Using MATLAB in the Teaching and Learning of Semiconductor Device Fundamentals

Ian Grout and Abu Khari Bin A'ain

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/46461

1. Introduction

Mathematical analysis tools provide an invaluable (and sometime essential!) tool for use within the engineering disciplines and are readily found in education, research and industrial applications. For example, within the industrial applications, mathematical analysis tools provide an essential aid at all stages of a product development from design through manufacture to test. Although there are a number of useful tools available, since its inception, MATLAB [1] has found a unique role within the engineering disciplines. Given the need to utilise this tool ultimately in both a research context and an industrial application context, there is a need to introduce students at the university level to the effective use of MATLAB, with a focus on the particular discipline area of the student.

In this chapter, the use of MATLAB is presented and discussed within a university education context and in particular the integration of MATLAB into the teaching and learning of semiconductor device fundamentals for electronic and computer engineering students. The aim is to support the student learning of semiconductor device operation, primarily diodes (silicon, germanium, Schottky barrier and Zener) and transistors (bipolar junction transistors (BJTs), junction field effect transistors (JFETs) and metal-oxide semiconductor field effect transistors (MOSFETs)).

MATLAB is primarily used as a data analysis, presentation and reporting tool in this context, but the natural integration of MATLAB into the teaching and learning environment has two real purposes:

1. Firstly, it is an introduction to the tool for generic engineering and scientific design and data analysis.
2. Secondly, it is used to support the learning of semiconductor device operation.

The basic idea is that experiments are undertaken on practical devices, the results obtained are then analysed in MATLAB and finally compared to the ideal device (mathematical) models. Hence, within MATLAB, actual data is used and mathematical models of ideal devices are developed. This is aimed as an introduction (targeting first year undergraduate students) to both semiconductor devices and to mathematical analysis tools (here MATLAB which would then be used by the students later on in more advanced subjects).

Three different teaching and learning scenarios are presented and the integration of MATLAB into a computer aided learning (CAL) environment that has been custom developed are provided:

Firstly, students undertake experiments in a *traditional learning scenario.* In the laboratory, electronic circuits using semiconductor devices are built and tested. Experimental results are then taken and analysed using MATLAB; specifically, results are entered into arrays (the term *array* used here to mean a *1 x m* matrix) within MATLAB which then allows these results to be analysed and graphically plotted. These results are also compared to the ideal mathematical equations for the devices considered (specifically diodes and transistors). Hence, the learning experience naturally includes an introduction to concepts such as scalar types and arrays (in a generic context, matrices and matrix manipulation), building and manipulating equations, manipulating experiment results, results comparison, graphical plotting and m-files. The student therefore gains experience in both electronic hardware build and test, and results analysis using MATLAB. This is suitable for electronic and computer engineering students at an introductory level. This idea is depicted in figure 1.

Secondly, students undertake experiments using *computer aided learning (CAL) environment*. The experiment electronic hardware is pre-built and connected to a PC via an *experiment interface electronics* unit (essentially a computer port connection such as RS-232 (readily extended to USB) interface that allows for analogue voltages to be created and sampled in the same manner as would manually be done, but now through a software graphical user interface (GUI)). The student therefore gains experience in computer control of experiments and results analysis using MATLAB. This is suitable for electronic and computer engineering students at an introductory level who would not necessarily need to physically build electronic hardware. This idea is depicted in figure 2.

Thirdly, students undertake experiments via a *distance mode of learning* in that they access the experiment electronic hardware and MATLAB via an Internet browser. This arrangement forms a *remote laboratory* whereby the experiment is controlled and results accessed remotely and via an Internet browser. Essentially, the CAL arrangement identified in figure 2 is "web enabled" – that is made accessible via the Internet. The student gains therefore experience in computer control and results analysis using MATLAB, but in a distance mode of learning. This idea is depicted in figure 3.

The above three teaching and learning scenarios provide ways in which MATLAB can be integrated into a flexible teaching and learning environment. However, given that the idea here is that both the use of MATLAB and the electrical characteristics of basic semiconductor devices are to be introduced, the structure of the laboratory experiments must be carefully considered. For example, it would be necessary to introduce the basic concepts of MATLAB as well as the key commands to be used before attempting to analyse experiment data. One possible laboratory experiment flow is shown in figure 4.

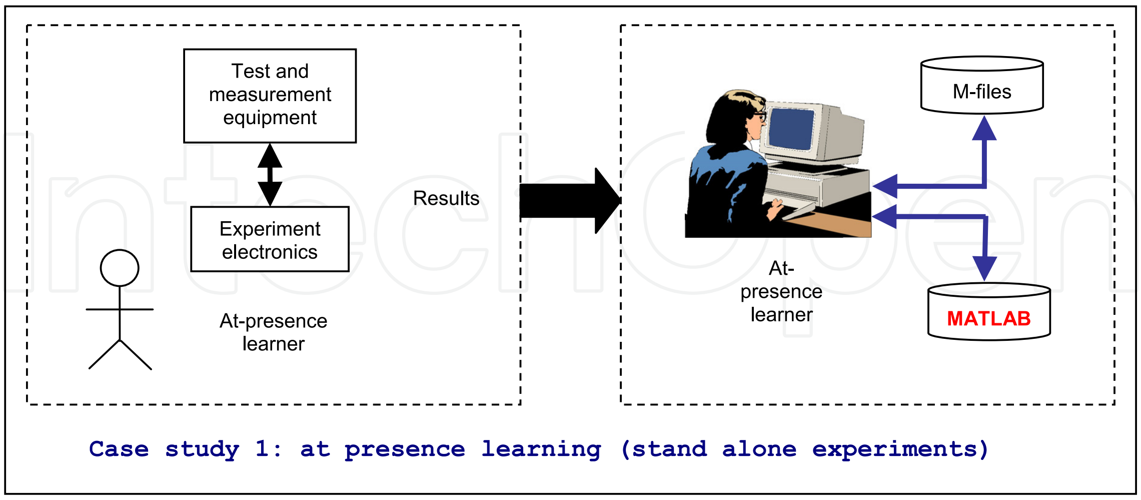

Test and measurement equipment
Experiment electronics
Results
At-presence learner
M-files
At-presence learner
MATLAB
Case study 1: at presence learning (stand alone experiments)

Figure 1. The traditional learning scenario

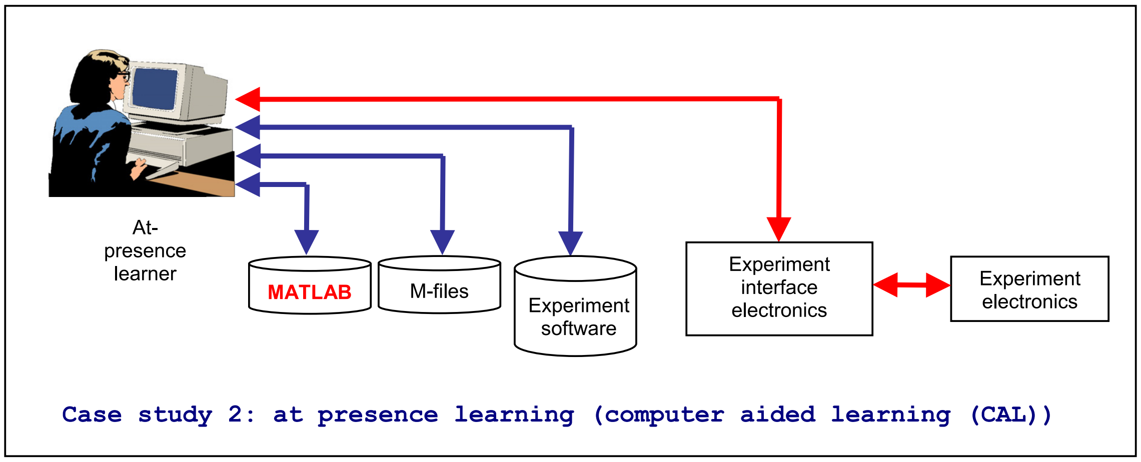

At-presence learner
MATLAB
M-files
Experiment software
Experiment interface electronics
Experiment electronics
Case study 2: at presence learning (computer aided learning (CAL))

Figure 2. The computer aided learning scenario

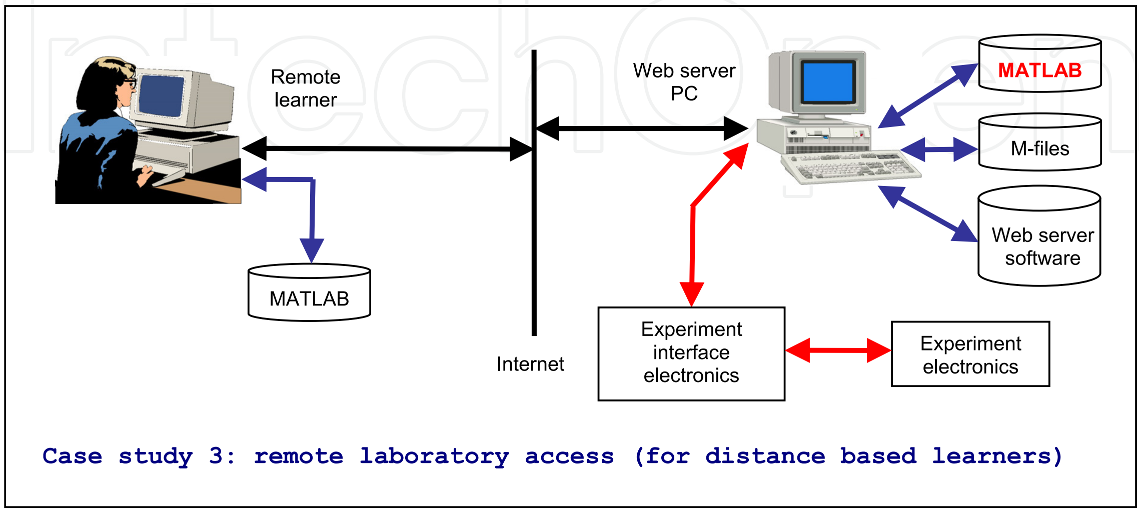

Remote learner
Web server PC
MATLAB
M-files
Web server software
MATLAB
Internet
Experiment interface electronics
Experiment electronics
Case study 3: remote laboratory access (for distance based learners)

Figure 3. Remote user access via an Internet browser learning scenario

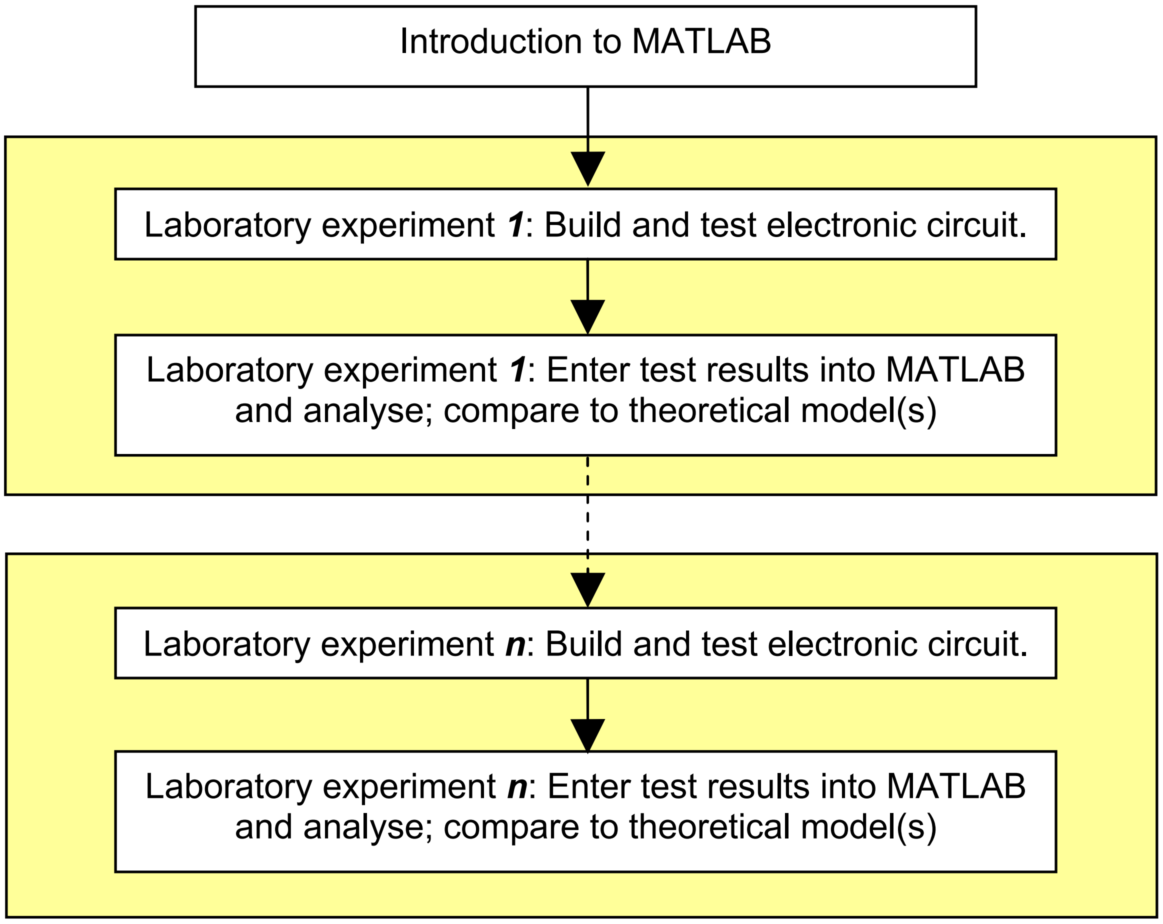

Introduction to MATLAB
Laboratory experiment *1*: Build and test electronic circuit.
Laboratory experiment *1*: Enter test results into MATLAB and analyse; compare to theoretical model(s)
Laboratory experiment *n*: Build and test electronic circuit.
Laboratory experiment *n*: Enter test results into MATLAB and analyse; compare to theoretical model(s)

Figure 4. Laboratory experiment "flow"

The development of the hardware-software infrastructure and use of the three above teaching and learning scenarios are introduced here with reference to an experiment consisting of a BAT86 Schottky barrier diode [2] and are presented in this chapter. The remainder of the chapter is structured as follows:

Section 2: The use of MATLAB within an education environment

The use of MATLAB as an aid to teaching and learning for a wide range of engineering and scientific applications is presented. A rationale for using MATLAB is provided and how it may be used is identified. Reference is made to the teaching and learning in the computer and electronic engineering disciplines.

Section 3: Teaching and learning semiconductor device fundamentals

The teaching of semiconductor device fundamentals at an introductory level within the university sector is presented with reference to current teaching undertaken by the authors. The need for teaching semiconductor devices and their application in the electronics and microelectronic industries is provided, along with the need to relate the theory to real (practical) devices through the use of suitable laboratory experiments undertaken by the students. The use of MATLAB as an integrated mathematical analysis tool is presented where theory and practice are compared.

Section 4: Case study 1: at presence learning (stand alone experiments)

The use of MATLAB is presented whereby physical circuits with the semiconductor devices of interest are developed and tested by the student. Results are then entered into MATLAB and analysis undertaken, comparing the real devices with their mathematical ideal models. Current and voltage relationships are then identified. The student gains hands-on experience with both electronic hardware and computer based software.

Section 5: Case study 2: at presence learning (computer aided learning (CAL))

The use of MATLAB is presented as in case study 1, but now pre-built experiments are accessed through a custom software application and the experiments are accessed via a PC interfaced electronic hardware arrangement. The student concentrates on the MATLAB and software side of the experiment activity.

Section 6: Case study 3: remote laboratory access (for distance based learners)

The use of MATLAB is presented as in case study 2, but now the interface is via a remote laboratory arrangement, accessed via an Internet browser and web server arrangement. With this arrangement, remote learners are supported.

Section 7: Conclusions

Conclusions to the work that has been undertaken are presented.

Section 8: References

References used in the development of the chapter are provided.

2. The use of MATLAB within an education environment

2.1. Introduction

MATLAB is an invaluable tool for use within the engineering disciplines for education, research and industrial purposes. In this chapter, the use of MATLAB is presented and discussed within a university education context. MATLAB is an almost universal tool for engineering education. It provides a cost-effective *what if* platform where users can manipulate and explore functions to discover and explore the response of a system. Here, the system is an electronic circuit where the focus of the circuit operation is discovering and exploring the behaviour of semiconductor devices.

2.2. Why use MATLAB?

MATLAB is a high-level language and interactive environment that enables a user to perform computationally intensive engineering and scientific calculations tasks faster than with traditional programming languages such as C. It includes a set of integrated graphics and plotting capabilities allowing users to visualise their data and analysis results and which can also be extended by the user to suit his or her own needs. As such, it provides the student and practising engineer with a suite of useful tools for analysing and solving engineering related problems. For semiconductor devices made from semiconductor materials such as silicon, which the work discussed in this chapter are aim at exploring, MATLAB is the perfect tool to use as it allows the student to undertake directed and self study activities, even outside laboratory. As the calls for innovation and creativity become stronger, students cannot afford to limit their experiments and exploration within physical laboratory. By integrating MATLAB within experiments and the course syllabus, it supports self-directed learning and also does not cost anything to make mistakes!

2.3. Important concepts to introduce

With the integration of MATLAB into a course syllabus, there is a need to identify the key concepts to introduce and for the students to practice. It is therefore important for the course developer to ensure that there is a seamless integration of MATLAB into the course syllabus and for there to be a clear focus on why and how this mathematical analysis tool is used. Therefore the course developer needs to consider a wide range of aspects including:

1. The role of MATLAB
   Why is MATLAB utilised in the course with a focus on the engineering discipline concerned? How would there be a suitable and seamless integration of the analysis tool with the core engineering topics in the course? How much time should be allocated to the teaching and learning of MATLAB core concepts Vs the use of MATLAB to solve engineering problems?
2. What is important for electronic and computer engineering students
   Why utilise mathematical analysis tools in electronic and computer engineering and how can they be used to support the practicing engineer? With MATLAB being introduced to the students for the first time, how can this support more advanced engineering topics? For example, MATLAB with its toolbox Simulink is widely used in control engineering and where students are introduced to control engineering concepts, their knowledge of MATLAB from this introductory course could be used to allow the teaching of the control engineering to concentrate on using MATLAB rather than reintroducing the core MATLAB concepts.
3. Consider a stand-alone module (i.e., just MATLAB) or integrate MATLAB into subject (as considered here)
   The introduction to students of MATLAB can be either the main focus of a course whereby the introduction to MATLAB is the purpose of the course, or MATLAB can be introduced as a tool to use in supporting engineering disciplines. Whilst allocating a complete course to MATLAB would allow students to consider both the introductory concepts and the more advanced concepts (such as the use of the toolboxes), it might not necessarily provide a link to the use of this tool in solving engineering discipline specific problems. It also means that valuable and restricted time within the overall programme of study (the available time needs to be allocated to many different aspects of engineering) which should be focused on the specific engineering discipline is not necessarily allocated to the focus area of the overall programme of study. The alternative approach, as considered here, is to provide a more generic introduction to

the tool before using it to solve problems relating to a specific electronic and computer engineering discipline, namely semiconductor devices.

4. Navigating the MATLAB desktop
   The MATLAB desktop provides the method in which a user interfaces to MATLAB and hence a working knowledge of the desktop must be obtained. However, learning the structure of the desktop should not become the focus of the learning and so detract from learning how to use the tool to solve real engineering problems.

5. Dealing with errors
   When learning how to use any software application and a new language, errors in the use of the software application, along with syntax and semantic errors with the language will inevitably be experienced by the student. How to deal with these errors can be a daunting task for a student and so the prompt correcting of these errors would be important. It would be expected that there would be a common set of errors encountered by many students and so many errors should be readily identifiable and corrected.

6. Matrix manipulation
   Within MATLAB, everything is treated as a matrix. Hence, the students would need to revise their previous learning of matrices and apply the concepts within the MATLAB environment, learning how to create and manipulate matrices using the native syntax. For example, a common problem encountered when learning how to use MATLAB is in the multiplication of matrices (for example, such as determining the square of an $n \times m$ matrix named y if attempting to use the command y^2 directly and y is not a square matrix).

7. Command line entry Vs m-files
   The starting point of learning the tool is how to effectively use the MATLAB command line for data and command entry. Once the basic concepts are learnt, the use of m-files (both script m-files and function m-files [3]) can then be introduced and from there onwards, m-files may become a more convenient manner in which to enter data and commands.

8. Arithmetic operators
   The use of arithmetic operators (addition, subtraction, multiplication, division, left division, power) when considering scalar values (1×1 matrices) and matrices ($n \times m$ matrices). How the arithmetic operations are undertaken – operations undertaken on the complete matrix or matrix element-by-element in turn. This requires a working knowledge of matrix algebra.

9. Logical operators
   The use of logical operators (less than, less than or equal to, greater than, greater than or equal to, equal to, not equal to, logical AND, logical OR) for determining conditions of the variables within the MATLAB workspace.

10. Functions
    A number of mathematical functions are available to create equations (including absolute value, square root, sine (value in radians), cosine (value in radians), tangent

(value in radians), the exponential operator, the logarithmic operator, the value of pi (Π), the imaginary unit (i or j = √(-1))).

11. Program (flow) control
    Program (flow) control allows for the development of scripts that control how the script (program) operates depending on specific conditions. There are three types of control statement in MATLAB, along with a program termination statement: **conditional control**, **loop control**, **error control** and the program termination statement **return**. **Conditional control** statements are *if* (together with *else* and *elseif*) and *switch* (together with *case* and *otherwise*) to execute MATLAB statements based on some logical condition. Loop control statements are *for* to execute MATLAB statements a fixed number of times, *while* to execute MATLAB statements an indefinite time based on some logical condition and *continue* to pass control to the next iteration of the *for loop* or *while loop* in which it appears and skips any remaining statements in the body of the loop. **Error control** (*try … catch*) changes the flow control if an error is detected during execution. The program termination statement **return** causes execution to return to the invoking function.

12. 2D and 3D plotting
    MATLAB includes a large number of plotting functions which allow the user to view their data as both two-dimensional (2D) plots and three-dimensional (3D) plots.

13. MATLAB toolboxes
    Whilst probably not appropriate to include in an introductory course, the MATLAB toolboxes such as Simulink and DSP toolbox provide for powerful extensions to the basic MATLAB commands and would typically be used in more advanced courses. For example, Simulink is widely used in the control engineering discipline and, with its block diagram graphical model generation approach, provides for a useful and important tool for the engineer.

14. Integration of other languages
    Whilst probably not appropriate to include in an introductory course, the ability to create MATLAB scripts which interact with code developed in other languages (such as FORTAN, C/C++ and Java) provide for a useful extension and enhanced flexibility in the use of MATLAB for system modelling and analysis.

15. Graphical user interface (GUI) development
    MATLAB provides for the ability to create graphical user interfaces which allow the user to interact with MATLAB through a suitable user interface rather than the command prompt or directly writing m-files.

16. Dealing with terminology
    Finally, as with anything that the engineer is involved in, there is a need to learn and correctly use the terminology specific to the domain that is being worked in.

3. Teaching and learning of semiconductor device fundamentals

For electronic and computer engineers, the use of semiconductor devices is integral to everything that they do, whether they design electronic circuits using semiconductor devices or program processors to control specific electronic circuits. Whilst practising engineers may not necessarily investigate the physics of the devices on a day-to-day basis, it is essential that they understand the behaviour of the devices at the material level (how they behave and why) in order to use these devices effectively within the electronic circuits they design or use. There are therefore two key aspects to the teaching and learning of semiconductor device fundamentals:

1. The **theory underpinning the device operation** – what is happening at the physical material level and how the behaviour can be related to device terminal behaviour in terms of voltages and currents (the development of theoretical mathematical models for idealised and more realistic device models).
2. How the **device terminal behaviour in terms of voltages and currents** (using the developed idealised and more realistic device models) can be used in practical and useful electronic circuits.

For example, consider an introduction to diodes. Both semiconductor (p-n junction) and Schottky barrier (metal-semiconductor contact) diodes are encountered and so must be introduced. The concepts identified in theses devices (such as a.c. signal rectification) would then be extended to more complex devices such as transistors, thyristors, triacs and integrated circuits (ICs).

Consider the Schottky barrier diode. The initial starting point is the structure and underlying mathematical equations (current-voltage (I-V) relationship) for this device. These are summarised in figure 5.

The starting point for understanding the device operation would need to introduce semiconductor materials, what their properties are and how they can be considered to behave (electrons and holes as charge carriers, intrinsic and extrinsic semiconductors, and the effects material doping [4, 5]). Based on these principles then the behaviour of the metal-semiconductor contact can be introduced and developed:

1. The behaviour of the materials around the contact junction and away from the contact junction at thermal equilibrium and non-equilibrium conditions (forward bias and reverse bias conditions).
2. The differences between ohmic and rectifying contacts. It is the rectifying contact (allowing current to flow through the metal-semiconductor contact under forward bias conditions but blocking current flow under reverse bias conditions) that would be of interest in the Schottkty barrier diode discussions.
3. The current-voltage (I-V) relationship at forward and reverse bias conditions at the anode and cathode device terminals would be developed and the use of the diode in

electronic circuits would be introduced and discussed. Reference would initially be made to a rectifier circuit using the diode and a resistor connected in series as shown in figure 6(a) [note that the standard diode symbol is shown here rather than the Schottky barrier diode symbol].

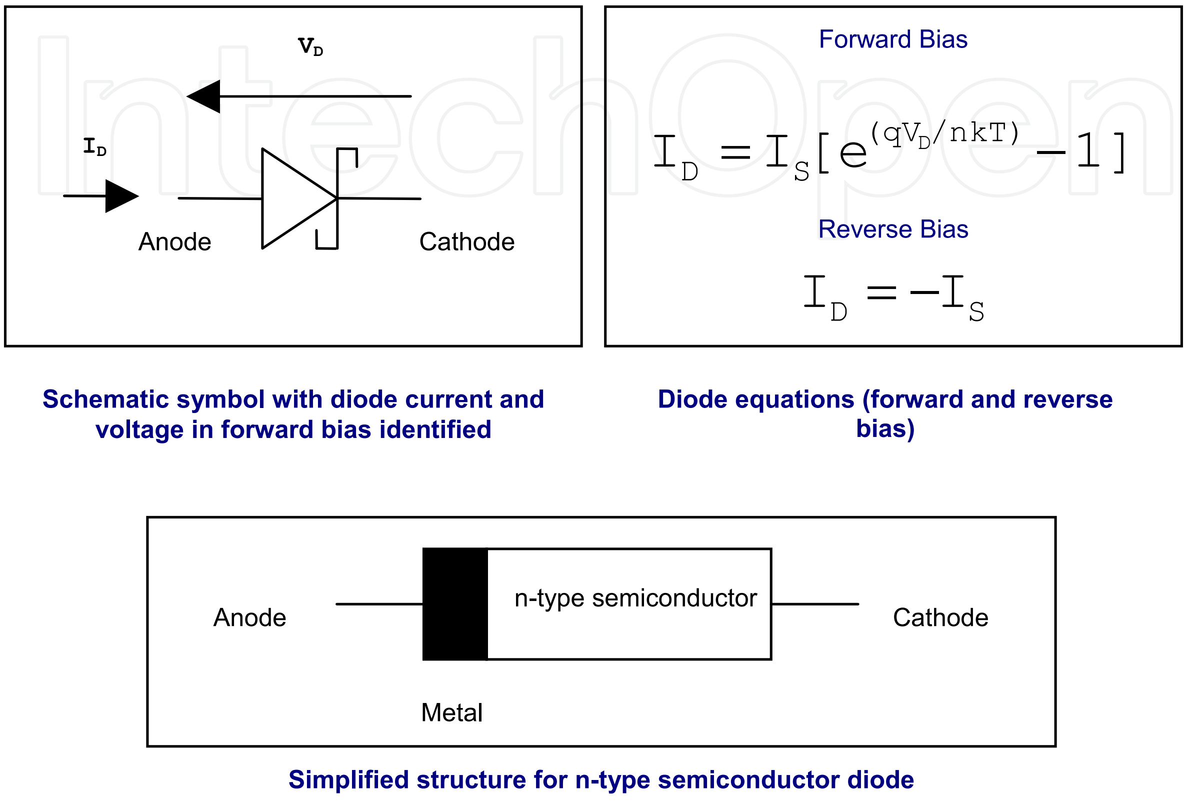

V_D
I_D
Anode
Cathode
Forward Bias
$I_D = I_S[e^{(qV_D/nkT)} - 1]$
Reverse Bias
$I_D = -I_S$
Schematic symbol with diode current and voltage in forward bias identified
Diode equations (forward and reverse bias)
Anode
n-type semiconductor
Cathode
Metal
Simplified structure for n-type semiconductor diode

Figure 5. Schottky barrier diode summary (n-type semiconductor type)

With reference to the diode rectifier circuit (figure 6(a)), this can be modelled mathematically for both forward bias and reverse bias in MATLAB to show the principle of operation. A sample m-file for modelling and simulating the Schottky barrier diode is shown in listing 1 and the output plot is shown in figure 7. Note that this code works for versions of MATLAB after v6.5. Here:

1. An idealised diode model is created which has a forward voltage drop of 0.3 V (*Vd*) when conducting (line 14).
2. The resistor (*R*) is set to 2 Ω for illustration purposes (line 15).
3. A sine wave voltage source (*Vin*) is created with an amplitude of 1 V and ten complete cycles of the sine wave are generated (lines 17 to 22).
4. The resistor voltage drop in forward bias and reverse bias is calculated (lines 24 to 30).
5. The resistor current and hence the diode current (*Id*) is calculated (line 32).
6. Four sub-plots are created with time along the x-axis (lines 38 to 73). There is then a delay of five seconds before the script code continues (line 74).
7. The plotted values are scrolled across the sub-plots (lines 80 to 90).

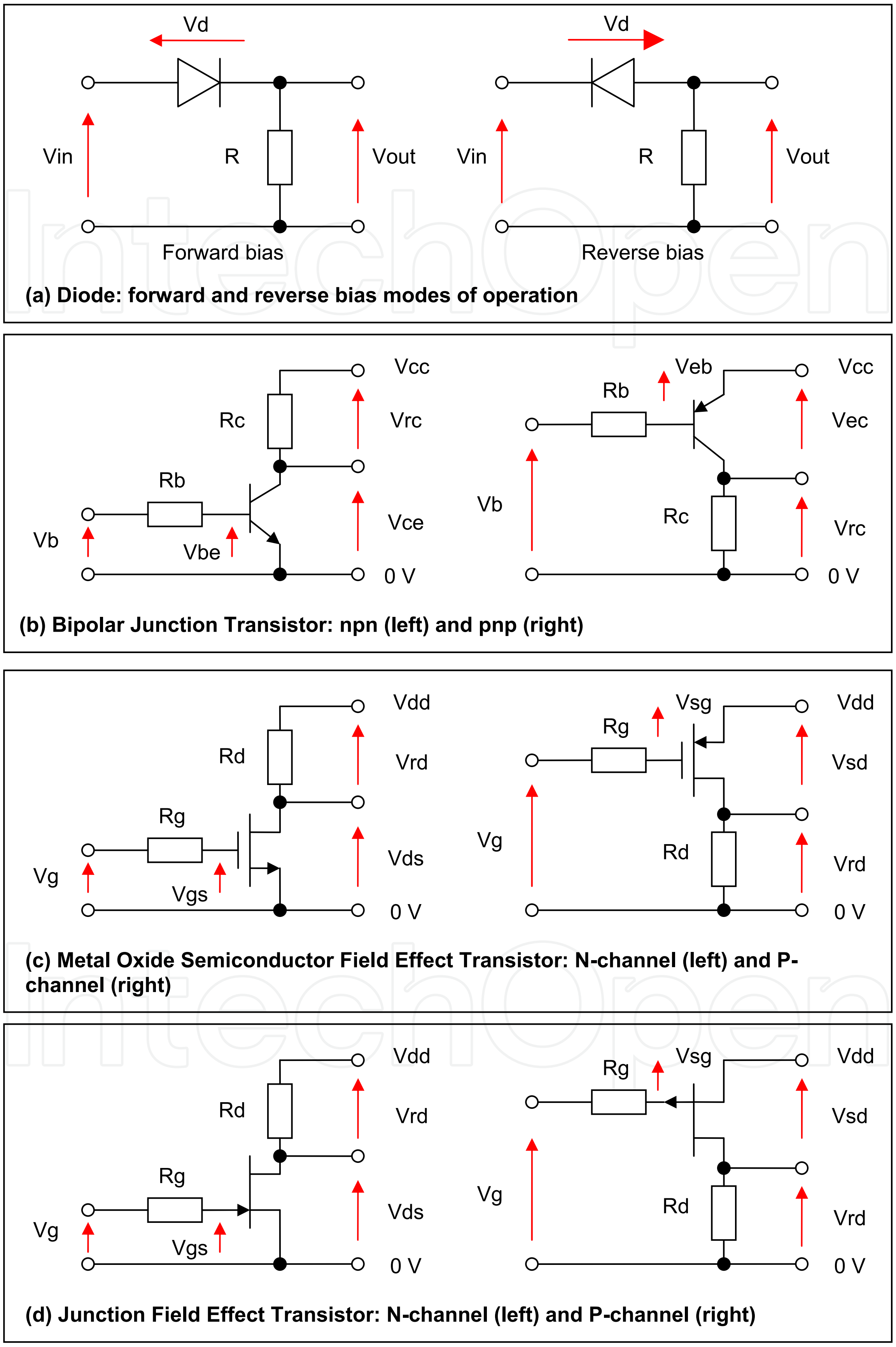

Vd
Vd
Vin
R
Vout
Vin
R
Vout
Forward bias
Reverse bias
(a) Diode: forward and reverse bias modes of operation
Vcc
Rc
Vrc
Rb
Vb
Vbe
Vce
0 V
Veb
Rb
Vcc
Vec
Vb
Rc
Vrc
0 V
(b) Bipolar Junction Transistor: npn (left) and pnp (right)
Vdd
Rd
Vrd
Rg
Vg
Vgs
Vds
0 V
Vsg
Rg
Vdd
Vsd
Vg
Rd
Vrd
0 V
(c) Metal Oxide Semiconductor Field Effect Transistor: N-channel (left) and P-channel (right)
Vdd
Rd
Vrd
Rg
Vg
Vgs
Vds
0 V
Vsg
Rg
Vdd
Vsd
Vg
Rd
Vrd
0 V
(d) Junction Field Effect Transistor: N-channel (left) and P-channel (right)

Figure 6. Basic test circuits for diodes and transistors

```
%%-----------------------------------------------------
%% Function to create the waveforms and subplots
%% for a Schottky diode model.
%%-----------------------------------------------------

function Schottky_animation()

%%-----------------------------------------------------
%% Create the variables and equations
%%----------------------------------------------------

clear Degrees Time R Vin Vr Vd Id;

Vd = 0.3;
R = 2;

for (j=0:1:9)
    for (k=1:1:360)
        Vin(k + (j* 360)) = 1.0 * sind(k - 1);
        Time(k + (j* 360)) = (k + (j * 360)) / 360;
    end
end

for (i=1:1:length(Vin))
    if (Vin(i) > 0.3)
        Vr(i) = (Vin(i) - Vd);
    else
        Vr(i) = 0;
    end
end

Id = (Vr / R);

%%-----------------------------------------------------
%% Create the subplots and wait for 5 seconds
%%----------------------------------------------------

scrsz = get(0,'ScreenSize');
figure('Name', 'Ideal Schottky Diode Half-Wave Rectifier Circuit', ...
   'OuterPosition',[scrsz(3)/8 scrsz(4)/16 (3 * (scrsz(3)/4)) (15 * (scrsz(4)/16))])

ha(1) = subplot(4,1,1);
plot(Time, Vin,'YDataSource', 'Vin');
hold on;
grid;
plot(Time, Vr, 'r', 'YDataSource', 'Vr');
title ('\it{Time Vs Vin & Vr}', 'Color','k', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Vin & Vr (V)');

ha(2) = subplot(4,1,2);
plot(Time, Vin,'YDataSource','Vin');
grid;
```

```
title ('\it{Time Vs Vin}', 'Color','b', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Vin (V)');

ha(3) = subplot(4,1,3);
plot(Time, Vr, 'r', 'YDataSource', 'Vr');
grid;
title ('\it{Time Vs Vr}', 'Color','r', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Vr (V)');

ha(4) = subplot(4,1,4);
plot(Time, Id, 'm', 'YDataSource', 'Id');
grid;
title ('\it{Time Vs Id}', 'Color','m', 'FontWeight', 'bold');
xlabel('Time (seconds)');
ylabel('Id (A)');

linkaxes(ha, 'xy');
axis([0 3 -1.1 1.1]);
pause(5);

%%-------------------------------------------------
%% Create the scrolling plot
%%-------------------------------------------------

for k = 0:0.1:7

    refreshdata(ha(1),'caller');
    refreshdata(ha(2),'caller');
    refreshdata(ha(3),'caller');
    refreshdata(ha(4),'caller');
    axis([k (k + 3) -1.1 1.1]);
    drawnow;
    pause(0.5);

end

end

%%-------------------------------------------------
%% End of code
%%-------------------------------------------------
```

Listing 1. Schottky barrier diode model – MATLAB m-file code to show signal rectification

In figure 7 then:

1. Time is plotted on the horizontal axis.
2. The top plot shows the input sine wave voltage (*Vin*) and the half-wave rectified voltage across the resistor (*Vr*) are plotted on the vertical axis.
3. The second plot shows the input voltage (*Vin*).
4. The third plot shows the resistor voltage (*Vr*).
5. The bottom plot shows the diode current (*Id*).

When the plot scrolls, after an initial delay of five seconds, it shows a similar waveform to that which would be seen on an oscilloscope display; it would be seen here that as time increments, the viewed waveforms scrolls across the screen. This shows how theoretical models can be created and analysed in MATLAB. Simulation however is only part of the overall story. Relating theory to the "real world" requires suitable the *design*, *build* and *test* of real circuits. Then, MATLAB could be used to analyse the results from a physical circuit prototype, and the theoretical model and practical circuits could be compared.

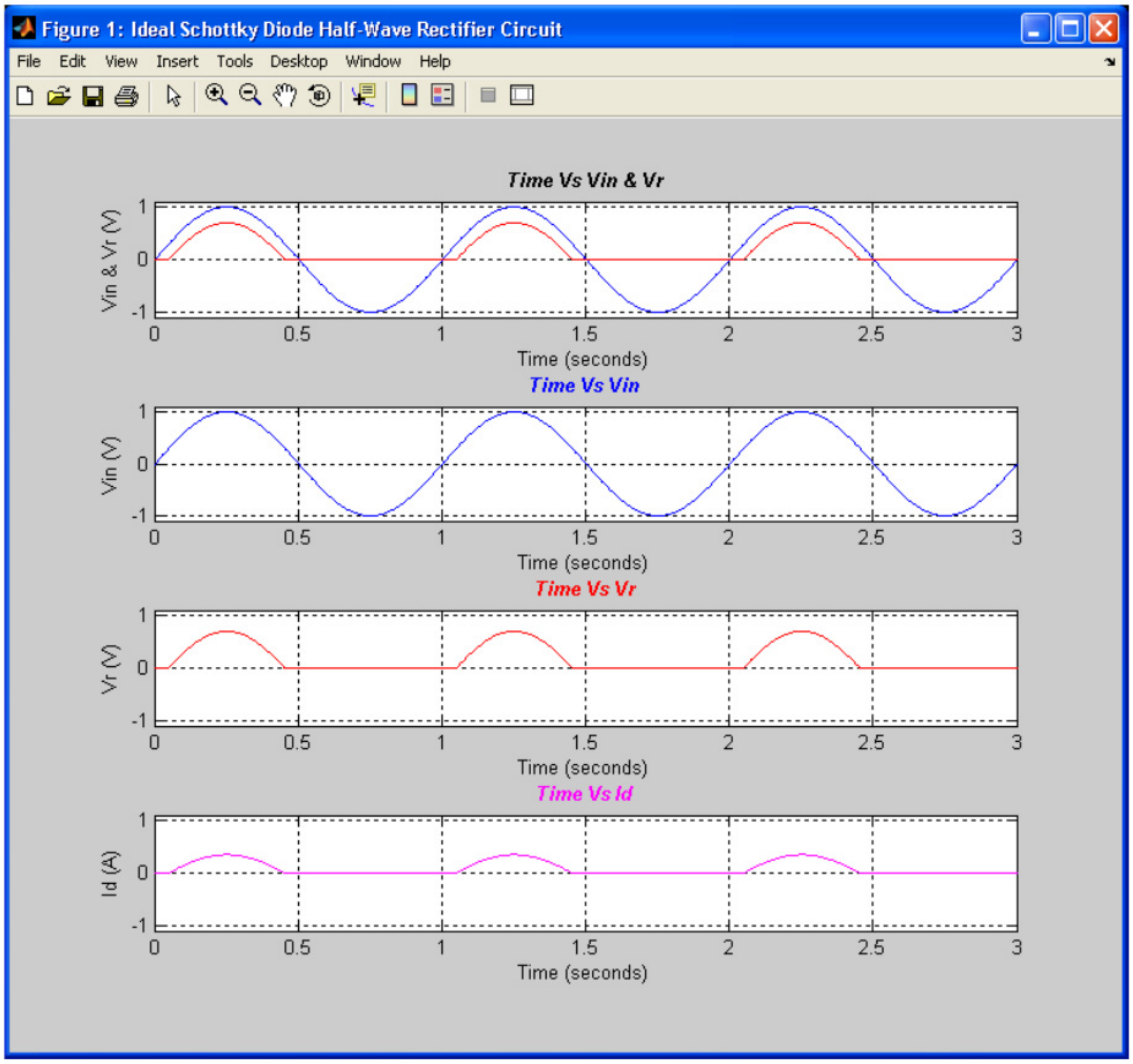


Figure 7. Schottky barrier diode model – MATLAB plot from listing 1

In more general terms, the above discussion flow would be extended to any semiconductor device, given the ability to create and measure the required range of voltage levels. For example, figure 6(b) shows a test circuit for a bipolar junction transistor (both npn and pnp types), figure 6(c) shows a test circuit for a metal oxide semiconductor field effect transistor (both N-channel an P-channel types) and finally, figure 6(d) shows a test circuit for a junction field effect transistor (both N-channel an P-channel types). These test circuits are suitable for the experimentation arrangements considered here. However, it is possible to create different test set-ups given the availability of suitable test and measurement equipment. Care however has to be taken in order to ensure that:

1. Suitable device types are used.
2. The applied voltages operate the devices in the intended modes of operation.
3. Damage to the devices would not occur in the physical test set-up if specified and used correctly (maximum device ratings are never encountered or exceeded).

Device damage would not occur in a simulated mathematical model, but could occur in a real device. For example, in N-channel JFET circuits, the gate-source voltage is to be zero or negative for correct device operation with the drain-source voltage zero or positive. The gate resistor (Rg) is included here to ensure that if a positive gate-source voltage were to be applied then the current through the JFET gate node would be limited to a safe value by suitable choice of the resistor value. When gate-source voltage is to be zero or negative, no current flows through the transistor gate (the p-n junction created is reverse biased) and for d.c. gate-source voltages, the resistor has no effect (although for a.c. signals the value of the resistor would affect the circuit operation).

4. Case study 1: The traditional learning scenario

This section will describe the use of the experiment via a traditional laboratory scenario. In this arrangement, the student builds a test circuit (either using a suitable solderless prototyping board or physically soldering the components to a suitable printed circuit board (PCB)) and runs a number of electrical tests on the electronic circuit. The tests are chosen to operate the particular device under the modes of operation that are of interest for the student to investigate. Once the tests have been completed, the student would plot the results (by-hand) on graph paper and then import the results into MATLAB for analysis and computer based graphing of the results. Here, it would then be possible to consider the physical process of setting-up an experiment, running the experiment and taking results before utilising a suitable mathematics tool for analysis purposes. However, the traditional *manual* way of plotting the graph from experiment data is slow and sometimes not convenient. Both the idealised (theoretical mathematical equation model) and the operation of an actual semiconductor device could then be analysed and compared, with differences between the practical device test results and idealised models analysed using MATLAB. Using the analysis tool to analyse the behaviour of and to plot the experiment data means that various analyses can be performed on the data and the results quickly plotted.

Within the teaching and learning of semiconductor device fundamentals, the basic devices to initially introduce to the student are the diode and transistor. To illustrate this, figure 8 shows the device to discuss here, the Schottky barrier diode. The circuit here shows the BAT86 Schottky barrier diode in a forward bias mode of operation. In this mode of operation, when the input voltage applied is positive, the diode will allow the flow of current through the load resistor and the diode will have a voltage drop of approximately 0.3 V when conducting (the actual device voltage drop being dependent on the level of current flowing through the diode). If the diode is connected in the reverse direction, the reverse bias mode of operation will be encountered and the diode will block the flow of current until a reverse bias junction breakdown voltage is encountered at which point the diode will conduct current. In reverse bias junction breakdown, if the current flow is not limited then damage to the diode will occur.

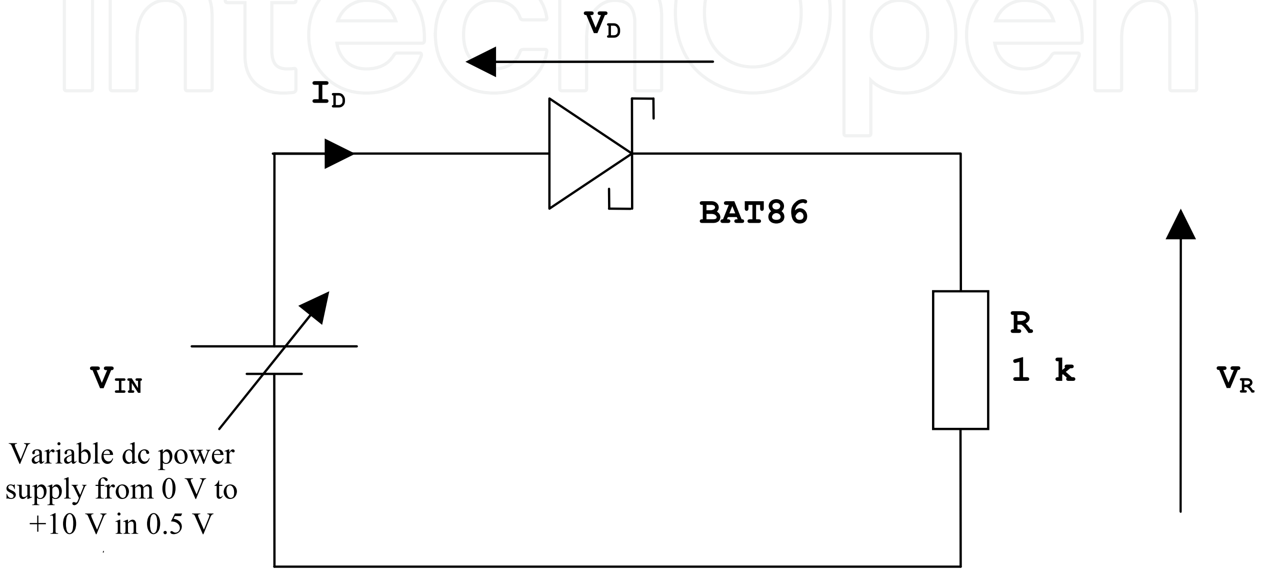


Figure 8. BAT86 Schottky diode experiment (forward bias)

The I-V mathematical model characteristic of the diode in figure 9 shows both the expected forward and reverse bias modes of operation and the ideal device equation are also noted:

Forward bias:

$$I_D = I_S\left[e^{(qV_D/nkT)} - 1\right]$$

Reverse bias (prior to breakdown):

$$I_D = -I_S$$

Here:

- I_D is the diode current.
- I_S is the diode saturation current.
- q is the charge on an electron.
- V_D is the forward bias diode voltage drop.
- n is the ideality factor and is set to 1.
- k is Botzmann's constant.
- T is the temperature in degrees Kelvin.

The current-voltage (I-V) relationship that should be encountered during an experiment is that as shown in figure 9. The regions of operation of interest are the forward bias (to the right of the I_D-axis) and the reverse bias (to the left of the I_D-axis) prior to reverse bias junction breakdown.

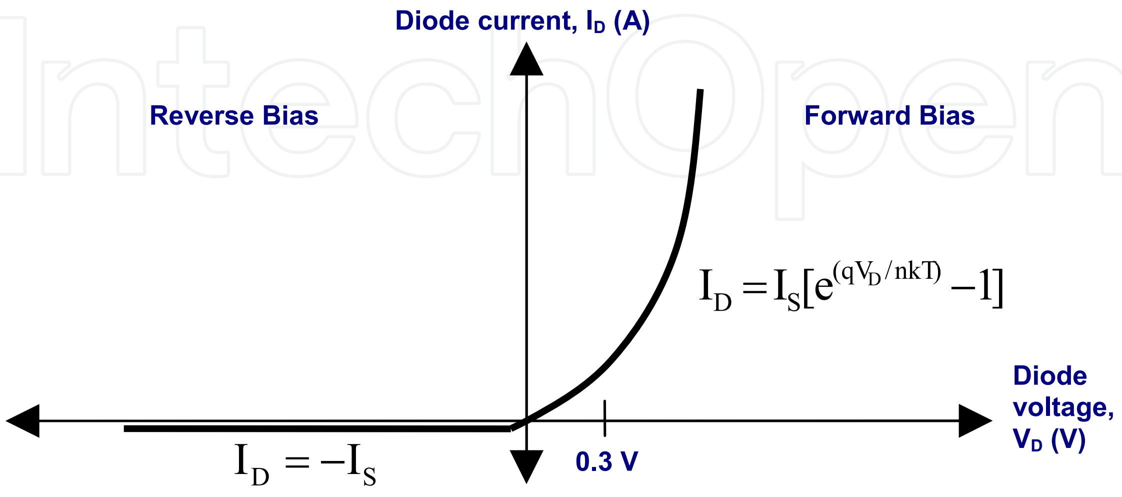


Figure 9. Schottky diode I-V characteristic (before reverse bias junction breakdown is encountered)

In forward bias, the diode current increases in an exponential manner with a linear increase in diode voltage. The diode voltage is around 0.3 V when current flows through the device, the exact value of diode voltage dependent on the value of the diode current. In reverse bias and prior to reverse bias breakdown occurring, the diode current is essentially independent of the diode voltage and is approximately the value of the saturation current (I_S). This effect can readily be modelled in MATLAB as shown in listing 2, here using the *for loop* in the calculation of the diode current for set values of diode voltage.

For comparison purposes, from the BAT86 Schottky barrier diode datasheet, the diode parameters can be identified. These are summarised in table 1.

Parameter	Conditions	Maximum value
Forward bias		
Forward bias voltage drop	Forward current = 0.1 mA	300 mV
	Forward current = 1 mA	380 mV
	Forward current = 10 mA	450 mV
	Forward current = 30 mA	600 mV
	Forward current = 100 mA	900 mV
Reverse bias		
Reverse bias current	Reverse bias voltage = 40 V	5 μA

Table 1. BAT86 Schottky barrier diode [2] datasheet forward and reverse bias parameters

In listing 2, both the forward bias mode of operation and the reverse bias mode of operation are modelled and plotted, where:

- The forward bias voltage is *Vd_forward*.
- The forward bias current is *Id_forward*.
- The reverse bias voltage is *Vd_reverse*.
- The reverse bias current is *Id_ reverse*.

One way in which the ideal device equations can be modelled in MATLAB is shown in listing 2:

```matlab
%%----------------------------------------------------------------------
%%-- Schottky barrier diode forward bias equation
%%-- x-axis scaling (voltage) from 0 V to +0.8 V
%%----------------------------------------------------------------------

       Vd_forward = (0:0.01:0.8)

       T = 300
       k = 1.38066e-23
       q = 1.60218e-19

       Is_sch = 1e-9
       n = 1.0

       for i=(1:1:length(Vd_forward))

           Id_forward = Is_sch * (exp((q * Vd_forward)/(n * k * T)) - 1)

       end

%%----------------------------------------------------------------------
%%-- Schottky barrier diode reverse bias equation
%%-- x-axis scaling (voltage) from 0 V to -10.0 V
%%----------------------------------------------------------------------

       Vd_reverse = (0:-0.01:-10.0)

       Is_sch = 1e-9

       for i = (1:1:length(Vd_reverse))

           Id_reverse = -Is_sch

       end

%%----------------------------------------------------------------------
%% End of code
%%----------------------------------------------------------------------
```

Listing 2. Calculating the forward and reverse bias operation of the Schottky diode

These values and equations can be entered into MATLAB and plots of the I-V characteristic can be produced (by adding code for plotting the results to the code shown in listing 2).

Figure 10 shows a figure where the forward and reverse bias diode characteristics are plotted. The top subplot shows the forward and reverse bias characteristic (V_D shown from -10 V to +0.6 V) and the bottom two subplots show the forward bias characteristic zooming in on different ranges of V_D. Hence, specific areas of device operation can easily be identified from the overall set of data and individual plots created for understanding and analysis purposes.

The experiment can then be prototyped on a solderless prototyping board as shown in figure 11. Connecting this circuit to a dual d.c. power supply and digital voltmeter will provide all the necessary circuitry and equipment to undertake the experiment shown in figure 8. Table 2 shows the forward bias results (measuring *Vin* and *Vr*, and calculating *Vd* and *Id*). Table 3 shows the reverse bias results. Both sets of diode currents are calculated using the calculated resistor current and the actual measured resistor value ($Rm = 997\ \Omega$)

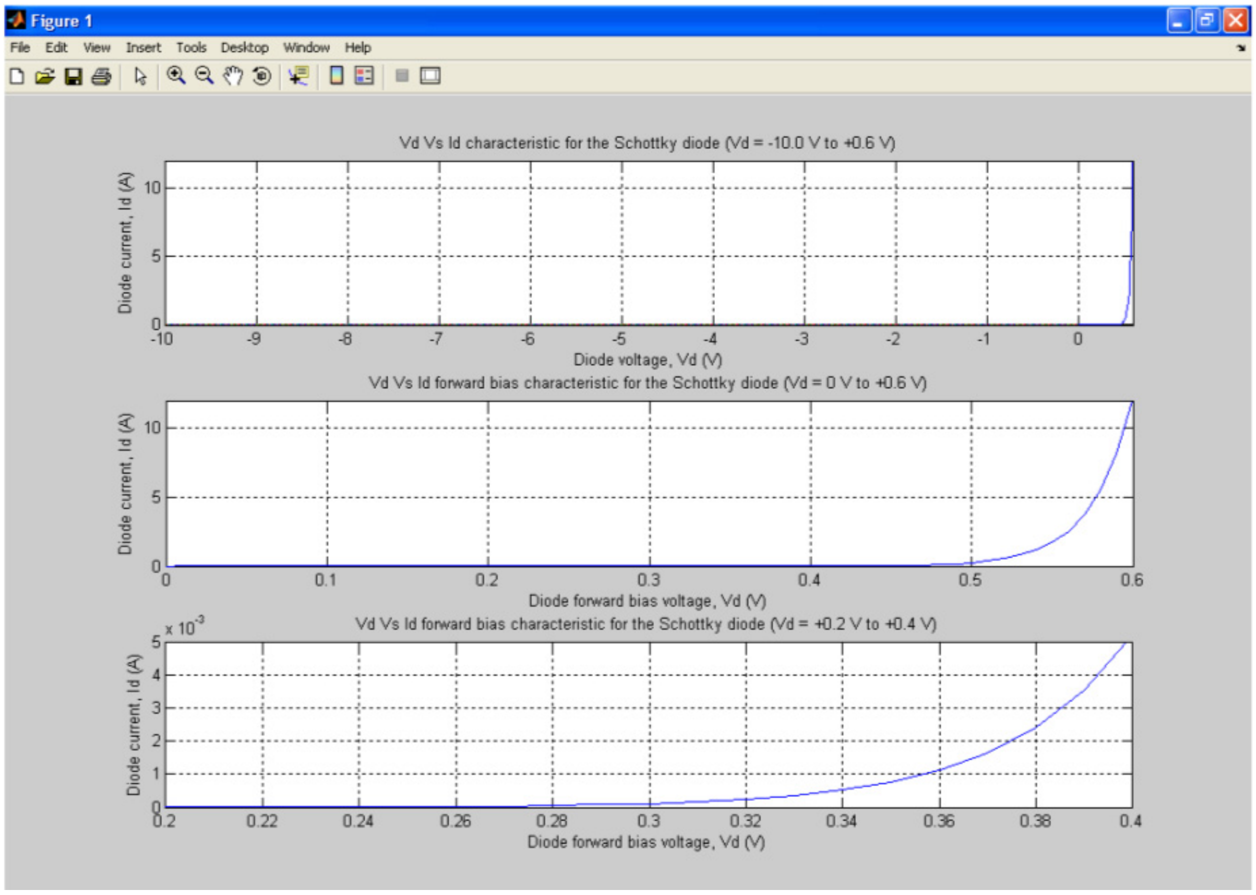


Figure 10. Ideal Schottky barrier diode equation plots

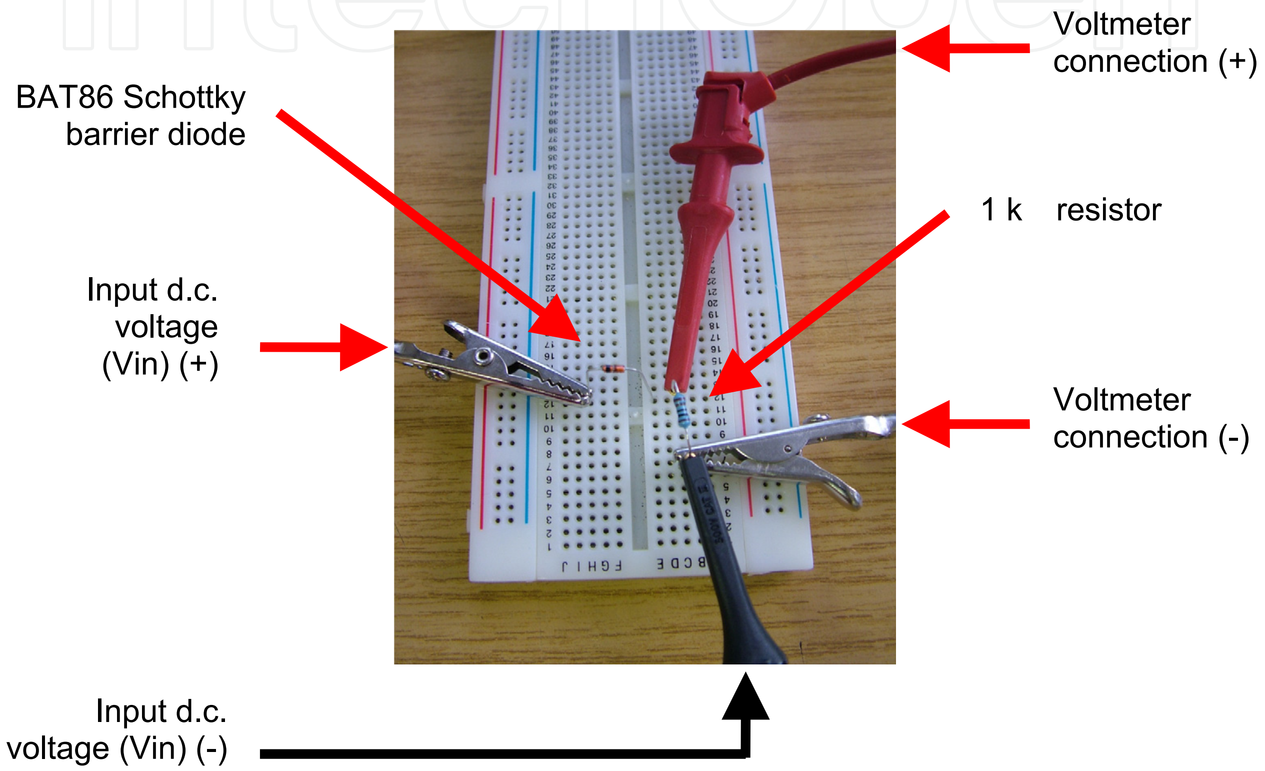

BAT86 Schottky barrier diode
Voltmeter connection (+)
1 k resistor
Input d.c. voltage (Vin) (+)
Voltmeter connection (-)
Input d.c. voltage (Vin) (-)

Figure 11. Prototyping the experiment

The measured values of voltage (*Vin* and *Vr*) can be entered into MATLAB and the diode voltage and current (*Vd* and *Id*) can then be calculated. The results can be entered into MATLAB and plotted with the MATLAB m-file code as shown in listing 3. Note also that the actual resistance value (*Rm*) of the 1 kΩ resistor was used in the current calculation in order to account for the tolerance of the resistor used. Figure 12 shows the resulting MATLAB plot with the forward bias shown on the top sub-plot and the both forward and reverse bias shown on the bottom sub-plot.

Vin (V)	Vr (V)	Vd = (Vin – Vr) (V)	Id = Ir = (Vr / Rm) (A)
Measured	**Measured** (to nearest 1 mV)	**Calculated** (to nearest 1 mV)	**Calculated** (to nearest 1 μA)
0	0	0	0
0.5	0.268	0.232	269 μA
1.0	0.739	0.261	741 μA
1.5	1.223	0.277	1.227 mA
2.0	1.712	0.288	1.717 mA
2.5	2.204	0.296	2.211 mA
3.0	2.697	0.303	2.705 mA
3.5	3.190	0.310	3.200 mA
4.0	3.685	0.315	3.696 mA
4.5	4.180	0.320	4.193 mA
5.0	4.675	0.325	4.689 mA
5.5	5.170	0.330	5.186 mA
6.0	5.666	0.334	5.683 mA
6.5	6.162	0.338	6.181 mA
7.0	6.659	0.341	6.679 mA
7.5	7.155	0.345	7.177 mA
8.0	7.651	0.349	7.674 mA
8.5	8.148	0.352	8.173 mA
9.0	8.645	0.355	8.671 mA
9.5	9.141	0.349	9.169 mA
10.0	9.637	0.363	9.666 mA

Table 2. Diode forward bias test results

Vin (V)	Vr (V)	Vd = -(Vin – Vr) (V)	Id = -Ir = -(Vr / Rm) (A)
Measured	**Measured** (to nearest 1 mV)	**Calculated** (to nearest 1 mV)	**Calculated** (to nearest 1 μA)
0	0	-0	0
0.5	0	-0.5	0
1.0	0	-1.0	0
1.5	0	-1.5	0
2.0	0	-2.0	0
2.5	0	-2.5	0
3.0	0	-3.0	0
3.5	0	-3.5	0
4.0	0	-4.0	0
4.5	0	-4.5	0
5.0	0	-5.0	0
5.5	0	-5.5	0
6.0	0	-6.0	0
6.5	0	-6.5	0
7.0	0	-7.0	0
7.5	0	-7.5	0
8.0	0	-8.0	0
8.5	0	-8.5	0
9.0	0	-9.0	0
9.5	0	-9.5	0
10.0	0	-10.0	0

Table 3. Diode reverse bias test results

```
%%----------------------------------------------------------------------
%%-- BAT86 Schottky barrier diode test
%%----------------------------------------------------------------------

%%-------------------------------------
%% Test conditions
%%-------------------------------------

Rm = 997
```

```

Vin = [0 0.5 1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6 6.5 7 7.5 8 8.5 9 9.5 10]

%%-------------------------------------
%% Diode in forward bias
%%-------------------------------------

Vr_forward = [0 0.268 0.739 1.223 1.712 2.204 2.697 3.190 3.685 4.180 ...
    4.675 5.170 5.666 6.162 6.659 7.155 7.651 8.148 8.645 9.141 9.637]

Vd_forward = (Vin - Vr_forward)
Id_forward = (Vr_forward / Rm)

%%-------------------------------------
%% Diode in reverse bias
%%-------------------------------------

Vr_reverse = [0 0 0 0 0 0 0 0 0 0 0 0 ...
     0 0 0 0 0 0 0 0 0]

Vd_reverse = -(Vin - Vr_reverse)
Id_reverse = -(Vr_reverse / Rm)

%%-------------------------------------
%% Plot results
%%-------------------------------------

subplot(2, 1, 1)
plot(Vd_forward, Id_forward, 'k')
hold on
grid
plot(Vd_forward, Id_forward, 'o')

title('Vd Vs Id for measured BAT86 Schottky Barrier Diode (forward bias)')
xlabel('Vd (V)')
ylabel('Id (A)')

subplot(2, 1, 2)
plot(Vd_forward, Id_forward, 'k')
hold on
grid
plot(Vd_forward, Id_forward, 'o')

plot(Vd_reverse, Id_reverse, 'r')
plot(Vd_reverse, Id_reverse, 'o')

title('Vd Vs Id for measured BAT86 Schottky Barrier Diode')
xlabel('Vd (V)')
ylabel('Id (A)')

%%-----------------------------------------------------------------------
%%-- End of BAT86 Schottky barrier diode test
%%-----------------------------------------------------------------------
```

Listing 3. M-file code for entering and plotting the test results for the BAT86 Schottky barrier diode

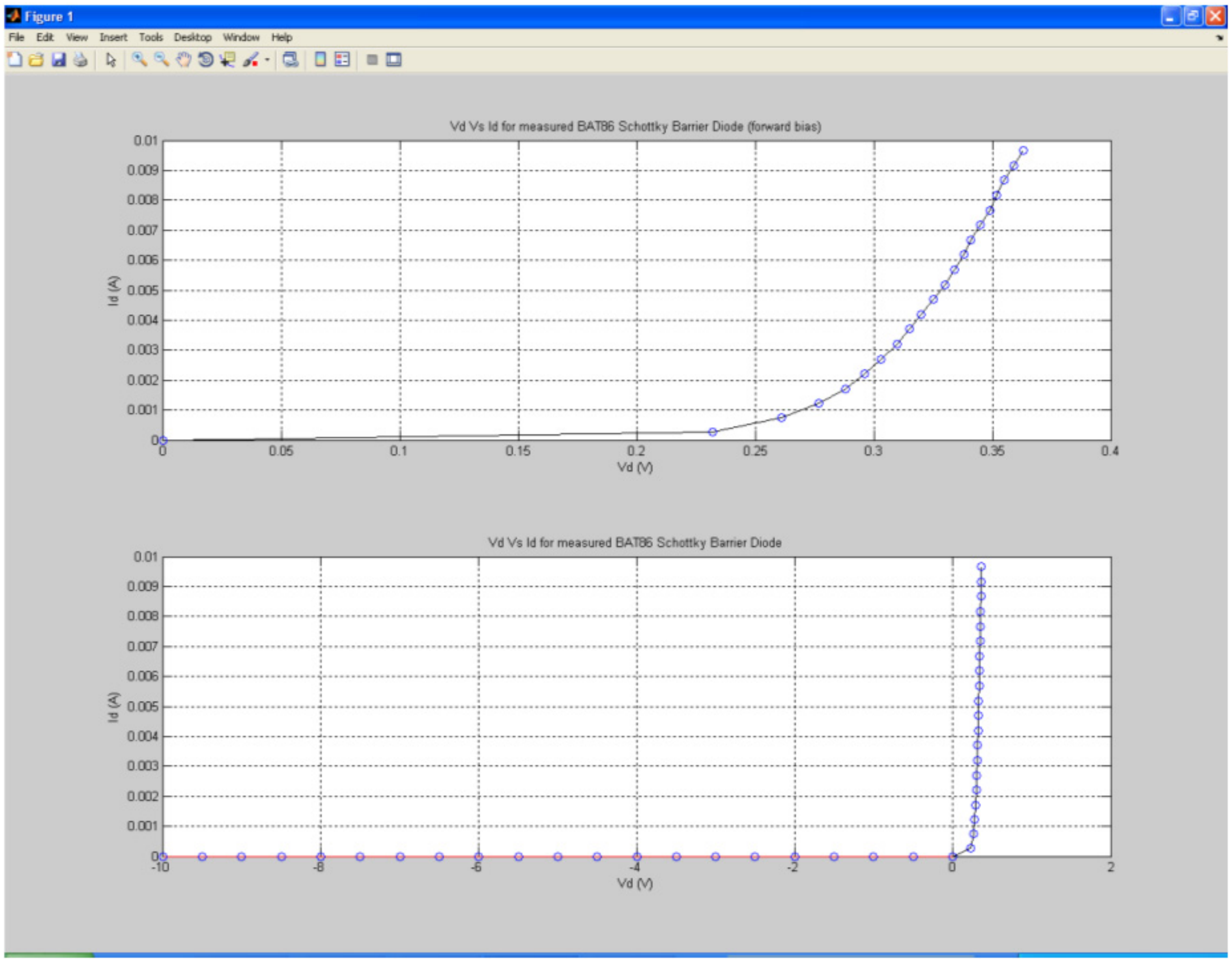


Figure 12. BAT86 Schottky barrier diode test results (forward and reverse bias)

In order to create the plots then there are three main parts to the m-file:

1. Entering the device test data (measured values) as scalar values and arrays:
   a. The measured resistor value (*Rm*) (line 9).
   b. The input voltage values (*Vin*) (line 11).
   c. The measured resistor voltage with the diode in forward bias (*Vr_forward*) (lines 17 and 18).
   d. The measured resistor voltage with the diode in reverse bias (*Vr_reverse*) (lines 27 and 28).
2. Creating the equations to determine the diode voltage and current values:
   a. The diode voltage in forward bias (*Vd_forward*) (line 20).
   b. The diode current in forward bias (*Id_forward*) (line 21).
   c. The diode voltage in reverse bias (*Vd_reverse*) (line 30).
   d. The diode current in reverse bias (*Id_reverse*) (line 31).
3. Plotting the currents and voltages on a single figure using subplots (lines 37 to 58).

5. Case study 2: Computer aided learning scenario

This section will describe the use of the experiment via a computer interface. In this arrangement, the student does not build the circuit – the circuit experiment is pre-built and connected to a computer via a suitable computer serial port. In this discussion, the RS-232 serial port is used and this interfaces to the diode experiment via an interface circuit consisting of a suitably configured Spartan-3 field programmable gate array (FPGA) [6, 7] and digital-to-analogue converter (DAC) and analogue-to-digital converter (ADC) arrangement.

This structure of the hardware and software interface is shown in figure 13.

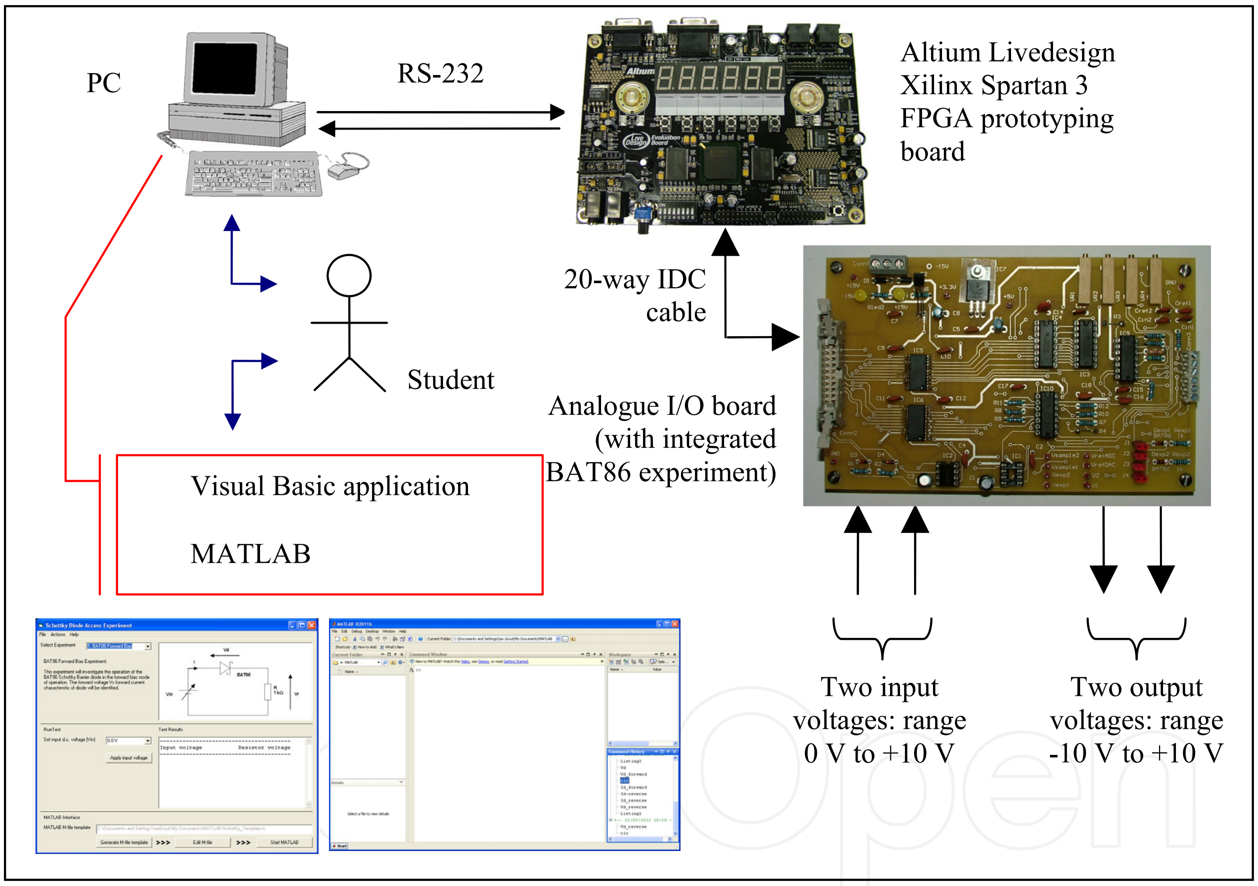

PC
RS-232
Altium Livedesign Xilinx Spartan 3 FPGA prototyping board
20-way IDC cable
Student
Analogue I/O board (with integrated BAT86 experiment)
Visual Basic application
MATLAB
Two input voltages: range 0 V to +10 V
Two output voltages: range -10 V to +10 V

Figure 13. Tester hardware interface set-up

Here, the PC connects to an FPGA prototyping board via an RS-232 interface. The FPGA prototyping board also houses a voltage level shifter circuit (the MAX3232 IC) to interface the RS-232 voltage levels to the FPGA +3.3 V power supply voltage levels. Digital inputs and outputs (I/O) of the FPGA then connect to an analogue I/O board. This houses a power supply, two DACs (providing two independent voltage outputs in the range -10 V to +10 V) and two ADCs (receiving two input voltages in the range 0 V to +10 V). The analogue I/O board consists of a custom made printed circuit board (PCB) with an on-board BAT86 Schottky barrier diode experiment and a connector to interface to additional experiments.

Figure 14 shows the manufactured PCB in more detail. It is of course possible to utilise an existing hardware arrangement rather than designing a custom made solution.

In addition, the computer runs a suitably designed software application which gives the user access to the tester hardware. It is possible to use any suitable programming language (including the MATLAB GUI (Graphical User Interface) builder) to provide for a user interface and suitable software applications to access to the computer RS-232 port. However, here, a Visual Basic [8] application was used with Visual Basic here being the language of preference. Figure 15 shows the GUI as designed. The user selects the experiment (diode in forward bias or reverse bias mode of operation) and then runs a test by setting the input voltage (*Vin*) to apply to the circuit and then sending this to the experiment. The results from the experiment then are displayed on the GUI.

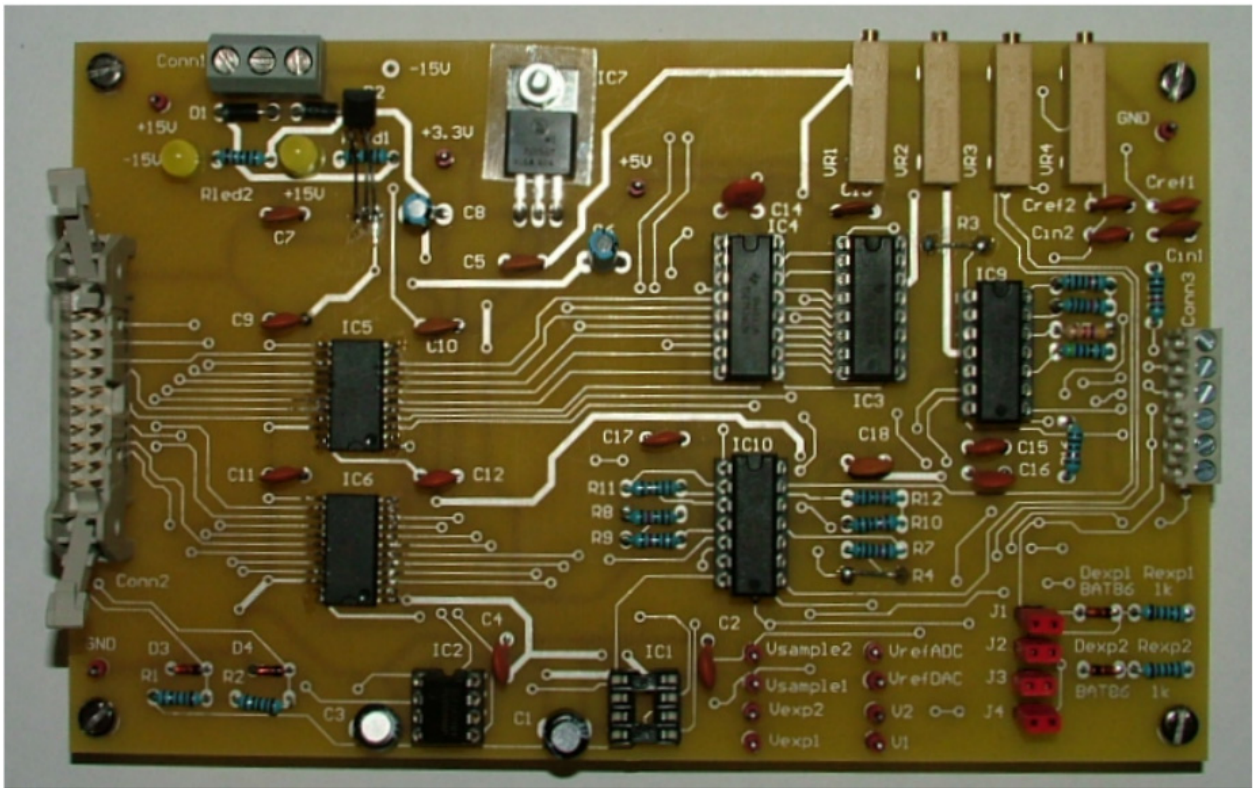
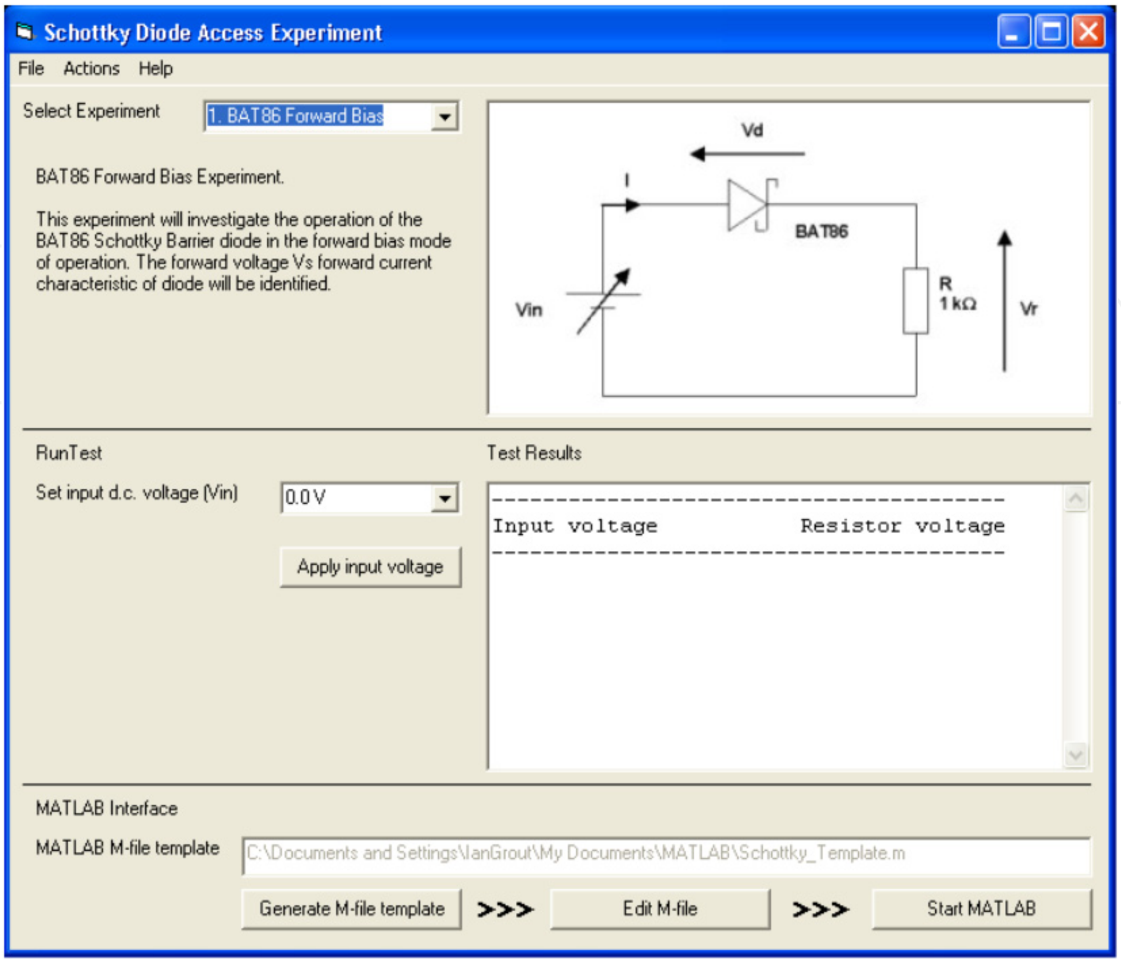


Figure 14. Analogue I/O board PCB

On completion of the experiment, the experiment results are saved into an automatically generated MATLAB m-file template (essentially the input and output voltages are saved in arrays in the m-file). The user then edits the m-file (adding his or her own results analysis and plotting commands), and runs MATLAB to analyse the test results. Listing 4 shows an example m-file template automatically generated by the software application.


Schottky Diode Access Experiment

File Actions Help

Select Experiment: 1. BAT86 Forward Bias

BAT86 Forward Bias Experiment.

This experiment will investigate the operation of the BAT86 Schottky Barrier diode in the forward bias mode of operation. The forward voltage Vs forward current characteristic of diode will be identified.

RunTest

Set input d.c. voltage (Vin): 0.0 V

Test Results

```
------------------------------------------
Input voltage             Resistor voltage
------------------------------------------
```

MATLAB Interface

MATLAB M-file template: C:\Documents and Settings\IanGrout\My Documents\MATLAB\Schottky_Template.m

Figure 15. User interface GUI

```
%-------------------------------------------------------------------%
% MATLAB template file for BAT86 Schottky barrier diode experiment.
% BAT86 Forward Bias mode of operation.
%-------------------------------------------------------------------%
% Experiment undertaken on 13/01/2012 14:25:43
%-------------------------------------------------------------------%

%%--------------------
%% Test Results
%%--------------------

R = 1e+3

Vin = [0, 0, 0]
Vr  = [0.00, 0.00, 0.00]

%%--------------------
%% Analysis section
%%--------------------



%-------------------------------------------------------------------%
```

Listing 4. Automatically generated m-file template

The equipment set-up is shown in figure 16. Here, a laptop PC runs the software application and interfaces to the FPGA prototyping board via an RS-232 serial data link. The analogue I/O board operates on a +/-15 V d.c. power supply and test points are provided on the board to allow for the measurement of specific circuit voltages. The BAT86 Schottky barrier diode experiment is also incorporated on the analogue I/O (experiment) board.

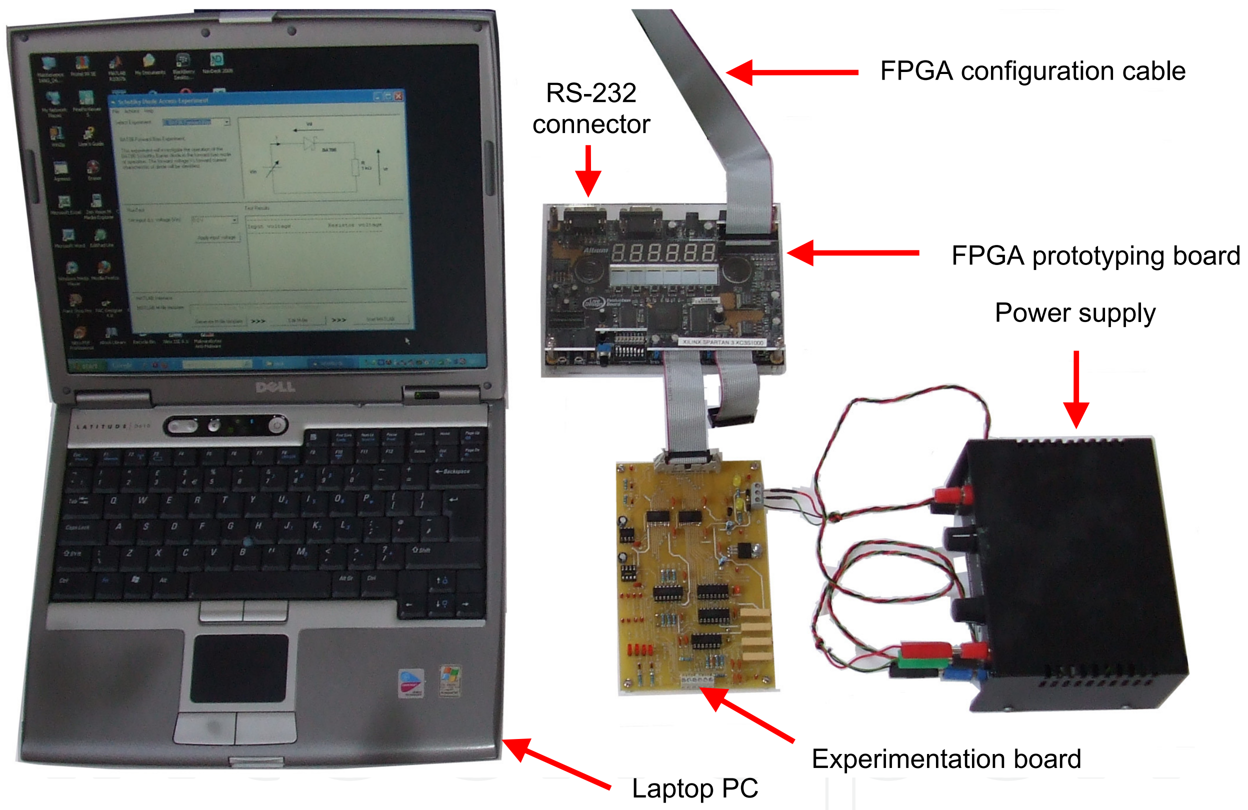


Figure 16. Computer set-up

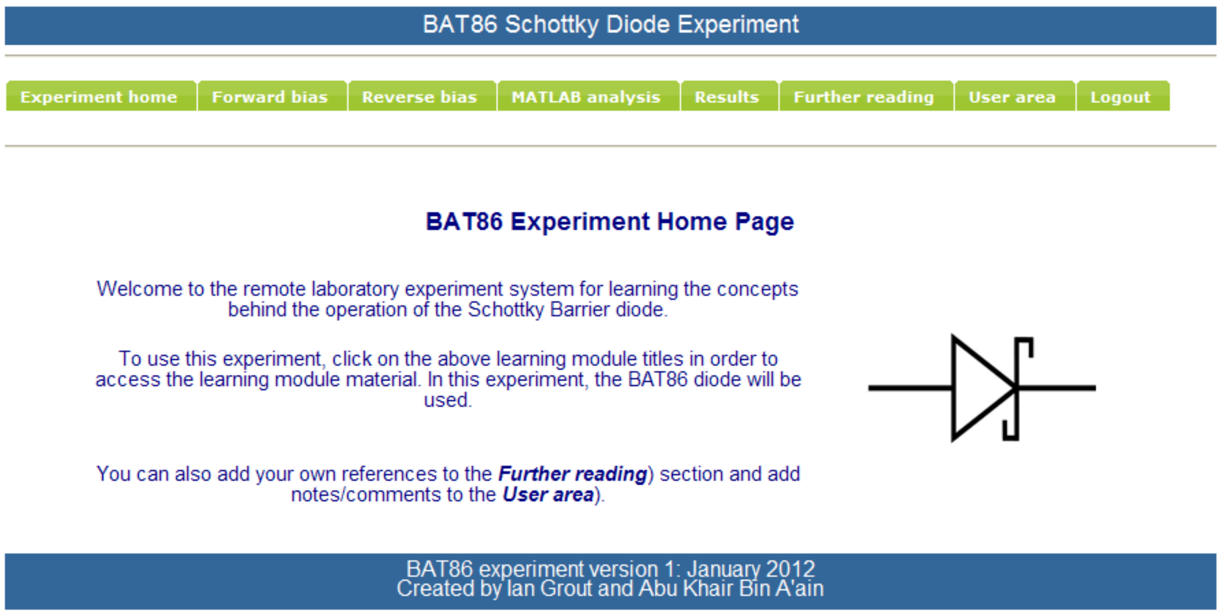
6. Case study 3: Distance education learning scenario

Increasingly, universities are providing access to courses in a distance mode of operation – that is students are not physically located within the institution, but learn from an alternative (remote) location. In the simplest terms, students are registered to study for a qualification within the institution but access course material (lecture notes and assignments) via a suitable Internet connection and learning management system (LMS) such as Moodle [9] or a custom LMS solution. However, such a simplistic statement does not tell the whole story.

In engineering and science, there is a need to undertake experiments and analyse experiment results. This is traditionally undertaken *at-presence* where the learner undertakes the experiment within the laboratory facilities hosted by the institution. This approach might not be possible for distance learners and so alternative approaches to providing access to experimentation have been developed – distance learning utilising remote experimentation accessed through a remote laboratory [10]. The role that distance learning now undertakes within the teaching and learning environments on a global scale has gained widespread acceptance over the last number of years. Textural based teaching material, enhanced with graphics and animation, is now supported through the use of remote experimentation. Remote experimentation is essentially physical laboratory experiments that are set-up to be accessible via the Internet. Many institutions and organisations now provide for their laboratory experimentation to be Internet-enabled, so providing access via a web browser for remote users who may be in a location in the world that provides the user with an Internet access capability. This has been shown to be of high value from university education through to E-Science applications [11].

Industry can also benefit from the concept of distance access and learning scenario. Imagine for example, a parent company which has invested heavily in expensive equipment, for example integrated circuit (IC) testers. When the company extends its operations into other locations, it does not then need not to invest in the same equipment again. This is where the remote logging into the tester for device testing and data collection purposes can be done, and the analysis can carried out using MATLAB. In another scenario, the after sales service of a product can be made more friendly, easier and cost effective. In traditional approach, when a machine is down (i.e., not operational due to a fault), the customer has to wait for the service engineer to arrive from another part of the world to repair the machine. Using the concept of distance access, the service engineer needs only to be remotely connected to the down machine and MATLAB can once again be useful to analysis the symptoms and suggest possible causes, along with solutions. Of course the symptom has to be first modelled accurately just like the diode and other devices are modelled.

Here and in this distance based learning scenario, the basic experimentation set-up and discussed in section 5 (Case study 2: computer aided learning scenario) can be modified to be accessible via the Internet. Essentially, a web-server arrangement (a WAMP (Windows, Apache [12], MySQL [13] and PHP [14])) system is set-up here and the experiment user interface is via a series of Internet browser pages (web pages). Figure 17 shows the home page set-up for the experiment.



BAT86 Experiment Home Page

Welcome to the remote laboratory experiment system for learning the concepts behind the operation of the Schottky Barrier diode.

To use this experiment, click on the above learning module titles in order to access the learning module material. In this experiment, the BAT86 diode will be used.

You can also add your own references to the ***Further reading***) section and add notes/comments to the ***User area***).

BAT86 experiment version 1: January 2012
Created by Ian Grout and Abu Khair Bin A'ain

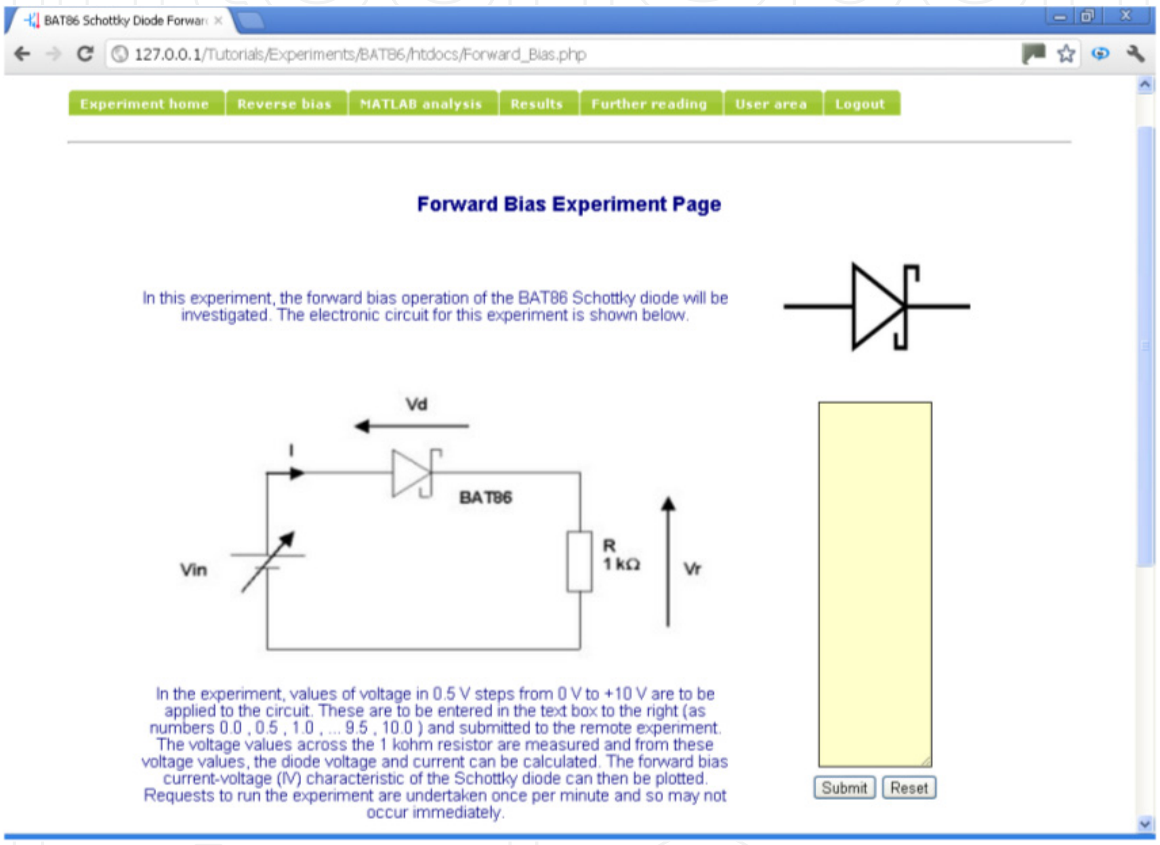
Figure 17. User interface (home page)

In the user interface, the user is considered to already have successfully logged in to the remote laboratory via a username/password arrangement and is then given access to a diode experiment menu system (top of the page) for accessing different parts of the experiment. These options are identified in table 4. The user is prompted to access the different web pages in order to access different aspects of the experiment, including access to MATLAB via the web server arrangement.

Option	Option description
Experiment home	Return to this (home) page.
Forward bias	Run the diode forward bias experiment.
Reverse bias	Run the diode reverse bias experiment.
MATLAB analysis	Create a MATLAB function to enter and analyse the experiment results.
Results	View the results from previously run experiments and MATLAB analyses.
Further reading	List of references (which can be added to by users).
User area	Area to allow users to add notes for all users to access.
Logout	Log out from the overall remote laboratory arrangement.

Table 4. User interface options

In order to run an experiment, the user chooses the diode in forward bias or reverse bias mode of operation via the top menu system. Figure 18 shows the forward bias experiment web page. This provides for an introduction to the experiment and an area to enter the values of the input voltage (*Vin*) to apply to the test circuit. This is via an HTML form seen to the right of the page. All values are entered into the form and then submitted to the web server (and hence the experiment) using the *Submit* button.



Forward Bias Experiment Page

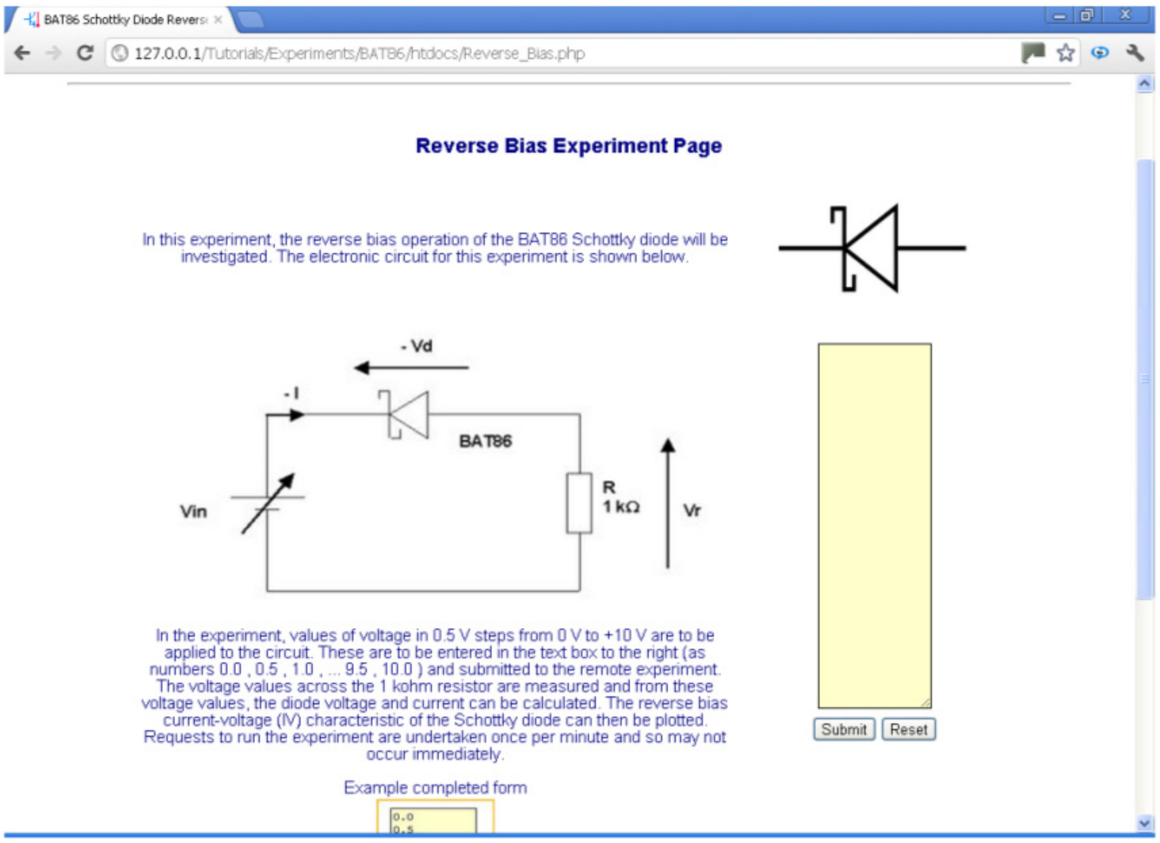
In this experiment, the forward bias operation of the BAT86 Schottky diode will be investigated. The electronic circuit for this experiment is shown below.

In the experiment, values of voltage in 0.5 V steps from 0 V to +10 V are to be applied to the circuit. These are to be entered in the text box to the right (as numbers 0.0 , 0.5 , 1.0 , ... 9.5 , 10.0) and submitted to the remote experiment. The voltage values across the 1 kohm resistor are measured and from these voltage values, the diode voltage and current can be calculated. The forward bias current-voltage (IV) characteristic of the Schottky diode can then be plotted. Requests to run the experiment are undertaken once per minute and so may not occur immediately.

Figure 18. Remote submission of diode forward bias experiment input voltage values

Figure 19 shows the reverse bias experiment web page. This page has the same form as the forward bias experiment, except now on submission of the input voltage values, the reverse bias experiment is selected rather than the forward bias experiment.

As the experiments are performed remotely with the experiment electronic hardware connected to the web server PC, the user has a much more restricted and controlled access to the experiment. In this arrangement, the experiment is run on the remote web server and the results are accessible via a web page. Here, the user does not have interactive control of the experiment, rather they submit their values and the web server treats this as a job to complete, allowing the same experiment to be used by multiple users without the need for the user to pre-book the experiment.



BAT86 Schottky Diode Revers

127.0.0.1/Tutorials/Experiments/BAT86/htdocs/Reverse_Bias.php

Reverse Bias Experiment Page

In this experiment, the reverse bias operation of the BAT86 Schottky diode will be investigated. The electronic circuit for this experiment is shown below.

In the experiment, values of voltage in 0.5 V steps from 0 V to +10 V are to be applied to the circuit. These are to be entered in the text box to the right (as numbers 0.0 , 0.5 , 1.0 , ... 9.5 , 10.0) and submitted to the remote experiment. The voltage values across the 1 kohm resistor are measured and from these voltage values, the diode voltage and current can be calculated. The reverse bias current-voltage (IV) characteristic of the Schottky diode can then be plotted. Requests to run the experiment are undertaken once per minute and so may not occur immediately.

Example completed form

```
0.0
0.5
```

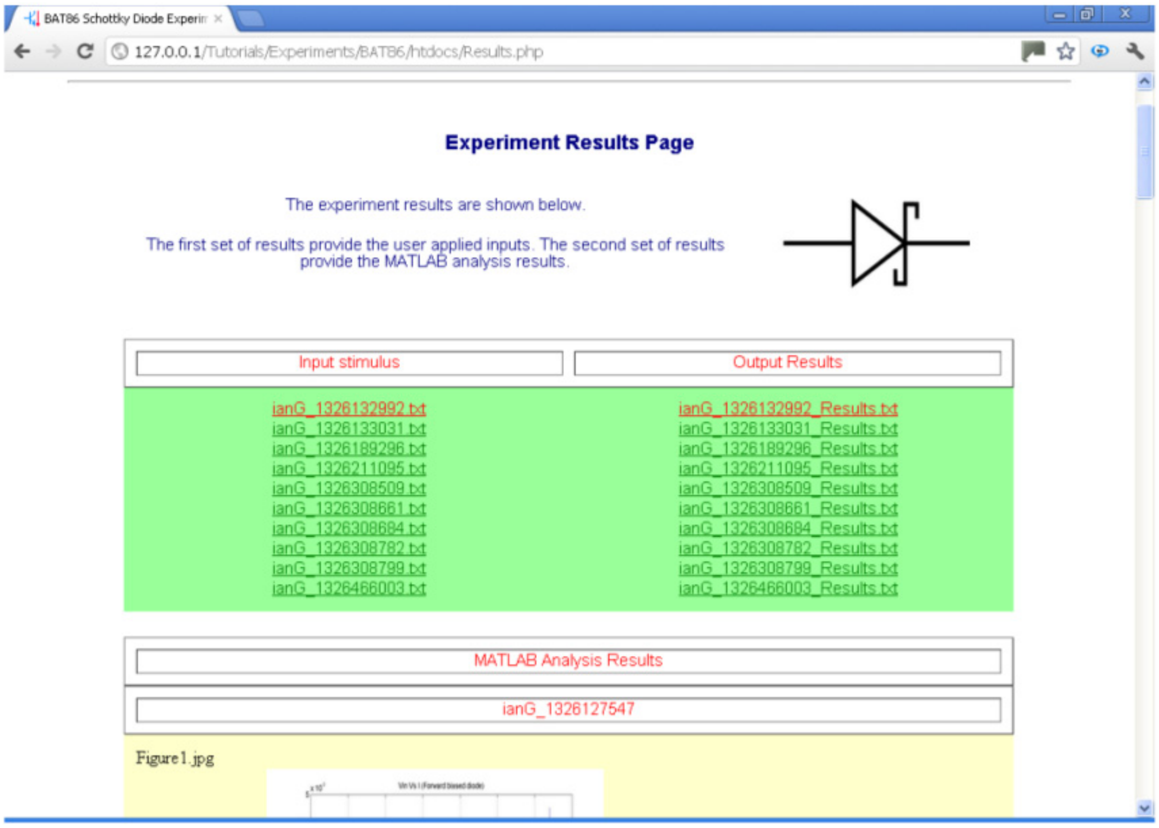
Figure 19. Remote submission of diode reverse bias experiment input voltage values

Figure 20 shows the experiment results page that the user sees. Their submitted test values and the experiment results are available as hypertext links via this web page. Hence, the user can see the text based experiment results which would then be used in MATLAB to analyse the results.

The MATLAB analysis is undertaken by entering the code for a MATLAB *function* into a form on the web page and then submitting the *function* to the web server for remote processing. On completion of the processing, the results are available for the user to access.

In figure 21, a template form for the MATLAB code as a function is automatically presented to the student and they complete the function with their own comments. This is shown in more detail in figure 22 and listing 5. The user does not modify this code structure, but inserts their own code between the "% Start of user commands" and "% End of user commands" comments.

On submission of the completed form, the code is automatically saved in a function m-file, MATLAB automatically executes the m-file commands and the results are saved to suitable results files.



BAT86 Schottky Diode Experin

127.0.0.1/Tutorials/Experiments/BAT86/htdocs/Results.php

Experiment Results Page

The experiment results are shown below.

The first set of results provide the user applied inputs. The second set of results provide the MATLAB analysis results.

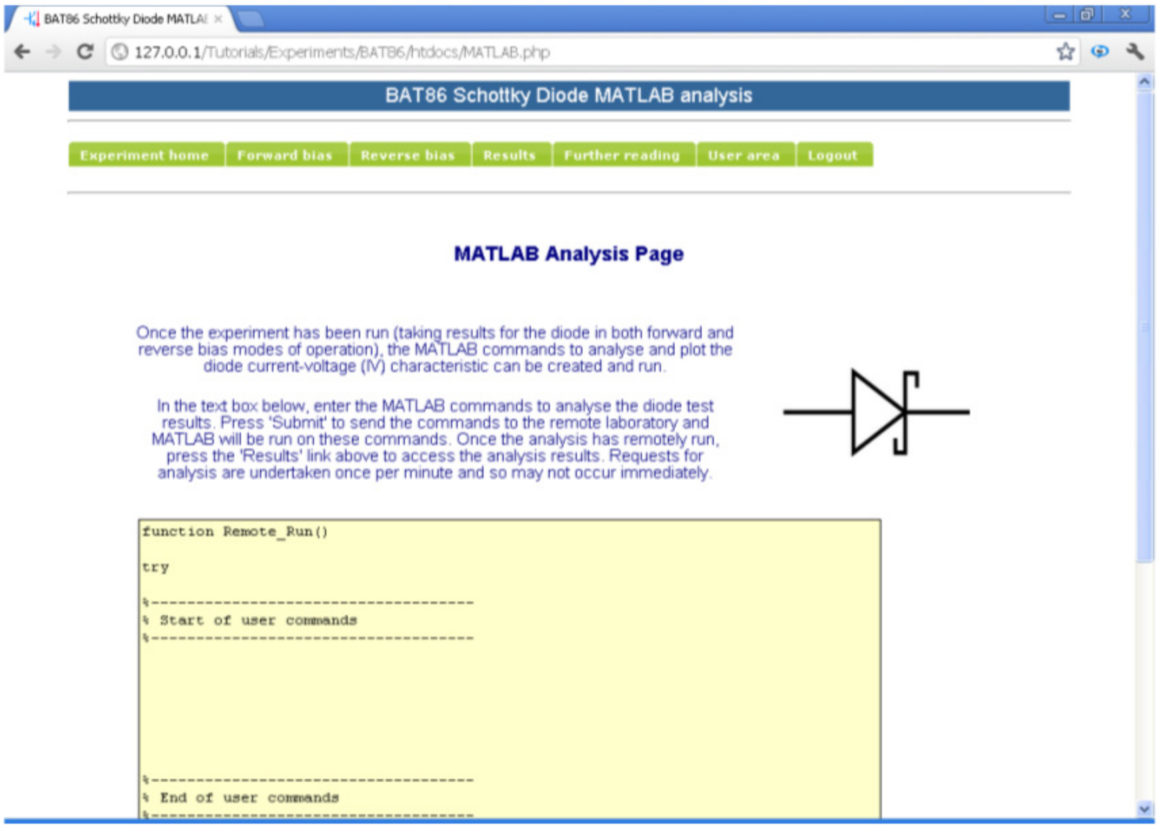
Input stimulus	Output Results
ianG_1326132992.txt	ianG_1326132992_Results.txt
ianG_1326133031.txt	ianG_1326133031_Results.txt
ianG_1326189296.txt	ianG_1326189296_Results.txt
ianG_1326211095.txt	ianG_1326211095_Results.txt
ianG_1326308509.txt	ianG_1326308509_Results.txt
ianG_1326308661.txt	ianG_1326308661_Results.txt
ianG_1326308684.txt	ianG_1326308684_Results.txt
ianG_1326308782.txt	ianG_1326308782_Results.txt
ianG_1326308799.txt	ianG_1326308799_Results.txt
ianG_1326466003.txt	ianG_1326466003_Results.txt

MATLAB Analysis Results

ianG_1326127547

Figure1.jpg

Figure 20. Viewing experiment results



BAT86 Schottky Diode MATLA

127.0.0.1/Tutorials/Experiments/BAT86/htdocs/MATLAB.php

BAT86 Schottky Diode MATLAB analysis

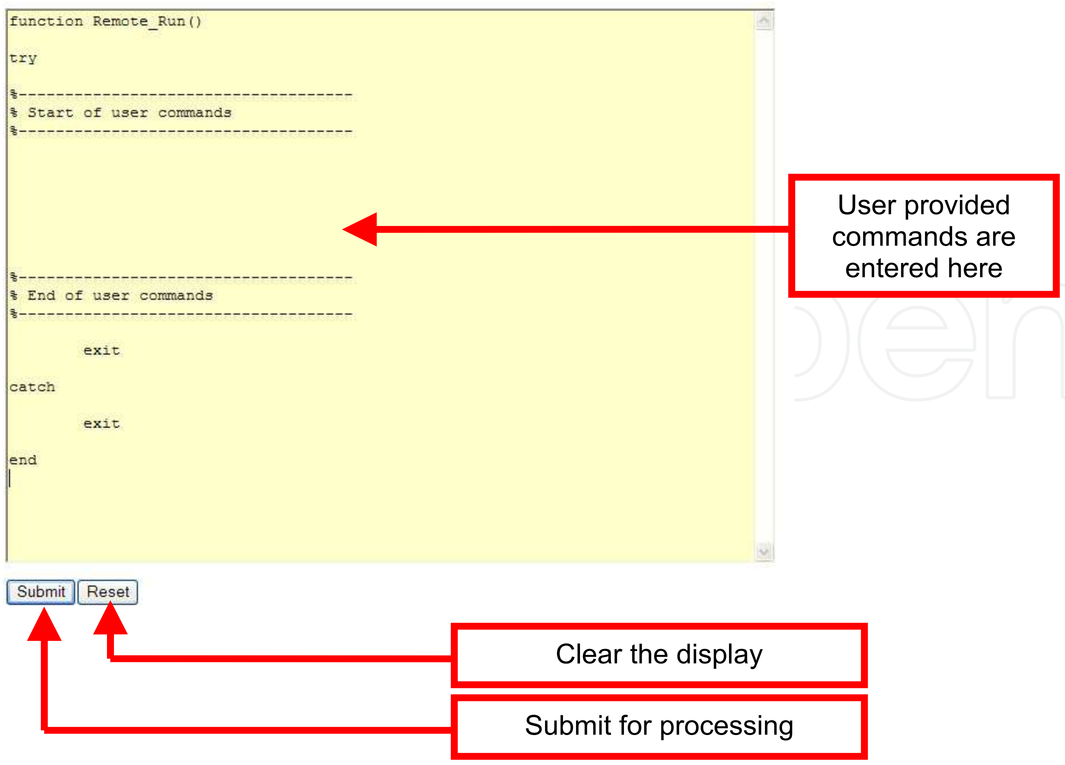
MATLAB Analysis Page

Once the experiment has been run (taking results for the diode in both forward and reverse bias modes of operation), the MATLAB commands to analyse and plot the diode current-voltage (IV) characteristic can be created and run.

In the text box below, enter the MATLAB commands to analyse the diode test results. Press 'Submit' to send the commands to the remote laboratory and MATLAB will be run on these commands. Once the analysis has remotely run, press the 'Results' link above to access the analysis results. Requests for analysis are undertaken once per minute and so may not occur immediately.

```
function Remote_Run()

try

%-------------------------------------
% Start of user commands
%-------------------------------------



%-------------------------------------
% End of user commands
%-------------------------------------
```

Figure 21. Remote submission of MATLAB commands to web server for remote processing



```
function Remote_Run()

try

%------------------------------------
% Start of user commands
%------------------------------------




%------------------------------------
% End of user commands
%------------------------------------

        exit

catch

        exit

end
```

Figure 22. Remote submission of MATLAB commands to web server for remote processing

The function template code is shown in listing 5. On submission, this function is saved into the m-file and MATLAB is executed on the web server PC. In order to handle possible errors in the submitted code, then the *try – catch* error control is used.

```
function Remote_Run()

try

%------------------------------------
% Start of user commands
%------------------------------------







%------------------------------------
% End of user commands
%------------------------------------

        exit

catch

        exit

end
```

Listing 5. MATLAB function template

Two important aspects of using MATLAB in this manner are:

1. MATLAB is run remotely on a different computer and so any instant visual feedback that a user would see when running MATLAB on his or her own computer is not immediately available.
2. If any errors are encountered in the user provided code, MATLAB must exit "gracefully" and not simply crash. It must also be able to provide suitable error feedback to the user.

To this extent, the above function template captures and reports errors, and the text feedback that would normally be seen in the MATLAB Command Window is automatically outputted to a text log file (and this file is available via the Results Page).

A third aspect of using MATLAB in this manner is that any figures to plot the results would not be seen if simply the plot(x, y) command was used.

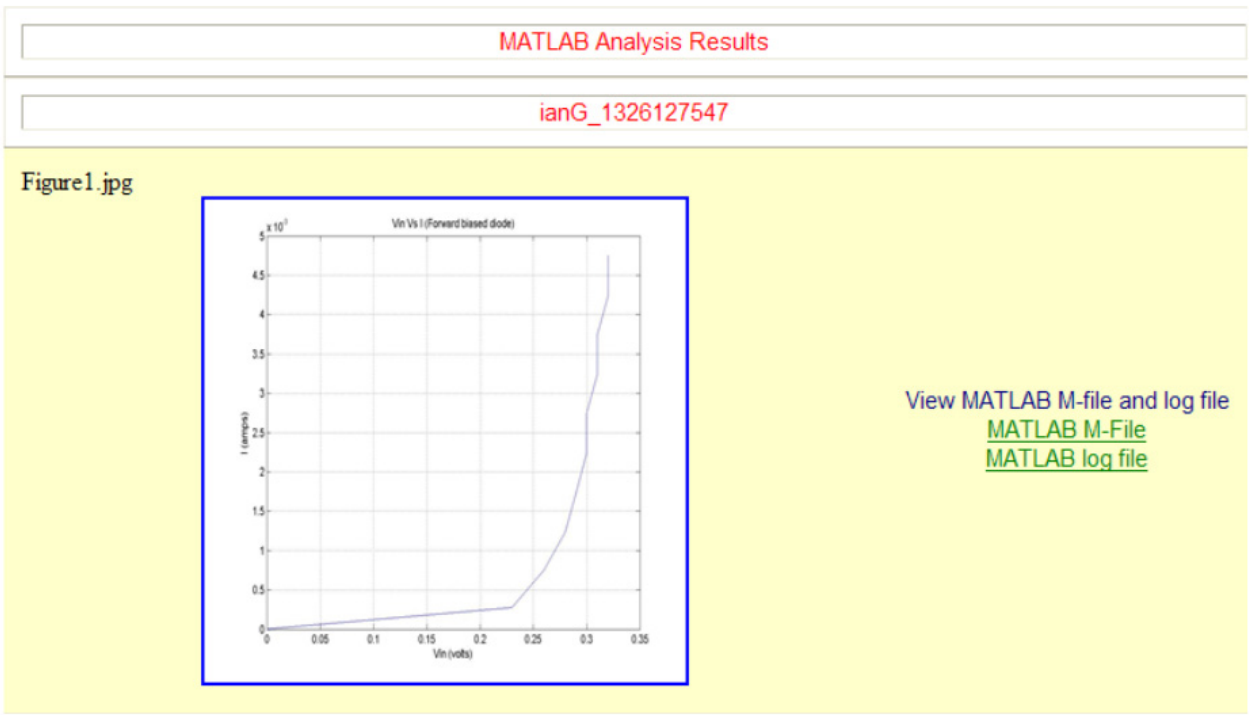
To this extent, whenever a figure plot is to be produced, this will need to be saved as an image file (using the commands of the form shown in listing 6) and if the image file is in JPEG format, it will be displayed in the Results page along with the MATLAB log file and the uploaded analysis code.

```
figH = figure('visible', 'off')                     Create figure


      plot(Vd_forward, Id_forward)                  Plot to figure
      grid                                          Add grid
      title('Vin Vs I (Forward biased diode)')      Add title
      xlabel('Vin (volts)')                         Add x-axis label
      ylabel('I (amps)')                            Add y-axis label


      print(figH, '-djpeg', 'Figure1.jpg')          Print figure to JPG file


close(figH)                                         Close figure
```

Listing 6. MATLAB: plotting the figure to a JPEG image file

Here in listing 6, a figure is created and plotted to (along with the annotation required by the user). Here, a figure called *figH* is created and plots the variables **Vd_forward** and **Id_forward** along with a figure title and axis labels. This is then saved to an image file in JPEG format with the file name Figure1.jpg.

Once an analysis has been undertaken, the results are available for viewing on the results page in the format as shown in figure 23. Here, the analysis run name (a unique run name), the image files produced, the generated MATLAB m-file and the MATLAB log file are available for viewing.



MATLAB Analysis Results

ianG_1326127547

Figure1.jpg

View MATLAB M-file and log file
MATLAB M-File
MATLAB log file

Figure 23. MATLAB analysis results access

The user can click on the computer mouse on the image and this shows the full-size figure in a new browser window. Where the user runs an m-file to create multiple figures and save these as image files, each generated image file is shown in the results window and is therefore accessible. The image file can then be saved to the student's own computer for later inclusion in experiment reports.

In this work, the infrastructure to support the use of MATLAB to be accessed via an Internet browser page is developed and discussed.

A final comment to make however relates to the use of the MATLAB software license where MATLAB runs on a particular computer, but is accessed remotely from a separate computer. It is not the intention to discuss specifics about the use of the software license and if there is any doubt about the use of the license in this mode then the provider of the license should be consulted.

7. Conclusions

This chapter has presented and discussed the use of MATLAB within an education environment with reference to the teaching and learning of semiconductor device fundamentals. Specifically, MATLAB can be integrated into the education curriculum as a tool to provide specific analysis and results presentation operations, these being (i) physical electronic circuit test results data entry, analysis and graphical plotting; (ii) idealised device characteristic equation modelling and graphical plotting; (iii) comparisons between idealised and actual device performance; and (iv) documentation preparation purposes. In this work, consideration was given to the use of MATLAB in three teaching and learning scenarios; (i) at-presence "traditional" laboratory experiments; (ii) at-presence computer aided learning laboratories; and (iii), distance based remote access to laboratory experiments. Each scenario was introduced and the development of the laboratory experiments discussed. The physical infrastructure (electronic hardware and software) was identified and the role in which technology is utilised in the education environment was presented. In particular, the way in which MATLAB was considered to be used and how it was integrated into custom developed education technology tools were highlighted. The discussions were based on the evaluation of the Schottky barrier diode, specifically the BAT86 Schottky barrier diode. However, the discussions, arguments and experimentation hardware and software can be readily adapted to other forms of semiconductor devices.

Author details

Ian Grout
Department of Electronic and Computer Engineering, University of Limerick, Limerick, Ireland

Abu Khari Bin A'ain
Faculty of Electrical Engineering, Universiti Teknologi Malaysia (UTM), Skudai, Johor, Malaysia

8. References

[1] The Mathworks Inc., MATLAB®, 2012, www.themathworks.com
[2] NXP, BAT86 Schottky barrier diode product datasheet, 2004, www.nxp.com/documents/data_sheet/BAT86.pdf
[3] Duane Hanselman and Bruce R. Littlefield, Mastering MATLAB 6, 6th Ed., Prentice Hall; 1 edition (14 Dec 2000), ISBN 0130194689
[4] Pierret R.F., Semiconductor Device Fundamentals, 1996, Addison-Wesley Publishing Company, ISBN 0-13-178459-5x
[5] Sze S.M., Semiconductor Devices Physics and Technology, 1985, John Wiley & Sons. Inc., ISBN 0-471-83704-0
[6] Xilinx Inc., Spartan-3 FPGA Family Data Sheet, 2009, www.xilinx.com/support/documentation/data_sheets/ds099.pdf
[7] Altium, TR0104 LiveDesign Evaluation Board Technical Reference Manual, 2004
[8] Microsoft Corporation, 2012, Visual Basic 6.0, msdn.microsoft.com/en-us/vstudio/ms788229
[9] Moodle, 2012, moodle.org/
[10] Ian Andrew Grout and Alexandre César Rodrigues da Silva, "Analysis Tool for Remote Laboratory Structures", Proceedings of the Remote Experimentation and Virtual Instrumentation conference (REV 2010), Stockholm, June 29 - July 2, 2010, www.rev-conference.org/REV2010/
[11] R. Ratering et al., "GridBeans: Supporting e-Science and Grid Applications", Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing", 4-6 Dec. 2006, Amsterdam, The Netherlands

[12] The Apache Software Foundation, Apache HTTP Server Project, 2012, httpd.apache.org/
[13] MySQL, 2012, dev.mysql.com/
[14] PHP, PHP Hypertext Preprocessor, 2012, www.php.net/