Material utilizado para a construção do B-007

1 arduino uno Atmelga 328P

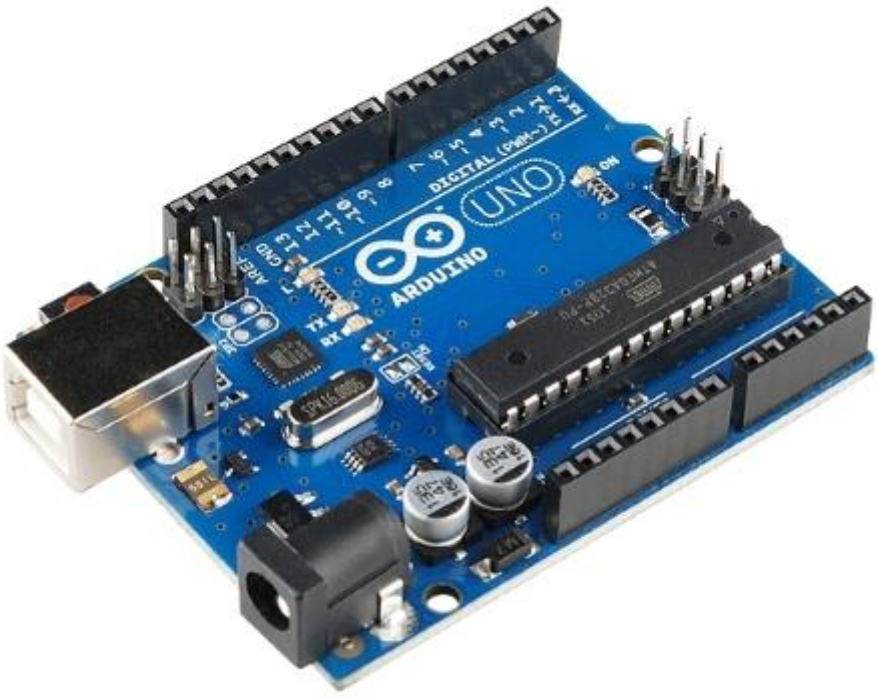
O QUE É ARDUINO?

Como nasceu o Arduino ? Para que serve um Arduino ? Quais as vantagens ? Como eu começo a programar? Nesse tutorial vamos apresentar um resumo sobre **o que é Arduino** e como você pode utilizá-la em seus projetos.

O QUE É ARDUINO

O **Arduino** foi criado em 2005 por um grupo de 5 pesquisadores : **Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis**. O objetivo era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além disso, foi adotado o conceito de hardware livre, o que significa que qualquer um pode montar, modificar, melhorar e personalizar o Arduino, partindo do mesmo hardware básico.

Assim, foi criada uma placa composta por um **microcontrolador Atmel**, circuitos de entrada/saída e que pode ser facilmente conectada à um computador e programada via **IDE** (*Integrated Development Environment*, ou *Ambiente de Desenvolvimento Integrado*) utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB.

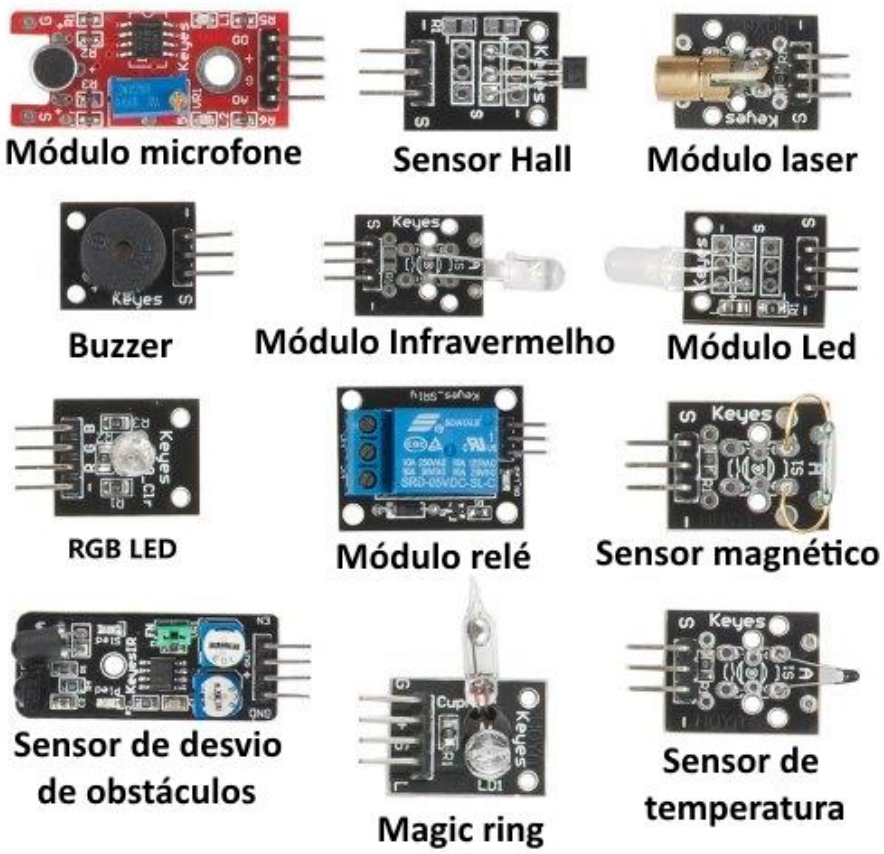


Depois de programado, o microcontrolador Arduino pode ser usado de forma independente, ou seja, você pode colocá-lo para controlar um robô, uma lixeira, um ventilador, as luzes da sua casa, a temperatura do ar condicionado, pode utilizá-lo como um aparelho de medição ou qualquer outro projeto que vier à cabeça.

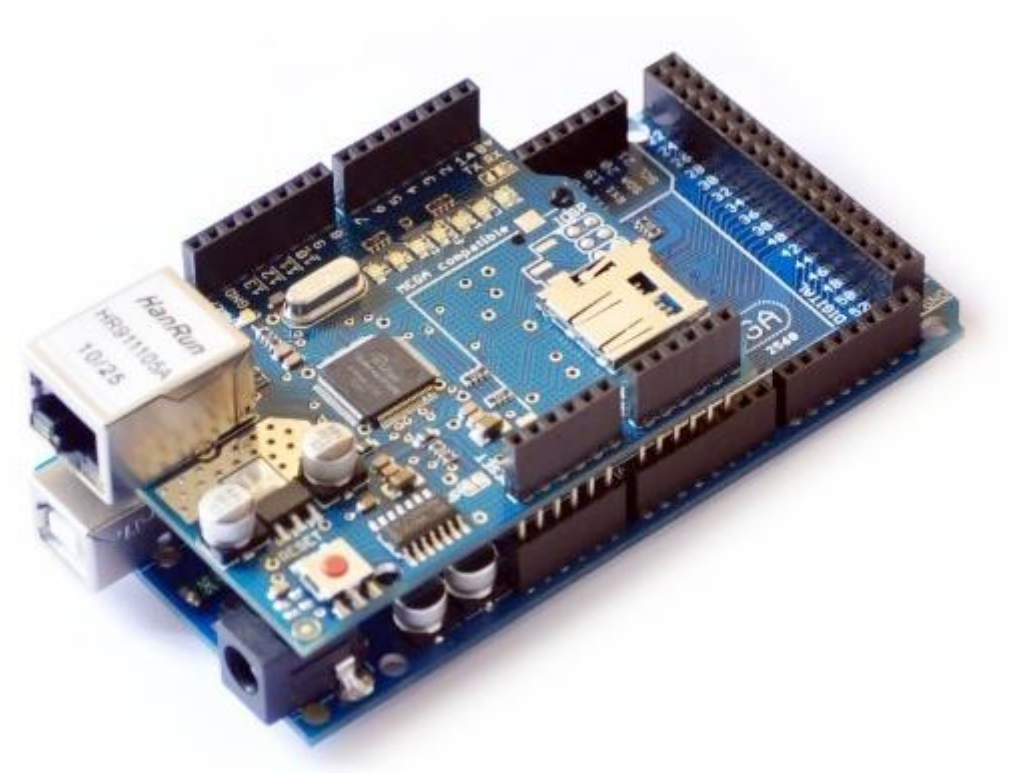
O QUE VOCÊ PODE FAZER COM O ARDUINO

A lista de possibilidades é praticamente infinita. Você pode automatizar sua casa, seu carro, seu escritório, criar um novo brinquedo, um novo equipamento ou melhorar um já existente. Tudo vai depender da sua criatividade.

Para isso, o Arduino possui uma quantidade enorme de sensores e componentes que você pode utilizar nos seus projetos. Grande parte do material utilizado no Arduino está disponível em módulos, que são pequenas placas que contém os sensores e outros componentes auxiliares como resistores, capacitores e leds.



Existem também os chamados Shields, que são placas que você encaixa no Arduino para expandir suas funcionalidades. A imagem abaixo mostra um **Arduino Ethernet Shield** encaixado no Arduino Mega 2560. Ao mesmo tempo que permite o acesso do Arduino à uma rede ou até mesmo à internet, mantém os demais pinos disponíveis para utilização, assim você consegue, por exemplo, utilizar os pinos para receber dados de temperatura e umidade de um ambiente, e consultar esses dados de qualquer lugar do planeta:



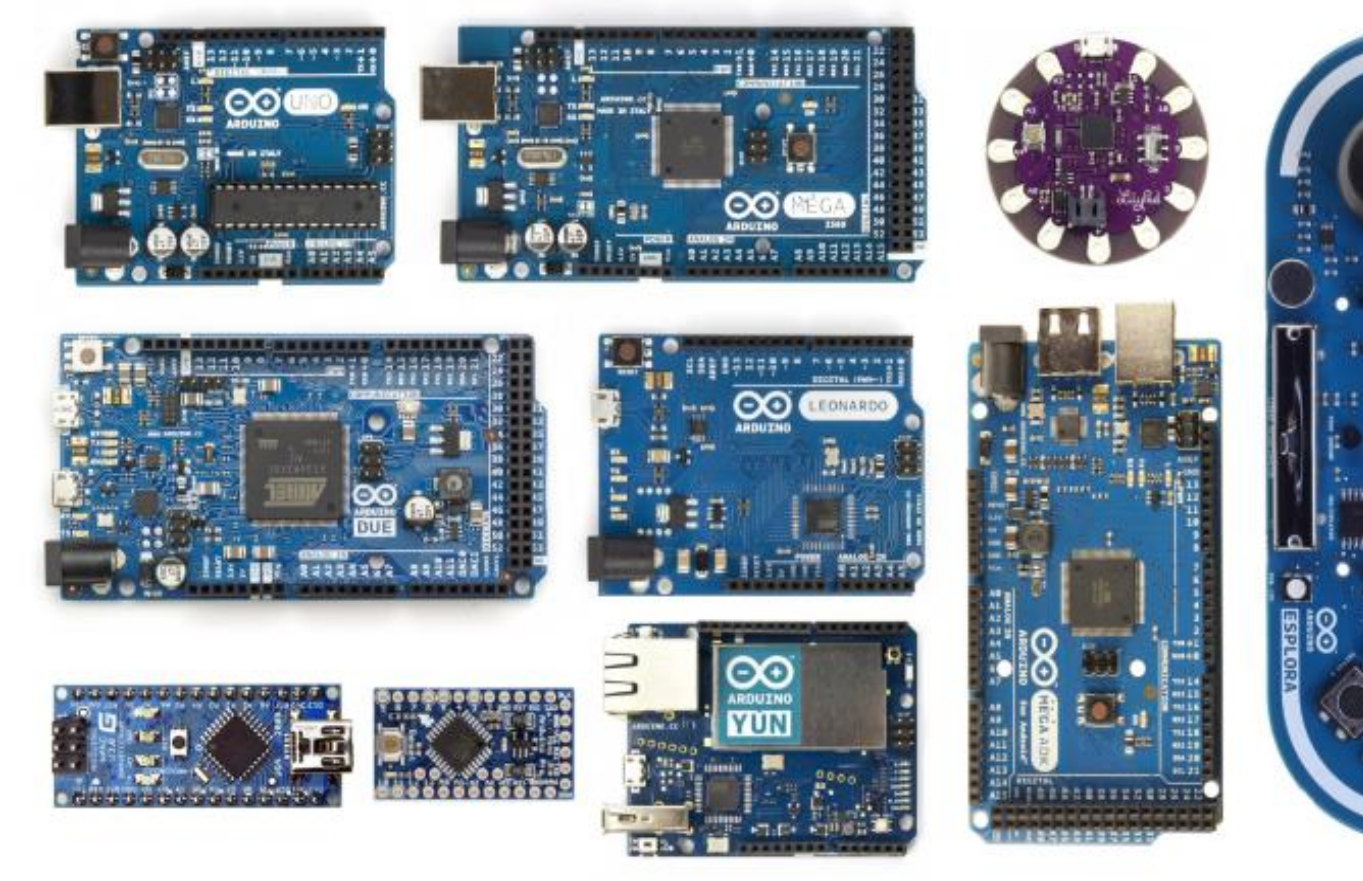
Para você ter uma idéia das possibilidades de criação com o Arduino, dê uma olhada nesses dois projetos (clique nas imagens para mais detalhes). O primeiro é de um tênis que se amarra sozinho…

… o outro é de um robô que sobe em árvores…

MODELOS DE PLACAS ARDUINO

O tipo de placa que você vai utilizar depende muito do projeto a ser desenvolvido e o número de portas necessárias. As opções vão das mais comuns, como o Arduino Uno e suas 14 portas digitais e 6 analógicas, passando por placas com maior poder de processamento, como o Arduino Mega, com **microcontrolador ATmega2560** e 54 portas

digitais, e o Arduino Due, baseado em **processador ARM de 32 bits** e 512 Kbytes de memória:

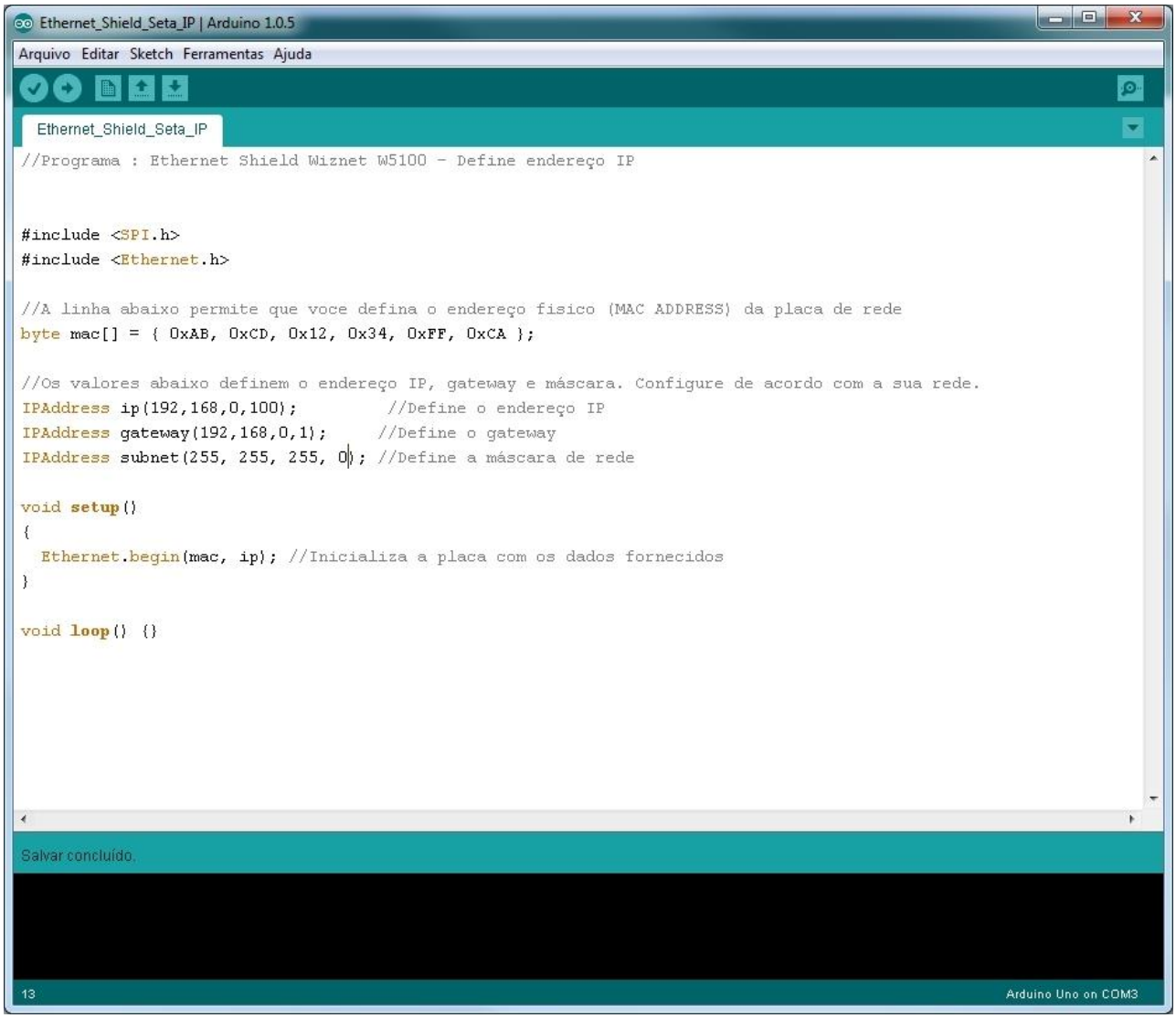


Aqui no blog temos um artigo específico abordando as principais placas Arduino disponíveis no mercado. Em Qual Arduino Comprar? Conheça os tipos de Arduino, você pode verificar as especificações, detalhes e características de cada placa.

ESTRUTURA DE UM PROGRAMA EM ARDUINO

Escrever um programa em Arduino é muito simples. Tudo o que você precisa é conectar o Arduino ao computador por meio de um cabo USB e utilizar um ambiente de programação chamado IDE, onde você digita o programa, faz os testes para encontrar eventuais erros e transfere o programa para o Arduino.

Na imagem abaixo temos a IDE já com um programa carregado. No site oficial do Arduino (arduino.cc), você pode fazer o download da IDE gratuitamente:



Uma vez feito o programa, basta transferí-lo para o Arduino e o mesmo começa a funcionar.

Você não precisa ser expert em linguagem C para programar o Arduino. Além da grande quantidade de exemplos que você encontra aqui no blog, você pode começar um programa utilizando a estrutura básica do Arduino, que é composta por duas partes, ou dois blocos:

setup() – É nessa parte do programa que você configura as opções iniciais do seu programa: os valores iniciais de uma variável, se uma porta será utilizada como entrada ou saída, mensagens para o usuário, etc.
loop() – Essa parte do programa repete uma estrutura de comandos de forma contínua ou até que alguma comando de "parar" seja enviado ao Arduino.
Vamos ver exatamente como isso funciona, levando em consideração o programa abaixo, que acende e apaga o led embutido na placa Arduino em intervalos de 1 segundo:

```
1    //Programa : Pisca Led Arduino

2    //Autor : FILIPEFLOP
```

```
3
4   void setup()
5   {
6     //Define a porta do led como saida
7     pinMode(13, OUTPUT);
8   }
9
    void loop()
10  {
11    //Acende o led
12    digitalWrite(13, HIGH);
13
14    //Aguarda o intervalo especificado
15    delay(1000);
16
17    //Apaga o led
18    digitalWrite(13, LOW);
19
20    //Aguarda o intervalo especificado
21    delay(1000);
    }
22
23
```

A primeira coisa que fazemos no início do programa é colocar uma pequena observação sobre o nome do programa, sua função e quem o criou:

```
1  //Programa : Pisca Led Arduino
2  //Autor : FILIPEFLOP
```

Comece uma linha com barras duplas (//) e tudo o que vier depois dessa linha será tratado como um comentário. Uma das boas práticas de programação é documentar o seu código por meio das linhas de comentário. Com elas, você pode inserir observações sobre como determinada parte do programa funciona ou o que significa aquela variável **AbsXPT** que você criou. Isso será útil não só para você, se precisar alterar o código depois de algum tempo, como também para outras pessoas que utilizarão o seu programa.

Após os comentários, vem a estrutura do **SETUP.** É nela que definimos que o pino 13 do Arduino será utilizado como saída.

```
4  void setup()
5  {
6    //Define a porta do led como saida
7    pinMode(13, OUTPUT);
8  }
```

Por último, temos o **LOOP**, que contém as instruções para acender e apagar o led, e também o intervalo entre essas ações:

```
10 void loop()
11 {
12   //Acende o led
13   digitalWrite(13, HIGH);
14
15   //Aguarda o intervalo especificado
16   delay(1000);
17
18   //Apaga o led
19   digitalWrite(13, LOW);
20
21   //Aguarda o intervalo especificado
22   delay(1000);
23 }
```

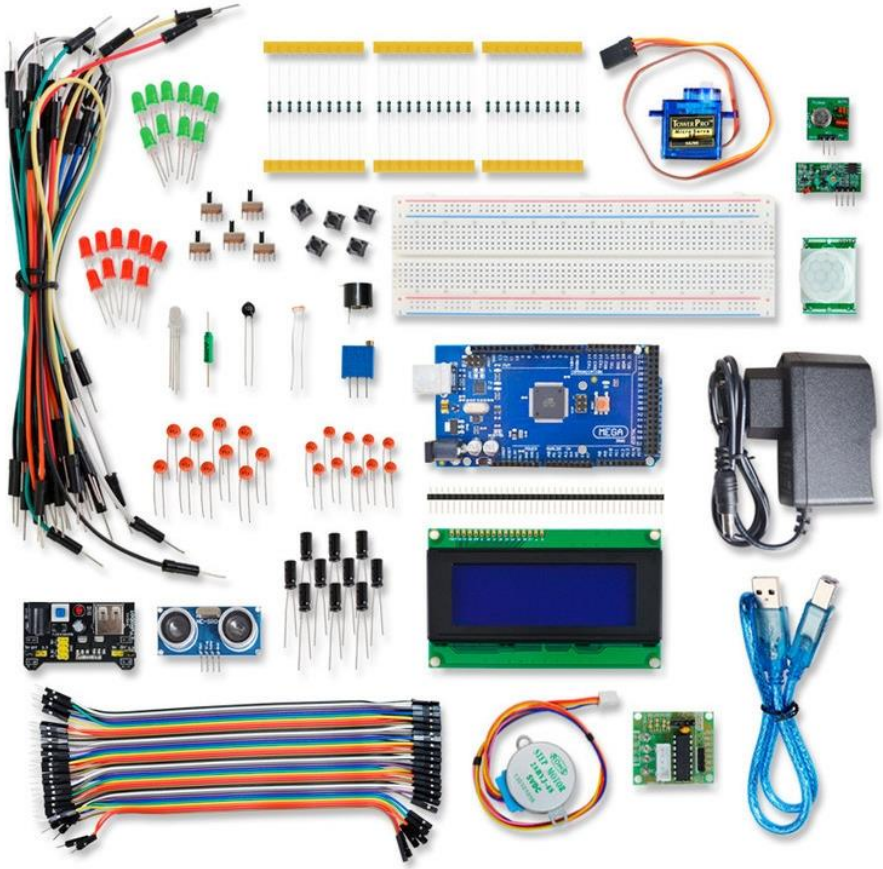
A linha do código contendo **digitalWrite(13, HIGH)** coloca a porta 13 em nível alto (**HIGH**, ou 1), acendendo o led embutido na placa. O comando **delay(1000)**, especifica o intervalo, em milissegundos, no qual o programa fica parado antes de avançar para a próxima linha. O comando **digitalWrite(13, LOW)**, apaga o led, colocando a porta em nível baixo (**LOW**, ou 0), e depois ocorre uma nova parada no programa, e o processo é então reiniciado.

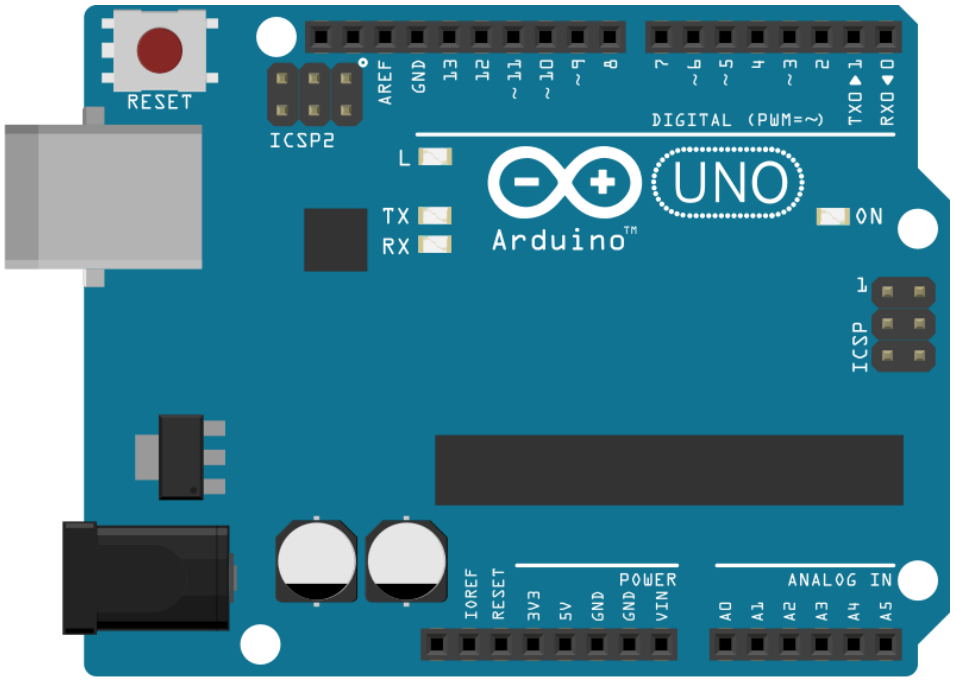
COMEÇANDO COM O ARDUINO

Como vimos acima, você não precisa de nenhum componente adicional para começar a programar um Arduino. Basta um computador, uma placa Arduino e a IDE para efetuar a programação e enviar o programa para a placa.

Depois que você der os primeiros passos com o Arduino, vale a pena investir em alguns módulos ou até mesmo nos kits de desenvolvimento,

Para quem está iniciando, temos o Kit Arduino Start, onde além da placa Arduino, você tem resistores, leds e sensor de temperatura, além de diversos outros componentes. Temos também o Kit Beginning, onde você pode começar a mexer com displays e sensores ultrassônicos. Para níveis mais avançados, temos o Kit Advanced, que já vem com o **Arduino Mega 2560:**




RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

Descrição

O Uno difere de todas as placas antecessoras no sentido de não utilizar o chip FTDI cara conversão do sinal serial. Utiliza no seu lugar um Atmega8U2 programado como conversor de USB para serial.

Revisão 3 da placa com as novas características:

- Pinos SDA e SCL adicionados próximos ao AREF.
- Dois outros pinos adicionados próximos ao RESET, o IOREF que permite aos shields se adaptarem à voltagem fornecida pela placa. No futuro os shields serão compatíveis tanto com as placas que utilizam o AVR e operam a 5V, como com o ARduino Due que operará a 3,3V. O segundo pino não está conectado e é resrvado para propósitos futuros.
- Circuito de RESET mais robusto.
- Atmega 16U2 em substituição ao 8U2.

"Uno" quer dizer um em italiano e é utilizado para marcar o lançamento do Arduino 1.0. O Uno e a versão 1.0 serão as versões de referência do Arduino, daqui para diante. O UNO é o mais recente de uma série de placas Arduino, e o modelo de referência para a plataforma Arduino. Para uma comparação com as versões prévias veja o índice de placas Arduino.

Características

Microcontrolador	ATmega328
Voltagem Operacional	5V
Voltagem de entrada (recomendada)	7-12V
Voltagem de entrada (limites)	6-20V
Pinos E/S digitais	14 (dos quais 6 podems ser saídas PWM)
analógica	6
Corrente CC por pino E/S	40 mA
Corrente CC para o pino 3,3V	50 mA
Flash Memory	32 KB (ATmega328) dos quais 0,5KB são utilizados pelo bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Velocidade de Clock	16 MHz

Alimentação

O arduino Uno pode ser alimentado pela conexão USB ou com uma fonte de alimentação externa. A alimentação é selecionada automaticamente.

Alimentação externa (não USB) pode ser tanto de um adaptador CA para CC ou bateria. Há um conector para alimentação de 2,1mm com o positivo no centro. Cabos vindos de uma bateria podem ser inseridos diretamente nos pinos Gnd e Vin do conector de alimentação.

Esta placa pode funcionar com uma fonte de alimentação externa de 6 a 20 volts. No entanto se a alimentação for inferior a 7V, o pino 5V pode fornecer menos de cinco volts e

a placa pode se mostrar instável. E se a alimetação for maior do que 12V o regulador de voltagem pode superaquecer e danificar a placa. A faixa recomendada é de 7 a 12 volts.

Os pinos de alimentação são os seguintes:

- **VIN.** A entrada de alimentação para a placa Arduino quando se está utilizando uma fonte de alimentação externa. (em oposição à conexão USB ou outra fonte de aliemteção regulada). Você pode fornecer alimentação através deste pino, ou se estiver utilizando o conector de alimentação acessar esta voltagem aqui.
- **5V.** A fonte de alimentação regulada usada para o microcontrolador e para outros componentes na placa. Pode vir tanto do VIN através do regulador embarcado ou da conexão USB ou outra fonte regulada em 5V.
- **3V3.** Uma fonte de 3,3V gerada pelo regulador embarcado. A corrente máxima suportada é de 50mA.
- **GND.** Pinos terra.

Memória

O ATmega328 têm 32KB (dos quais 0,5 são utilizados pelo bootloader). Também tem 2KB de SRAM e 1KB de EEPROM (que pode ser lido ou gravado com a biblioteca EEPROM).

Entrada e Saída

Cada um dos 14 pinos digitais do Uno podem ser utilizados como uma entrada ou uma saída utilizando-se as funções pinMode(), digitalWrite(), e digitalRead(). Eles operam a 5V. Cada pino pode fornecer ou receber um máximo de 40mA e tem um resistor pull-up interno (desconectado por padrão) de 20-50kΩ. Além disso alguns pinos tem funções especializadas:

- **Serial: 0 (RX) e 1 (TX).** Usados para receber (RX) e transmitir (TX) dados seriais TTL. Estes pinos são conectados aos pinos correspondentes do chip serial USB-para-TL ATmega8U2.
- **Interruptores Externos: 2 e 3.** Estes pinos podem ser configurados para disparar uma interrupção de acordo com alguma variação sensível pelo circuito. Veja a função attachInterrupt() para mais detalhes.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** Estes pinos dão suporte à comunicação SPI utilizando a biblioteca SPI.
- **LED: 13.** Há um LED integrado ao pino digital 13. Quando este pino está no valor HIGH este LED está aceso, quando o pino está em LOW o LED está apagado.
- **I²C: 4 (SDA) and 5 (SCL).** Fornecem suporte a comunicação I²C (TWI) utilizando a biblioteca Wire.

- **AREF.** Voltagem de referência para as entradas analógicas. Utilizado com a função analogReference().
- **Reset.** Envio o valor LOW para esta linha para resetar o microcontrolador. Tipicamente usado para adcionar um botão de de reset para shileds montados sobre a placa original.

O Uno tem 6 entradas analógicas, etiquetadas de A0 a A5, cada uma tem 10 bits de resolução (i.e. 1024 valores diferentes). Por padrão elas medem de 0 a 5V, embora seja possível alterar o limite superior utilizando o pino AREF e a função analogReference().

Comunicação

O Arduino Uno possui uma série de facilidades para se comunicar com um computador, outro Arduino, ou outros microcontroladores. O ATmega328 fornece comunicação serial UART TTL (5V) que está disponível nos pinos digitais 0 (RX) e 1 (TX). Um ATmega8U2 na placa canaliza esta comunicação para a USB e aparece como uma porta virtual para o software no computador. O firmware do '8U2 utiliza os drivers padrão USB COM e nenhum driver externo é necessário. Entretanto, no Windows, um arquivo .inf é necessário. Ainda faltam as instruções específicas mas em breve estarão disponíveis. O software do Arduino inclui um monitor serial que permite dados textuais ser enviados e recebidos da placa. LEDs conectados ao RX e TX piscarão enquanto dados estiverem sido transmitidos pelo chip USB-para-serial e pela conexão USB (mas não para comunicação serial nos pinos 0 e 1).

Uma biblioteca de SoftwareSerial permite comunicação serial em qualquer dos pinos digitais do Uno.

O ATmega328 também suporta comunicação I^2C (TWI) e SPI. O software do Arduino inclui uma biblioteca Wire para simplificar o uso do bus I^2C, veja a documentação para mais detalhes. Para comunicação SPI utilize a biblioteca SPI.

Programação

O Arduino Uno pode ser programado com o software Arduino (download). Simplesmente selecione "Arduino Uno" no menu Tools > Board.

O ATmega328 no Arduino Uno vem pré-gravado com um bootloader que permite a você enviar código novo para ele sem a utilização de um programador de hardware externo. Ele se comunica utilizando o protocolo original STK500 (referência, arquivos C header).

Você também pode saltar o bootloader e programar o microcontroaldor através do conector ICSP (In-Circuit Serial Programming); veja estas instruções para mais detalhes.

O código fonte do firmware do ATmega8U2 também está disponível. Este chip é carregado com um bootloader DFU, que pode ser ativado conectando o jumper de solda na parte posterior da placa (próximo ao mapa da Itália) e depois resetando o 8U2. Você pode utilizar o software FLIP da Atmel (Windows) ou o programador DFU (Mac OS X e Linux) para carregar um novo firmware. Ou ainda utilizar um programador externo (sobrescrevendo o bootloader DFU).

Reset automatico por software

Ao invés de necessitar do pressionamento físico de um botão antes de um upload, o Arduino Uno é desenvolvido de maneira que permita que esta operação seja feita por meio do software rodando em um computador. Uma das linhas de controle de fluxo do hardware (DTR) do ATmega8U2 é conectado à linha de reset do ATmega328 através de um capacitor de 100nF. Quando esta linha é declarada (rebaixada) a linha de reset cai o suficiente para resetar o chip. O software do Arduino utiliza esta capacidade para permitir o envio de código novo simplesmente pressionando o botão de upload na IDE. Isto significa que o bootloader pode ter um intervalo mais curto, uma vez que o rebaixamento do DTR pode ser melhor coordenado com o início do upload.

Esta configuração tem outras implicações. Quando o Uno é conectado a um computador rodando Mac OS X ou Linux, ele é resetado cada vez que uma conexão é estabelecida com o software (via USB). Durante o próximo meio segundo o bootloader estará rodando no Uno. Uma vez que ele está programado para ignorar dados malformados (i.e. qualquer coisa diferente do upload de um novo código), ele irá interceptar os primeiros bytes de informação após a abertura da conexão. Se um programa rodando na placa recebe alguma configuração ou outra informação quando começa a rodar esteja seguro de que o software com o qual ela se comunica espere por um segundo antes de começar a enviar dados.

O Uno contem uma trilha que pode ser interrompida (cortada fisicamente) para desabilitar o auto-reset. Os conectores de cada lado da trilha podem ser soldados para reabilitar esta função. Ela está identificada como "RESET-EN". Você também pode desabilitar o auto-reset conectando um resistor de 110Ω do 5V à linha de reset, veja este tópico do fórum para mais detalhes.

Proteção contra sobre-corrente na USB

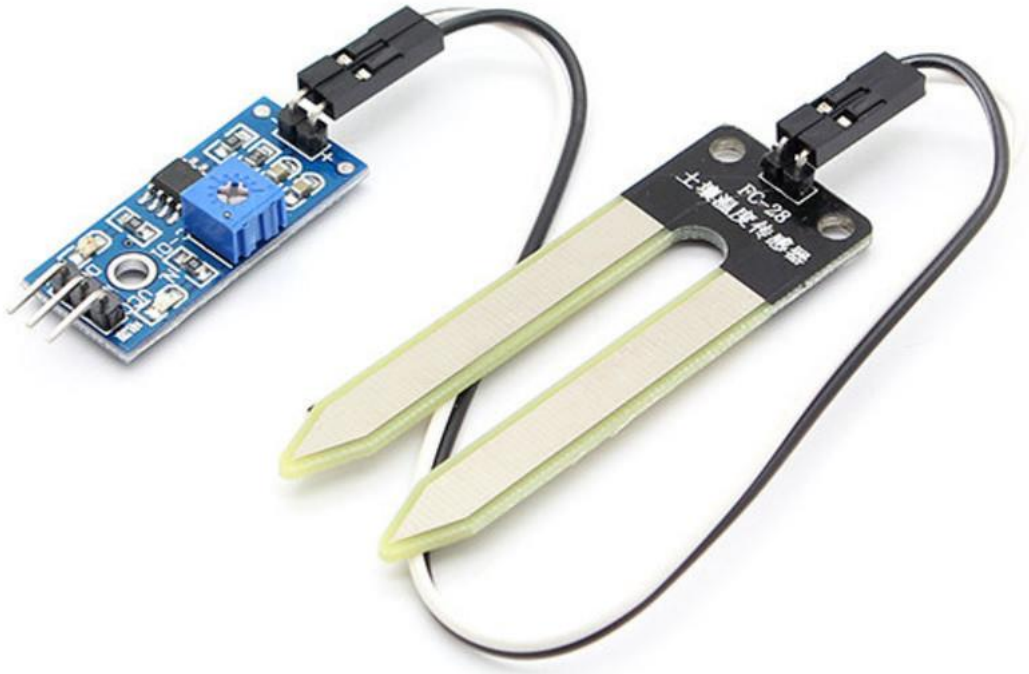
O Arduino Uno possui um *polyfuse* resetável que protege a porta USB do seu computador contra sobre-corrente e curtos circuitos. Embora muitos computadores tenham sua própria proteção interna, o fuso fornece uma camada a mais de proteção. Se mais de 500mA forem aplicados a porta USB ele automaticamente irá interromper a conexão até que o curto ou a sobrecarga seja removido.

Características físicas

A largura e o comprimento máximos do PCB do Uno são 68,58 e 53,34mm respectivamente (2,7" x 2,1"), com os conectores USB e de alimentação extendendo-se além destas dimensões. Quatro orifícios para parafusos permitem que a placa seja fixada a uma superfície ou encapsulamento. Verifique que a distância entre os pinos digitais 7 e 8 é de 160mil (milésimos de polegada ou 0,16"), não é nem mesmo um múltiplo dos 100mil que separam os outros pinos.

Fonte : https://multilogica-shop.com/arduino-uno-r3

Sensor de Umidade do Solo Higrômetro



Este sensor foi feito para detectar as variações de umidade no solo, sendo que quando o solo está seco a saída do sensor fica em estado alto, e quando úmido em estado baixo.

O limite entre seco e úmido pode ser ajustado através do potenciômetro presente no sensor que regulará a saída digital D0. Contudo para ter uma resolução melhor é possível utilizar a saída analógica A0 e conectar a um conversor AD, como a presente no Arduino por exemplo.

Especificações:
- Tensão de Operação: 3,3-5v
- Sensibilidade ajustável via potenciômetro
- Saída Digital e Analógica
- Fácil instalação
- Led indicador para tensão (vermelho)
- Led indicador para saída digital (verde)
- Comparador LM393
- Dimensões PCB: 3x1,5 cm

- Dimensões Sonda: 6x2 cm
- Comprimento Cabo: 21 cm

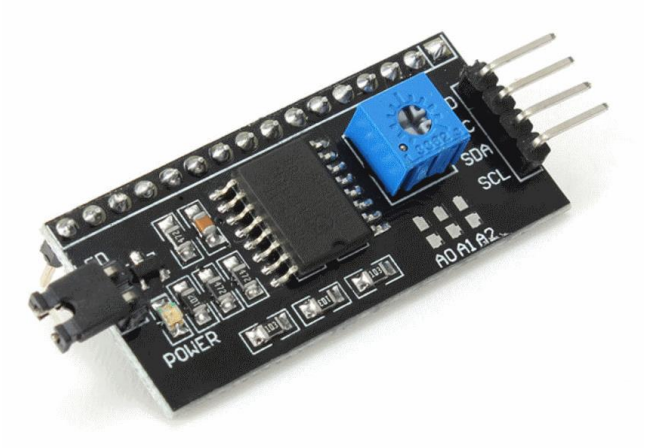
Pinagem:
- VCC: 3,3-5v
- GND: GND
- D0: Saída Digital
- A0: Saída analógica

1 Módulo I2C com display LCD

Como utilizar o módulo I2C com display LCD

Quem precisa conectar um display LCD 16x2 ou 20x4 ao Arduino sabe que vai precisar de pelo menos 6 fios para conexão. Em placas com um número menor de portas, como o Arduino Uno, isso significa sacrificar algumas portas que poderiam ser utilizadas para ligação de outros componentes, como sensores ou motores.

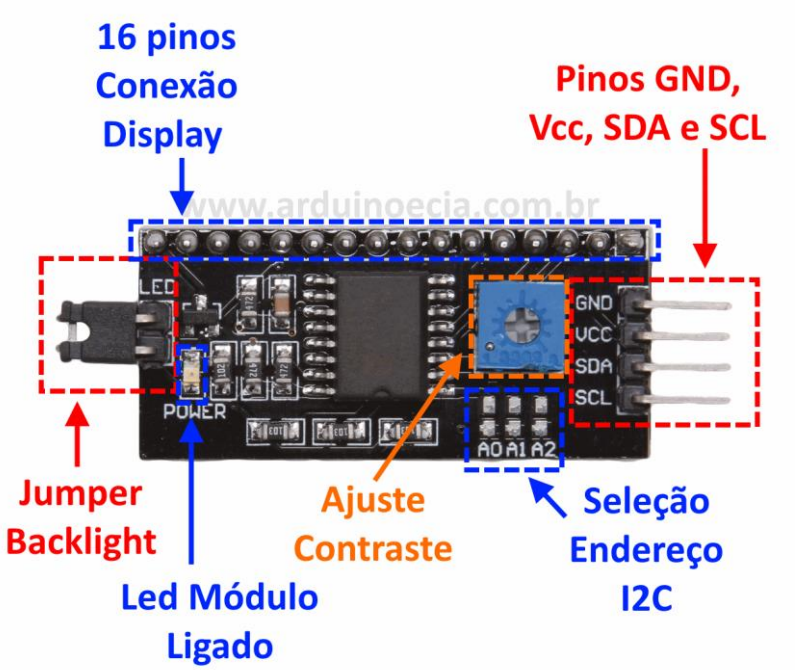
Um módulo que pode ser utilizado para contornar esse problema é o módulo I2C para display LCD com CI **PCF8574** (datasheet) :



Com esse módulo, você consegue controlar um display LCD, seja ele 16x2 ou 20x4, utilizando apenas dois pinos do Arduino : o **pino analógico 4 (SDA)** e o **pino analógico 5 (SCL)**, que formam a interface de comunicação I2C.

Estrutura do módulo I2C

Na lateral esquerda do módulo temos 4 pinos, sendo que dois são para alimentação (**Vcc** e **GND**), e os outros dois são da interface **I2C (SDA e SCL)**. O potenciômetro da placa serve para ajuste do contraste do display, e o jumper na lateral oposta permite que a luz de fundo (backlight) seja controlada pelo programa ou permaneça apagada.



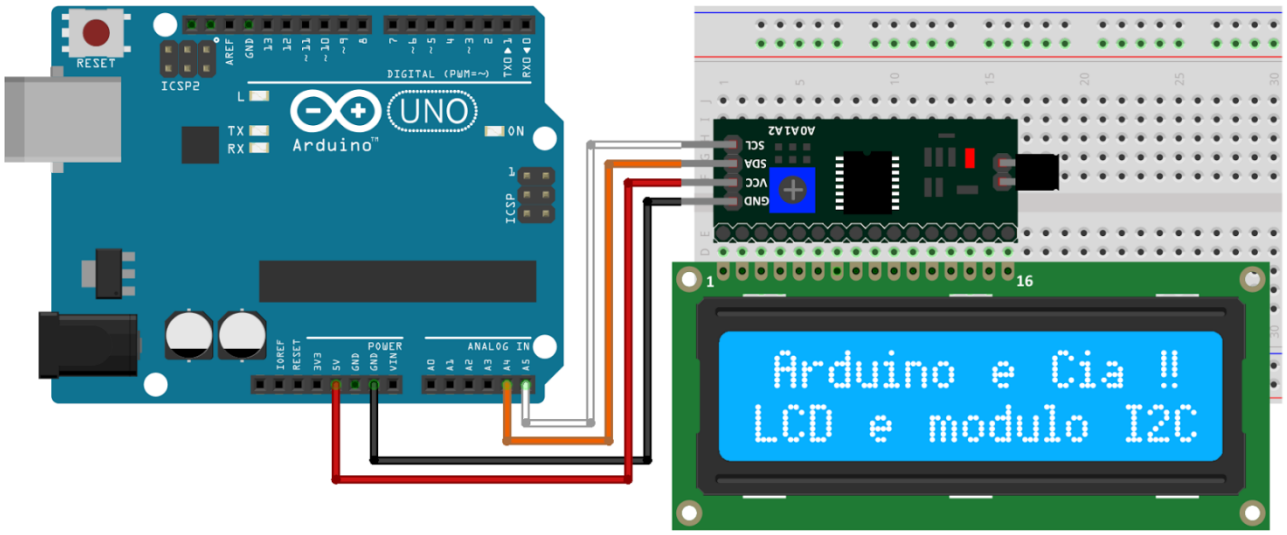
Por padrão o módulo vem configurado com o endereço **0x27**, mas você pode alterar esse endereço utilizando os pinos **A0, A1 e A2** seguindo a tabela abaixo :

Endereço	A0	A1	A2
0x20	0	0	0
0x21	1	0	0
0x22	0	1	0
0x23	1	1	0
0x24	0	0	1
0x25	1	0	1
0x26	0	1	1
0x27	1	1	1

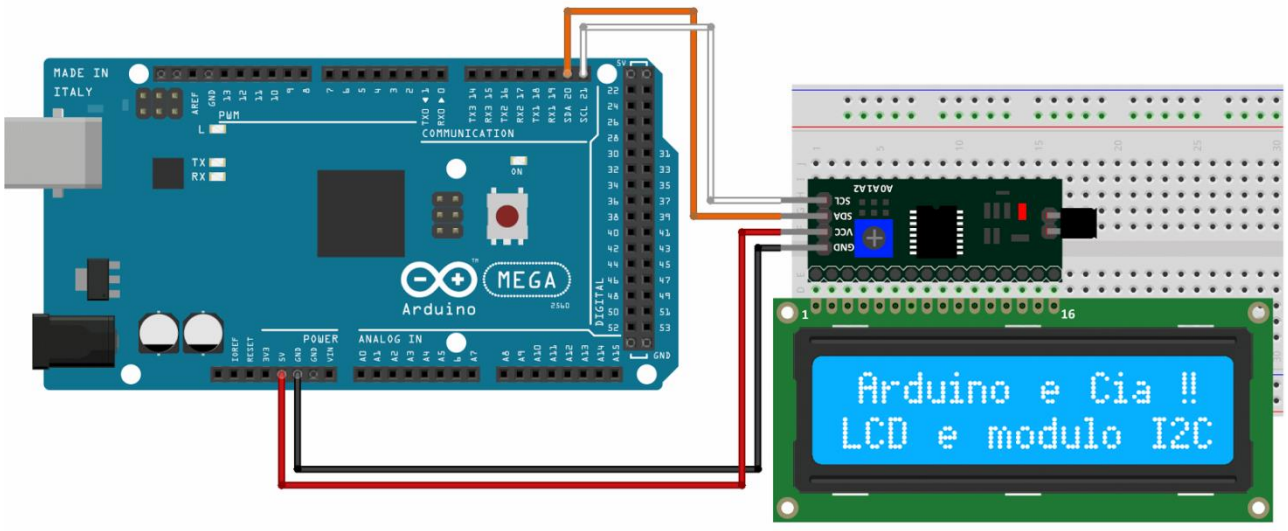
Em alguns casos o módulo I2C pode vir configurado com uma faixa de endereços diferente, para descobrir qual o endereço do módulo, utilize o programa **I2C Scanner**, que eu mostro nesse post.

Ligação do módulo I2C ao display LCD

O módulo possui 16 pinos que podem ser ligados diretamente ao display, ou você pode testar a conexão na protoboard, como eu fiz montando o circuito abaixo, onde utilizei um display LCD 16x2 com controlador HD44780 ligado ao Arduino Uno :



Caso você esteja usando um Arduino Mega 2560, utilize os pinos **20 (SDA)** e **21 (SCL)** :



Programa e biblioteca LiquidCrystal_I2C

Para controlar esse módulo I2C, utilize a biblioteca **LiquidCrystal_I2C**, disponível nesse link. Descompacte o arquivo ZIP e renomeie a pasta **LiquidCrystal** para **LiquidCrystalI2C**, copiando-a para a pasta **LIBRARIES** da **IDE** do seu Arduino. Renomear a pasta evita que você tenha conflitos com a biblioteca LiquidCrystal que já vem embutida na IDE.

Os comandos para controle do display são praticamente os mesmos da biblioteca *LiquidCrystal* que utilizamos normalmente, com comandos como **lcd.begin()**, **lcd.print()** e **lcd.setCursor()**. Na biblioteca I2C, o comando **lcd.setBacklight()** liga (**HIGH**) ou desliga (**LOW**) a luz de fundo do display.

```
1  // Programa : Display LCD 16x2 e modulo I2C
2  // Autor : Arduino e Cia
3
4  #include <Wire.h>
5  #include <LiquidCrystal_I2C.h>
6
7  // Inicializa o display no endereco 0x27
8  LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7,3, POSITIVE);
9
10 void setup()
11 {
12  lcd.begin (16,2);
13 }
14
15 void loop()
16 {
17   lcd.setBacklight(HIGH);
18   lcd.setCursor(0,0);
19   lcd.print("Arduino e Cia !!");
20   lcd.setCursor(0,1);
21   lcd.print("LCD e modulo I2C");
22   delay(1000);
23   lcd.setBacklight(LOW);
24   delay(1000);
25 }
```

Fonte: http://www.arduinoecia.com.br/2014/12/modulo-i2c-display-16x2-arduino.html

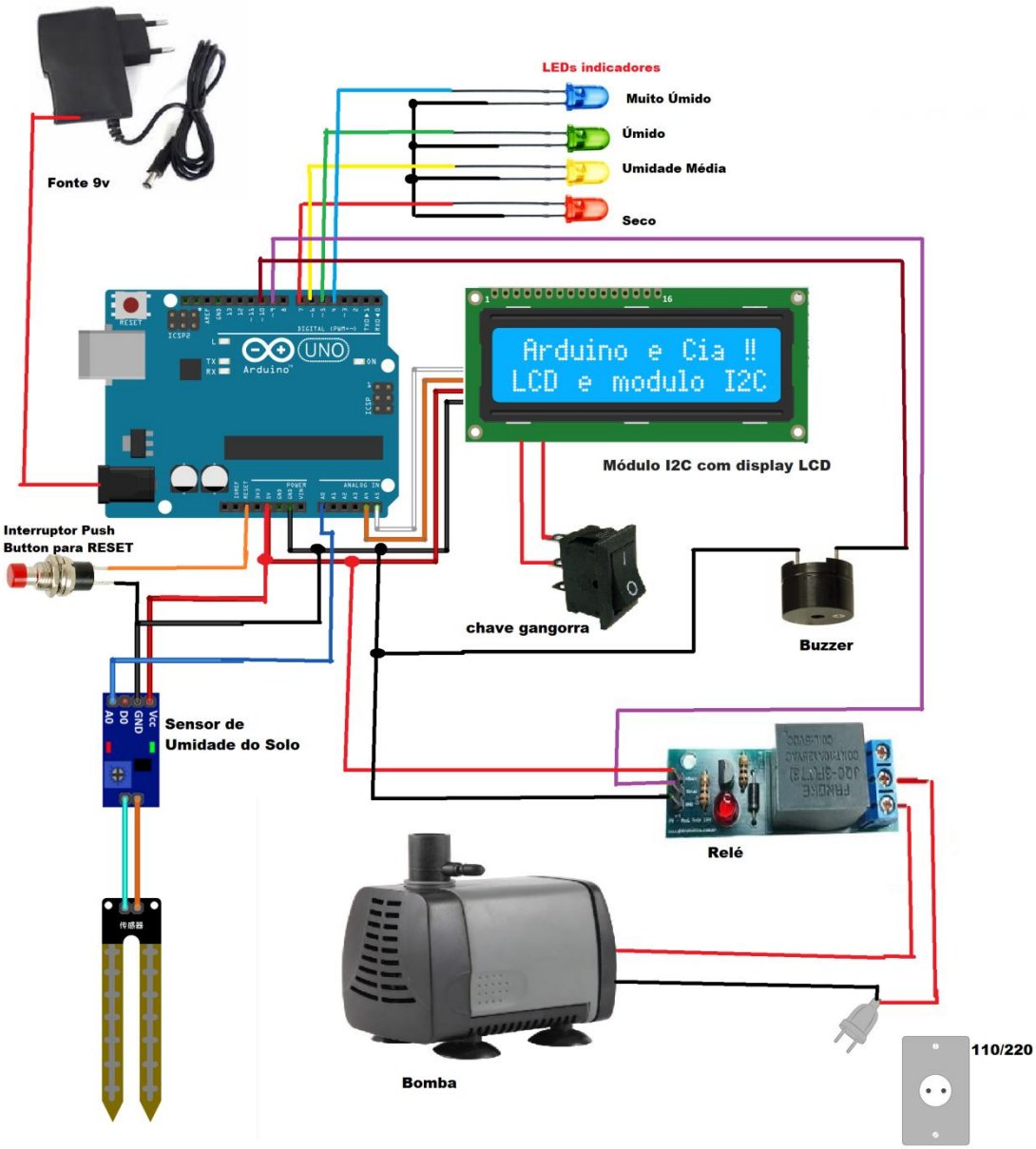
1 chave gangorra ON/OFF

1 Push Button Switch

4 Leds

1 Buzzer

Diagrama de Montagem do B-007 v:1.0



Obs: certifique-se de que todas as bibliotecas estejam instaladas na IDE do Arduino para evitar erros.