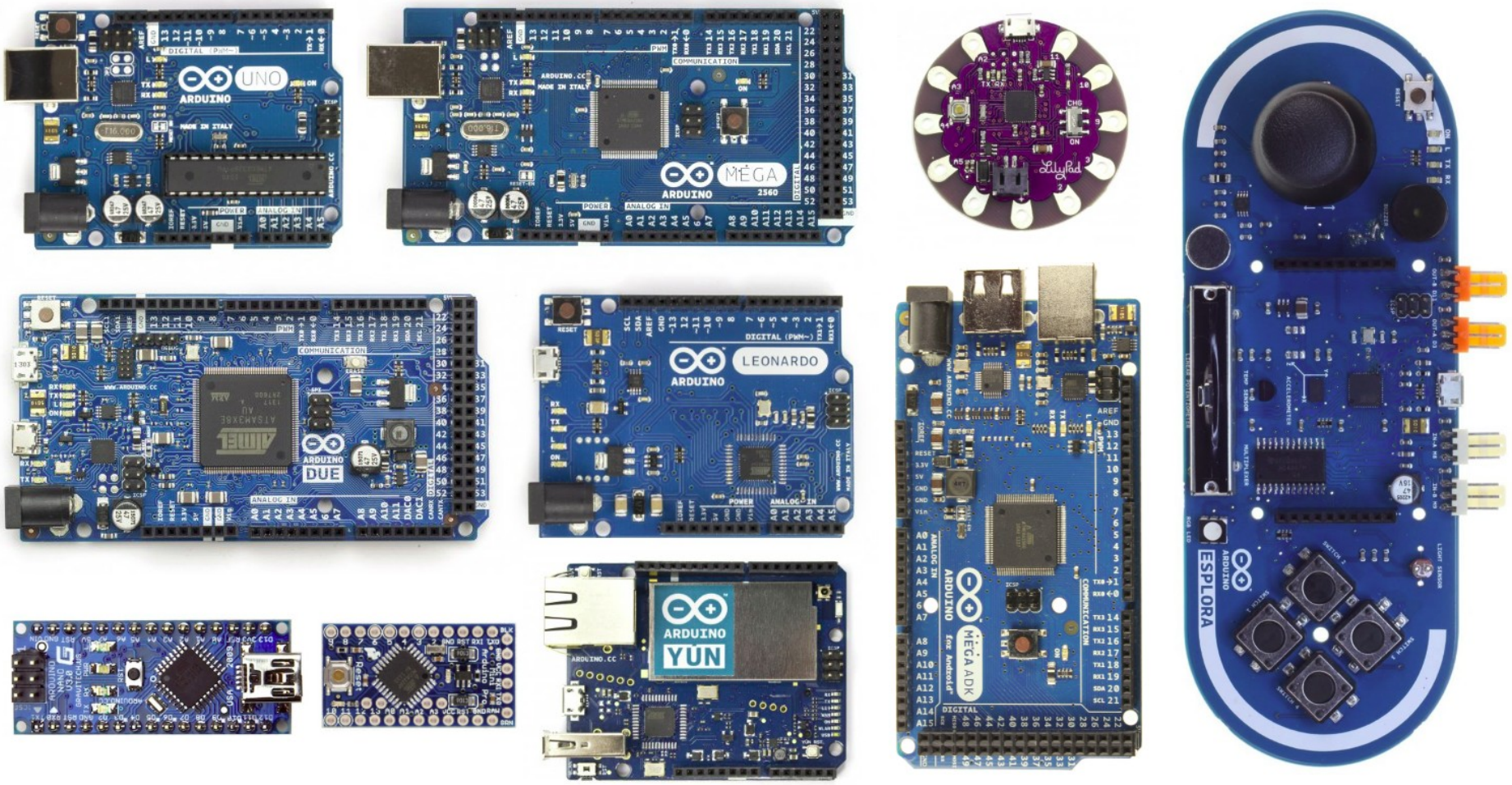
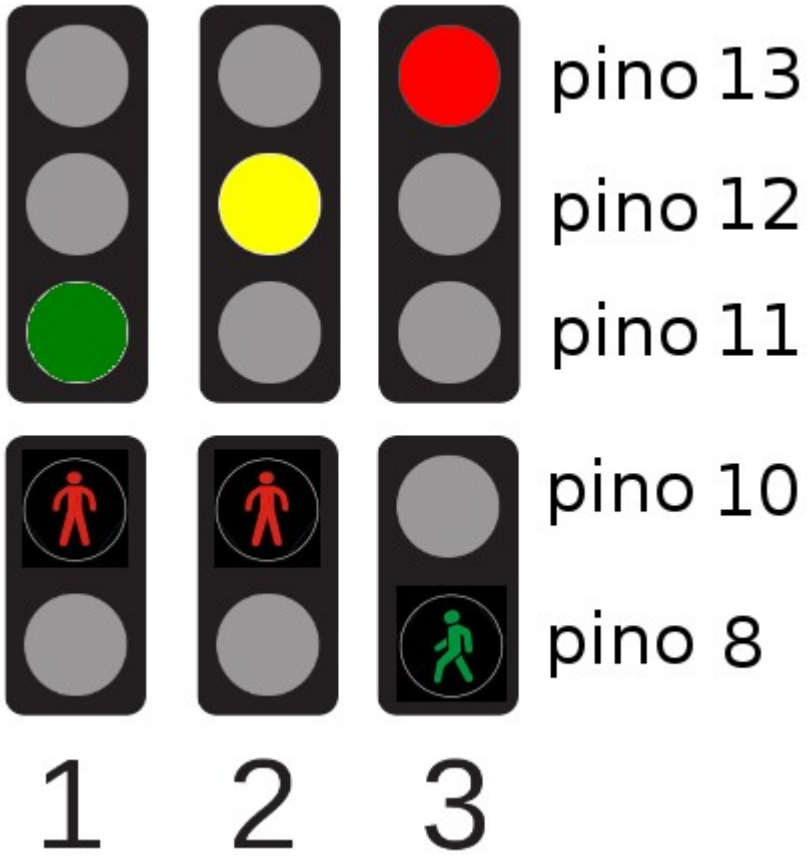
Curso Arduíno – Aula 2



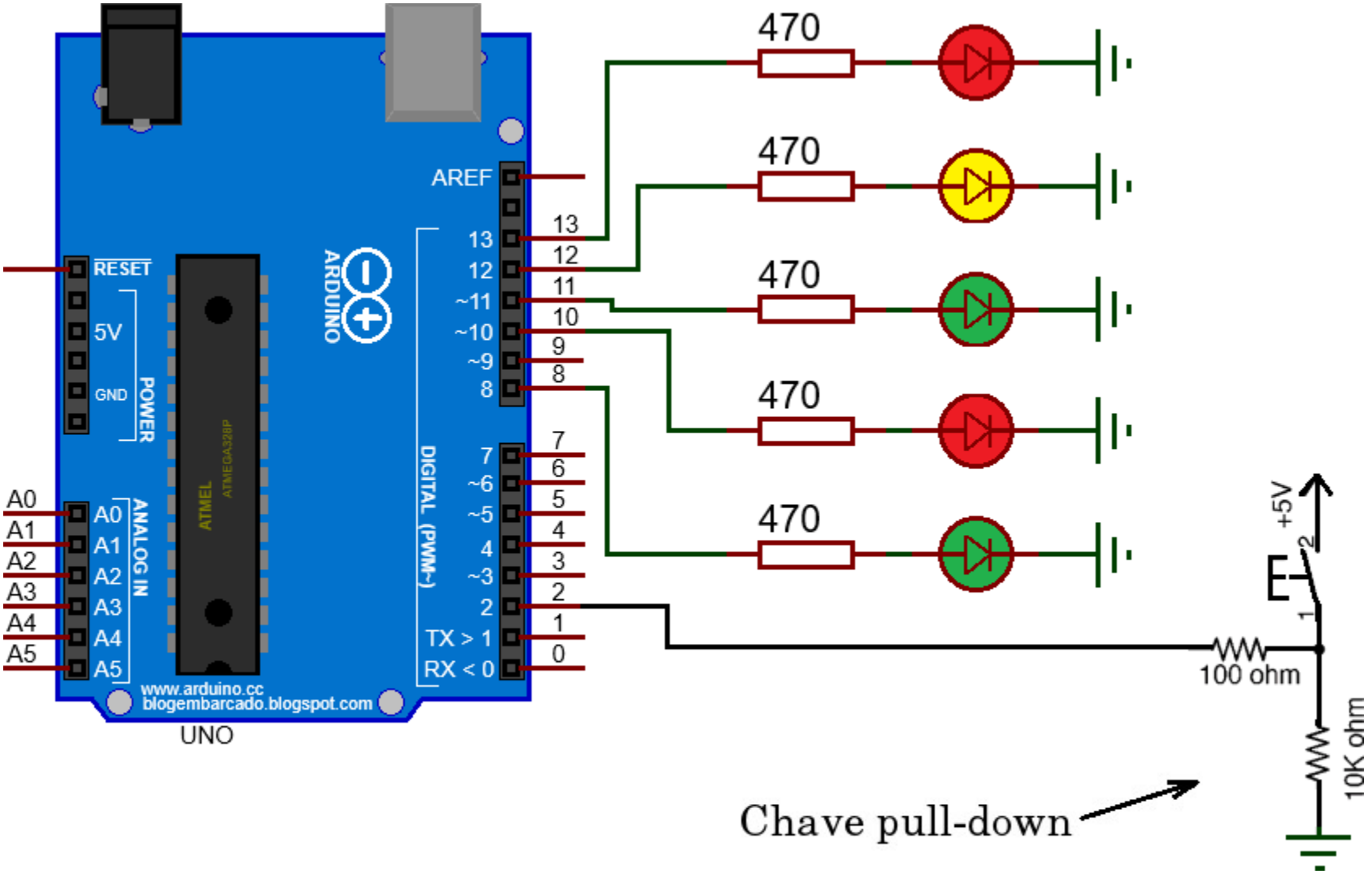
Prática 4 – Semáforo com botão

Crie um programa que quando um pedestre for atravessar a rua, ele tenha que apertar um botão e o semáforo mude de estado, permitindo que o pedestre possa seguir e os carros parem.

Ao lado a sequência das luzes dos semáforos.

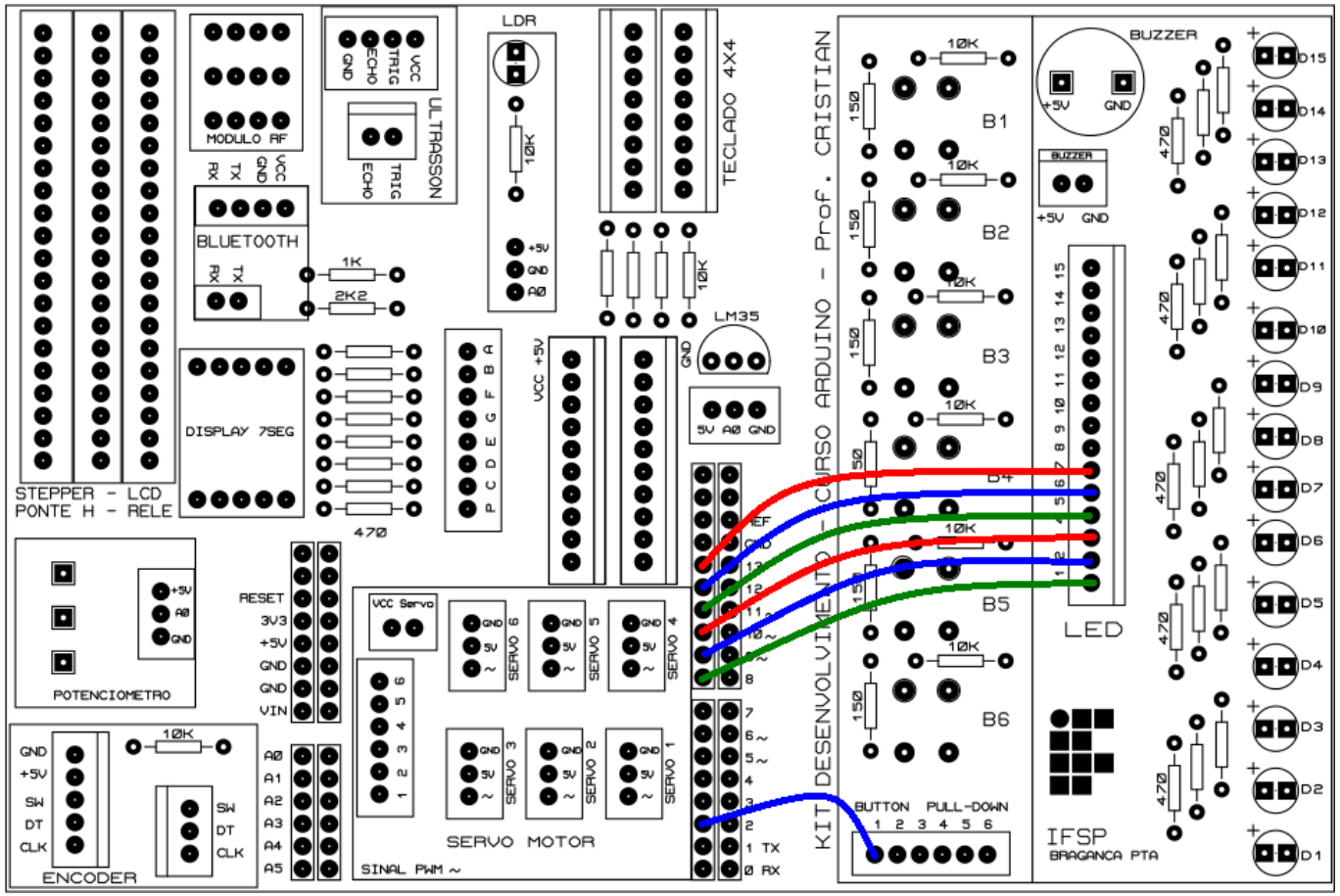


Prática 4 – Semáforo com botão



Prática 4 – Semáforo com botão

A ligação do circuito deve ser realizada como visto a seguir:



Prática 4 – Semáforo com botão

```
int carRed = 13;    // atribuiu pino 13 a palavra carRed
int carYellow = 12;// atribuiu pino 12 a palavra carYellow
int carGreen = 11; // atribuiu pino 11 a palavra carGreen
int pedRed = 10;    // atribuiu pino 10 a palavra pedRed
int pedGreen = 8;  // atribuiu pino 8 a palavra pedGreen
int button = 2;     // atribuiu pino 2 a palavra button

void setup() {
  pinMode(carRed, OUTPUT);    // declara como variavel de saida
  pinMode(carYellow, OUTPUT);// declara como variavel de saida
  pinMode(carGreen, OUTPUT); // declara como variavel de saida
  pinMode(pedRed, OUTPUT);    // declara como variavel de saida
  pinMode(pedGreen, OUTPUT); // declara como variavel de saida
  pinMode(button, INPUT);     // declara como variavel de entrada
}
```

Prática 4 – Semáforo com botão

```
void loop() {
  digitalWrite(carGreen, HIGH); // acende a luz verde para carro
  digitalWrite(pedRed, HIGH);   // acende a luz vermelha para pedestre

  int state = digitalRead(button); // verifica se o botão foi pressionado
                                   // e armazena resultado na variavel state
  if (state == HIGH) {             // Se state igual a nivel ALTO (1) continue;
    delay(1000);
    digitalWrite(carGreen, LOW);   // apaga o verde para carro
    digitalWrite(carYellow, HIGH);// acende o amarelo para carro
    delay(2000);                   // espera 2 segundos
    digitalWrite(carYellow, LOW); // apaga o amarelo para carro
    digitalWrite(carRed, HIGH);    // acende o vermelho para carro
    digitalWrite(pedRed, LOW);     // apaga o vermelho para pedestres
    digitalWrite(pedGreen, HIGH); // acende o verde para pedestres
    delay(3000);                   // tempo para que os pedestres atravessem 3s
    digitalWrite(pedGreen, LOW);   // acende o vermelho para pedestres
    digitalWrite(carRed, LOW);     // apaga o vermelho para carro
  }
}
```

Função **void**

A função **void()**, é usada para criar uma rotina a parte no programa, onde não retorna nenhuma informação, apenas volta no mesmo ponto de onde saiu do programa principal.

Veja o exemplo com o programa “ Semáforo com botão” onde é criado a função

void semaforo_acionamento()

Essa função é bastante empregada em partes do programa onde é necessário repetir funções iguais.

Prática 4 – Semáforo com botão

```
void loop() {
  digitalWrite(carGreen, HIGH); // acende a luz verde para carro
  digitalWrite(pedRed, HIGH);   // acende a luz vermelha para pedestre

  int state = digitalRead(button); // verifica se o botão foi pressionado
                                   // e armazena resultado na variavel state
  if (state == HIGH){              // Se state igual a nivel ALTO (1) continue
    delay(1000);
    semaforo_acionado(); // chama a funçao void semafor_acioando
  }                      //  e depois retorna no mesmo ponto
}

void semaforo_acionado(){
    digitalWrite(carGreen, LOW);  // apaga o verde para carro
    digitalWrite(carYellow, HIGH);// acende o amarelo para carro
    delay(2000);                  // espera 2 segundos
    digitalWrite(carYellow, LOW); // apaga o amarelo para carro
    digitalWrite(carRed, HIGH);   // acende o vermelho para carro
    digitalWrite(pedRed, LOW);    // apaga o vermelho para pedestres
    digitalWrite(pedGreen, HIGH); // acende o verde para pedestres
    delay(3000);                  // tempo para que os pedestres atravessem 3s
    digitalWrite(pedGreen, LOW);  // acende o vermelho para pedestres
    digitalWrite(carRed, LOW);    // apaga o vermelho para carro
}
```

Comando **for**

A instrução **for** é usada para repetir um bloco.

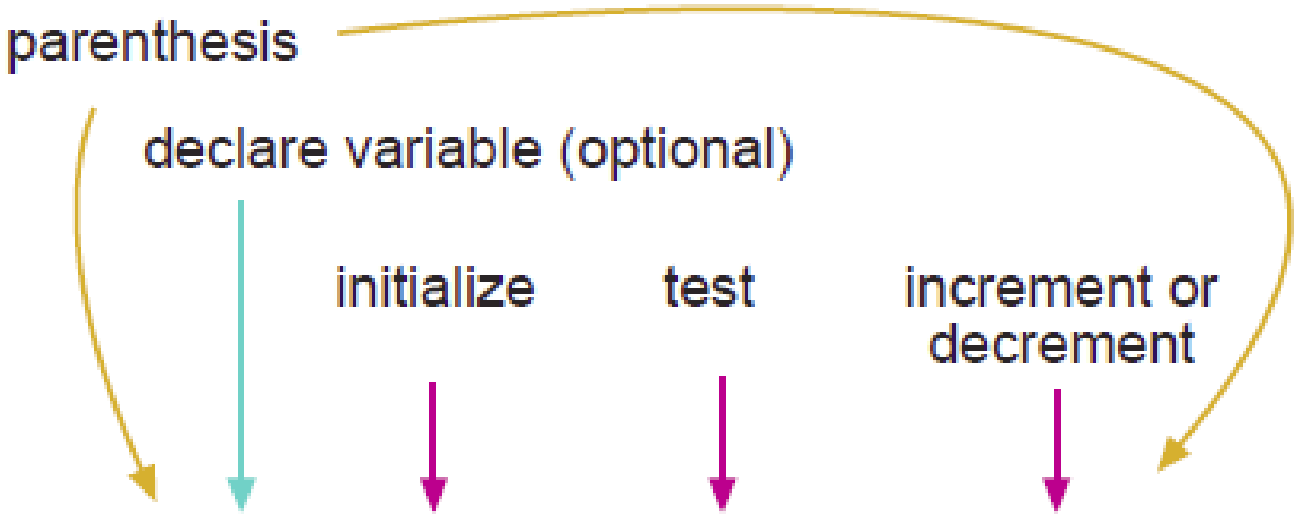
Um contador de incremento geralmente é usado para incrementar e encerrar o loop.

A instrução **for** é útil para qualquer operação repetitiva:

```
for (inicialização; condição; incremento) {

}
```

Comando **for**

Exemplo:




```
for(int x = 0; x < 100; x++){

   println(x);   // prints 0 to 99
}
```

Fonte: https://www.arduino.cc/en/Reference/For

Prática 5 – Semáforo com botão e pisca LED verde do pedestre

```
void semaforo_acionado(){
    digitalWrite(carGreen, LOW);   // apaga o verde para carro
    digitalWrite(carYellow, HIGH);// acende o amarelo para carro
    delay(2000);                   // espera 2 segundos
    digitalWrite(carYellow, LOW); // apaga o amarelo para carro
    digitalWrite(carRed, HIGH);    // acende o vermelho para carro
    digitalWrite(pedRed, LOW);     // apaga o vermelho para pedestres
    digitalWrite(pedGreen, HIGH); // acende o verde para pedestres
    delay(3000);                   // tempo para que os pedestres atravessem 3s

      for (int x=0; x<10; x++) {       // pisca o verde dos pedestres
        digitalWrite(pedGreen, HIGH);
        delay(250);
        digitalWrite(pedGreen, LOW);
        delay(250);
      }

    digitalWrite(pedGreen, LOW);   // acende o vermelho para pedestres
    digitalWrite(carRed, LOW);     // apaga o vermelho para carro
}
```

Criando uma Matriz (**Array)**

Em **pinos [ ]** declaramos uma matriz sem escolher explicitamente um tamanho. O compilador conta os elementos e cria uma matriz do tamanho apropriado.

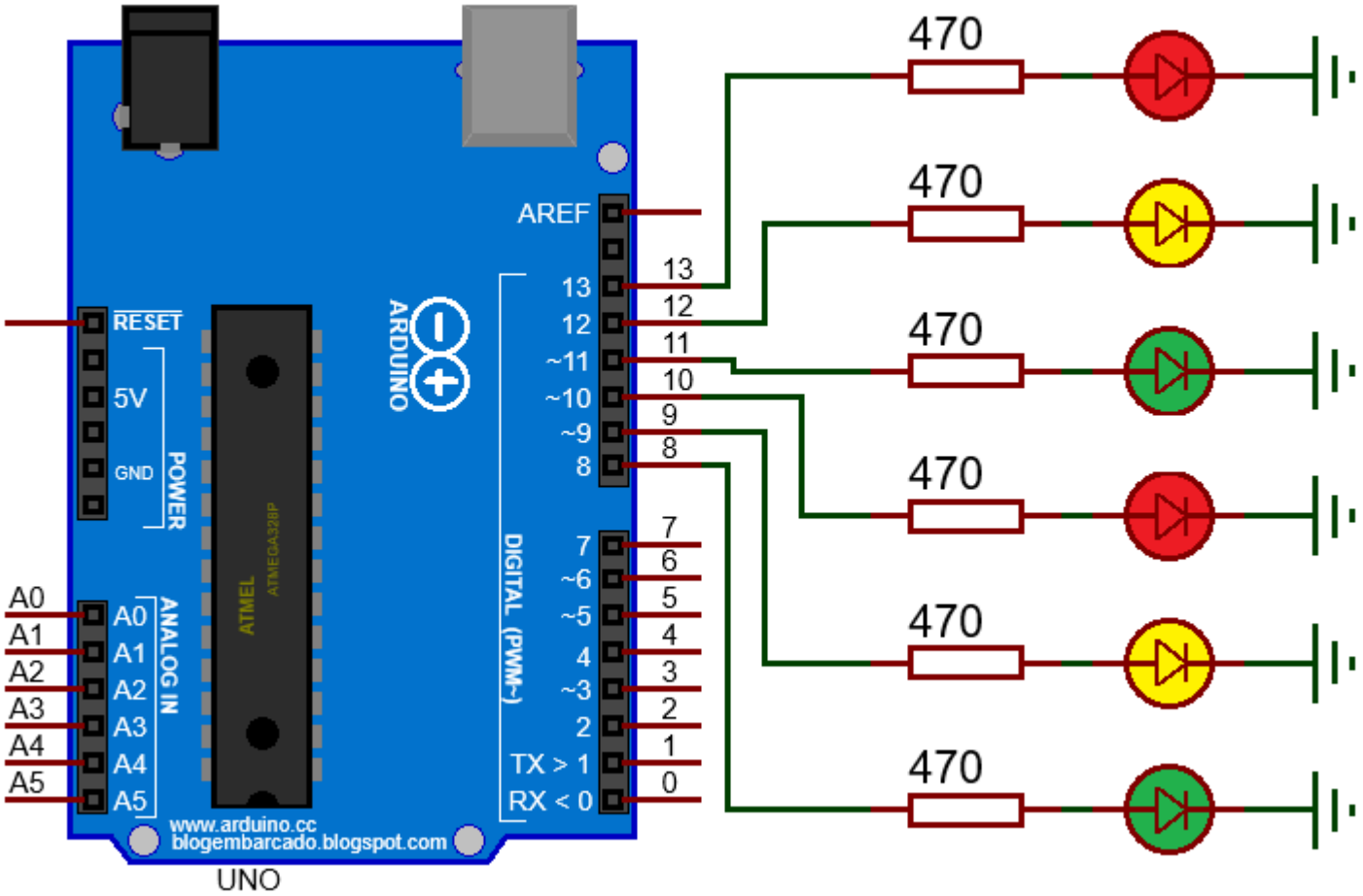
```
int pinos[] = {2, 4, 8, 3, 6};
```

Nas matrizes o primeiro elemento da matriz está no índice 0, portanto,

```
pinos [0] == 2; , pinos [1] == 4;
```

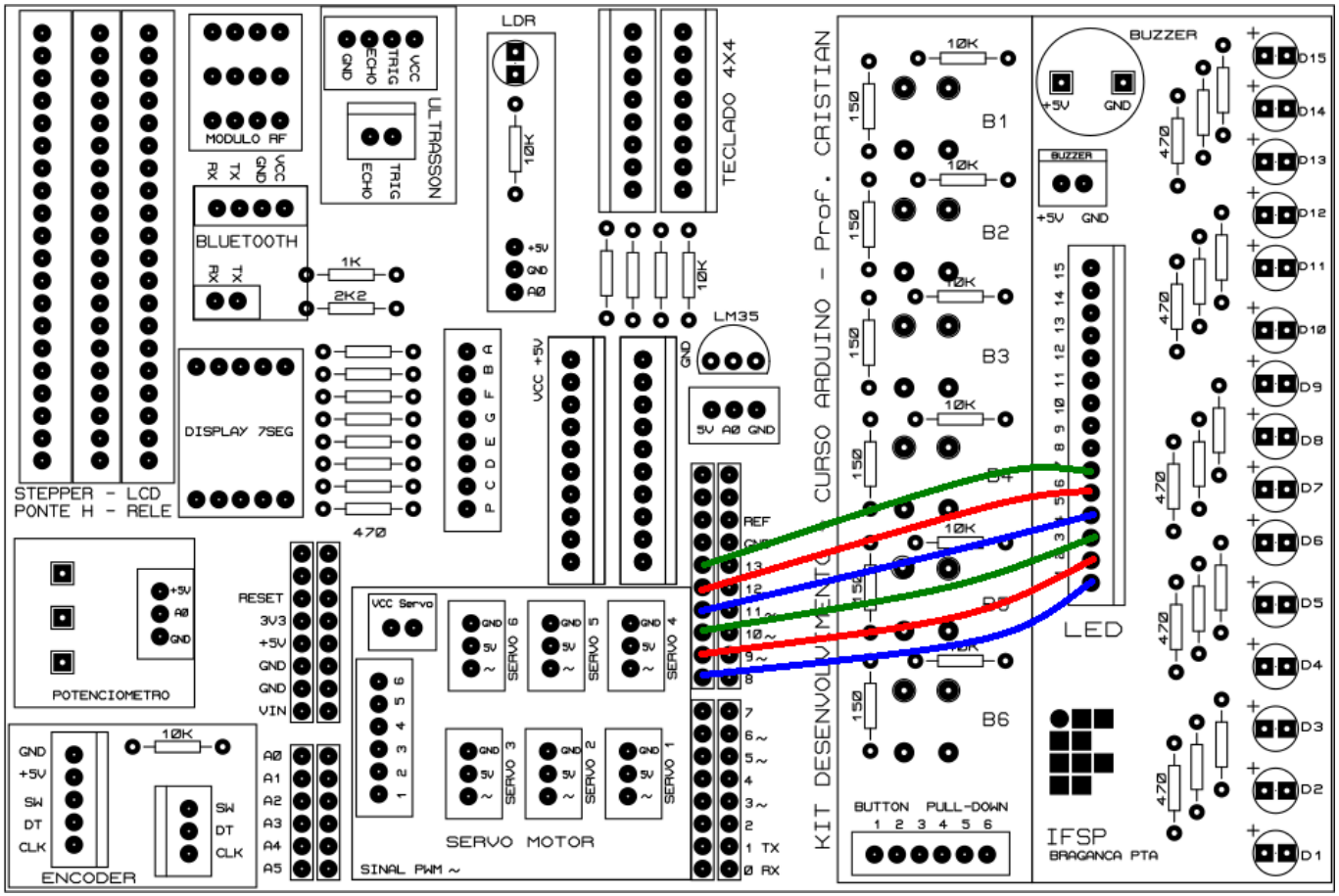
e assim por diante.

Prática 6 – Sequencial de LEDs



Prática 6 – Sequencial de LEDs

A ligação do circuito deve ser realizada como visto a seguir:



Prática 6 – Sequencial de LEDs

Programa A

```
int leds[]={8,9,10,11,12,13};

void setup(){

  for(int i=0; i<6; i++){
    pinMode(leds[i],OUTPUT);
  }
}
```

Programa B

```
int led0 = 8;
int led1 = 9;
int led2 = 10;
int led3 = 11;
int led4 = 12;
int led5 = 13;

void setup(){

    pinMode(led0,OUTPUT);
    pinMode(led1,OUTPUT);
    pinMode(led2,OUTPUT);
    pinMode(led3,OUTPUT);
    pinMode(led4,OUTPUT);
    pinMode(led5,OUTPUT);
}
```

Prática 6 – Sequencial de LEDs

Programa A

```
void loop(){

  for(int i=0; i<6; i++){
    digitalWrite(leds[i], HIGH);
    delay(500);
  }
```

Programa B

```
void loop(){

  digitalWrite(led0,HIGH);
  delay(500);
  digitalWrite(led1,HIGH);
  delay(500);
  digitalWrite(led2,HIGH);
  delay(500);
  digitalWrite(led3,HIGH);
  delay(500);
  digitalWrite(led4,HIGH);
  delay(500);
  digitalWrite(led5,HIGH);
  delay(500);
```

Prática 6 – Sequencial de LEDs

Programa A

```
  for(int i=0; i<6; i++){
    digitalWrite(leds[i], LOW);
    delay(500);
  }
}
```

Programa B

```
  digitalWrite(led0,LOW);
  delay(500);
  digitalWrite(led1,LOW);
  delay(500);
  digitalWrite(led2,LOW);
  delay(500);
  digitalWrite(led3,LOW);
  delay(500);
  digitalWrite(led4,LOW);
  delay(500);
  digitalWrite(led5,LOW);
  delay(500);
}
```

Prática 6 – Sequencial de LEDs

Como visto o programa A, utilizando o conceito de matrizes (Array) e a função **for** , foi possível criar um programa com menor número de linhas.

```
int leds[]={8,9,10,11,12,13};

void setup(){

  for(int i=0; i<6; i++){
    pinMode(leds[i],OUTPUT);
  }
}

void loop(){

  for(int i=0; i<6; i++){
    digitalWrite(leds[i], HIGH);
    delay(500);
  }

  for(int i=0; i<6; i++){
    digitalWrite(leds[i], LOW);
    delay(500);
  }
}
```

Entrada Analógica do Arduíno

Como foi mencionado no início do curso, o Arduíno possui um microcontrolador Atmega328 de 8 bit, isso significa que os dados internos podem variar de 00000000 até 11111111.

A entrada analógica recebe sinais que pode variar entre 0V e 5V, como o Arduíno só trabalha com números binários, é necessário que esse valores analógicos sejam convertidos em informações que o Arduíno consiga entender.

Definições e Conceitos

Números binários são representados apenas pelos algarismos 0 e 1 e é denominado base 2.

Fazendo um paralelo ao número decimal que mais utilizamos, o decimal é representado pelos algarismos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9, e é denominado base 10.

Mas os microcontroladores e computadores só entendem números binário, 0 ou 1, por isso precisamos aprender a converter esses números de forma a interpretá-los.

Definições e Conceitos

Sequência binária e seus pesos

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
1	1	1	1	1	1	1	1

→ Potencia na base 2

→ Peso correspondente

→ Sequencia Binaria de 8 Bits

Exemplo de conversao Binaria para Decimal

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
0	0	1	1	0	0	1	1

1+2+16+32=51 ← **Valor Decimal**

Numero Binario 110011 = 51 em Decimal

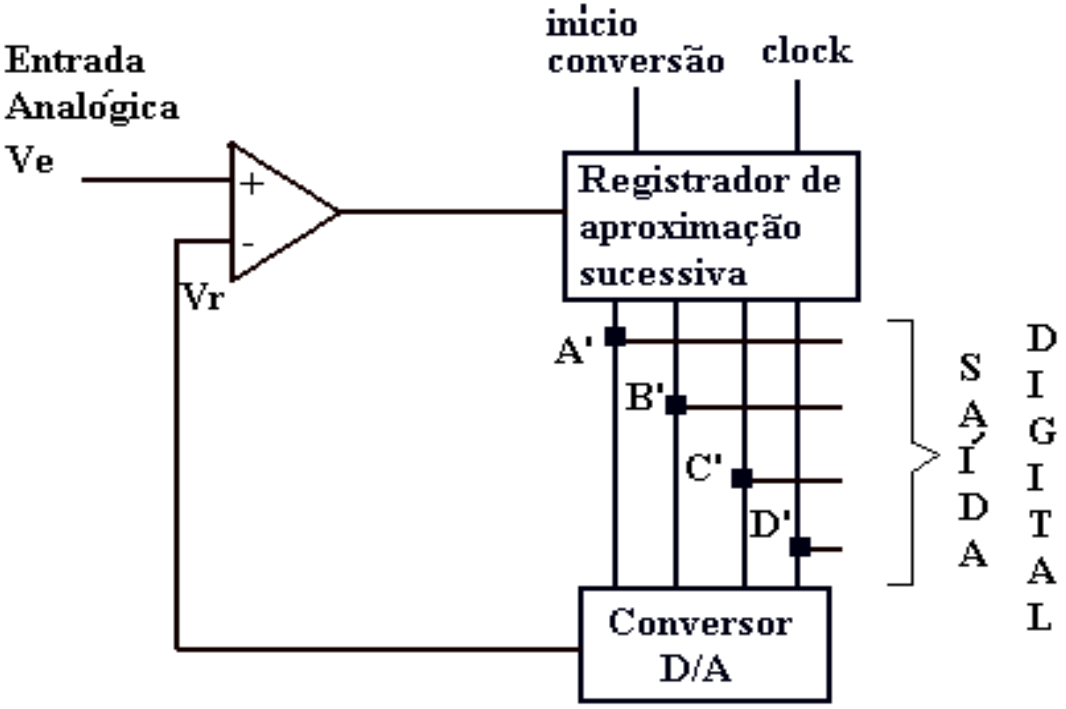
Definições e Conceitos

Desta forma, sabendo como interpretar os números binários através dos pesos, de acordo com sua posição na sequencia binária, podemos facilmente interpretar os números binários que os computadores e microcontroladores utilizam internamente.

Decimal	Binário
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

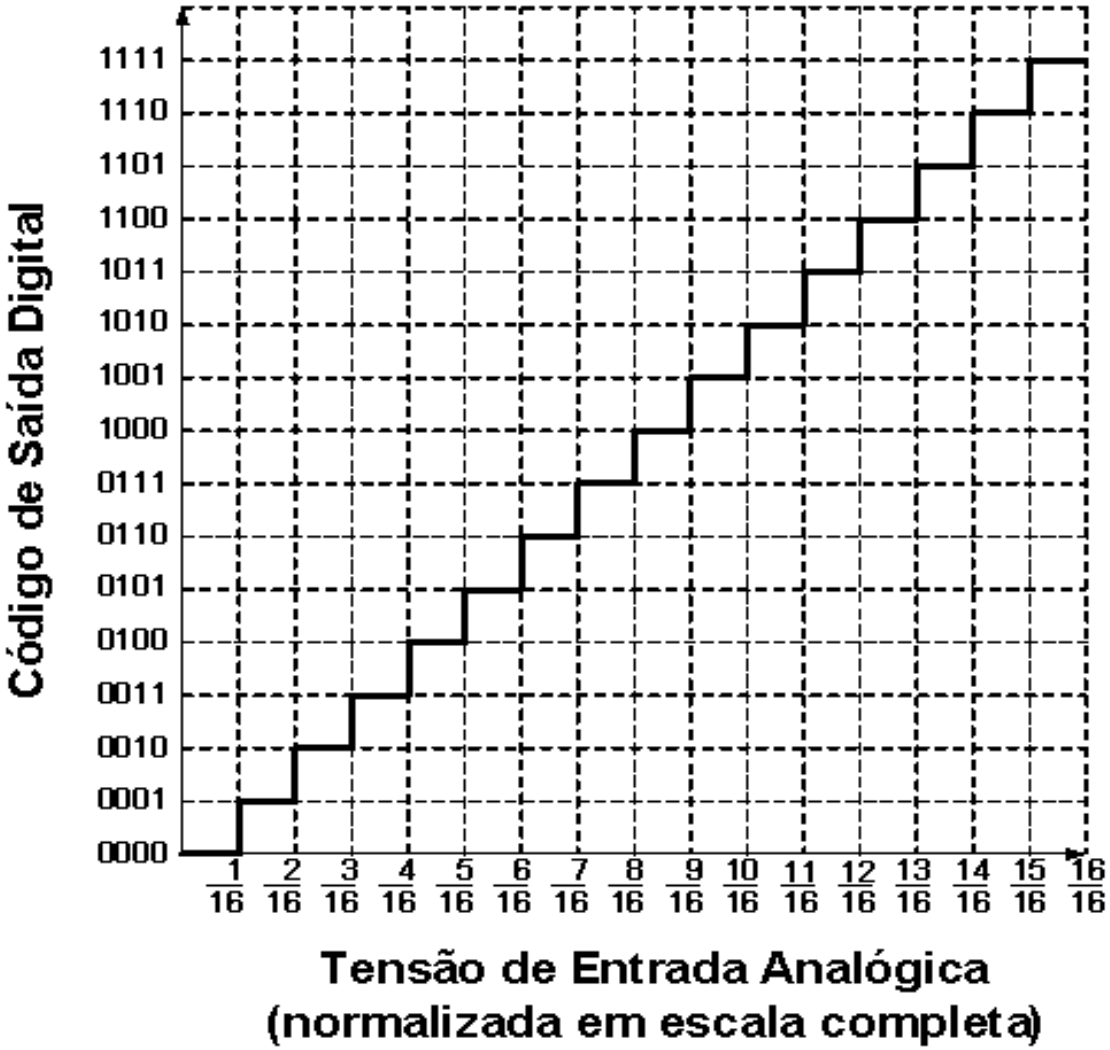
Entrada Analógica do Arduíno

Quando aplicamos um sinal entre 0V e 5V na entrada analógica, esse sinal passa por um conversor Analógico/Digital (ADC).



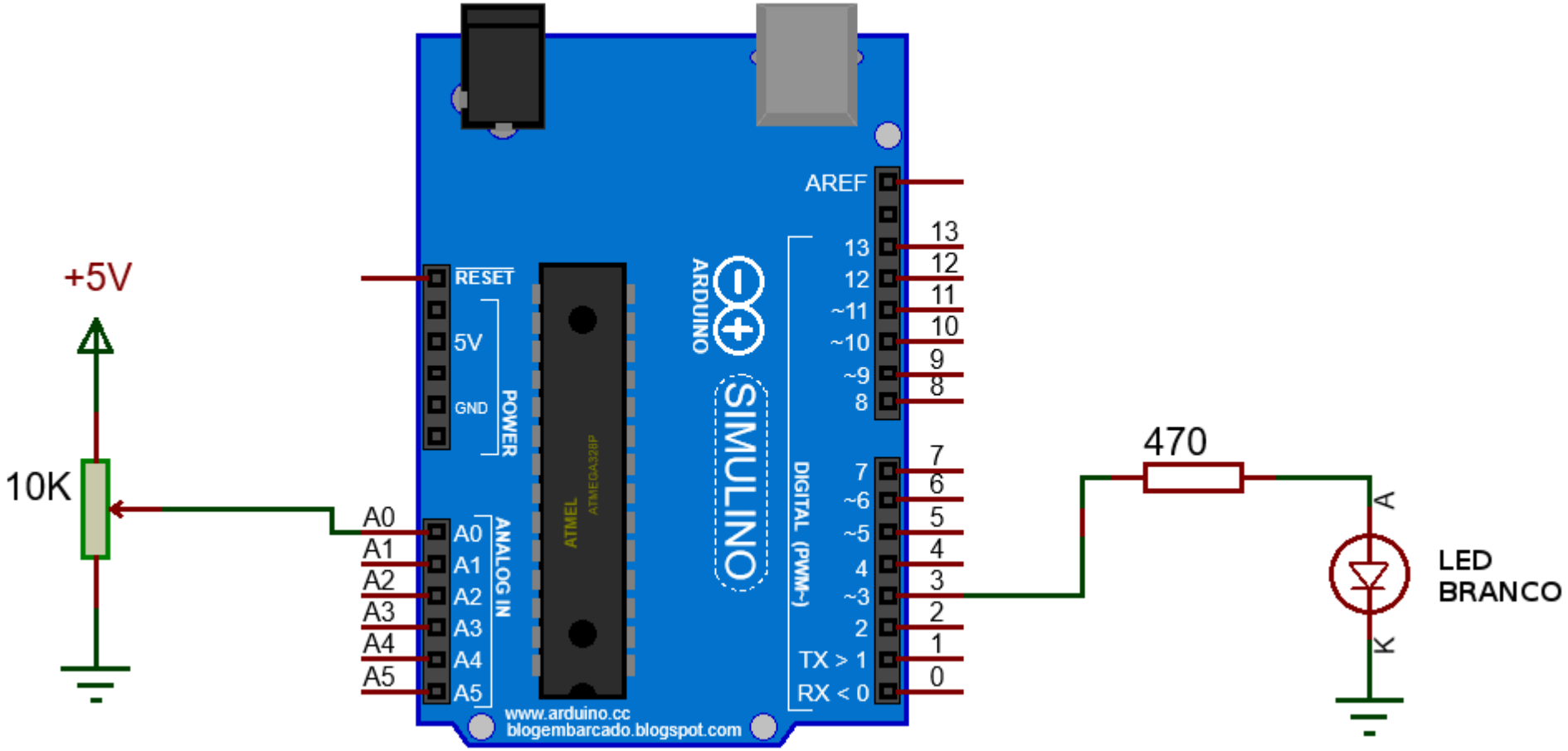
Fonte: http://endigital.orgfree.com/sequencial/conversorAD.htm

Entrada Analógica do Arduíno

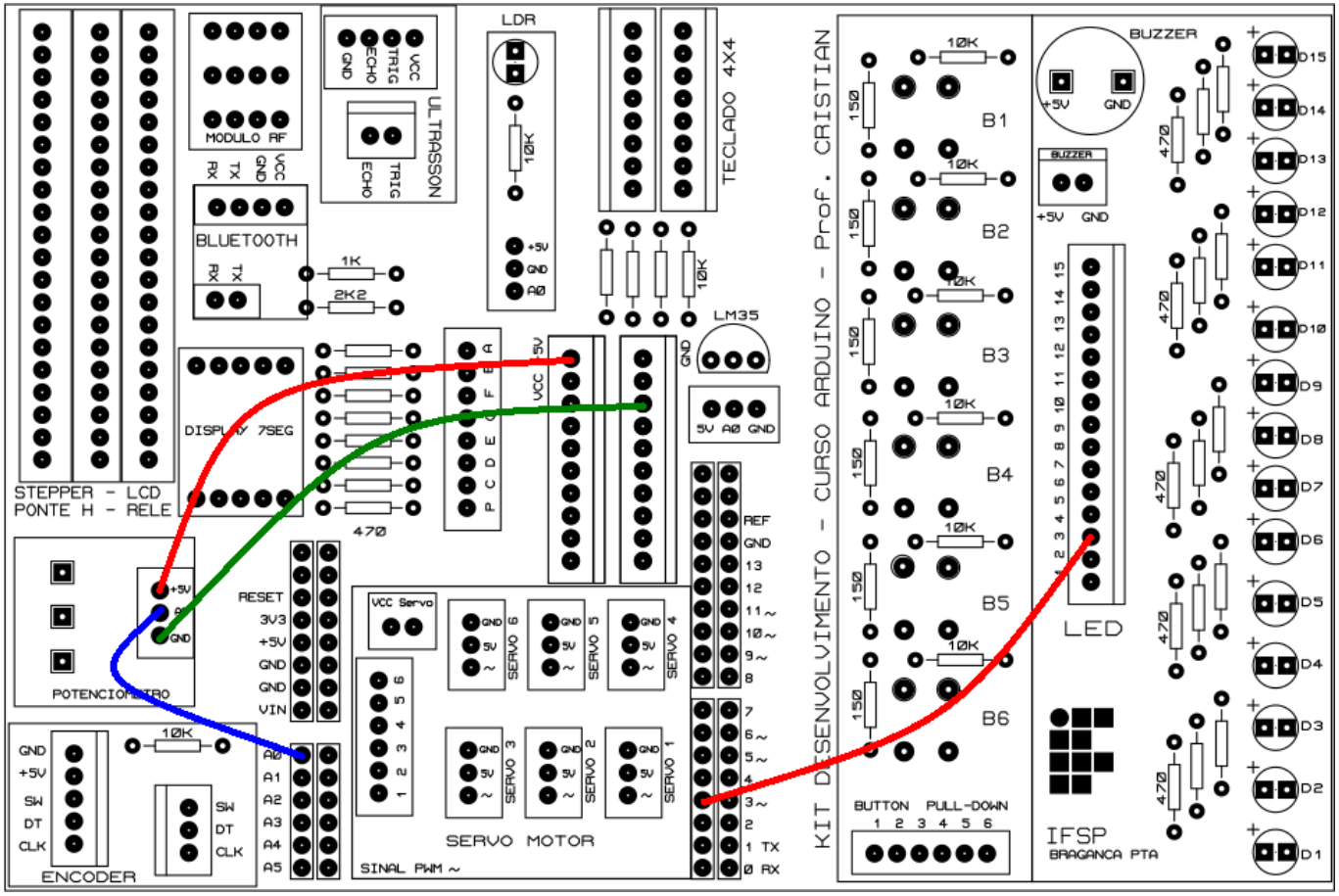


Fonte:http://macao.communications.museum/por/Exhibition/secondfloor/moreinfo/ADConvert

Prática 7 – Controle de Luminosidade do LED utilizando potenciômetro



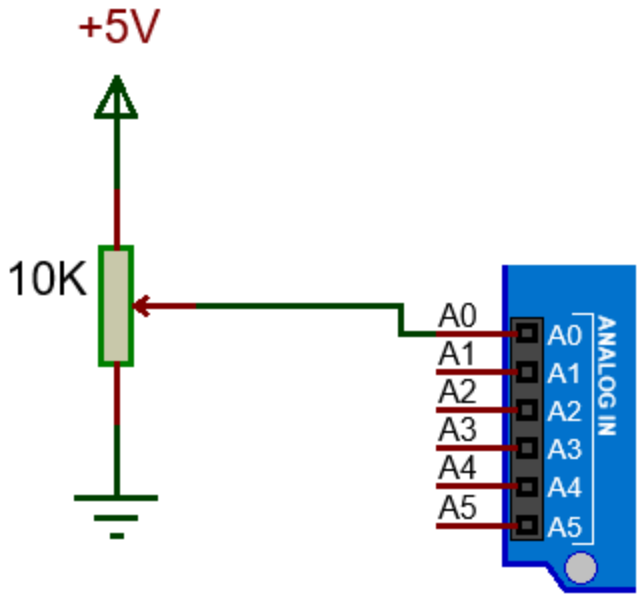
Prática 7 – Controle de Luminosidade do LED utilizando potenciômetro



Prática 7 – Controle de Luminosidade do LED utilizando potenciômetro

O sinal de 0V a 5V aplicado nas entradas analógicas serão convertidos para valores binários de 10 bits, com isso teremos valores de 0 a 1023 que equivale a números binários de 0000000000 a 1111111111.

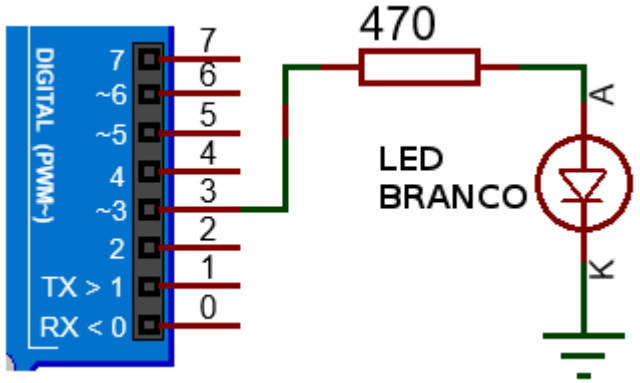
Para realizar essa leitura usaremos o comando **analogRead()**



Prática 7 – Controle de Luminosidade do LED utilizando potenciômetro

O sinal de saída é na forma digital, com nível lógico 0 (0V) e nível lógico 1 (5V), mas para controlar a luminosidade no LED, controlamos o tempo que esse nível logico permanece alto ou baixo, através da saída PWM (Modulação por Largura de Pulso).

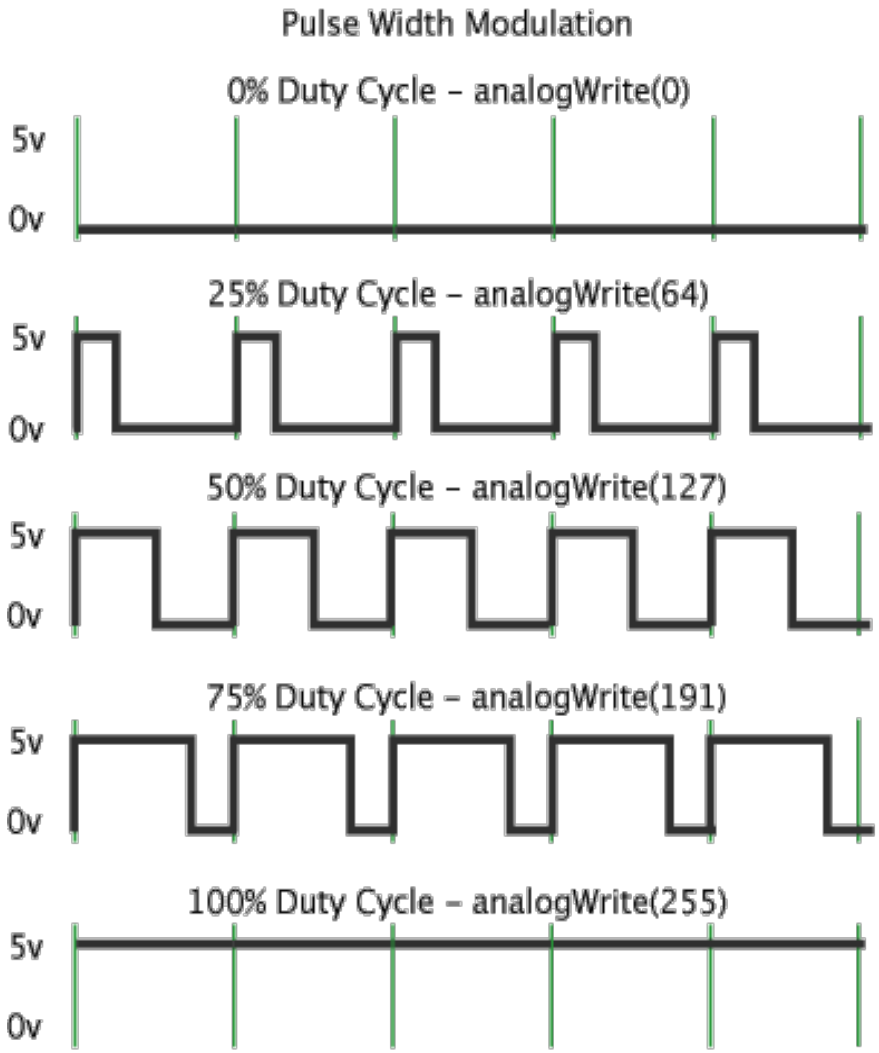
Para escrever esse sinal na saída do LED, usaremos o comando **analogWrite()**



Sinal PWM (Pulse Width Modulation)

O sinal PWM controla a largura do pulso, controlando assim o nível médio de potência aplicada na carga.

Esse controle é dentro de uma escala de 0 à 255 (8 bits).



Fonte: https://www.arduino.cc/en/Tutorial/PWM

Prática 7 – Controle de Luminosidade do LED utilizando potenciômetro

```
int Led = 3;                // atribuiu pino 3 digital a palavra Led
int Pot = 0;                // atribuiu pino A0 analogico a palavra Pot
int valor = 0;              // atribuiu 0 a varivel valor

// float  variavel com maior resoluçao de numeros inteiros
// pode chegar a 3.4028235 E+38 (32 bits)
float luminosidade = 0;  // atribuiu 0 a varivel luminosidade

void setup()
{
  pinMode(Led, OUTPUT); //Define o pino do 3 como saída
}

void loop()
{
  valor = analogRead(Pot);     // Le o valor analogico do potenciometro (0 a 1023)
  luminosidade = map(valor, 0, 1023, 0, 255);  // Converte o valor lido do Pot
  analogWrite(Led, luminosidade); // Acende o Led - Sinal PWM (0 a 255)
 }
```

Comando **Serial.print** e **Serial.println**

Para que seja possível visualizar os valores de entrada do potenciômetro e da saída PWM, usaremos o comando **Serial.print** e **Serial.println**

Serial.print() → Imprime dados para a porta serial como texto ASCII legível por humanos.

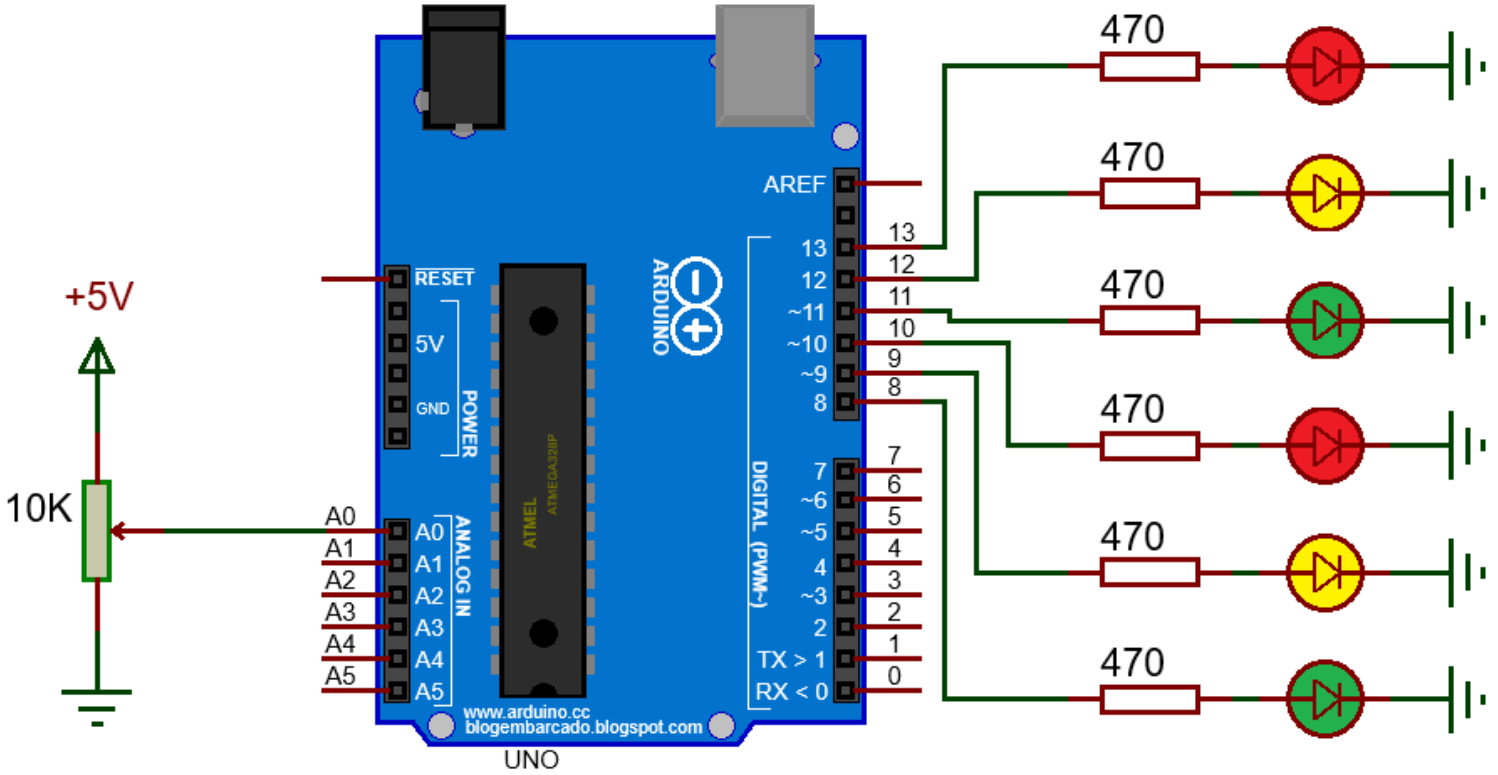
Serial.println() → Esse comando tem a mesma função que o Serial.print() seguido por um caractere ASCII de retorno e um caractere de nova linha.

Prática 7 – Controle de Luminosidade do LED utilizando potenciômetro

```
void setup()
{
  Serial.begin(9600);     //Inicializa a comunicacao serial
  pinMode(Led, OUTPUT); //Define o pino do 3 como saída
}

void loop()
{
  valor = analogRead(Pot);     // Le o valor analogico do potenciometro (0 a 1023)
  luminosidade = map(valor, 0, 1023, 0, 255);  // Converte o valor lido do Pot

  Serial.print("Valor lido do Potenciometro : ");
  Serial.print(valor);
  Serial.print(" = Luminosidade : ");
  Serial.println(luminosidade);

  analogWrite(Led, luminosidade); // Acende o Led - Sinal PWM (0 a 255)
 }
```

Prática 8 – Controle de acionamento de LEDs utilizando potenciômetro

Crie um programa onde ao variar o potenciômetro os LED's se acendam proporcionalmente.



Prática 8 – Controle de acionamento de LEDs utilizando potenciômetro