# Different possibilities to implement control strategies for HVAC systems

## Table of contents

1. Introduction
2. Dymola 2018 – Programming language: Modelica
3. WinPython 3.6 – Programming language: Python

### Introduction

This short documentation will show a few different possibilities to implement control strategies and run them on a PLC. The focus will be on the programming languages Python and Modelica. Afterwards, the basic programming languages Functional Block Diagram (FBD) and Structured Text (ST), which are basic programming languages for PLCs based on IEC-61131-3 standard, will be considered.

The main topic is the step by step implementation and to identify the pros and cons of each language.

We will use a model created in Dymola and its standard library, which simulates “a final air valve to regulate the room temperature”. The aim will be to implement this Model into a PLC, by using different programming languages and tools.  
  
The used PLC will be an Embedded PC with TwinCAT 3 runtime from BECKHOFF, which provides variable reading and writing access via ADS and OPC.

### Dymola – Programming language: Modelica

The programming language Modelica allows the user to create complex Models of multi-physical systems. Simulating these complex models to understand the interaction between different components is the goal.

The implementation of a control strategy will be presented in two different ways.

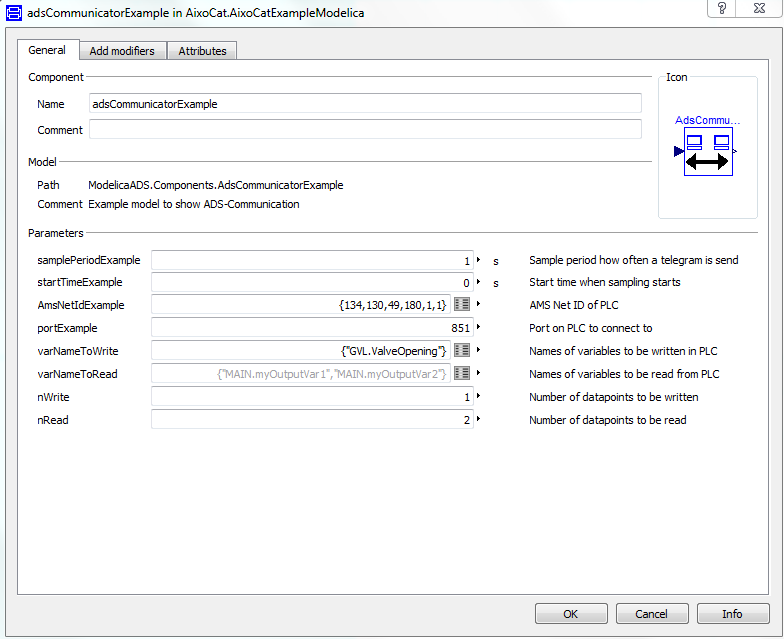
**The first possibility: FMU(Functional Mockup Units)**

It is possible to export our model as an FMU. This FMU can be imported via the Beckhoff Interface “TC3 Target for FMI” directly. This feature has yet to be published.

**The second possibility: ModelicaADS**

ModelicaADS is a free library provided by Georg Ferdinand Schröder. It provides functions for communicating with TwinCAT devices and uses the C API AdsDLL.dll which is available on infosys.beckhoff.com.

The implementation of Dymola to TwinCAT 3 via ModelicaADS is easy to use. The only requirement is to use a Dymola in 32-bit Mode. Load the ModelicaADS package into Dymola and connect the Real-Output (“ValveOpening”) of our example to the ADS-component (“ADSCommunicatorExample”). Now you have to open the parameters-window.



1: Parameters of adsCommunicatorExample

You have to change three parameters within the window above. The AMS Net Id of your TwinCAT 3 , the Port and the Variable Name which you want to send to TwinCAT3. After that you can start the simulation. Now the Variable “GVL.ValveOpening” receives its value from our Dymola model.

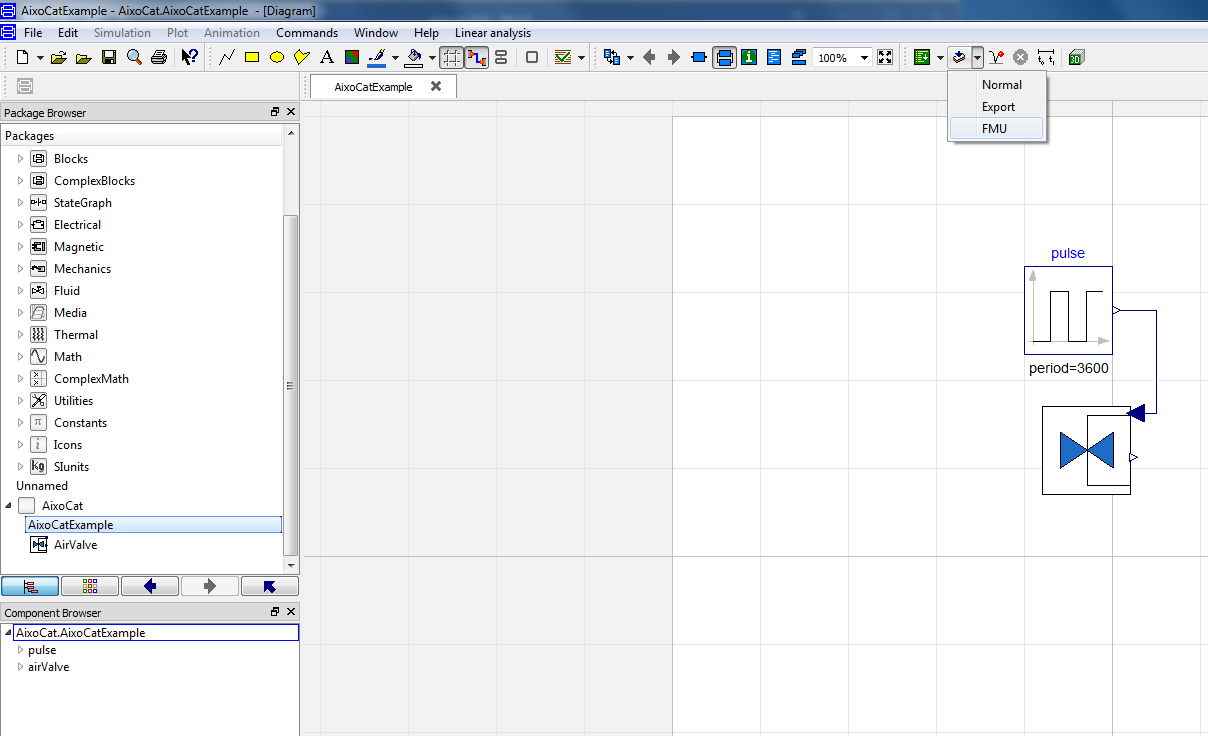
### WinPython 3.6 – Programming language: Python

The same model is used in all programming languages. For this reason the Modelica model was exported through the FMU Export-function of Dymola. Since it is part of the implementation we will begin here.

How do we export our Dymola model via FMU?

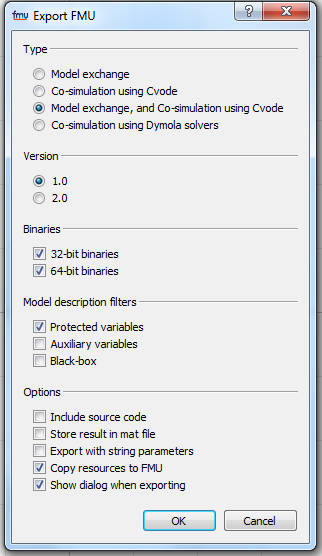
We have two classes in our model. The first one is the model of an air valve control named “AirValve”.

The second one is our Example where we initiate a pulse for the air valve control. This class is named “AixoCatExample” and is the only class we need to export by clicking on the translator and choosing the FMU-Option.



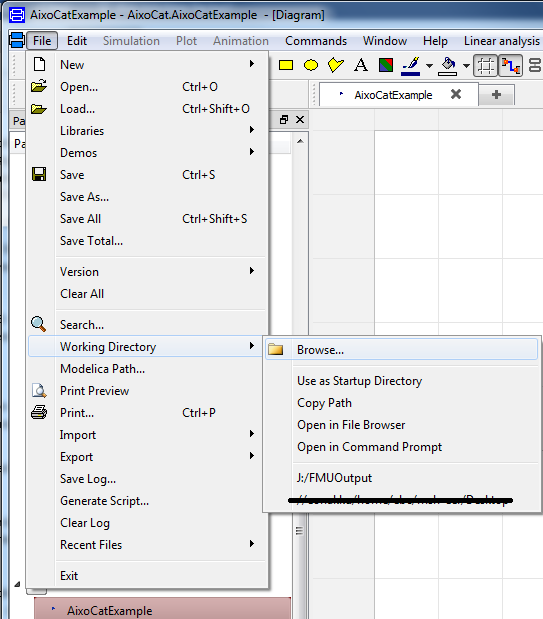
2: Dymola: FMU Export

The next thing you see are the configurations for the FMU –Export. It is recommended to use the same configurations as shown below.



3: Dymola: FMU-Export configurations

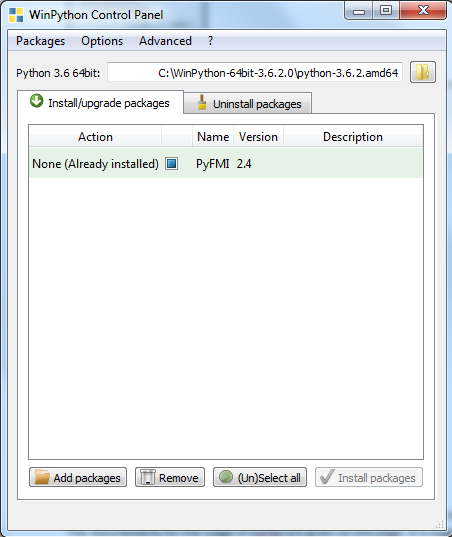
The FMU will be saved in the working directory, so you should make sure to either know where the working directory is or where you want it to be.



4:Where to change the working directory

How can we import FMU-Files to WinPython ?

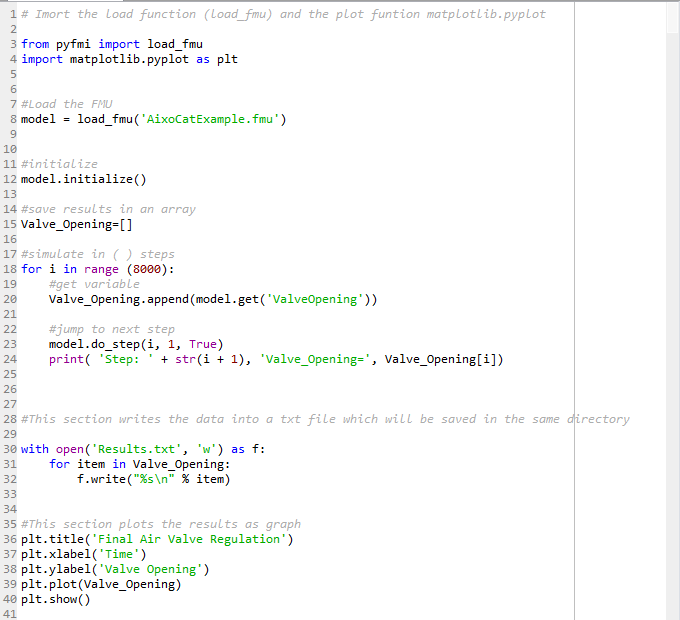
First you have to install the package called PyFMI for WinPython. Which you can find on [www.PyPI.org](http://www.PyPI.org). The requirements for the usage of PyFMI are given on this page. It is suggested to install the necessary packages via Python Wheels. The requirements regarding PyFMI are still the same. First you need to download the necessary Wheel-Files, suitable four your working system, of the following page: [www.lfd.uci.edu/~gohlke/pythonlibs/](http://www.lfd.uci.edu/~gohlke/pythonlibs/). The next step is to open the WinPython Control Panel from the installation directory. First you have to install the following packages: numpy+mkl, assimulo, lxml and scipy. Now you can install PyFMI and are ready to write the code for the implementation.



5:WinPython Control Panel

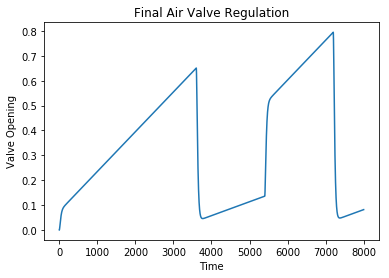
As well the FMU\_File as the .py-File should be filed in the same directory. You can always control the installed packages and their versions at the index tab “Uninstall packages”.

The next thing you see is the Python code for a successful export.



6: Python implementation

After running the code you should get results plotted in a graph like the following one.



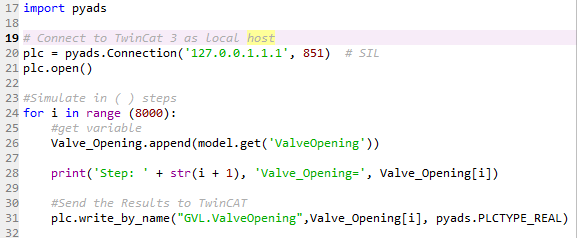
7: Python Graph from our Dymola Model

How to connect Winpython to TwinCAT 3 ?

For our example you have to create a PLC-Project with a local target system and the port 851. Now you can declare a global variable with the name “ValveOpening” and the data type REAL in the GVL (Global Variable List). Next you go to PRG Main and write GVL.ValveOpening in the code. TwinCAT 3 is ready for the connection with WinPython now.

To get a connection between Python and TwinCAT 3 you have to install the package “pyads”, which is a Python wrapper for TwinCAT ADS. For installation and documentation you can read this website: <https://pyads.readthedocs.io/en/latest/>.

TwinCAT will ask you to create port 851. This is the default port of TwinCAT PLC. Click yes and proceed. Always run the plc after login, otherwise the connection wouldn’t work. It is better to make a new JSON file to load PLC addresses when the default PLC addresses are not used.



8:Python-Code for a Connection to TwinCAT 3