# Sistemas Microprocessados

## Sensores

Ricardo Balbinot

IFRS - Campus Canoas

2017

# Sumário

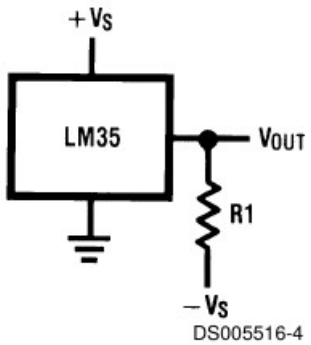
# Sumário

# Sensor LM35

- É um sensor de temperatura de precisão - em graus centígrados
- O LM35 gera uma saída de tensão, a qual é proporcional a temperatura medida
- O circuito do sensor é selado, não estando sujeito a oxidação e outras inteferências externas
- Pelo seu baixo consumo de energia, ele não aquece de forma significativa, não causando aumento superior a 0.1 graus centígrados no ar
- Não requer nenhum tipo de calibração e fornece uma acurácia de cerca de 1/4 de grau na temperatura ambiente e 3/4 de graau na faixa completa de operação

# Sensor LM35

- Sua faixa de operação varia conforme o esquema de utilização do sensor
  - Se estiver operando com uma alimentação simétrica (fonte simétrica), a temperatura medida vai da faixa de $-55\,^{\circ}C$ até $150\,^{\circ}C$




Choose $R_1 = -V_S/50$ A
$V_{OUT}$ = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

**FIGURE 2. Full-Range Centigrade Temperature Sensor**

# Sensor LM35

- Sua faixa de operação varia conforme o esquema de utilização do sensor
  - Se estiver operando com uma alimentação simples (caso que vamos utilizar com o Arduino) então sua faixa de operação vai de 2 °C até 150 °C

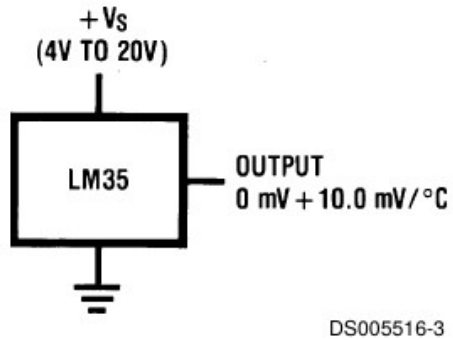



**FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)**

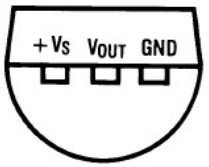
# Sensor LM35

- Note que para o caso de operação com intervalo extendido exige um circuito adicional capaz de comprimir a faixa de tensão gerada pelo sensor para a faixa de operação do conversor ADC do Arduino
- No caso simplificado de operação, a tensão gerada pelo sensor oferece uma acréscimo de $\frac{10\text{mV}}{^\circ\text{C}}$
- Nessa referência, devemos considerar a saída do LM35 como 0mV para o caso de uma temperatura de 0°C
- Dessa forma, note que na máxima temperatura de 150 °C, o sensor vai geraar uma tensão de 1.5 V
- Devemos trabalhar essa faixa de tensão mínima e máxima no Arduino de forma a aproveitar ao máximo a resolução do conversor ADC do microcontrolador

# Sensor LM35

Pinagem do sensor (olhando por baixo)

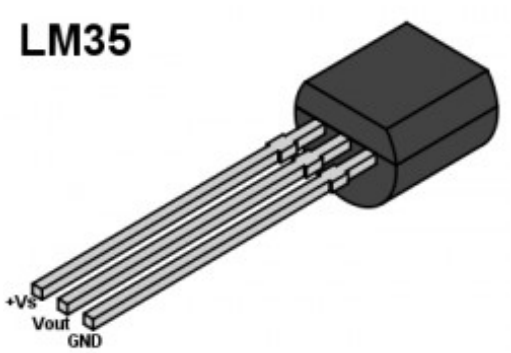
**TO-92**
**Plastic Package**




**BOTTOM VIEW**
DS005516-2

**Order Number LM35CZ,**
**LM35CAZ or LM35DZ**
**See NS Package Number Z03A**

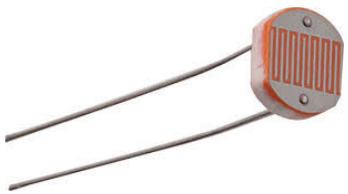
Numa outra visão...



# Sumário

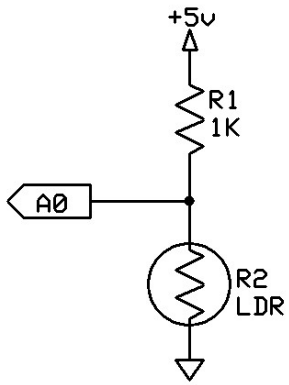
# LDR

- LDR = *Light Depedant Resistor*
- Ou seja, uma resistência dependente da luz incidente
- Quanto maior for a luminosidade incidente, menor será a resistência observada
- De fato, com escuridão total, teremos a máxima resistência
- Com luz muito brilhante, teremos uma resistência mínima, da ordem de alguns ohms
- A depender do tipo, o LDR também pode ser sensível às faixas de luz: infravermelho, luz visível e ultravioleta
- Um LDR é construído com um semicondutor de alta resistência, que, de acordo com a luz incidente, melhora a sua condutividade, reduzindo dessa maneira a sua resistência



# LDR

- O LDR deve ser utilizado sempre na forma de um divisor de tensão, conforme o esquema abaixo (obviamente, o valor das resistências pode variar)




- Dessa forma, o valor de tensão disponivel na entrada analógica do Arduino será diretamente associado a luminosidade observada.

# LDR

- Para a figura exibida, a tensão no pino A0 será:

$$V_{A0} = \frac{R2}{R1 + R2} \cdot 5\text{V}$$

# Leitura Analógica I

- Resumindo o que vimos até o momento:
  - A menos que seja feita uma configuração específica para tanto, o conversor AD do Arduino (modelos mais comuns) trabalha com sinais ingressando em seus pinos analógicos na faixa de 0 V a 5 V
  - O conversor trabalha com 10 bits, gerando um número inteiro entre 0 e 1023 para a conversão realizada (correspondendo, respectivamente, a 0 V e a 5 V)
- Dessa forma, temos que recorrer a uma regra de três a fim de realizar as conversões para os valores desejados

# Leitura Analógica II

- Primeiro passo: obter a tensão associada ao número inteiro obtido pela conversão

$$0V \rightarrow 0$$
$$5V \rightarrow 1023$$
$$x \rightarrow y$$
$$(5).y = (1023).x$$
$$x = \frac{5.y}{1023}$$

- onde x será o valor em volts (arredondado) da tensão que gerou a conversão AD para o valor y (inteiro)

# Leitura Analógica III

- Caso ainda exista uma regra de conversão adicional, a mesma tem que ser elaborada a parte. Nesse caso, normalmente iremos converter a tensão x em um valor de uma outra grandeza.
- Por exemplo, considere que tenhamos uma temperatura T, a qual pode ser obtida através da seguinte relação:
  - $-20\,^{\circ}\mathrm{C}$ correspondendo a $1\,\mathrm{V}$
  - $200\,^{\circ}\mathrm{C}$ correspondendo a $4.5\,\mathrm{V}$
- Dessa forma, teríamos que relacionar a tensão x à temperatura T

$$+1V \rightarrow -20^{\circ}\mathrm{C}$$
$$+4.5V \rightarrow +200^{\circ}\mathrm{C}$$
$$x \rightarrow T$$

# Leitura Analógica IV

- Temos que encontrar a proporção entre as grandezas nas faixas declaradas

$$(+4.5 - (+1)) \rightarrow (200 - (-20))$$
$$x_n \rightarrow T_n$$

- Fazendo a compensação para os pontos em zero, tempos que

$$x_n = x - (+1)$$
$$T_n = T - (-20)$$

# Leitura Analógica V

- Dessa forma

$$(+4.5 - (+1)) \rightarrow (200 - (-20))$$
$$x_n \rightarrow T_n$$

- Por regra de três se torna

$$(3.5) \rightarrow (220)$$
$$x_n \rightarrow T_n$$
$$3.5.T_n = 220.x_n$$

# Leitura Analógica VI

- Logo

$$x_n = x - (+1)$$
$$T_n = T - (-20)$$
$$3.5.T_n = 220.x_n$$
$$3.5.(T - (-20)) = 220.(x - (+1))$$

- E, por fim:

$$T = \frac{220.(x-1)}{3.5} - 20$$

- Esse mesmo procedimento pode ser repetido para converter qualquer faixa de tensão na faixa da grande física desejada

# Entradas analógicas

- Vamos lembrar que:
  - O ADC opera de 0 a +5V normalmente
  - Sua resolução é de 10 bits
  - Dessa forma, o intervalo de quantização do ADC é de aproximadamente 4.9mV
- Para uma acurácia de 1/4 de grau na temperatura ambiente, e considerando que a variação de tensão é de $\frac{10\text{mV}}{^\circ\text{C}}$, isso implica que o LM35 pode provocar variações de 2.5mV na sua saída
- Isso é menor do que a faixa de variação do ADC $\Longrightarrow$ vamos perder informação da temperatura se não ajustarmos o ADC

# Exemplo 1 I

Uma versão simplificada de um exemplo com o LM35 (desconsiderando a discussão anterior, pode ser vista como:

```
1  // inclui a biblioteca
2  #include <LiquidCrystal.h>
3
4  // inicializa a biblioteca com os pinos adequados do shield
5  // note que vamos usar o objeto lcd em todas as chamadas posteriores
6  LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
7
8  // a saida do LM35 esta ligado ao pino A8
9  const int LM35 = A8;
10 float temperatura; // onde vamos colocar a temperatura lida
11
12 void setup() {
13   // inicializa o display (16 caracteres x 2 linhas)
14   lcd.begin(16, 2);
15   // manda uma mensagem para o display
16   lcd.print("Temperatura");
17 }
18
19 void loop() {
20   lcd.setCursor(0, 1);
21   // a faixa de tensao eh de 5V para um 1024 possiveis valores
22   // e a variacao por grau eh de 10mV
23   temperatura = (float(analogRead(LM35)) * (5.0/1023.)/(0.01));
24   lcd.print(temperatura);
```

# Exemplo 1 II

```
25      delay(200);
26  }
```

# Problemas do exemplo 1

- Muito embora "funcione" o exemplo 1 não aproveita plenamente os recursos do Arduino
- Na verdade, estamos trabalhando numa faixa de 0–+1.5 volts com o ADC operando de 0–+5 volts
- Ou seja, desperdiçamos a maior parte da resolução do conversor

# Função analogReference

- Sintaxe: analogReference(type)
- Parâmetros:
  - type – indica o tipo de referência de tensão que deve ser usada para o conversor AD. Os possíveis valores são:
    - DEFAULT – referência de 5 V em placas de 5 volts e de 3.3 V em placas de 3.3 volts
    - INTERNAL – referência interna, igual a 1.1 V para os controladores ATmega168 ou ATmega328 e 2.56 V no ATmega8 (não disponível no Arduino Mega)
    - INTERNAL1V1 – referência interna de 1.1V (só no Arduino Mega)
    - INTERNAL2V56 – referência interna de 2.56V (só no Arduino Mega)
    - EXTERNAL: a tensão aplicada no pino AREF (somente 0 a 5V) é usada como referência
- Retorna: nada
- Uso: altera a faixa de operação do conversor ADC, melhorando a resolução utilizada

# Exemplo2 I

- Com a resolução de +5 volts, cada acrésimo de uma unidade no valor lido analógico corresponde a 4.9 mV
- Usando, por exemplo, a referência interna de 1.1V, cada acréscimo de uma unidade corresponde a 1.07 mV
- Dessa forma, sabendo que o aumento de 1 grau ocasiona o aumento de 10 mV, no primeiro caso (default) teríamos um aumento de 2.04 no valor lido, enquanto no caso de maior acurácia, o aumento será de 9.3 aproximadamente. (OBS: note que a referência interna não é exatamente 1.1V)
- É evidente que no segundo caso, teremos a capacidade de representar melhor o valor lido
- Note que com essa limitação interna, contudo, estamos limitando a leitura a uma temperatura de 110 °C

# Exemplo2 II

- Com o código a seguir, uma resolução de cera de 1/10 de grau é possível (contra somente meio grau no caso usual)

```
1  // inclui a biblioteca
2  #include <LiquidCrystal.h>
3
4  // inicializa a biblioteca com os pinos adequados do shield
5  // note que vamos usar o objeto lcd em todas as chamadas posteriores
6  LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
7
8  // a saida do LM35 esta ligado ao pino A8
9  const int LM35 = A8;
10 float temperatura; // onde vamos colocar a temperatura lida
11
12 void setup() {
13   analogReference(INTERNAL1V1);
14   // inicializa o display (16 caracteres x 2 linhas)
15   lcd.begin(16, 2);
16   // manda uma mensagem para o display
17   lcd.print("Temperatura");
18 }
19
20 void loop() {
21   lcd.setCursor(0, 1);
22   // a faixa de tensao eh de 5V para um valor indo ate 1023
23   // e a variacao por grau eh de 10mV
24   temperatura = (float(analogRead(LM35)) * (1.1/1023.)/(0.01));
```

# Exemplo2 III

```
25      lcd.print(temperatura);
26      delay(200);
27    }
```

# Exemplo LDR I

Um exemplo com o LDR, pode ser visto a seguir:

```
1  // inclui a biblioteca
2  #include <LiquidCrystal.h>
3
4  // inicializa a biblioteca com os pinos adequados do shield
5  // note que vamos usar o objeto lcd em todas as chamadas posteriores
6  LiquidCrystal lcd(8, 9, 4, 5, 6, 7);
7
8  const int LDR = 0;      // pino analgico do LDR
9  int lido = 0; // valor lido do sensor
10
11 const int alta = 100;
12 const int baixa = 700;
13
14 void setup(){
15   // inicializa o display (16 caracteres x 2 linhas)
16   lcd.begin(16, 2);
17   // manda uma mensagem para o display
18   lcd.print("Valor lido");
19 }
20
21 void loop(){
22   // lendo valor do sensor
23   lido = analogRead(LDR);
24   // manda o valor ao LCD
25   lcd.setCursor(0,1);
26   lcd.print("          ");
```

# Exemplo LDR II

```
27      lcd.setCursor(0,1);
28      lcd.print(lido);
29
30      // classifica a luminosidade
31      // baixa
32      if (lido > baixa) {
33        lcd.setCursor(8, 1);
34        lcd.print("          ");
35        lcd.setCursor(8, 1);
36        lcd.print("Baixa");
37      }
38      // media
39      if (lido<= baixa && lido >= alta) {
40        lcd.setCursor(8, 1);
41        lcd.print("          ");
42        lcd.setCursor(8, 1);
43        lcd.print("Media");
44      }
45      // alta
46      if (lido < alta) {
47        lcd.setCursor(8, 1);
48        lcd.print("          ");
49        lcd.setCursor(8, 1);
50        lcd.print("Alta");
51      }
52
53      delay(50);
54    }
```

# Exemplo LDR III

# Problemas com analogRead() I

- Um ponto necessita de cuidado no uso de analogRead()
- De acordo com a documentação dos microcontroladores Atmel usado nas placas Arduino, a toca de canais analógicos internamente no microcontrolador pode cauxar problemas em leituras sucessivas entre os diferentes canais
- Ou seja, ao tentar executar o código abaixo, teríamos problemas na leitura de um dos canais analógicos ou em ambos

```
1         var1 = analogRead(pino1);
2         var2 = analogRead(pino2);
```

Possivelmente irá resultar em erros ou problemas na leitura ou do pino 1, ou do pino 2 ou de ambos

# Problemas com analogRead() II

- A fim de resolver esse problema, a solução recomendada para o Arduino é forçar a troca do canal do ADC, inserir um pequeno delay a fim de garantir a estabilização do canal de entrada do ADC e então realizar a leitura.
- De modo que a solução recomendada, para o exemplo indicado, é algo como:

```
1         analogRead(pino1);
2         delay(1);
3         var1 = analogRead(pino1);
4         analogRead(pino2);
5         delay(1);
6         var2 = analogRead(pino2);
```