Google Season of Docs 2023

MicroPython Case Study

Organization: MicroPython

Organization Description: MicroPython provides a way to run Python 3.x code on a microcontroller or small embedded system

Organization Docs project title: Implement stubfile-based API documentation for MicroPython

Authors: Damien George, Matt Trentini, Jim Mussared

Problem Statement

This project attempted to address several different problems with the MicroPython documentation, but the two primary areas were:

- To make it easier to write, maintain, and ensure consistency and accuracy of developer-focused API documentation. As MicroPython is a developer tool, this API documentation is the main area of our documentation for most users. A particular goal was also to make this documentation easier to access through other methods (e.g. IDE auto-completion) via stub-files.
- To improve the overall hierarchy and structure of the documentation to reduce duplication, make it easier to find information. The documentation has grown organically over the past decade, and it can be difficult to find the right information.

Proposal Abstract

1. Adapt two techniques used by the PyBricks project (PyBricks is a project to integrate MicroPython into Lego products, e.g. MindStorms, and they have excellent documentation tailored to their specific needs):
    - Inline annotations in API documentation to indicate supported platforms.
    - Move existing rST API docs to inline API docs in stub files, loaded automatically into Sphinx. This is also significantly easier to write and maintain than the existing rST-based approach.
2. Audit the existing platform-specific documentation and extract common tutorials and guides into new general-purpose guides that apply to all ports that support the relevant feature. e.g. all boards that support WiFi should be able to share the same “connecting to an access point” guide.
    - In general, move as much platform-specific documentation into the main documentation, and de-duplicate this information in the process
3. Capture comments from the core team that have been made in various community forums (GitHub Discussions, Issues, PRs, Discord) into proto-guides. There is a

tremendous amount of very detailed explanation of technical details available in such forums. It is better to have partial guides (clearly marked as such) than to have no guide at all, and this also provides a good starting point for further contributions. Some areas in particular:
    - Using MicroPython in real products (i.e. additional steps and information above what a hobby project would need)
    - Using MicroPython with existing IoT infrastructure
    - Working with packages and package management
    - Customising MicroPython for project-specific requirements
4. Restructure the overall hierarchy of the documentation:
    - Separate tutorial- and guide-like content from the existing API documentation, and combine it with de-duplicated platform-specific content into a new “Guides and Tutorials” section, with a focus on specific, common tasks and workflows. This requires mostly reorganising and updating existing content to make it more discoverable and useful, rather than writing new content from scratch.
    - Re-organise the existing “Internals” and “Language and implementation” sections into a hierarchy of topic- and audience-specific guides. There is a lot of existing content that can be reused and recycled here, but new content needs to be written too. One specific target of this goal is to establish a common template and style for these guides, making it easier for other contributors to extend and add content.
5. Add documentation about how to effectively contribute to the documentation. This exists for code contributions, but a documentation-specific contributors guide would be helpful, especially for new contributors.

Project Description

Creating the proposal

The MicroPython project has participated in Google Season of Docs twice before (2020, 2021), so the process was familiar.

The five areas of work proposed are based on ideas that are well-known to the project and have been on our TODO-list for a long time. The stub-file area was prioritised because improving IDE auto-completion is a frequently-requested feature, and also because the difficulty of authoring Sphinx/reStructuredText documentation is a barrier to new contributors.

Budget

The budget was largely decided by considering the total time available for the project, the time that could be committed by the tech writer, and a reasonable hourly rate. It was anticipated that the tech writer would spend between 120-160 hours on this project in total, in a flexible schedule between April and November. We deliberately prepared a priority order of work to be done with the expectation that this would not all be completed in the time available.

The project also receives funding through GitHub Sponsors, and this will be helpful to finance continued work on the documentation. In particular, any remaining incomplete tasks at the end of the Google Season of Docs 2023 timeframe will hopefully be able to be funded by available GitHub Sponsors funds. This source of income may also be able to subsidise ongoing improvements to the documentation in the future.

Participants

The mentors for this season were Damien George (MicroPython creator and core developer) and Matt Trentini (MicroPython community member and contributor), who have both been mentors in the previous two years of Google Season of Docs that MicroPython was part of.

The tech writer for this year was Jim Mussared. He was already a regular contributor to the project (code, issue triage, forums) and also made occasional documentation contributions. Having prior experience with and knowledge about the project made him a very good candidate for the tech writer. Working with an existing contributor simplified the experience considerably because there were no on-boarding or process overhead issues, and the time commitment required by the project mentors was minimal.

Timeline

April-May 2023
Groundwork for making Sphinx consume stub files as part of the documentation generation process. This involved prototyping and working on Sphinx extensions.

May-June 2023
Importing generated stub files using the micropython-stubber project. Start adding annotations (e.g. supported ports, version-added, etc), as well as type annotations, and formatting of function arguments and return types.

June-July 2023
Start improving the content/wording of API documentation in the stub files. Audit structure of existing docs. Investigate using a tab component to isolate port-specific differences (generated via type annotation in the stub files)

July-September 2023
Replace the "MicroPython internals" and "MicroPython language & implementation" sections with a more structured "Guides" section, start adding new guides and updating existing content to fit this structure. Rename "MicroPython libraries" to "Library & API reference" and update the top-level index to provide a more straightforward introduction.

September-October 2023
No progress, mostly due to work on MicroPython v1.21 release.

October-November 2023
Add top-level FAQ with useful entry points to the guides. Table-of-contents improvements so that top-level classes are listed in the TOC making it easier to navigate to these

frequently-used classes. Improve the glossary. Ongoing work on stub files, API documentation, annotations, guides.

Results

Of the five main areas of work described in the proposal, there was significant progress made on #1 (stub files), #2 (audit) and #4 (restructure). No work was done on #3 (capturing comments from forums), or #5 (contributors), however relating to #5, an extensive guide was written to assist contributors (not just documentation writers) for effectively working with Git and GitHub in the context of the MicroPython project.

The stub file work ran into some early technical limitations with the Sphinx documentation generator that MicroPython uses. We had misunderstood some details of the approach used by PyBricks, but our tech writer was able to find some workarounds by implementing Sphinx extensions (which we have also shared with Pybricks). These are completely reasonable limitations and not a criticism of Sphinx, rather they are quite specific to the task of documenting MicroPython because we are essentially re-documenting Python language and Python standard library features. This confuses Sphinx as it wants to use the Python version of these APIs instead. Other extensions were to improve Sphinx's ability to handle overloaded functions as well as a mechanism to extract annotations from the stub files.

Of the 43 modules in the API documentation, all have been converted to stub files and are at varying stages of updates (type annotations, port/version annotations, rewording and extending content, adding missing methods). Five modules have been fully completed.

The audit of the existing documentation structure identified a way to separate API documentation from the guide/tutorial-style documentation in a way that doesn't require so much port-specific content. The subsequent restructure work is still ongoing, but significant progress has been made. This turned out to require a lot of effort to de-duplicate similar content, as well as identifying subtle improvements to API documentation to highlight port-specific differences. Having a new way to annotate the API documentation in the stub files made it much easier to capture these differences.

Also, by doing this audit across the documentation we were able to identify areas that were lacking and add placeholders to indicate where new contributions could be made. Additionally, a number of consistency and style issues were identified and refactorings were done to improve usage of Sphinx/rST features.

Metrics

Some long-term metrics are described in the proposal, however as this work is still ongoing it will be some time before we can make any meaningful evaluations. We will continue to monitor the documentation PRs (total docs PRs, distinct committers, % by non-core-team) and forum questions that can be answered with a simple link to the docs, as described in the proposal. These metrics can be reported in the follow-up surveys in 2024.

Analysis

Although the new documentation is not yet "live", we are very pleased with the progress made in this project and are looking forward to the final result.

By working with an existing contributor, the Google Season of Docs work was able to be very efficient, with little overhead imposed on the mentors. On the other hand, because Jim was very involved with other aspects of the project (for example a number of features and testing for the v1.21 release, as well as other unrelated documentation tasks), there were significant other demands on his time, making it difficult to make as much progress on the documentation work as we would have desired. However, as an active and ongoing contributor, Jim will continue to work on the documentation tasks, and we plan to allocate some of our GitHub Sponsorship funding for this.

Summary

MicroPython was once again very grateful for the opportunity to participate in Google Season of Docs. This year's project has kick-started a much-needed overhaul of our documentation that likely would not have happened without the Season of Docs project and associated funding. Jim had very strong background knowledge of the project, the ability to formulate precise wording to document the subtleties of the MicroPython APIs, and a good plan to tackle the proposed documentation work. We are very happy with the progress that he made and look forward to him continuing and completing the work.

Advice for other projects participating in Google Season of Docs:

- Season of Docs allowed us to support an existing contributor to this project. This significantly improved the efficiency of mentoring and documentation, but one should watch out for other/existing project commitments, that they don't hinder the progress on the documentation efforts. We would love to repeat this in the future with other contributors to the project.
- Having a range of different tasks allowed a bit more variety in the project. Manually converting existing documentation into stub files was tedious, so mixing it up with the other tasks was beneficial.
- The set of five documentation tasks were ordered by priority, and we made it clear that the goal was just to make as much progress as possible, not to complete *all* the tasks. The tasks were chosen so that any progress towards them was valuable. This allowed for more flexibility (on both the mentor and tech writer sides) and more chance of success.
- GitHub Sponsorship funds allow us to have some budget to finish off the parts of the documentation changes that were started but not completed as part of Google Season of Docs. This allows us to maximise the benefit we get from the tech writer.

Acknowledgements

We are extremely grateful to Jos Verlinde and his micropython-stubber project. This project includes a tool that parses the existing MicroPython documentation and applies some heuristics to generate stub files. These stub files still required a considerable amount of

manual rewriting and enhancement as well as type and other annotations but it was a great starting point for building these files.