

CERN

CERN SUMMER STUDENT 2015 - REPORT

GEM voltage supply and real-time monitoring

RD51

Physics - Detector Technologies

Vasilios Dimitris Karaventzas

August 2015

CERN

Abstract

Physics - Detector Technologies

CERN Summer Students 2015 - Report

GEM voltage supply and real-time monitoring

The operation of a GEM requires strong electrical fields that are able to initiate an electron avalanche. These electrical fields are generated by the application of high voltage in μm gaps. The scope of this project was to monitor the electrical parameters of the applied voltages, such as level and current flowing towards the detector. Therefore a real-time monitoring scheme was implemented. Additionally, a scheme for having all the voltages required for the operation of a GEM controllable, is proposed. The work conducted during this time-frame was targeted into developing the necessary parts for the operation of such device, whereas the integration of those into one functional circuit is to be conducted in a future work.

Keywords: GEM, High Voltage Monitoring, picoCurrent Monitoring, Controllable High Voltage

Contents

Abstract	i
Contents	ii
List of Figures	iii
1 Summer Student Report	1
1.1 Voltage Monitoring	1
1.2 GEM Current Monitoring	2
1.3 Controllable Voltage Generator	4
1.4 Utility Circuits	4
1.5 Design of the complete monitoring system	5
1.6 Conclusions & Future Work	6
A Serial Interface code	7
A.1 Arduino Code	7
A.2 Matlab Code	8

List of Figures

1.1	Voltage monitoring schematic	2
1.2	Voltage and Readout Current monitoring	2
1.3	Current Monitor Schematic	3
1.4	Diode Characterization for Current Monitoring	3
1.5	Controllable high voltage for GEM biasing	4
1.6	Positive to Negative -5V DC/DC Converter	5
1.7	Complete circuit design for power supply and monitoring	5
1.8	Arduino carrier board with the voltage and current measuring schemes	6

Chapter 1

Summer Student Report

The analysis of the operation of a gaseous detector, can become cumbersome task and is out of the scope of this report. Therefore, in this case, the gaseous detector physics will be simplified, i.e. the particle or photon interaction with the gas will be hereby not discussed, but instead it will be taken for granted that a free electron is generated in the gas. In order for this electron to be able to be studied in manner of energy and position, an amplification mechanism has to take place, such that at the end of the amplification stage, an adequate amount of charge is created. This is needed in order to have enough charge for the electronics to produce a substantially large signal. This amplification is done by strong enough electrical fields that can initiate an electron avalanche.

The electrical fields required for the amplification stage are generated by applying high voltage at the gaseous detector electrodes, each of which has its own special geometry. The report at hand focuses at the generation of these high voltages as well as the real-time observation of their amplitude and the current flowing towards the detector electrodes. Additionally, there will be also described some utility circuits that had to be implemented and tested, necessary for achieving the aforementioned two goals.

1.1 Voltage Monitoring

For the specific gaseous detector utilized during this project, triple GEM, 7 different voltage levels have to be generated from one high voltage supply through a resistive divider, and as the highest of them is the one of the voltage supply, the rest 6 of them have to be monitored. The scheme for this to be achieved can be seen below and 3 main parts can be distinguished, a resistive voltage divider, a voltage inverter, as the high voltages applied are negative and finally an Arduino based data acquisition system.

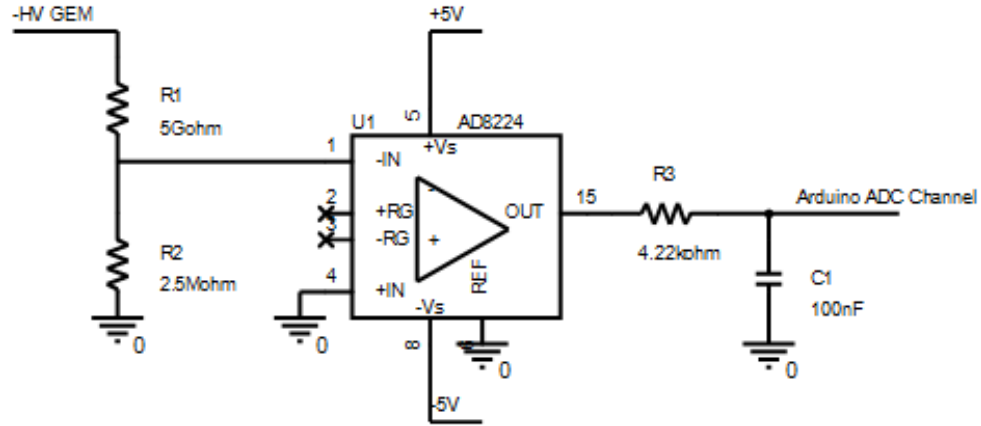


FIGURE 1.1: Voltage monitoring schematic

Due to the resistor divider, the voltage is attenuated by a factor of 1:2000, so it can meet the maximum voltage criteria of the Arduino (3.3V), which digitizes the value and through a serial port sends the data to a computer. Both the code for the Arduino, as well as the code to read the values and plot them in real time can be found in the Appendix A. Hereby, at Figure 1.2 the plotting of the GEM voltages as well as the current of the grounded readout electrode which was also monitored, can be depicted during the turning on of the GEM detector.

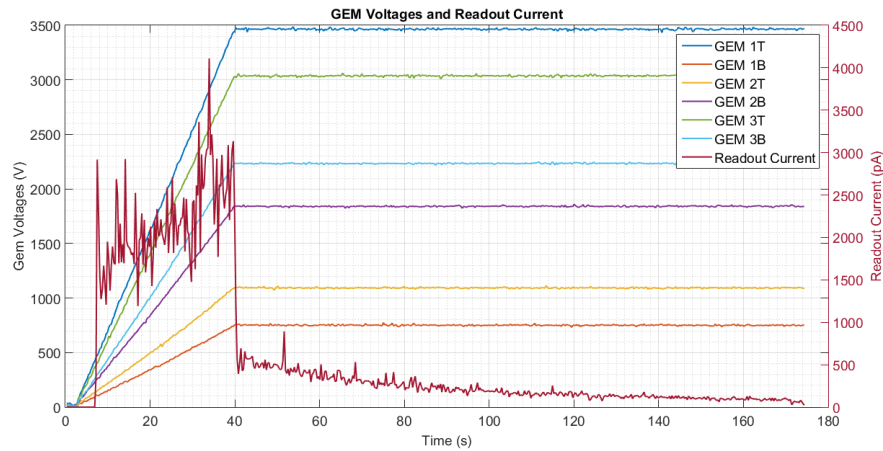


FIGURE 1.2: Voltage and Readout Current monitoring

1.2 GEM Current Monitoring

The monitoring of the current flowing to each GEM is also of great significance. Since this current is in the order of magnitude of 1pA - $1\mu\text{A}$, it cannot be monitored by the voltage drop on a resistor, therefore a diode is chosen and the voltage drop due to this current is monitored. The scheme of a differential measurement of the voltage can be seen in Figure 1.3

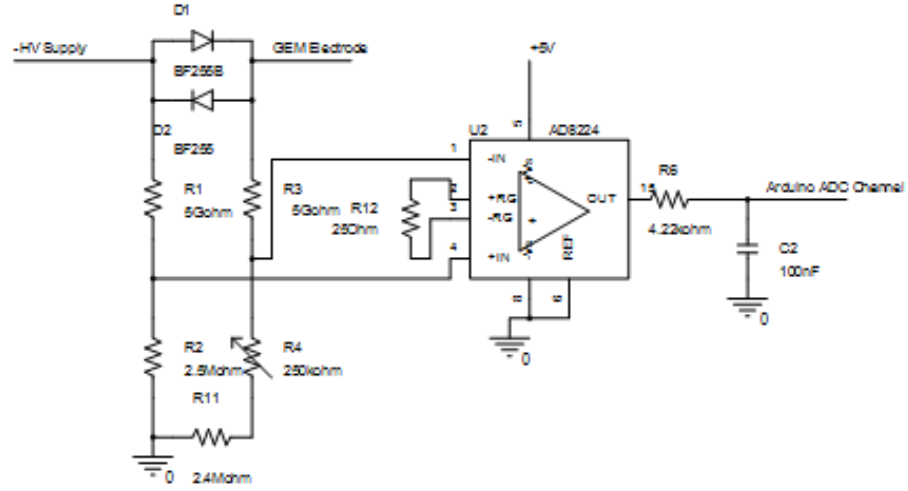


FIGURE 1.3: Current Monitor Schematic

There are 2 crucial parts that have to be taken into consideration for the correct operation of the current to operate. Firstly, since there is a voltage divider (1:2000) in the input of the operational amplifier, the output has to have a gain of 2000 in order to compensate for this attenuation. Additionally, the current-voltage characteristic of this particular diode has to be experimentally investigated and modeled, in order to correlate the measured voltage to the actual current. This can be observed in the following figure, where the assumption is made that the resistive behavior of the diode is negligible and therefore the V-I of the diode is completely exponential, according to $I = I_0 e^{\frac{V}{n k T}}$.

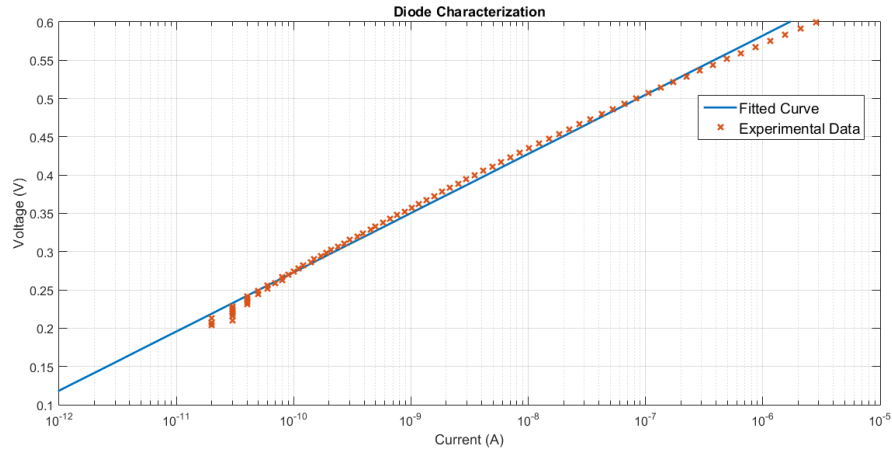


FIGURE 1.4: Diode Characterization for Current Monitoring

The equation that gives the V-I relation of this diode is found to be:

$$I = 2.937 \cdot 10^{-14} e^{\frac{V}{0.03357}} (A) \quad (1.1)$$

1.3 Controllable Voltage Generator

As mentioned earlier, the operation of a GEM requires 7 different high voltage levels. Currently, these are created with a single voltage generator, and a resistive voltage divider, but such a method, when it comes to investigating the operation of a GEM, makes the control of the voltage levels difficult, as the only solution is to re-solder different values of resistors. This could be overcome by creating a simple circuit that would control each voltage through a high voltage MosFET as can be seen in Figure 1.5.

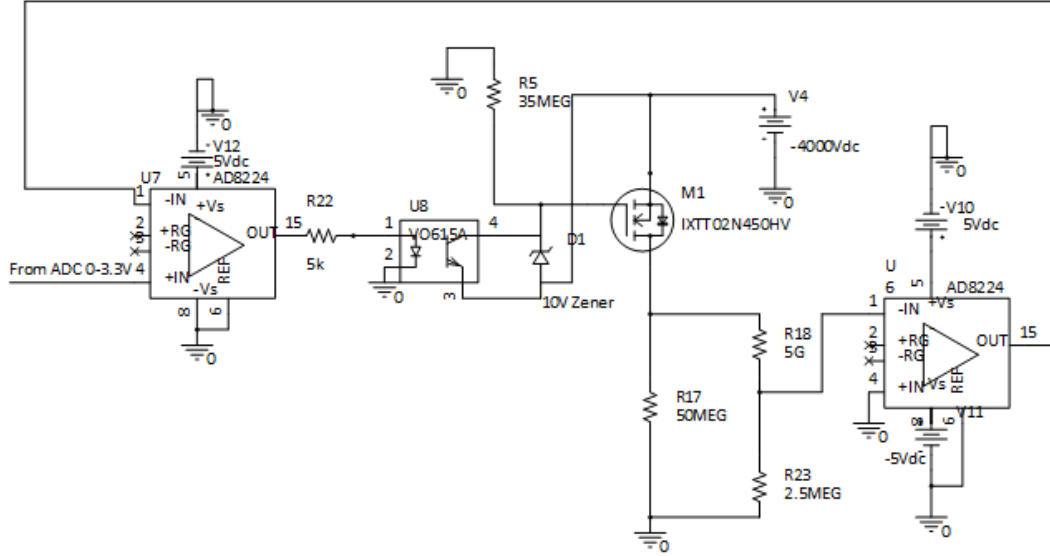


FIGURE 1.5: Controllable high voltage for GEM biasing

With this scheme although the output voltage is in the kV range, through an optocoupler and the two operational amplifiers, a negative feedback loop is implemented and a simple proportional controller is fabricated operating at a few volt range. The output has a linear correlation to the input, with the output gain being regulated only from the value of the resistive divider. Therefore in this particular schematic, it can be seen that $Gain = \frac{output}{input} = \frac{2.5 \cdot 10^6}{(2.5 \cdot 10^6 + 5 \cdot 10^9)} \approx 2000$. One issue to be taken into consideration hereby is the need for reducing the noise, as it can drive the system to unstable behavior.

1.4 Utility Circuits

It can be seen that most of the electronics required for the above mentioned reasons require 3 voltage biases, -5V, 0 and 5V, but usual commercially available power supplies provide only single sign voltages. Therefore, to avoid the need for special voltage supplies, a circuit to generate -5V from a positive 5V supply was designed and implemented. Such a circuit can be seen hereby. It has to be taken into consideration that the stability of the negative output voltage is of great significance and that the exact voltage that is output is strongly

dependent on the amount of current drawn. It has been found that the amount of current able to be delivered before a major drop on the negative voltage is observed is about $100mA$, with the voltage remaining around $-4.9V$.

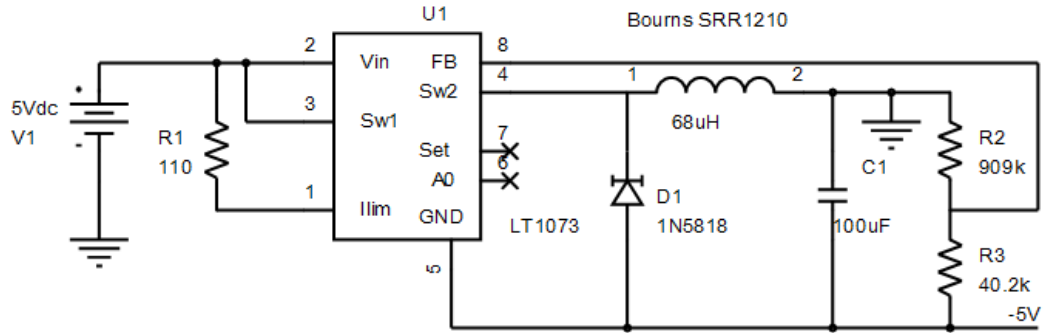


FIGURE 1.6: Positive to Negative -5V DC/DC Converter

Additionally, the monitoring of the current flowing through the active voltage divider, namely resistive divider acting as a reference voltage in a chain of BJTs, utilized currently, is required. For that, a BYV26E diode was placed in series to the divider and the ground and the equation describing its behavior is modeled in Equation 1.2.

$$I = 3.23 \cdot 10^{-9} e^{25.42V} + 2.7 \cdot 10^{-5} (A) \quad (1.2)$$

1.5 Design of the complete monitoring system

The previously described circuits are essential components of the upcoming novel voltage supply and monitoring device. It can be seen in the figures bellow the design of this novel supply as well as each of the previously described circuits depicted and pointed out.

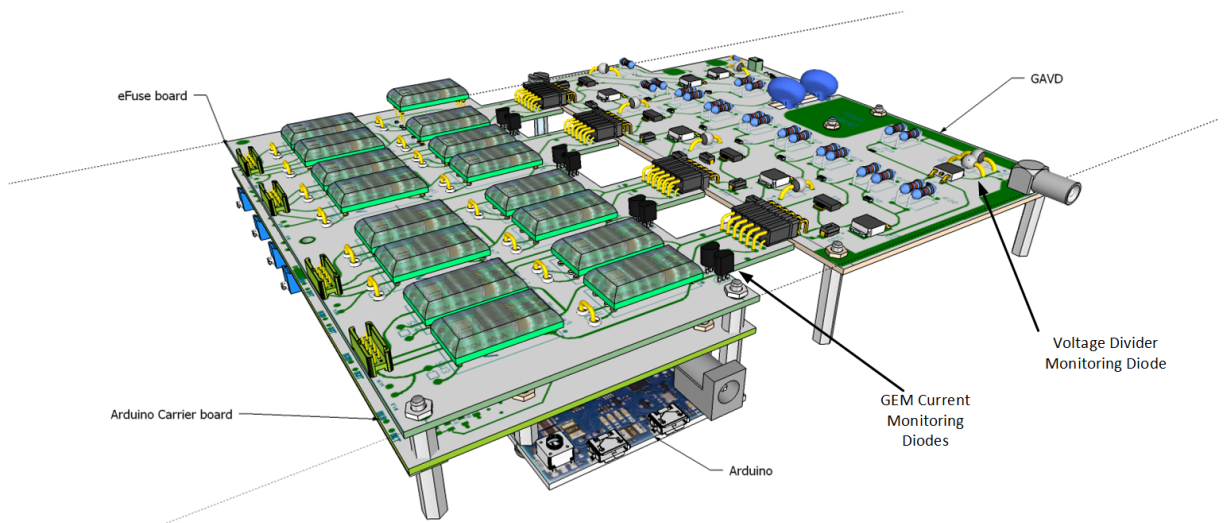


FIGURE 1.7: Complete circuit design for power supply and monitoring

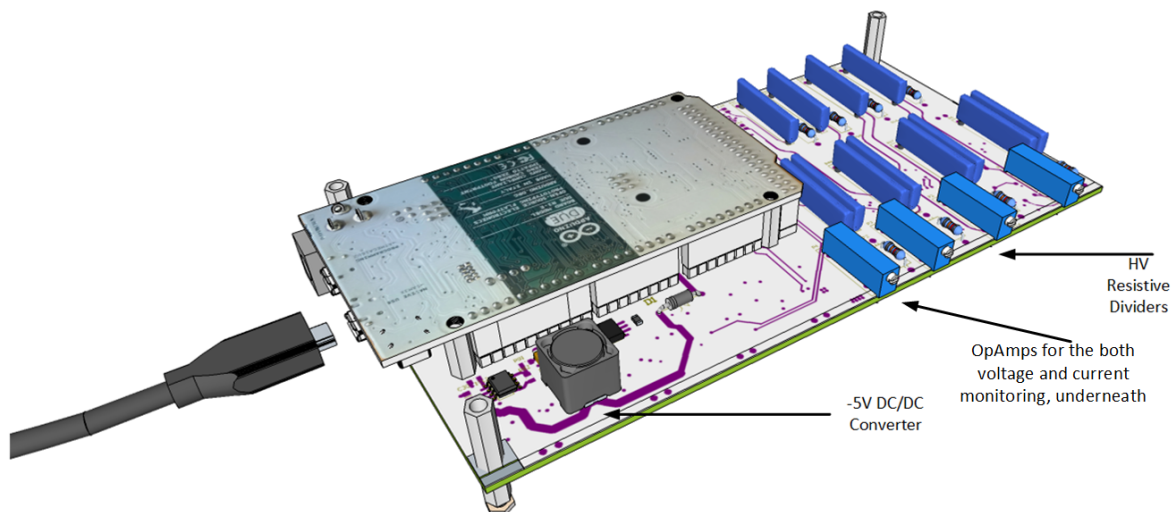


FIGURE 1.8: Arduino carrier board with the voltage and current measuring schemes

In Figure 1.7 the complete circuit can be seen, where on the right the voltage divider is pictured, essential for generating the 7 different voltage levels required for the operation of a triple GEM. On the left, the high voltage side is depicted, where the outputs on the left are the inputs to the GEM. Underneath it, the arduino carrier board can be seen. This board is pictured in greater detail in Figure 1.8. Here, the low voltage part of the circuit is observed and the resistive dividers alongside to the DC/DC converter can be seen.

1.6 Conclusions & Future Work

The previously described circuits are necessary parts for the design of a novel voltage supply, for the operation of a GEM, where all the crucial electrical parameters should be monitored.

During the period of the summer student program the above mentioned circuits were implemented and for each of those a proof of concept was conducted.

As a future work, these parts have to be brought together and be integrated in one single circuit and be tested as a whole in a real experimental setup. Also, the stability and the output ripple of the controllable voltage generator has to be investigated. When this is done, an appropriate DAC chip has to be purchased and a program in a μ Controller has to be written to control both the output as well as by monitoring the voltages, as done during this project, to decide whether the operation of the voltage supply is nominal or not. If this decision is made, then there could be produced a TTL signal to shut down the high voltage supply.

Appendix A

Serial Interface code

A.1 Arduino Code

```
1  #define n_rep 10
2  void setup() {
3      Serial.begin(115200);
4      analogReadResolution(12);
5      //For low input voltages choose a preamplifier for the ADC input
6      ADC->ADC_MR = ADC->ADC_MR|0x00800000; //Allow to Write on ADC_CGR
7      ADC->ADC_CGR = 0x0000000B; // Each pair of bits changes the gain for each
      //ADC preamplifier between 1,2 or 4. Here ADC0 has gain of 4, ADC1 of 2
      //and the rest 1.
8  }
9  void loop()
10 {
11     float a=0,b=0,c=0,d=0,e=0,f=0,cur=0;
12     for (int rep=0;rep<n_rep;rep++) { //Averaging over 10 samples
13         a+=analogRead(A1)*1.5; //1:1500 divider chosen instead of 1:1000
14         b+=analogRead(A3);
15         c+=analogRead(A4);
16         d+=analogRead(A5);
17         e+=analogRead(A6)*1.5;
18         f+=analogRead(A7);
19         cur+=analogRead(A9);
20         delay(1);
21     }
22     a=a/n_rep;
23     b=b/n_rep;
24     c=c/n_rep;
25     d=d/n_rep;
26     e=e/n_rep;
```

```

27     f=f/n_rep;
28     cur=cur/n_rep;
29     Serial.print(a); // Print everything in tab separated format
30     Serial.print("\t");
31     Serial.print(b);
32     Serial.print("\t");
33     Serial.print(c);
34     Serial.print("\t");
35     Serial.print(d);
36     Serial.print("\t");
37     Serial.print(e);
38     Serial.print("\t");
39     Serial.print(f);
40     Serial.print("\t");
41     Serial.print(cur);
42     Serial.print("\n"); //EOL for package transimition ending
43     delay(200); //Allow time for transmitting
44 }

```

A.2 Matlab Code

```

1  clear
2  clc
3  answer = inputdlg('COM Port?','Give the Com Port') %Choose Port
4  %User Defined Properties
5  serialPort = answer;          % define COM port #
6  plotTitle = 'GEM Voltages'; % plot title
7  xLabel = 'Elapsed Time (s)'; % x-axis label
8  yLabel = 'Voltage in kV';    % y-axis label
9  plotGrid = 'on';            % 'off' to turn off grid
10 min = 0;                     % set y-min
11 max = 3.8e3;                 % set y-max
12 scrollWidth = 60;            % display period in plot, plot entire data log
    if <= 0
13 delay = 1e-5;                % make sure sample faster than resolution
14 %Define Function Variables
15 time = 0;
16 data = zeros(7,1);
17 count = 0;
18 con=3.3e3/4095; %Integer value to actual voltage mapping constant
19
20 subplot(2,1,1) % Create on plot for the voltages
21 %Set up Plot
22 plotGraph = plot(time,data(1,:),'-or',...

```

```

23 'LineWidth',2,...
24 'MarkerFaceColor','w',...
25 'MarkerSize',2);
26 hold on
27 plotGraph1 = plot(time,data(2,:),'-og',...
28 'LineWidth',2,...
29 'MarkerFaceColor','w',...
30 'MarkerSize',2);
31 hold on
32 plotGraph2 = plot(time,data(3,:),'-ok',...
33 'LineWidth',2,...
34 'MarkerFaceColor','w',...
35 'MarkerSize',2);
36 hold on
37 plotGraph3 = plot(time,data(4,:),'-ob',...
38 'LineWidth',2,...
39 'MarkerFaceColor','w',...
40 'MarkerSize',2);
41 hold on
42 plotGraph4 = plot(time,data(5,:),'-oy',...
43 'LineWidth',2,...
44 'MarkerFaceColor','w',...
45 'MarkerSize',2);
46 hold on
47 plotGraph5 = plot(time,data(6,:),'-oc',...
48 'LineWidth',2,...
49 'MarkerFaceColor','w',...
50 'MarkerSize',2);
51 hold off
52
53 h_leg=legend('a','b','c','d','e','f','Location','northwest');
54
55 set(h_leg,'FontSize',14);
56
57 title(plotTitle,'FontSize',25);
58 xlabel(xLabel,'FontSize',15);
59 ylabel(yLabel,'FontSize',15);
60 ylim([min max])
61 grid(plotGrid);
62 grid('minor');
63
64 subplot(2,1,2) % Create a second plot for the current
65 plotGraph6 = plot(time,data(7,:),'-ob',...
66 'LineWidth',2,...
67 'MarkerFaceColor','w',...

```

```

68  'MarkerSize',2);
69  axis([0 10 0 800]);
70  grid(plotGrid);
71  grid('minor');
72  h_c=legend('c');
73  set(h_c,'FontSize',14);
74
75  %Open Serial COM Port
76  s = serial(serialPort, 'BaudRate', 115200);
77  disp('Close Plot to End Session');
78  fopen(s);
79
80  tic
81
82  while ishandle(plotGraph) && ishandle(plotGraph2) && ishandle(plotGraph1)
      %Loop when Plot is Active
83
84  dat = fscanf(s,'%f')*con; %Read Data from Serial as Integer & Translate to
      actual values
85
86  dat(7,:)=0.005*exp(0.033*dat(7,:)); %Diode V-I mapping
87
88  if(~isempty(dat) && isfloat(dat)) %Make sure Data Type is Correct
89  count = count + 1;
90  time(count) = toc; %Extract Elapsed Time in seconds
91  data(:,count) = dat(:,1); %Extract 1st Data Element
92
93  %Set Axis according to Scroll Width
94  if(scrollWidth > 0)
95  set(plotGraph,'XData',time(time > time(count)-scrollWidth),...
96  'YData', data(1,time > time(count)-scrollWidth));
97  set(plotGraph1,'XData',time(time > time(count)-scrollWidth),...
98  'YData', data(5,time > time(count)-scrollWidth));
99  set(plotGraph2,'XData',time(time > time(count)-scrollWidth),...
100 'YData', data(6,time > time(count)-scrollWidth));
101 set(plotGraph3,'XData',time(time > time(count)-scrollWidth),...
102 'YData', data(4,time > time(count)-scrollWidth));
103 set(plotGraph4,'XData',time(time > time(count)-scrollWidth),...
104 'YData', data(3,time > time(count)-scrollWidth));
105 set(plotGraph5,'XData',time(time > time(count)-scrollWidth),...
106 'YData', data(2,time > time(count)-scrollWidth));
107 xlim([time(count)-scrollWidth time(count)]);
108
109 set(plotGraph6,'XData',time(time > time(count)-scrollWidth),...
110 'YData', data(7,time > time(count)-scrollWidth));

```

```

111
112 h_leg.String={strcat('1T: ',num2str(dat(1,1))),strcat('1B:
      ',num2str(dat(5,1))),strcat('2T: ',num2str(dat(6,1))),strcat('2B:
      ',num2str(dat(4,1))),strcat('3T: ',num2str(dat(3,1))),strcat('3B:
      ',num2str(dat(2,1)))}; %Display at the legend the latest received value
113 h_c.String={strcat('Current: ',num2str(dat(7,1)),'pA')};
114 else
115 set(plotGraph,'XData',time,'YData',data(1,:));
116 set(plotGraph1,'XData',time,'YData',data(5,:));
117 set(plotGraph2,'XData',time,'YData',data(6,:));
118 set(plotGraph3,'XData',time,'YData',data(4,:));
119 set(plotGraph4,'XData',time,'YData',data(3,:));
120 set(plotGraph5,'XData',time,'YData',data(2,:));
121 %set(plotGraph6,'XData',time,'YData',data(7,:));
122 axis([0 time(count) min max]);
123 end
124
125 if count>5000 %Clear buffer after 5000 samplers
126 time = 0;
127 data = zeros(7,1);
128 count = 0;
129 end
130
131 %Allow MATLAB to Update Plot
132 pause(delay);
133 end
134 end
135 fclose(s);

```