READ ME

Establish connection between Ignition and SQL DB [Part 1]

Setup details

I am using Ignition 7.8.2 for the implementation of the given task. I have used MySQL database for recording data of my OPC variable.

Configuring Database

I created a schema named “test” and used the Ignition server configuration wizard to connect it to Ignition.

Edit Database Connection

Main Properties	
Name	MyServer Warning: Changing the name of a database connection is risky. Any projects that refer to this connection by name (instead of refering to their project default) will start causing errors trying to connect to a connection that no longer exists. Please verify that no projects refer to this connection by name, and update the ones that do.
Description	
JDBC Driver	MySQL ConnectorJ The JDBC driver dictates the type of database that this connection can connect to. It cannot be changed once created.
Connect URL	jdbc:mysql://localhost:3306/test The Connect URL is JDBC-driver specific. It usually contains the address of the machine that the database is running on. The format of the MySQL connect URL is: jdbc:mysql://**host**:**port**/**database** With the three parameters (in bold) **host**: The host name or IP address of the database server. **port**: The port that the database server is running on. MySQL default port is **3306**. **database**: The name of the logical database that you are connecting to on the MySQL server.
Username	root
Change Password?	Check this box to change the existing password.
Password	
Password	Re-type password for verification.
Extra Connection Properties	zeroDateTimeBehavior=convertToNull;connectTimeout=120000;socket There is an extensive list of extra connection properties available for MySQL Connector/J. See the documentation for a table describing all connection properties.
Enabled	Disabling a connection will prevent communication to the target database. (default: true)
Validation Timeout	10000 The time in milliseconds between database validation checks. (default: 10,000)
Failover Datasource	- none - Another datasource that will be used to handle queries if this datasource faults.
Failover Mode	STANDARD STANDARD Standard failover mode means that this datasource will fail over when a connection cannot be retrieved, but when connectivity is restored, connections will again come from this datasource. STICKY Sticky failover mode means that once this datasource fails over, connections will continue coming from the failover datasource until the failover datasource itself fails or the Gateway is restarted. (default: STANDARD)
Slow Query Log Threshold	60000 Queries that take longer than this amount of time, in milliseconds, will be logged. This helps to find queries that are not performing well. (default: 60,000)

After putting all the details using help from Ignition online help I got a valid database connection.

Database Connections

Name	Description	JDBC Driver	Translator	Status	
MyServer		MySQL ConnectorJ	MYSQL	Valid	edit delete

Create new Database Connection...

Note: For details about a connection's status, see the Database Connection Status page.

Problems Encountered

I was first trying to connect Ignition to the server without making a schema named “test”. I was expecting Ignition to create it itself. I later created the schema myself and my connection got validated.

Establish connection between Ignition and OPC-UA Server [Part 2]

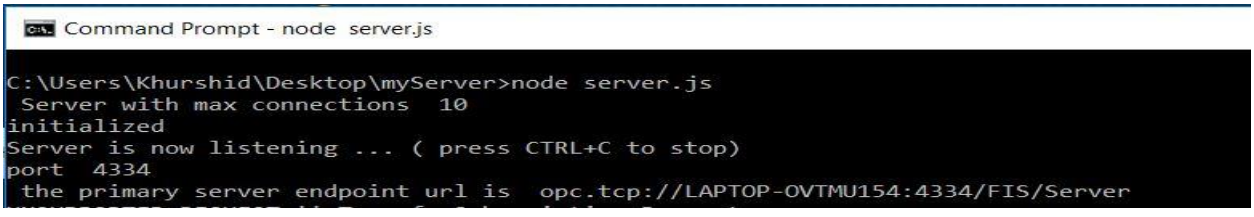
Setup details

I have used the instructions provided online from a github link (https://github.com/node-opcua/node-opcua/blob/master/documentation/sample_server.js) provided in the assignment requirement as well as that in the lecture to create my own OPC server.

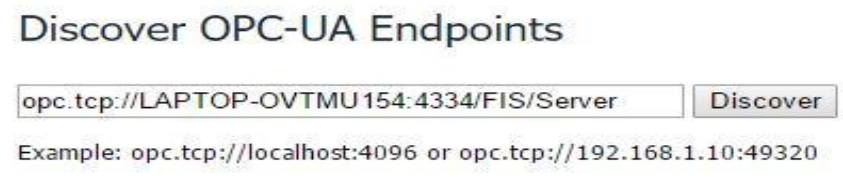
Configuration decisions

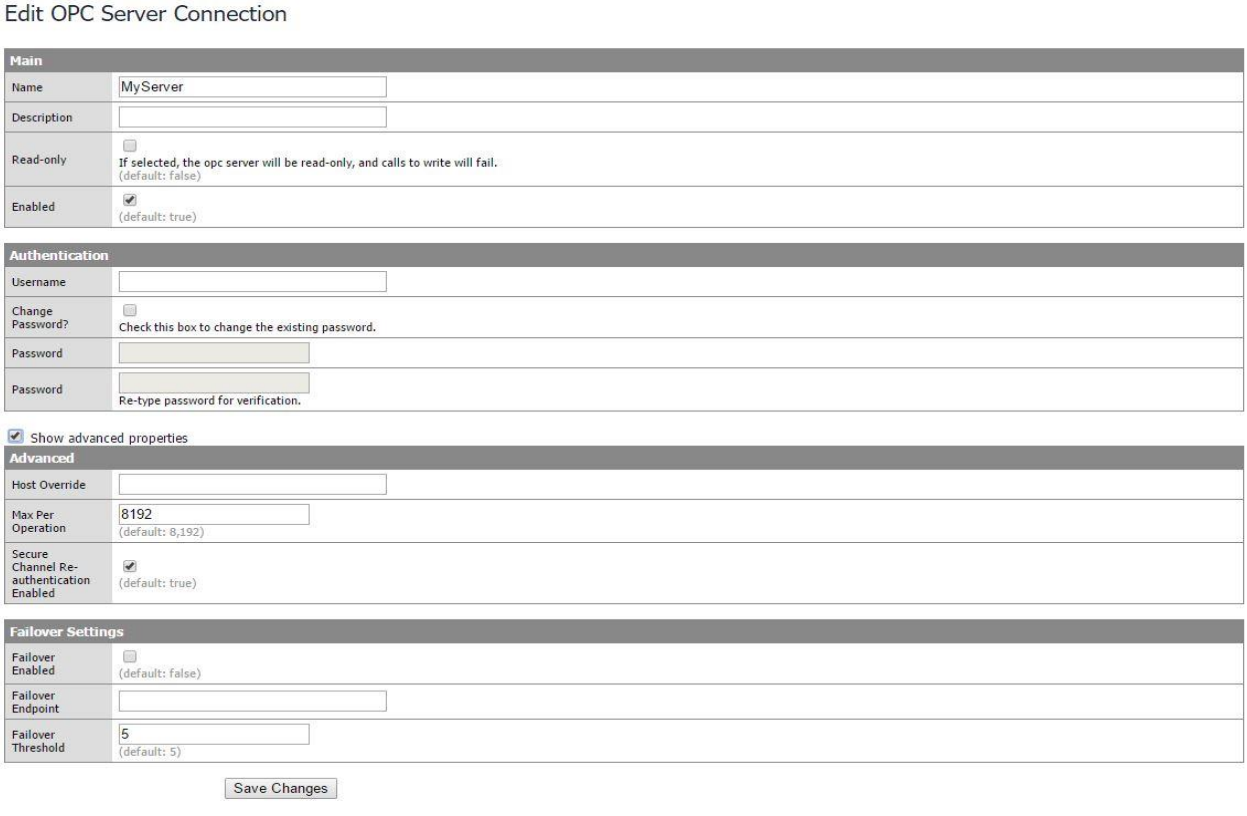
After creating a basic OPC Sever and Attaining its endpoint I used the OPC server Connection wizard to connect ignition to my OPC Server

Endpoint

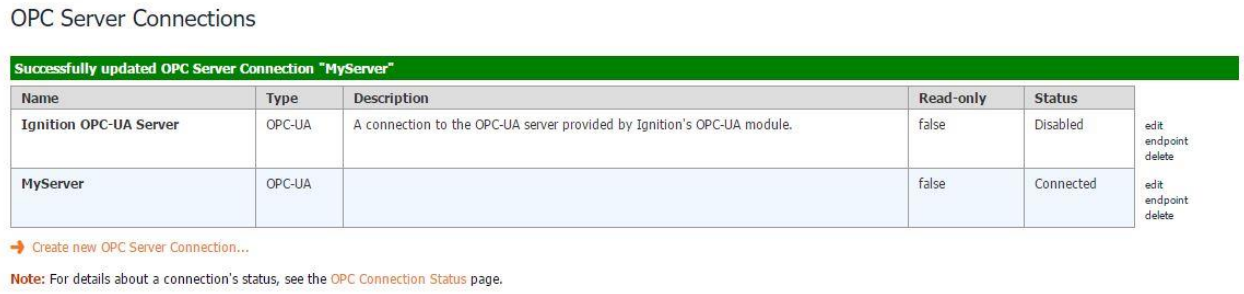


Wizard





Valid Connection



Programming decisions

For now I have just included the OPC module in my code. Wrote the Initialization function and started my server.

Code

```
var opcua = require("node-opcua");

var server = new opcua.OPCUAServer({

   port: 4334, // the port of the listening socket of the server

   resourcePath: "FIS/Server",

});

function initialization() {

        console.log("initialized");

        server.start(function() {

                console.log("Server is now listening ... ( press CTRL+C to stop)");

                console.log("port ", server.endpoints[0].port);

                var endpointUrl = server.endpoints[0].endpointDescriptions()[0].endpointUrl;

                console.log(" the primary server endpoint url is ", endpointUrl );

        });

}

server.initialize(initialization);
```

Problems Encountered

In the initial setup of the server, my connection was faulted because I gave some authentication details in the configuration which was preventing my connection from getting connected. I also cannot connect my server simultaneously with default ignition server. At least one of them gets “Faulted”. I tried to figure out the problem but was unable to do that so I simply disconnected the default one because I was not working with that.

Create OPC-UA Server with read-write variable [Part 3]

Programming decisions for the Variable

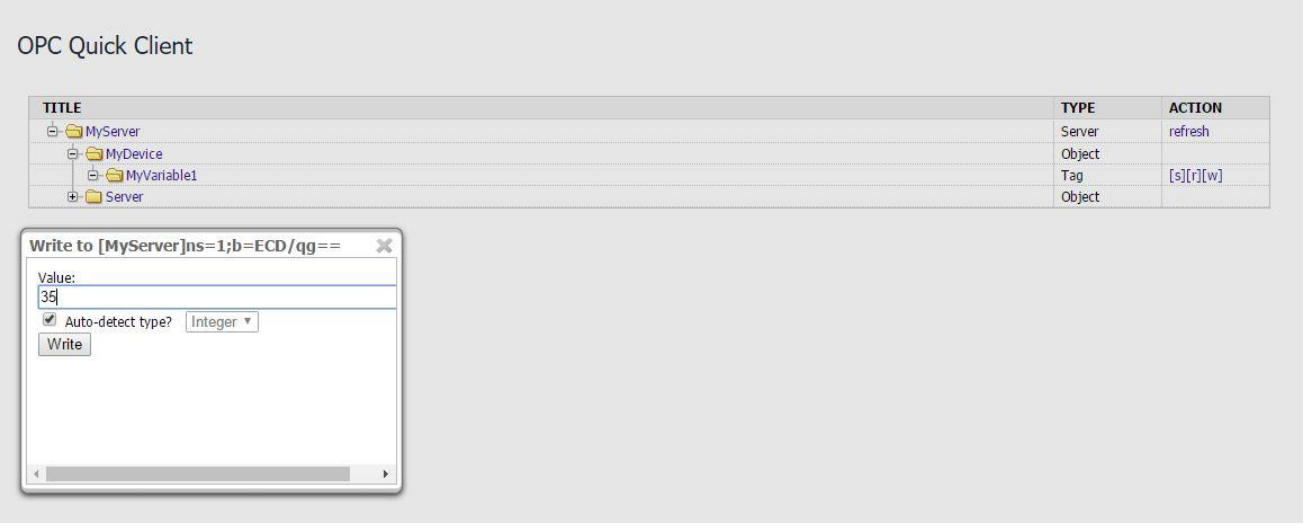
I defined a function within the initialization function in which I have initiated and defined the parameters of my read write variable. I have used the variable definition function template from github (https://github.com/node-opcua/node-opcua/blob/master/documentation/sample_server.js).

Code

```
function address_space(server) {
	var addressSpace = server.engine.addressSpace;
	var device = addressSpace.addObject({
		organizedBy: addressSpace.rootFolder.objects,
		browseName: "MyDevice"
	});
	var variable1 = 10.0;
	server.engine.addressSpace.addVariable({
		componentOf: device,
		nodeId: "ns=1;b=1020FFAA", // some opaque NodeId in namespace 4
		browseName: "MyVariable1",
		dataType: "Double",
		value: {
			get: function () {
				return new opcua.Variant({dataType: opcua.DataType.Double, value:
variable1 });
			},
			set: function (variant) {
				variable1 = parseFloat(variant.value);
				return opcua.StatusCodes.Good;
			}
		}
	});
}
```

Testing and modifying variable value

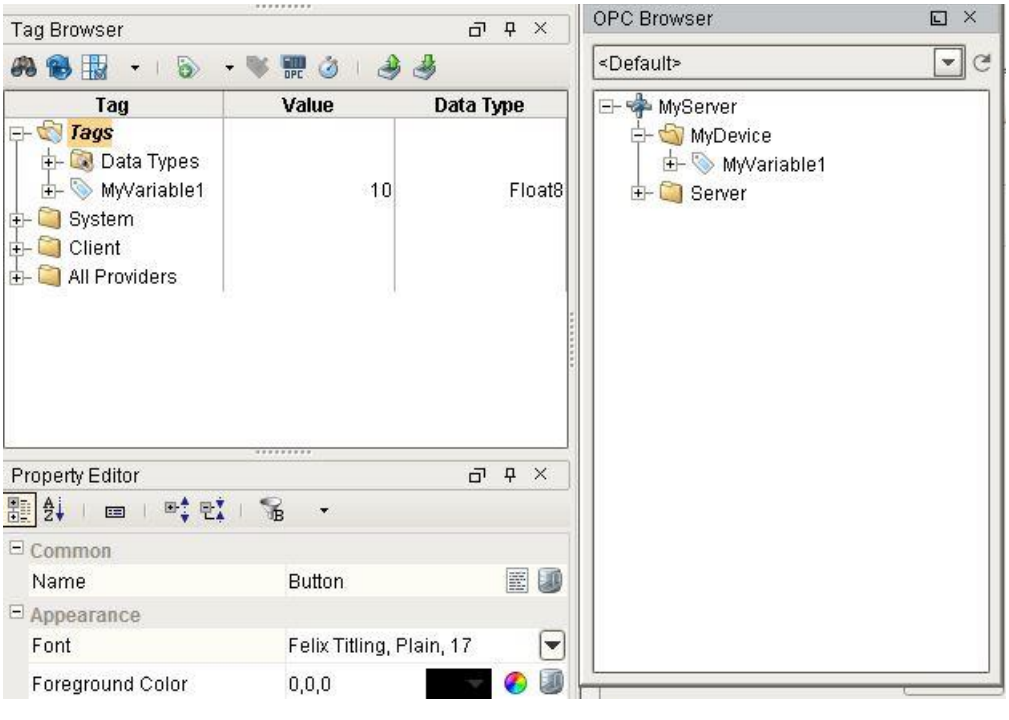
I have used the “Quick Client” module of ignition to write to my OPC variable and read it.



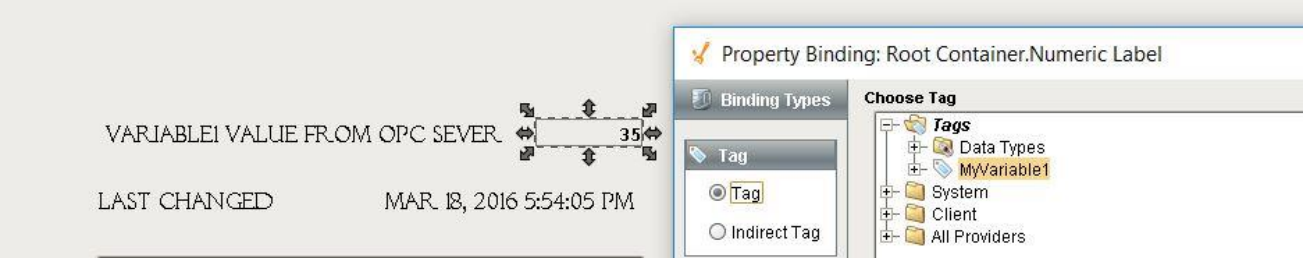
Create UI connected to the variable [Part 4]

Configuration Details

After Launching the Ignition designer I played around with it a little bit. Dropped a few fields and buttons to get the essence of what I have on my hand. I first imported my OPC variable tag to my project.



After importing the tag I associated it with a numeric field and found out that whenever I wrote a value to my OPC variable from “Quick Client” the data in my numeric field in the UI automatically got updated.



After that I also included text fields which show the timestamp when the OPC variable was last updated and an Easy Chart to my UI. Below is picture of my End UI.

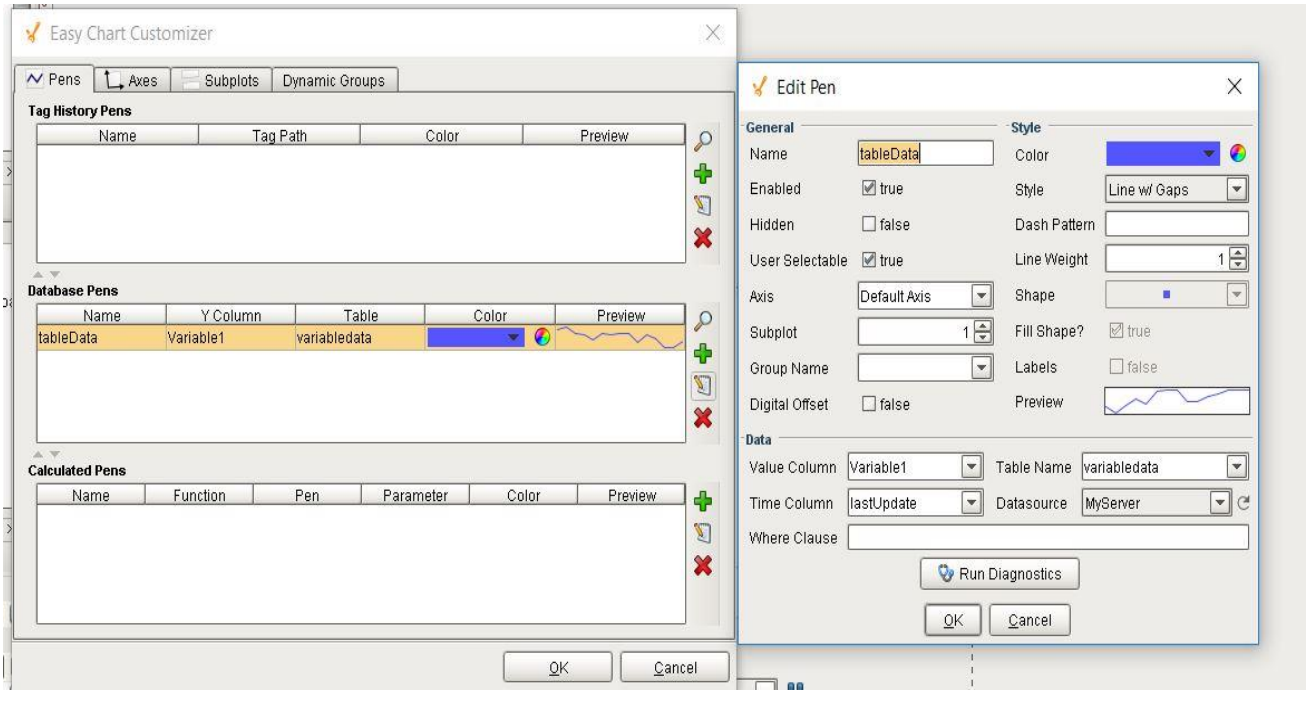
Problems faced

I first had questions regarding which medium should be used to create this UI as it was not very clearly mentioned in the assignment requirements. I had to take help from my instructor to figure it out.

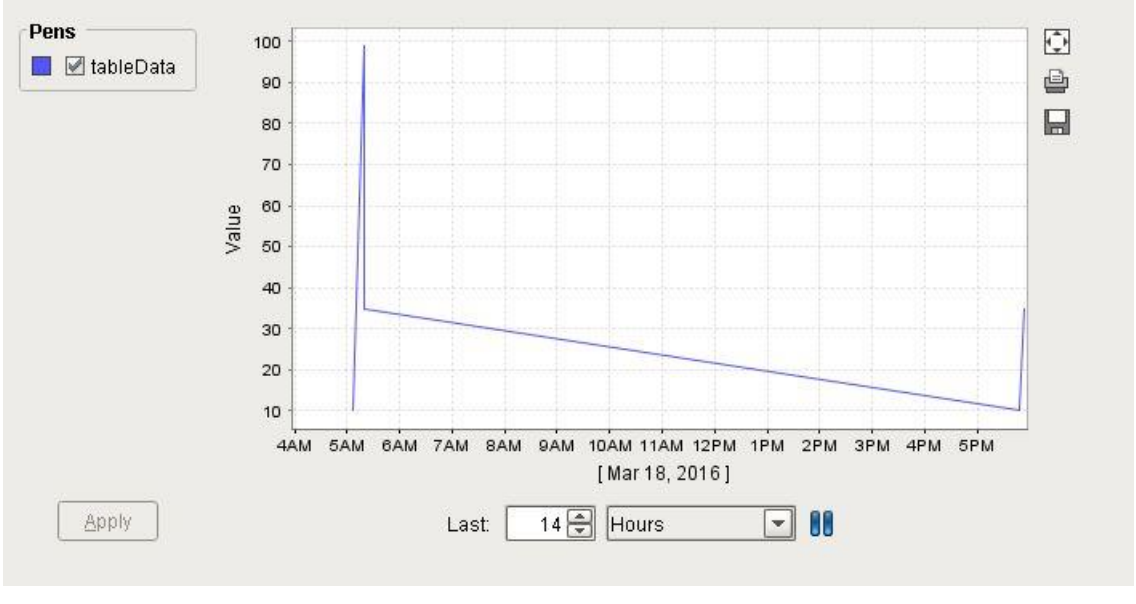
Plot variable history [Part 5]

As suggested in the assignment requirement I have used the Easy Chart to show the historical data of my variable from the database.

Below is the Configuration of Easy Chart to get data from the database:



I used the real-time option from the graph properties to give me value plot in real time.



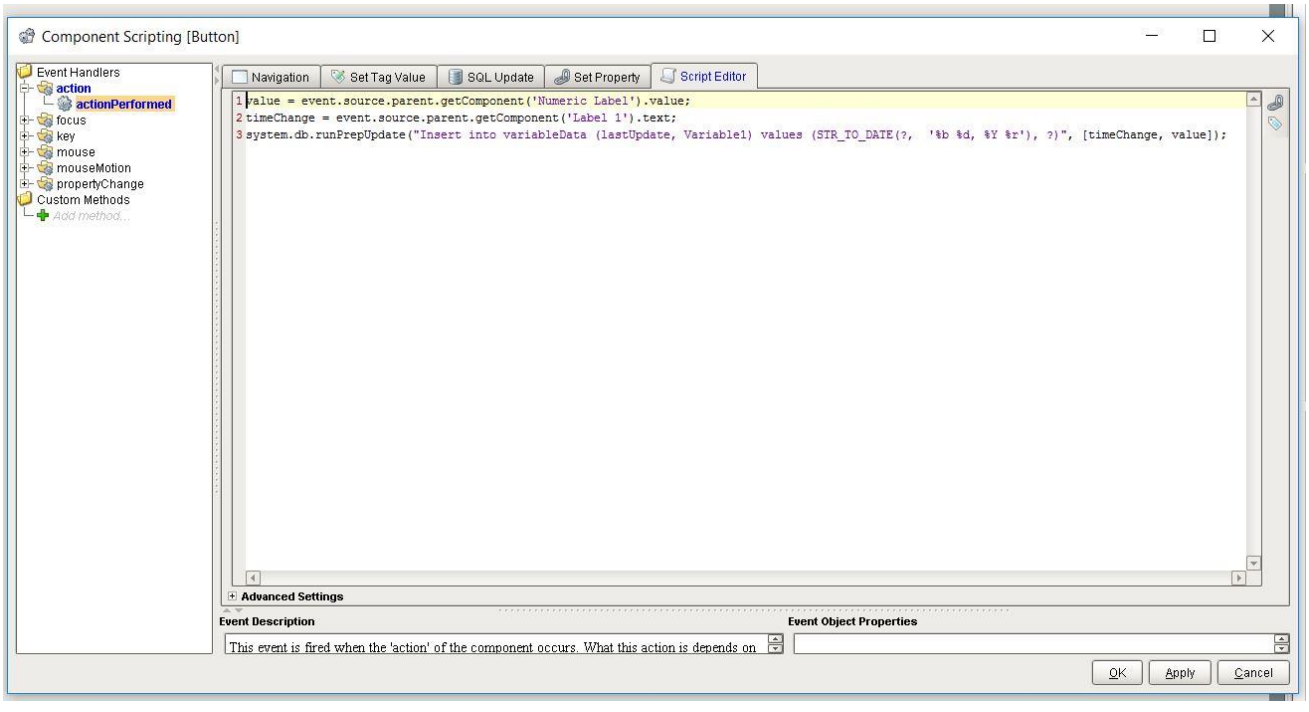
The X axis is the value of my OPC variable with time on the Y axis. The whole graph shows the changing value of my variable over a period of time.

Problems Faced

The time stamp filed in the database had an attribute type of text so the Easy Chart was not able to take its value to populate the chart. I later had to change to datetime for it to get it properly.

Analyze interactions with DB [Part 6]

I used the “Update Database” button to put the value and recorded time stamp of the OPC variable to my Database table. Below is the screenshot and SQL script I used for it.



Script

value = event.source.parent.getComponent('Numeric Label').value;

timeChange = event.source.parent.getComponent('Label 1').text;

system.db.runPrepUpdate("Insert into variableData (lastUpdate, Variable1) values (STR_TO_DATE(?, '%b %d, %Y %r'), ?)", [timeChange, value]);

Table View

Result Grid | Filter Rows: | Edit:

id	lastUpdate	Variable1
1	2016-03-18 05:06:28	10
2	2016-03-18 05:18:48	99
3	2016-03-18 05:20:20	35
4	2016-03-18 17:48:24	10
5	2016-03-18 17:54:05	35
NULL	NULL	NULL

Problems Faced

I tried to update my database automatically every time the OPC variable was updated but I had a few errors which included multiple entries to the database variable for a single update, so I introduced a separate button for it and did it manually.

I had to tailor the script to change the time stamp from “text” format to “datetime” using proper syntax division in MySQL.