Experimento/Trabalho - Aquisição com Arduino - Curva I-V de um Diodo

Prof. Tiago Oliveira Weber

2018

1 Objetivo

Construir um sistema de aquisição de sinais utilizando a plataforma Arduino e aplicando na obtenção da curva I-V de um diodo.

1.1 Objetivos Específicos

- Entender a utilização de PWM e um filtro passa-baixas como um DAC (conversor digital/analógico);
- Ser capaz de utilizar a interface serial no Arduino;
- Ser capaz de utilizar a interface serial no GNU Octave;
- Entender a utilização do conversor analógico digital;
- Entender a utilização de um sistema que utilize um DAC para arbitrar uma tensão em um circuito e um ADC (conversor analógico/digital) para medir uma tensão e que seja controlado pelo GNU Octave.
- Aplicar este sistema para extrair a curva I-V de um diodo.

2 Descrição

Para cada parte do trabalho, já é fornecido pelo professor códigos simples para Arduino e Octave. O aluno deverá ser capaz de utilizar estes códigos para extrair a curva corrente por tensão (I-V) de um diodo. Para tal, serão necessários:

- 1 diodo de propósito geral; (componente que será caracterizado);
- 1 resistor com resistência entre 10 e 100 Ω; (para medição de corrente);
- 1 resistor de 2 kΩ (ou próximo) e um capacitor de 10 µ F (ou próximo) para fazer um filtro passa-baixas para o PWM;
- 1 Arduino Uno;
- alguns fios para conexões;

Os códigos e circuitos passados neste trabalho são um exemplo inicial. Ao longo da disciplina, trabalharemos em tópicos que auxiliarão a melhorar o desempenho do sistema. O roteiro para execução do trabalho é:

- o aluno deverá rodar cada código/montar circuito fornecido e fazer anotações sobre os resultados e dificuldades obtidas;
- a execução do trabalho em laboratório poderá ser realizado em grupos de até 3 integrantes.

Os códigos deste trabalho estão escritos e expostos de maneira incremental. Assim, o corpo do primeiro código é usado para o segundo, o do segundo para o terceiro, etc.

2.1 Ler ADC e escrever na serial

```
1  #define DEBUG
2  //*********************************
3  // Include Libraries
4  //*********************************
5  #include <stdarg.h>
6
7  //*********************************
8  // Serial Communication Variables
9  //*********************************
10 boolean serial_received = false;
11
12 // Functions
13 void p(char *fmt, ... ){ //http://playground.arduino.cc/Main/Printf
14         char buf[128]; // resulting string limited to 128 chars
15         va_list args;
16         va_start (args, fmt );
17         vsnprintf(buf, 128, fmt, args);
18         va_end (args);
19         Serial.print(buf);
20 }
21
22 void setup() {
23   Serial.begin(9600);
24   pinMode(A0, INPUT);
```

```
25 }
26
27 void loop() {
28
29   //**********************************
30   // Read the input on analog pin 0:
31   //**********************************
32   int adc_value = analogRead(A0);
33
34   //**********************************
35   // Serial output
36   //**********************************
37   // Can't be written only as Serial.write(adc_value) because the ADC has a
         10-bit resolution and we are sending 8 bits at a time
38   // Will be on code only if not in debug mode
39   #ifndef DEBUG
40   Serial.write(0xff);              // send init byte
41   Serial.write(1);                 // send channel
42   Serial.write(adc_value >> 8);
43   Serial.write(adc_value);
44   #endif
45
46   //**********************************
47   // ASCII output
48   //**********************************
49   // Print the values to the serial terminal for debugging purposes in the
         serial monitor
50   // Will be on code only in debug
51   #ifdef DEBUG
52   p("Debug␣Mode\n");
53   #endif
54
55   delay(1000);
56 }
```

- Mantenha a linha #define DEBUG e veja no "serial monitor" da IDE do Arduino se "Debug Mode" está sendo recebido;
- Comente a linha #define DEBUG e teste conectar a entrada do ADC ao nível lógico baixo ou alto para observar se os dados que chegam na serial mudam de acordo.

Observação: utilize um monitor serial capaz de ver os dados em hexadecimal (por exemplo, no Linux há o **gtkterm** e no Windows, **hyperterminal**) e conferir se os valores estão de acordo com o esperado. Utilizar programas como esses que também são capazes de **escrever** em hexadecimal será importante para os próximos passos e projetos.

2.2 Ler ADC, ler serial e escrever na serial

```
1 #define DEBUG
2 //*********************************
3 // Include Libraries
4 //*********************************
5 #include <stdarg.h>
```

```
6
7  //*********************************
8  // Serial Communication Variables
9  //*********************************
10 boolean serial_received = false;
11 int inByte = 0;          // incoming serial byte
12 int received_value = 0;
13
14
15 // Functions
16 void p(char *fmt, ... ){ //http://playground.arduino.cc/Main/Printf
17         char buf[128]; // resulting string limited to 128 chars
18         va_list args;
19         va_start (args, fmt );
20         vsnprintf(buf, 128, fmt, args);
21         va_end (args);
22         Serial.print(buf);
23 }
24
25 void setup() {
26   Serial.begin(9600);
27   pinMode(A0, INPUT);
28 }
29
30 void loop() {
31   //*********************************
32   // checks and receive data from serial
33   //*********************************
34   if (Serial.available() > 0) {
35     int i=0;
36     while (Serial.available() > 0)
37       {
38         inByte = Serial.read();
39         if (inByte==255)
40             {
41             received_value = Serial.read();
42             serial_received = true;
43             }
44       }
45   }
46
47   //*********************************
48   // Read the input on analog pin 0:
49   //*********************************
50   int adc_value = analogRead(A0);
51
52   //*********************************
53   // Serial output
54   //*********************************
55   // Can't be written only as Serial.write(adc_value) because the ADC has a
         10-bit resolution and we are sending 8 bits at a time
56   // Will be on code only if not in debug mode
57   #ifndef DEBUG
58   Serial.write(0xff);             // send init byte
59   Serial.write(1);                // send channel
60   Serial.write(adc_value >> 8);
61   Serial.write(adc_value);
62   #endif
63
64   //*********************************
65   // ASCII output
66   //*********************************
```

```
67    // Print the values to the serial terminal for debugging purposes
68    // Will be on code only in debug mode
69    #ifdef DEBUG
70    p("Debug␣Mode\n");
71    if (serial_received) {
72         p("Received␣value:␣%d", received_value);
73          }
74    #endif
75
76    delay(1000);
77    serial_received = false; // resets serial_received variable
78  }
```

- Mantenha a linha #define DEBUG e veja no "serial monitor" da IDE do Arduino se "Debug Mode" está sendo recebido.

- Envie uma informação que comece com o valor em hexadecimal FF seguido de um valor qualquer (exemplo: FF 01). Veja se este valor aparece como resposta no terminal "Received value: valor".

2.3 Ler ADC, escrever na saída PWM, ler serial, escrever na serial

```
1   //Author Tiago Oliveira Weber - 2018   - tiago (dot) oliveira (dot) weber (
        at) gmail (dot) com
2
3   //#define DEBUG
4   //*********************************
5   // PWM
6   //*********************************
7   #define pwmpin 3
8
9   //*********************************
10  // Include Libraries
11  //*********************************
12  #include <stdarg.h>
13
14  //*********************************
15  // Serial Communication Variables
16  //*********************************
17  boolean serial_received = false;
18  int inByte = 0;         // incoming serial byte
19  int received_value = 255;
20  int old_value = 255;
21
22  // Functions
23  void p(char *fmt, ... ){ //http://playground.arduino.cc/Main/Printf
24          char buf[128]; // resulting string limited to 128 chars
25          va_list args;
26          va_start (args, fmt );
27          vsnprintf(buf, 128, fmt, args);
28          va_end (args);
29          Serial.print(buf);
30  }
31
32  void setup() {
33    Serial.begin(9600);
```

```
34    pinMode(A0, INPUT);
35    pinMode(pwmpin, OUTPUT);
36  }
37
38  void loop() {
39    //**********************************
40    // checks and receive data from serial
41    //**********************************
42    serial_received = false; // resets serial_received variable
43    Serial.flush();
44    if (Serial.available() > 0) {
45      int i=0;
46      while (Serial.available() > 0)
47        {
48          inByte = Serial.read();
49          if (inByte==255)
50              {
51              received_value = Serial.read();
52              serial_received = true;
53              }
54        }
55    }
56
57    //**********************************
58    // Read the input on analog pin 0:
59    //**********************************
60    int adc_value = analogRead(A0);
61
62    //**********************************
63    // Serial output
64    //**********************************
65    // Can't be written only as Serial.write(adc_value) because the ADC has a
          10-bit resolution and we are sending 8 bits at a time
66    // Will be on code only if not in debug mode
67    #ifndef DEBUG
68    Serial.write(0xff);             // send init byte
69    Serial.write(1);                // send channel
70    Serial.write(adc_value >> 8);
71    Serial.write(adc_value);
72    #endif
73
74    //**********************************
75    // ASCII output
76    //**********************************
77    // Print the values to the serial terminal for debugging purposes in the
          serial monitor
78    // Will be on code only in debug
79    #ifdef DEBUG
80    p("Debug␣Mode\n");
81    if (serial_received) {
82         p("Received␣value:␣%d", received_value);
83          }
84    #endif
85
86    //**********************************
87    // PWM output
88    //**********************************
89    if (serial_received && (received_value != old_value)) {
90         analogWrite(pwmpin, received_value);
91         old_value = received_value;
92    }
93
```

```
94    Serial.flush();
95    delay(50);
96  }
```

- Faça um filtro passa-baixas usando um resistor (1k Ω) e um capacitor (10 µ F) e o conecte na saída PWM do arduino.

- Altere os valores do PWM através de comandos pela serial e veja se a saída do circuito fica correspondente ao valor desejado. Faça medições iniciais com o multímetro e depois com o osciloscópio. Capture imagens do osciloscópio mostrando a saída antes do filtro e depois do filtro. Quanto tempo demora para a saída ficar estável após uma alteração de tensão de 0 para 5V?

2.4 Código em GNU Octave

Para executar estes códigos, é necessário ter instalado o pacote instrument-control. Para executar, digite "serial_characterize" a partir do terminal do GNU Octave, garantindo que esteja na pasta onde todos os arquivos ".m" listados a seguir estejam.

- investigue quaisquer erros de execução e faça adequações no código caso necessário;

- investigue a curva I-V resultante. Caso acredite que exista algo que você possa fazer para melhorar os resultados, adapte o circuito, firmware e software;

- descreva em poucas palavras o funcionamento do sistema, descrevendo o papel de cada função do código.

2.4.1 Arquivo serial_characterize.m

```
1  function serial_characterize();
2  pkg load instrument-control
3  warning ("on", "Octave:int-math-overflow")
4  % Based on https://www.edn.com/design/analog/4440674/Read-serial-data-
       directly-into-Octave
5
6  if (exist("serial") != 3)
7      disp("No␣Serial␣Support");
8  endif
9
10 s1 = serial("/dev/ttyACM0");   % Open the port
11 set(s1, 'baudrate', 9600);
12 set(s1, 'bytesize', 8);        % 5, 6, 7 or 8
13 set(s1, 'parity', 'n');        % 'n' or 'y'
14 set(s1, 'stopbits', 1);        % 1 or 2
```

```
15 set(s1, 'timeout', 20);        % 10.0 seconds
16
17 srl_flush(s1);
18
19 inserted_voltage = 0:0.1:5;
20 meas_adc = zeros(1,length(inserted_voltage));
21 inserted_code=  zeros(1,length(inserted_voltage));
22
23 for i=1:length(inserted_voltage)
24          fprintf(1,'Testing␣for␣value␣#%d:␣%f␣Volts␣\n',i,inserted_voltage(i
               ));
25          inserted_code(i) = uint8(inserted_voltage(i)*(2**8)/5.0);
26          meas_adc(i) = test_codevalue(s1,inserted_code(i));
27          meas_voltage(i) = meas_adc(i)*5.0/(2**10);
28 end
29 resistance = 75;
30 current = (inserted_voltage - meas_voltage)/resistance;
31 fclose(s1)
32
33 subplot(1,2,1);
34 plot(meas_voltage,current,'*r');
35 subplot(1,2,2);
36 plot(inserted_voltage,meas_voltage,'*b');
37
38 end
```

2.4.2 Arquivo test_codevalue.m

```
1  function [meas_adc] = test_codevalue(s1,value)
2  for i=1:5
3           send_value(s1,value);
4           [test_value(i),test_success] = receive_value(s1);
5           if (test_success)
6                    %fprintf(1,'Received value: \t %d \n',test_value);
7           end
8           fflush(stdout);
9  end
10
11 meas_adc = mean(double(test_value(3:end)));
12 end
```

2.4.3 Arquivo send_value.m

```
1  function send_value(s1,value);
2  srl_write(s1,uint8(255));
3  srl_write(s1,uint8(value));
4  end
```

2.4.4 Arquivo receive_value.m

```
1  function [rx_value,rx_success] = receive_value(s1);
2  rx_success = 0;
3  rx_tries = 0;
4    while ((rx_tries < 10) && ~rx_success)
5              rx_tries = rx_tries + 1;
6              rx_int = srl_read(s1, 1); %array of ints. Each int receives one
                   byte
7              if (rx_int == 255)
8                        channel = srl_read(s1, 1);
9                        byte1 = srl_read(s1, 1);
```

```
10                      byte2 = srl_read(s1, 1);
11                      rx_value(channel) = uint16(byte1)*(2**8)+uint16(byte2);
12                      rx_success = 1;
13              end
14    end
15  end
```