UNIDADE 5: SINAIS ANALÓGICOS

OBJETIVOS

Até ao momento, assumimos que o Arduino utiliza apenas sinais digitais com níveis de “1” ou “0”. Os sinais de *input* podem ser emitidos por botões, interruptores, detectores entre muitas outras origens, mas devem ser sempre precedidos de um “1” ou de um “0”. Os sinais de *input* podem alimentar luzes LED, interruptores diferenciais, motores e muitos outros dispositivos, mas é sempre necessário incluir-se um nível “1” ou “0”. Mesmo os sinais PWM que utilizou para controlar o brilho de uma luz LED, ou o posicionamento de um servomotor são digitais, assim como a frequência e o som de uma altifalante.

No entanto, nem tudo no mundo natural é digital. Há também o que são chamados de sinais “analógicos” e o seu valor ou voltagem podem variar entre um mínimo e um máximo durante um período de tempo.

Nesta unidade, vamos focar-nos em que consistem esses sinais, como os utilizar e o que pode fazer com eles. Vamos centrar-se nas funções da linguagem Arduino que manipulam os *inputs* analógicos do controlador.

SECÇÃO TEÓRICA

SECÇÃO PRÁTICA

- EXERCÍCIO 1: Conversão ADC
- EXERCÍCIO 2: Limites
- EXERCÍCIO 3: Comparador analógico
- EXERCÍCIO 4: Controlador de brilho
- EXERCÍCIO 5: Leme elétrico
- EXERCÍCIO 6: Fotómetro
- EXERCÍCIO 7: Controlador de luz
    - Agora é a sua vez!
- EXERCÍCIO 8: Medição de reflexos
    - Agora é a sua vez!
- EXERCÍCIO 9: Detetar cores
- EXERCÍCIO 10: Temperatura
- EXERCÍCIO 11: Aparelhos de ar condicionado

MATERIAL

-Laptop ou computador de secretária.

-Ambiente de trabalho Arduino IDE; deve incluir o material suplementar já instalado e configurado.

-Placa de controlo Arduino UNO.

-Um cabo USB.

-O servomotor FUTABA S3003.

ÍNDICE

SECÇÃO TEÓRICA

1. INTRODUÇÃO

A expressão “sinais de input analógicos” pode ter surgido em algum momento do curso, mas o que são, para que são utilizados, onde estão e como são controlados? Chegou o momento de responder a estas perguntas.

Tudo no mundo “digital” funciona com o pressuposto de que existem apenas dois valores ou níveis possíveis: nível “1” e nível “0”. Um botão, um interruptor ou um detector podem estar ligados (“1”) ou desligados (“0”). Uma luz LED, um interruptor de transmissão ou um motor podem ser ligados ou desligados. O som é apenas um sinal que passa do nível “1” para o nível “0” a uma determinada velocidade ou frequência. Um sinal PWM é digital e podemos variar o comprimento do nível “1” ou o ciclo de funcionamento e assim regular a voltagem. A comunicação em série é nada mais do que a transferência de bits com níveis de “1” e “0”.

Quando temos um número de bits, agregamo-los em bytes e usamo-los para criar números de diferentes tamanhos, codificar carateres e enviar mensagens. Esta é a forma como tem trabalhado até agora.

No entanto, o mundo “real” não é assim. Deparamo-nos com quantidades físicas no mundo natural que podem ter valores múltiplos ou outras características. De qualquer forma, de certo que está bem consciente de que nem tudo é “preto ou branco”; existem gradações de cinzento também.

Pense na temperatura ambiente, por exemplo. É uma quantidade física que se altera constantemente. A temperatura não é a mesma de manhã, a meio do dia ou à noite. Se tivessemos um sensor que pudesse medir a temperatura e gerar uma voltagem proporcional, aperceber-nos-íamos de que varia constantemente através dos tempos.

A temperatura é uma quantidade física analógica. O sensor fornece uma voltagem de 2 V às 5 horas da manhã. Às 9 horas, a temperatura aumenta e, portanto, a voltagem também aumenta, subindo para 4 V. Às 3 horas, a voltagem atinge 5 V e, a partir das 4 horas, começa a decrescer com a descida da temperatura. Tudo o que temos a fazer é procurar uma relação entre a temperatura e a voltagem comunicada pelo o sensor.

Agora tenha em conta as inúmeras quantidades físicas que nos rodeiam. Utilizando os sensores, ou “transdutores”, pode transformar estas quantidades físicas em voltagens analógicas correspondentes:

- ✓ Humidade e/ou humidade relativa. Permite-nos calcular a quantidade de vapor de água na atmosfera.
- ✓ Pressão atmosférica. Podemos descobrir a pressão que o ar exerce na Terra usando um sensor adequado.
- ✓ Peso. Podemos medir a força que um corpo exerce num ponto em repouso.
- ✓ Velocidade. Com um sensor adequado, poderia medir a velocidade com que o ar se move, a velocidade a que um veículo se movimenta, ou a sua própria rapidez de movimento.
- ✓ Luz. A luz ambiente ou a luz que atinge um objeto pode ser medida. Existem sensores que detectam a luz visível, a luz infravermelha, a luz ultravioleta entre outras.

✓ Som. Podemos medir, estudar, e/ou detetar ruídos, volume e outros fenómenos acústicos.

Existem muitas outras áreas em que sensores ou “ransdutores” podem transformar as quantidades físicas em voltagens analógicas equivalentes.

2. CONVERSÃO DIGITAL

Como pode imaginar, existem diversos tipos de sensores e “transdutores” que são capazes de medir e de fornecer voltagem analógica equivalente à quantidade física que estiver a medir. Infelizmente, nenhum sistema digital é capaz de lidar ou de processar essas correntes analógicas diretamente; nem mesmo o Arduino.

O que é preciso fazer-se em primeiro lugar é converter as tensões analógicas em valores digitais ou bináriosequivalentes. Para realizar esta conversão usamos circuitos eletrónicos chamados “conversores analógico-para-digital” ou, abreviado, ADC.

A Figura 1 apresenta os componentes necessários para processar uma quantidade física analógica, como a temperatura, por exemplo.

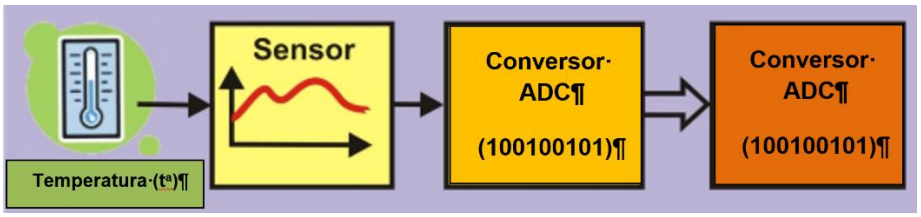


Figura 1

1. O sensor detecta a quantidade física que é projetado para medir; neste caso é a temperatura. Geralmente gera uma tensão proporcional à quantidade física.

2. A voltagem é, então, introduzida no conversor ADC. O circuito fornece um valor binário equivalente ao da voltagem introduzida.

3. O valor binário pode, agora, ser lido por um controlador como o Arduino.

4. O controlador pode executar qualquer processo com este valor binário: armazená-lo na memória, executar operações aritméticas ou lógicas, exibi-lo, transferi-lo para outro sistema entre muitos outros.

Quase todos os controladores modernos, incluindo o Arduino, têm um circuito conversor ADC integrado. Tudo o que será necessário é ligar o sensor ou o transdutor apropriado ao pin de entrada analógico. O tipo de sensor ou transdutor que utilizar dependerá da quantidade física que vai medir.

Para além disso, esses controladores normalmente possuem vários pins de entrada para sinais analógicos. A maioria dos controladores tem apenas um circuito conversor ADC com varios pins de

input ou “canais analógicos” conectados. O Arduino UNO possui um único conversor e seis pins de *input* analógicos conectados – números A0 a A5 – o que lhe permite recolher amostras de voltagens analógicas de até seis sensores diferentes.

Claro que só pode tirar uma amostra de cada vez. Utilizando funções adequadas, é possível converter as voltagens de cada canal de *input* analógico, um de cada vez. Dizemos que os canais de *input* são “multiplexados”.

Observe atentamente o gráfico na Figura 2. Sempre que solicitar uma conversão, o conversor ADC tira uma amostra da voltagem analógica na entrada que especificar. O resultado, ou o seu equivalente binário, é obtido em 100 µS (0,0001”), o tempo necessário para se efetuar a conversão. O Arduino pode realizar aproximadamente 10 mil conversões por segundo.

A voltagem analógica é de 2 V no primeiro momento de tempo. O conversor ADC gera um valor ou número binário equivalente a essa voltagem. A voltagem analógica é de 2,8 V no segundo momento, de 3,5 V no terceiro, 4,2 V no quarto e assim por diante. Pode verificar como funciona um sinal analógico: flutua constantemente entre um mínimo – 0 V no exemplo – e um máximo – 5 V.

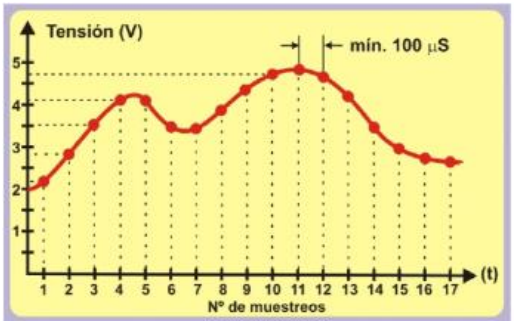


Figura 2

Já se abordou o facto de o Arduino UNO necessitar de aproximadamente 100 µS para realizar uma conversão. Isto é relativamente rápido, mas existem outros programas que podem fazê-lo muito mais rápido. Observe a Figura 3, onde se mostra um sinal analógico com uma frequência, F, de 1000 Hz. O seu período, T, é 1000 µS (1 mS). Isto significa que o Arduino seria capaz de conduzir, no máximo, dez amostras do sinal (1000 µS / 100 µS).

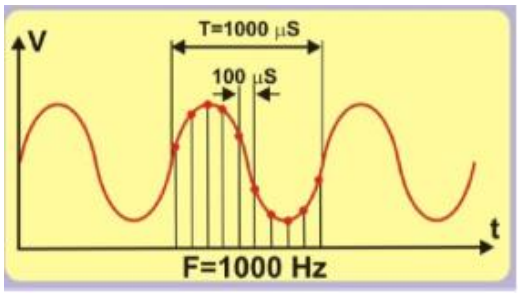


Figura 3

Se tivesse um sinal analógico com uma frequência, F, de 500 Hz e um período, T, de 2000 µS, poderia alcançar um total de vinte amostras (2000 µS/ 100 µS). Isto significa que poderia obter uma “digitalização” mais exata do que quando a frequência do sinal analógico era de 1000 Hz.

Nem sempre é necessário ter muitas amostras por segundo. Lembra-se do sensor que mede a temperatura ambiente? Esta é uma quantidade física que não é muito moldável. A temperatura não aumenta de -10ºC a + 15ºC em 0,0001 segundos. O mesmo acontece com a luz natural. Não passamos de noite para dia em 100 µS. E quanto ao peso? Nada pesa 75 kg num momento e 31 kg 100 µS mais tarde. O mesmo se aplica a outras quantidades físicas como a velocidade, a pressão atmosférica, a humidade entre muitas outras.

A velocidade de conversão do Arduino UNO é mais do que suficiente para medir a maioria das quantidades físicas analógicas!

3. RESOLUÇÃO

Para além da velocidade de conversão, outro fator importante num circuito conversor ADC é a precisão ou "resolução". Dizemos que o conversor incorporado no Arduino UNO possui uma resolução de 10 bits. Isso significa que o resultado de uma conversão pode ter 1024 valores binários possíveis (2^{10}).

Como estabelecemos uma relação entre a voltagem analógica e o valor binário? Precisamos de conhecer uma constante chamada "tensão de referencia" ou VREF, que é a tensão que o circuito conversor utiliza para executar as suas operações internas. Calculamos a resolução por bit usando a seguinte equação, que depende do VREF e do número de bits que o conversor obteve. Supondo que V_{REF} = 5V:

$$Resolución = \frac{V_{REF}}{2^{10}} = \frac{5}{1024} = 0.0048V/Bit \cong 0.005V = 5mV$$

Se tiver consciencia relativamente a este aspeto, pode prever o valor de *output* binário que o conversor oferece com base na voltagem de entrada analógica. Observe atentamente a Tabela 1.

Tabela 1

VOLTAGEM DE INPUT	OUTPUT			VOLTAGEM DE OUTPUT	OUTPUT		
	BINÁRIO	DEC.	HEX.		BINÁRIO	DEC.	HEX.
0.000	0000000000	0	0x000	**2.500**	1000000000	512	0x200
0.005	0000000001	1	0x001	**3.000**	1001100110	614	0x266
0.010	0000000010	2	0x002	**4.000**	1100110011	819	0x333
0.015	0000000011	3	0x003	**4.985**	1111111100	1020	0x3FC
0.020	0000000100	4	0x004	**4.990**	1111111101	1021	0x3FD
1.000	0011001100	204	0x0CC	**4.995**	1111111110	1022	0x3FE
2.000	0110011000	408	0x198	**5.000**	1111111111	1023	0x3FF

Tudo o que precisa de fazer é dividir a voltagem analógica de *input* pela resolução do bit do conversor, que neste caso é 0,0048.

A. AGORA É A SUA VEZ!

Supondo que a tensão de referência VREF seja de 3 V e o conversor tenha uma resolução de 8 bits, complete a Tabela 2 para cada uma das diferentes voltagens de *input* analógicas.

Tabela 2

RESOLUÇÃO DE BIT							
VOLTAGEM DE INPUT	OUTPUT			**VOLTAGEM DE OUTPUT**	OUTPUT		
	BINÁRIO	**DEC.**	**HEX.**		**BINÁRIO**	**DEC.**	**HEX.**
0.00				**1.50**			
0.01				**1.67**			
0.50				**1.85**			
0.65				**2.12**			
0.83				**2.57**			
0.95				**2.93**			
1.15				**3.00**			

ISTO É MUITO IMPORTANTE! A voltagem de *input* analógica que medir NUNCA pode exceder a voltagem V_{REF} de referência. Por exemplo, se V_{REF} = 5 V, a voltagem de *input* analógica tem de ser no máximo também de 5 V.

4. FUNÇÕES NA LINGUAGEM ARDUINO

Usar o conversor ADC incorporado no Arduino UNO não podia ser mais fácil. Precisa apenas de duas funções da linguagem de programação Arduino.

- **A função analogReference()**

Esta função permite definir o valor da voltagem de referência (VREF) que o circuito conversor ADC deve usar para converter uma voltagem analógica no seu equivalente binário ou decimal.

Isto é importante porque, sabendo o que é, pode calcular a resolução por bit como fez anteriormente.

A voltagem VREF não deve exceder a voltagem que alimenta o controlador Arduino UNO (5 V). Por outro lado, a voltage de input analógica convertida não pode exceder o VREF.

Sintaxe:

analogReference(type);

type: Configura a V_{REF} utilizada para o input analógico (i.e. o valor utilizado como o máximo da média de input). As opções são:

DEFAULT: a referência analógica por defeito de 5 volts (nas placas Arduino 5V) ou de 3.3 volts (nas placas Arduino 3.3V).

INTERNAL: Esta é uma V_{REF} gerada dentro do controlador. No caso do Arduino UNO é 1.1 V

EXTERNAL: A V_{REF} requerida é descarregada no pin do controlador AREF.

Exemplo:

analogReference(INTERNAL); *//É utilizada a voltagem interna de 1.1 V*

- **A função analogRead()**

Essa é a frase que vai utilizar quando quiser realizar uma conversão de analógico para digital. Cada vez que é executada, é necessária uma amostra da voltagem no pin ou canal analógico especificado e, em seguida, realiza-se a conversão.

Sintaxe:

analogRead(pin);

pin: O número do pin que corresponde ao canal analógico que deseja converter. No caso do Arduino UNO, pode ser de A0 a A5.

Exemplo:

int V;

V = analogRead(A2); *//Realiza a conversão da voltage em A2*

- **A função map()**

Mesmo que esta função não seja expressamente desenhada para a conversão de ADC, pode ser interessante para o que estamos a fazer no momento. Torna possível remapear, reatribuir ou redefinir um valor entre um mínimo e um máximo.

Revendo: já sabe que o conversor Arduino ADC tem uma resolução de 10 bits e que pode expressar um valor entre 0 e 1023 (2 ^ 10). Por outro lado, os números com os quais o Arduino trabalha vêm em pacotes de bytes (8 bits) ou múltiplos de bytes (int, unsigned int, long e unsigned long). Às vezes, é melhor usar o resultado de uma conversão (10 bits) como se fosse um byte entre 0 e 255 (8 bits) ou um int (16 bits) e assim por diante. Bytes ou *integers* são mais formatos padrão

Sintaxe:

map(value, fromLow, fromHigh, toLow, toHigh);

value: o número do mapa. Normalmente é um int (16 bits) ou um long (32 bits). Não podem ser utilizados valores variáveis.

fromLow: Expressa o limite inferior do intervalo atual do valor.

fromHigh: Expressa o limite superior do intervalo atual do valor

toLow: Expressa o limite inferior do valor que se pretende alcançar.

toHigh: Expressa o limite superior do valor que se pretende alcançar.

Exemplo:

```
int V;

V = analogRead(A2);          //Realiza a conversão da voltage em A2

V = map(V,0,1023,0,255);     //Atribui novamente o valor que leu e converte-o num byte
```

O valor analógico que se lê no pin A2 fica entre 0 e 1023 e é armazenado na variável “V”. Esse valor é remapeado para um equivalente entre 0 e 255. De acordo com o Arduino, essa função é derivada do seguinte cálculo matemático (que incluímos apenas pelo seu interesse):

$$Resultado = \frac{(valor - min) * (Nmax - Nmin)}{(max - min) + Nmin}$$

5. PERIFÉRICOS ANALÓGICOS

Existe uma ampla variedade de sensores no mercado que podem fornecer voltagem analógica entre um mínimo e um máximo, dependendo da quantidade física em questão. Pode considerá-los como periféricos analógicos.

- ✓ **Potenciometros Analógicos.** Mover os eixos fornece uma voltagem analógica variável entre 0 e 5 V. Estão ligados aos pins A0 e A1 e podem ser considerados como sensores de movimento analógicos.
- ✓ **Sensor de Luminosidade**. Está ligado ao pin A2. A voltagem que gera depende da quantidade de luz ambiente que o atingir.
- ✓ **Sensor de Infravermelhos (IR)**. Este é um sensor reflexivo de luz infravermelha (IR) não visível. Está ligado ao pin A3 e mede a luz infravermelha que o atinge.
- ✓ **Sensor de Temperatura**. Mede a temperatura ambiente e gera voltagem proporcionalmente. Está ligado ao pin A4.

A. POTENCIOMETROS

Os potenciometros são registências variáveis, cujo valor pode ser modificado movendo o eixo ou um controle chamado “alavanca de contacto”. De certo que já os usou muitas vezes sem perceber; o controlo remoto que utiliza para ajustar o volume de um rádio, televisão ou sistema de som são alguns exemplos.

Estes são os periféricos analógicos mais simples e económicos que poderá encontrar. Existem em todas as formas e tamanhos.

Como pode verificar na Figura 4, tem três terminais ou pins. Os pins 1 e 2 emcontram-se nas extremidades da resistência e representam o seu valor total. O pin 3 é a alavanca de contacto. Existe um mecanismo que a movimenta de uma extremidade da resistência à outra, variando assim o seu valor. Se a posicionar no meio do percurso entre os pins 1 e 3, terá metade da resistência. Se a colocar entre os pins 3 e 2, terá a outra metade (Figura 4).

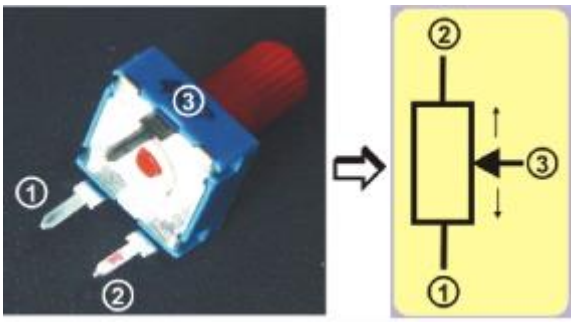


Figura 4

Agora preste atenção às ligações de ambos os potenciometros. As terminações estão ligadas ao 0 V e +5 V. Se mover a alavanca de contacto em direção ao limite inferior, a voltagem analógica irá

decrescer até atingir o máximo de +5 V. As alavancas de controlo de ambos os potenciometros estão ligadas aos pins de input analógicos A0 e A1 no controlador.

B. SENSORES FOTOGRÁFICOS

Os sensores fotográficos baseiam-se num pequeno dispositivo chamado “fototransistor”. Dispensando longas explicações técnicas, podemos afirmar que este componente aumenta ou diminui a quantidade de corrente que passa por ele com base na quantidade de luz que o atinge. Há dispositivos visíveis ou sensíveis à luz ambiente e outros que são sensíveis à luz infravermelha (IR). Existem em diversas formas e tamanhos.

Como se apresenta na figura 5, podemos usar uma lanterna para manipular a quantidade de luz que atinge o sensor. Isto aumenta ou diminui a corrente elétrica que flui através do sensor. Esta corrente, I, circula através de uma resistência, R, e produz uma voltagem, V, que também varia proporcionalmente: V = R * I. Em suma, quando a luz muda, a intensidade da corrente, I, também se altera, o que, por sua vez, modula a voltagem, V, entre entre 0 e 5 V. Esta tensão é fornecida ao input analógico A2 do Arduino.

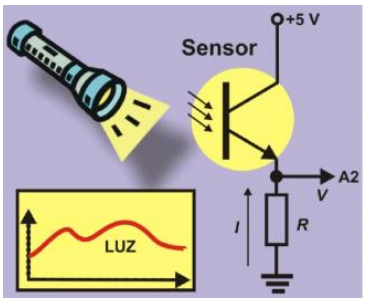


Figura 5

C. SENSORES REFLETORES IR

Os sensores refletores IR são outro tipo de sensor de luz: detectam a luz infravermelha (IR) não visível ao olho humano. O sensor refletor IR detecta a quantidade de luz refletida que o atinge. O sensor é composto por dois componentes. A luz LED ou emissor (E), que funciona em +5 V, emite um feixe constante de luz infravermelha. Quando se reflete num objeto, atinge o fotodiodo ou receptor (R) que absorve a luz (figura 6). Dependendo da quantidade de luz refletida, o receptor aumenta ou diminui a intensidade, I, que alimenta o resistor, R, que, por sua vez, aumenta ou diminui a voltagem analógica, V. Lembre-se: V = R * I. Resumindo: quanto mais próximo o objeto, maior a quantidade de luz refletida.

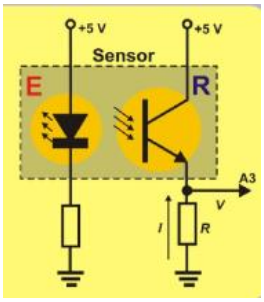


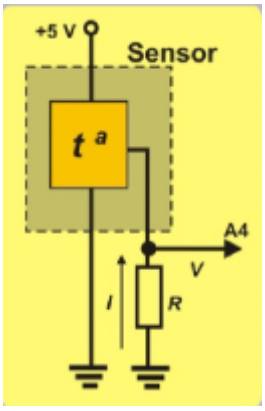
Figura 6

A intensidade, I, é maior e, portanto, a voltagem, V, também. Esta voltagem é aplicada ao pin de input analógico A3 no Arduino UNO.

D. SENSOR DE TEMPERATURA

Dispositivo LM35, este componente tem três pins. Dois deles estão ligados à voltagem de alimentação de 0 e +5 V. A intensidade, I, que circula pelo terceiro pin, é diretamente proporcional à temperatura. Essa intensidade atravessa a resistência, R, criando uma voltagem, V (V = R * I). Está ligado ao pin de input analógico A4.

De acordo com o fabricante deste dispositivo, a resolução do sensor é de 10 m V / ºC. Pode usar a equação baseada na voltagem analógica (Va) para calcular a temperatura ambiente em graus centígrados, como se apresenta na figura 7.



$$^{o}C = \frac{5 * Va * 100}{1024} = \frac{Va * 500}{1024}$$

Figura 1

6. OUTPUT PSEUDO ANALÓGICO

A plataforma Arduino e inputs e outputs digitais. Verificou que a placa controladora Arduino UNO tem 14 pins que podem ser configurados como inputs ou outputs digitais. Na verdade, usou a maioria deles em exercícios que tem feito até agora.

Recorde que seis desses pins também podem funcionar como pins de output de sinal PWM. Referimo-nos aos pins 3, 5, 6, 9, 10 e 11, os precedidos do sinal "~"; eles têm uma seta aque também os sinaliza.

Vai utilizar estes pins como outputs de sinal PWM nesta unidade.

As saídas de sinal PWM números 6, 9, 10 e 11 são conectadas aos LEDs. Os números 3 e 5, que também são pins de output de sinal PWM, permitem ligar dois servomotores ou um motor CC; nós vamos ligar dois servomotores nesta unidade.

A. O QUE SÃO SINAIS PWM?

A abreviatura "**PWM**" significa "*Pulse Width Modulation*", um sinal periódico digital "*asimétrico*" de "1"s e "0"s que se repete constantemente na mesma frequência, **F** (existem X números de ciclos de repetições num segundo). Isto significa que o momento em que o sinal está no nível "1" pode ser completamente diferente da hora em que o sinal está no nível "0".

Observe o sinal representado na Figura 8. É um sinal digital com um período, **T**, de 2 mS. Por outras palavras, existem 500 ciclos idênticos em um segundo (1000 mS / 2). A frequência, **F**, é, portanto, de 500 Hz (F = 1/T).

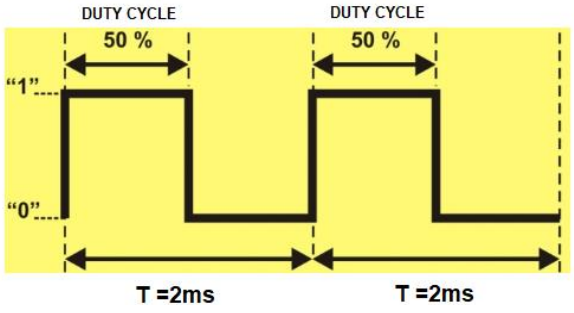


Figura 2

Neste caso, o nível "1" de cada ciclo dura o mesmo tempo que o nível "0": 1 mS (se adicionarmos nível "1" e nível "0" são iguais a 2 mS do período, T). A isto chama-se sinal "simétrico". A hora em que o sinal está no nível "1" denomina-se de "ciclo de serviço". O seu valor é 50% (1 mS) do valor total do período no exemplo. Denomina-se de sinal PWM de 50%.

Vamos observar os quatro sinais PWM na Figura 9. Todos têm o mesmo período, **T**, de 2 ms, ou uma frequência, **F**, de 500 Hz, o que na verdade é exatamente o mesmo. No entanto, a duração do nível "1", isto é, o ciclo de trabalho, é diferente em todos eles.

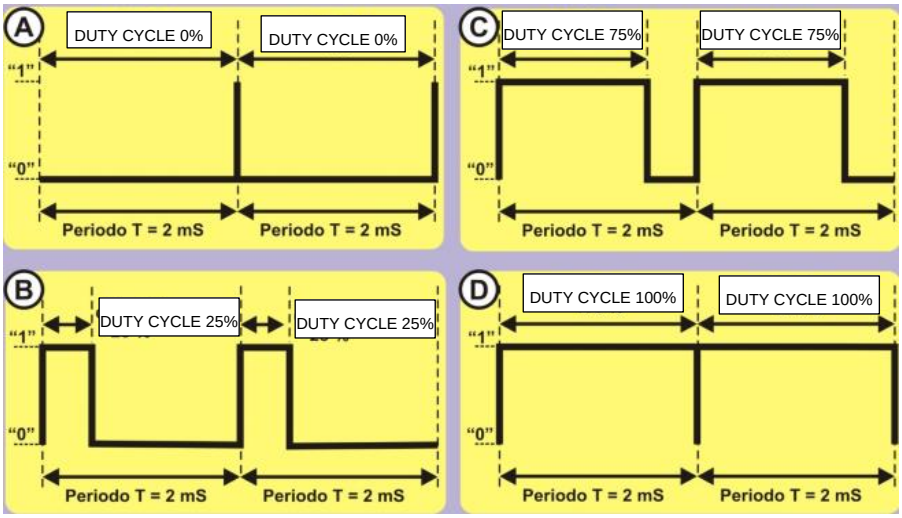


Figura 3

- ✓ **Sinal A**. Este é um sinal PWM com um ciclo de trabalho de 0% do comprimento do período. Isto é, permanece no nível "1" para 0 mS (0% * 2/100) e no nível "0" para 100% do tempo restante, ou seja, 2 mS (100% * 2 / 100).
- ✓ **Sinal B**. Este é um sinal PWM com um ciclo de trabalho de 25% do comprimento do período. Por outras palavras, permanece no nível "1" para 0.5 mS (25% * 2/100) e no nível "0" para 75% do tempo restante, ou seja, 1.5 mS (75% * 2 / 100).
- ✓ **Sinal C**. Este é um sinal PWM com um ciclo de trabalho de 75%. Permanece no nível "1" para 1.5 mS (75% * 2/100) e no nível "0" para 25% do tempo restante, ou seja, 0.5 mS (25% * 2 / 100).
- ✓ **Sinal D**. Este é um sinal PWM com um ciclo de trabalho de 100%. Permanece no nível "1" por 2 mS (100% * 2/100) e no nível "0" por 0% do tempo restante, ou seja, 0 mS (0% * 2 / 100).

Já referimos que a maioria dos controladores modernos são capazes de gerar um ou mais sinais PWM em alguns dos seus pins. No caso do Arduino UNO estes pins são 3, 5, 6, 9, 10 e 11. Quando utiliza qualquer um deles como outputs PWM tem de saber qual é a frequência, **F**, dos sinais que o Arduino gera. Observe a tabela 3:

Tabela 3

PIN Nº	FREQUÊNCIA, F	PERÍODO, T
5 e 6	**980 Hz**	**1.02 mS ou 1020 µS**
3, 9, 10 e 11	**490 Hz**	**2.04 mS ou 2040 µS**

B. PARA QUE SÃO UTILIZADOS?

Podemos controlar o tempo que permanecemos no nível “1” para cada período com sinais PWM; a isto chama-se ciclo de trabalho. Também podemos usá-los para controlar e ajustar a voltagem que fornecemos a detereminados periféricos de output, como globos de luz, luzes LED, um motor, um servidor entre outros.

Suponha que um sinal PWM como o mostrado na Figura 10 esteja conectado a uma luz LED. O ciclo de trabalho é de 50%, então o LED liga-se metade do tempo e desliga-se na outra metade. Acredite ou não, o LED só vai brilhar metade do brilho para cada período.

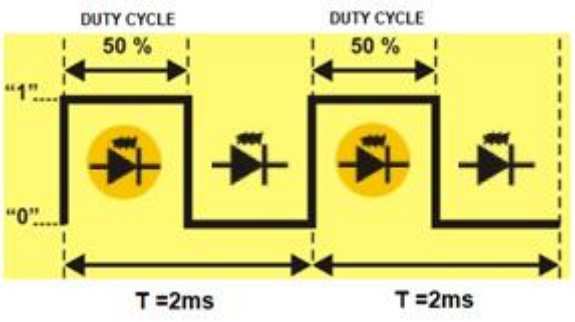


Figura 10

Pelas mesmas razões, se o ciclo de trabalho for 0%, o sinal permanecerá sempre no nível “0”. O LED não vai brilhar. Se o ciclo de trabalho é de 25%, o LED irá brilhar com um quarto do seu brilho e se o ciclo de trabalho for de 75%, brilhará em três quartos do seu brilho total. Se o ciclo de trabalho for 100%, o sinal permanecerá no nível “1” permanentemente. O LED vai brilhar com o brilho total. Isto dá-nos um intervalo entre 0% e 100% do sinal PWM para alcançar o brilho necessário.

Da mesma forma que pode ajustar o brilho de uma luz LED ou de uma lâmpada, também pode ajustar a velocidade de um motor ou a posição do eixo de um servidor. O princípio é o mesmo e poderá experimentá-lo em breve. Também deve estar ciente de que há um número considerável de periféricos que são controlados através de sinais PWM.

C. COMO SÃO GERADOS?

Os controladores capazes de gerar sinais PWM possuem uma série de circuitos eletronicos encorporados, como osciladores, temporizadores, comparadores, registradores entre outros. São, pois, bastante complexos. Ao sincronizar todos esses instrumentos e ao fazê-los trabalhar juntos, podemos gerar um ou mais sinais desse tipo. No entanto, precisa de muita perícia técnica para pôr tudo isto a funcionar.

Felizmente, o Arduino torna tudo realmente fácil. Esqueça os circuitos elétricos, como funcionam e como usá-los. Tudo o que precisa é de uma única função da linguagem de programação para gerar um sinal PWM; o Arduino cuida do resto.

- **A função analogWrite()**

  Esta função permite ajustar a extensão do ciclo de serviço de um sinal de saída PWM.

Sintaxe:

analogWrite(pin, value);

pin: pin a escrever. Refere-se ao pin que vai gerar o sinal PWM. Recorde-se que o nosso controlador Arduino UNO pode utilizar os pins número 3, 5, 6, 9, 10 e 11.

value: determina a extensão do ciclo de serviço. É um byte entre 0 e 255. O valor 0 representa 0% do ciclo de serviço, 127, um cioclo de serviço de 50%, e 255 corresponde a um sinal PWM com umm ciclo de serviço de 100%.

Exemplo:

```
byte Voltage = 25;                  //Aponta a % de voltage necessária
int Cycle= Voltage*255/100;         //Calcula a extensão de um ciclo de serviço entre 0 and 255
analogWrite(6,Cycle);               //Gera um sinal PWM de 25% no pin 6
```

Não é preciso configurar o pin que gera o sinal PWM como output; já está na função. O pin gera o sinal de output PWM indefinidamente até que a função analogWrite() seja executada novamente com um valor diferente, ou as funções digitalRead() ou digitalWrite() no mesmo pin. Em todos estes três casos, o output do sinal PWM termina.

7. MAIS FUNÇÕES NA LINGUAGEM ARDUINO

Mesmo que as funções que se seguem não tenham nada a ver com sinais de PWM, vamos vê-las de perto para que possa alargar os seus conhecimentos da própria linguagem.

Estão relacionadas com a capacidade de gerar números aleatoruamente, ou seja, números de qualquer valor dentro de determinados limites.

- **A função random()**

Esta função gera um número aleatório entre um mínimo e um máximo. Retorna um valor (long) inteiro sinalizado de 32 bits.

Sintaxe:

random(min, max);

min: estabelece e inclui o valor mínimo do número aleatório a ser Gerado; é opcional. Se o valor mínimo não for indicado, considera-se 0.

max: estabelece mas exclui o valor máximo do número aleatórfio a ser gerado.

Exemplo:

//Imitate the rolling of two dice

byte Die_1;

byte Die_2;

Die_1 = random(1,7); *//Gera um número aleatório entre 1 e 6*
Die_2 = random(1,7); *// Gera um número aleatório entre 1 e 6*

- **A função randomSeed()**

Este é outro sistema que o Arduino usa para gerar números aleatórios. Consiste numa sequência ou longa lista de números que se mantém constante. Pode começar a gerar os números aleatórios de qualquer lugar da lista ou sequência

Sintaxe:

randomSeed(value);

value: pode ser qualquer número inteiro de 16 bits (int) ou de 32 bits (long). É usado como uma "semente" para iniciar a sequência de números aleatórios em qualquer momento. Quanto mais aleatória for a "semente", mais aleatória será a sequência.

SECÇÃO PRÁTICA

8. EXEMPLO 1: Conversão ADC

Aqui está o primeiro exercício. Tudo o que precisa de fazer é ler o valor analógico usando o potenciometro ligado ao pin A0. A conversão efetua-se cada vez que pressionar o botão ligado ao pin 4. A ideia deste exercício é reforçar o conhecimento adquirido nas unidades anteriores para que a comunicação em série e uma variedade de formatos sejam usados para transferir o resultado da conversão para o PC.

O aspeto realmente novo deste exemplo está destacado na Figura 11. A função analogRead(A0) lê e converte o sinal analógico do input AO no seu equivalente no código binário e guarda-o na variável “AN0”.

```
while(digitalRead(4));
delay(20);                  //Esperar a recibir un pulso desde D4

AN0=analogRead(A0);         //Realiza la conversión de A0 (potenciom
V=AN0*0.00488;              //Calcula la tensión equivalente

Serial.print(AN0);          //Transmite la medida en decimal
Serial.print("\t");         //Tabulación
Serial.print(AN0,BIN);      //Transmite la medida en binario
```

Figura 4

Este valor é convertido em voltagem, multiplicando-o pela constante, 0.0048 (a resolução em bits) e é guardado na variável “V”. Por último, todos os resultados são transmitidos usando a comunicação em série nos formatos decimal, binário e hexadecimal e em voltagem. A Figura 12 apresenta várias conversões. Comece com o potenciometro posicionado à esquerda (0 V) e mova-o gradualmente até chegar à direita (5 V).

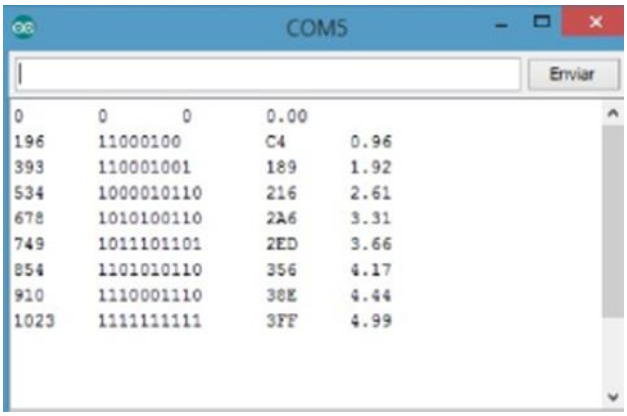


Figura 5

9. EXEMPLO 2: Limitações

Pode executar qualquer tipo de operação com o valor binário de uma conversão. Neste exercício vai fazer duas conversões: uma com o potenciometro ligado à entrada analógica A0 e outra com o potenciometro ligado à entrada analógica A1. Se o resultado da conversão na entrada A0 for superior a 4 V, a luz vermelha acende-se. Se o resultado na entrada A1 for superior a 2.5 V, a luz branca acende-se. Se os resultados forem menores do que as voltagens indicadas, as luzes LED permanecem apagadas.

Quando fizer o upload do programa, verifique se a luz vermelha acende quando gira o eixo do potenciometro quase todo para a direita; certifique-se também de que só tem que girar o eixo do potenciometro a meio caminho para a esquerda para que a luz branca se acenda.

Lembre-se que um potenciometro tem um comportamento semelhante ao de um joystick e que pode detectar em que posição se encontra.

10. EXEMPLO 3: Comparador analógico

Com base no exercício anterior, podemos criar programas que comparam duas voltagens ou sinais e identificar se um é maior do que o outro ou se são ambos iguais. É isso que pretemdemos que faça neste exercício.

Comparamos duas voltagens analógicas, V1 e V2, ligadas aos pins A0 e A1; ambos oriundos dos potenciometros. Os LEDs de output continuam como se apresenta na tabela 4.

Tabela 4

IF...	LED
V1 = V2	Ambar
V1 > V2	Vermelho
V1 < V2	Verde

11. EXEMPLO 4: Controlador do brilho

Já deve ter visto um sistema para controlar a intensidade da iluminação numa sala. Pode clarear ou escurecer as luzes e criar atmosferas diferentes na sala usando o controlo. E é exatamente isso que se pretende que faça neste exercício.

Usamos a voltagem analógica proveniente de um dos potenciometros e vamos ao pin A0 para gerar um sinal PWM que regula a voltagem que alimenta o LED branco ligado ao pin 6.

Dê uma olhada ao excerto do corpo principal do programa (Figura 13). É uma tarefa simples! A função analogRead(A0) lê o valor analógico no pin A0.

```
void loop()

  AN0=analogRead(A0);
  AN0=map(AN0,0,1023,0,255);

  analogWrite(6,AN0);
```

Figura 6

Como sabe, o resultado da conversão pode estar entre 0 e 1023. Usando a função map() pode arredondar esse valor para outro equivalente, entre 0 e 255, que é o parâmetro que a função analogWrite() precisa para gerar um sinal PWM no output 6 (o LED branco).

Carregue o programa e surpreenda a sua família e amigos. Pode escurecer ou intensificar a iluminação movendo o potenciometro da esquerda para a direita ou vice-versa.

12. EXEMPLO 5: Orientador eléctrico

Vamos usar os mesmos princípios do exercício anterior. Suponha apenas que está a navegar. Movendo o potenciometro, movimenta o eixo do servomotor que, por sua vez, move o leme do barco.

Se observar o programa de perto, verá que a função map() arredonda o resultado da conversão para um valor entre 0 e 180, o número máximo de graus que o eixo do servomotor pode girar.

13. EXEMPLO 6: O fotómetro

Vamos agora trabalhar com o sensor de luz e imitar um fotómetro ou instrumento de medição de luz ambiente. Sempre que pressionamos o botão ligado ao pin 4, a voltagem proveniente do sensor de luz ligado ao pin A2 é convertida no seu equivalente analógico.

O programa mede o valor analógico no input A2 com base na luz que atinge o sensor e, em seguida, calcula a voltagem. Os resultados são transmitidos usando comunicação em série.

Pode considerar este exercício como um programa experimental. Faça uma série de medidas. A primeira medição que fizer é com o sensor no escuro e a última na iluminação máxima.

Também pode procurar uma relação entre essas medidas e as fornecidas por um instrumento fabricado comercialmente. Com base no que descobrir, poderá criar o seu próprio instrumento para medir a luz em lux ou em lumens.

14. EXEMPLO 7: Controlo da iluminação

Este exercício não oferece nada especialmente novo, mas tem aplicações concretas, por exemplo, em sistemas de iluminação pública. É, de certo, usado para iluminar as ruas da sua cidade ou cidade.

Quando começa a escurecer, as luzes da rua acendem-se. Neste exercício, o sensor de luz ligado ao input analógico A2 mede a luz ambiente. Certamente haverá um sensor semelhante na sua rua. Vai encontrá-lo algures fora do caminho, mas em local seguro. O LED ligado ao pin 6 simula a luz da rua.

É interessante notar que mais uma vez uma função foi criada: Measure_light(). Esta nova função reune uma série de amostras de luz ambiente em intervalos regulares e, em seguida, calcula e retorna a média de todas elas.

Este sistema é usado para "filtrar" o sinal analógico vindo do sensor de luz, evitando-se assim variações muito pequenas que fazem as luzes da rua ligarem-se e desligarem-se sem motivo.

O programa principal chama a função **Measure_light()** e compara a média com o valor mínimo estabelecido para ligar o LED branco.

- **Agora é a sua vez!**

Analise este exemplo: é de noite e as luzes da rua estão acesas. O que acontece se houver uma tempestade e um raio atingir o sensor? As luzes da rua provavelmente apagam, pois o nosso programa vai pensar que é hora do dia. Vão voltar a acender-se depois, é claro. Este é um exemplo real; as coisas acontecem mesmo assim. Devemos encontrar uma maneira de evitar isso? Encontre uma solução que evite esta situação e que melhore o programa.

15. EXEMPLO 8: Medir reflexos

Este exercício é muito simples e não oferece nada especialmente novo. A ideia é medir a luz refletida detectada pelo sensor de infravermelho (IR) ligado ao input analógico A3. O resultado é transmitido usando comunicação em série.

- **Agora é a sua vez!**

Quando gravar o programa, abra também o monitor da porta *serial* para poder ver as diferentes medições. Como o conversor tem uma resolução de 10 bits, os valores estarão entre 0 e 1023 (2^{10}). Aqui estão algumas aspetos que deve verificar:

- ✓ Se não colocar nada na frente do sensor, perceberá que a leitura é muito baixa. A luz infravermelha emitida dispersa-se, não tem retorno, e o sensor detecta apenas uma pequena quantidade da luz refletida.

- ✓ Se colocar alguma coisa de cor brilhante a cerca de 10mm do sensor, vai verificar que a leitura aumenta consideravelmente.

As cores brilhantes refletem-se melhor do que a luz infravermelha; viram-se e voltam mais para o sensor.

- ✓ Faça a mesma experiência, mas desta vez com um objeto mais escuro. A leitura no sensor desce. As cores mais escuras absorvem mais a luz infravermelha e o sensor recebe uma quantidade menor de luz refletida.

- ✓ Também pode experimentar mover o objeto para mais próximo do sensor ou mais longe. Vai aperceber-se que a leitura muda quando aproxima o objeto do sensor ou o afasta. Quanto mais próximo o objeto estiver do sensor, maior será a quantidade de luz refletida que o sensor recebe.

16. EXEMPLO 9: Detetar cores

Este é um exercício meramente experimental e existem maneiras de o melhorar. É essencialmente sobre como detectar a cor de um objeto. Utilize os valores da tabela no exercício anterior como referência. Com este exercício, vamos tentar distinguir entre preto e branco.

O programa aguarda que pressione o botão conectado ao pin 12. De seguida, o sensor de infravermelhos ligado à entrada analógica A3 realiza a medição. Se for inferior a 300, transmite a mensagem “BLACK”. Se o valor for igual ou superior a 300, transmite a mensagem “WHITE”. Estas mensagens são transmitidas usando comunicação em série.

Pode criar um objeto preto e branco com um simples pedaço de papelão. Coloque-o à frente do sensor a uma distância de aproximadamente 15 mm e pressione D12 para fazer a medição. A cor do objeto aparecerá no monitor da porta de série.

17. EXEMPLO 10: Sensores de temperatura

Este exercício é semelhante aos anteriores. Aqui, a ideia é medir e mostrar a temperature ambiente detetada pelo sensor.

A medição é realizada quando se pressiona o botão ligado ao pin D4. O resultado da conversão (AN4), o seu equivalente em voltagem (V = AN4 * 0.0048) e a temperatura ambiente em ºC (T = AN4 * 500 / 1024) são transmitidas utilizando comunicação em série.

18. EXEMPLO 11: Aparelhos de ar condicionado

Aqui está outro exemplo com uma aplicação prática óbvia. Vamos simular um sistema simples de ar condicionado.

Quando a temperatura excede um valor estabelecido na variável “Max”, a luz LED verde no pin 9 acende; isto simula o início de funcionamento do sistema de refrigeração. Se a temperatura ambiente cair abaixo de um certo valor estabelecido na variável “Min”, a luz LED vermelha do pin 11 acende; isto simula o início do sistema de aquecimento. Se o sensor detectar uma temperatura entre o "Max" e o "Min", os sistemas de refrigeração e de aquecimento desligam-se. Presume-se que esta seja a zona de conforto de temperatura.

19. EXEMPLO 12: Um sinal PWM

Este é um exercício muito simples e uma boa oportunidade para usar a função analogWrite(). A ideia é gerar um sinal PWM no pin 6, conectado ao LED branco na placa de desenvolvimento.

Tem de indicar a percentagem do ciclo de trabalho ou a tensão necessária na variável “Power”. A variável "Ciclo" calcula o valor do ciclo de trabalho equivalente, entre 0 e 255.

A função setup() está vazia, mas é obrigatório incluí-la.

O corpo principal do programa está na função loop(). Ele gera um sinal PWM com o ciclo de serviço que solicitou no pin 6 usando analogWrite();.

Aqui está um detalhe importante: observe com atenção a função while(1);. Esses loops continuam infinitamente até que a expressão dentro do parênteses () se torne falsa. Algo deve alterar a variável testada ou o loop while nunca termina. Forma um loop infinito. Isto significa que o controlador não executa nenhuma outra função.

Mas o sinal PWM continua sempre no pin 6. Porquê? Porque os circuitos eletrônicos internos do controlador geram os sinais PWM nos pins 3, 5, 6, 9, 10 e 11. Uma vez que a função analogWrite() começa a gerá-los, eles continuam infinitamente até ao momento em que dê uma ordem contrária. Referimo-nos a estes sinais como sinais gerados pelo “Hardware”.

20. EXEMPLO 13: Efeitos óticos

Este exemplo demonstrará claramente o efeito ótico criado com o brilho de uma luz LED através de um sinal PWM com um ciclo de trabalho que aumenta e diminui gradualmente.

O primeiro loop for() aumenta o ciclo de trabalho de 0 a 255 em etapas de 5. O brilho do LED aumentará de mínimo para o máximo.

O segundo loop for() diminui o ciclo de trabalho de 255 para 0 etapas internas de 5 também. O brilho da luz Led diminuirá do máximo para o mínimo.

21. EXEMPLO 14: Regulação manual

Este é realmente um excelente exercício prático. Vai regular manualmente o brilho do LED branco ligado ao pin 6 e vai tornar a luz LED mais brilhante aumentando o ciclo de sinal de PWM em múltiplos de cinco, usando o botão 4. Pode desvanecê-lo utizando o botão botão 7 para reduzir a extensão do ciclo de trabalho, mais uma vez por múltiplos de cinco.

Recorde-se que pode aplicar exatamente os mesmos princípios de regulação a um motor elétrico, por exemplo; poderá controlar a velocidade de giro usando o mesmo princípio.

22. EXEMPLO 15: Luzes aleatórias

Este exercício é uma novidade. Talvez o ajude a decorar a sua árvore de Natal, montra, quarto, jardim ou o que quiser. Vamos gerar quatro sinais PWM simultâneos nos pins 6, 9, 10 e 11.

A função random() que estudamos nesta unidade irá tornar aleatório o comprimento de cada ciclo de trabalho do sinal PWM. O que, por sua vez, torna aleatória a voltagem recebida pelas luzes Led.

REFERÊNCIAS

LIVROS

[1]. EXPLORING ARDUINO, Jeremy Blum, Ed.Willey
[2]. Practical ARDUINO, Jonathan Oxer & Hugh Blemings, Ed.Technology in action
[3]. Programing Arduino, Next Steps, Simon Monk
[4]. Sensores y actuadores, Aplicaciones con ARDUINO, Leonel G.Corona Ramirez, Griselda S. Abarca Jiménez, Jesús Mares Carreño, Ed.Patria
[5]. ARDUINO: Curso práctico de formación, Oscar Torrente Artero, Ed.RC libros
[6]. 30 ARDUINO PROJECTS FOR THE EVIL GENIUS, Simon Monk, Ed. TAB
[7]. Beginning C for ARDUINO, Jack Purdum, Ph.D., Ed.Technology in action
[8]. ARDUINO programing notebook, Brian W.Evans

SÍTIOS WEB

[1]. https://www.arduino.cc/
[2]. https://www.prometec.net/
[3]. http://blog.bricogeek.com/
[4]. https://aprendiendoarduino.wordpress.com/
[5]. https://www.sparkfun.com