ARDUINO
Básico

Apostila do minicurso

Sumário

1. História 4
2. Aplicações de Arduino 4
3. Hardware 6
3.1. Microcontrolador: Atmel ATMEGA328P 6
3.2. Porta USB 6
3.3. Botão Reset 6
3.4. Conector P4 7
3.5. Pinos de alimentação e referência 7
3.6. Pinos analógicos: 7
3.7. Pinos digitais: 7
3.8. LED ON 8
3.9. Capacitores 8
4. Grandezas analógicas e digitais 8
4.1. Relembrando: O que são Bits 8
4.2. Grandezas 9
4.3. PWM 10
5. Instalação da IDE 11
5.1. Verificação do Driver 14
5.2. Arduino IDE 16
6. Componentes Eletrônicos 18
6.1. Protoboard 18
6.2. Capacitores 18
6.3. Diodo 19
6.4. Diodos Emissores de Luz (LEDs) 19
6.5. Ponte H 20
6.6. Display de Cristal Líquido (LCD) 22
6.7. Resistor 22
6.8. Potenciômetro 23
6.9. Resistor dependente de luz 23
6.10. Servo motor 24
6.11. Piezo 25

6.12. Motor DC 26
6.13. Botões 26
6.14. Sensor de temperatura – LM35 27
6.15. Transistor 27
7. Variáveis 29
7.1. Variáveis mais usadas 30
7.2. Variáveis Constantes 31
8. Funções próprias do arduino 32
8.1. Função Setup 32
8.2. Função Loop 33
8.3. Modo dos pinos 33
8.4. Leitura e envio de sinal digital 34
8.5. Leitura e envio de sinal analógico 34
8.6. Referência analógica de tensão 34
8.7. Funções de comunicação serial 35
8.8. Outras funções 35
9. Referências 36

1. História

Segundo o site oficial da Arduino, o Arduino é uma plataforma open-source de prototipagem eletrônica com hardware e softwares flexíveis e fáceis de usar, destinado a artistas, designers, hobbistas e qualquer pessoa interessada em criar objetos ou ambientes interativos. Um projeto dito open-source (código aberto) é um modelo de desenvolvimento que o licenciamento é livre, isto é, é possível examinar e modificar o produto sem a necessidade de pagar uma licença comercial sendo a única exigência que o produto modificado também seja open-source. A placa Arduino foi originalmente desenvolvida pelos pesquisadores Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis no Instituto de Desing de Interação de Ivrea em meados dos anos 2000. O Arduino se baseia no projeto Processing, linguagem destinada ao desenvolvimento de artes visuais bem como interfaces gráficas. A primeira placa surgiu em 2005 tendo em vista a necessidade de ajudar estudantes de design, que não possuíam conhecimentos em eletrônica e programação de microcontroladores, a desenvolver protótipos de trabalho. Desde então o uso destas placas cresceu substancialmente se tornando extremamente populares no ambiente de engenharia, design, educacional, etc. É possível encontrar na internet inúmeros projetos e fóruns envolvendo a extensa família de plataformas Arduino.

2. Aplicações de Arduino

O Arduino pode enviar ou receber informações a partir de qualquer sistema eletrônico mesmo não tendo conexão de rede, ele pode ser combinado com outras placas formando o que chamamos de shields, que fazem papel semelhante, devido a essas facilidades ele possui diversas aplicações, para diversas áreas e objetivos.

Sistemas de automação são os mais comuns quando o assunto é Arduino, devido aos shields como bluetooth e ethernet**,** as pessoas conseguem automatizar ações rotineiras para que sejam feitas apenas com o celular ou algum tipo de hardware que pode se conectar, como por exemplo acender e apagar uma lâmpada, abrir um portão, ligar e controlar a velocidade de um ventilador, sensores de presença, de incêndio, de luminosidade e etc.

Outros projetos que estão cada vez mais sendo feitos com Arduinos são os que envolvem robótica, pelo fato de a placa ser barata e de fácil uso. As pessoas que não possuem muito conhecimento na área acabam por começar suas pesquisas com o Arduino pela facilidade.

O Arduino hoje em dia é a porta de entrada para muitas pessoas na área de programação e montagem de projeto eletroeletrônicos, devido ao seu custo baixo e facilidade de manuseio.

Por isso é de grande valia aprender sobre esse tipo de hardware para o futuro na área acadêmica e de trabalho. Para facilitar ainda mais a utilização foi criada uma

família de Arduinos diferentes para diferentes tipos de aplicações, onde cada um tem sua particularidade.

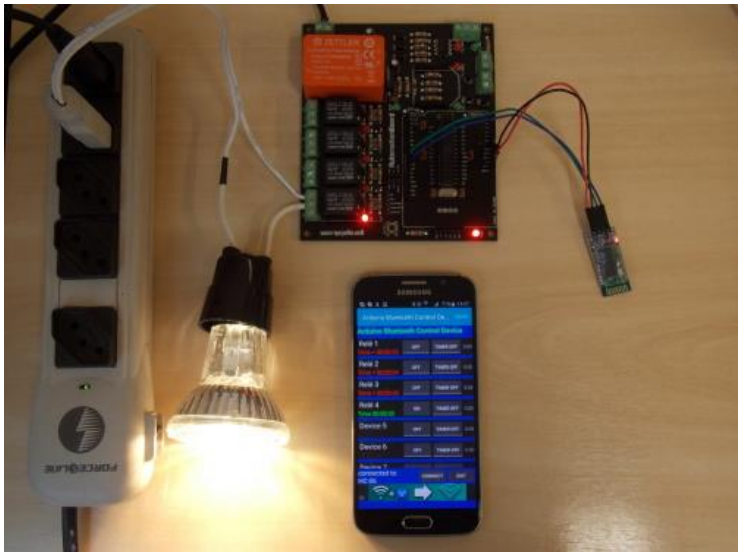

Figura 1: Automação simples de interruptor com bluetooth
Fonte: Site Labdegaragem

Figura 2: Aranha e braço robóticos com uso do Arduino
Fonte: Site Usinainfo

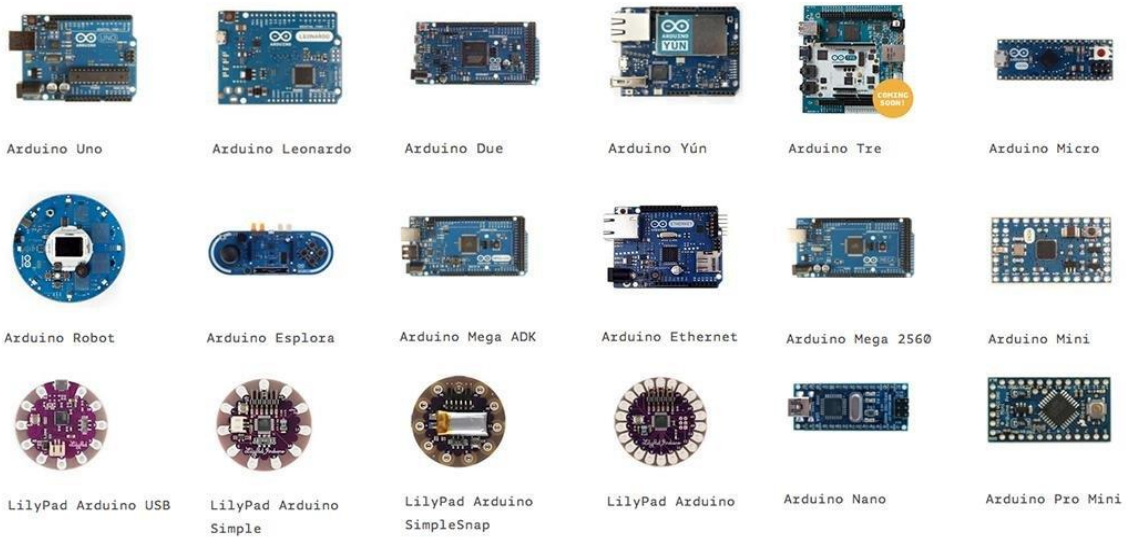

Figura 3: Tipos de Arduino
Fonte: Site Nodebots

3. Hardware

Vamos agora conhecer um pouco do nosso hardware, que nesse minicurso será o Arduino Uno. Para falarmos onde fica cada parte e para o que ela serve seguiremos a imagem abaixo.

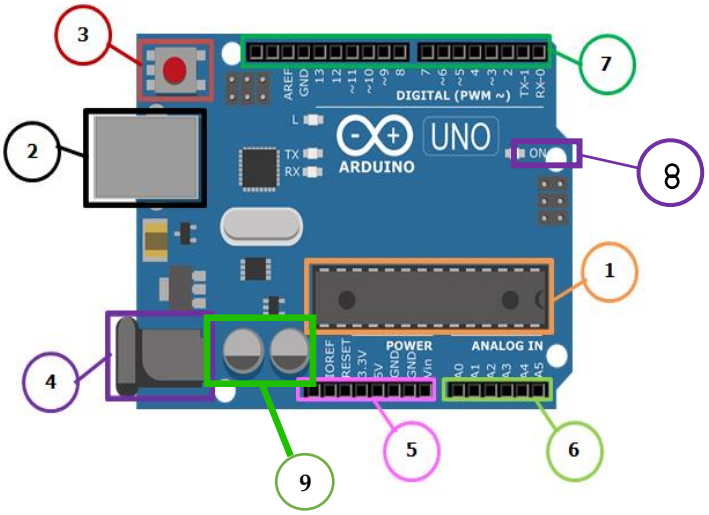

Figura 4: Placa Arduino Uno
Fonte: Autoria

3.1. Microcontrolador: Atmel ATMEGA328P

Um microcontrolador é um pequeno computador em um único circuito integrado, que contém processador, memória e periféricos de entrada e saída. Eles são feitos para aplicações de propósito específico, como aplicações embarcadas, em contraste com os microprocessadores usados em computadores pessoais, que são para aplicações de propósito geral. Esse microcontrolador funciona como "CPU" do Arduino e tem o desempenho de 20 mil instruções por segundo com uma frequência de 20MHz. Ele também protege a placa de curtos-circuitos, a maioria dos defeitos que podem aparecer com o tempo de uso podem ser sanados com a troca do mesmo.

3.2. Porta USB

É a principal porta do Arduino usada tanto para alimentação da placa quanto para programação e comunicação com a IDE. É uma entrada USB do tipo B2.

3.3. Botão Reset

Serve para reiniciar a placa durante a ocorrência de possíveis erros, ou quando se quer a execução de uma parte inicial do programa que está na estrutura "setup" ao invés do "loop".

3.4. Conector P4

Outra forma de alimentar a placa, utilizando uma fonte com conector do tipo P4 que forneça qualquer fonte de corrente continua entre 7V e 12V.

3.5. Pinos de alimentação e referência

IOREF: Este pino na placa Arduino fornece a referência de tensão com que o microcontrolador está operando. Um determinado circuito configurado corretamente pode ler a tensão do pino IOREF e selecionar a fonte de alimentação adequada ou habilitar tradutores de tensão nas saídas para trabalhar com o 5V ou 3.3V.

RESET: Entrada para criar um botão de reset externo, no momento que uma tensão é aplicada à essa porta, o Arduino é reiniciado de forma semelhante ao do botão de reset.

3.3 e 5V: Pinos com saída fixa de 3.3V e 5V, respectivamente.

GND: Também conhecido como ground, é a referência negativa do circuito, costuma ser adotado como 0V.

Vin: É outra forma para alimentação da placa, caso as portas USB e P4 não sejam viáveis no momento pode-se alimentar seu Arduino por esse pino. Assim como também é um pino de saída de energia, porém, não tem valor fixo, a mesma energia recebida pelo conector P4 ou cabo USB será emitida no pino Vin.

3.6. Pinos analógicos:

Estes pinos são de entradas analógicas e podem ler valores analógicos variando de 0 a 5V. Possuem resolução de 10 bits. Para medir os valores de entrada o Arduino utiliza um conversor interno analógico-digital (ADC), que transforma o valor analógico lido em um valor digital entre 0 e 1023 (pois 210 é 1024), de acordo com o valor de referência da placa.

Obs.: O Arduino não possui um conversor digital-analógico, mas ele possui algumas saídas que podem gerar pulsos, que se aproximam de saídas analógicas. Explicação sobre grandezas digitais e analógicas será feita em breve.

3.7. Pinos digitais:

Todos os pinos digitais podem ser configurados tanto como entradas quanto como saídas. Em geral, podem fornecer no máximo 40mA para outro circuito, e possuem 8 bits. São de fácil entendimento pois possuem apenas dois estados:

HIGH (ou ON) – representa o valor de 5V.
LOW (ou OFF) – representa o valor de 0V.

Dentre os 14 pinos digitais que o Arduino UNO possui, 6 deles (3, 5, 6, 9, 10 e 11) são PWM.

AREF: Configura a tensão de referência para a entrada analógica (o valor máximo do intervalo de entrada), se não tiver conectado ao Arduino, o mesmo está com referência de 0-5V para 0 -1023.

TX e *RX*: Portas de comunicação serial, usadas para comunicar a placa com algum módulo ou shield externo.

Pino 13: De todos os pinos digitais o pino 13 particularmente possui o seu próprio Led, que é muito útil para pequenos projetos ou testes quando não se quer usar uma protoboard por exemplo.

3.8. LED ON

O Arduino possui um LED que fica acesso quando o mesmo se encontra alimentado com corrente elétrica.

3.9. Capacitores

Quando compilamos algum código na placa, esse por sua vez fica salvo na memória do Arduino, sendo executado toda vez que ele for ligado. Esses capacitores mantém o mínimo de energia necessária para que essas informações não sejam perdidas.

4. Grandezas analógicas e digitais

4.1. Relembrando: O que são Bits

O termo Bit vem de Binário e é a menor unidade de transmissão de dados usada na computação. Um bit pode possuir apenas um único valor, zero ou um. Por uma convenção de uso, grande parte dos códigos utilizados pelos computadores era em pacotes de oito bits, e por isso foi convencionado o nome de Byte, também conhecido como octeto.

Devido ao crescimento da quantidade de dados transportada e utilizada, surgiu-se a necessidade de se criar espaços maiores e para chama-los utilizamos os mesmos prefixos do SI da física, química e matemática, mesmo resultando em valores diferentes eles possuem valores próximos.

Nome	Abrev	Tamanho em Bytes	Tamanho no SI
quilo	K	2^{10} ="1024"	10^{3} ="1000"
mega	M	2^{20} ="1 048 576"	10^{6} ="1 000 000"
giga	G	2^{30} ="1 073 741 824"	10^{9} ="1 000 000 000"
tera	T	2^{40} ="1 099 511 627 776"	10^{12} ="1 000 000 000 000"
peta	P	2^{50} ="1 125 899 906 842 624"	10^{15} ="1 000 000 000 000 000"
exa	E	2^{60} ="1 152 921 504 606 846 976"	10^{18} ="1 000 000 000 000 000 000"
zetta	Z	2^{70} ="1 180 591 620 717 411 303 424"	10^{21} ="1 000 000 000 000 000 000 000"
yotta	Y	2^{80} ="1 208 925 819 614 629 174 706 176"	10^{24} ="1 000 000 000 000 000 000 000 000"

Figura 5: Tabela Bytes
Fonte: Tecmundo

4.2. Grandezas

Grandezas digitais, podemos dizer que são aquelas que não apresentam valores intermediários, possuem saltos numa faixa de valores bem definidos.

Grandezas analógicas, diferente das digitais podem assumir esses valores intermediários entre uma faixa de valores, uma boa comparação para os dois casos seriam os relógios digital e analógico respectivamente. Podemos comparar a grandeza digital como uma escada, que tem subidas em "saltos" na vertical, e a analógica como uma rampa que sobe suave e continuamente adotando valores que na "escada" se perdem por conta dos saltos.

Você já deve estar se perguntando, "e se eu fizer cada vez mais degraus até a escada ficar parecida com uma rampa?" Falaremos quando abordarmos as portas analógicas.

Como foi dito anteriormente os pinos digitais podem ser configurados tanto com entrada quanto saída, pelo fato de ler apenas HIGH ou LOW esses tipos de pinos são usado como entrada geralmente quando a intenção é fazer um interruptor externo, fazendo um caminho diferente dentro do programa dependendo do estado do mesmo, porém, nesse caso devido à alta impedância que lhe é atribuído, o pino fica sensível a ruídos elétricos do ambiente, podendo mudar seu estado indesejavelmente, para corrigir esse problema usamos o que chamamos de resistores pull-up e pull-down, ou definimos o pino como esse tipo de resistor no próprio programa, trataremos deles mais à frente.

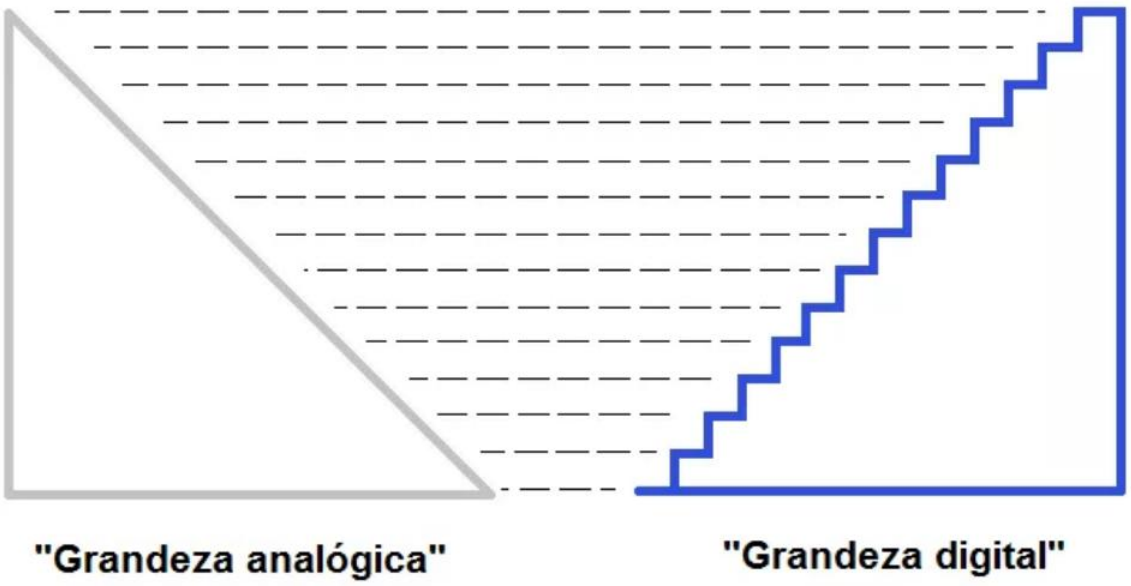

Figura 6: Representação grandeza digital e analógica.
Fonte: Site Portal Vida de Silício

Na vida real basicamente tudo é formado por valores **analógicos**, por conta disso o Arduino possui os pinos de entrada analógicas como já foi mostrado anteriormente, mas, no Arduino tudo é processado de forma digital e por conta disso se faz necessário a utilização de conversor, para converter sinal analógico para digital. Pensando na ideia da rampa, quanto mais degraus tivermos melhor vai ser a aproximação de um sinal digital para um analógico, o pino analógico possui 10 bits de resolução então são 1024 "degraus" que podemos dividir uma faixa de tensão de 0V a 5V. Aumentando ainda mais o número de degraus, isto é, o número de bits, o valor de tensão equivalente a um bit seria tão pequeno que consideraríamos como um sinal analógico.

4.3. PWM

Mas se o Arduino só funciona com grandezas digitais como podemos usar pinos de saídas analógicas? Bom nesse caso, existem pinos especiais que chamamos de PWM (Pulse Width Modulation), que possuem resolução de 8 bits e que são capazes de simular um sinal analógico por meio de sinais digitais. Esses pinos são representados pelas portas digitais com o (~) ao lado do número. Esse recurso consiste na geração de uma onda quadrada onde controla-se a porcentagem de tempo em que a tensão fica em HIGH, essa porcentagem é chamada de Duty Cycle, e dependendo do seu valor a tensão média pode variar, podendo assim tomar valores intermediários que os sinais digitais não conseguem. Então, por exemplo, se tivermos 0% de Duty Cycle teremos 0V marcando 0 bits e 100% de Duty Cycle teremos 5V marcando 255 bits. De forma mais geral é o tempo em que o sinal permanece em 5V dividido pelo tempo total de oscilação. A figura a seguir mostra alguns Duty Cycles.

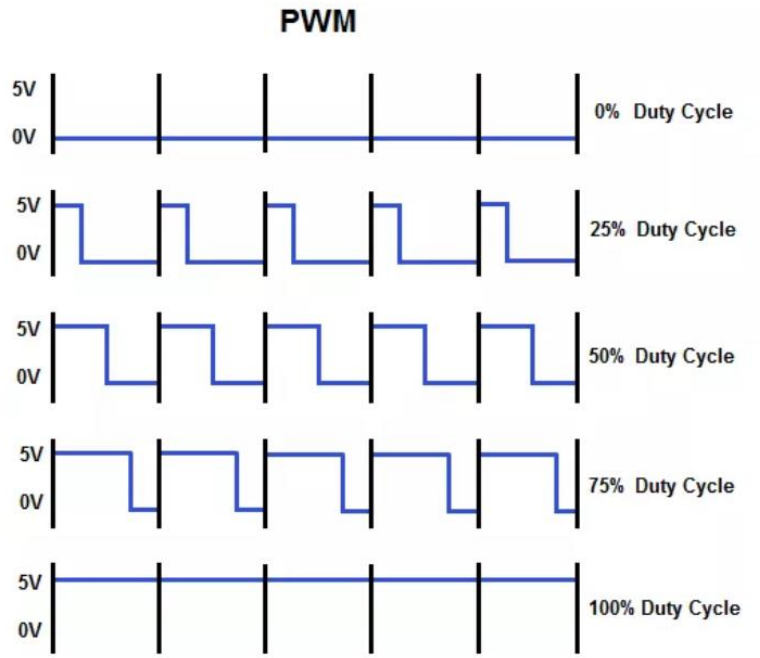

Figura 7: Duty Cycle.
Fonte: Site Portal Vida de Silício

5. Instalação da IDE

O software de código aberto do Arduino (IDE) torna simples escrever códigos, compilar e os enviar diretamente para a placa. Ele funciona nos sistemas operacionais Windows, Mac OS X e Linux. O ambiente é escrito em Java e baseado em Processing e outros softwares de código aberto. O software pode ser utilizado com qualquer placa Arduino.

A instalação do ambiente de desenvolvimento (IDE) do Arduino deve ser feita de forma diferente para cada sistema operacional.

Para tal, devemos acessar o site oficial do Arduino: https://www.arduino.cc

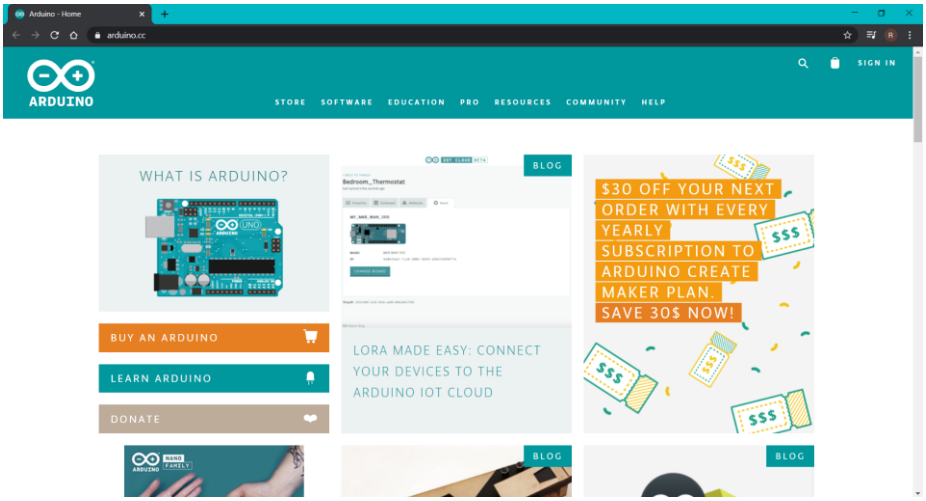

Figura 8: Pagina inicial.
Fonte: Site Arduino

Depois disso, devemos clicar na aba Software e depois em Downloads para ter acesso a página onde se encontra o software.

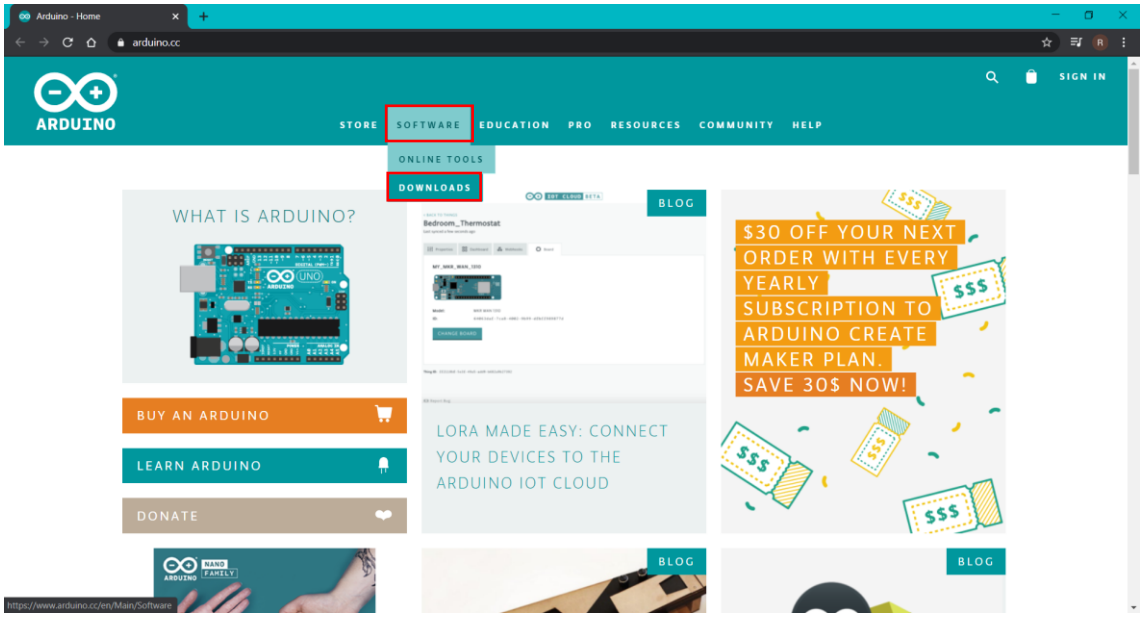

Figura 9: Página inicial.
Fonte: Site Arduino

Na parte "Download the Arduino IDE", clicar em "Windows Installer, for Windows XP and up" para fazer download do software para Windows. Também possível fazer o download para outros sistemas operacionais como Linux ou Mac OS X.

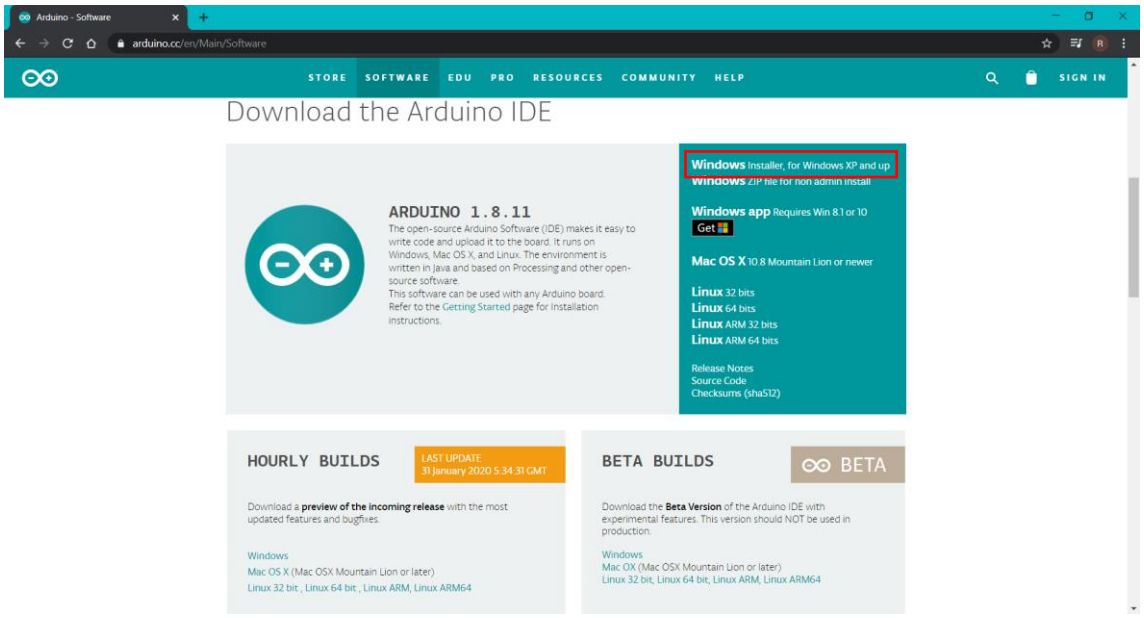

Figura 10: Pagina Downloads.
Fonte: Site Arduino

Em seguida, clicar em "Just Download" e o download será iniciado. Para o instalador ser executado em seguida devemos clicar em "I Agree".

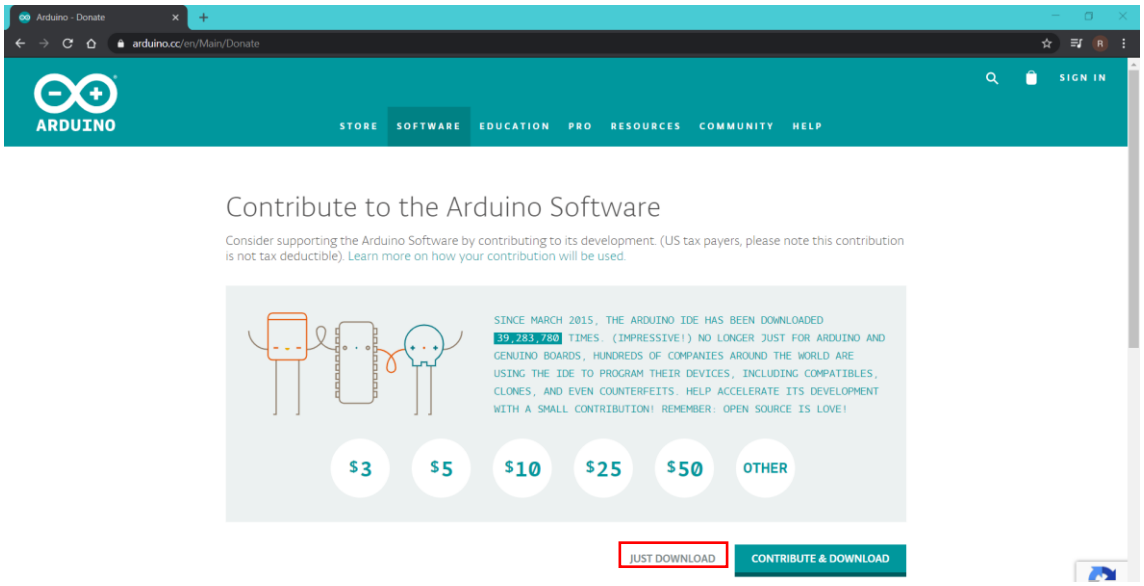

Figura 11: Pagina Donate.
Fonte: Site Arduino

Figura 12: Etapa da instalação.
Fonte: Arduino AG

Em seguida, clicar em “Next” e depois em “Install” para dar início a instalação do software em seu computador.

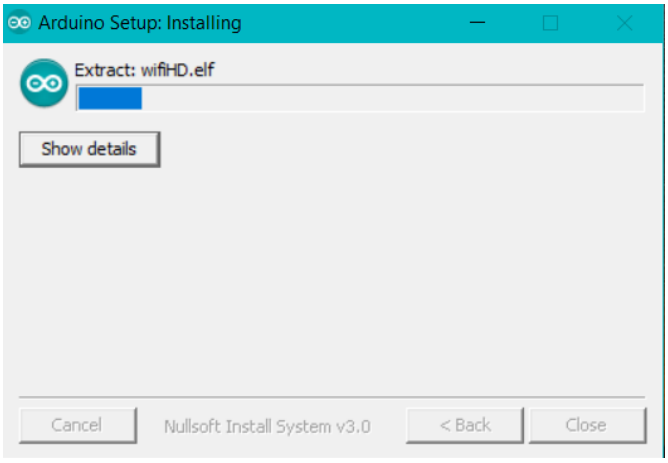

Figura 13: Etapa da instalação.
Fonte: Arduino AG

Por fim, o software está finalmente instalado e pronto para ser utilizado.

5.1. Verificação do Driver

O Arduino possui um dispositivo *Plug & Play* (PnP), ou seja, quando conectado a uma porta USB de um computador ele é automaticamente reconhecido e instalado. Logo após a instalação ele aparece como uma porta COM.

As portas seriais, também chamadas de portas de comunicação (COM), são consideradas uma das conexões externas mais básicas para um computador e permitem que cada dispositivo conectado a elas receba e envie dados simultaneamente.

Conectando o cabo USB AB no Arduino e depois no computador, seu computador reconhecerá o Arduino automaticamente e uma nova porta COM será criada para o sistema operacional Windows.

Para configurar essa porta COM siga os passos como mostrados abaixo.

Clique em Gerenciamento do Computador > Gerenciador de Dispositivos > Portas (COM e LTP) > Arduino UNO (COMX).

Pelo que for encontrado, podemos confirmar se o Arduino foi realmente reconhecido pelo computador e qual o número da porta COM associada a ele.

*Obs.: Nesse caso a porta genérica COMX foi encontrada como COM3, mas ela pode variar o valor de "X" de acordo com a configuração de cada computador.

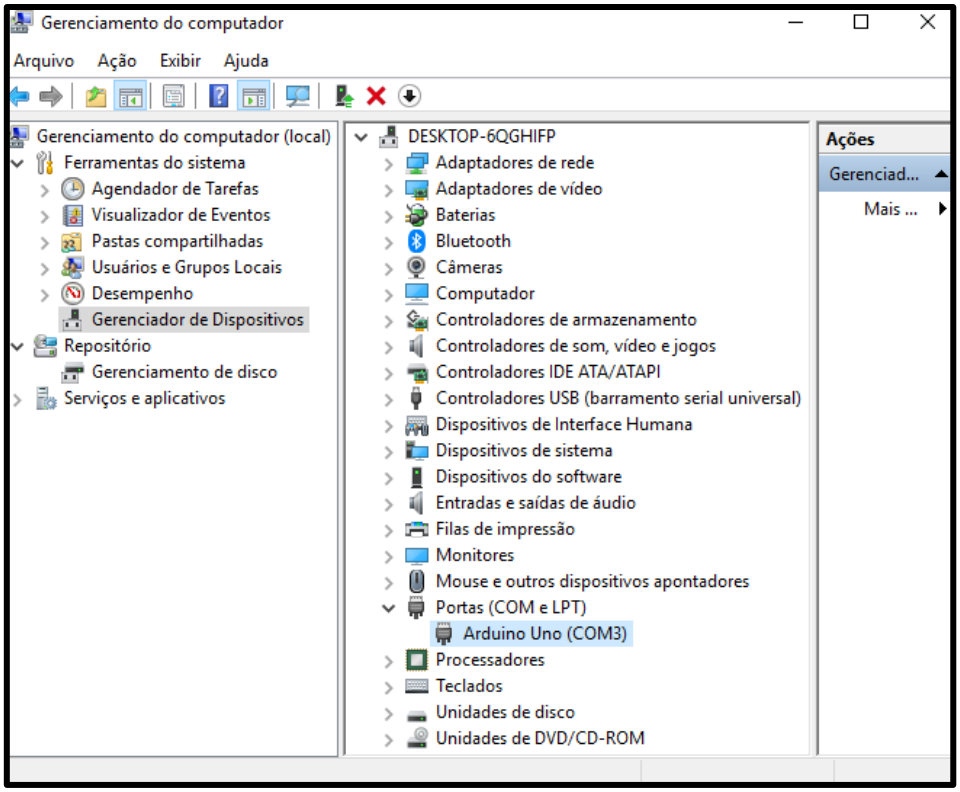

Figura 14: Gerenciador do computador.
Fonte: Autoria

De volta para o software do Arduino, devemos indicar para o IDE em qual porta o Arduino está conectado para garantir que a comunicação do microcontrolador com o computador seja executada de forma adequada.

Ferramentas > Porta > COMX (Porta do Arduino)

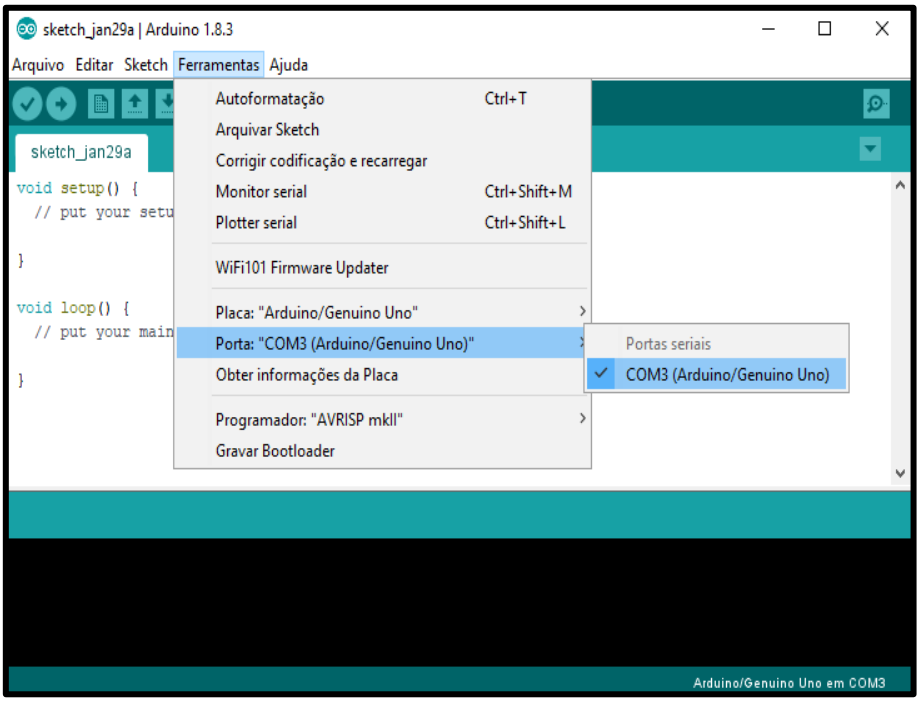

Figura 15: IDE do Arduino
Fonte: Autoria

Para que seus códigos sejam transferidos corretamente, devemos selecionar o modelo da placa Arduino que será utilizada na opção Placa do menu.

Clique em Ferramentas > Placa > Arduino/Genuino Uno (Modelo da Placa).

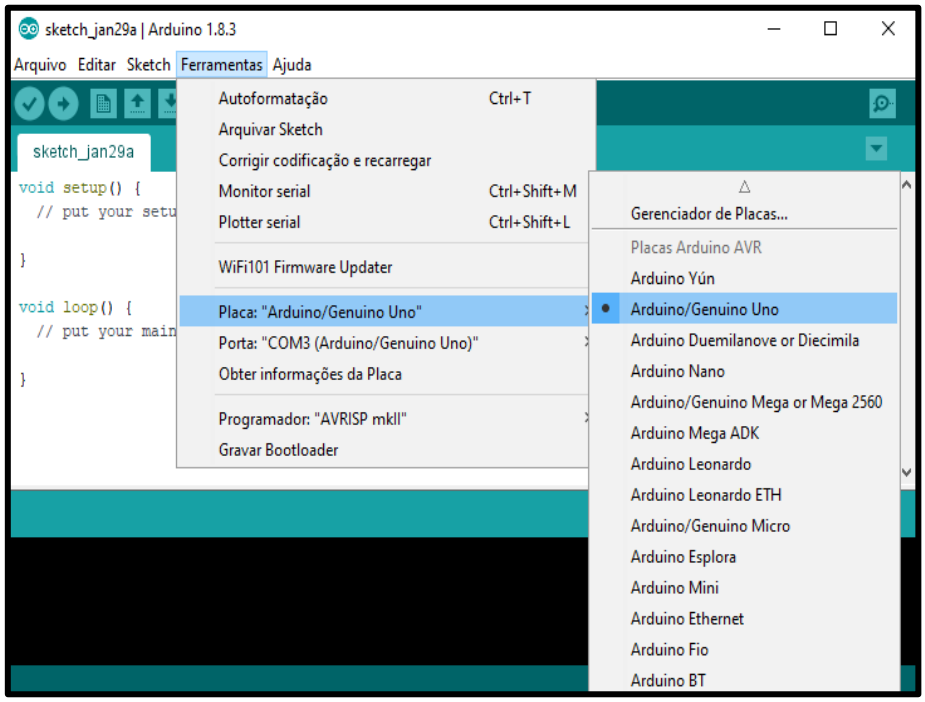

Figura 16: IDE do Arduino
Fonte: Autoria

5.2. Arduino IDE

O software Arduino IDE (Integrated Development Enviroment) é um ambiente que permite a criação de sketches para a placa Arduino, assim como compilar e fazer upload do código para a placa. A linguagem de programação é modelada a partir da linguagem Processing.

O ciclo de programação do Arduino é o seguinte:

- Conectar a placa a uma porta USB do computador;
- Desenvolver um sketch com comandos para a placa;
- Upload do sketch para a placa utilizando a comunicação USB;

Aguardar a reinicialização da placa. Após a reinicialização, o sketch passa a ser executado pela placa.

É importante ressaltar que a partir do momento que o sketch é gravado na placa Arduino, não há necessidade de conexão com o computador. Dessa forma, é possível utilizar o Arduino apenas alimentando a placa através da porta P4.

Quando se abre o programa Arduino IDE, uma interface semelhante à figura abaixo é exibida:

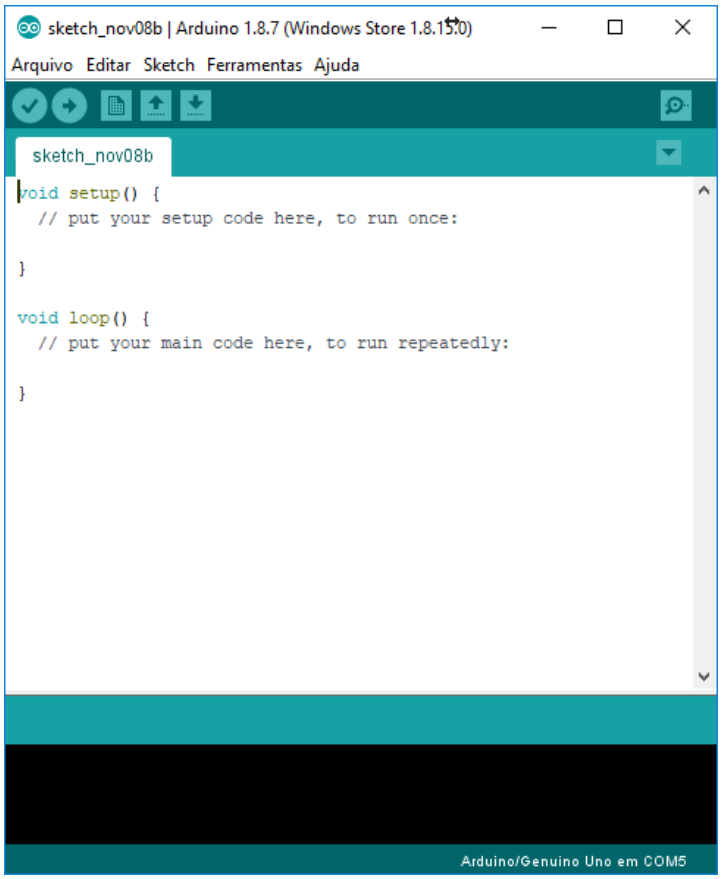

Figura 17: IDE do Arduino
Fonte: Autoria

A IDE pode ser dividida em três partes: Toolbar, Sketch Window e Janela de Mensagens.

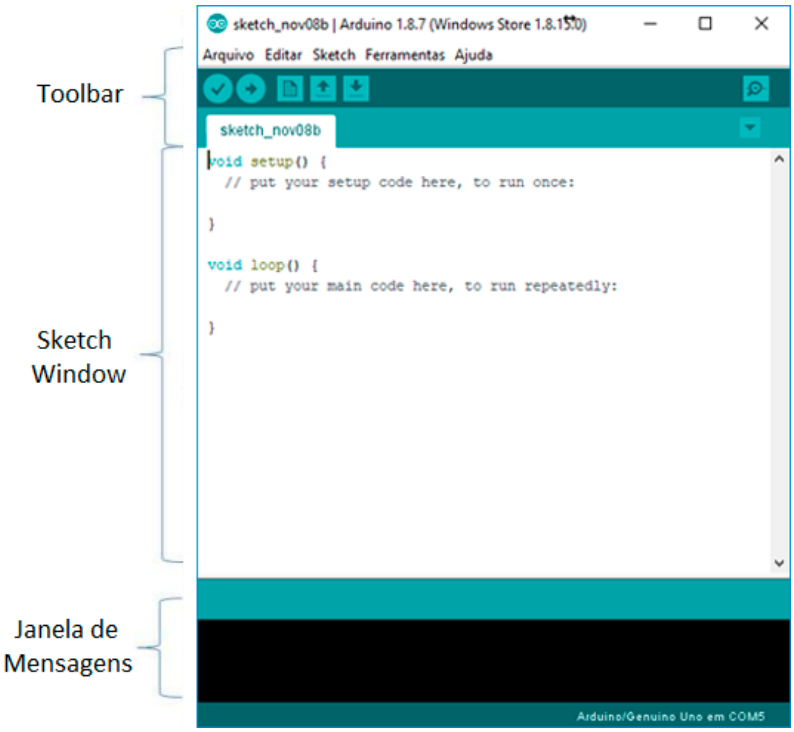

Figura 18: IDE do Arduino
Fonte: Autoria

A Sketch Window é onde os códigos são escritos. A Janela de Mensagens serve para informar mensagens de sucesso ou erros durante a verificação ou a compilação do código.

A Toolbar é composta por três elementos:

- Um menu que contém opções de arquivo (novo, salvar, abrir, sketches exemplos do Arduino, etc), editar, sketch, ferramentas e ajuda;
- Botões que servem de atalho para as funções dos menus acima que são mais utilizadas pelos usuários: **Verificar**, **Carregar**, **Novo**, **Abrir**, **Salvar** e **Monitor Serial**;
- Guias que permitem transitar entre os diferentes sketches presentes em um projeto.

Os botões da IDE e sua função são identificados abaixo:

- Verificar (Verify): Verifica se existe erro no código digitado antes de enviar para a placa;
- Carregar (Upload): Compila o código e grava na placa Arduino;
- Novo (New): Abre uma janela com um sketch em branco;
- Abrir (Open): Abre um sketch;
- Salvar (Save): Salva o sketch presente naquela janela;
- Monitor Serial (Serial Monitor): Abre o monitor serial.

As demais funções podem ser consultadas no menu Ajuda do Arduino.

6. Componentes Eletrônicos

Os componentes eletrônicos são a estrutura de um circuito eletrônico, isto é, são os componentes que fazem parte de qualquer circuito eléctrico ou eletrônico (desde os mais simples aos mais complexos) e que estão interligados entre si. Pode também ser definido como componente eletrônico todo dispositivo eléctrico que transmite a corrente eléctrica através ou de um condutor ou semicondutor.

6.1. Protoboard

É uma placa que permite a construção do circuito eletrônico. É um painel de conexão, com linhas de furos que permite conectar fios e componentes eletrônicos. A vantagem em usá-las é pela não necessidade de soldagem dos componentes eletrônicos para construir o circuito eletrônico.

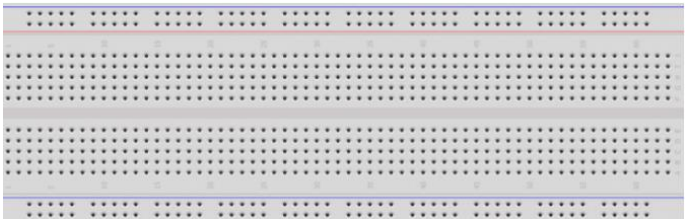

Figura 19: Protoboard
Fonte: Autoria

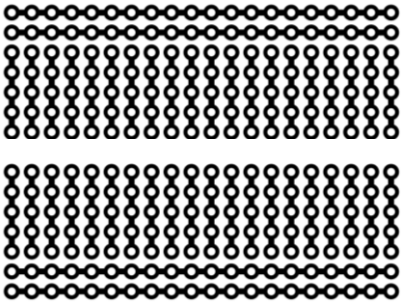

Figura 20: Conexões entre os furos
Fonte: Wikimedia Commons

6.2. Capacitores

Armazenam e liberam energia elétrica no circuito. Quando a tensão do circuito é maior do que o armazenado, a corrente flui carregando o capacitor. Caso contrário, a corrente flui no sentido de descarga. São comumente usados em fontes de energia que suavizam a saída de uma onda retificada completa ou meia onda. Por passarem sinais de Corrente Alternada (AC) mas bloquearem Corrente Contínua (DC), capacitores são frequentemente usados para separar circuitos Corrente alternada de corrente continua. Este método é conhecido como acoplamento AC. Capacitores também são usados na correção de fator de potência.

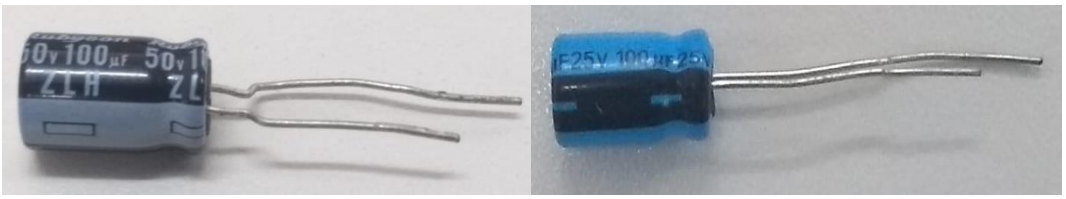

Figura 21: Capacitores
Fonte: Autoria

Figura 22: Simbologia do capacitor
Fonte: Wikimedia Commons

6.3. Diodo

Garante que a corrente flua apenas em uma direção. Útil quando há alta carga de corrente/tensão no circuito. Este componente é polarizado, o que significa que a direção em que ele é posicionado importa, porque colocado em um sentido ele permite a passagem de corrente e, no outro, ele bloqueia. O catodo, que normalmente marcado com uma faixa em um dos lados do componente, é tipicamente conectado ao ponto de menor energia, ou à terra.

Figura 23: Diodos
Fonte: Autoria

Figura 24: Simbologia do diodo
Fonte: Wikimedia Commons

6.4. Diodos Emissores de Luz (LEDs)

É um tipo de diodo que fica iluminado quando eletricidade passa através dele. Seu funcionamento se dá de forma semelhante ao diodo. O anodo, que é tipicamente conectado à maior tensão, tem normalmente a perna mais longa. Em geral, os LEDs operam com nível de tensão de 1,6 a 3,3 V, sendo compatíveis com os circuitos de estado sólido.

É interessante notar que a tensão é dependente do comprimento da onda emitida. Assim, os LEDs infravermelhos geralmente funcionam com menos de 1,5V, os

vermelhos com 1,7V, os amarelos com 1,7V ou 2.0V, os verdes entre 2.0V e 3.0V, enquanto os LEDs azuis, violeta e ultravioleta geralmente precisam de mais de 3V. A potência necessária está na faixa típica de 10 a 150 mW, com um tempo de vida útil de 100.000 ou mais horas.

Figura 25: LEDs coloridos e LED RGB (na extremidade direita)
Fonte: Autoria

Figura 26: Simbologia do LED
Fonte: Wikimedia Commons

6.5. Ponte H

É um circuito de eletrônica de potência do tipo chopper de classe E. Um chopper de classe E converte uma fonte fixa de corrente contínua fixa em uma tensão de corrente contínua variável abrindo e fechando diversas vezes e, portanto, pode determinar o sentido da corrente, a polaridade da tensão e a tensão em um dado sistema ou componente.

Seu funcionamento dá-se pelo chaveamento de componentes eletrônicos usualmente utilizando do método de PWM para determinar além da polaridade, o módulo da tensão em um dado ponto de um circuito. Tem como principal função o controle de velocidade e sentido de motores DC escovados, podendo também ser usado para controle da saída de um gerador DC ou como inversor monofásico.

O circuito de ponte H é usado para determinar um sentido de corrente e valor de tensão no controle de um motor DC, o diagrama abaixo pode ser usado para ilustrar de modo genérico o funcionamento de tal. Acionando-se em conjunto, as chaves *S1* e *S4*, o terminal direito do motor fica com uma tensão mais positiva que o esquerdo, fazendo a corrente fluir da direita para a esquerda. Acionando-se em conjunto as chaves *S3* e *S2*, o terminal esquerdo do motor fica com uma tensão mais positiva que o direito, fazendo a corrente fluir da esquerda para a direita. Deste modo, o motor adquire sentido de giro que denotaremos por *Sentido 2*, que é inverso ao *Sentido 1*.

Ao acionar em conjunto as chaves *S1* e *S3* ou *S2* e *S4* provocamos um curto nos terminais do motor. Isso é necessário quando deseja-se frear um motor já em movimento ou aumentar a dificuldade de giro do eixo por um fator externo. Tal efeito é alcançado pois a máquina DC passa a se comportar com um gerador quando tem seu eixo em movimento, tanto no caso da rotação, quanto no caso do giro do eixo por fator externo. Ao criar-se um curto circuito entre os terminais da máquina nesse estado, o torque necessário para manter ou colocar o motor em giro cresce, visto a necessidade de corrente exigida da máquina para seu movimento, o que causa o efeito chamado freio motor.

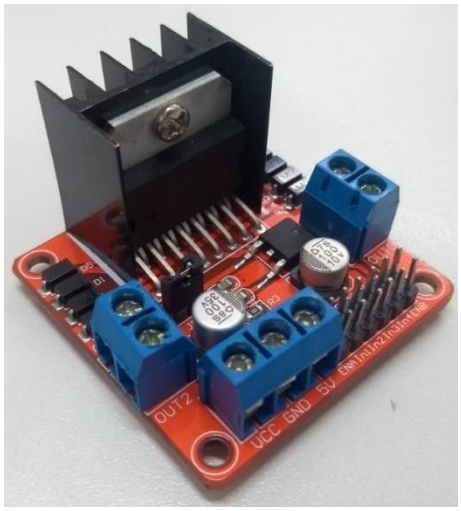

Figura 27: Ponte H dupla L298N
Fonte: Autoria

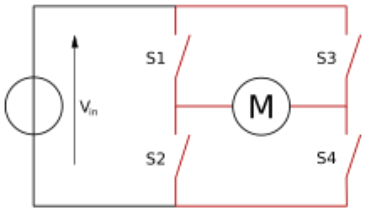

Figura 28: Diagrama de um circuito ponte H
Fonte: BUTTAY (2006)

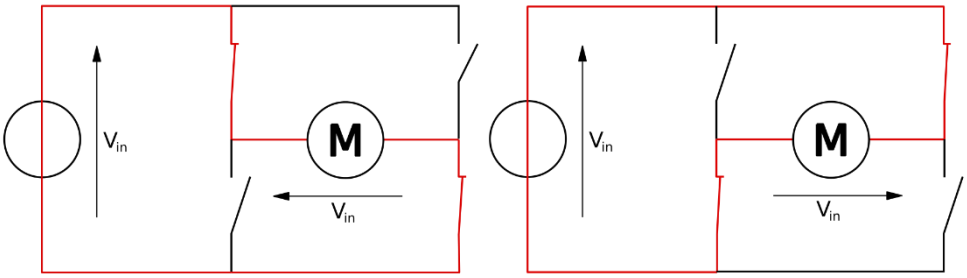

Figura 29: Dois estados básicos de uma ponte H
Fonte: BUTTAY (2006)

6.6. Display de Cristal Líquido (LCD)

Um tipo de display alfanumérico ou gráfico baseado em cristais líquidos. LCDs são encontrados em vários tamanhos, formatos e estilos. Cada pixel de um LCD tipicamente consiste de uma camada de moléculas alinhadas entre dois eletrodos transparentes e dois filtros polarizadores. A superfície dos eletrodos que estão em contato com o material de cristal líquido são tratados de forma a alinhar as moléculas de cristal líquido em uma determinada direção. Este tratamento consiste tipicamente em uma fina camada de polímero que é esfregada unidirecionalmente. A direção do alinhamento do cristal líquido é então definida pela direção da fricção. Os eletrodos são feitos de um condutor transparente chamado Indium Tin Oxide (ITO).

Figura 30: Display de cristal líquido (LCD)
Fonte: Autoria

6.7. Resistor

Resiste ao fluxo de corrente elétrica no circuito, ora possui a finalidade de transformar energia elétrica em energia térmica por meio do efeito joule, ora possui a finalidade de limitar a corrente elétrica em um circuito.

Causam uma queda de tensão num circuito elétrico, porém jamais causam quedas de corrente elétrica, apesar de limitar a mesma. Isso significa que a corrente elétrica que entra em um terminal do resistor será exatamente a mesma que sai pelo outro terminal.

Tem valor medido em ohms, representado por Ω. As listras coloridas nas laterais do resistor indicam seu valor (ver tabela de código de cores dos resistores).

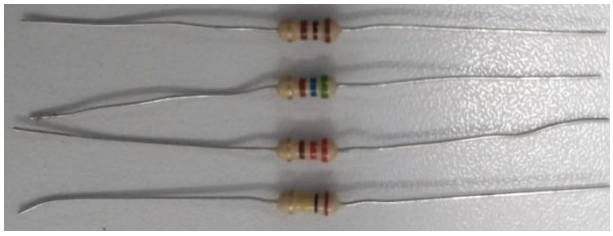

Figura 31: Resistores diversos
Fonte: Autoria

Figura 32: Simbologia do resistor
Fonte: Wikimedia Commons

6.8. Potenciômetro

É um resistor de três terminais onde a conexão central é deslizante e manipulável. Se todos os três terminais são usados, ele atua como um divisor de tensão. Normalmente são usados em controle de volumes de aparelhos de som, controle de posicionamento em controles de vídeo games, controle de brilho e contraste em telas LCD, para controlar os movimentos do braço de um servomotor e a velocidade de um motor DC.

Figura 33: Potenciômetros de 10 kΩ
Fonte: Autoria

Figura 34: Simbologia do potenciômetro
Fonte: Wikimedia Commons

6.9. Resistor dependente de luz

Também conhecido pela sigla inglesa LDR (Light Dependent Resistor). Sua resistência muda de acordo com a quantidade de luz que incide na sua superfície, quando a luz que incide sobre o semicondutor tem uma frequência suficiente, os

fótons que incidem sobre o semicondutor libertam elétrons para a banda condutora que irão melhorar a sua condutividade e assim diminuir a resistência.

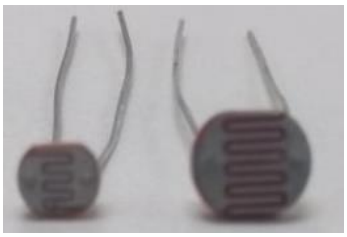

Figura 35: Resistores dependentes de luz (LDR)
Fonte: Autoria

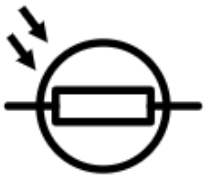

Figura 36: Simbologia do LDR
Fonte: Wikimedia Commons

6.10. Servo motor

É uma máquina eletromecânica que apresenta movimento proporcional a um comando. Recebe um sinal de controle, que verifica a posição atual para controlar o seu movimento, indo para a posição desejada, com velocidade monitorada externamente, sob feedback de um dispositivo denominado taco ou encoder.

Em contraste com os motores contínuos que giram indefinidamente, o eixo dos servo motores possui a liberdade de apenas cerca de 180º graus, mas são precisos quanto à sua posição. Para isso possuem três componentes básicos:

- Sistema atuador – É constituído por um motor elétrico, de corrente alternada ou contínua. Também está presente um conjunto de engrenagens que forma uma caixa de redução, com uma relação que ajuda a amplificar o torque. Tamanho, torque, velocidade do motor, material das engrenagens, liberdade de giro do eixo e consumo são características-chave para especificação de servo motores.
- Sensor – É um potenciômetro solidário ao eixo do servo. O valor de sua resistência elétrica indica a posição angular em que se encontra o eixo e sua qualidade vai interferir na precisão, estabilidade e vida útil do servo motor.
- Circuito de controle – O circuito de controle é formado por componentes eletrônicos discretos ou circuitos integrados e geralmente é composto por um oscilador e um controlador PID (controle proporcional integrativo e derivativo) que recebe um sinal do sensor (posição do eixo) e o sinal de controle e aciona o motor no sentido necessário para posicionar o eixo na posição desejada.

Figura 37: Servomotor SG90
Fonte: Autoria

6.11. Piezo

Geram tensão elétrica por resposta a uma pressão mecânica. O efeito piezoelétrico é entendido como a interação eletromecânica linear entre a força mecânica e o estado elétrico (forças de Coulomb) em materiais cristalinos (cerâmicos, polímeros). É um processo reversível em que os materiais exibem o efeito piezoelétrico direto e o efeito piezoelétrico reverso, onde, o direto, se dá pela geração interna de carga elétrica resultante de uma força mecânica aplicada e, o reverso, pela a geração interna de uma tensão mecânica resultante de um campo elétrico aplicado.

O efeito piezoelétrico não existe em materiais que apresentam simetria central, e desta forma, podem ser polarizados, ou seja, a piezoeletricidade pode ser explicada pela assimetria de polarização iônica. Nestes casos, a polarização elétrica induzida é atribuída à distribuição eletrônica, que é alterada pela ação externa.

Figura 38: Piezo buzzer
Fonte: Autoria


Figura 39: Piezo transducer
Fonte: Autoria

6.12. Motor DC

Converte energia elétrica em energia mecânica quando eletricidade é aplicada às suas extremidades. Bobinas de fio dentro do motor tornam-se magnetizadas quando a corrente flui através delas. Os campos magnéticos atraem e repelem ímãs, causando o giro do eixo. Se a direção da eletricidade for invertida, o motor girará na direção oposta.

Figura 40: Motores DC
Fonte: Autoria

Figura 41: Simbologia do motor DC
Fonte: Wikimedia Commons

6.13. Botões

É uma chave e, quando pressionado, abre ou fecha os contatos do dispositivo, abrindo ou fechando o circuito onde ele está conectado. Esse tipo de chave pode ser Normalmente Fechada / NF (Normally Closed / NC), quando a conexão entre os contatos está estabelecida por padrão e é interrompida ao pressionamento do botão; ou então Normalmente Aberta / NA (Normally Open / NO), caso no qual a conexão é fechada (estabelecida) ao pressionarmos o botão.

Figura 42: Chave tátil (push button)
Fonte: Autoria

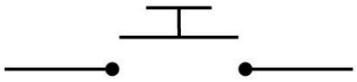

Figura 43: Simbologia da chave normalmente aberta
Fonte: Site Squids Arduino

6.14. Sensor de temperatura – LM35

É um sensor de precisão que apresenta uma saída de tensão linear proporcional à temperatura em que ele se encontrar no momento, tendo em sua saída um sinal de 10mV para cada Grau Célsius de temperatura. Esse sensor não necessita de qualquer calibração externa para fornecer com exatidão, valores de temperatura com variações de $^1/_4$ºC ou até mesmo $^3/_4$ºC dentro da faixa de temperatura entre –55ºC e 150ºC.

Ele pode ser usado de duas formas, com alimentação simples ou simétrica, em cada uma dessas duas formas de alimentação, a temperatura máxima e mínima medida com exatidão, é diferente.

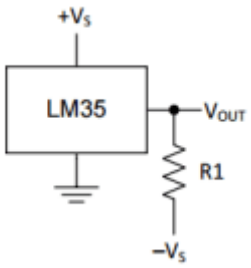

Figura 44: Simbologia do LM35
Fonte: Site TOT Engenharia

Onde $R_1 = -V_s/50$ µA; $V_{out} = 1500\ mV$ a 150°C; $V_{out} = 250\ mV$ a 25°C e $V_{out} = -550\ mV$ a −55°C.

6.15. Transistor

O termo provém do inglês transfer resistor (resistência de transferência). São utilizados principalmente como amplificadores e interruptores de sinais elétricos, além de retificadores elétricos, comutadores de circuitos e reguladores de corrente.

Entende-se por "amplificar" o procedimento de tornar um sinal elétrico mais forte. Um sinal elétrico de baixa intensidade, como os sinais gerados por um microfone, é injetado num circuito eletrônico, cuja função principal é transformar este sinal fraco gerado pelo microfone em sinais elétricos com as mesmas características, mas com potência suficiente para excitar os alto-falantes.

O transistor é montado justapondo-se uma camada P, uma N e outra P (unindo-se dois diodos), criando-se um transistor do tipo PNP. O transistor do tipo NPN é

obtido de modo similar. Para formar a camada P, um material de silício, é cortado em pastilhas finas que seguirão para um processo chamado de dopagem, o qual consiste em adicionar impurezas com 3 elétrons na última camada, faltando assim um elétron na ligação covalente, formando os buracos e caracterizando a pastilha como pastilha P. Quando adicionamos uma impureza com 5 elétrons na última camada, vai sobrar um elétron na ligação, assim se forma a pastilha N, que tem esse nome por ter maior número de elétrons livres.

Existem diferentes tipos de transístores, o bipolar (BJT) e o unipolar (FET). Este último tem diferentes variantes: JFET, mosfet, Nmosfet e Pmosfet. O BJT é o mais utilizado, tendo sido aquele que foi primeiro fabricado. É constituído por duas junções PN ligadas entre si, podendo obter-se as configurações NPN e PNP. Destas junções resultam três zonas de condução, às quais foram dados os nomes Coletor (C), Base (B) e Emissor (E). A Base é a região intermédia, o Coletor e o Emissor ficam nos extremos; o Emissor é diferente do Coletor, visto que possui mais impurezas. O transístor bipolar fica, portanto, com duas junções designadas por Coletor-Base e Base-Emissor.

O transistor apresenta exteriormente três terminais que estão ligados internamente a cada uma das três zonas de condução. O *datasheet* do transistor indica quais são os terminais de cada um, ou seja, é sempre necessário consultar o manual técnico ou o site do fabricante. O princípio de funcionamento BJT é o seguinte: a Base B, com corrente reduzida I_B (µA ou mA), permite controlar a corrente I_C (muito mais elevada, mA ou A) da carga ligada no coletor C ou a potência fornecida à carga ligada ao coletor; pelo emissor, faz-se o escoamento das correntes anteriores que somadas originam a corrente de emissor $I_E = I_B + I_C$. Polariza-se diretamente a junção Base-Emissor e inversamente a junção Coletor-Base, para que o transistor funcione na zona ativa, como amplificador de corrente. O ganho de corrente b é calculado pela expressão $b = I_C/I_B$, e pode variar entre 10 e 450.

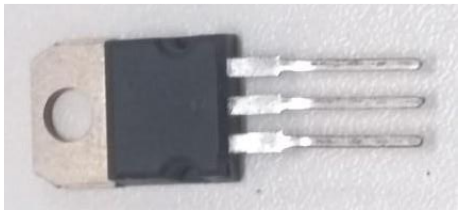

Figura 45: Transistor NPN TIP120
Fonte: Autoria

Figura 46: Simbologia do transistor NPN
Fonte: Wikimedia Commons

Figura 47: Simbologia do transistor PNP
Fonte: Wikimedia Commons

7. Variáveis

Cada variável tem função de armazenar seu respectivo tipo de dado, tornando possível o uso do dado posteriormente no programa. Para armazenar o dado é preciso usar o operador de atribuição, ou seja, o sinal de igual. Antes de serem usadas as variáveis devem ser declaradas, sem a necessidade de atribuir valor a elas.

Exemplo:

```
int var1;     // ambas as declarações estão corretas.
int var2 = 0; // foi atribuído ou armazenado o valor 0 em
                 var2.
```

Para saber qual tipo de variável usar, o programador deve saber o tamanho do conteúdo que quer armazenar. O local em que ela é declarada também influencia, porque isto muda a forma como ela é vista pelas funções. Isso é chamado de *escopo de variável*.

Quando é atribuído um valor inicial à variável, é o mesmo que dizer que a mesma foi *inicializada*. Vale lembrar que é uma boa prática de programação verificar o dado dentro da variável, antes dela ser acessada.

Quando a variável excede sua capacidade máxima, volta a sua capacidade mínima, ou máxima, dependendo da operação.

Exemplo:

```
byte x;    // 1byte = 8bits = 2^8 = 256 valores = 0 ~ 255


x = 0;
x = x - 1; // x agora contém 255


x = 255;
x = x + 1; // x agora contém 0
```

Recomenda-se dar nomes descritivos às variáveis, para que o código fique mais legível. É possível adotar qualquer nome para as variáveis, desde que não sejam usadas as palavras-chaves do Arduino.

7.1. Variáveis mais usadas

Tipo	Descrição	Exemplo

char	Armazena um valor de caractere, que é escrito com aspas simples: 'A'. Uma variável char é armazenada com um número, especificado na tabela ASCII, possibilitando fazer aritmética em caracteres. Este tipo de dado codifica números de -128 a 127.	`//As declarações a seguir são equivalentes:`
		`char letraA = 'A';`
		`char letraA = 65;`

byte	Possui 8 bits e armazena um valor de 0 a 255.	
double	No Arduino UNO armazena até 4 bytes, ou seja, sua implementação é exatamente igual ao float, sem ganho de precisão. Em placas com SAMD, *doubles* usam 8 bytes.	

float	Números de ponto flutuante frequentemente usados para aproximar valores analógicos e contínuos, porque possuem maior resolução que int. São armazenados em 4 bytes, variando de $3{,}4028235E+38$ ou pequenos quanto $-3{,}4028235E+38$. Possuem precisão de 6 a 7 dígitos e para fazer cálculos com floats, é preciso adicionar um ponto decimal, do contrário o número será tratado como um int.	`int x;`
		`int y;`
		`float z;`
		`x = 1;`
		`y = x / 2;`
		`z = (float)x/ 2.0;`
		`//y contém 0.0 e z contém 0.5.`

int	No Arduino Uno possui um intervalo de -2^{15} até $2^{15}-1$. Nas placas baseadas em Arduino Due o range é de -2^{31} e $2^{31}-1$.	`int cont = 0;`
long	Variáveis de tamanho estendido para armazenamento de números inteiros. Armazenam 4 bytes, variando de $-2{,}147{,}483{,}648$ a $2{,}147{,}483{,}647$. Ao se fazer cálculos com inteiros, pelo menos um dos números deve ser seguido por um L, forçando-o a ser um *long*.	`long speed = 186000L;`
string	É declarada como uma matriz de *char* e sua declaração pode ser feita de mais de uma forma. O caractere nulo deve ser usado para indicar o final da *string*. Para grandes quantidades de texto, como num projeto	`char Str3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o', '\0'};` `char myString[] = "Esta eh a primeira linha"`

	com display LCD, pode-se configurar uma matriz de *string*.	`" esta eh a segunda"` `" etecetera";`

unsigned int	São o mesmo que o tipo *int* no que esses também armazenam um valor de 2 bytes. Armazenam apenas valores positivos, garantindo um intervalo útil de 0 a 65,535 ($2^{16} - 1$).	`unsigned int x;`
		`x = 0;`
		`x = x - 1;`
		`//x contem 65535`
		`x = x + 1;`
		`//x contem 1`

unsigned long	Possuem tamanho estendido e armazenam 4 bytes e, da mesma maneira que os *unsigned int*, não armazenam números negativos, portanto, variam de 0 a 4.294.967.295 ($2^{32} - 1$).	`unsigned long time;`
		`time = millis();`
		time armazena o número de milissegundos a partir do início da execução do programa atual.

7.2. Variáveis Constantes

Há expressões predefinidas na linguagem Arduino. Essas são usadas para tornar os programas mais legíveis. As constantes da linguagem Arduino são classificadas nos grupos:

a) Constantes Booleanas: true e false, usadas para representar verdade e falsidade. false é definido apenas como 0 (zero). true é frequentemente definido como 1, o que está correto, porém qualquer inteiro que não seja zero é true, em um sentido booleano. Então -1, 2 e -200 são todos definidos como true.

b) Modos para Pinos Digitais: Pinos digitais pode ser usados como INPUT, INPUT_PULLUP ou OUTPUT. Mudar o modo de um pino com pinMode() muda o comportamento elétrico do pino.
Pinos configurados como INPUT são ditos estarem em um estado de *alta-impedância*. Isso significa que é preciso muito pouca corrente para mover o pino de entrada de um estado para outro e pode tornar os pinos úteis para tarefas, como, implementar um sensor de toque capacitivo, ler um LED como um fotodiodo ou ler um sensor analógico com um esquema como RCTime.
Existem resistores de empilhamento de 20K embutidos no chip Atmega que podem ser acessados a partir do software. Esses resistores de pullup integrados são acessados configurando o pinMode () como INPUT_PULLUP. Isso inverte o comportamento do modo INPUT, onde HIGH significa que o sensor está desligado e LOW significa que o sensor está ligado. Ao conectar um sensor a um pino configurado com INPUT_PULLUP, a outra extremidade deve estar conectada à terra. No caso de um interruptor simples, isto faz com que o pino leia HIGH quando o interruptor está aberto e LOW quando o interruptor é pressionado.
Pinos configurados como OUTPUT são ditos estarem em um estado de *baixa-impedância*. Isso significa que esses podem fornecer uma quantidade

substancial de corrente para outros circuitos. Isso faz com que sejam úteis para alimentar LEDs, pois estes usam menos de 40 mA. Cargas que exigem mais de 40 mA (ex. motores) irão requerer um transistor ou um outro circuito de interface. Pinos configurados como saídas podem ser danificados ou destruídos se forem conectados diretamente ao ground ou na tensão de alimentação.

c) Níveis lógicos: ao ler ou escrever o estado de um pino digital, o mesmo assume um dos dois valores possíveis: HIGH ou LOW. O significado de HIGH e LOW dependem se o pino está configurado como INPUT ou OUTPUT.
Quando um pino é configurado como INPUT com pinMode(), e lido com digitalRead(), o Arduino (ATmega) irá retornar HIGH se:
- uma tensão maior que 3.0V está presente no pino (em placas 5V);
- uma tensão maior que 2.0V está presente no pino (em placas 3.3V).

Quando um pino é configurado como OUTPUT com pinMode(), e colocado em estado HIGH com digitalWrite(), a tensão no pino é:
- 5V (em placas 5V);
- 3.3V (em placas 3.3V).

Quando um pino é configurado como INPUT com pinMode(), e lido com digitalRead(), o Arduino (ATmega) irá retornar LOW se:
- uma tensão menor que 1.5V está presente no pino (em placas 5V);
- uma tensão menor que 1.0V está presente no pino (em placas 3.3V).

Quando um pino é configurado como OUTPUT com pinMode(), e colocado em estado LOW com digitalWrite(), a tensão no pino é 0 volts (tanto em placas de 5V como 3.3V boards).

8. Funções próprias do arduino

8.1. Função Setup

A função setup é do tipo *void* e é classificada como uma das funções principais do Arduino. Nela é onde setamos os comandos que serão executados uma única vez, na inicialização do programa arquivo Arduino, algumas atribuições iniciais podem ser executadas nessa função. Sua estrutura segue o modelo:

```
//Declaração de variável
void setup() {
/*
Código do setup
Ou Inicializações
Setar entradas e saídas
```

```
 */
}
```

8.2. Função Loop

A função loop é também do tipo *void* e é classificada como uma das funções principais do Arduino. Em programação loop e laço são sinônimos e trazem a ideia de ciclo, portanto é nela que escrevemos os comandos que serão executados repetidamente pelo arquivo, seguindo o fluxo de comandos que possui interação com o circuito. Sua estrutura segue o modelo:

```
//Declaração de variável
void setup() {
// Comandos
}

void loop() {
/*
Código do loop
Fluxo de comandos
Interatividade com o circuito
 */
}
```

8.3. Modo dos pinos

Na função *setup*, a função dos pinos pode ser declarada com o comando *pinMode*. Nesse comando, deve ser fornecido como dados de entrada o endereço do pino (é um número inteiro) e o modo de fluxo de dados, que pode ser de saída (OUTPUT), ou entrada (INPUT) ou um modo especial que conecta o circuito a resistores internos do Arduino (INPUT_PULLUP).

```
int var_1 = 3;

void setup() {
pinMode(var_1 , OUTPUT);
}
```

8.4. Leitura e envio de sinal digital

Declarando um pino digital como `OUTPUT`, você poderá alternar o valor da tensão entre HIGH (5V) e LOW (0V). Para isso é utilizada a função `digitalWrite(pino,HIGH);` ou com o parâmetro `digitalWrite(pino,LOW);`

Declarando um pino digital como `INPUT`, você poderá ler o valor da tensão recebida no pino como HIGH ou LOW. Para isso é utilizada a função `digitalRead(pino);`

8.5. Leitura e envio de sinal analógico

A Pinos analógicos não necessitam de declaração, os comandos utilizados para ler e fornecer intensidade de tensão são:

`analogWrite(pino,valor);` → Pode enviar um sinal de tensão entre 0 e 5V, utilizando como parâmetro de entrada, valores entre 0 e 255, proporcionalmente.

Lembrando que as portas utilizadas no caso do Arduino Uno para esta função são as PWM, referenciadas com um (~).

`analogRead(pino);` → Pode ler valores de tensão entre 0 e 5V, mas retorna variável inteira em bits entre 0 e 1023, proporcionalmente. A taxa máxima de leitura é de 10 000 leituras por segundo, considerando que se leva um tempo de cerca de 0,0001 s para a leitura ser realizada.

8.6. Referência analógica de tensão

Para alterar a referência de limite superior, de 5 volts mencionada acima, existe um comando utilizado, que é o `analogReference(tipo).` Existem *tipos* padronizados e eles estão listados a seguir:

DEFAULT: A referência analógica é de 5 volts nas placas de 5V e 3.3 nas placas de 3.3 V.

INTERNAL: Uma referência embutida igual a 1.1 volts no ATmega168 ou ATmega328P e 2.56 volts no ATmega8 (Não disponível no Arduino Mega).

INTERNAL1V1: Uma referência de 1.1V embutida (Arduino Mega somente).

INTERNAL2V56: Uma referência de 2.56V embutida (Arduino Mega somente).

EXTERNAL: A voltagem aplicada ao pino AREF (0 à 5V somente) é usada como referência. Arduino SAMD Boards (Zero, etc.).

AR_DEFAULT: A referência padrão analógica de 3.3V.

AR_INTERNAL: Uma referência embutida de 2.23V.

AR_INTERNAL1V0: Uma referência embutida de 1.0V.

AR_INTERNAL1V65: Uma referência embutida de 1.65V.

AR_INTERNAL2V23: Uma referência embutida de 2.23V.

AR_EXTERNAL: A voltagem aplicada ao pino AREF pin é usada como referência. Arduino SAM Boards (Due).

8.7. Funções de comunicação serial

São funções utilizadas para viabilizar a comunicação entre o usuário e o monitor serial do Arduino, desta forma é possível detectar falhas e corrigi-las.

`Serial.begin(9600);` → Deve ser executada na função setup para inicializar a comunicação entre a janela de comandos e o monitor serial, o valor inteiro dado como parâmetro é a taxa de transmissão de bits e é setada de maneira padrão como 9600.

`Serial.end();` → Encerra a comunicação com o serial.

`Serial.Write("Mensagem");` → Essa função escreve no monitor serial uma mensagem definida pelo programador. Outros parâmetros de entrada da função são valores (inteiros) de bytes. Essa função também pode ser atribuída a uma variável, uma vez que retorna a quantidade de bytes enviados.

`Serial.println("Mensagem");` → Funciona de forma análoga ao Write, mas printa dados para o monitor serial com texto legível aos humanos (ASCII) e pula uma linha.

`Serial.available();` → Retorna o número de bytes disponíveis para a leitura, da porta serial.

8.8. Outras funções

`millis();` → Essa função retorna o tempo decorrido, em milissegundos, desde que a placa Arduino começou a ser trabalhada pelo programa em execução, o tempo máximo contado é de 50 dias aproximadamente, quando a contagem volta para zero (overflow).

`micros();` → Essa função retorna o tempo decorrido, em microssegundos, desde que a placa Arduino começou a ser trabalhada pelo programa em execução, o tempo máximo contado é de 70 minutos aproximadamente, quando a contagem volta para zero (overflow).

`delay(ms);` → Executa uma pausa no programa para a quantidade de tempo fornecida como parâmetro de entrada, em milissegundos.

`delayMicroseconds(µs);` → Executa uma pausa no programa para a quantidade de tempo fornecida como parâmetro de entrada, em microsegundos.

`tone(pino,frequência,duração);` → Gera uma onda quadrada em uma frequência especificada e a duração pode ser ou não fornecida, se não for, pode ser interrompida pelo comando `noTone();`

9. Referências

Acesso em 10/02/2020: http://blogmasterwalkershop.com.br/arduino/arduino-utilizando-o-sensor-ultrasonico-hcsr04-e-buzzer-5v/

Acesso em 10/02/2020: https://blog.usinainfo.com.br/alarme-com-arduino-e-sensor-ultrassonico-projeto-pascoa-segura/

Acesso em 10/02/2020: https://www.tinkercad.com/

Acesso em 10/02/2020: https://www.arduinoecia.com.br/2013/06/sons-no-arduino.html

Acesso em 10/02/2020: https://www.filipeflop.com/blog/sensor-ultrassonico-hc-sr04-ao-arduino/

Acesso em 10/02/2020: http://mundoprojetado.com.br/efeito-piezoeletrico-entenda-como-funciona-o-buzzer/

Acesso em 10/02/2020: http://fontesalternativaspth.blogspot.com/

Acesso em 10/02/2020: https://portal.vidadesilicio.com.br/medindo-temperatura-com-termistor-ntc/

Acesso em 10/02/2020: https://www.tecmundo.com.br/programacao/227-o-que-e-bit-.htm

Acesso em 10/02/2020: https://www.filipeflop.com/blog/controle-modulo-rele-arduino/

Acesso em 10/02/2020: https://portal.vidadesilicio.com.br/modulo-rele-com-arduino/

Acesso em 10/02/2020: https://www.infoescola.com/eletronica/rele/

Acesso em 10/02/2020: https://portal.vidadesilicio.com.br/sensor-de-presenca-hc-sr501/

Acesso em 10/02/2020: https://blog.usinainfo.com.br/automacao-residencial-arduino-ideias-para-deixar-sua-casa-igual-homem-de-ferro/

Acesso em 10/02/2020: http://fritzing.org/projects/iot-switch-onoff-220-240v-device-with-nodemcu-5v-r

Acesso em 10/02/2020: https://www.filipeflop.com/blog/controlando-um-lcd-16x2-com-arduino

Acesso em 10/02/2020: http://blog.eletrogate.com/guia-completo-do-display-lcd-arduino/

Acesso em 10/02/2020: https://www.addicore.com/2004-20x4-Character-LCD-with-I2C-backpack-p/157.htm

Acesso em 10/02/2020: www.orientlcd.com

Acesso em 10/02/2020: https://pt.wikipedia.org/wiki/Ponte_H

Acesso em 10/02/2020: http://tot.eng.br/sensor-de-temperatura-lm35-no-arduino/

MOTA, Allan Deangelle. Apostila Arduino Básico: Vol. 2. Serra – ES: Vida de Silício, 2015. 65p. Apostila