# FACULDADE DE COMPUTAÇÃO E INFORMÁTICA

# BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

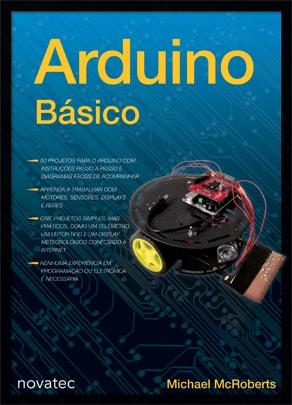
**FUNDAMENTOS DE COMPUTAÇÃO E SISTEMAS**

**Atividade 02 – 1º SEMESTRE/2018**

**PRÁTICA: VOID LOOP (CICLO DE INSTRUÇÃO)**

 Nossos **objetivos** nesta aula são:

* Conhecer a UCP, o Ciclo de Instrução e o Clock.
* Conhecer o ciclo de instrução Void Loop ().
* Praticar o conceito de busca e execução utilizando LEDs.

Para esta aula, usamos como referência a **Seção 1 (IDE do Arduino)** do **Capítulo 1 (Introdução)** do nosso livro-texto:

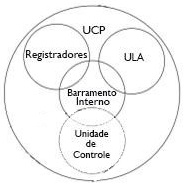
McRoberts, Michel. **Arduino básico, tradução Rafael Zanolli.**1.ed. São Paulo: Novatec, 2011. Disponível em formato eletrônico.

E também, a Seção 5 **(Velocidade do clock e instruções por segundo)** do Capítulo 2 **(Evolução e desempenho do computador)** e a Seção 2 **(Busca e execução de instruções)** do Capítulo 3 **(Visão de alto nível da função e interconexão do computador)**.

STALLINGS, William; VIEIRA, Daniel; BOSNIC, Ivan (Trad.). Arquitetura e organização de computadores. 8. ed. São Paulo: Pearson Education do Brasil, 2011. Disponível em formato eletrônico.

*Não deixem de ler estas seções depois desta aula!*

**Unidade central de processamento (UCP)**

 Figura 1 –Visão geral da UCP

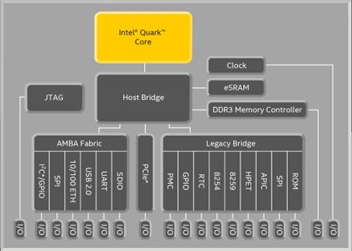
Seus principais componentes estruturais são:

* **Registradores:** oferece armazenamento interno à UCP.
* **Unidade de Controle:** controla a operação da UCP e também encaminha sinais de controle para fora da UCP, ou seja, controla a operação do computador.
* **Unidade Lógica e Aritmética (ULA):** realiza operações lógicas e aritméticas com binários, ou seja, executa as funções de processamento de dados do computador.
* **Interconexão da UCP:** é um caminho de dados para a comunicação entre os dispositivos internos à UCP.

# Fonte: STALLINGS, 2011, p. 10. (adaptado)

**Processador Intel Quark SoC X1000**

Figura 2 – Processador Intel

* **Processador Intel Quark SoC X1000:** o processador Quark SoC X1000 possui um núcleo e 1 encadeamento de execução. Ele trabalha com uma frequência de 400MHz e tem um cache de 16KB.
* **Núcleo:** os núcleos funcionam como um processador independente, são neles que as tarefas são executadas, por exemplo se você tem dois programas em atividade e seu processador é um dual-core (dois núcleos) cada programa pode ser rodado em um núcleo.

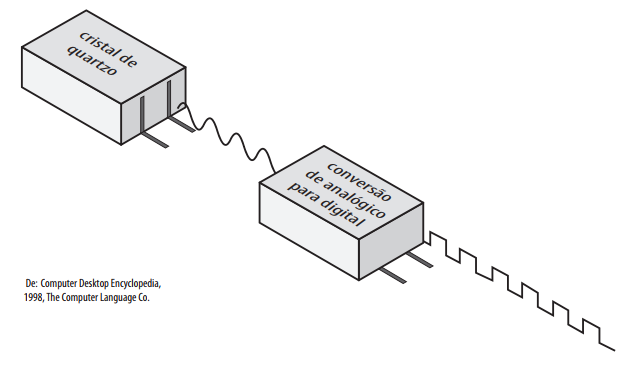
Fonte: Página da internet¹ disponível em: <https://www.coreboot.org/git-docs/Intel/SoC/quark.html> acesso em fev. 2018.

* **Encadeamento de Execução:** conhecido como thread, é um pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se autodividir em duas ou mais tarefas.
* **Frequência:** afrequência é uma grandeza que indica quantas vezes um determinado evento ocorre em um intervalo de tempo. O clock gera um sinal usado para determinar a frequência.
* **Cache:** é uma memória de alto desempenho, localizada na UCP. Esta memória agiliza o acesso às instruções e dados requeridos pela UCP e que estão contidos na memória RAM.

# Clock

Clock é um sinal que é usado pela unidade de controle para controlar as operações em um computador, tais como busca de instruções, decodificação ou até mesmo uma operação aritmética. Os sinais de clock são gerados por um cristal de quartzo, que gera uma onda de sinal constante enquanto a alimentação é aplicada. (Figura 3).

Figura 3 –Conversão analógica/digital.

****

Fonte: STALLINGS, 2011, p. 39.

* **Taxa De Execução De Instrução:** embora o processador seja controlado por uma unidade de controle, é o clock com uma frequência constante **f**, que determina a taxa de execução por instrução, ou seja, é a média de ciclos que a instrução precisa para ser executada. Sendo **f** uma medida do número total de vibrações, ou oscilações feitas dentro de um período determinado de tempo, ou, de modo equivalente, um tempo de ciclo constante **τ**, onde **τ = 1/f.** Por exemplo podemos comparar a frequência com o comando **delay** (veremos posteriormente nessa aula), em que, quanto maior o seu valor menor será a taxa de execução (o piscar de um LED).

# Void Loop () – (Ciclo de Instrução)

# Essa função é executada ciclicamente pelo processador Intel Quark SoC X1000 no Galileo. A função é executada em uma estrutura de repetição, sendo interrompida apenas por uma chamada de sub-rotina ou interrupção (estes conceitos serão abordados no curso posteriormente). Essa função recebe o nome de ciclo de instrução. Segundo STALLINGS “*Um ciclo de instrução consiste em uma busca de instrução, seguida por zero ou mais buscas de operandos, seguidas por zero ou mais armazenamentos de operandos, seguidos por uma verificação de interrupção (se as interrupções estiverem habilitadas*”. De forma geral essas microoperações (operações menores dentro do ciclo de instrução) podem ser definidas como:

# Cálculo de endereço de instrução: (IAC, do inglês Instruction Address Calculation): determina o endereço da próxima instrução a ser executada. Normalmente, isso envolve acrescentar um número fixo ao endereço da instrução anterior.

# Busca da instrução: (IF, do inglês Instruction Fetch): lê a instrução do seu local da memória para o processador.

* **Decodificação da operação da instrução:** (IOD, do inglês **I**nstruction **O**peration **D**ecoding): analisa a instrução para determinar o tipo de operação a ser realizada e o operando ou operandos a serem utilizados.
* **Cálculo do endereço do operando:** (OAC, do inglês **O**peration **A**ddress **C**alculation): se a operação envolve referência a um operando na memória ou disponível via E/S, então determina o endereço do operando.
* **Busca do operando:** (OF, do inglês **O**peration **F**etch): busca o operando da memória ou o lê da E/S.
* **Operação dos dados:** (DO, do inglês **D**ata **O**peration): realiza a operação indicada na instrução.
* **Armazenamento do operando:** (OS, do inglês **O**perand **S**tore): escreve o resultado na memória ou envia para a E/S.

# EXERCÍCIO TUTORIADO - PISCA LED

Para o primeiro projeto você conectará um LED a pinos digitais fazendo-o piscar a cada segundo em um loop (repetição), tendo como objetivo ver na prática como funciona o ciclo de instrução.

# *Componentes necessários:*

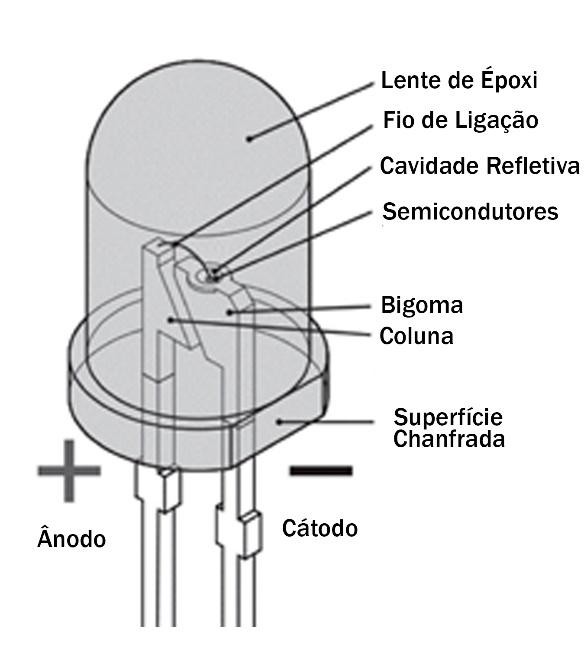
Figura 4 – Exemplo de protoboard

# Protoboard:

* Também conhecida como matriz de contato ou placa de ensaio, ela é uma placa com furos, utilizada para fazer projetos provisórios. Sua vantagem é que não necessita de soldagem para montagem de circuitos eletrônicos.
* Esta imagem de uma protoboard serve para indicar o sentido da corrente

Fonte: McRoberts, 2011, p. 46

Figura 5 – LED



**LED:**

* LED (**L**ight **E**mitting **D**iode em português **D**iodo **E**missor de **L**uz). São diodos que quando percorridos por uma corrente elétrica são capazes de emitir luz. Um diodo é um dispositivo que permite o fluxo de corrente em apenas uma direção, em outras palavras, se você conectar acidentalmente o LED em terminais errados em um circuito, ele evita a danificação dos componentes.

Fonte: McRoberts, 2011, p. 50

Figura 6 –resistor

# Resistor:

* É um dispositivo projetado para provocar resistência a uma corrente elétrica, transformando esta em energia térmica, limitando assim a corrente do circuito.
* Como o resistor provoca resistência na corrente elétrica, se por exemplo, se almeja diminuir a luminosidade de um LED, uma de suas opções é aumentar o valor da resistência.

Fonte: McRoberts, 2011, p. 49

Figura 7 – Fio jumper

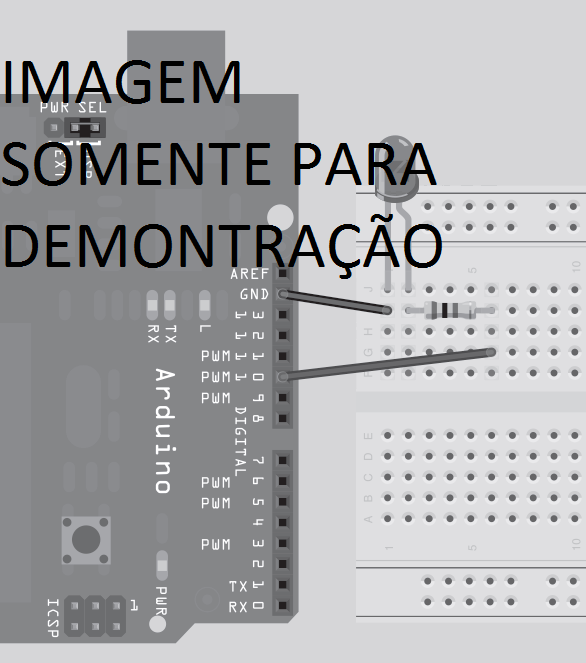
# C:\Users\Grég\Downloads\Jumper (1).jpg

# Fios Jumper:

* É uma peça plástica que contém um pequeno filamento de metal responsável pela condutividade, usado para conectar circuitos elétricos.

Fonte: Página da internet² disponível em: http://www.dreaminc.com.br/sala\_de\_aula/fios-e-cabos/ acesso em fev. 2018.

*Conectando os componentes:*



**1º Passo:** certifique-se de que o Galileo esteja desligado (para não correr o risco de uma peça encostar em outra que não devia e causar um curto).

**2º Passo:** conecte os componentes como mostra a imagem ao lado.

**Obs.** Certifique-se de que seu LED esteja conectado corretamente, com o terminal mais longo (positivo ou ânodo) conectado ao pino digital 10, que é a saída de energia. Já o mais curto (negativo cátodo) deve ir no terra (GND).

**3º Passo:** abra seu IDE (**I**ntegrated **D**evelopment **E**nvironment, ou Ambiente de Desenvolvimento Integrado) do Arduino.

**4º Passo:** digite o código como aprendido na aula passada: (o código que será analisado posteriormente)

**// Projeto 1 - LED pisca led**

**int ledPin = 10; // define variável com valor inteiro**

**void setup() {**

**// função de configuração do microcontrolador**

**pinMode(ledPin, OUTPUT);**

**// define porta de E/S digital como saída**

**}**

**void loop() {**

**digitalWrite(ledPin, ,**

**GH);// ativa o E/S digital com nível lógico alto**

**delay(1000);**

**digitalWrite(ledPin, LOW); // ativa o E/S digital com nível lógico baixo**

**delay(1000);**

**}**

**5º Passo:** pressione o botão *Verify/Compile* no topo do IDE para certificar-se de que não há erros em seu código. Se não houver erros, clique no botão Upload para fazer o upload do código ao seu Arduino. Caso tudo esteja correto, agora você deverá ver o LED vermelho, na Protoboard, acendendo e apagando em intervalos de um segundo.

# *Análise do Código:*

A primeira linha do código do projeto é:

**// Projeto 1 –Pisca LED**

Trata-se apenas de um comentário em seu código. Já a linha seguinte no programa é:

**int ledPin = 10;**

Isso é que chamamos de variável. Uma variável é um local em que podemos armazenar dados. Nesse caso, você está definindo uma variável de tipo int. O valor 10 define qual pino no Galileo você utilizará para conectar o LED.

Em seguida, vemos a função setup ():

**void setup() {**

**pinMode(ledPin, OUTPUT);**

**}**

Nela que você emitirá instruções gerais para preparar o programa antes que o loop principal seja executado, como a definição dos modos dos pinos, das taxas de transmissão serial etc.

Agora é execução do loop principal do programa:

**void loop() {**

**digitalWrite(ledPin, HIGH);**

**delay(1000);**

**digitalWrite(ledPin, LOW);**

**delay(1000);**

**}**

A função loop () é a função principal do programa e executa continuamente enquanto o Galileo estiver ligado. Todas as declarações dentro da função loop () (dentro das chaves) são executadas uma de cada vez, passo a passo, até que se alcance o fim da função; nesse ponto, o loop reinicia desde o princípio e assim infinitamente, ou até que o Galileo seja desligado ou o botão Reset pressionado.

A primeira instrução dentro do laço é:

**digitalWrite(ledPin, HIGH);**

Ela escreve um valor HIGH (ligado) ou LOW (desligado) para o pino dentro da instrução (nesse caso ledPin, que é o pino digital 10). Quando define como LOW, o pino se torna 0 volts, ou terra. Essa instrução, portanto, envia 5 V para o pino 10 e acende o LED. Depois dela, temos:

**delay(1000);**

O que essa instrução faz? Qual a frequência que o LED pisca?

**digitalWrite(ledPin, LOW);**

Que desliga o pino 10, logo apaga o LED. Então, há outra instrução de espera e por fim a função termina.

# 

# EXERCÍCIO COM DISCUSSÃO EM DUPLAS

Faça um programa para o LED piscar em uma frequência de 2 Hz.

# EXERCÍCIOS EXTRA-CLASSE

1. Ler a **Seção 1 (IDE do Arduino)** do **Capítulo 1 (Introdução)** do nosso livro-texto Arduino básico. **Seção 5 (Velocidade do clock e instruções por segundo)** do **Capítulo 2 (Evolução e desempenho do computador)**.  **Seção 2 (Busca e execução de instruções)** do **Capítulo 3 (Visão de alto nível da função e interconexão do computador),** do livro-texto Arquitetura e Organização de Computadores.
2. **(PARA ENTREGA).** Crie um sinalizador de Código Morse. Ele deverá fazer com que o LED sinalize as letras S.O.S, sinal de socorro internacional em Código Morse. Lembre-se: “O código Morse é um tipo de codificação de caracteres que transmite letras e números utilizando padrões de ligado e desligado. ” **Obs. A estrutura física pode ser feita no site *https://www.tinkercad.com/circuits.***
3. **(PARA ENTREGA).** Quando duas se encontram existe a necessidade de controlar o fluxo de veículos para evitar acidentes, pensando nisso faça uma simulação de um semáforo de trânsito, usando três LEDs (verde, amarelo e vermelho). Ele irá do verde ao vermelho, passando pelo amarelo, usando o sistema brasileiro de 3 estados.