Capítulo 9

Conversor A/D

Além de um grande número de linhas de entrada/saída usadas para comunicação com circuitos externos, o Arduino Mega 2560 possui 16 entradas analógicas. Elas permitem que o microcontrolador reconheça não apenas se um pino está em lógica zero ou um, mas também permitem medir com precisão uma tensão e convertê-la em um valor numérico que pode ser expresso no formato digital.

9.1 Conversores analógico/digitais

Um conversor analógico-digital (frequentemente abreviado por conversor A/D ou ADC) é um dispositivo eletrônico capaz de gerar uma representação digital a partir de uma grandeza analógica, normalmente um sinal representado por um nível de tensão ou intensidade de corrente elétrica. Os ADCs são muito úteis na interface entre dispositivos digitais (microprocessadores, microcontroladores, DSPs, etc) e dispositivos analógicos e são utilizados em aplicações como leitura de sensores, digitalização de áudio e vídeo, etc. O uso do conversor A/D é bastante intuitivo, sendo um dos periféricos mais simples de serem utilizados nos microcontroladores.

A grosso modo, a conversão de uma tensão analógica em um número binário no conversor é baseada na comparação da tensão de entrada com uma escala interna, que tem 2^n pontos, onde *n* é chamado de resolução do conversor. A marca mais baixa da escala representa a tensão de referência inferior, digamos 0 V, enquanto que a sua maior marca, representada por 2^n-1, se refere a tensão de referência superior, digamos 5 V.

Para exemplificar o uso do ADC, consideremos um conversor A/D de 10 bits, que suporta um sinal de entrada analógico de tensão variável de 0 V a 5 V. Esse conversor pode fornecer como resultado da conversão os valores binários de 0 (0b0000000000) a 2^{10}-1=1023 (0b1111111111 – 0x3FF), ou seja, é capaz de distinguir 1024 níveis discretos de um determinado sinal, o que garante uma resolução de 4,9 mV por unidade. Se o sinal de entrada desse conversor A/D estiver em 2,5 V, por exemplo, o valor binário gerado a partir da conversão será 512, metade da faixa disponível. Dessa forma, o conversor A/D tem o comportamento mostrado no gráfico da figura 1, onde no eixo Y temos a tensão analógica presente na entrada do conversor e no eixo X temos os valores binários resultantes do processo de conversão.

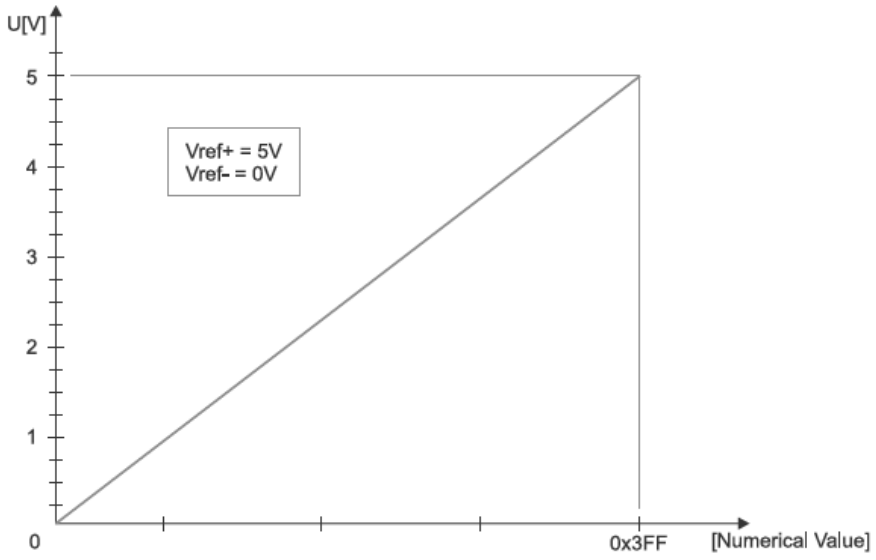

Figura 1 – Relação entre a tensão de entrada e o código da saída do conversor A/D de 10 bits.

Embora a principal função dos pinos analógicos seja a leitura de sensores analógicos, eles também têm toda a funcionalidade dos pinos de entrada e saídas digitais, podendo ser também usados como pinos digitais.

As principais características do conversor A/D do Arduino Mega são:

* conversor de aproximação sucessiva de 10 bits de resolução;
* baixo tempo de conversão de 13µs a 260µs;
* até 76,9 kSPS (até 15 kSPS com máxima resolução);
* 16 canais de entrada;
* range de entrada de 0 V a V_{CC};
* dois modos de conversão: livre ou simples;
* geração de sinal de interrupção na conclusão da conversão.

Para exemplificar o uso do conversor A/D no Arduino UNO, consideremos o exemplo mostrado a seguir, que faz a medição de temperatura usando um sensor analógico LM35 e exibe o resultado em um LCD. O circuito que implementa a função de leitura de temperatura é mostrado na figura 2 e o programa no Arduino é apresentado no quadro seguinte.

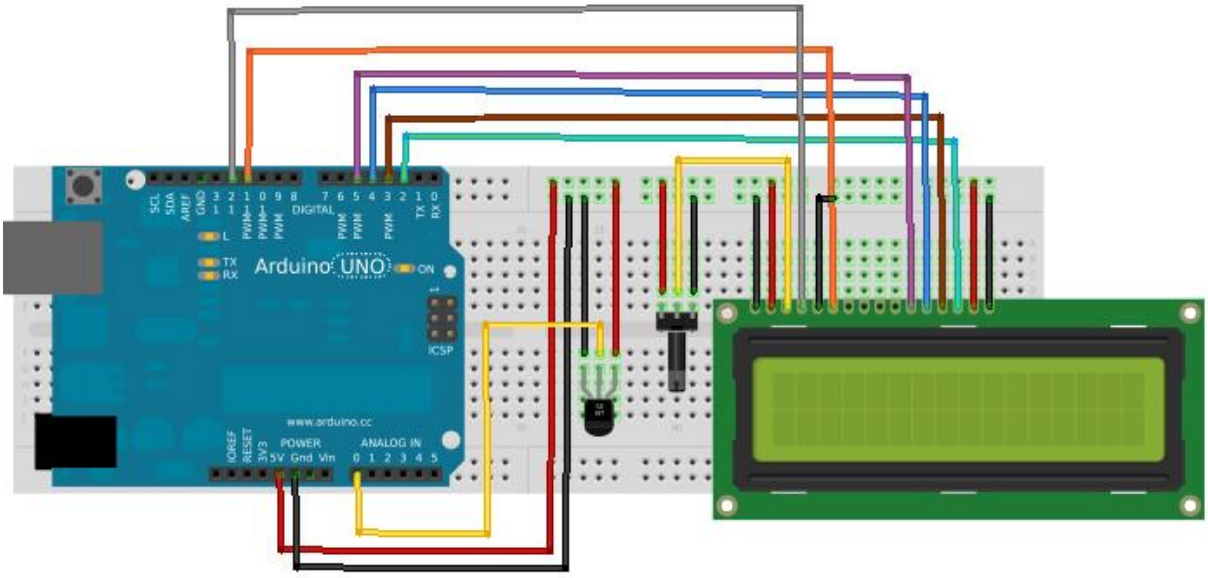

Figura 2 – Circuito para leitura da temperatura ambiente com sensor LM35.

```
#include <LiquidCrystal.h>

#define  rs  12
#define  en  11
#define  d4  5
#define  d5  4
#define  d6  3
#define  d7  2

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

void setup() {
  lcd.begin(16, 2);
}

void loop()
{
  unsigned int leitura = analogRead(A0);
  unsigned int temperatura = (leitura * 500/1023);
  lcd.clear();
  lcd.print("Temperatura:");
  lcd.setCursor(0, 1);
  lcd.print(temperatura);
  delay(1000);
}
```

No programa apresentado, temos a inclusão da biblioteca do LCD, as definições dos pinos de controle e dados e a instanciação do objeto "lcd", conforme feito no capítulo anterior, e na função setup( ) é feita a inicialização do LCD no modo de 4 bits.

Na primeira linha da função loop( ) é declarada a variável leitura. A essa variável é atribuída a função analogRead(A0) que realiza a conversão do sinal analógico presente na entrada A0, oriundo do sensor LM35, e retorna um valor de 10 bits (0 a 1023) resultante da conversão. A variável leitura não poderia ter sido declarada como um tipo com tamanho menor que 10 bits, pois isso faria com que o resultado da conversão A/D, que é de 10 bits, fosse truncado.

Como o resultado da conversão é um número binário proporcional à tensão medida, e não o próprio valor binário da temperatura, é necessário utilizar uma técnica de escalonamento de forma a calcular o valor de temperatura equivalente à saída do conversor. O escalonamento é feito pelo cálculo mostrado na atribuição da variável temperatura na segunda linha da função loop( ). Esse escalonamento será explicado com mais detalhes em seguida. Com isso, a variável temperatura possui o valor da temperatura atual.

Em seguida, a tela do display é limpa com o método lcd.clear( ) antes de ser escrito o texto "Temperatura:" com o método lcd.print( ).

No LCD, os espaços dos caracteres e as linhas são referenciados por índices com início em zero, da forma (caractere, linha). Assim, o caractere localizado na posição (3, 0), por exemplo, é o quarto caractere da primeira linha. Dessa forma, a linha seguinte usa o método lcd.setCursor(0, 1) para posicionar o cursor do LCD no primeiro caractere (índice 0) da segunda linha (índice 1), a partir do qual é escrito o valor da temperatura com o método lcd.print(temperatura). Para finalizar, há um atraso de um segundo antes do processo ser repetido.

9.2 Escalonamento de grandezas medidas pelo conversor A/D

Para um conversor de 10 bits, o resultado da conversão do sinal analógico está compreendido entre 0 e 1023. Entretanto, esse resultado não expressa o valor real da grandeza que está sendo medida. É preciso então descrever qual a relação existente entre os valores resultantes da conversão e os valores reais das grandezas.

No exemplo analisado, o sensor LM35 fornece 10 mV/°C. Assim, quando a temperatura é de 0 °C, a tensão de saída fornecida por ele é de 0 V, e se a temperatura é de 500 °C, a tensão de saída é de 5V, embora seja especificado para operar apenas em temperaturas de até 150 °C. Assim, A relação entre a tensão fornecida pelo sensor, o valor lido pelo conversor A/D e o valor da temperatura é expressa nas escalas da figura 3.

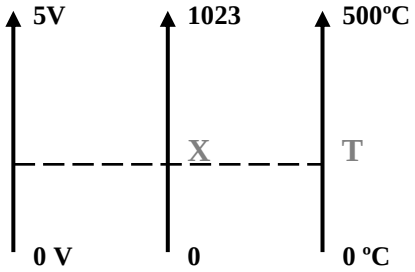

Figura 3 – Relação entre tensão fornecida pelo sensor, resultado da conversão do ADC e valor real da temperatura.

Analisando a relação acima, consegue-se demonstrar que a temperatura T equivalente a um resultado da conversão com valor X é expressa por:

$$T = \frac{(X * 500)}{1023}$$

Assim, esse deve ser o escalonamento feito para que o LCD exiba corretamente o valor de temperatura ambiente, conforme é feito na segunda linha da função loop( ) do programa exemplo.

De modo geral, para um sensor que fornece uma faixa de tensão de V_1 a V_2, resultando em uma faixa de variação de 0 a 2^n-1 como resultado da conversão de um ADC, correspondente à variação de uma grandeza física na faixa de F_1 a F_2, o escalonamento que deve ser feito é dado por:

$$F_y = \frac{F_x}{2^n - 1}.(F_2 - F_1)$$

onde F_y é o valor real da grandeza física associada a um determinado valor F_x fornecido como resultado da conversão do ADC.

9.3 Velocidade da conversão A/D

No Arduino, a operação do conversor A/D é configurada automaticamente pelas bibliotecas padrão. Essa camada de abstração auxilia os programadores iniciantes a trabalharem com o conversor A/D. Como limitação, entretanto, isso faz com que a função analogRead( ) leve cerca de 112 µs para retornar um valor convertido com a precisão máxima especificada na folha de dados do chip ATmega 2560 (a configuração completa pode ser vista no arquivo wiring.c da pasta de instalação do Arduino). A taxa de amostragem do conversor é, nesse caso, de aproximadamente 9 kSPS (9000 amostras por segundo). Para aplicações de processamento de sinal em tempo real, a depender do espectro de frequências do sinal analisado, isso pode se tornar uma séria limitação.

A taxa de amostragem pode ser sensivelmente elevada. Para isso, é necessário que o programador configure o conversor A/D a partir da manipulação direta dos registradores de controle do periférico. O conversor A/D é configurado a partir de 3 registradores (**ADMUX, ADCSRA** e **ADCSRB**) e o resultado da conversão é armazenado em dois registradores (**ADCH** e **ADCL**).

O conversor de aproximações sucessivas do ATmega 2560 requer um sinal de clock entre 50 e 200 kHz, derivado do clock principal do sistema a partir de uma estrutura interna chamada de prescaler, que nada mais é do que um divisor de frequência configurável a partir dos bits ADPS (2:0) do registrador ADCSRA. Para o caso do Arduino Mega 2560, as bibliotecas internas do IDE dividem o clock principal do sistema, de 16 MHz, por 128, o que resulta em um clock de 125 kHz e período de 8 µs para o conversor. Segundo especifica o *datasheet*, cada conversão leva 13 ciclos de clock (104 µs) para sua conclusão, que somados à sobrecarga das funções de configuração e leitura dos registradores ADCH e ADCL resultam no tempo de 112 µs informado anteriormente.

Ao dividir o clock do sistema por 64, por exemplo, elevamos o clock do conversor para 250 kHz que, apesar de estar fora da especificação para máxima precisão, permite uma conversão completa em cerca de 60 µS, garantindo uma taxa de amostragem de 16,6 kSPS.

É possível atingir ainda maiores taxas de amostragem com menores precisões (8 bits). De toda forma, a máxima taxa alcançada com a precisão máxima especificada é de aproximadamente 15 kSPS (com um clock de 200 kHz no conversor).

Por exemplo, para configurar o clock do conversor para 250 kHz, é necessário ajustar os bits ADPS2:0 do prescaler para dividir o clock do sistema por 64. Para isso, ADPS2:0 = 0b110, que pode ser feito incluindo as seguintes linhas na função setup( ):

```
ADCSRA = ADCSRA | 0b00000110;       //seta os bits ADPS2 e ADPS1
ADCSRA = ADCSRA & 0b11111110;       //reseta o bit ADPS0
```

Para um completo entendimento de como o conversor AD pode ser configurado para atender às exigências de uma determinada aplicação, consulte a seção 26 (*ADC - Analog to Digital Converter*) do *datasheet* do chip ATmega 2560. ■