artkeller@gmx.de
2018-02-01

Important criteria for selection by #include <INCLUDE.h>

Multiple libraries with identical names in the Arduino IDE

What is the strategy for resolving the naming competition of several standard libraries of the same name?

Example:

```
#include <SD>
```

Arduino libraries contain in the file "library. properties" information about the architecture (s) supported by this library.

Example:

architectures=**esp32**

or

architectures=*

This information is apparently not the only criteria responsible for selecting the correct library if names occur more than once.

The specification

```
#include <SD>
```

causes the search for "SD. h" and different SD libraries can be found in different architectures.

But why are the information from "library. properties" not used?

Because Arduino first looks for the name in the **hardware** library path of the associated architecture.

Example:

With the "avr"-architecture in the path

"hardware\arduino**avr**\libraries\SD"

With the esp32 architecture in the path

"hardware\espressive**esp32**\libraries\SD"

Thus, in the case of competition from libraries of the same name, the order is resolved.

Finds Arduino at the compilation of

```
#include <SD>
```

now multiple header files "SD. h", the one that occurs in the pathname of the architecture is selected.

If this library is missing, however, the next step is to search for "default directory" of the avr architecture in Arduino's default directory and after that look for it in the general "Arduino\libraries" using the above-mentioned architecture filters.

This means that in the case of "SD. h", the corresponding avr library is used if such a one is missing in the actual target architecture, e. g. in "esp32".

Therefore, according to convention, all so-called standard libraries must exist in all architectures in order to avoid side effects and linker errors that are difficult to understand. Normally, standard libraries are adapted to the respective target architecture.

However, missing of a standard library in an architecture can happen deliberately in the Arduino world. As a result, there is an effect from the optimization of the avr world, as most architectures there are compatible and access to the "default directory" is almost useful.

After opening the Arduino IDE to other architectures, this side-effect must be avoided.

Error in naming new architecture hardware path

Popular problems are the use of wrong "SD.h" and "WiFi.h" in the ESP architectures, because the architecture names are not correct defined in the hardware paths.

Example:

Wrong

"hardware\espressif**arduino-esp32**"

Correct

"hardware\espressif**esp32**"