How to use the Phoenix Simulator

(With associated programs)

1. Overview (Setting up)

The Phoenix simulator is a modified version of the AGA-simulator that can read data from the Euro Truck Simulator 2 game (ETS2) by using ETS2's SDK and a third-party ETS2 Telemetry Server. The following instructions assume that you want to edit the automotive signals read from the game, so we strongly recommend having Eclipse IDE installed.

1. Buy and install Euro Truck Simulator 2.
2. Download and install ETS2 Telemetry Server. Dependencies should be installed automatically.
3. Download and install Gradle version 1.12. Gradle is needed to properly setup the Phoenix Simulator, which is not compatible with newer Gradle versions.
4. Edit your Path environment variable (In Windows 8, you can find it by opening Control Panel - System and Security - System and selecting *Advanced System Settings* (Figure 1). In *System Properties* select the *Advanced* tab and click on the *Environment Variables* button (Figure 2). Find the *Path* variable in the System Variables list, select it and click the *Edit* button. In the variable field, move the cursor to the end and add the path to your newly installed Gradle/bin directory (Figure 3).
5. Download the Phoenix Simulator. Using a terminal, go into the directory and run *gradle build eclipse*. The command should build an Eclipse project and download all required dependencies. You should be able to import it as an Eclipse project afterwards.

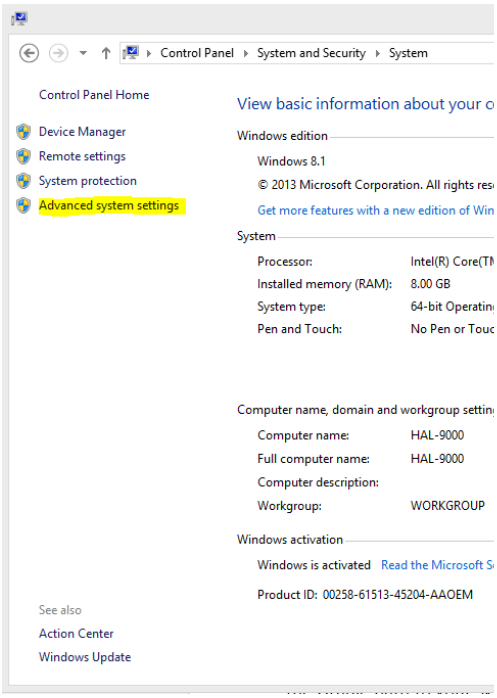

Figure 1 - Advanced System Settings

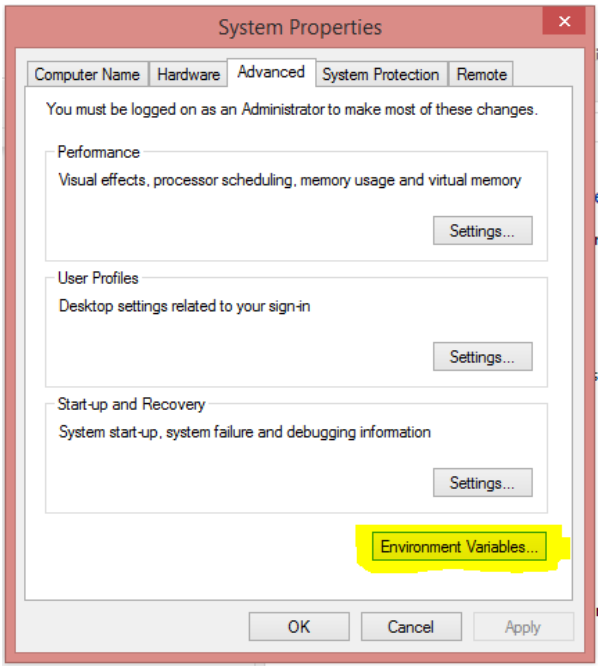

Figure 2 - System Properties

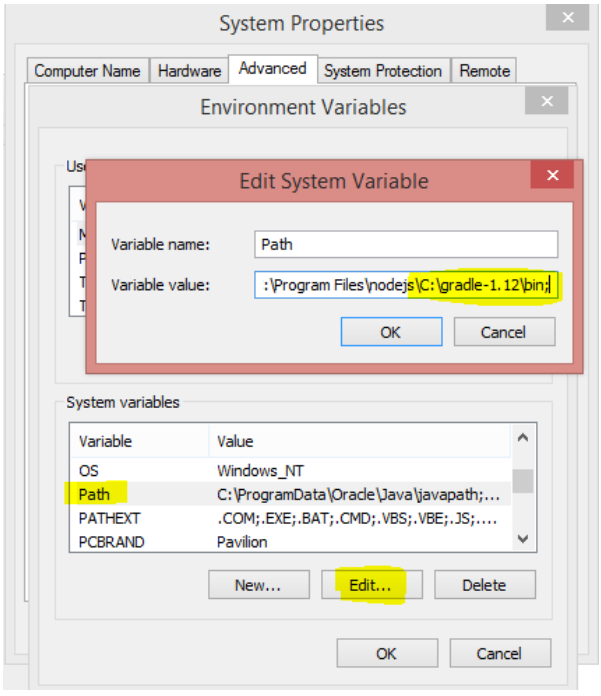

Figure 3 - Editing the PATH variable

2. Using the Phoenix Simulator

When you've completed part 1, you are ready to start using the Phoenix simulator (and develop for it).

The class ets2/EuroTruck.java is the only file needed to add, remove and edit automotive signals. Be sure to be consistent on which variables you need for your AGA application.

1. Add/remove/edit needed declared variables (Figure 4). Pay attention on their type, and make sure they match FMS specifications.
2. Inside the *getProvidingSignals()* method, add/remove needed automotive signals.
3. At the end of the *run()* method, add/remove/edit the mapping of variables into FMS signals (Figure 5). Make sure to properly match the declared variable names from step 1 and the automotive signals in step 2.
4. Go to the *extractValues()* method (Figure 6). This method reads the raw data from the ETS2 Telemetry Server and extracts the signal values, assigning them to your previously declared variables in step 1.
5. When editing the *extractValues()* method, you should also open your browser and point it to the raw data file located in *http://127.0.0.1:25555/api/ets2/telemetry*. Figure 7 shows a snippet of the data.
6. Example of how to properly assign a data value: the variable *speed* in EuroTruck.java is the 6:th JSON-value on the ETS2 server, and the actual data value starts at the 13th character of the string. Therefore the command to properly assign the variable *speed* value is:

   speed = Float.parseFloat(values[6].substring(13));

```
    //Declare signals variables
    short currentGear;
    float fuelLevel;
    float fuelConsumption;
    float speed;
    float brake;
    float engineSpeed;

    //Simulator Gateway Constructor
    public EuroTruck(SimulatorGateway gateway) {
        super(gateway);
    }

    //Define provided signals
    @Override
    public int[] getProvidingSignals() {
        return new int[] { AutomotiveSignalId.FMS_WHEEL_BASED_SPEED,
                AutomotiveSignalId.FMS_CURRENT_GEAR,
                AutomotiveSignalId.FMS_ENGINE_SPEED,
                AutomotiveSignalId.FMS_TACHOGRAPH_VEHICLE_SPEED,
                AutomotiveSignalId.FMS_FUEL_RATE,
//              AutomotiveSignalId.FMS_FUEL_LEVEL_1,
//              AutomotiveSignalId.FMS_HIGH_RESOLUTION_TOTAL_VEHICLE_DISTANCE,
//              AutomotiveSignalId.FMS_ACCELERATOR_PEDAL_POSITION_1
                };
    }
```

Figure 4: EuroTruck.java, variable declarations and getProvidingSignals() method

```
            if (signalUpdate.length() > 30) {
                //map ETS2 values into AGA/FMS signals
                extractValues(signalUpdate);
                gateway.sendValue(
                        AutomotiveSignalId.FMS_WHEEL_BASED_SPEED,
                        new SCSFloat(speed));
                gateway.sendValue(AutomotiveSignalId.FMS_CURRENT_GEAR,
                        new SCSShort(currentGear));
                gateway.sendValue(AutomotiveSignalId.FMS_ENGINE_SPEED,
                        new SCSFloat(engineSpeed));
                gateway.sendValue(AutomotiveSignalId.FMS_TACHOGRAPH_VEHICLE_SPEED,
                        new SCSFloat(brake));
                gateway.sendValue(AutomotiveSignalId.FMS_FUEL_RATE,
                        new SCSFloat(fuelConsumption));
            }
```

Figure 5: run() method, mapping telemetry data into AGA/FMS signals.

```java
/*Method that stores telemetry data values into a string array,
and initialise signals variables with corresponding values*/
private void extractValues(String signalUpdate) {
    String[] values = signalUpdate.split(",");

    try {
        speed = Float.parseFloat(values[6].substring(13));
        currentGear = Short.parseShort(values[16].substring(7));
        engineSpeed = Float.parseFloat(values[20].substring(12));
        brake = Float.parseFloat(values[31].substring(12));
        fuelConsumption = Float.parseFloat(values[24].substring(25));
        fuelConsumption *= 100;
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 6: extractValues() method.


```
{"connected":false,"gameTime":"0001-01-
01T00:00:00Z","gamePaused":false,"telemetryPluginVersion":"0","gameVersion":"0.0","trailerAttached":false,"truckSpeed":0.0,
.0,"rotationZ":0.0,"gear":0,"gears":0,"gearRanges":0,"gearRangeActive":0,"engineRpm":0.0,"engineRpmMax":0.0,"fuel":0.0,"fue
0.0,"gameBrake":0.0,"gameClutch":0.0,"truckMass":0.0,"truckModelLength":0,"truckModelOffset":0,"trailerMass":0.0,"trailerId
01T00:00:00Z","sourceCity":"","destinationCity":"","sourceCompany":"","destinationCompany":"","retarderBrake":0,"shifterSlo
linkerLeftActive":false,"blinkerRightActive":false,"blinkerLeftOn":false,"blinkerRightOn":false,"lightsParkingOn":false,"li
ghtsReverseOn":false,"batteryVoltageWarning":false,"airPressureWarning":false,"airPressureEmergency":false,"adblueWarning":
eConsumpton":0.0,"oilPressure":0.0,"oilTemperature":0.0,"waterTemperature":0.0,"batteryVoltage":0.0,"lightsDashboard":0.0,"
```


Figure 7: Snippet of the raw data generated by the ETS2 Telemetry Server.

3. Running everything

1. Start the ETS2 Server (.../*ets2-telemetry-server/server/Ets2Telemetry.exe)*
2. Start your AGA App virtually or in an android device connected to the PC with the simulator
3. Use adb to forward ports using the following commands:
   adb forward tcp:9898 tcp:9898
   adb forward tcp:9899 tcp:9899
   adb forward tcp:8251 tcp:8251
4. Start the Phoenix Simulator by running the *Main.java* class in Eclipse. It is also possible to save the project as an executable jar file.
5. Start Euro Truck Simulator 2, and enjoy your new test environment!