PDM: Projeto para Dispositivos Móveis
Aula 05: Trabalhando com LEDs e Potenciômetro

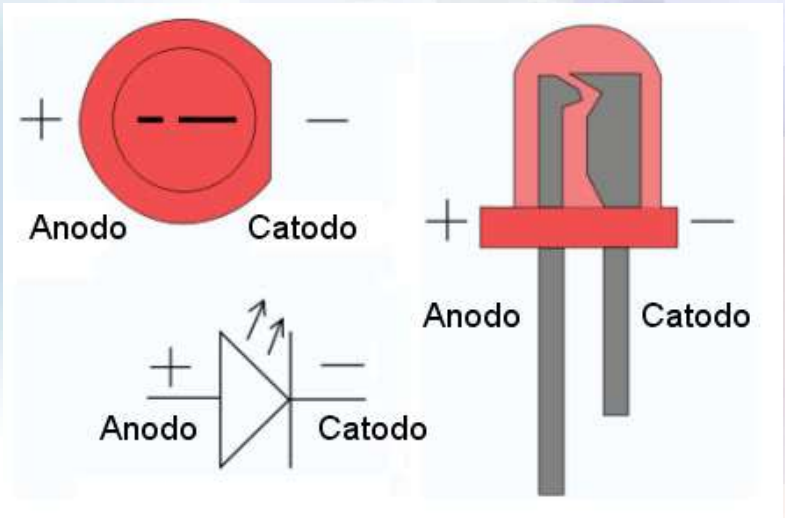
Breno Lisi Romano

http://sites.google.com/site/blromano

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Especialização em Desenvolvimento para Dispositivos Móveis

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus São João da Boa Vista

LEDs



Problem Based Learnd - PBL 01

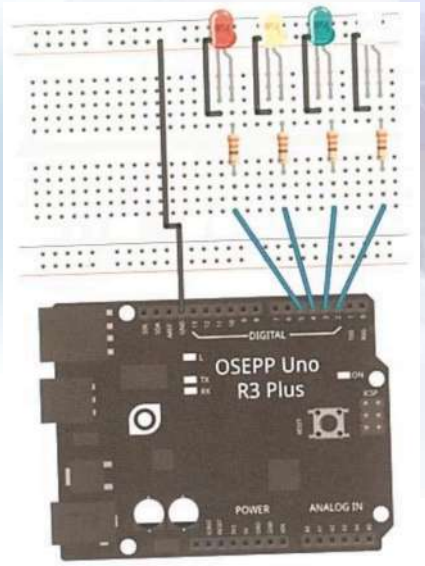
- **Problema**: Acender e Apagar LEDs de cores diferentes utilizando como entrada informações do Usuário
  - Princípio básico para aprender a conversar com o Bluetooth

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- 4 LEDs (vermelho, verde, amarelo e branco)
- 4 Resistores 330 ohms
- 9 cabos de jumpers

Construindo o Circuito no Protoboard



Funções Necessárias para o Desenvolvimento do Sistema Embarcado (1)

Função	Exemplo	Notas
Serial.begin(taxa) Essa função habilita a porta serial e fixa a taxa de transmissão e recepção em bits por segundo entre o computador e o Arduino.	**Serial.begin(9600);** Nesse exemplo essa função fixa a taxa de comunicação em 9600 bps. Os pinos digitais 0 e 1 não podem ser utilizados como entrada ou como saída de dados quando a porta serial é habilitada por essa função.	Essa função vai sempre dentro da função **setup( ).**
Serial.available( ) A função **Serial.available( )** retorna o número de bytes disponíveis para leitura no buffer da porta serial.	**int total = Serial.available( );** Aqui a variável inteira 'total' vai guardar o número de caracteres que estão disponíveis para leitura na porta serial.	O valor 0 é retornado quando não há nenhuma informação para ser resgatada na porta serial.
Serial.read( ) A função **Serial.read( )** lê o primeiro byte que está no buffer da porta serial.	**int valor = Serial.read( );** Aqui a variável inteira 'valor' vai guardar o primeiro byte (caracter) disponível na porta serial.	O valor -1 é retornado quando não há nenhuma informação para ser resgatada na porta serial.

Funções Necessárias para o Desenvolvimento do Sistema Embarcado (2)

Função	Exemplo	Notas
pinMode(pino,modo) Serve para estabelecer a direção do fluxo de informações em qualquer dos 14 pinos digitais. Dois parâmetros devem ser passados à função: o primeiro indica qual pino vai ser usado; o segundo, se esse pino vai ser entrada ou se vai ser saída dessas informações.	**pinMode(2,OUTPUT);** Aqui o pino 2 é selecionado para transmitir informações do Arduino para um circuito externo qualquer. Para configurar esse pino como entrada, o segundo parâmetro dessa função deve ser INPUT.	Essa função é sempre escrita dentro da função **setup( )**.
digitalRead(pino) Uma vez configurado um certo pino como entrada com a função **pinMode( )**, a informação presente nesse pino pode ser lida com a função **digitalRead( )** e armazenada numa variável qualquer.	**int chave = digitalRead(3);** Nesse exemplo a variável inteira 'chave' vai guardar o estado lógico (verdadeiro/falso) presente no pino digital 3.	
digitalWrite(pino,valor) Para enviar um nível lógico para qualquer pino digital do Arduino utiliza-se essa função. Dois parâmetros são requeridos: o número do pino e o estado lógico (HIGH/LOW) em que esse pino deve permanecer.	**digitalWrite(2,HIGH);** Aqui uma tensão de 5 volts é colocada no pino 2. Para enviar terra para esse pino o segundo parâmetro deverá ser LOW.	É necessário configurar previamente o pino como **saída** com a função **pinMode( )**.

Código Fonte para Solução (1)

```
1  // The LEDs are connected to these pins
2  int LEDRedPin = 5;
3  int LEDYellowPin = 4;
4  int LEDGreenPin = 3;
5  int LEDWhitePin = 2;
6
7  void setup()
8  {
9      // set up serial at 9600 baud
10     Serial.begin(9600);
11
12     // set all four LED pins to output mode
13     pinMode(LEDRedPin, OUTPUT);
14     pinMode(LEDYellowPin, OUTPUT);
15     pinMode(LEDGreenPin, OUTPUT);
16     pinMode(LEDWhitePin, OUTPUT);
17 }
18
19 void toggleLED(int LEDPin)
20 {
21     // toggle the LED on the pin passed as an argument
22     digitalWrite(LEDPin, !digitalRead(LEDPin));
23 }
```

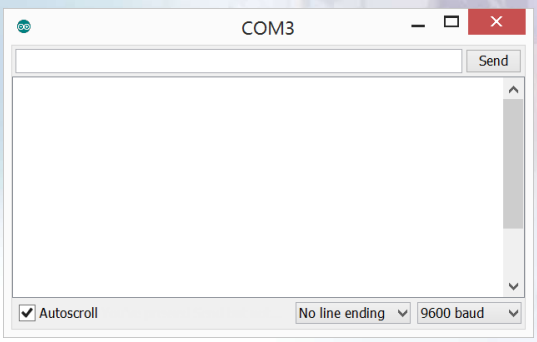
Código Fonte para Solução (2)

```
25 void loop()
26 {
27     if (Serial.available() > 0)
28     {
29         // read a single character over serial
30         int inByte = Serial.read();
31
32         switch (inByte)
33         {
34             // if we receive r, y, g, or w toggle the respective LED using our function
35             case 'r':
36                 toggleLED(LEDRedPin);
37                 break;
38             case 'y':
39                 toggleLED(LEDYellowPin);
40                 break;
41             case 'g':
42                 toggleLED(LEDGreenPin);
43                 break;
44             case 'w':
45                 toggleLED(LEDWhitePin);
46                 break;
47             default:
48                 // if we receive any other character turn all the LEDs off
49                 digitalWrite(LEDRedPin, LOW);
50                 digitalWrite(LEDYellowPin, LOW);
51                 digitalWrite(LEDGreenPin, LOW);
52                 digitalWrite(LEDWhitePin, LOW);
53                 break;
54         }
55     }
56 }
```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Abrir o Monitor Serial



- Enviar um dos seguintes caracteres para acender um dos LEDs:
  - r – Vermelho / y – Amarelo / g – Verde / w - Branco

Instituto Federal de São Paulo – IFSP São João da Boa Vista

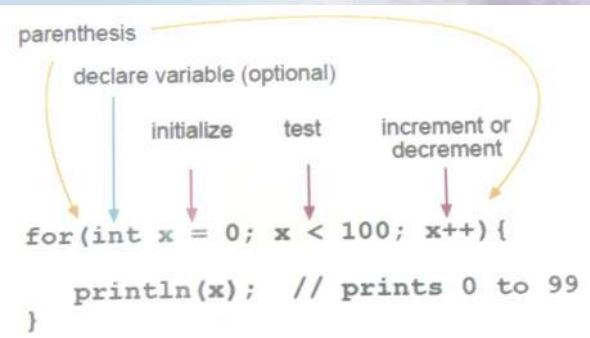
Problem Based Learnd - PBL 02

- **Problema**: Realizar efeitos luminosos com LEDs simulando o Scanner do Pontiac do seriado dos anos 80 chamado Knigh Rider em que os LEDs do para-choque do carro acendiam e apagavam indo e voltando sobre uma linha

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Informações Necessárias (*Background*)

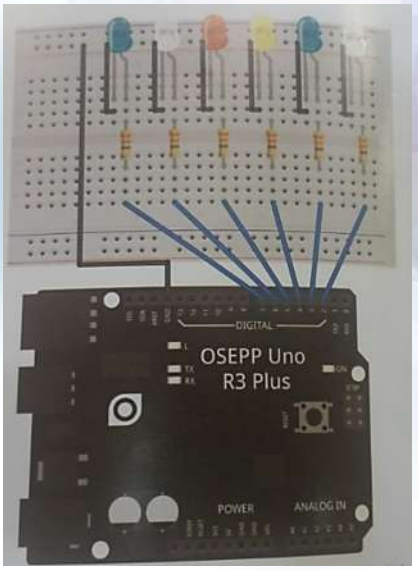
- Utilizaremos os conceito de Looping para realizar a simulação, ou seja, será adotado o laço de repetição de FOR



Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- 6 LEDs de quaisquer cores
- 6 resistores de 330 ohms
- 13 cabos de jumpers

Construindo o Circuito no Protoboard



Código Fonte para Solução

```
int timer = 100;

void setup()
{
    for (int thisPin = 2; thisPin < 8; thisPin++) {
        pinMode(thisPin, OUTPUT);
    }
}

void loop()
{
    for (int thisPin = 2; thisPin < 8; thisPin++)
    {
        // turn the pin on
        digitalWrite(thisPin, HIGH);
        // wait to turn it off so we can see it
        delay(timer);
        // turn the pin off
        digitalWrite(thisPin, LOW);
    }

    // loop from the highest pin to the lowest
    for (int thisPin = 7; thisPin > 1; thisPin--)
    {
        // turn the pin on
        digitalWrite(thisPin, HIGH);
        // wait to turn it off so we can see it
        delay(timer);
        // turn the pin off
        digitalWrite(thisPin, LOW);
    }
}
```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Verificação e Simulação do Código Desenvolvido

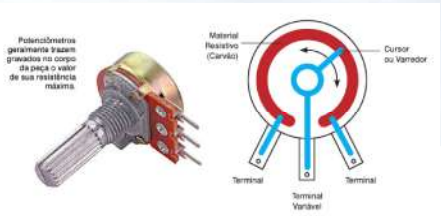
- Embarcar o código-fonte na aplicação
- Verificar se o objetivo foi atendido
- Altere o valor do timer para deixar mais rápido ou mais devagar a simulação

Problem Based Learnd - PBL 03

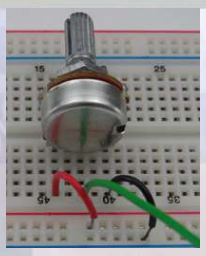
- **Problema**: Simular um gráfico de barras utilizando o conceito de *Arrays* no Arduino como também o resistor variável Potenciômetro
  - Conforme ajustamos a resistência no potenciômetro, um dos LEDs deve acender apagando o último aceso

Informações Necessárias (*Background*) (1)

- Sobre o Potenciômetro:
  - Um resistor variável que permite reajustar continuamente a resistência, de praticamente zero ohm a um valor determinado de fábrica
    - É lido um valor analógico de acordo com a variação da resistência ao girar o botão
    - Usa-se um potenciômetro quando quer variar a quantidade de corrente ou voltagem que está fornecendo a um componente do seu circuito
    - Alguns exemplos de onde encontrar potenciômetros: botões de *dimmer* de luzes, controles de volume em sistemas de som e controles para jogos ou aeromodelos (*joysticks*)



Informações Necessárias (*Background*) (2)

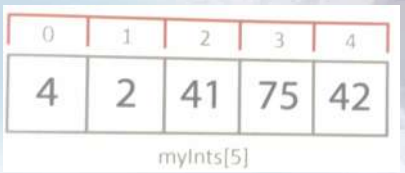


- Sobre o Potenciômetro:
  - Conectamos três fios ao ligar um potenciômetro
    - O primeiro vai do terra a partir do terminal esquerdo do potenciômetro
    - O segundo fio vai dos 5 volts ao terminal direito
    - O terceiro vai da entrada analógica escolhida ao terminal central do potenciômetro
  - Se girarmos o cursor do potenciômetro, alteramos a resistência em cada lado do contato elétrico que vai conectado ao terminal central do botão.
    - Isso provoca a mudança na proximidade do terminal central aos 5 volts ou ao terra, o que implica numa mudança no valor analógico de entrada
    - Quando o cursor for levado até o final da escala, teremos por exemplo zero volts a ser fornecido ao pino de entrada do Arduino e, assim, ao lê-lo obteremos 0
    - Quando giramos o cursor até o outro extremo da escala, haverá 5 volts a ser fornecido ao pino do Arduino e, ao lê-lo, teremos 1023
    - Em qualquer posição intermediária do cursor, teremos um valor entre 0 e 1023, que será proporcional à tensão elétrica sendo aplicada ao pino do Arduino

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Informações Necessárias (*Background*) (3)

- Sobre *Arrays* no Arduino:
  - É uma coleção de valores que pode ser acessado por um número indexado
  - Arrays podem ser multidimensionais (Matrizes)

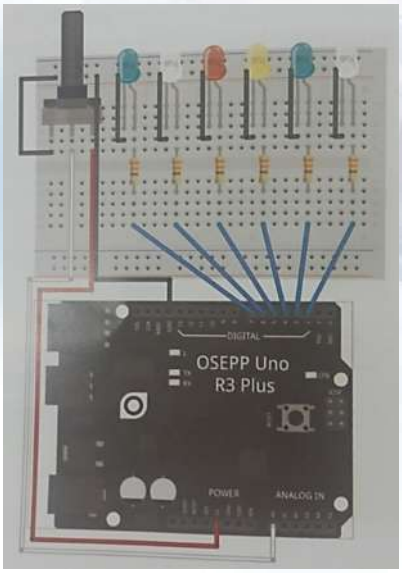


  - tipo Arrays[tamanho]: são declarados com o tipo de dado que ele irá armazenar e mais a quantidade de elementos que ele pode armazenar
    - Ex.: int myInts[5]
  - Para acessar os elementos do array, utiliza-se o index
    - Ex.: myInts[0] = 4 / myInts[4] = 42

Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- 6 LEDs de quaisquer cores
- 6 resistores de 330 ohms
- 1 Potenciômetro de 10k ohm
- 16 cabos de jumpers

Construindo o Circuito no Protoboard



Funções Necessárias para o Desenvolvimento do Sistema Embarcado (1)

Função	Exemplo	Nota
map(valor,min1,max1,min2,max2) A função **map( )** converte uma faixa de valores para outra faixa. O primeiro parâmetro 'valor' é a variável que será convertida; o segundo e o terceiro parâmetros são os valores mínimo e máximo dessa variável; o quarto e o quinto são os novos valores mínimo e máximo da variavel 'valor'.	**int valor = map(analog Read(A0),0,1023,0,255));** A variável 'valor' vai guardar a leitura do nível analógico no pino A0 convertida da faixa de 0-1023 para a faixa 0-255.	Com essa função é possível reverter uma faixa de valores, exemplo: **int valor = map(x,1,100,100,1);**
sizeof(elemento) A função **sizeof()** retorna a quantidade de elementos que existem em um array.	**myInts[] = [2, 4, 4]** **int t = sizeof(myInts) = 3** A variável t irá guardar o tamanho do array myInts[].	N/A

Funções Necessárias para o Desenvolvimento do Sistema Embarcado (2)

Função	Exemplo	Notas
analogRead(pino) Essa função lê o nível analógico presente no pino indicado pelo parâmetro entre parênteses e, após a conversão para o seu equivalente em bits, o guarda em uma variável determinada pelo programador.	**int sensor = analogRead(A0);** Aqui a variável inteira 'sensor' vai armazenar a tensão analógica convertida para digital presente no pino A0. Essa informação vai ser um valor inteiro entre 0 (para 0 volt no pino) e 1023 (se 5 volts no pino). Uma tensão de 2,5 volts no pino A0 vai fazer a variável 'sensor' guardar o valor inteiro 512.	Os pinos analógicos são reconhecidos pela linguagem C do Arduino tanto como A0 a A5 como 14 a 19. Assim, a mesma expressão acima pode ser escrita tambem da seguinte forma: **int sensor = analogRead(14);**
Uma observação importante em relação a esses pinos analógicos é que eles podem ser configurados tambem como pinos digitais pela função **pinMode( )**, aumentando assim o número desses pinos para 20. Assim, a expressão **pinMode(14,OUTPUT);** transforma o pino analógico A0 em pino de saída digital como qualquer outro presente nas duas barras de pinos digitais.		

Código Fonte para Solução (1)

```
// the pin that the potentiometer is attached to
int potPin = A0;
// an array of pin numbers to which LEDs are attached
// to add more LEDs just list them here in this array
int ledPins[] = {2, 3, 4, 5, 6, 7};
// the number of LEDs in the bar graph
int ledCount = sizeof(ledPins) / sizeof(ledPins[0]);

void setup()
{
    // use a for loop to initialize each pin as an output
    for (int thisLed = 0; thisLed < ledCount; thisLed++)
    {
        pinMode(ledPins[thisLed], OUTPUT);
    }
}
```

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Código Fonte para Solução (2)

```
void loop()
{
    // read the potentiometer
    int potReading = analogRead(potPin);
    // map the result to a range from 0 to the number of LEDs
    int ledLevel = map(potReading, 0, 1023, 0, ledCount);

    // loop over the LED array:
    for (int thisLed = 0; thisLed < ledCount; thisLed++)
    {
        // if the array element's index is less than ledLevel
        // turn the pin for this element on
        if (thisLed < ledLevel)
        {
            digitalWrite(ledPins[thisLed], HIGH);
        }
        // turn off all pins higher than the ledLevel
        else
        {
            digitalWrite(ledPins[thisLed], LOW);
        }
    }
}
```

Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Mexer no potenciômetro para visualizar as mudanças no gráfico de barras

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Problem Based Learnd - PBL 04

- **Problema**: Como construir um dimmer, ou seja, um circuito que serve para controlar a intensidade (luminosidade) de uma lâmpada/LED
  - Deve-se utilizar um potenciômetro para definir a luminosidade do LED
  - Deve-se imprimir na porta serial os valores lidos do potenciômetro e da luminosidade do LED

Informações Necessárias (*Background*)

- Utilizaremos a função **map()** para converter o valor lido da entrada analógica (entre 0 e 1023), para um valor entre 0 e 255 (8 bits), que será utilizado para ajustar a luminosidade do LED
- Repare na imagem que no Arduino os pinos 3, 5, 6, 9, 10 e 11 são PWM (*Pulse Width Modulation* - Modulação por Largura de Pulso), o que permite que eles sejam usados de forma analógica

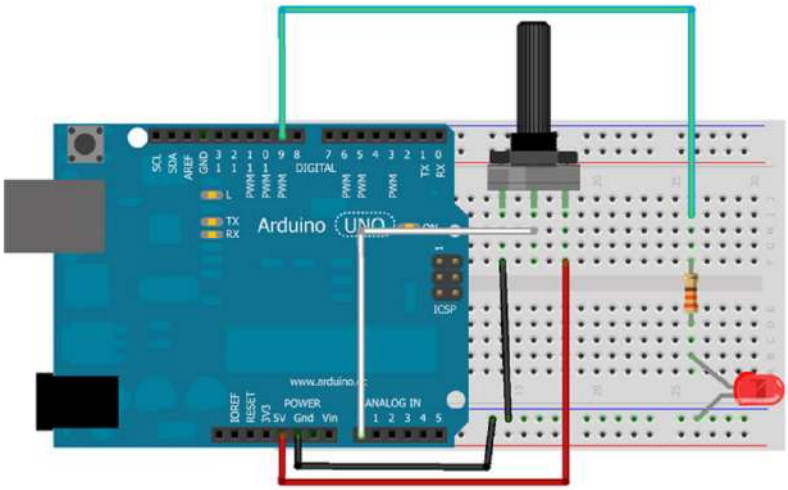
- O "truque" do PWM é ligar e desligar essa porta rapidamente, simulando uma voltagem entre 0 e 5 volts
  - O tempo que a porta permanece ligada ou desligada determina o valor da saída analógica, assim, se você quiser 50% da luminosidade, a porta ficaria 50% do tempo em modo ON (ligada), e 50% em modo OFF (desligada). Essa variação é feita de forma tão rápida que não é percebida pelo olho humano

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Componentes Utilizados

- Arduino UNO / Arduino MEGA
- Cabo USB
- Protoboard
- 1 potenciômetro de 10k ohms
- 1 LED Vermelho
- 1 resistor de 330 ohms

Construindo o Circuito no Protoboard



Funções Necessárias para o Desenvolvimento do Sistema Embarcado (1)

Função	Exemplo	Notas

Serial.print(valor, formato)
Essa função envia para a porta serial um caracter ASCII, que pode ser capturado por um terminal de comunicação. O segundo parâmetro, 'formato', é opcional e especifica com quantas casas decimais ou com que base numérica vai ser o número transmitido.
	Serial.print(1.23456); // transmite 1.23 (default)
	Serial.print(1.23456,3); // transmite 1.234
	Serial.print("Alô Mundo!"); // transmite a frase (string)
	Serial.print('A'); // transmite o caracter A
	Serial.print('A',BIN); // transmite 01000001
	Serial.print('A',OCT); // transmite o octal 101
	Serial.print('A',HEX); // transmite o hexa 41
	Serial.print('A',DEC); // transmite o decimal 65	

Serial.println(valor, formato) Como a anterior essa função envia para a porta serial um caracter ASCII com os mesmos parâmetros opcionais de 'formato', porem acrescenta ao final da transmissão o caracter **Carriage Return** (retorno ao início da linha) e o caracter **New Line** (mudança para a próxima linha).		

Funções Necessárias para o Desenvolvimento do Sistema Embarcado (2)

Função	Exemplo	Notas
analogWrite(pino,valor) O Arduino pode gerar tensões analógicas em 6 de seus 14 pinos digitais com a função **analogWrite( )**. Dois parâmetros devem ser passados à função: o primeiro indica em qual pino será gerada a tensão; o segundo determina a amplitude dessa tensão, e deve ter valores entre 0 (para 0 volt) e 255 (para 5 volts).	**analogWrite(10,128);** Com esses parâmetros uma tensão analógica de 2,5 volts vai aparecer no pino 10. Não é necessário configurar um pino PWM como saída com a função **pinMode( )** quando se chama função **analogWrite( )**.	Modulação por Largura de Pulsos, ou PWM (Pulse Width Modulation) na lingua inglesa, é uma técnica usada para gerar tensões analógicas a partir de uma sequência de pulsos digitais.

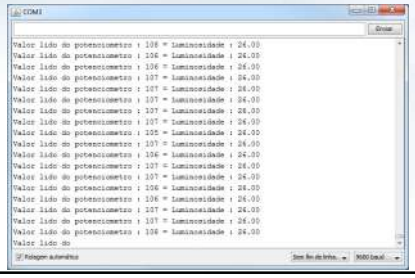
Instituto Federal de São Paulo – IFSP São João da Boa Vista

Código Fonte para Solução

```
//Dimmer - LED + Potenciometro
int pinoled = 9;  //Pino ligado ao anodo do led
int pinopot = A0;   //Pino ligado ao pino central do potenciometro
int valorpot = 0;  //Armazena valor lido do potenciometro, entre 0 e 1023
float luminosidade = 0; //Valor de luminosidade do led

void setup()
{
   Serial.begin(9600);         //Inicializa a serial
   pinMode(pinoled, OUTPUT);  //Define o pino do led como saida
   pinMode(pinopot, INPUT);   //Define o pino do potenciometro como entrada
}

void loop()
{
  // Le o valor - analogico - do pino do potenciometro
  valorpot = analogRead(pinopot);

  //Converte e atribui para a variavel "luminosidade" o valor lido do potenciometro
  luminosidade = map(valorpot, 0, 1023, 0, 255);

  //Mostra o valor lido do potenciometro no monitor serial
  Serial.print("Valor lido do potenciometro : ");
  Serial.print(valorpot);

  //Mostra o valor da luminosidade no monitor serial
  Serial.print(" = Luminosidade : ");
  Serial.println(luminosidade);

  //Envia sinal analogico para a saida do led, com luminosidade variavel
  analogWrite(pinoled, luminosidade);
}
```

Verificação e Simulação do Código Desenvolvido

- Embarcar o código-fonte na aplicação
- Alterar o potenciômetro e verificar se a luminosidade do LED foi modificada
  - Para facilitar a visualização de como as portas analógicas se comportam, o programa mostra no monitor serial não só o valor lido do potenciômetro, como também o valor da luminosidade após a utilização da função map :



Miniprojeto 02

- Alterar a velocidade que um único LED pisca utilizando como entrada o valor obtido de um potenciômetro
  - Quanto maior for o valor medido pelo potenciômetro, mais rápido o LED deve piscar.
  - Quanto menor a velocidade, mais devagar ele deve piscar

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Miniprojeto 03

- Simular o funcionamento de um Semáforo de carros utilizando-se 03 LEDs (Vermelho, Amarelo e Verde)
  - A mudança de estado do semáforo é: Vermelho -> Verde -> Amarelo -> Vermelho
  - Tempo em que o semáforo fica em estado de “Proibido o Tráfego”: 20 segundos
  - Tempo em que o semáforo fica em estado de “Alerta”: 5 segundos
  - Tempo em que o semáforo fica em estado de “Passagem Liberada”: 30 segundos
- O estado do semáforo devem ser impressos no Monitor Serial
- Acender e apagar os LEDs de acordo com o Estado do Semáforo

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Miniprojeto 04

- Baseado na entrada do Usuário, deve-se simular a transmissão de uma palavra utilizando-se o Código Morse. O circuito necessita ter apenas um LED VERMELHO para emitir a mensagem. A Tabela abaixo representa os sinais para cada letra do alfabeto e números existentes. Considerar a BOLA como sendo um sinal breve (150 ms) e o TRACINHO um sinal longo (500 ms).

A	·−	J	·−−−	S	···	2	··−−−
B	−···	K	−·−	T	−	3	···−−
C	−·−·	L	·−··	U	··−	4	····−
D	−··	M	−−	V	···−	5	·····
E	·	N	−·	W	·−−	6	−····
F	··−·	O	−−−	X	−··−	7	−−···
G	−−·	P	·−−·	Y	−·−−	8	−−−··
H	····	Q	−−·−	Z	−−··	9	−−−−·
I	··	R	·−·	1	·−−−−	0	−−−−−

- Exemplo para a palavra SOS:

(SOS: A letra “S” consiste de três sinais breves e letra “O”, de três sinais longos)

Instituto Federal de São Paulo – IFSP São João da Boa Vista

Miniprojeto 05

- Simular o funcionamento de um Semáforo de Carros utilizando-se 03 LEDs (Vermelho, Amarelo e Verde) interagindo com um Semáforo de Pedestres com 02 LEDs (Vermelho e Verde)
  - A mudança de estado do semáforo de carros é: Vermelho -> Verde -> Amarelo -> Vermelho. A mudança de estado do semáforo de pedestres é: Vermelho -> Verde
  - O semáforo de pedestres abre quando o semáforo de carros está em vermelho
  - O semáforo de pedestres fecha e automaticamente o semáforo de carros muda para o estado de alerta
  - Tempo em que o semáforo de carro fica em estado de “Proibido a Tráfego”: 20 segundos
  - Tempo em que o semáforo de carro fica em estado de “Alerta”: 5 segundos
  - Tempo em que o semáforo de carro fica em estado de “Passagem Liberada”: 30 segundos
  - Tempo em que o semáforo de pedestres fica em estado de “Proibido a Tráfego”: 35 segundos
  - Tempo em que o semáforo de pedestres fica em estado de “Passagem Liberada”: 30 segundos
- Os estados dos semáforos devem ser impressos no Monitor Serial

PDM: Projeto para Dispositivos Móveis

Aula 05: Trabalhando com LEDs

Breno Lisi Romano

Obrigado!

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Especialização em Desenvolvimento para Dispositivos Móveis

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus São João da Boa Vista