Welcome to the ArduinoML VS Code Extension

It's using langium and is another example based on Sebastien Mosser arduinoml zoo, used for education
Please see here for more information: https://github.com/mosser/ArduinoML-kernel

What's in the folder

This folder contains all necessary files for your language extension.

- `package.json` - the manifest file in which you declare your language support.
- `language-configuration.json` - the language configuration used in the VS Code editor, defining the tokens that are used for comments and brackets.
- `src/extension.ts` - the main code of the extension, which is responsible for launching a language server and client.
- `src/language-server/arduino-ml.langium` - the grammar definition of your language.
- `src/language-server/main.ts` - the entry point of the language server process.
- `src/language-server/arduino-ml-module.ts` - the dependency injection module of your language implementation. Use this to register overridden and added services.
- `src/language-server/arduino-ml-validator.ts` - an example validator. You should change it to reflect the semantics of your language.
- `src/cli/index.ts` - the entry point of the command line interface (CLI) of your language.
- `src/cli/generator.ts` - the code generator used by the CLI to write output files from DSL documents.
- `src/cli/cli-util.ts` - utility code for the CLI.

Get up and running straight away

- Run `npm run langium:generate` to generate TypeScript code from the grammar definition.
- Run `npm run build` to compile all TypeScript code.
- Press `F5` to open a new window with your extension loaded.
- Create a new file with a file name suffix matching your language.
- Verify that syntax highlighting, validation, completion etc. are working as expected.
- Run `./bin/cli` to see options for the CLI; `./bin/cli generate <file>` generates code for a given DSL file.

Make changes

- Run `npm run watch` to have the TypeScript compiler run automatically after every change of the source files.
- Run `npm run langium:watch` to have the Langium generator run automatically afer every change of the grammar declaration.
- You can relaunch the extension from the debug toolbar after making changes to the files listed above.
- You can also reload (`Ctrl+R` or `Cmd+R` on Mac) the VS Code window with your extension to load your changes.

Install your extension

- To start using your extension with VS Code, copy it into the `<user home>/.vscode/extensions` folder and restart Code.
- To share your extension with the world, read the VS Code documentation about publishing an extension.

To Go Further

Documentation about the Langium framework is available at https://langium.org