.

Para utilizar este serviço é necessário criar uma conta na Autodesk ou utilizar uma conta google.

1.16 Primeiro acesso:

No seu web browser digite: <https://www.tinkercad.com/>

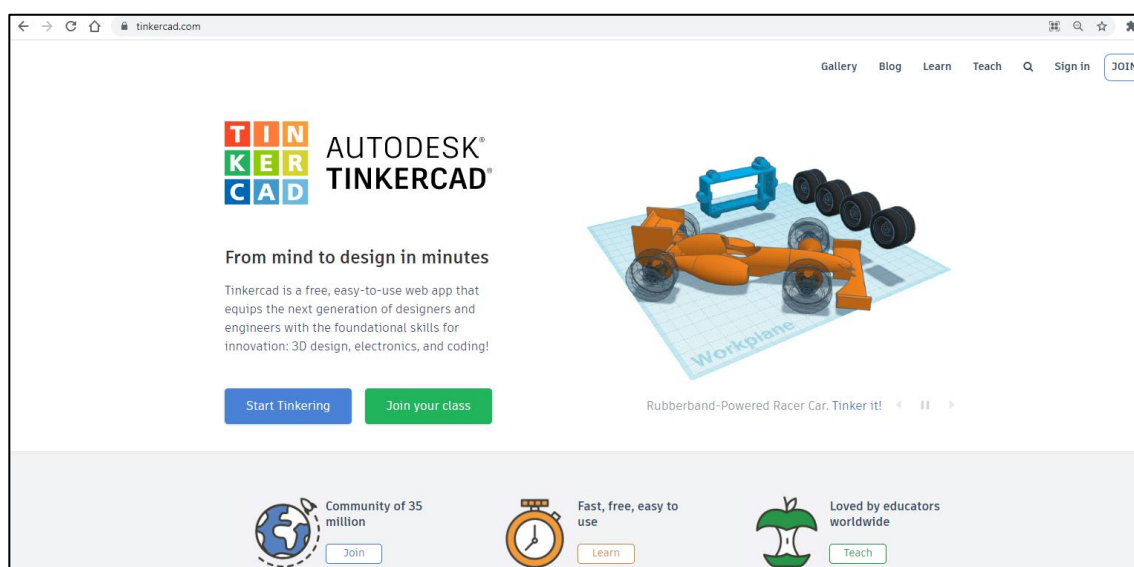


Figura 19 - Tela inicial do Site. Fonte: TINKERCAD.

No canto superior direito clique em “JOIN NOW”:

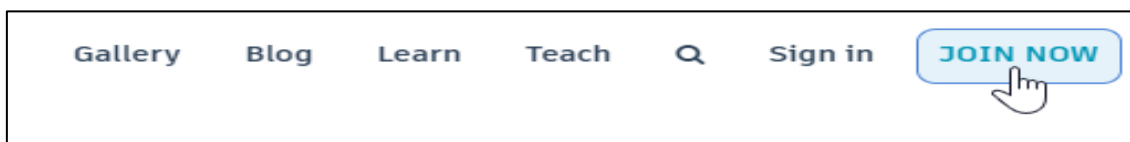


Figura 20 - Botão JOIN NOW. Fonte: TINKERCAD.

Se você já tem uma conta clique em “Sign In”, caso contrário, clique em “Create a personal account”:

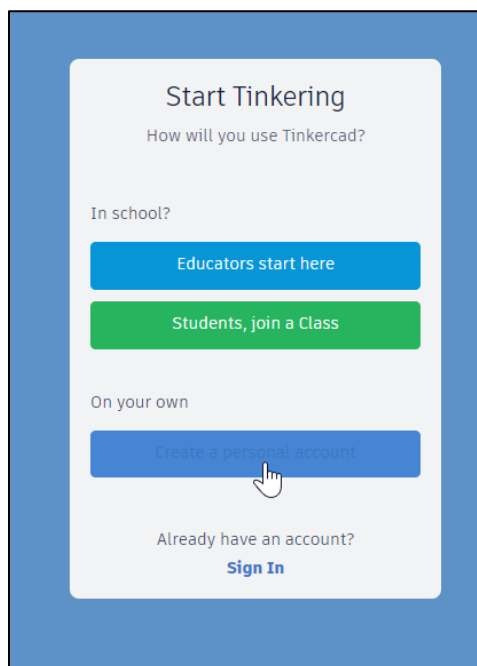


Figura 21 - Create a personal account. Fonte: TINKERCAD.

Escolha uma das três opções: via e-mail, via conta google ou via conta apple. Nos exemplos a seguir utilizaremos uma conta google:

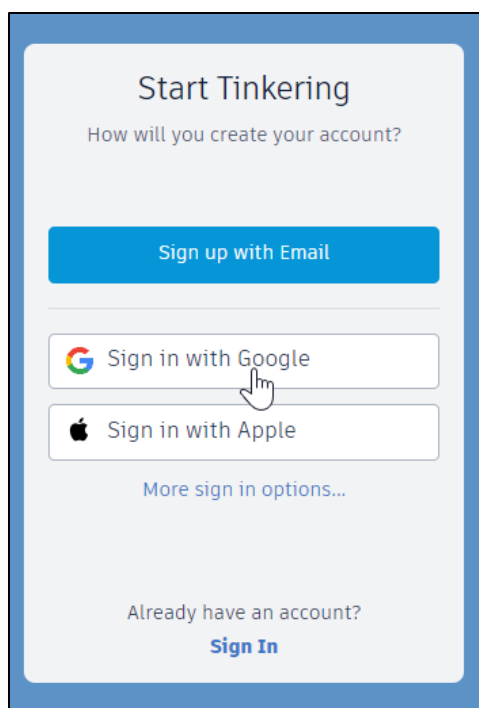


Figura 22 - Escolha do tipo de conta. Fonte: TINKERCAD.

Clique sobre a conta que deseja utilizar:

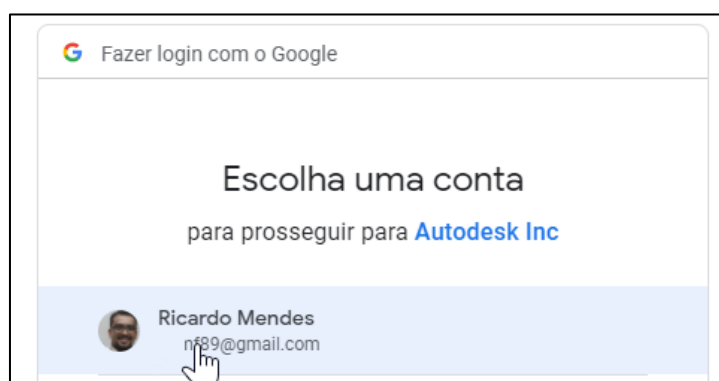


Figura 23 – Exemplo de uso de conta Google. Fonte: TINKERCAD.

Você será direcionado para a tela abaixo:

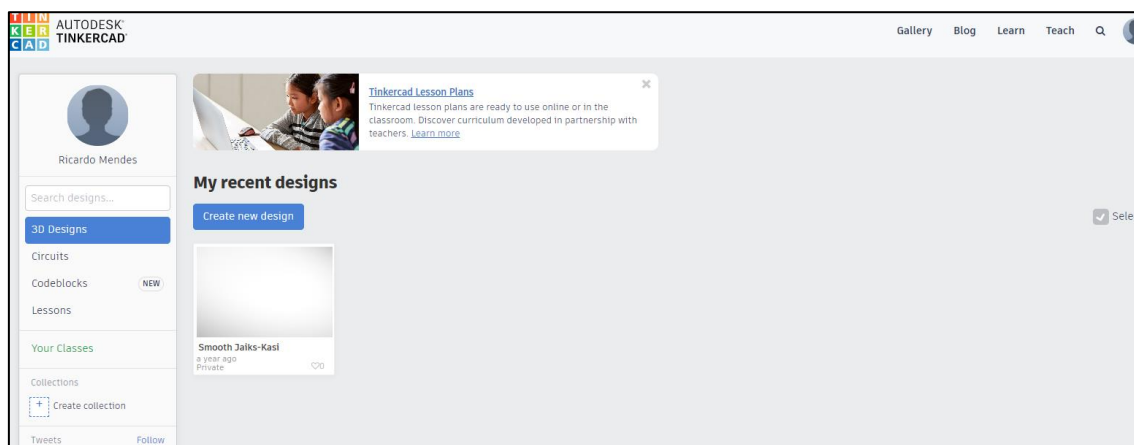


Figura 24 -Tela inicial My recent designs.Fonte: TINKERCAD.

No canto lateral esquerdo, clique sobre o botão “Circuits”:

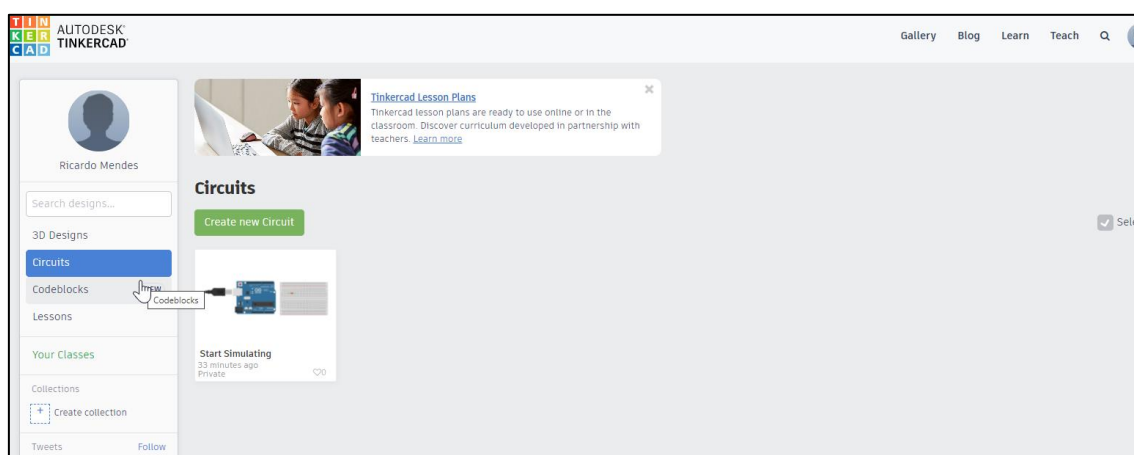


Figura 25 - Escolha da opção Circuits. Fonte: TINKERCAD.

Depois, clique no botão verde “Create new Circuit”:

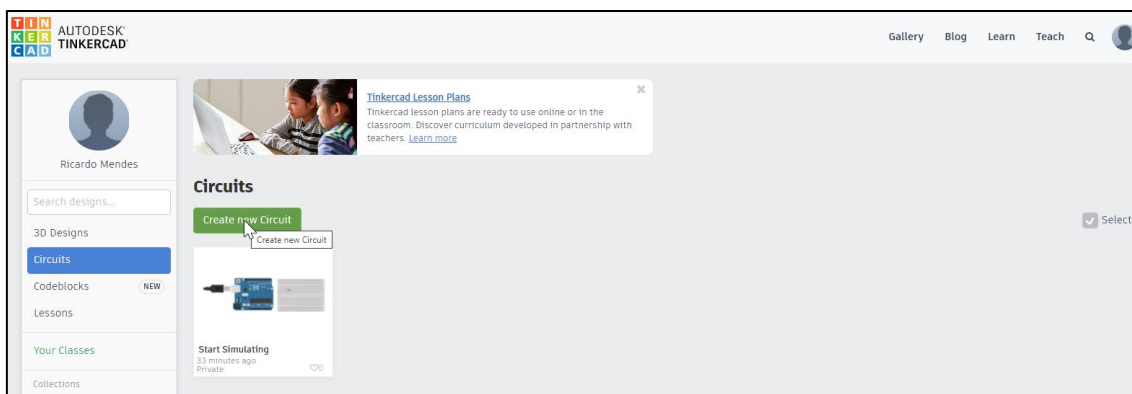


Figura 26 - Clicando sobre o botão Create New Circuit. Fonte: TINKERCAD.

Você será direcionado(a) para a seguinte tela:

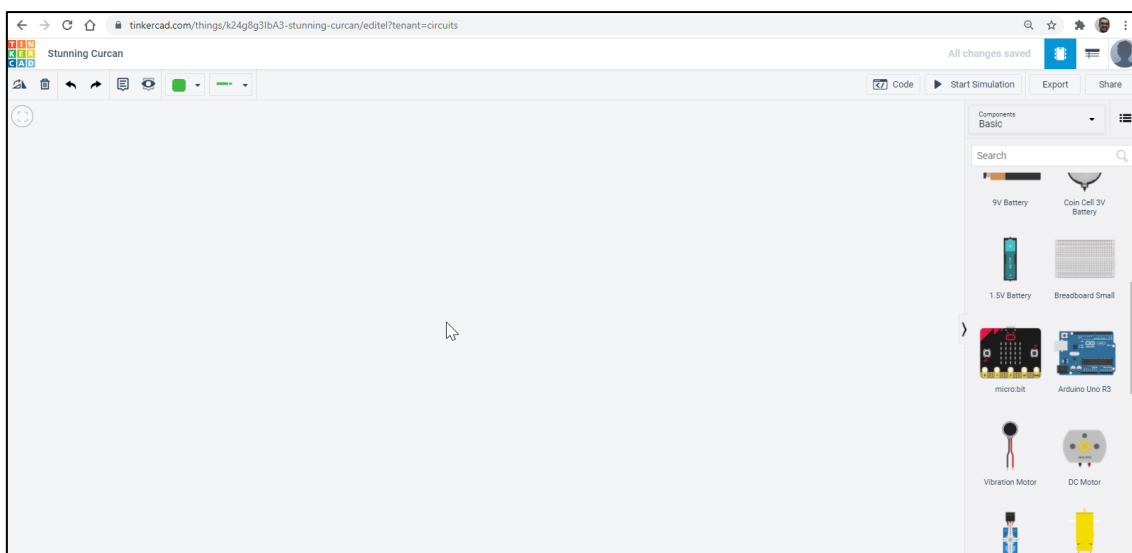


Figura 27 - Tela de criação de circuito eletrônico. Fonte: TINKERCAD.

No canto superior esquerdo, clique na aba “Components Basic” e Mude para “Arduino”

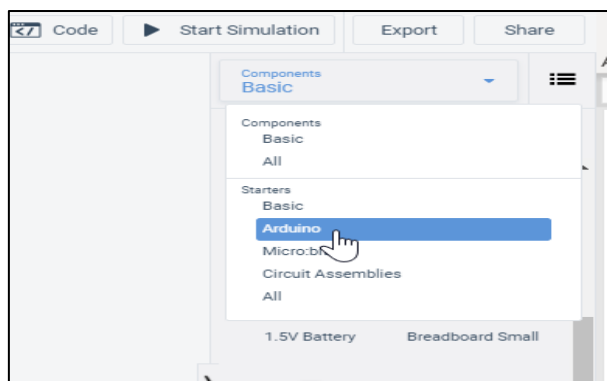


Figura 28 - Escolha da placa Arduino. Fonte: TINKERCAD.

As seguintes opções de circuitos serão apresentadas:

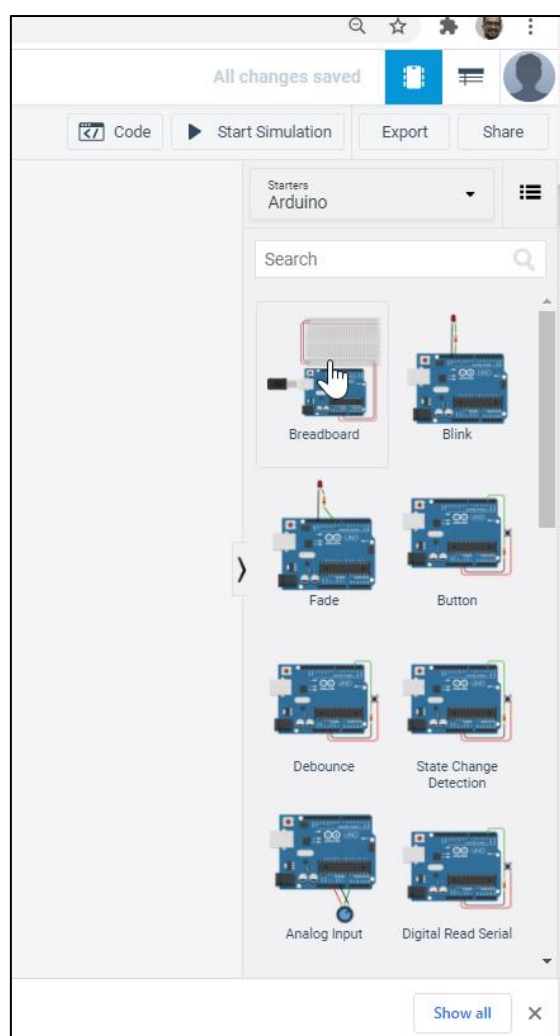


Figura 29 - Escolha das montagens padrões com Arduino. Fonte: TINKERCAD.

Escolha a opção “Breadboard”:

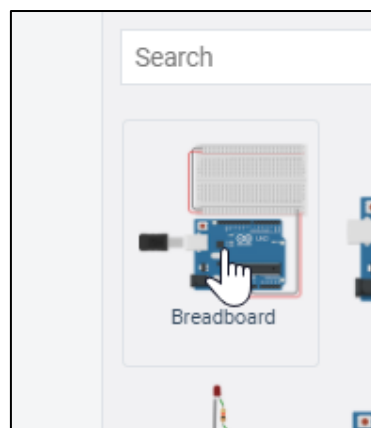


Figura 30 - Seleção da placa Arduino com Breadboard. Fonte: TINKERCAD.

Um Arduino conectado a um protoboard aparecerá na tela, você controla o posicionamento com o ponteiro do mouse, selecione a posição que ele deve ficar na tela confirmando com o botão esquerdo do mouse:

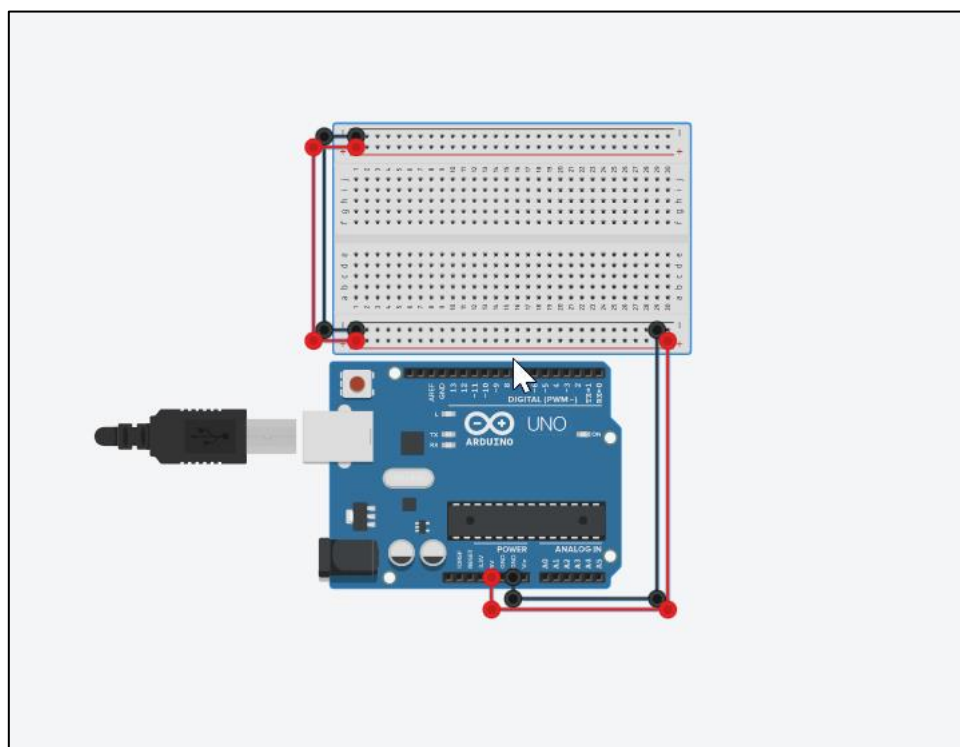


Figura 31 - Placa selecionada aguardando posicionamento final. Fonte: TINKERCAD.

A sua tela deve ficar assim:

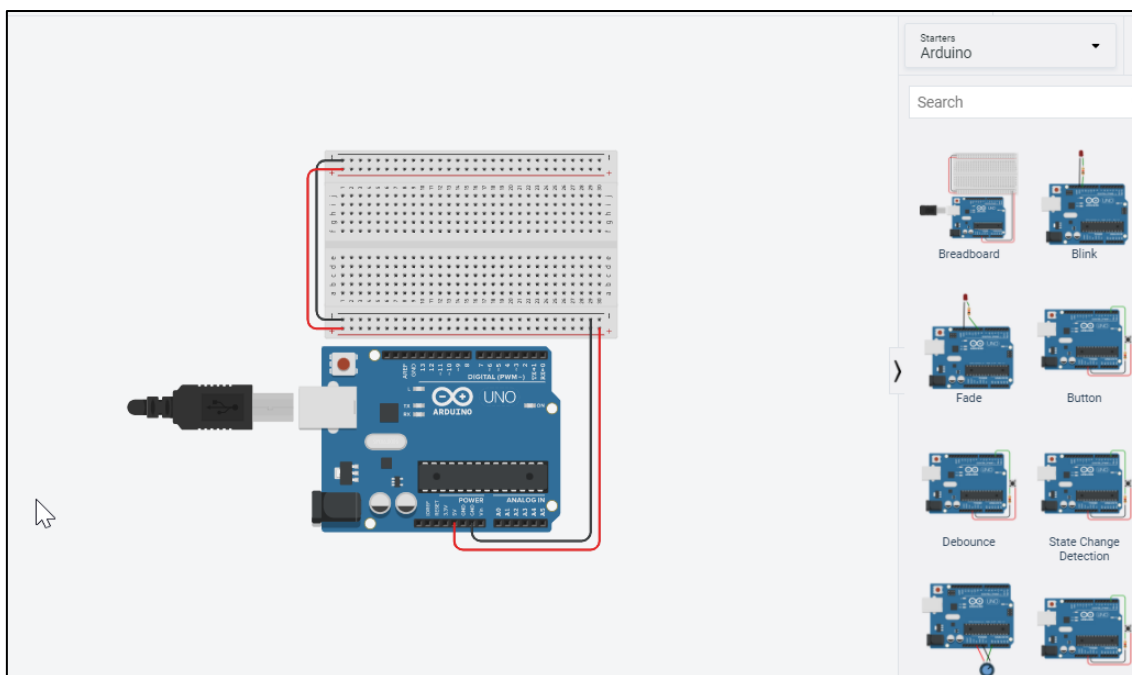


Figura 32 - Placa posicionada. Fonte: TINKERCAD.

No canto superior direito, clique em “Code”:

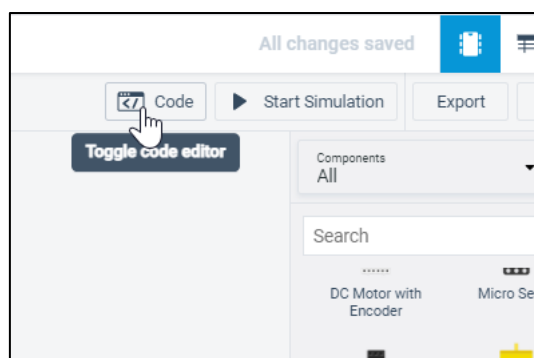


Figura 33 - Modificação da linguagem de programação. Fonte: TINKERCAD.

Clique em “Blocks” e mude para Text. Uma mensagem de advertência aparecerá na tela, clique em Continue:

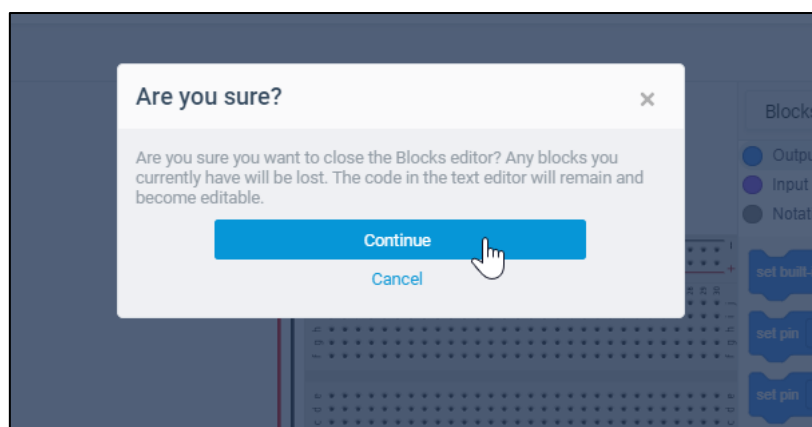


Figura 34 - Mensagem de Advertência sobre a escolha da linguagem de programação.
Fonte: TINKERCAD.

Sua tela deverá ficar assim:

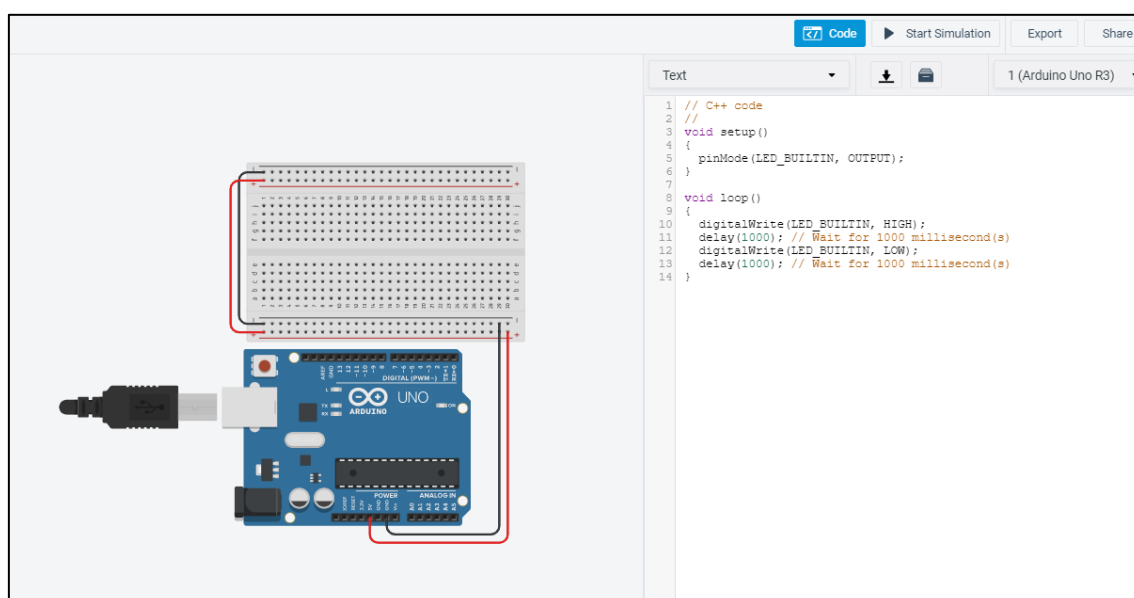


Figura 35 - Montagem padrão após a seleção da linguagem. Fonte: TINKERCAD.

Dica: para arrastar a montagem sem mover os componentes, clique na área cinza, fora da montagem e segure qualquer botão do mouse enquanto arrasta.

Objetivos de Aprendizagem

- Utilizar um simulador de Arduino;
- Escrever, carregar e executar programas básicos;
- Realizar montagens eletrônicas utilizando um simulador;
- Aprender conceitos básicos sobre o funcionamento de equipamentos de medição como multímetro e osciloscópio;
- Conhecer funções e estruturas de controle através de *software* sobre o microcontrolador e controlados.

Desafio 1: Escrita digital:

Neste exercício a proposta é criar um semáforo simples utilizando o processo de escrita digital.

Aprendizados esperados com este exercício:

- Conhecimento acerca do microcontrolador e *LED*
- Conceito de escrita digital
- Compreensão da função `delay()`

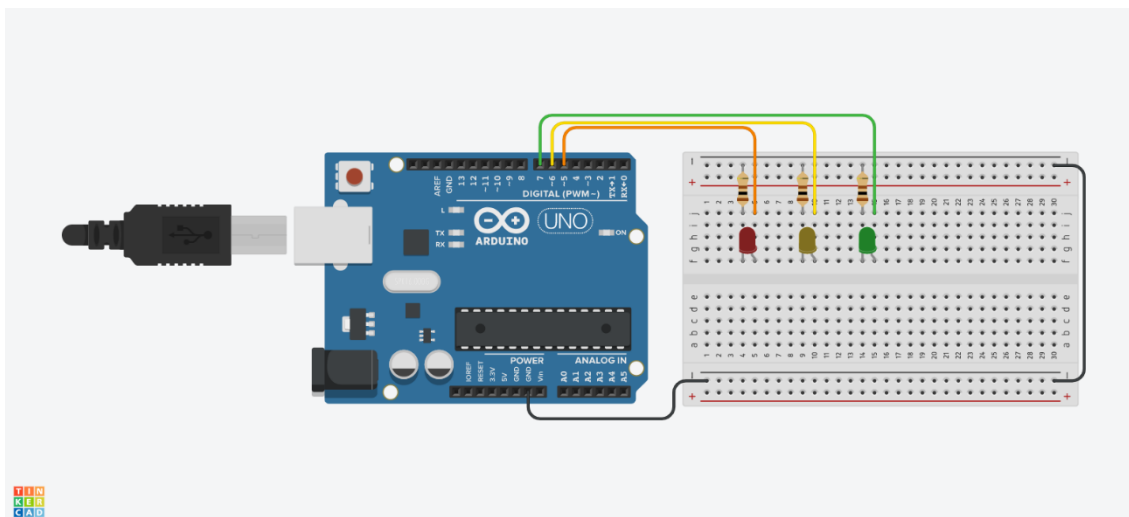


Figura 36 - Exemplo de montagem para o desafio 1. Fonte: Bruno Agrofoglio Ferreira através doAutodesk Tinkercad.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, INO ou TXT podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

Foto da Montagem (*print screen* ou exportação da montagem)

Código do microcontrolador Arduino (arquivo .ino)

Desafio 2: Leitura digital:

Neste exercício a proposta é adicionar um botão de solicitação de travessia ao semáforo utilizando o processo de leitura digital.

Aprendizados esperados com este exercício:

- Conhecimento acerca do *push-button*
- Conceito de leitura digital
- Realizar o *debounce* do botão

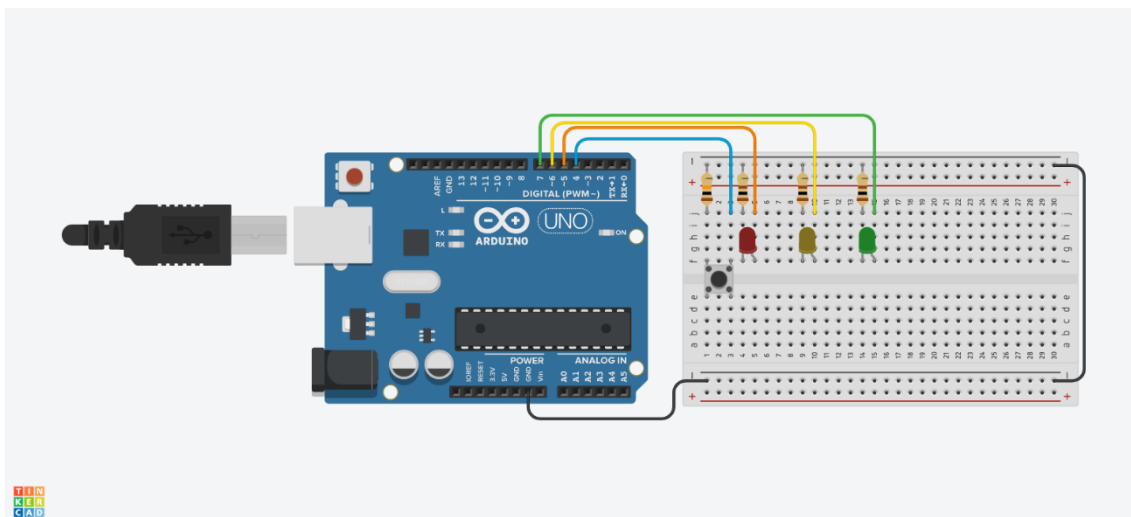


Figura 51 - Exemplo de montagem para o desafio 2. Fonte: Bruno Agrofoglio Ferreira através do Autodesk Tinkercad.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, INO ou TXT podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

- Foto da Montagem (*print screen* ou exportação da montagem)
- Código do microcontrolador Arduino (arquivo .ino)

Desafio 3: Leitura analógica:

Neste exercício a proposta é temporizar o semáforo através de um potenciômetro utilizando o processo de leitura analógica.

Aprendizados esperados com este exercício:

- Conhecimento acerca do potenciômetro
- Conceito de leitura analógica
- Compreensão da função `map()`

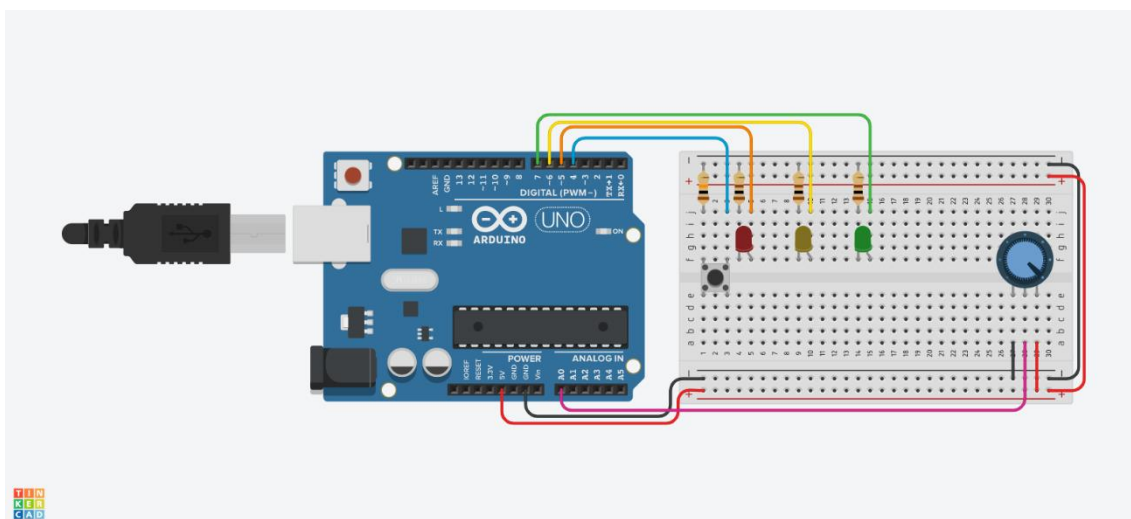


Figura 372 - Exemplo de montagem para o desafio 3. Fonte: Bruno Agrofoglio Ferreira através doAutodesk Tinkercad.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, INO ou TXT podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

Foto da Montagem (*print screen* ou exportação da montagem)

Código do microcontrolador Arduino (arquivo .ino)

Desafio 4: Escrita analógica:

Neste exercício a proposta é criar um semáforo simples utilizando o processo de escrita digital.

Aprendizados esperados com este exercício:

- Conhecimento acerca do *LDR*
- Conceito de escrita analógica
- Compreensão acerca de divisor de tensão

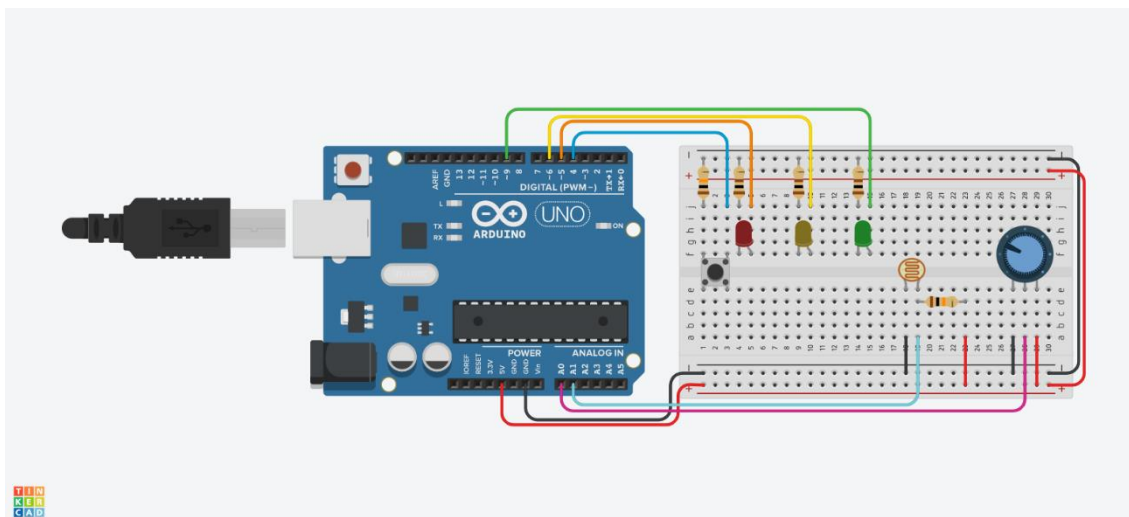


Figura 38 - Exemplo de montagem para o desafio 4. Fonte: Bruno Agrofoglio Ferreira através doAutodesk Tinkercad.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, INO ou TXT podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

Foto da Montagem (*print screen* ou exportação da montagem)

Código do microcontrolador Arduino (arquivo .ino)

Desafio 5: Semáforo inteligente:

Neste exercício a proposta é criar um semáforo simples utilizando o processo de escrita digital.

Aprendizados esperados com este exercício:

- Conhecimento acerca de projetos (*PoC* e *MVP*)
- Conceito de acessibilidade
- Compreensão da função `tone()`

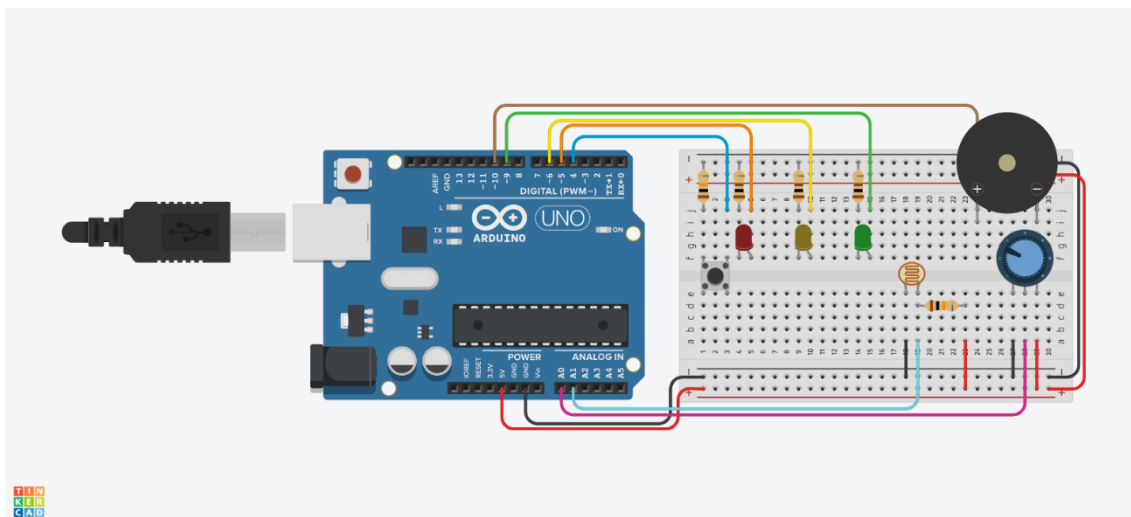


Figura 39 - Exemplo de montagem para o deasfio 5. Fonte: Bruno Agrofoglio Ferreira através doAutodesk Tinkercad.

Ao final da execução do exercício, insira evidências sobre a atividade em arquivos: PDF, JPG e/ou PNG, INO ou TXT podendo inserir na plataforma quantos arquivos quiser ou pode-se zipar os arquivos- ZIP e RAR, inserindo todos os arquivos de uma vez só. As evidências são:

Foto da Montagem (*print screen* ou exportação da montagem)

Código do microcontrolador Arduino (arquivo .ino)

Conclusão

A abordagem desta apostila permite ao leitor uma interação ativa com as tecnologias que são comuns a vários sistemas embarcados. Ensinar como um sistema embarcado funciona, suas entradas, saídas, interfaces de comunicação e como criar um pensamento computacional para resolver problemas, dá ao aluno autonomia, estímulo, senso crítico e contribui para uma aprendizagem mais efetiva do conteúdo apresentado.

Com esta apostila o aluno poderá aprender como criar um sistema embarcado mínimo, capaz de ler sensores, acionar atuadores e realizar comunicação sem fio, possibilitando o desenvolvimento de dispositivos IoT na sua forma integral.

Assim, o conteúdo do curso PROGRAMAÇÃO DE SOFTWARE EMBARCADO EM IOT familiariza o leitor com o mundo tecnológico, onde o desenvolvimento de novos produtos é constante e possibilita ao mesmo o entendimento de tecnologias de comunicação atuais que podem ser utilizadas para desenvolvimento de ideias e aplicações em diferentes áreas de conhecimento. Obrigado por fazer parte do curso e ter realizado a leitura desta apostila. Espero que o interesse pelo desenvolvimento de sistemas embarcados em IoT, apresentado neste curso esteja aguçado, para que você pratique e conheça ainda mais as maravilhas do mundo de desenvolvimento de produtos eletrônicos embarcados.

Aos alunos que acharam o conteúdo desta apostila interessante, sugiro que busquem mais informações sobre o assunto para aumentar o leque de possibilidades de aplicações. Abaixo deixo uma lista de livros, sites, podcasts, filmes e youtubers channels que abordam o conteúdo desta apostila de forma interessante e sobre óticas distintas.

Podcast:

FIT-F.24.1.01-04Rev. B



Teorema de Shannon

<https://www.deviante.com.br/podcasts/scicast-398/>

Linguagens de Programação

<https://www.deviante.com.br/podcasts/scicast-271/>

Youtube Channels

GreatScott:

<https://www.youtube.com/channel/UC6mlxFTvXkWQVEHPsEdflzQ>

Bem Heck Hacks

<https://www.youtube.com/c/BenHeckHacks/videos>

EEVBlog

<https://www.youtube.com/user/EEVblog>

Andreas Spiess

https://www.youtube.com/channel/UCu7_D0o48KbfhpEohoP7YSQ

Bem Eater

<https://www.youtube.com/channel/UCS0N5baNIQWJCUrhCEo8WIA>

Sites interessantes

Portal Embarcados

<https://www.embarcados.com.br/>

Hackaday

<https://hackaday.com/>

Arduino Project Hub

<https://create.arduino.cc/projecthub>

Filme:

O jogo da imitação

A revolução tecnológica do Japão o nascimento do transistor

Livros:

Nahin, Paul J. The Logician and the Engineer: How George Boole and Claude Shannon Created the Information Age. Princeton University Press, 2013.

Oliveira, André Schneider de, and Fernando Souza de Andrade. Sistemas embarcados: hardware e firmware na prática. Editora Érica Ltda, 2006.

Referências

BANZI, M. e SHILOH, M. (2015). Getting Started with Arduino. Third edition, MakerMedia.

"dispositivo", in Dicionário Priberam da Língua Portuguesa [em linha], (2008-2021). Disponível em: <<https://dicionario.priberam.org/dispositivo>> Acesso em: Agosto de 2021.

ETHNOLOQUE (2021). Ethnologue: Languages of the World. [online] Disponível em: <<http://www.ethnologue.com/>> Acesso em: Setembro 2021.

"ferramenta", in Dicionário Priberam da Língua Portuguesa [em linha], (2008-2021), Disponível em: <<https://dicionario.priberam.org/ferramenta>> Acesso em: Agosto de 2021.

FISL 16: Cafeteira com software livre tuíta quando seu café está pronto.TechTudo, [online] Disponível em: <http://www.techtudo.com.br/noticias/noticia/2015/07/fisl-16-cafeteira-com-software-livre-tuita-quando-seu-cafe-esta-pronto.html>>. Acesso em: Agosto de 2021.

GAITATZIS, T. (2017). Bluetooth Low Energy A Technical Primer Your Guide to the Magic Behind the Internet of Things. First edition, BackupBrain.

GTA. UFRJ. (2021). IEEE 802.15 Bluetooth - Camada Física e Camada de Acesso ao Meio. [online] Disponível em: <[https://www.gta.ufrj.br/grad/09_1/versao-final/bluetooth/Page706.htm#:~:text=A%20hist%C3%B3ria%20do%20Bluetooth%20come%C3%A7a,celulares%20e%20seus%20respectivos%20acess%C3%B3rios.&text=Por%20esse%20motivo%2C%20em%201998,SIG%20\(Special%20Interest%20Group\).](https://www.gta.ufrj.br/grad/09_1/versao-final/bluetooth/Page706.htm#:~:text=A%20hist%C3%B3ria%20do%20Bluetooth%20come%C3%A7a,celulares%20e%20seus%20respectivos%20acess%C3%B3rios.&text=Por%20esse%20motivo%2C%20em%201998,SIG%20(Special%20Interest%20Group).>)> Acesso em: Setembro 2021.

LIMA, T.,(2021). AGC - O primeiro grande Sistema Embarcado - Embarcados. [online] Embarcados - Sua fonte de informações sobre Sistemas Embarcados. Disponível em:<<https://www.embarcados.com.br/agc-primeiro-grande-sistema-embarcado/>> Acesso em: Setembro 2021.

"rádio", in Dicionário Priberam da Língua Portuguesa [em linha], (2008-2021). Disponível em:<<https://dicionario.priberam.org/r%C3%A1dio>> Acesso em: Agosto de 2021.

RICARTE, I. L. M. (2003). Programação de sistemas: uma introdução (2021). [online] Disponível em:

><https://www.dca.fee.unicamp.br/cursos/EA876/apostila/HTML/node37.html>
ml> Acesso em: Setembro 2021.

SCHILDT, H. e MAYER, R. C. (2008). C completo e total. Pearson Education do Brasil.

"sistema", in Dicionário Priberam da Língua Portuguesa [em linha], (2008-2021). Disponível em: <<https://dicionario.priberam.org/sistema>> Acesso em: Agosto de 2021.

TECHTUDO. (2021). O que é o código ASCII e para que serve? Descubra. [online] Disponível em: <https://www.techtudo.com.br/noticias/noticia/2015/02/o-que-e-o-codigo-ascii-e-para-que-serve-descubra.html#:~:text=O%20ASCII%20%C3%A9%20um%20c%C3%B3digo,comum%20entre%20todas%20as%20m%C3%A1quinas>. Acesso em: Setembro 2021.

UNIT-CONVERSION.INFO (2021). Convert text to binary - Converters. [online] Disponível em: <<http://www.unit-conversion.info/texttools/convert-text-to-binary/>> [Acesso em: Setembro 2021].

WIKIPEDIA. (2021). Linguagem de programação – Wikipédia, a enciclopédia livre. [online] Disponível em: <https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o> Acesso em: Setembro 2021.

WILCHER, D. (2014). Make: Basic Arduino Projects: 26 Experiments with Microcontrollers and Electronics. First edition, Maker Media.

Apêndice

NOTA: Abaixo seguem uma série de códigos que podem ser utilizados como sugestão da resolução dos exercícios propostos. Nenhum dos códigos elaborados possui estrutura complexa, pois serão utilizados para explicar os conceitos básicos para os alunos. Eventuais bugs devem ser corrigidos nas próximas revisões.

Desafios:

```
// Desafio 1
#define vermelho 5
#define amarelo 6
#define verde 7
byte status = 2;    //0 verde, 1 amarelo, 2 vermelho
int tempoVerde = 5000;
int tempoAmarelo = 1000;
int tempoVermelho = 3000;
void sinalAmarelo(){
    digitalWrite(vermelho, 0);    //apagou o led vermelho
    digitalWrite(amarelo, 1);    //acendendo o led amarelo
    digitalWrite(verde, LOW);    //apagou o led verde
    status=1;
}
void sinalVerde(){
    digitalWrite(vermelho, 0);    //apagou o led vermelho
    digitalWrite(amarelo, 0);    //apagou o led amarelo
    digitalWrite(verde, 1);    //acendendo o led verde
    status=0;
}
void sinalVermelho(){
    digitalWrite(vermelho, HIGH); //acendendo o led vermelho
    digitalWrite(amarelo, LOW);    //apagou o led amarelo
    digitalWrite(verde, LOW);    //apagou o led verde
    status=2;
}
void setup(){
    pinMode(vermelho,OUTPUT);
    pinMode(amarelo,OUTPUT);
```

```
pinMode(verde,OUTPUT);
Serial.begin(9600);
Serial.println("Inicializando o semaforo...");
sinalAmarelo();
}
void loop(){
  if(status==0){
    sinalAmarelo();
    Serial.println("Acionando o sinal amarelo");
    delay(tempoAmarelo);
  }
  if(status==1){
    sinalVermelho();
    Serial.println("Acionando o sinal vermelho");
    delay(tempoVermelho);
  }
  if(status==2){
    sinalVerde();
    Serial.println("Acionando o sinal verde");
    delay(tempoVerde);
  }
  Serial.println("rodando...");
}
```

//Desafio 2

#define botao 4

```
#define vermelho 5
#define amarelo 6
#define verde 7
byte status = 2;    //0 verde, 1 amarelo, 2 vermelho
int tempoVerde = 5000;
int tempoAmarelo = 1000;
int tempoVermelho = 3000;
unsigned long relógio = millis();
void sinalAmarelo(){
    digitalWrite(vermelho, 0);    //apagou o led vermelho
    digitalWrite(amarelo, 1);    //acendendo o led amarelo
    digitalWrite(verde, LOW);    //apagou o led verde
    status=1;
}
void sinalVerde(){
    digitalWrite(vermelho, 0);    //apagou o led vermelho
    digitalWrite(amarelo, 0);    //apagou o led amarelo
    digitalWrite(verde, 1);    //acendendo o led verde
    status=0;
}
void sinalVermelho(){
    digitalWrite(vermelho, HIGH); //acendendo o led vermelho
    digitalWrite(amarelo, LOW);    //apagou o led amarelo
    digitalWrite(verde, LOW);    //apagou o led verde
    status=2;
}
void setup(){
    pinMode(botao, INPUT_PULLUP);
    pinMode(vermelho, OUTPUT);
    pinMode(amarelo, OUTPUT);
    pinMode(verde, OUTPUT);
```

```

Serial.begin(9600);
Serial.println("Inicializando o semaforo...");
sinalAmarelo();
}
void loop(){
  if(status==0&&(millis()-relogio>=tempoVerde)){
    sinalAmarelo();
    Serial.println("Acionando o sinal amarelo");
    relogio = millis();
  }
  if(status==1&&(millis()-relogio>=tempoAmarelo)){
    sinalVermelho();
    Serial.println("Acionando o sinal vermelho");
    relogio = millis();
  }
  if(status==2&&(millis()-relogio>=tempoVermelho)){
    sinalVerde();
    Serial.println("Acionando o sinal verde");
    relogio = millis();
  }
  if(digitalRead(botao)==0&&status!=2){
    sinalAmarelo();
    Serial.println("Apertado o botao de travessia");
    relogio = millis();
  }
  Serial.println("rodando...");
}

```

//Desafio 3

#define potenciometro A0

#define botao 4

#define vermelho 5

```

#define amarelo 6
#define verde 7
byte status = 2;    //0 verde, 1 amarelo, 2 vermelho
int tempoVerde = 2000;
int tempoAmarelo = 1000;
int tempoVermelho;
unsigned long relógio = millis();
void sinalAmarelo(){
    digitalWrite(vermelho, 0);    //apagou o led vermelho
    digitalWrite(amarelo, 1);    //acendendo o led amarelo
    digitalWrite(verde, 0);    //apagou o led verde
    status=1;
}
void sinalVerde(){
    digitalWrite(vermelho, 0);    //apagou o led vermelho
    digitalWrite(amarelo, 0);    //apagou o led amarelo
    digitalWrite(verde, 1);    //acendendo o led verde
    status=0;
}
void sinalVermelho(){
    digitalWrite(vermelho, 1); //acendendo o led vermelho
    digitalWrite(amarelo, 0); //apagou o led amarelo
    digitalWrite(verde, 0);    //apagou o led verde
    status=2;
}
void setup(){
    pinMode(potenciometro,INPUT);
    pinMode(botao,INPUT_PULLUP);
    pinMode(vermelho,OUTPUT);
    pinMode(amarelo,OUTPUT);
    pinMode(verde,OUTPUT);

```

```
Serial.begin(9600);
Serial.println("Inicializando o semaforo...");
sinalAmarelo();
}
void loop(){
  int ajuste=analogRead(potenciometro);
  tempoVermelho=map(ajuste,0,1023,1000,4000);
  Serial.print("Potenciometro: ");
  Serial.println(ajuste);
  Serial.print("Tempo do sinal vermelho: ");
  Serial.println(tempoVermelho);
  if(status==0&&(millis()-relogio>=tempoVerde)){
    sinalAmarelo();
    Serial.println("Acionando o sinal amarelo");
    relogio = millis();
  }
  if(status==1&&(millis()-relogio>=tempoAmarelo)){
    sinalVermelho();
    Serial.println("Acionando o sinal vermelho");
    relogio = millis();
  }
  if(status==2&&(millis()-relogio>=tempoVermelho)){
    sinalVerde();
    Serial.println("Acionando o sinal verde");
    relogio = millis();
  }
  if(digitalRead(botao)==0&&status!=2){
    sinalAmarelo();
    Serial.println("Apertado o botao de travessia");
    relogio = millis();
  }
}
```

```
Serial.println("rodando...");  
}
```

```
//Desafio 4  
#define potenciometro A0  
#define ldr A1  
#define botao 4  
#define vermelho 5  
#define amarelo 6  
#define verde 9
```



```

byte status = 2;    //0 verde, 1 amarelo, 2 vermelho
int brilho=255;
int tempoVerde = 2000;
int tempoAmarelo = 1000;
int tempoVermelho = 3000;
unsigned long relógio = millis();
void sinalAmarelo(){
  analogWrite(vermelho, 0);    //apagou o led vermelho
  analogWrite(amarelo, brilho);    //acendendo o led amarelo
  analogWrite(verde, 0);    //apagou o led verde
  status=1;
}
void sinalVerde(){
  analogWrite(vermelho, 0);    //apagou o led vermelho
  analogWrite(amarelo, 0);    //apagou o led amarelo
  analogWrite(verde, brilho);    //acendendo o led verde
  status=0;
}
void sinalVermelho(){
  analogWrite(vermelho, brilho); //acendendo o led vermelho
  analogWrite(amarelo, 0); //apagou o led amarelo
  analogWrite(verde, 0);    //apagou o led verde
  status=2;
}
void setup(){
  pinMode(potenciometro,INPUT);
  pinMode(ldr,INPUT);
  pinMode(botao,INPUT_PULLUP);
  pinMode(vermelho,OUTPUT);
  pinMode(amarelo,OUTPUT);
  pinMode(verde,OUTPUT);

```

```
Serial.begin(9600);
Serial.println("Inicializando o semaforo...");
sinalAmarelo();
}

void loop(){
  int luminosidade=analogRead(ldr);
  brilho=map(luminosidade,0,1023,0,255);
  Serial.print("Luminosidade LDR: ");
  Serial.println(luminosidade);
  Serial.print("Intensidade do brilho no LED: ");
  Serial.println(brilho);
  int ajuste=analogRead(potenciometro);
  tempoVermelho=map(ajuste,0,1023,1000,4000);
  Serial.print("Potenciometro: ");
  Serial.println(ajuste);
  Serial.print("Tempo do sinal vermelho: ");
  Serial.println(tempoVermelho);
  if(status==0&&(millis()-relogio>=tempoVerde)){
    sinalAmarelo();
    Serial.println("Acionando o sinal amarelo");
    relogio = millis();
  }
  if(status==1&&(millis()-relogio>=tempoAmarelo)){
    sinalVermelho();
    Serial.println("Acionando o sinal vermelho");
    relogio = millis();
  }
  if(status==2&&(millis()-relogio>=tempoVermelho)){
    sinalVerde();
    Serial.println("Acionando o sinal verde");
    relogio = millis();
  }
}
```

```
}  
if(digitalRead(botao)==0&&status!=2){  
    sinalAmarelo();  
    Serial.println("Apertado o botao de travessia");  
    relógio = millis();  
}  
Serial.println("rodando...");  
}
```

```
//Desafio 5  
#define potenciometro A0  
#define ldr A1  
#define botao 4  
#define vermelho 5  
#define amarelo 6  
#define verde 9  
#define buzzer 10  
byte status = 2;    //0 verde, 1 amarelo, 2 vermelho  
int brilho=255;
```

```
int tempoVerde = 2000;
int tempoAmarelo = 1000;
int tempoVermelho = 3000;
unsigned long relógio = millis();
void sinalAmarelo(){
    analogWrite(vermelho, 0);      //apagou o led vermelho
    analogWrite(amarelo, brilho);   //acendendo o led amarelo
    analogWrite(verde, 0);          //apagou o led verde
    tone(buzzer,440,tempoAmarelo);
    status=1;
}
void sinalVerde(){
    analogWrite(vermelho, 0);      //apagou o led vermelho
    analogWrite(amarelo, 0);       //apagou o led amarelo
    analogWrite(verde, brilho);     //acendendo o led verde
    tone(buzzer,100,tempoVerde);
    status=0;
}
void sinalVermelho(){
    analogWrite(vermelho, brilho); //acendendo o led vermelho
    analogWrite(amarelo, 0);       //apagou o led amarelo
    analogWrite(verde, 0);         //apagou o led verde
    tone(buzzer,900,tempoVermelho);
    status=2;
}
void setup(){
    pinMode(potenciometro,INPUT);
    pinMode(ldr,INPUT);
    pinMode(botao,INPUT_PULLUP);
    pinMode(vermelho,OUTPUT);
    pinMode(amarelo,OUTPUT);
```

```
pinMode(verde,OUTPUT);
Serial.begin(9600);
Serial.println("Inicializando o semaforo...");
sinalAmarelo();
}
void loop(){
  int luminosidade=analogRead(ldr);
  brilho=map(luminosidade,0,1023,0,255);
  Serial.print("Luminosidade LDR: ");
  Serial.println(luminosidade);
  Serial.print("Intensidade do brilho no LED: ");
  Serial.println(brilho);
  int ajuste=analogRead(potenciometro);
  tempoVermelho=map(ajuste,0,1023,1000,4000);
  Serial.print("Potenciometro: ");
  Serial.println(ajuste);
  Serial.print("Tempo do sinal vermelho: ");
  Serial.println(tempoVermelho);
  if(status==0&&(millis()-relogio>=tempoVerde)){
    sinalAmarelo();
    Serial.println("Acionando o sinal amarelo");
    relogio = millis();
  }
  if(status==1&&(millis()-relogio>=tempoAmarelo)){
    sinalVermelho();
    Serial.println("Acionando o sinal vermelho");
    relogio = millis();
  }
  if(status==2&&(millis()-relogio>=tempoVermelho)){
    sinalVerde();
    Serial.println("Acionando o sinal verde");
  }
}
```

```
    relógio = millis();  
  }  
  if(digitalRead(botao)==0&&status!=2){  
    sinalAmarelo();  
    Serial.println("Apertado o botão de travessia");  
    relógio = millis();  
  }  
  Serial.println("rodando...");  
}
```

CONTROLE DE REVISÃO DO DOCUMENTO / *DOCUMENT REVISION*

CONTROL

| Revisão <i>Review</i> | Descrição <i>Description</i> | Razão <i>Reason</i> | Autor <i>Author</i> | Data <i>Date</i> |
|--------------------------|---|--|---------------------------|---------------------|
| A | - | Revisão inicial | Larissa Alves | 21/09/21 |
| B | Inclusão do código do setor de qualidade do FIT | Conformidade com o setor de qualidade | Larissa Alves | 23/09/21 |
| C | Atualização do conteúdo | Processo de melhorias | Larissa Alves | 22/11/21 |
| D | Adequações de exercícios | Após os feedbacks dos alunos houve a necessidade de adequações dos exercícios. | Larissa Alves | 25/05/22 |
| E | Alteração de logotipia do rodapé do documento | Revisão do modelo do formulário FIT-F.24.1.01-04Rev. B | Bruno Agrofoglio Ferreira | 01/07/22 |



FUTURO DO TRABALHO, TRABALHO DO FUTURO

Bom curso