Programação de software básico

Aula 07: Controle de dispositivos com microcontroladores

Apresentação

Microcontroladores se encontram em todos os dispositivos e eletrodomésticos que usamos no dia a dia. Você provavelmente já deve ter se perguntado se o controle de um dispositivo de hardware com microcontroladores é algo de dificuldade elevada. Nesta aula, nós começaremos a ver que não, principalmente quando podemos usar um simulador que permita testar a integração software e hardware nesses dispositivos.

Por meio de exemplos de uma plataforma muito popular atualmente, o Arduino, veremos como funções em C se aplicam com facilidade à programação de software básico com microcontroladores. O entendimento no uso de funções em C, para acesso ao hardware do Arduino e dispositivos externos a ele, é importante também como experiência para o aprendizado de programação em outros microcontroladores com aplicações diversas na indústria.

Objetivos

- Discutir o básico do hardware de microcontroladores;
- Desenvolver programas em C para acesso às portas de microcontroladores em C;
- Criar programas em C para o controle de dispositivos acoplados a microcontroladores.

Controle de dispositivos com microcontroladores

Um microcontrolador pode ser visto como um computador completo em um único dispositivo eletrônico (chip), pois tem integrado a seu hardware vários sistemas básicos, como

- a unidade central de processamento (CPU);
- a circuito de clock;
- a memória de dados (RAM);
- A memória de programa armazenado (EEPROM ou Flash Eprom).

Ainda há a possibilidade de integrar vários outros.

Uma das características mais importantes de um microcontrolador são seus pinos de entrada e saída (E/S), as portas, que permitem que o microcontrolador se comunique com o mundo externo. Embora alguns microcontroladores tenham pinos de entrada e pinos de saída separados, a maioria dispõe de pinos de E/S compartilhados que podem ser programados como entrada ou saída.

Os pinos de E/S normalmente usam a interface lógica TTL básica: HIGH (lógica 1) é representada por +5 V; e LOW (lógica 0) é representada por 0 V.

A maioria dos microcontroladores pode manipular apenas uma pequena quantidade de corrente diretamente pelos pinos. A corrente típica é 20-25 mA. Isso é suficiente para acender um LED, mas os circuitos que exigem mais corrente devem isolar a carga de corrente mais alta dos pinos de E/S do microcontrolador.

Por meio dos pinos, uma infinidade de sensores, atuadores e dispositivos para comunicação pode ser acoplada ao microcontrolador, construindo sistemas complexos de controle. A figura 1 mostra algumas das diversas possibilidades de hardware que podem ser integrados nesses dispositivos.

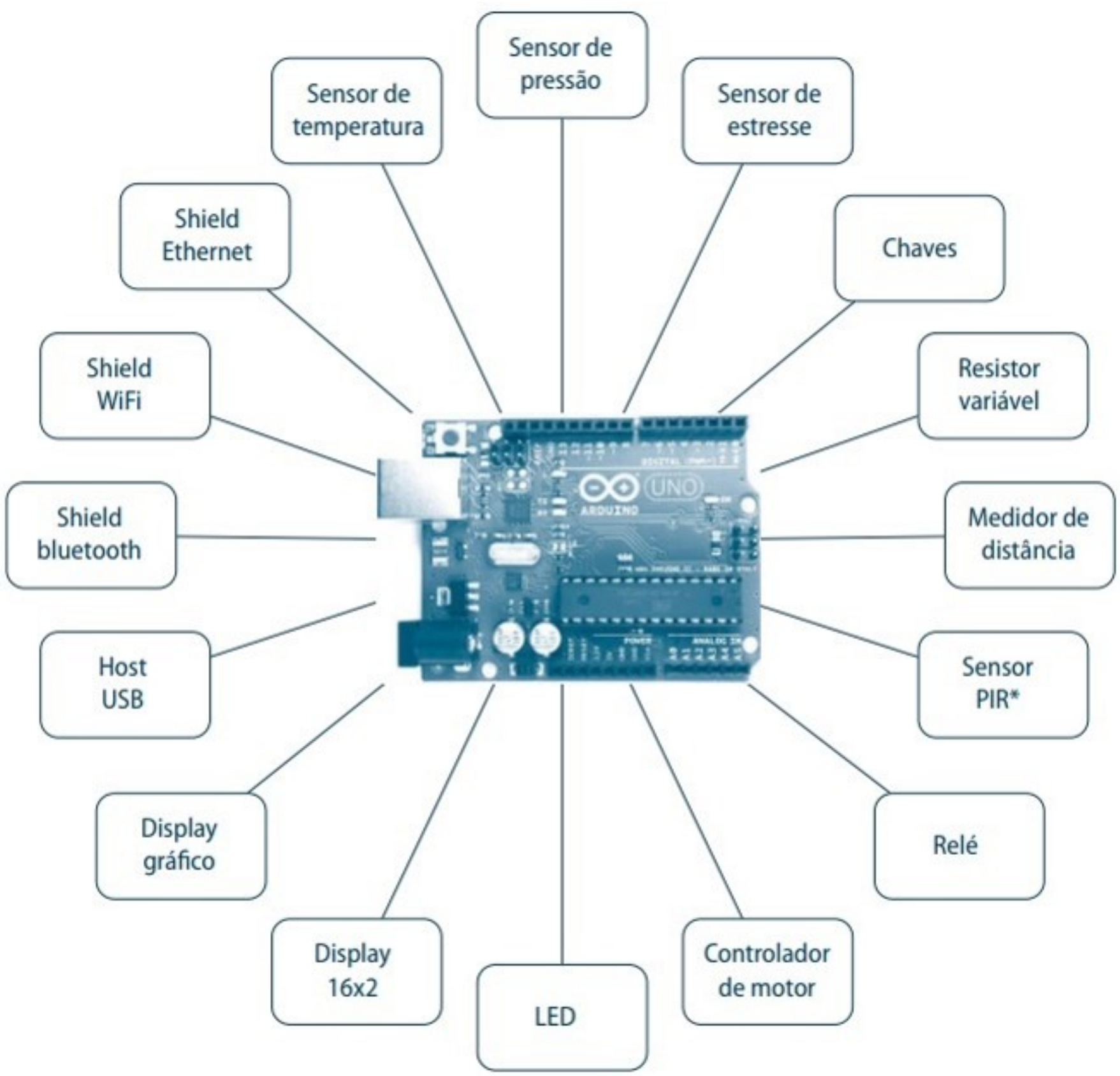

Figura 1. Placa com microcontrolador e dispositivos que podem ser acoplados. (Fonte: Taller Eletrônica).

Programando microcontroladores

Mencione a programação em sistemas embarcados, também conhecidos como embutidos (embedded), e a maioria dos profissionais pensará automaticamente na linguagem C. As linguagens de programação de sistemas embarcados são diferentes das outras, no sentido de que devem ser feitas para acesso a sistemas de baixo nível e requererem recursos relativamente menores do que outras.

Por isso, a linguagem C ainda é dominante nesse meio, apesar de, mais recentemente, aplicações como Internet das Coisas (IoT) e acesso a redes diversas, com protocolos de comunicação e acesso a dados na nuvem, terem atraído linguagens como Python e Javascript para o domínio embarcado. Atualmente, os projetos podem abranger uma variedade de linguagens executadas em plataformas diversas.

Para programar os microcontroladores, usa-se um compilador específico para a família da qual ele faz parte. Em um PC, cria-se o programa que se deseja executar no microcontrolador. Em seguida, conecta-se a placa com o microcontrolador ao PC e se transfere o programa para o microcontrolador. A placa cria uma interface serial COM para se comunicar com o PC. O microcontrolador, então, executa as instruções estabelecidas no programa.

Mesmo não dispondo de uma placa para programar e testar o acesso às portas de microcontroladores, é possível compreender e desenvolver programas para esses dispositivos com o uso de simuladores.

Plataforma Arduino

01 Criada em 2005 por um grupo pesquisadores italianos, a plataforma tinha o objetivo de disponibilizar um dispositivo barato e fácil de programar, sendo, dessa forma, acessível a estudantes e projetistas amadores.

02 Com o conceito de hardware livre (qualquer um pode montar, modificar e personalizar as placas Arduino a partir do hardware básico), a plataforma se popularizou e ganhou seguidores e desenvolvedores que disponibilizam inúmeros recursos de hardware e bibliotecas para a programação facilitada dos sistemas.

03 O modelo mais popular, Arduino Uno R3, é uma placa baseada no microcontrolador ATmega328, do fabricante Atmel. Tem 14 pinos de E/S digital, seis entradas analógicas (usadas para medir tensão, por exemplo, de sensores), um cristal oscilador de 16 MHz, uma conexão Universal Serial Bus (USB), uma entrada de alimentação e um botão de reset.

04 A ideia de disponibilizar placas de baixo custo para popularizar o uso de seus microcontroladores foi seguida por outros fabricantes, como Texas Instruments (LaunchPads MSP430), Microchip (família PIC) e outros.

A programação facilitada na plataforma Arduino é um dos motivos para a grande adoção de seu uso, não pela linguagem de programação, que é a mesma C usada por outras plataformas, mas pela grande quantidade de código disponível para várias funcionalidades.

A Integrated Development Environment (IDE), ou Ambiente de Desenvolvimento Integrado, é repleta de bibliotecas com funções para programar as placas de Arduino e facilitar a integração de periféricos (shields) que são acoplados à placa (figura 1).

Programando o Arduino

No ambiente de programação para Arduino, um programa, chamado de sketch, apresenta duas funções básicas:

1

setup()

é chamada quando um programa começa a executar. Inicializa as variáveis e os tipos dos pinos e declara o uso de bibliotecas, entre outras utilizações. Essa função será executada apenas uma vez depois de a placa Arduino ser ligada ou reiniciada.

2

loop()

Depois de criar uma função setup(), a função loop() executa sempre o mesmo bloco de código, continuamente, permitindo ao programa fazer mudanças e responder às interações com o exterior da placa.

Entrada e saída

A figura 2 apresenta o Arduino Uno, modelo muito popular da plataforma. Essa placa tem 14 pinos digitais que podem ser usados como entrada ou saída, recebendo informação ou enviando sinais para atuação respectivamente. Na linguagem C para o Arduino, as funções pinMode(), digitalWrite() e digitalRead() configuram e realizam as funções de escrita e leitura, operando com 5 V.

Um LED integrado à placa está ligado ao pino 13 do microcontrolador. Colocando esse pino com valor alto (HIGH), o LED acende. Com um valor baixo (LOW), o LED apaga. O programa a seguir estabelece o piscar desse LED a cada segundo.

Observe os comentários no programa para entender o passo a passo.

```
void setup()
{
 pinMode(13, OUTPUT);       //Configura o pino 13 como saída
{
void loop()
{
 digitalWrite(13, HIGH);      //Configura o pino 13 como HIGH
 delay(1000);           //Espera 1.000 ms (1 segundo)
 digitalWrite(13, LOW);       //Configura o pino 13 como LOW
 delay(1000);              //Espera 1.000 ms (1 segundo)
{
```

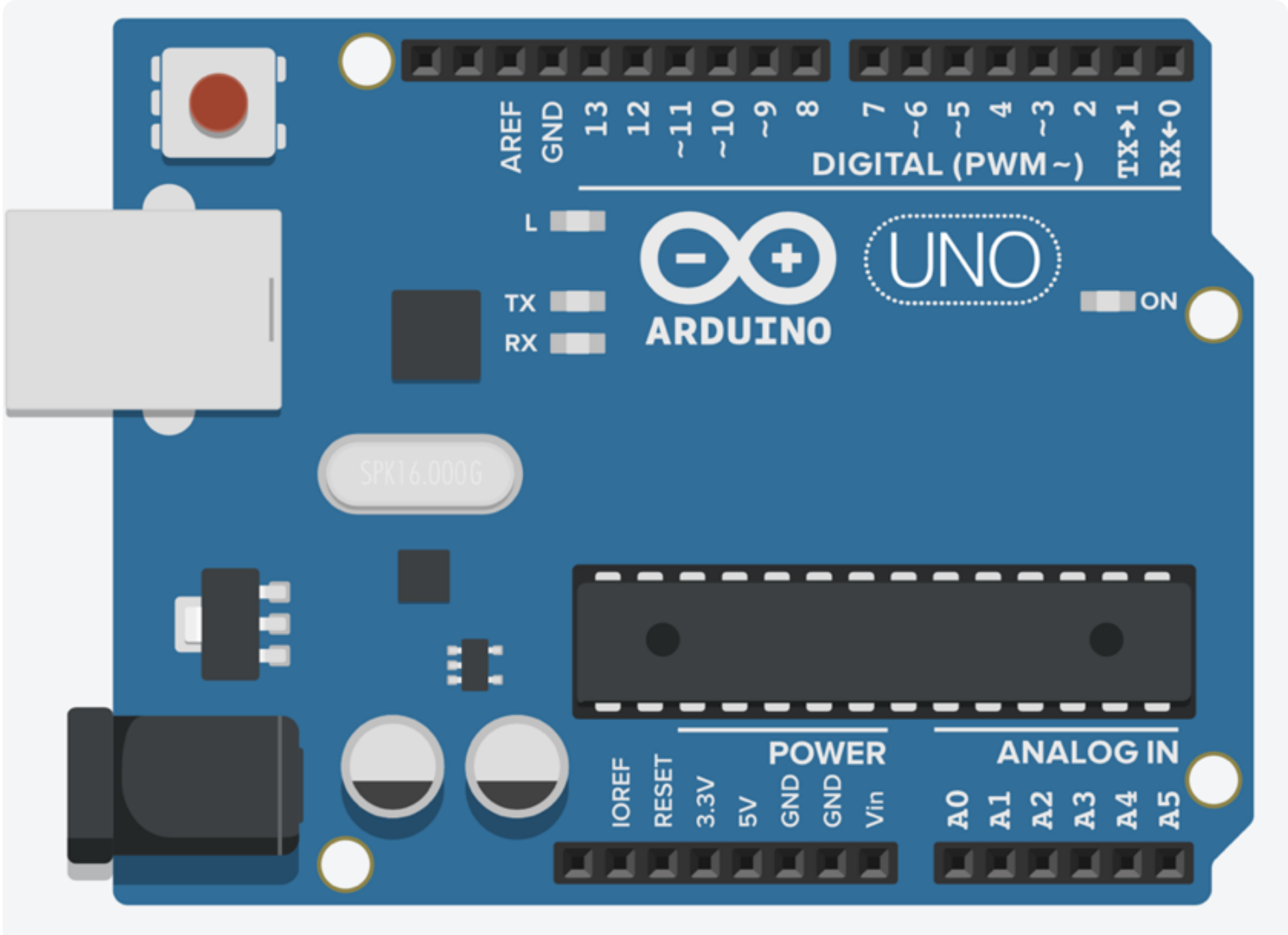

Figura 2 Placa Arduino Uno

Pinos com funções especiais

Clique nos botões para ver as informações.

A0-A5

Os pinos A0-A5 do Arduino Uno são capazes de ler tensões analógicas. No Arduino, o ADC tem resolução de 10 bits, o que significa que pode representar a tensão analógica em 1.024 níveis digitais. O ADC converte a tensão em bits que o microprocessador pode entender. A tensão de referência para essas entradas é dada pelo pino AREF.

0 (RX) e 1 (TX)

A comunicação serial é usada para trocar dados entre a placa Arduino e outro dispositivo serial, como computadores, monitores, sensores e muito mais. Cada placa Arduino tem pelo menos uma porta serial. A comunicação serial ocorre nos pinos digitais 0 (RX) e 1 (TX), bem como via USB.
O Arduino também suporta comunicação serial por meio de pinos digitais com uma biblioteca de comunicação serial via software (SoftwareSerial Library). Isso permite que o usuário conecte vários dispositivos habilitados para serial e deixe a porta serial principal disponível para o USB.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK)

A Interface Periférica Serial (SPI) é um protocolo de dados seriais usado pelos microcontroladores para se comunicar com um ou mais dispositivos externos em uma conexão tipo barramento.
O SPI também pode ser usado para conectar dois microcontroladores. No barramento SPI, sempre há um dispositivo indicado como dispositivo mestre, e o restante como escravos. Na maioria dos casos, o microcontrolador é o dispositivo mestre.

I2C: 4 (SDA) e 5 (SCL):

I2C é um protocolo de comunicação conhecido como “barramento I2C”. O protocolo I2C foi projetado para permitir a comunicação entre componentes em uma única placa de circuito. Com o I2C, existem dois fios chamados SCL e SDA.

Saídas com PWM

Dos 14 pinos de E/S, seis podem ser usados como saídas Pulse Width Modulation (PWM), ou Modulação de Largura de Pulso. São os pinos marcados com um til (~) na placa mostrada na figura 2. PWM é uma técnica de modulação de largura de pulso que pode ser usada para controlar a corrente que flui através de um circuito.

A função analogWrite() é usada para esse fim. Essa função recebe dois parâmetros. O primeiro parâmetro é o pino a ser controlado, e o segundo parâmetro é a quantidade de tempo que o pino permanecerá no nível alto (HIGH).

A figura 3 mostra exemplos de ciclo ativo (ciclo de trabalho) que podem ser atribuídos às saídas PWM dos pinos 3, 5, 6, 9, 10 e 11 do Arduino Uno, com a função analogWrite()

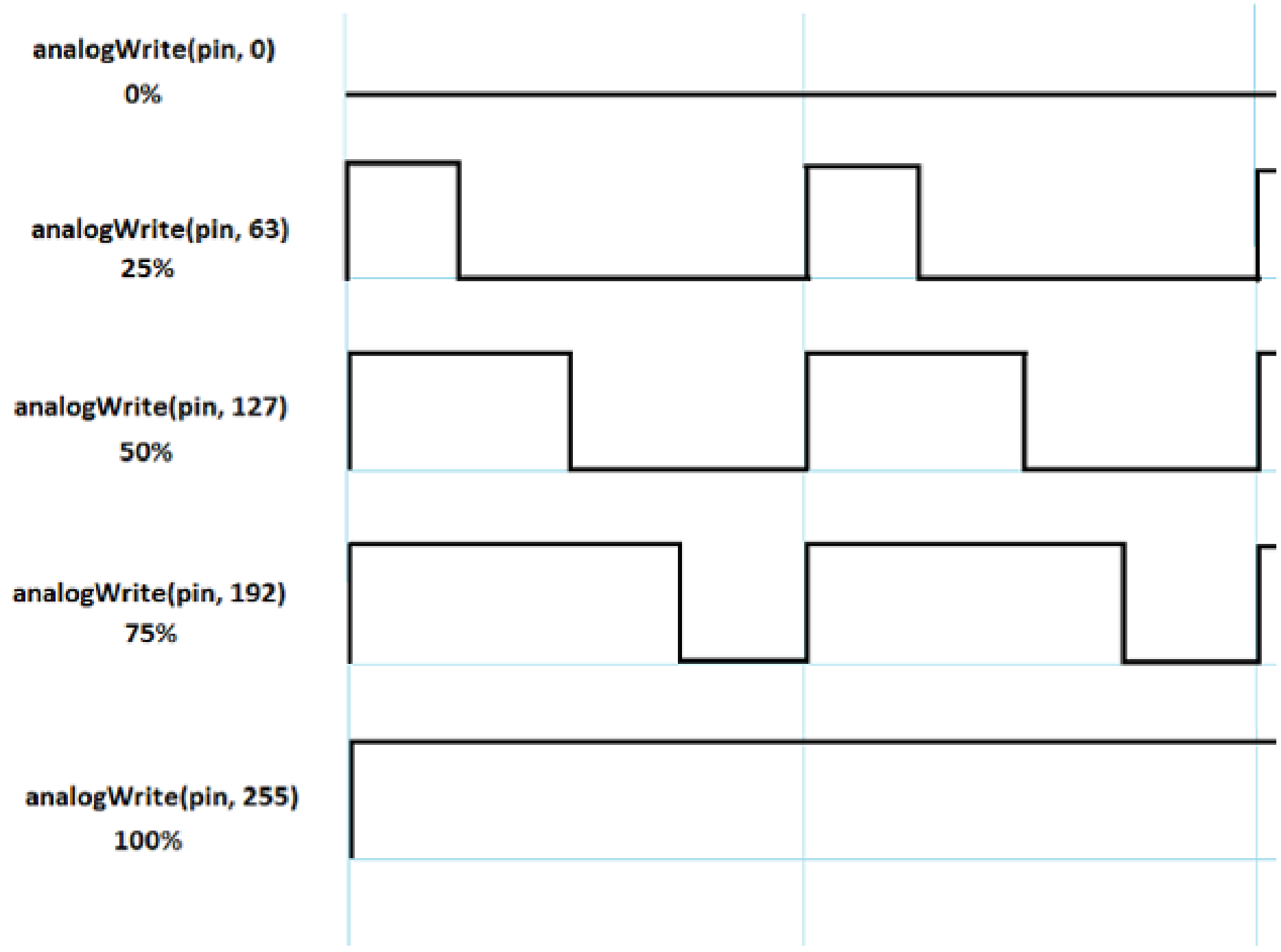

Figura 3 Ciclos ativos que podem ser atribuídos para as saídas com função PWM no Arduino

Sistemas de controle eletrônico

01 Um sistema eletrônico é uma interconexão física de componentes, ou partes, que reúne várias informações juntas. Isso é feito com o auxílio de dispositivos de entrada, como sensores, que respondem de alguma forma a essas informações e, em seguida, usam energia elétrica na forma de uma ação de saída para controlar um processo físico ou executar algum tipo de operação matemática no sinal.

02 Os sistemas de controle eletrônico também podem ser considerados um processo que transforma um sinal em outro para dar a resposta desejada ao sistema. Então, podemos dizer que um sistema eletrônico simples consiste em uma entrada, um processo e uma saída, com a variável de entrada para o sistema e a variável de saída do sistema, sendo ambos sinais. Portanto, sistemas eletrônicos têm entradas e saídas, com a saída ou as saídas sendo produzidas pelo processamento das entradas.

03 Esse processamento pode se dar por um software que esteja sendo executado em um sistema computadorizado, seja um poderoso PC ou um simples microcontrolador. Podemos exemplificar um sistema eletrônico como um simples acionamento de um motor DC em resposta ao apertar de um botão.

Motores DC

O motor DC ou CC é um tipo de motor que trabalha com corrente contínua, produzindo uma velocidade de rotação contínua que pode ser facilmente controlada. Motores DC pequenos são ideais para uso em aplicações onde o controle de velocidade é necessário, como em pequenos brinquedos, robôs e outros circuitos eletrônicos.

Uma maneira simples e fácil de controlar a velocidade de um motor DC é regular a quantidade de corrente por meio de seus terminais, o que pode ser conseguido usando **PWM**.

Comentário

Como o próprio nome sugere, o controle da velocidade de PWM funciona acionando o motor com uma série de pulsos on-off e variando o ciclo de trabalho, a fração de tempo em que a tensão de saída está ligada comparada à quando está desligada, mantendo a frequência constante.

A potência aplicada ao motor pode ser controlada variando a largura desses pulsos aplicados e, assim, variando a tensão DC média aplicada aos terminais dos motores. Alterando ou modulando o tempo desses pulsos, a velocidade do motor pode ser controlada, ou seja, quanto mais tempo o pulso estiver on, mais rápido o motor irá girar e, da mesma forma, quanto mais tempo o pulso estiver off, mais lento o motor vai girar.

Em outras palavras, quanto maior a largura do pulso, mais tensão média será aplicada aos terminais do motor, mais forte será o fluxo magnético dentro dos enrolamentos da armadura e mais rápido o motor irá girar.

Exemplos de controle de dispositivo eletrônico com microcontrolador

Motor DC acionado por apertar de botão de pressão

A figura 4 mostra a montagem de um motor DC ligado ao pino 10 e ao terra (GND) de uma placa Arduino Uno. Um botão de pressão liga o pino 6 ao GND quando acionado.

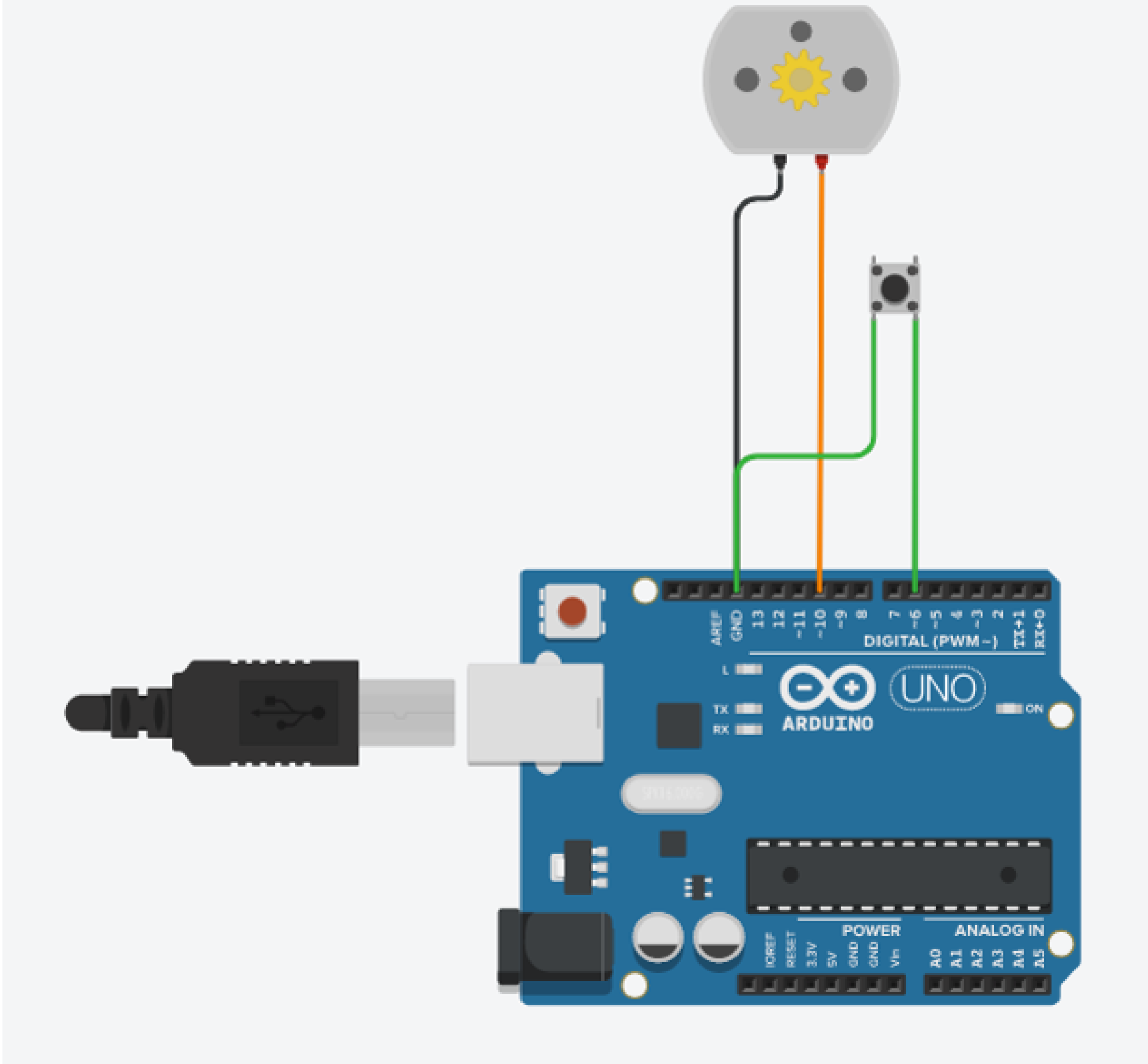

Figura 4. Montagem de um motor DC ligado ao Arduino Uno. (Fonte: Tinkercad).

O código para processar o acionamento do motor ao pressionar o botão é mostrado a seguir.

```
void setup() //Executado apenas uma vez
{
 pinMode(10, OUTPUT);
 pinMode(6, INPUT_PULLUP);
}
void loop()
{
 if (digitalRead(6) == LOW){
 analogWrite(10, 255);
}
 else{
 analogWrite(10, 0);
}
}
```

Comentário

Veja que é necessário colocar o modo do pino de entrada do botão (pino 6) em INPUT_PULLUP para garantir que, quando o botão não estiver apertado, o nível lógico será 1 (HIGH).

Veja que é necessário colocar o modo do pino de entrada do botão (pino 6) em INPUT_PULLUP para garantir que, quando o botão não estiver apertado, o nível lógico será 1 (HIGH).

O código em loop apenas verifica se o pino 6 está ligado ao GND, ou seja, se o botão está apertado (LOW no pino 6). Caso esteja, a função analogWrite() manda o valor 255 para o pino 10, o que significa enviar um nível DC constante (veja a figura 3), ou seja, máxima tensão para o motor DC e máxima velocidade.

A existência das funções usadas na IDE do Arduino torna a tarefa de estabelecer um controle muito simples. A programação em C para processar o que esse programa faz, sem essas funções da biblioteca Arduino já criadas, seria bem mais trabalhosa.

Não se vê, ao usar essas funções, o acesso a registros do microcontrolador, que irão definir o comportamento dele em cada instante.

Saiba mais

O link mostra onde se encontra a função analogWrite() dentro do compilador e pode ser consultado para saber um pouco mais do código mais próximo ao hardware.

Parte da função analogWrite(), que está contida no código da função wiring-analog.c, é mostrada a seguir:

```
void analogWrite(uint8_t pin, int val)
{
          pinMode(pin, OUTPUT);
          if (val == 0)
          {
               digitalWrite(pin, LOW);
          }
               else if (val == 255)
          {
               digitalWrite(pin, HIGH);
          }
          else
          {
          switch(digitalPinToTimer(pin))
          {
          #if defined(TCCR0) && defined(COM00) && !defined(__AVR_ATmega8__)
               case TIMER0A:
               //Conecta o PWM para o pino no timer 0
                    sbi(TCCR0, COM00);
                    OCR0 = val; // set pwm duty
                    break;
               #endif
               .
.
.
```

```
                case NOT_ON_TIMER:
                default:
                        if (val < 128) {
                                digitalWrite(pin, LOW);
                        }else{
                                digitalWrite(pin, HIGH);
                        }
        }
        }
}
```

Comentário

Observe que o código da função analagWrite() aciona funções em nível mais baixo, como o timer (temporizador) do microcontrolador ATmega: Algo que já deveríamos esperar.

Como o controle por PWM monitora o tempo de nível alto e baixo em um pino do microcontrolador, em algum momento uma função teria que controlar esse tempo. Isso é importante para se ter a noção de que funções de bibliotecas, que facilitam programar software básico, dependem de outras funções que atuam em mais baixo nível para acionar o hardware da forma como pretendemos.

Simulador on-line para projetos com Arduino

Os simuladores, muitos rodando on-line em páginas da internet, são ferramentas importantes de aprendizado de sistemas embarcados em microcontroladores, simulando também dispositivos básicos de eletrônica, como chaves, LEDs e motores, com a vantagem de não haver risco de danificar componentes em caso de erros de montagem.

Saiba mais

Na plataforma Tinkercad da Autodesk é possível simular circuitos com a placa Arduino Uno. É necessário um cadastro simples na plataforma para copiar, editar e criar circuitos e códigos.
O vídeo mostra como acessar a plataforma, caso tenha dúvida sobre como fazê-lo.
O exemplo de circuito e código do acionamento do motor, mostrado anteriormente, pode ser encontrado no link. Acessando esse link, você verá uma página semelhante à figura 5. Você poderá clicar em Simular e ver o circuito funcionando, de forma semelhante ao que veria em um circuito real. Clicando no botão Código, você verá o código para acionamento do motor DC, já informado acima.

Com o cadastro, e fazendo login na plataforma, você poderá copiar o circuito para sua pasta e realizar alterações nele. Faça isso para poder realizar as atividades desta aula.

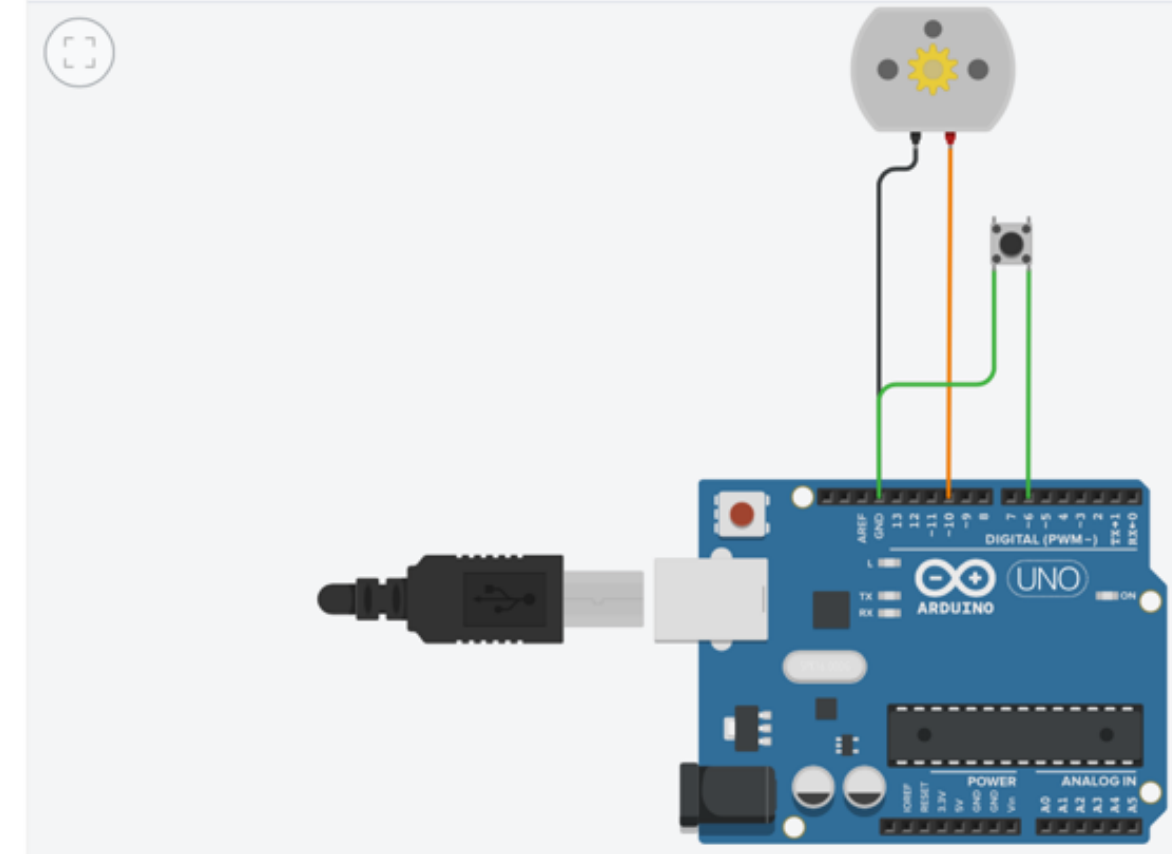

Figura 5. Página da montagem na plataforma Tinkercad. (Fonte: Tinkercad).

Vejamos um exemplo mais elaborado, onde a velocidade do motor pode ser controlada.

Motor DC alterando a velocidade por apertar de botão de pressão

Uma alteração no código para mudar o processamento da informação de apertar um botão (essa entrada passa a fazer a velocidade do motor variar entre quatro valores diferentes) é mostrada a seguir:

TÍTULO

```
int velocidade = 0;

void setup()
{
 pinMode(10, OUTPUT);
 pinMode(6, INPUT_PULLUP);
}

void loop()
{
 delay(200);
 if (digitalRead(6) == LOW){
 velocidade++;

 switch (velocidade)
{
   case 1:
   analogWrite(10, 64);
   break;
   case 2:
   analogWrite(10, 128);
   break;
```

```
    case 3:
    analogWrite(10, 192);
    break;
    case 4:
    analogWrite(10, 255);
    break;
    default:
    velocidade = 0;
    analogWrite(10, 0);
    }
}
}
```

Comentário

O exemplo pode ser copiado e editado na plataforma Tinkercad.

Atividade

1. Podemos afirmar que a linguagem de programação do Arduino é um subconjunto da linguagem C/C++? Por quê?

2. Sabe-se que a função de leitura de pino digital no ambiente Arduino tem a sintaxe digitalRead(pino), na qual pino é o número do pino digital que se quer verificar o estado, e que essa função retorna HIGH ou LOW para indicar o estado do pino. Faça um programa que acenda o LED no pino 13 com o apertar de uma chave (botão) no pino 2. Use o Tinkercad para simular.

3. Suponha que seja passada para você uma tarefa em que precise descobrir como acionar outro tipo de motor, um servomotor, utilizando Arduino. O movimento pedido é para que o servomotor gire em 90 graus e retorne ao ponto inicial em aproximadamente nove segundos, com movimento contínuo e repetitivo. Como realizar essa tarefa?

4. Podemos afirmar que a linguagem de programação do Arduino pode ser dividida em três partes principais: Funções, valores (variáveis e constantes) e estruturas. Dê exemplos de cada uma dessas partes.

5. Qual o resultado real da execução das duas instruções seguintes no Arduino Uno?

```
pinMode(5,OUTPUT);
analogWrite(5,128);
```

Notas

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Título modal [1]

Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos. Lorem Ipsum é simplesmente uma simulação de texto da indústria tipográfica e de impressos.

Referências

DI RENNA, Roberto Brauer; PAIVA, Lorraine de Miranda. **Tópicos especiais em eletrônica II**: Introdução ao microcontrolador Arduino. Niterói, 2014. Disponível em: https://www.telecom.uff.br/pet/petws/downloads/apostilas/arduino/apostila_de_programacao_arduino.pdf. Acesso em: 29 dez. 2019.

OLIVEIRA, Cláudio Luís Vieira; ZANETTI, Humberto Augusto Piovesana. **Arduino descomplicado**: Como elaborar projetos de eletrônica. São Paulo: Érica, 2015.

TUTORIAL Arduino. Disponível em: //www.dsc.ufcg.edu.br/~joseana/IC_TutorialArduino_TinkerCad.pdf. Acesso em: 28 dez. 2019.

Próxima aula

- Conceitos de aquisição de dados e uso de sensores, utilizando o Arduino.
- Programação cliente/servidor.
- Processos em sistemas operacionais.

Explore mais

O Arduino é fácil de operar e, depois de algum tempo de dedicação, você pode experimentar muitas montagens que lhe darão mais conhecimento sobre a programação de software básico com microcontroladores em várias aplicações, como a domótica. Os materiais a seguir mostram algumas dessas aplicações e referências das bibliotecas:

- Automação residencial de baixo custo com Arduino Mega e Ethernet Shield.
- Documentação de referência da linguagem Arduino.
- Introdução ao Arduino – exemplos de programação, circuitos e uso de sensores.