UNIVERSIDADE FEDERAL DE UBERLÂNDIA - UFU

FACULDADE DE ENGENHARIA ELÉTRICA – FEELT

SINAIS E SISTEMAS EM ENGENHARIA BIOMÉDICA

Utilização das portas analógicas do Arduino

Alunos:

1. Ítalo Gustavo Sampaio Fernandes - 11511EBI004
2. Nathalia Rodrigues - 11411EBI018
3. Nicolle Ribeiro Vaz - 11511EBI014
4. Paulo Camargos Silva - 11611EBI023

Prof. Sérgio Ricardo de Jesus Oliveira

Uberlândia, **20** de **setembro** de 2017

Utilização das portas analógicas do Arduino

1 - Introdução

O Arduino é uma prototipagem eletrônica open-source que se baseia em hardwares e softwares flexíveis e fáceis de usar. Pode ser utilizado por qualquer pessoa interessada em criar objetos ou ambientes interativos. Além da placa, o Arduino conta com uma IDE que pode ser baixada gratuitamente da internet e permite programação do dispositivo utilizando a linguagem C.

O mesmo pode sentir o estado do ambiente que o cerca por meio da recepção de sinais de sensores e pode interagir com os seus arredores, controlando luzes, motores e outros atuadores. O microcontrolador na placa é programado com a linguagem de programação Arduino, baseada na linguagem Wiring e o ambiente de desenvolvimento Arduino, é baseado na ambiente Processing. Os objetos desenvolvidos com o Arduino podem ser autônomos ou podem comunicar-se com um computador para a realização da tarefa, com uso de software específico.

Arduino Due é a primeira placa de desenvolvimento Arduino baseada em ARM. Essa placa possui um poderoso microcontrolador ARM CortexM3 32bit programável através da familiar IDE do Arduino, ou seja, com essa placa você tem uma capacidade computacional muito superior aos outros Arduinos com a vantagem de poder utilizar a mesma linguagem de programação dos outros Arduinos.

O Arduino Due tem 54 entradas/saídas digitais (das quais 12 podem ser utilizadas como saídas PWM), 12 entradas analógicas, 4 UARTs (portas seriais de hardware), clock de 84MHz, capacidade de conexão USB-OTG, 2 DAC (digital para analógico), 2 TWI, conector para alimentação, pinos SPI, pinos JTAG, botão de reset e botão para apagar o código. Possui também outras funções bem interessantes como DACs, Áudio, DMA, uma biblioteca experimental para multi tasking entre outras.

Neste experimento utilizaremos o Arduino Due com a finalidade de mostrar o funcionamento das portas analógicas do mesmo. Para isso será realizado o monitoramento de uma porta analógica e o acionamento de 3 LEDs, estes em função do valor analógico apresentado na entrada analógica. Utilizaremos o Serial Monitor da IDE Arduino para apresentação do valor analógico lido.

2 - Materiais e Métodos

Os componentes utilizados nos dois experimentos estão listados abaixo:

- Arduino;
- 3 resistores de 220Ω;
- 3 LEDs (2 Vermelhos e 1 Verde)
- 1 potenciômetro de 10kΩ;
- Matriz de contatos (protoboard);

- Multímetro;

A Figura 1 abaixo exibe o circuito utilizado para montagem dos dois experimentos.

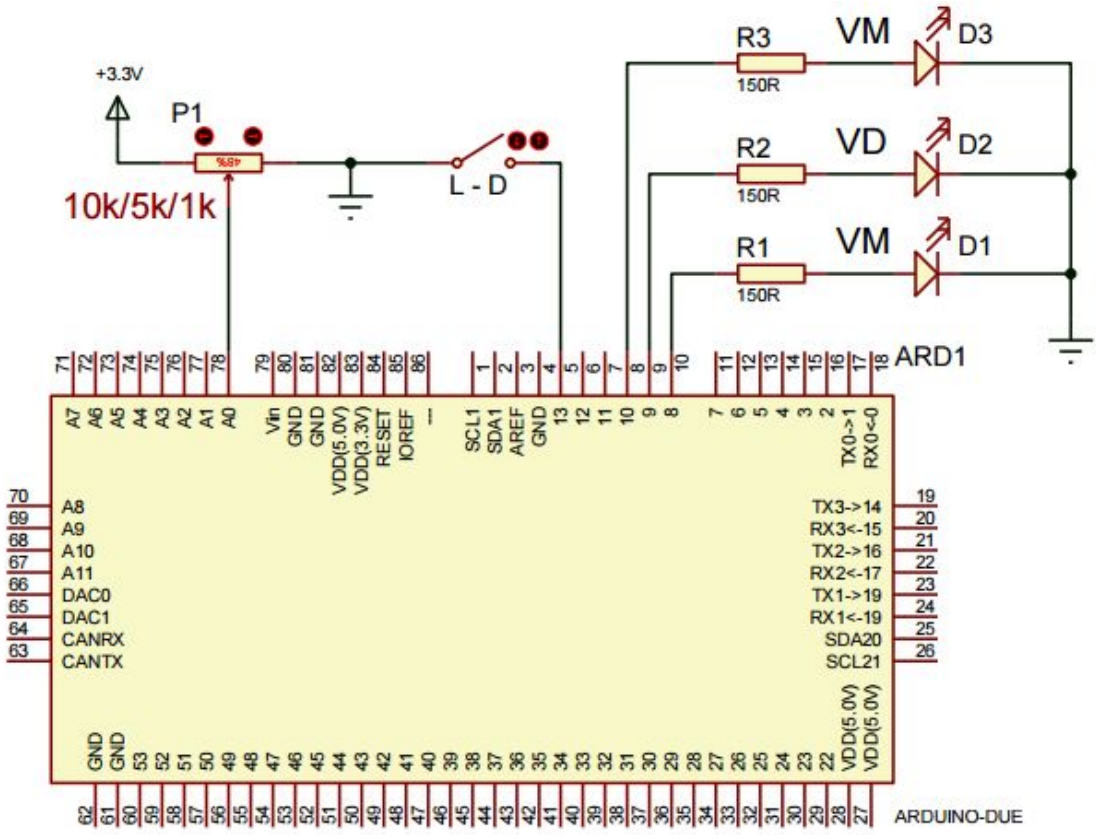

Figura 1: Representação do circuito montado para ambos os experimentos com portas analógicas.

2.1 - Experimento 1

O circuito montado e representado pela Figura I mostra 3 LED's conectados às portas digitais (pinos 8, 9 e 10), com seus respectivos resistores. O botão é conectado ao Terra (GND) e ao pino digital 13, que é responsável por verificar se o botão foi pressionado. Um potenciômetro é conectado a pino de alimentação de 3.3V e ao GND. O pino do meio, é conectado ao pino analógico A0, que fara a recepção do sinal variante de acordo com a posição da chave do potenciômetro.

O código do primeiro experimento é apresentado no Anexo I. As linhas 1 a 6 fazem a definição dos pinos do Arduino a constantes. As linhas 8 a 13 fazem a configuração dos pinos de acordo do tipo de tarefa (INPUT ou OUTPUT) que os pinos exercem. A linha 14 faz a inicialização da comunicação serial.

2.2 - Experimento 2

O circuito para o 2º experimento é o mesmo da 1º experimento (Figura 1). O código escrito para este experimento é apresentado no Anexo II no fim do documento.

As linhas 2 a 5 definem os pinos a constantes. A variável *ledAtual* armazena o número do pino cujo LED está ligado. As linhas 5 a 10 cria e inicializa variáveis para auxílio do controle do tempo (utilizado na função *millis()*). As linhas 12 a 18 fazem a configuração dos pinos de acordo do tipo de tarefa (INPUT ou OUTPUT) que os pinos exercem. A linha 17 faz a inicialização da comunicação serial.

3 - Resultados e discussões

Com a montagem seguindo o esquema eletrônico mostrado anteriormente e o desenvolvimento do software para leitura dos valores do potenciômetro, obtivemos os seguintes resultados:

No primeiro experimento dentro da função (linhas 17 a 48) *void loop()* é realizada a leitura do valor de tensão no pino A0. A função *analogRead()* retorna um valor em unidade entre 0-1023 de acordo com a tensão aplicada ao pino, podendo ser de até 3.3V. O valor é posteriormente convertido em tensão utilizando a função *map()*. A função *Serial.print()* é responsável pela exibição no monitor serial da mensagem:

"A tensao analogica apresentada na entrada A0 do ARDUINO e igual a: x.x"

A estrutura condicional *if()/else* faz a verificação do pino 13, que está conectado ao botão. Caso o botão seja pressionado, a primeira condição é satisfeita, e os LED's são ligados de acordo com o valor de tensão obtido. Outras estruturas condicionais fazem a verificação para garantir que o cada LED seja ligado somente quando o valor de tensão estiver dentro de um intervalo específico. Segue as condições:

- Quando a tensão na entrada analógica estiver abaixo de 1 V, o LED D1 acende;
- Quando a tensão na entrada analógica for maior que 1 V e menor que 2 V, o LED D2 acende;
- Quando a tensão na entrada analógica estiver acima de 2 V, o LED D3 acende;

Foi observado no experimento que ao acrescentar um delay de 300 ms no software, a mensagem de atualização da tensão passou a ser atualizada com uma frequência menor. Ao alterar a taxa de comunicação entre o arduino e o serial motor para 300 bps, a mensagem não foi exibida no monitor serial. Diversos caracteres incomuns foram observados. Isto se deve ao fato de o microcontrolador não suportar taxas de baud rate tão baixas. Ao colocar a taxa mínima possível e suportada (1200 bps), foi possível observar a frase sendo escrita na tela durante a execução do programa. No segundo experimento a função *exibeTensao()* é responsável por exibir no monitor serial a mensagem informando o valor de tensão obtido e o intervalo em que os LED's são ligados e desligados. A função *lerTensao()* é responsável por realizar a leitura e conversão dos valores lido retornando a tensão no pino analógico A0. A função *ligaLED()* é responsável por ativar o LED na sequência.

A função *void loop()* é executada repetidamente. Em seu interior, a função *millis()* retorna o tempo em milissegundos desde o início do programa. O tempo de on/off dos leds depende da variável intervalo e tensão. A estrutura *if()* faz a verificação se o tempo passado desde a última execução foi maior que o intervalo multiplicado pelo valor de tensão. Caso seja verdade, a função *ligaLED()* e *exibeTensao()* são chamadas e

executadas e o acendimento sequencial acontece. Observe que se houver um aumento de tensão, o tempo on/off aumenta.

4 - Conclusão

Através do experimento foi possível ver como o Arduino due trabalha com os sinais digitais e analógicos. Foi possível observar e aprender como o Arduino realiza a conversão analógica/digital através de um circuito que varia a tensão (potenciômetro) utilizando a função *map()*, bem como observar o comportamento do sistema ao trabalhar com funções que são executadas apenas em um intervalo de tempo específico. Para isso foi utilizado a função *delay()* e *millis()*. A função *millis()* é uma ferramenta útil quando não se deseja para a execução de outras rotinas implementadas.

5 - Referências

https://multilogica-shop.com/arduino-due
https://www.robocore.net/loja/produtos/arduino-due.html

ANEXO I - Código Lab. 02

```
1.  //Definição dos pinos
2.  #define pinAnalogico    A0
3.  #define pinButton       13
4.  #define ledVM_D3        10
5.  #define ledVD_D2        9
6.  #define ledVM_D1        8
7.
8.  void setup() {
9.    //Configuração do regime de trabalho dos pinos e comunicação serial
10.   pinMode(pinButton, INPUT_PULLUP);
11.   pinMode(ledVM_D3, OUTPUT);
12.   pinMode(ledVD_D2, OUTPUT);
13.   pinMode(ledVM_D1, OUTPUT);
14.   Serial.begin(57600); // Inicia porta serial
15. }
16.
17. void loop() {
18.   //Realiza a leitura retornando valores de unidade entre 0-1023 (0-3.3V)
19.   int valorSensor = analogRead(pinAnalogico);
20.   //Realiza conversão
21.   int  tensao_int = map(valorSensor, 0, 1023, 0, 330);
22.   float tensao = tensao_int / 100.0;
23.
24.   Serial.print("A tensao analogica apresentada na entrada A0 do ARDUINO e igual
     a: ");
25.   Serial.println(tensao);
26.   if (!digitalRead(pinButton)) {
27.     if (tensao < 1.0) {
28.       digitalWrite(ledVM_D1, 1);
29.       digitalWrite(ledVD_D2, 0);
30.       digitalWrite(ledVM_D3, 0);
31.     } else {
32.       if (tensao < 2) {
33.         digitalWrite(ledVD_D2, 1);
34.         digitalWrite(ledVM_D1, 0);
35.         digitalWrite(ledVM_D3, 0);
36.       } else {
37.         digitalWrite(ledVM_D3, 1);
38.         digitalWrite(ledVM_D1, 0);
39.         digitalWrite(ledVD_D2, 0);
40.       }
41.     }
42.   } else {
43.     digitalWrite(ledVM_D1, 0);
44.     digitalWrite(ledVD_D2, 0);
45.     digitalWrite(ledVM_D3, 0);
46.   }
47.   delay(300);
48. }
```

ANEXO II - Código Lab. 03

```
1.  //definição dos pinos
2.  #define pinAnalogico    A0
3.  #define ledVM_D3        10
4.  #define ledVD_D2        9
5.  #define ledVM_D1        8
6.  int ledAtual = 10; // Pino do LED que está ON
7.  // Variáveis de controle da função millis()
8.  const int intervalo =   200;
9.  int tempo_atual =       0;
10. int ultimo_tempo =      0;
11.
12. void setup() {
13.   //Configuração do regime de trabalho dos pinos e comunicação serial
14.   pinMode(ledVM_D3, OUTPUT);
15.   pinMode(ledVD_D2, OUTPUT);
16.   pinMode(ledVM_D1, OUTPUT);
17.   Serial.begin(57600); // Inicia porta serial
18. }
19.
20. void exibeTensao(float tensao, int intervalo) {
21.   //Função para exibição da tensão no monitor serial
22.   Serial.print("A tensao analogica apresentada na entrada A0 do ARDUINO e igual
    a: ");
23.   Serial.println(tensao);
24.   Serial.print("Intervalo: ");
25.   Serial.println(intervalo * tensao);
26. }
27.
28. float lerTensao() {
29.   //Função para leitura e conversão dos valores de tensão
30.   int valorSensor = analogRead(pinAnalogico);
31.   int  tensao_int = map(valorSensor, 0, 1023, 0, 330);
32.   float tensao = tensao_int / 100.0;
33.   return tensao;
34. }
35.
36. void ligaLED() {
37.   //Função para ativação sequencial dos LED's
38.   switch (ledAtual) { //Verifica qual LED está ligado no momento
39.     case (10): {
40.         // Liga LED do pin 9 (ledVD_D2)
41.         digitalWrite(ledVM_D1, 0);
42.         digitalWrite(ledVD_D2, 1);
43.         digitalWrite(ledVM_D3, 0);
44.         ledAtual = 9;
45.         break;
46.       }
47.     case (9): {
48.         // Liga LED do pin 8 (ledVM_D1)
49.         digitalWrite(ledVM_D1, 1);
50.         digitalWrite(ledVD_D2, 0);
51.         digitalWrite(ledVM_D3, 0);
52.         ledAtual = 8;
53.         break;
54.       }
55.     case (8): {
56.         // Liga LED do pin 10 (ledVM_D3)
57.         digitalWrite(ledVM_D1, 0);
58.         digitalWrite(ledVD_D2, 0);
59.         digitalWrite(ledVM_D3, 1);
```

```
        ledAtual = 10;
        break;
      }
    default: {
        // Desliga todos LED's
        digitalWrite(ledVM_D1, 0);
        digitalWrite(ledVD_D2, 0);
        digitalWrite(ledVM_D3, 0);
        ledAtual = 10;
      }
  }
}

void loop() {
  tempo_atual = millis();    // Tempo desde o inicio do funcionamento (ms)
  float tensao = lerTensao();   // Armazena valor de tensão lido em A0
  // Verifica se o tempo de intervalo estourou
  if ((tempo_atual - ultimo_tempo) > (intervalo * tensao)) {
    ultimo_tempo = tempo_atual;
    exibeTensao(tensao, intervalo); // Exibe mensagem no serial monitor
    ligaLED();                      // Liga o LED
  }
}
```