4. Data Processing System

The data from the microcontroller will be collected and sent to a MySQL database using LabVIEW. LabVIEW is software designed to create applications that interact with real-world data or signals. LabVIEW also allows for third-party add-ons which made sending data to MySQL a possibility. One such add-on used was LabSQL. LabSQL is a collection of VIs created by Jeffrey Travis that use the ADO object collection in LabVIEW in order to connect to databases, and perform SQL queries. ActiveX Data Objects or ADO is an application program interface or API that gives access to a database.

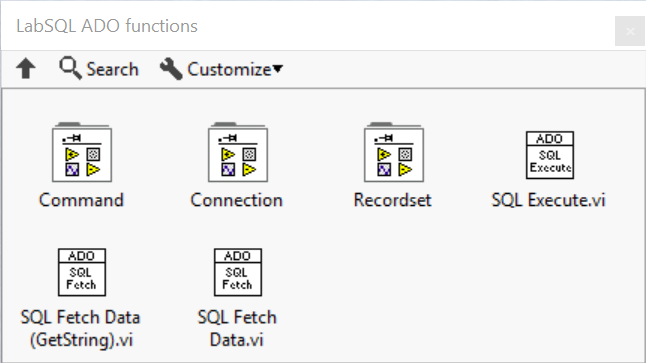


Figure 7: LabSQL ADO functions

LabSQL allows one to connect to a MySQL database and send data to a MySQL table. However, in order to send data to a MySQL database, one must first create a MySQL database. XAMPP Control Panel is a free web server solution developed by Apache Friends that allows for the easy creation of MySQL databases.

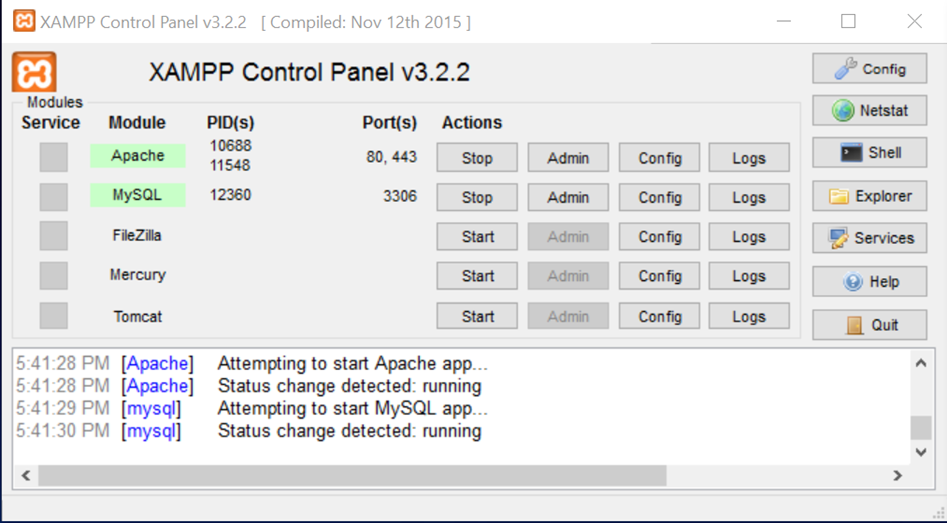


Figure 8: XAMPP Control Panel

Once XAMPP is active typing the address http://127.0.0.1/phpmyadmin brings you to the control hub.

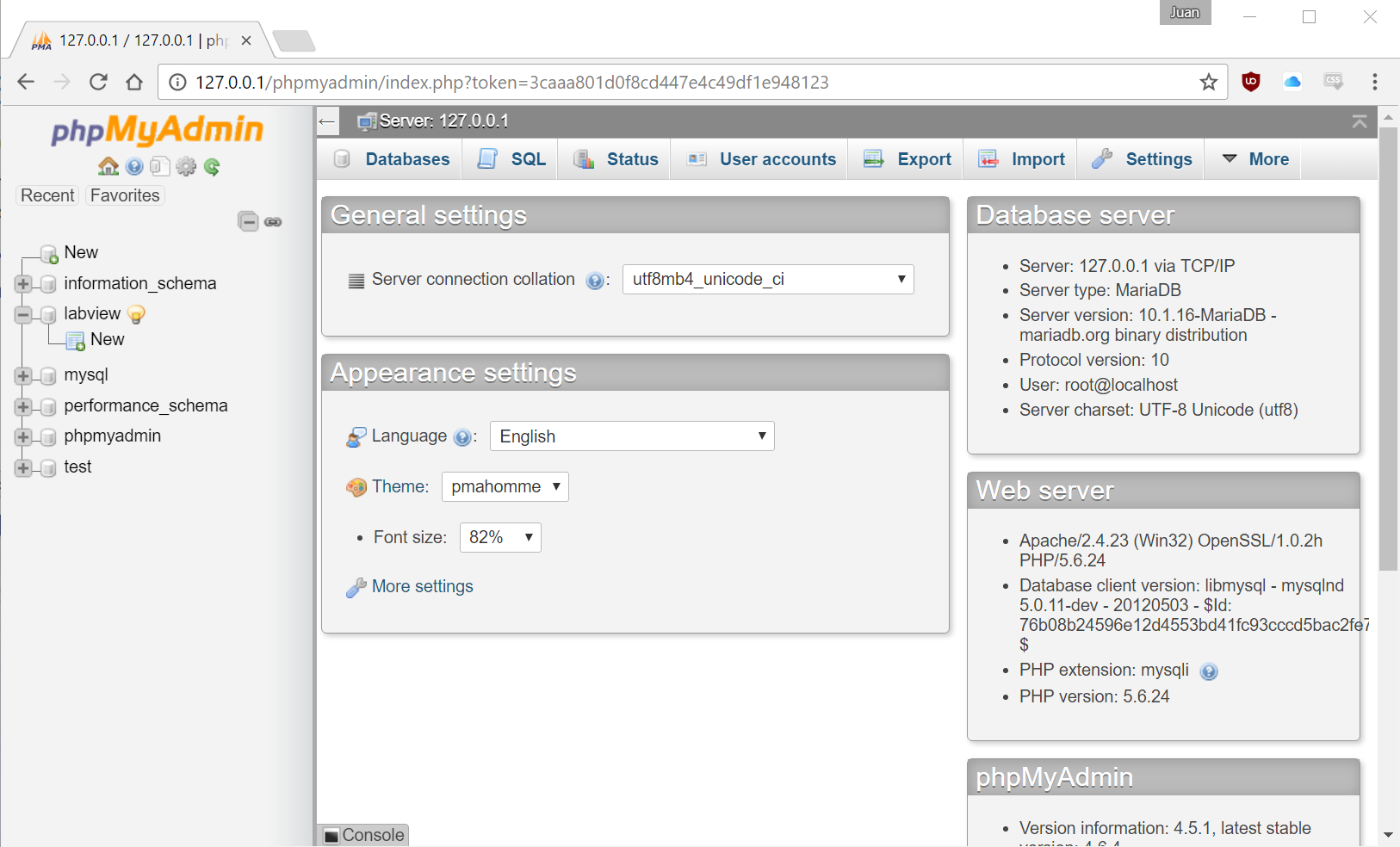


Figure 9: phpMyAdmin index page

Using XAMPP Control Panel one can create and edit MySQL tables.

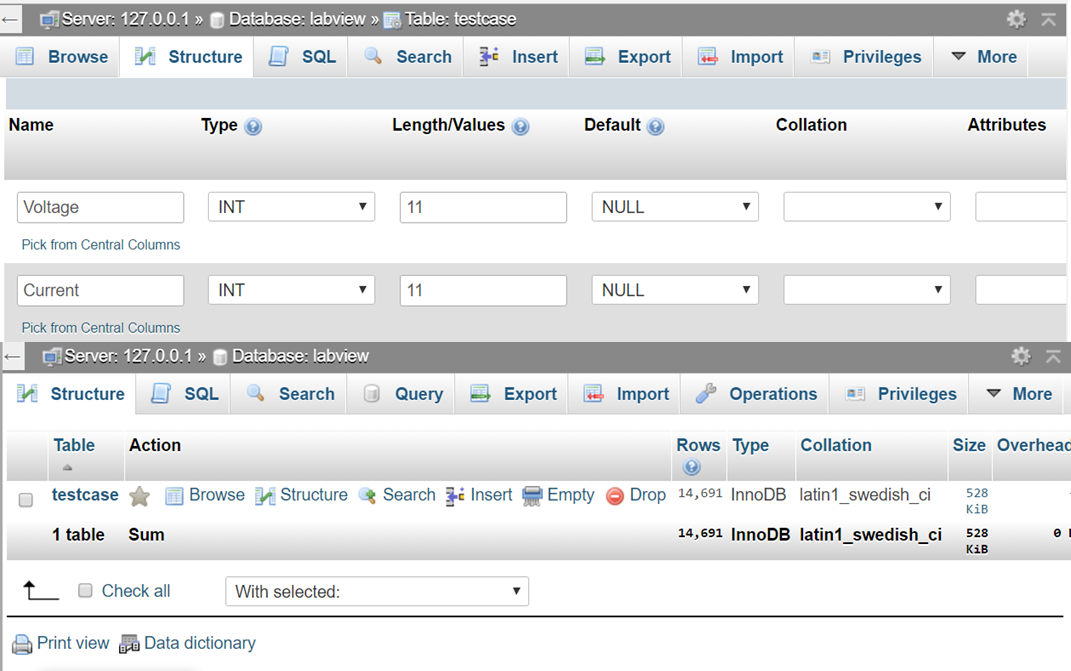


Figure 10: XAMPP MySQL Table creation and edit

By using the LabSQL add-on one can connect LabVIEW to the MySQL table created in XAMPP using the ADO Open Connection VI. The ADO Open Connection VI requires a DSN or data source name that is configured to connect to the testcase table created. DSN’s contain the information Open Database Connectivity driver (ODBC) needs in order to connect to the database. DSN’s require the IP address of the server and also the port. In this case the MySQL databases IP address is 127.0.0.1 and the port number is 3306.

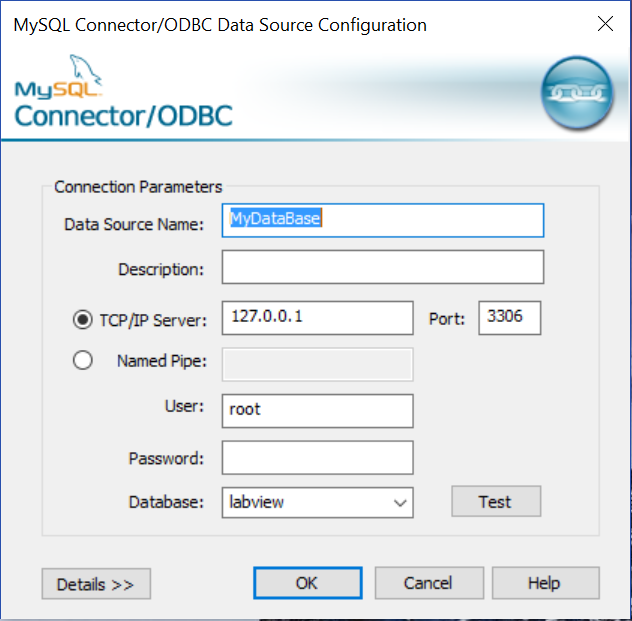


Figure 11: MySQL Data Source Configuration

Once the DSN is set up correctly the ADO Open Connection VI can be used without error. The front panel of the LabVIEW allows a user to input any DSN and change what values are sent to the MySQL database. The front panel also allows a user to decide what the name of each data set is and also have that sent to the MySQL database.

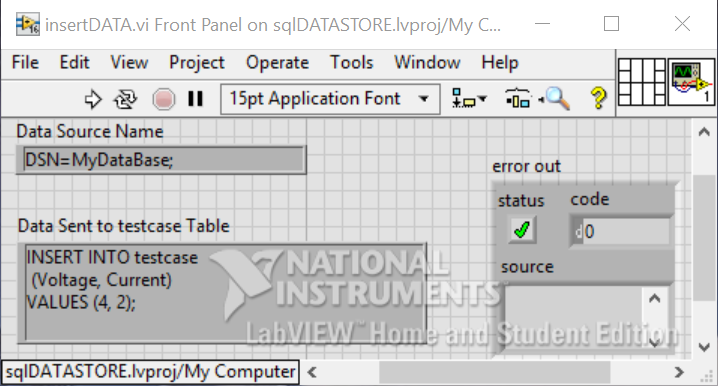


Figure 12: LabVIEW Front Panel

The block diagram shows how the program works. ADO Create Conncetion.vi creates a connection to the database. The ADO Create Conncetion.vi then connects to the ADO Open Connection.vi which opens the connection to the database. The ADO Open Conncetion.vi also requires the DSN input in order to connect to the correct database. Then the data and connection information is then sent through the ADO SQL Execute.vi which transmits the data to the MySQL database.

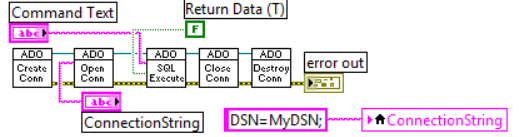


Figure 13: LabVIEW Block Diagram

Once the data is stored in the MySQL database it can be easily manipulated into graphs and be customized depending on user defined specifications.

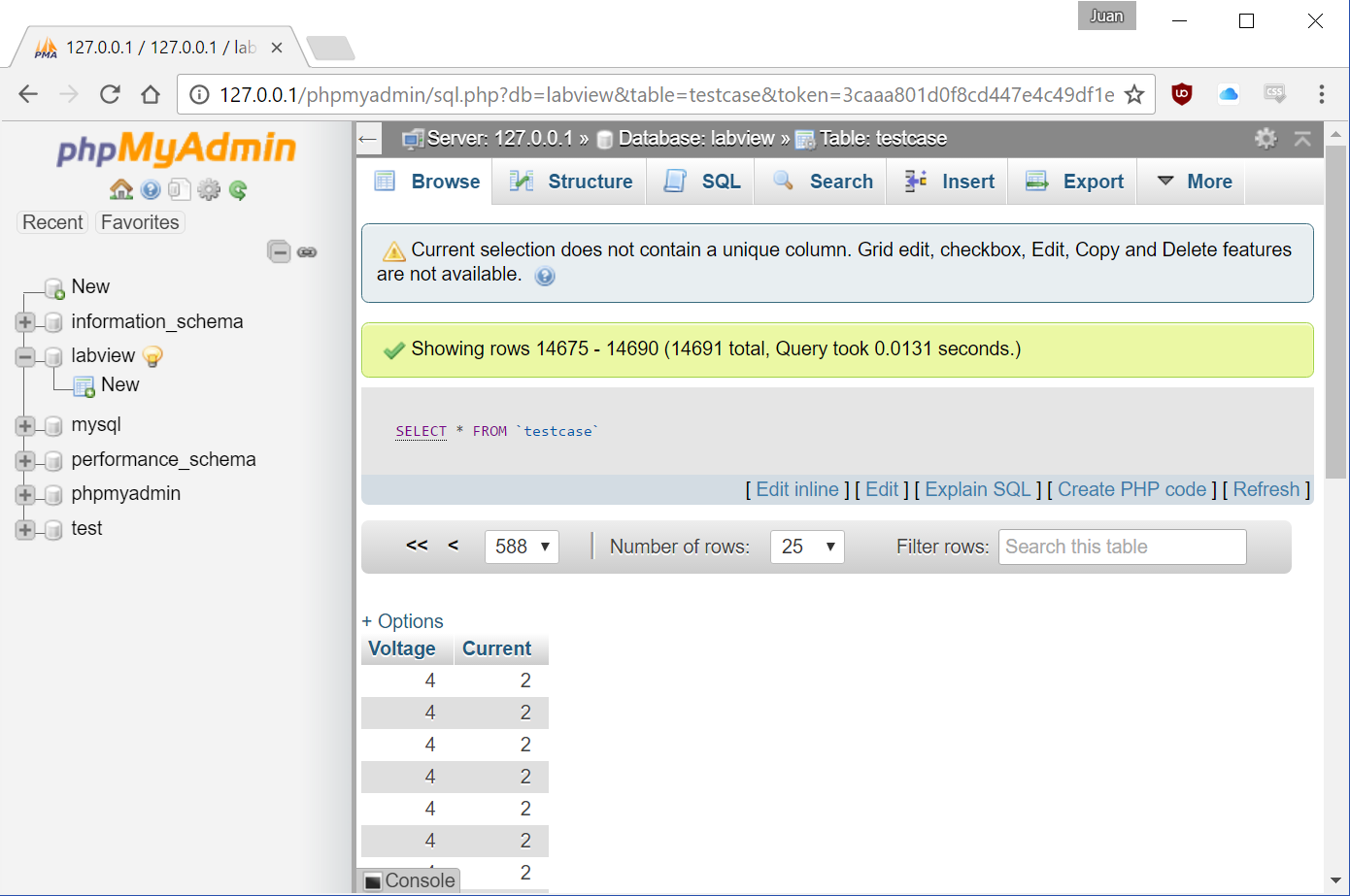


Figure 14: MySQL Table with transmitted LabVIEW data

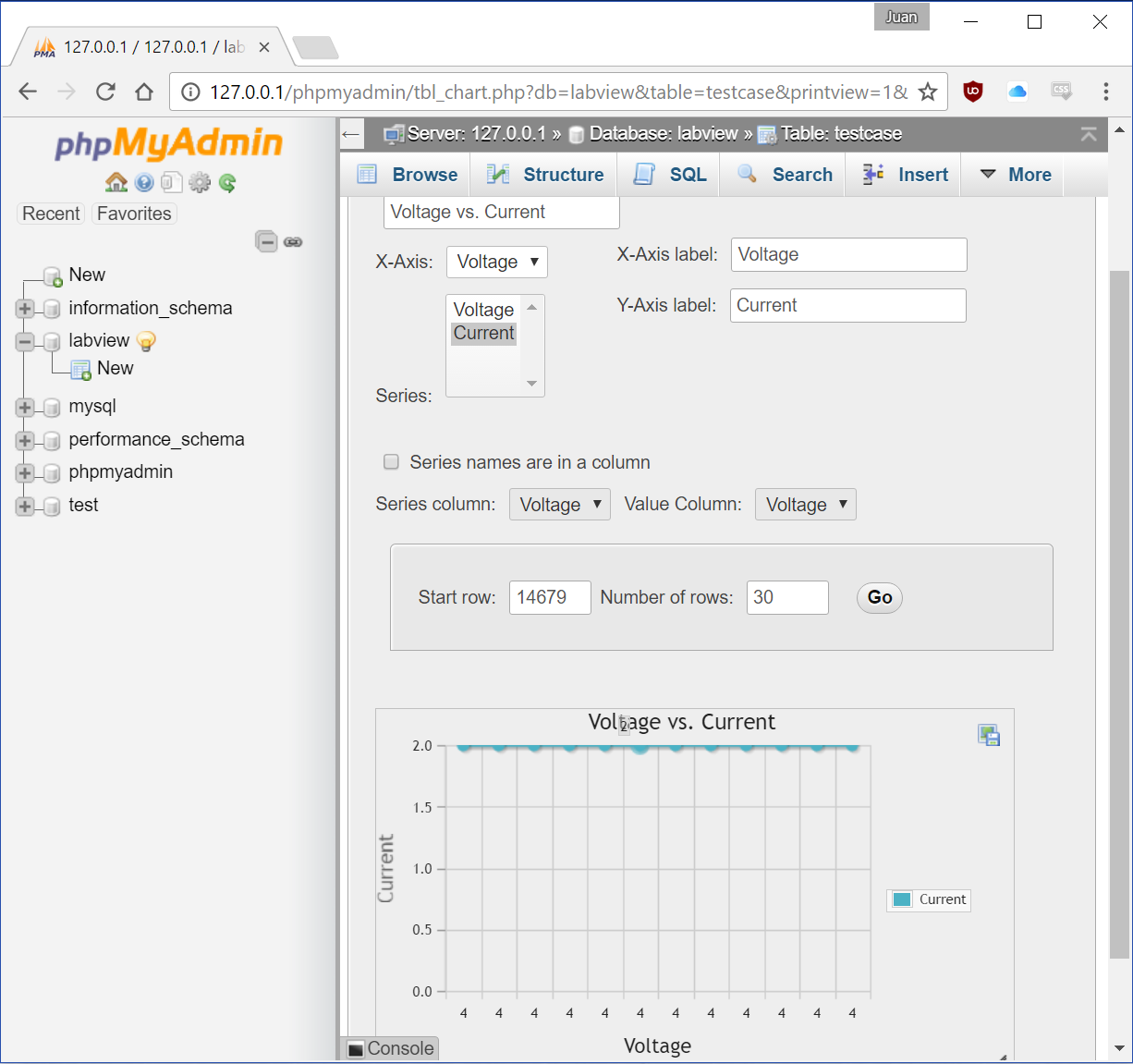


Figure 15: MySQL Graph of transmitted LabVIEW data

The front panel of the LabVIEW code has three graphs which each show the data acquired from each DUT. The value sent value is what determines the type of DUT. One for Diode, two for MOSFET and three for BJT.

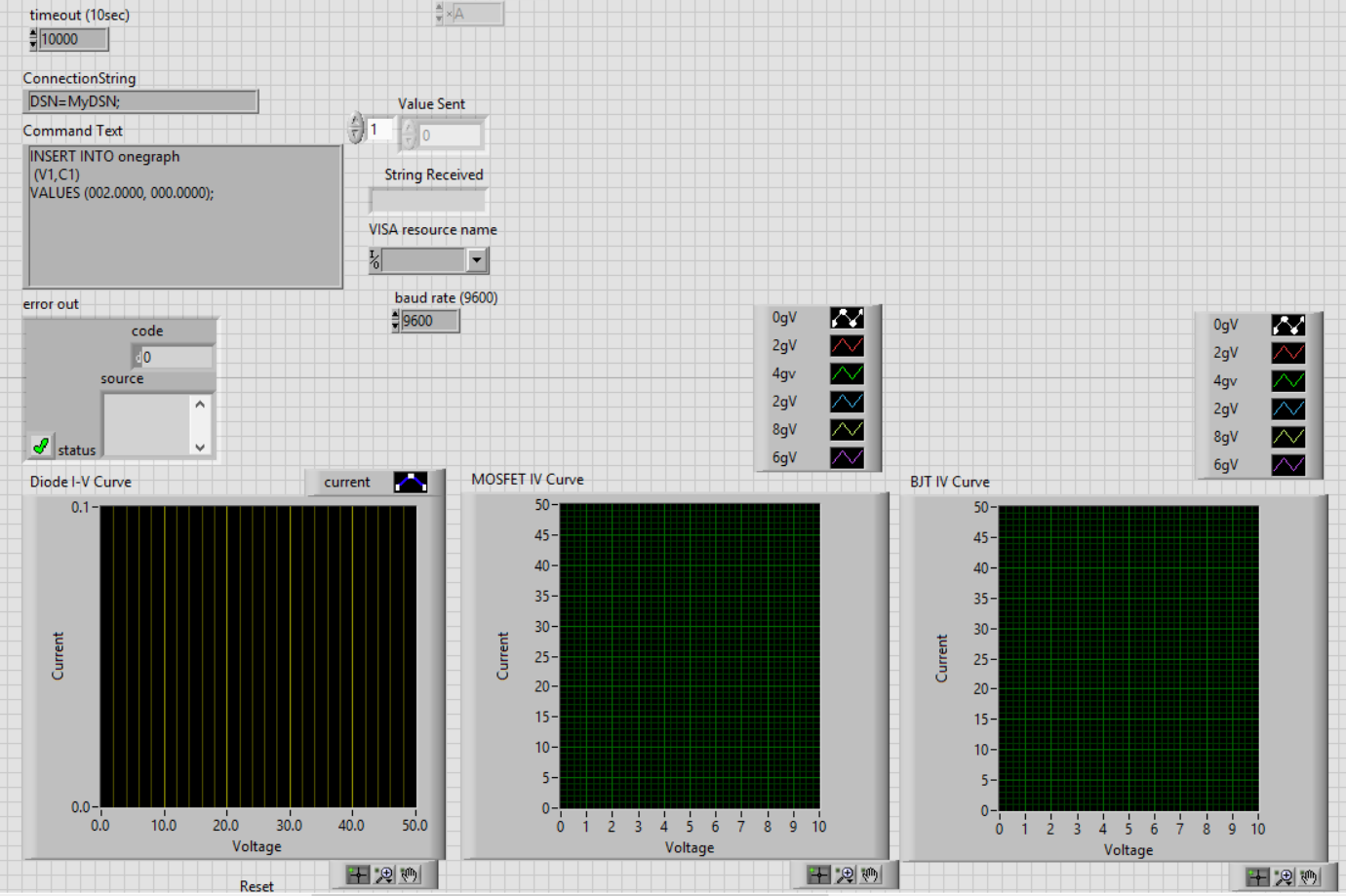


Figure 16: Front Panel of Project

Part 1 of the block diagram shows the inputs required to connect the microcontroller to the computer correctly. Part one also shows how a value is sent to the microcontroller. Part 2 of the block diagram shows how our data acquired. Data received from the microcontroller is initially voltage data, to acquire the correct current, the value received must be manipulated using division. Then an array indexed from 0 – 50 for our voltage value is plotted against the current value received. Data received must also be sent to a MySQL database. Part 3 shows how the Command text function is split into parts so that data coming in can be updated live while maintaining the strict syntax that Command Text requires. Command Text is a string value type, thus requiring the data received to be converted from integer to string value type. Part 3 also shows how this is done using the typecasting function in LabVIEW. Finally, part 4 shows how data received from the microcontroller is stored into an array using a shift register.

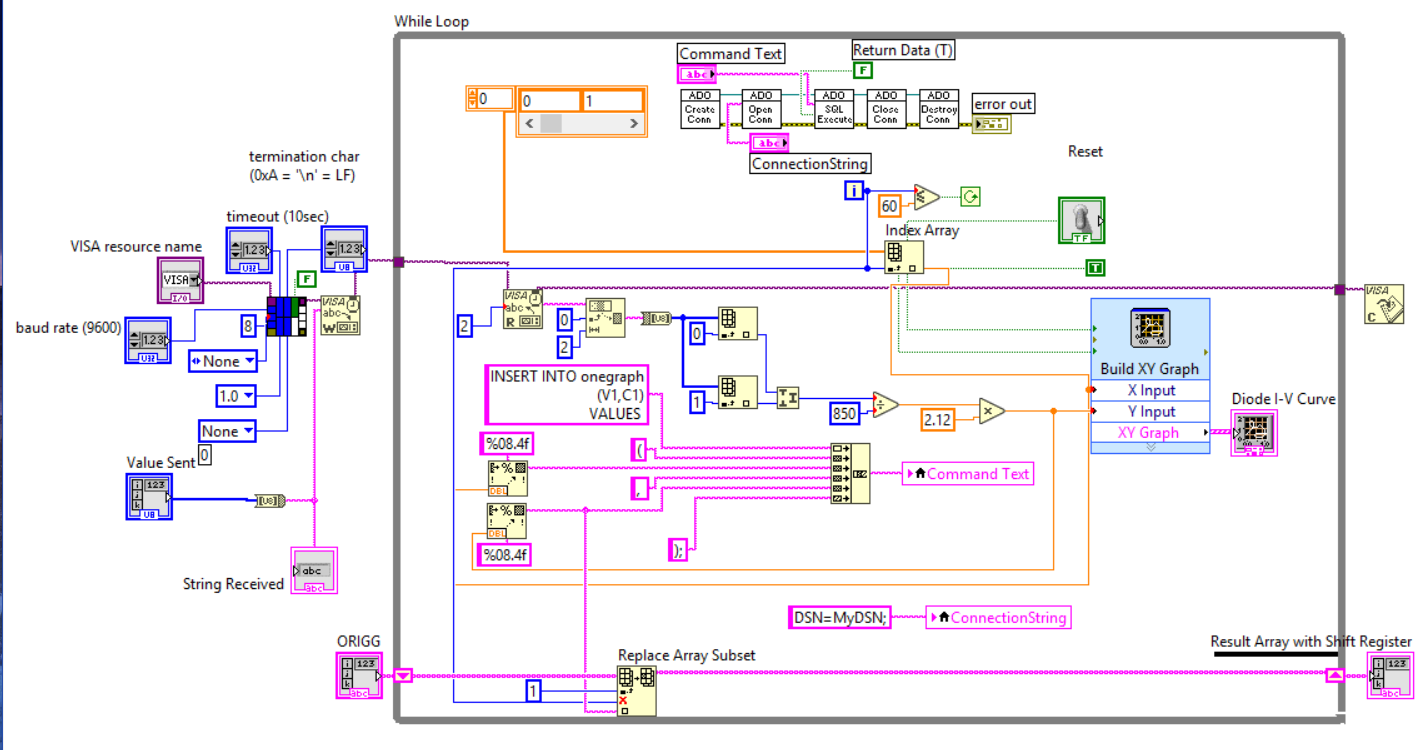


Figure 17: Block Diagram of Project

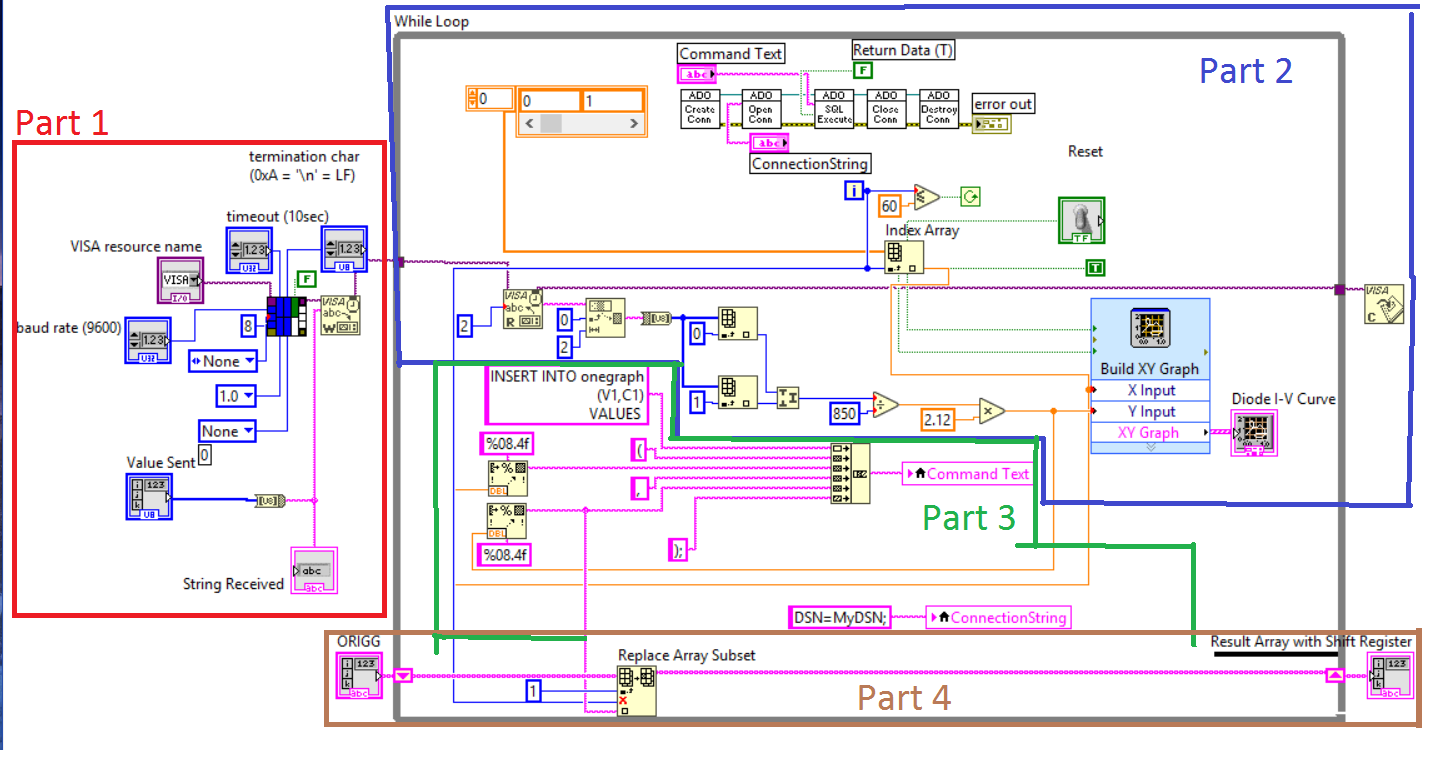
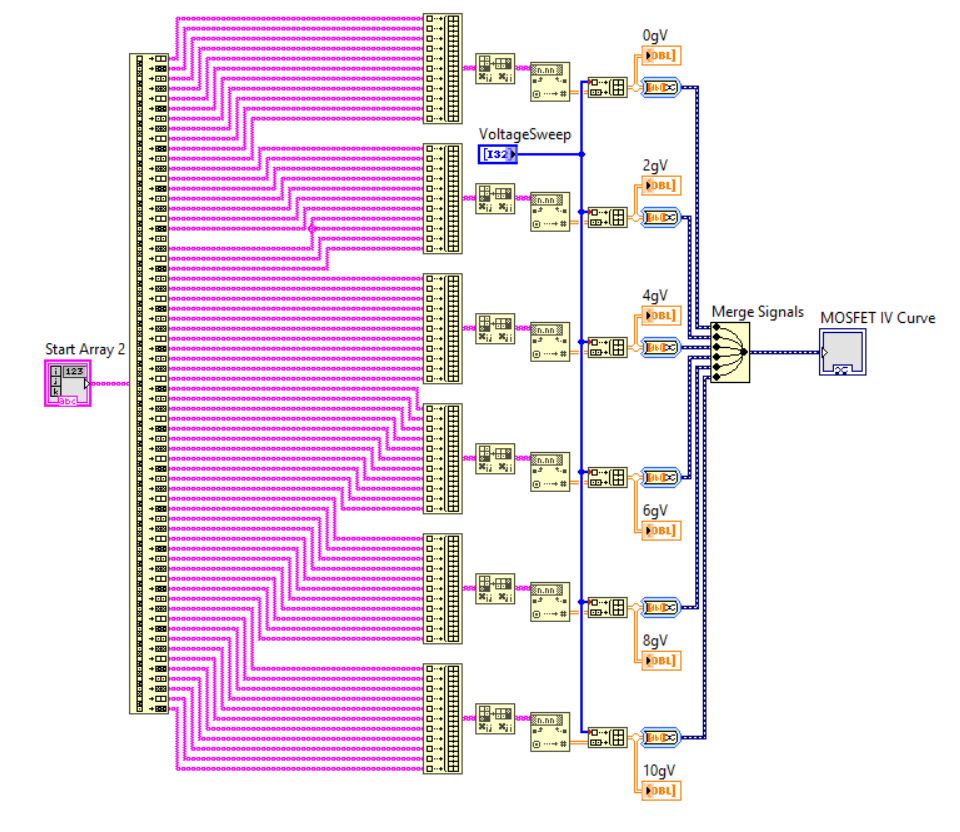


Figure 18: Block Diagram of Project with parts labels

Correcting testing a MOSFET and BJT diode requires testing at varying gate voltages. Therefore MOSFET and BJT data values received from the microcontroller must be grouped and plotted according to the data values gate voltage. The microcontroller test 6 varying gate voltages: 0, 2, 4, 6, 8, 10v. Each gate voltage is tested at 10 voltages: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50v. Figure 18 shows the array created in part 4 broken into 6 smaller arrays one for each gate voltage. The broken up current value 1D arrays are then combined with a 1D array consisting of voltage values creating a 2D array of voltage and current. Finally, the Merge Signals VI allows for multiple 2D arrays to be displayed on one graph.

Figure 19: Part 5 of Block Diagram

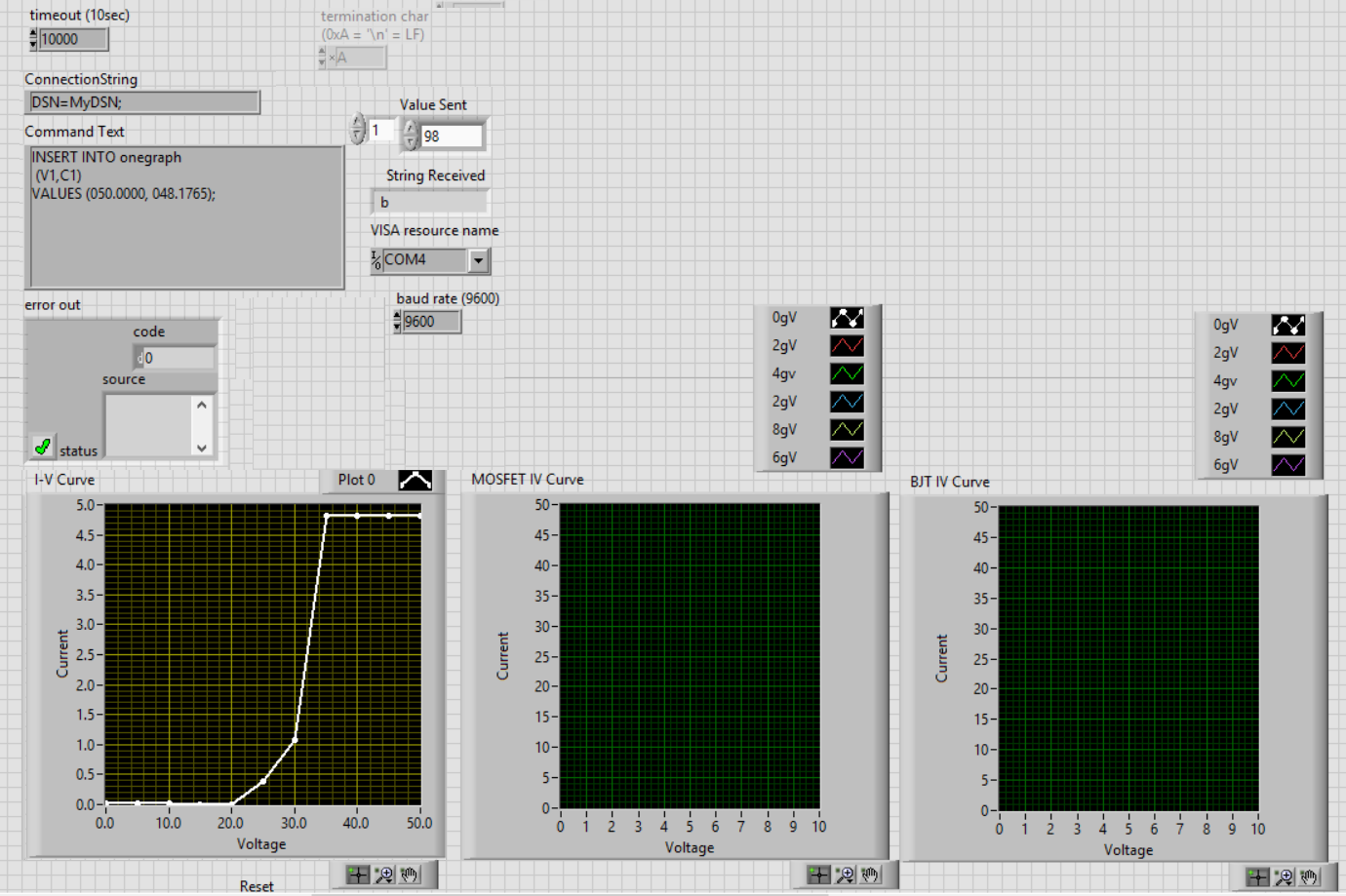


Figure 20: 25v Zener under test

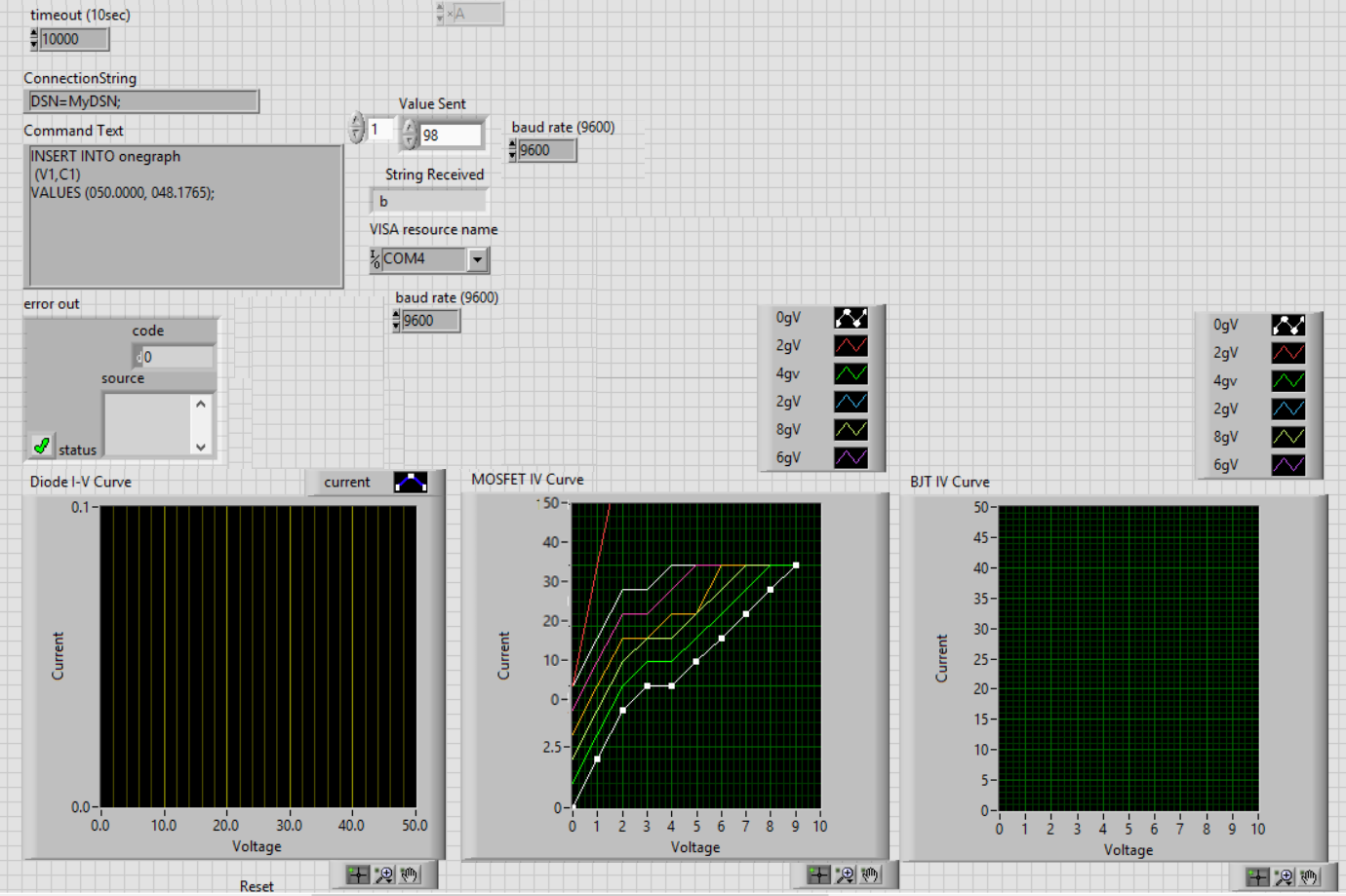


Figure 21: MOSFET under test