* Fala galera, espero que vocês estejam bem.
* Antes de começar vamos relembrar um pouco do que vimos semana passada.

Na terça passada vimos sobre as protoboards e como elas facilitam a montagem dos nossos circuitos e protótipos em eletrônica.

Detalhei pra vcs o seu funcionamento.

Falamos também sobre alguns conceitos fundamentais para a montagem de circuitos.

Que foram os conceitos de corrente, tensão e resistência elétricas.

Para podermos dimensionar e manipular essas grandezas nos nossos circuitos a partir da primeira lei de Ohm e dei um exemplo de como dimensionar os resistores em circuitos que possuem leds.

Demos um intervalo para que vocês pudessem começar a exercitar esses conceitos na plataforma tinkercad

Com um exercício que tinha a proposta de montar um pequeno circuito com leds e resistores e dimensionar esses resistores.

Depois desenvolvemos um pouco mais outro pilar do nosso curso, além da prototipagem

Na lógica de programação vimos os conceitos de constantes e variáveis

Como usá-los nos nossos circuitos e como eles facilitam a legibilidade e organização dos nossos códigos.

Além disso, apresentei pra vocês o conceito de laço condicional, pelas estruturas de seleção if e else.

Com elas podemos executar ou não, partes do nosso código de acordo com uma condição específica a ser validada.

Li junto com vocês duas versões de uma mesma implementação em código. Onde um motor era desligado caso superaquecesse. Para demonstrar os usos das variáveis e constantes, além da própria estrutura condicional if.

E finalizamos com o exercício proposto que simulava uma sirene policial. Lembrando que esse exercício ficará aberto por tempo indeterminado e não contará como requisito de avaliação. Isso ficará a cargo dos quizzes.

Vou passar pra vocês pelo chat o link com os materiais das aulas anteriores e dessa aula.

Lá vocês vão encontrar todo o material das aulas com os slides, a solução em código dos exercícios e os materiais complementares

[Aulas público](https://drive.google.com/drive/folders/1mS8Tcu3JZwmZJJCS-X_y4vMsAanW9i-n?usp=sharing)

Também vou propor pra vcs a implementação de 2 exercícios, um com botões e outro pra treinar o uso das funções implementando um semáforo

Bom... primeiramente iremos apresentar um componente fundamental para projetos interativos: o botão pushbutton. Vamos entender o seu funcionamento, como conectá-lo ao Arduino e como programar para detectar eventos de pressionar e soltar.

O botão pushbutton é um componente eletrônico simples que permite controlar o fluxo da corrente elétrica em um circuito. Ele funciona como uma chave que, quando pressionada, estabelece uma conexão elétrica e, quando solta, interrompe essa conexão.

Existem dois tipos principais de botões pushbutton:

O botão normalmente aberto (NA), ele vai conduzir corrente no circuito só quando for pressionado e deixará de conduzir quando solto

E o botão normalmente fechado que conduz quando solto e interrompe quando pressionado

Esse é um exemplo do diagrama de um botão normalmente aberto, nele temos quatro ligações. Um que vai pro arduino, o output. Um que vai pro gnd e outro que vai pra fonte de tensão o vcc. Quando o botão não está pressionado o sinal vai do arduino em output direto para o gnd.

Já quando pressionamos o botão o circuito fecha, e o arduino recebe sinal do vcc

Bem… em relação a conexão do arduino e do botão

Quando conectamos um botão no Arduino, precisamos utilizar um resistor pull-up ou pull-down para evitar leituras instáveis.

Vamos entender um pouco como eles são conectados

No resistor pull-up, conectamos o botão entre o pino digital do Arduino e o GND e nosso resistor, que geralmente é de 4 a 10 k ohms. é conectado entre o pino digital e o 5V. Quando o botão não está pressionado, o pino lê 5volts.

Já quando pressionamos, o pino lê 0 volts.

Já no resistor pull-down, conectamos o botão entre o pino digital e 5V. O nosso resistor, é conectado entre o pino digital e o gnd. Quando o botão é pressionado é que o pino lê 5volts

Certo

Para ler o estado de um botão, utilizamos a função digitalRead(). ele basicamente vai lê o sinal de um pino de entrada.

Utilizamos o pinmode para configurar nosso pino como entrada. Nosso input

E o digitalread para ler o sinal do botão

À direita temos um exemplo de aplicação

Primeiramente definimos nosso botão para o pino 2 e nosso led para o 13

Configuramos pelo pinmode nosso botão como entrada e nosso led como saída

E na nossa função principal inicializamos uma variável chamada estadobotão que vai receber o valor do botão pela função digitalread

se o botão tiver em high, estando pressionado, ligamos o led

ja se estiver em repouso desligamos

Os botões são bastante utilizados no controle de interface de usuário, navegação em menus tambem, muito usado no acionamento de dispositivos, como nesse exemplo em que acionamos um led

Muito usado em Jogos, sistemas de segurança e alarmes

Agora vocês vão realizar um exercício na plataforma Tinkercad, onde irão montar um circuito com

Um botão pushbutton

Dois LEDs

Um Arduino

E o objetivo de vcs será implementar um botão que

quando for pressionado e mantido, o primeiro LED deve acender

E quando o botão for solto, o primeiro LED deve apagar

A cada vez que o botão for pressionado, o segundo LED deve alternar seu estado entre ligar e desligar

A tarefa desse exercício é a tarefa de número 4 e vou passar o link pra vcs

<https://www.tinkercad.com/joinclass/8EREQRMRQ>

Se vocês tiverem alguma dúvida sobre os conceitos ou sobre o funcionamento dos botões pushbutton, eu posso responder nesse intervalo e tambem vou acessar a atividade para mostrar o funcionamento do circuito

Lembrando que esse exercício não será validado como avaliação, então é opcional. É obrigatório somente responder aos quizzes

vou aproveitar pra passar o link da presença também

Lista de presença :

<https://docs.google.com/forms/d/e/1FAIpQLSeg8LHs1IA6gS6YYO5kqo5COdpi0kmEPjFURBYlEeZ4IlgORw/viewform?usp=header>

Quiz aula03 :

<https://docs.google.com/forms/d/e/1FAIpQLSd8eabbMuo8ZPCjnevJpWDIYs7wdaW4pMiTXLZtuVCJCLdYPA/viewform?usp=header>

vou colocar certinho no chat

Continuando nosso treinamento

Vamos ver um conceito fundamental da programação que vai nos ajudar a organizar melhor nosso código. as chamadas funções. Vamos entender o que são funções, como criá-las e como utilizá-las em nossos projetos.

Começando pelo conceito de funções, elas são blocos de código organizados que executam uma tarefa específica.

Elas funcionam como uma receita de bolo dentro do programa principal.

Elas são utilizadas quando precisamos realizar uma tarefa repetitiva ou organizar melhor nosso código

As funções oferecem algumas vantagens importantes

Como a Reutilização de código

Escrevemos o código uma única vez e podemos fazer uso dele várias vezes

A Organização

Dividimos problemas complexos em partes menores e mais gerenciáveis

A Manutenção . Pois facilita a correção de erros, só precisamos modificar o código em um lugar quando é necessário alteração

E a Legibilidade: torna o código mais fácil de entender

Certo, mas como criar funções

A estrutura de uma função em Arduino segue o seguinte formato

O tipo retorno.

Que é o tipo de dado que a função vai devolver. Podendo ser int, float, void, etc.)

O nome\_da\_funcao: é o nome que você escolhe para sua função

Os parametros. Que são valores que a função pode receber para trabalhar

E o return. usado para devolver um valor, quando necessário pra função que chamou ela

Quando uma função não retorna nenhum valor, usamos a palavra "void" como tipo de retorno.

Vejamos alguns exemplos práticos

Na primeira função sem parâmetros e sem retorno

Ela apenas pisca o LED integrado do Arduino, ligando-o por um segundo e desligando-o por mais um segundo.

Na segunda função. Ela possui parâmetros mas não tem retorno

Neste caso, a função recebe três parâmetros

O pino do LED e os tempos de ligado e desligado.

Já na terceira função, temos uma função com retorno, onde ela recebe dois números inteiros, soma eles e retorna o resultado.

Mas onde declarar as funções no código

Bem…

No Arduino, geralmente declaramos as funções após a função loop().

No entanto, precisamos criar uma declaração prévia chamada protótipo antes da função setup() para informar ao compilador que a função existe. Caso contrário ele não compila.

Outra opção é simplesmente definir a função completa antes das funções setup() e loop(). como no exemplo

Mas bem pessoal, existem algumas funções que são nativas do arduino e que utilizamos frequentemente nos nossos códigos

Como a função pinmode, que já usamos diversas vezes

Assim como a digitalWrite

Mas também tem a digitalRead( ). Que lê o estado do pino

A analogRead( ). Que lê o valor de um pino analógico

A analogWrite( ). Que envia um valor PWM para um pino

(PWM é uma ferramenta que modula a largura do pulso, para que possamos manipular os valores de tensão)

O delay(). Que pausa o programa por um tempo específico em milissegundos

E o millis(). Que retorna o tempo em milissegundos desde que o Arduino foi ligado

Bom..

Agora vamos falar um pouco sobre o escopo de variáveis em funções

É importante entender que as variáveis declaradas dentro de uma função têm escopo local, ou seja, elas só existem dentro da função. Como no exemplo do slide. Na função loop temos a variável valorLocal. Já na função mostraValor também temos uma variável valorLocal. Apesar das duas possuirem o mesmo nome, pelo fato delas terem sido criadas dentro de cada função, sendo elas variáveis locais, elas só vão funcionar dentro dos seus respectivos blocos de código. Se eu tentar mudar o valor delas. Essa mudança não vai se refletir na outra função

Se quisermos que uma variável seja acessível em qualquer parte do código, precisamos declarár ela como global, fora de qualquer função

No exemplo do slide. Declaramos a variável valor global fora das funções com o valor 10

E na primeira vez que ela é impressa na função loop. Temos o valor 10.

Depois chamamos a função mostraValor. Acrescentando 5 ao valor 10 e imprimindo o valor 15 da mesma variável.

Depois voltando em loop, acrescentamos mais 5 à variável. Tendo então o valor 20.

Mas pra exemplificar as principais vantagens de usar funções seja em relação a reutilização de código, organização, manutenção e legibilidade

Vamos comparar dois programas que basicamente fazem a mesma coisa. Que é piscar três leds de maneira alternada.

No da esquerda, onde não é usado funções. Notem que o bloco de código pra ligar e desligar os leds é feito três vezes. Um para cada cor. Se tivéssemos 100 cores. repetiríamos 100 vezes.

Já no segundo código esse bloco é feito uma única vez pela função acenderLed. Sendo a função loop responsável somente por informar qual led deve ser ligado.

Isso deixa o código muito mais organizado e mais fácil de entender. Além de escrevermos menos digamos assim

Agora vou propor para vocês um exercício que simula um semáforo. O exercício será feito na plataforma Tinkercad e vou enviar o link com a tarefa e a imagem do circuito para vocês.

O exercício vai ficar aberto por tempo indeterminado

Caso vocês tenham alguma dúvida sobre o conteúdo passado até hj podem perguntar pelo chat. Caso contrário acho que podemos finalizar por hj

Vou passar novamente o link dos quizzes pra vcs

Lista de presença :

<https://docs.google.com/forms/d/e/1FAIpQLSeg8LHs1IA6gS6YYO5kqo5COdpi0kmEPjFURBYlEeZ4IlgORw/viewform?usp=header>

Quiz aula03 :

[https://docs.google.com/forms/d/e/1FAIpQLSd8eabbMuo8ZPCjnevJpWDIYs7wdaW4pMiTXLZtuVCJCLdYPA/vieswform?usp=header](https://docs.google.com/forms/d/e/1FAIpQLSd8eabbMuo8ZPCjnevJpWDIYs7wdaW4pMiTXLZtuVCJCLdYPA/viewform?usp=header)