# Progress Natlink project, splitting off python code?

With the current state of the Natlink repository, we are pretty close to a stable release. However, there are reasons to consider splitting off the python code from the C++/natlink.pyd code.

After considering making another repository “natlinkcore” for this (the python modules), I come to ideas, that can probably be fulfilled in the current repository:

* The compiler installer step (with Visual Studio Code) as it is, but without all the python files
* A pip install procedure for “natlinkcore” of all the python stuff, like “loader.py”, … and also the config program (CLI or GUI)

## Keep Natlink small

This is, for me, Quintijn, a minor issue, but for others this was a reason to bring this up. It seems useful to maintain not extend the natlink repository more than necessary.

## Only need a new release of Natlink when things change in the C++ code or the installer procedure.

This can be established, I believe, without the python stuff. Only inno setup and C++ code, leading to the installer file, which installs into “C:\Program Files (x86)\Natlink\site-packages\natlink”

For the python stuff, the “make flit install” trick of Doug can be used, in the same repository. I hope.

As last step of the installer, the “natlinkcore” project is installed/updated via a call to pip. This step should also lead to starting the natlinkconfig CLI or (hopefully in future again, Aaron, GUI).

When choosing a “package”, like vocola, unimacro, dragonfly, caster, …, these packages are also automatically installed via pip.

## Updating procedure of python packages

With the ideas of (automatic) updating of packages when Dragon/Natlink starts, as Aaron (written by Synkarius) has for Caster, this could be a major improvement.

Upgrading the different packages is then a responsibility of each package itself (except for “natlinkcore”, which is tightly connected with “natlink”, the .pyd module). (And as I formulate by now, in the same “natlink” repository.)

Concern (Aaron) is about releasing packages via pip: when upgrades are done automatically at startup of Dragon, much care has to be taken to not break things with a new release. Always, but even more when end users do not suspect changes and are confronted with them.

Maybe the testpypi environment should be adopted (again) for development intermediate releases.

## Documentation

With the current state of “natlink”, both the functions of the natlink.pyd (“getClipboard”, “setMicOn” etc) as the central python modules are imported via the site-package “natlink”, which is in C:\Program Files (x86)\Natlink\site-packages\natlink.

So also the “loader.py” module, and for end users important “natlinkstatus.py” are imported via “natlink”.

The drawback is that the automatic docstring documentation of the python files (“loader.py”, etc) is not working. According to Dane is a fix for this not very easily done.

Splitting off all the python stuff into a pipped package “natlinkcore”, can lead to “natlink.readthedocs.io”, in which the docstring documentation is working.

## Consequenses for programmers

Traditionally, “natlink” was on the python path, so “import natlink”, “from natlink import \*” or “from natlink import setMicState” did work.

With the current implementation, natlink.pyd (even a different file name) and python modules at one place, all import “qualified” via “natlink”:

from natlink import setMicState

from natlink import natlinkstatus

When splitting off the python code to a pip package, the former line remains the same, the latter becomes:

from natlinkcore import natlinkstatus

Programmers of end user grammars, as the maintainers of Dragonfly, Unimacro, etc will have to adjust a few things.

Note: upgrading from python2 to python3 involves quite a bit of work anyway.

## Conclusion

? I (Quintijn) might be in favour of keeping the stuff in one repository, when it is possible to make both:

* Inno setup installer
* Make flit install for the python code, into “natlinkcore”

Into this one repository.

Only when there are changes in the C++ code or inno setup procedure, we need a new inno installer step.

The “flit install –symlink” keeps the same possibility for developing python modules, as is now for other python packages (as Doug wrote for Unimacro, Vocola and Dtactions).