Arduino CLI Plugins

Adding plugins to duino_cli

by Dave Hylands
dhylands@gmail.com
https://github.com/dhylands/duino
https://github.com/dhylands/duino_plugin_example
https://github.com/dhylands/DuinoEsp32PluginExample

Arduino CLI Plugins

This duino_cli program makes use of the entry_point metadata that setup.py creates.

entry_points can be used to create a console script by adding something

like the following tp your setup.py

```
    entry_points={

            'console_scripts': ['cli=duino_cli.duino_cli:main'],

    },
```

This causees a script called `cli` to be created when you `pip intall` your package. The `cli` script will call the function `main` from the `duino_cli.duino_cli` module. Under linux, the script will typically look like the following:

```
    #!/home/dhylands/Arduino/.direnv/python-3.9.21/bin/python3.9
    # -*- coding: utf-8 -*-
    import re
    import sys
    from duino_cli.duino_cli import main
    if __name__ == '__main__':
        sys.argv[0] = re.sub(r'(-script\.pyw|\.exe)?$', '', sys.argv[0])
        sys.exit(main())
```

The line `from duino_cli.duino_cli import main` is generated using the portion after `cli=`. The name of the file will be the name before the `=`. The second to the last line sets up `sys.argv` to be the arguments passed into the `cli` script and the last line calls the `main` function (or whatever fucntion you specified after the colon).

You can also export additional meta-data which can be queried fromwithin your python program. You can use `importlib.metadata.entry_points()` to get a dictionary of all entry_points currently installed in your python installation.

The keys from this dictionary come from the `setup.py` script, so the example console script above would have added an entry to the `console_scripts` key. If you were to do:

`importlib.metadata.entry_points()['console_scripts']` this would return a tuple, and one of the entries of that tuple would look something like this:

```
EntryPoint(name='cli', value='duino_cli.duino_cli:main', group='console_scripts')
```

You can add your own metadata, which is how the plugin mechanism is implemented. For the duino_cli program, it looks for metadata with the key `duino_cli.plugin`. Each entry in the tuple corresponds to a plugin entry point. The value before the `=` is the name of the plugin, the value after the `=` is considered to be the name of a class derived from the `CliPluginBase` class.

For example, the following entry:

```
entry_points={

        'duino_cli.plugin': ['my_name=my_module.filename_base:ClassName']

},```
```

would look for a class named `ClassName` from the filename_base.py file found in the module named `my_module`

Do a walkthru of https://github.com/dhylands/duino_plugin_example

Also look at: https://github.com/dhylands/DuinoEsp32PluginExample

Do a demo