PROFESSOR DANILO *ROBÓTICA – 9 ANO – 17/02/2022*

PRIMEIROS CONTATOS COM O ARDUINO
AULA 02

Esta aula será bem prática. Você receberá um kit com diversos componentes, mas na aula de hoje usaremos apenas o Arduino e um cabo USB.

Figura 1: Cabo USB padrão A/B

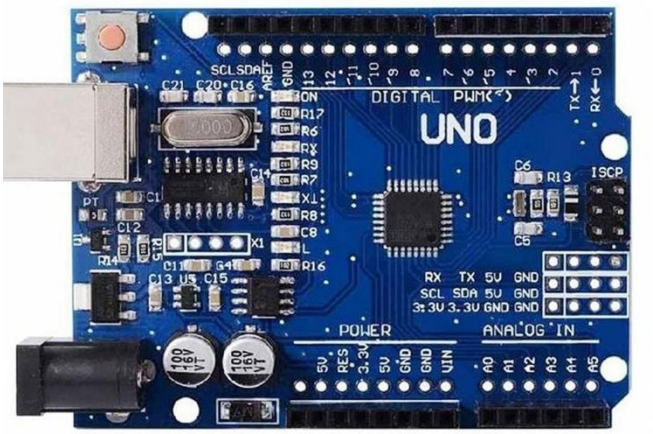

Figura 2: Arduino UNO R3 SMD. Note que o microcontrolador Atmega328 é soldado com um tipo de solda chamada SMD

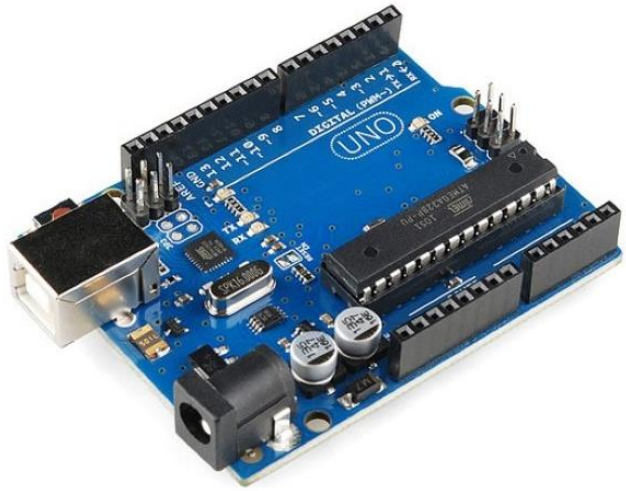

Figura 3: Arduino UNO R3. Note que o microcontrolador Atmega328 é encaixado em um soquete, o que permite que ele seja removido e encaixado em um circuito distinto, como numa impressora 3D ou num sistema que controla os eletrônicos de uma casa

O computador disponível para o seu grupo tem instalado um programa também chamado de Arduino. Para acessá-lo, você pode pressionar a tecla do Windows e depois começar a digitar a palavra Arduino. Quando o aplicativo aparecer, pressione enter.

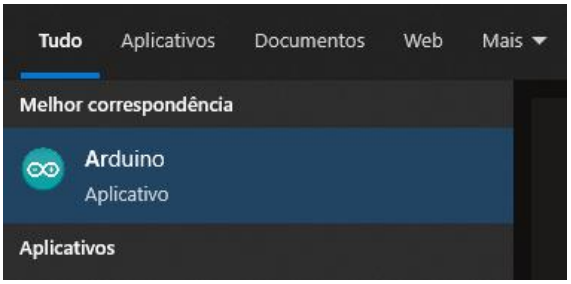

Figura 4: ao digitar “Arduino” após pressionar a tecla de Windows, você deverá acessar o programa mostrado na figura acima

O programa abaixo deverá abrir.

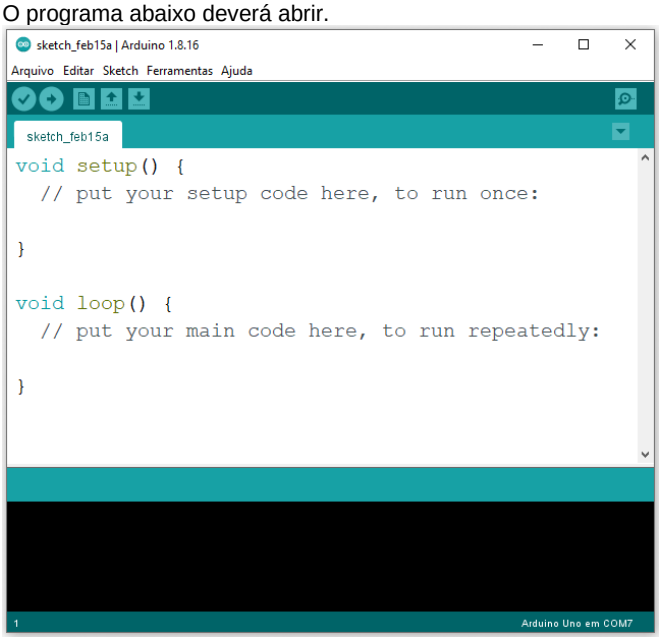

Figura 5: Programa Arduino

Como primeira atividade, você terá que inserir o cabo USB no computador e no Arduino, selecionar a porta correta na qual o Arduino está conectado.

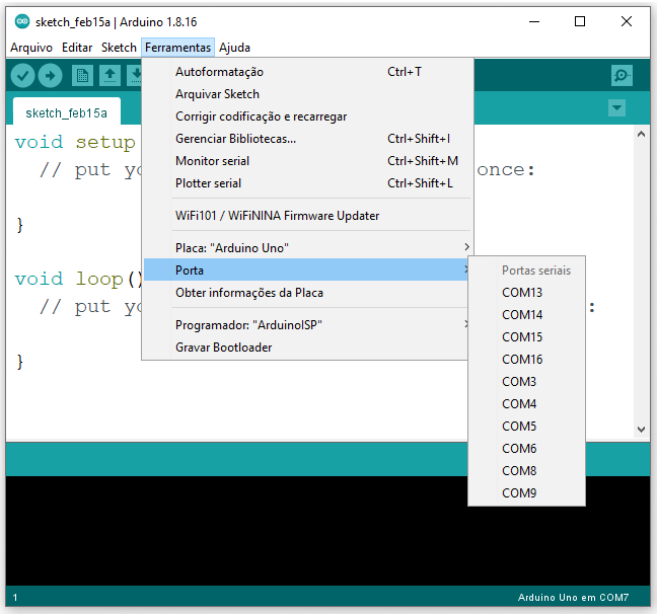

Figura 6: Você terá que selecionar a porta COM correta. Se você não tiver acesso como administrador no computador que você estiver, teremos que tentar uma a uma

PROFESSOR DANILO *ROBÓTICA – 9 ANO – 17/02/2022*

O próximo passo é carregar um programa simples no Arduino. Para isso, carregue um exemplo simples chamado Blink.

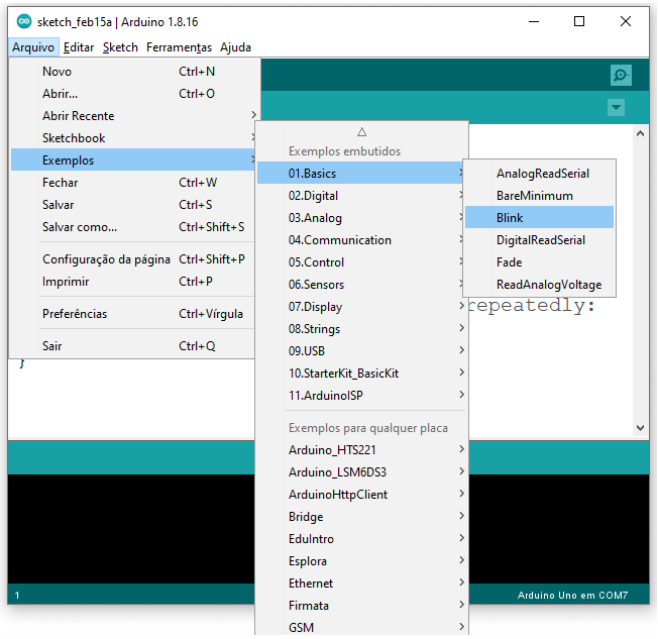

Figura 7: Clique em Arquivo > 01. Basics > Blink

O programa abaixo será aberto:

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/

// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Chamamos de skech os programas feitos para rodar no Arduino. No canto superior esquerdo tem um botão que serve para verificar se o programa está adequado .

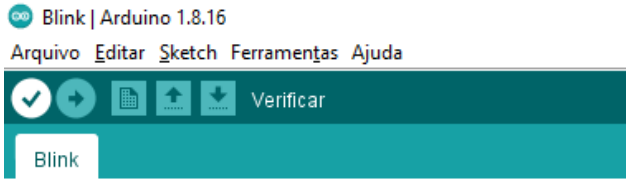

Figura 8: Ao clicar em verificar, o sketch que você fez será verificado e, caso tenha algum erro, será apresentada uma mensagem de erro. Caso tudo ocorra bem, o programa irá converter as instruções neste skech de forma que o Arduino entenda. Isso é chamado de compilação.

Depois, clique em carregar, conforme apresentado na figura abaixo. Se você selecionou a porta correta, então um LED no Arduino começará a piscar. Caso dê errado, você deverá tentar selecionar outra porta.

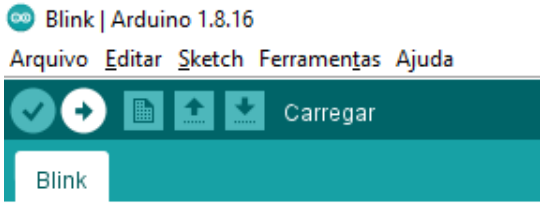

Figura 9: Clique agora em carregar. Se tudo correr bem, um LED incorporado no Arduino começará a piscar. Caso dê errado, selecione outra porta e tente novamente.

Figura 10: Arduino Uno com LED incorporado destacado. Veja que se tudo correu bem, este LED terá que ficar ligado por um segundo e desligado por um segundo.

Se a porta escolhida não for a correta, o erro a seguir irá aparecer e você deverá tentar conectar em outra porta.

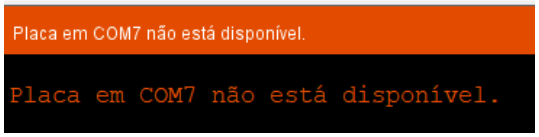

Figura 11: Erro que poderá aparecer caso a porta escolhida não seja a que está conectada o Arduino

PROFESSOR DANILO *ROBÓTICA – 9 ANO – 17/02/2022*

Agora, vemos tentar entender o que o programa anterios faz.

ANALISANDO O PROGRAMA BLINK

A primeira parte do programa, que está destacada abaixo, entre os símbolos /* e */, representa um comentário: nele, você escreve o que quiser, preferencialmente colocando informações sobre o programa e o circuito que deverá ser montado para este programa.

```
/*
  Blink

  Turns an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino
  model, check the Technical Specs of your board at:
  https://www.arduino.cc/en/Main/Products

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/
```

Tente traduzir o comentário no programa. Veja ele diz que o LED incorporado pisca, ligando por um segundo e desligando por um segundo.
Abaixo, temos mais uma forma de comentário: tudo o que tiver no mesmo parágrafo e à direita do símbolo // será entendido como comentário, portanto, não será compilado e enviado para o Arduino.

```
// the setup function runs once when you press reset or power the board
```

Veja que ao longo do programa há muitas linhas com comentários deste tipo.
Então chegamos na função se configuração, chamada setup(), que significa “configuração”. Veremos mais para frente o que é uma função, veremos que o que está entre parêntesis é o que chamamos de argumento, mas a função setup() não recebe argumento nenhum, ou seja, não colocamos nada entre parêntesis.
Também aprenderemos que uma função retorna algum tipo de dado, podendo ser um número, letra, vetor (conjunto de números), string (conjunto de caracteres) ou algum outro tipo de dado. O tipo de dado que a função retorna é escrito à esquerda da função. Note então que esta função retorna um tipo void, cuja tradução seria “vazio”. Ou seja, a função setup não recebe argumento e não retorna nada (retorna vazio).
À frente de void setup() tempos um abre chaves e, algumas linhas abaixo, um fecha chaves. Tudo o que estiver entre ambos será usado para configurar o Arduino.
Portanto, tudo o que estiver à diante de void setup() é o que será usado para configurar o Arduino.

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}
```

Veja que na primeira linha dentro do setup temos um comentário. Na linha seguinte, temos uma nova função.

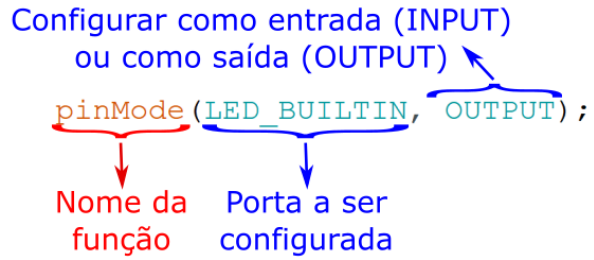

Figura 12: Função pinMode e seus argumentos. Esta função serve para configurar as portas do Arduino como entrada ou saída. Vamos ver mais detalhes do que é entrada ou saída nas próximas aulas.

Temos uma nova função: a função loop(). Como o próprio nome dela sugere, o que estiver dentro desta função será executado indefinidamente, e só irá parar quando o Arduino for desligado. Vamos explorar o que esta função faz.

```
// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

O que estiver dentro do colchete, ou seja, à direita ou abaixo de void loop() {, será repetido sem parar. Veja que na primeira linha abaixo de void loop() temos uma função.

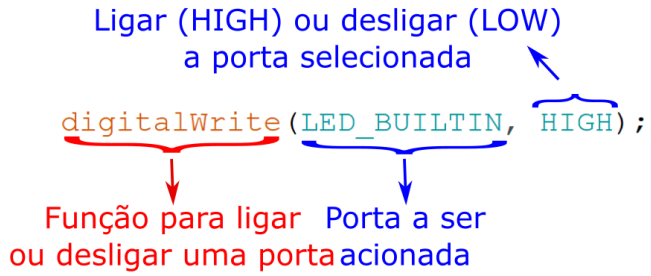

Figura 13:Função que é usada para ligar ou desligar uma das portas do Arduino. Podemos ligar ou deligar as portas de 1 à 13 e também as portas de A0 até A5.

Vejam as portas destacadas nas figuras abaixo.

Figura 14: Portas que funcionam apenas como saídas digitais, portanto, serve para ligar e desligar um LED, por exemplo.

PROFESSOR DANILO *ROBÓTICA – 9 ANO – 17/02/2022*

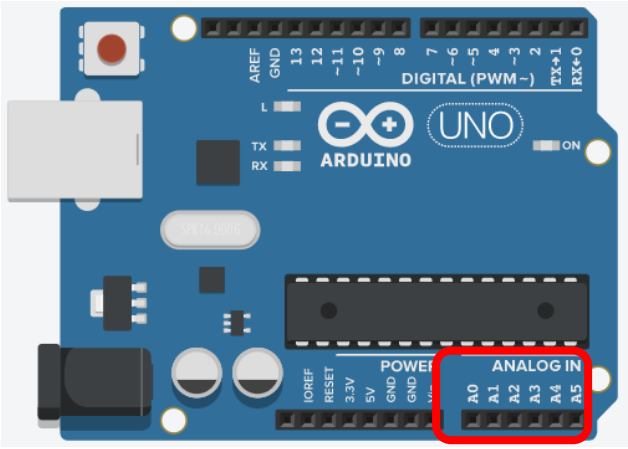

Figura 15: Portas normalmente chamadas de analógicas, porém também podem ser utilizadas como portas digitais, isto é, podem ser usadas para ligar ou deligar um LED, por exemplo

Voltando ao último código destacado, falta falar da função delay() que instrui o Arduino à ficar parado, sem fazer nada. Entre os parênctesis, devemos escolher quanto tempo o Arduino deve ficar ocioso, porém o tempo informado é dado em milissegundos.
Um milissegundo é um milésimo de um segundo, portanto, se quiser que o Arduino fique um tempo *t*, em segundo, ocioso você deve multiplicar este número por 1000. Por exemplo, se você quer que ele fique esperando por 3 segundos, basta muiltiplicar por 1000, ou seja, delay(3000) faz com que o Arduino fique parado por 3 segundos.

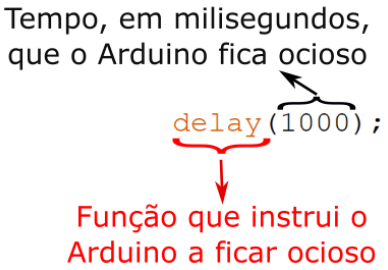

Figura 16: Função que faz com que o Arduino fique padaro, esperando para seguir com as novas instruções.

Agora você é capaz de descrever o que o Arduino deverá fazer olhando apenas sketch. Tente fazer isso...

BUSCANDO AJUDA NA COMUNIDADE

Faça alguns testes:

- Busque na internet por “delay arduino”;
- Busque também por “pinmode Arduino”;
- Busque também por “digitalwrite Arduino”.

Seja que todos estes resultados levam para um mesmo site:
https://www.arduino.cc/

Este site tem muita coisa legal: você pode fazer uma conta, salvar seus programas online, descobrir a porta que você usa automaticamente etc. Não iremos por este caminho, pois assim não precisaríamos criar uma conta online.
Há muitas outras funcionalidades que você pode ter acesso mesmo sem fazer nenhuma conta, como baixar o programa Arduino ou buscar por ajuda para usar as funções do Arduino.
É muito bom que você se habitue a fazer buscas na internet e consultar este site.
Você também pode, e deve, entrar no tinkercad para mexer com o Arduino, mesmo sem ter o Arduino.

Como exemplo, veja uma nota sobre a função delay(), copiada do site oficial (https://www.arduino.cc/reference/pt/language/functions/time/delay/)

Notas e Advertências

Mesmo que seja fácil fazer um LED piscar usando a função delay(), e muitos sketches usam delays pequenos para tarefas como debouncing de botões, o uso de delay() em um sketch possui desvantagens significantes. Nenhuma leitura de sensores, cálculos matemáticos, ou manipulação de pinos podem ocorrer durante a função delay(), para resumir, causa a pausa de qualquer atividade. Para métodos alternativos de controlar temporizações, veja o sketch Blink Sem Delay (Em Inglês), que verifica a função millis() até que o tempo suficiente tenha passado. Programadores mais habilidosos usualmente evitam o uso da função delay() para timing de eventos mais longos que dezenas de milissegundos, a menos que o sketch Arduino seja muito simples.

No entanto, certas coisas continuam a acontecer enquanto a função delay() está controlando o microcontrolador, porque a função delay não desativa interrupções. Comunicação serial recebida no pino RX é armazenada, valores PWM de (analogWrite) e estados dos pinos são mantidos, e interrupções externas irão funcionar como devem.

Não se assuste caso você veja alguma coisa e não consiga entender, pois ainda estamos nos nossos primeiros passos. Não deixe de perguntar para seu professor caso esteja tentando apresnder alguma coisa só e sinta alguma dificuldade para continuar.