ANUBIS-IDE SRS

AUGUST 20, 2020

AIN SHAMS UNIVERSITY – FACULTY OF ENGINEERING
1 Al-Sarayyat st, Abbassiya, Cairo 11517, Egypt

FACULTY OF ENGINEERING

AIN SHAMS UNIVERSITY

CSE426: Maintenance and Evolution

ANUBIS-IDE SRS

Submitted By:

Abdelrahman Ibrahim ELGhamry

16P3043@eng.asu.edu.eg
Ghamry1998@gmail.com

Submitted to:

Prof. Dr. Ayman Bahaa

AUGUST 20, 2020

A REPORT FOR SOFTWARE MAINTENANCE AND EVOLUTION COURSE CODDED CSE426 WITH THE REQUIREMENTS OF AIN SHAMS UNIVERSITY

TABLE OF CONTENTS

1. PURPOSE

The aim of this document is to build a desktop-based open-source Integrated development environment (IDE) which enables its users to write, edit, compile, and run python codes on micro-controllers.

It will also explain the aim and the key features of the system, the various scenarios of the system, and the constraints under which the system must operate.

2. BRIEF DESCRIPTION

Anubis-IDE is an open source desktop-based text editor, which aims to provide a simple integrated development environment to write, edit, compile, and run python scripts on various micro-controllers.

This new product will assist a huge community of embedded-systems engineers to compile, build and run their python codes directly on micro-controllers in an efficient and manageable manner.

3. SYSTEM REQUIREMENTS

3.1 Functional Requirements

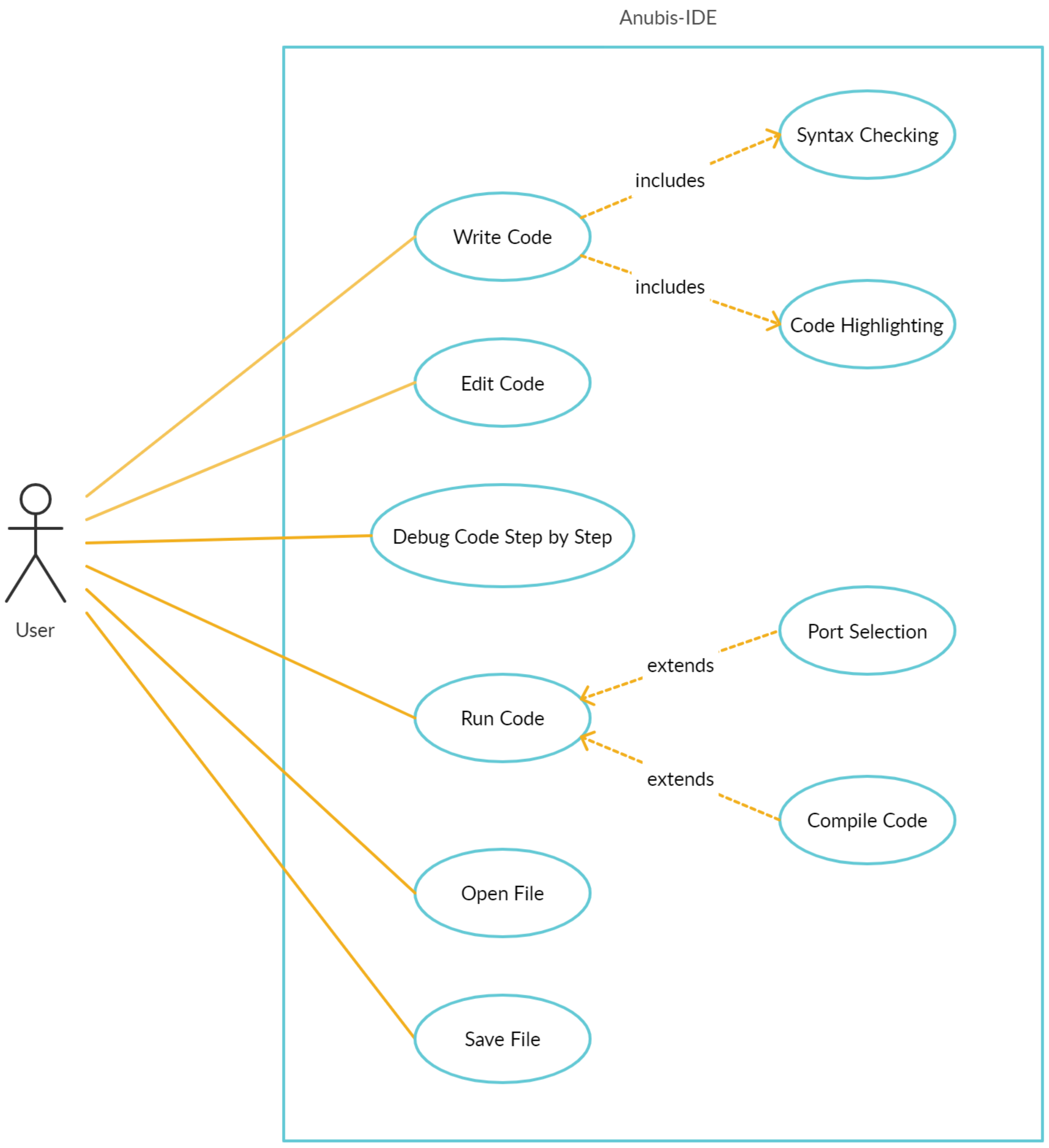
1. The software should support opening and editing any text file.
2. The user can write micro-python codes to files.
3. The IDE must support code highlighting, syntax checking and auto-completion.
4. Debugging tools must be provided to the user.
5. The IDE must provide a list of all available ports on the running host in order to select one of them.
6. The user must select the attached micro-controller port before compiling and running the code.
7. The IDE must compile, flush, and run the code on the selected micro-controller.

8. The IDE has a panel to display the class hierarchy view.
9. The IDE has a panel to display the project structure as folders and files in the current directory.
10. The IDE has a code editor panel that supports code highlighting for all reserved words, numbers, comments and variables in the currently supported programming language.
11. The IDE should save files in the currently opened directory.

3.2 Non-Functional Requirements

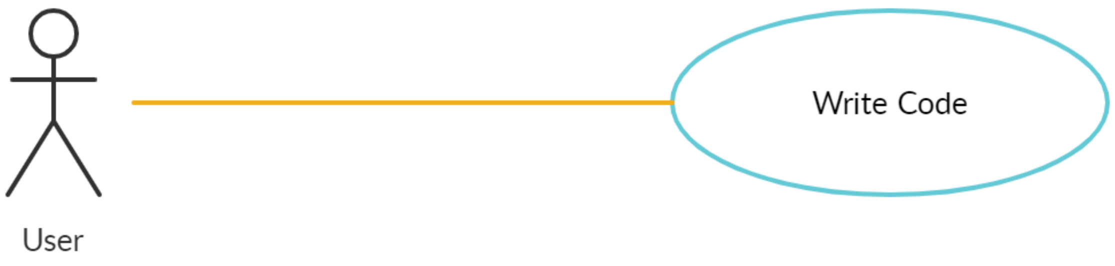
1. The software must be written using python.
2. The software must be compatible with various operating systems: Windows, macOS and Linux.
3. Syntax errors detection and feedback must happen within one second.
4. The software running requirements must not exceed 4GB of RAM.
5. The software must use Git for version controlling and the repo must be public on GitHub.
6. The software must follow agile process model and deliver increments within 3 weeks cycles.
7. The software must be delivered within 9 months on 3 main releases where 3 months is considered for each one.
8. The software must declare dependencies and instructions to install them.

4. USE CASE DIAGRAM



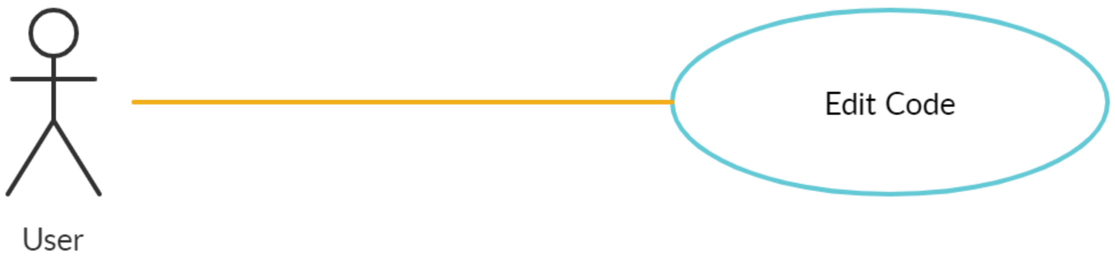
5. NARRATIVE DESCRIPTION

1. Write Code



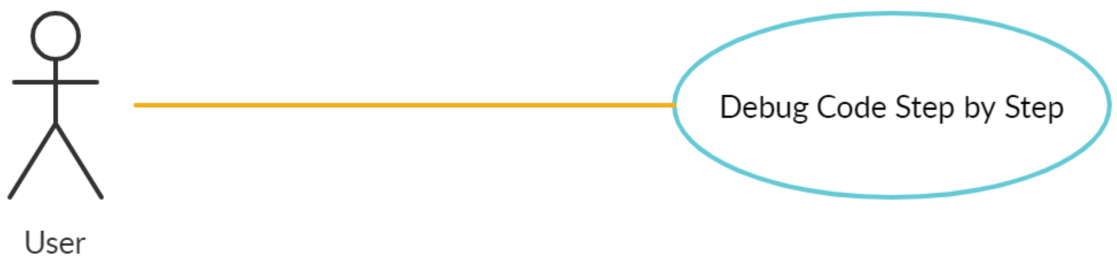
Description	The user should be able to write python code in the text-editing panel.
Primary Actor	The user.
Main Flow	1. User opens the IDE 2. User starts typing in the text-editing panel.

2. Edit Code



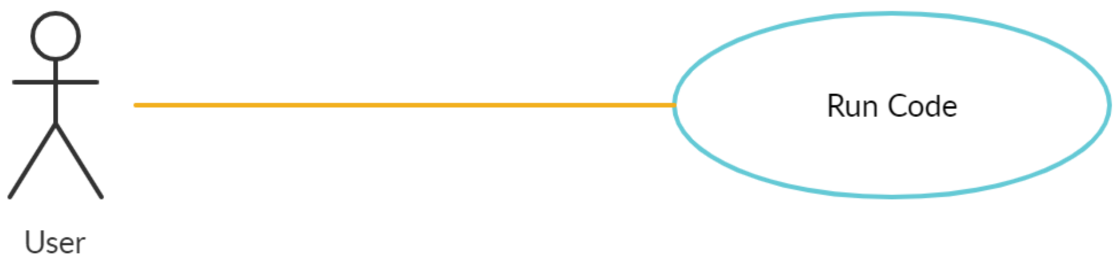
Description	The user should be able to edit an existing python code in the text-editing panel.
Primary Actor	The user.
Main Flow	1. User selects a python file from the tree view. 2. Code opens in the text-editing panel. 3. User starts editing code.

3. **Debug Code Step by Step**



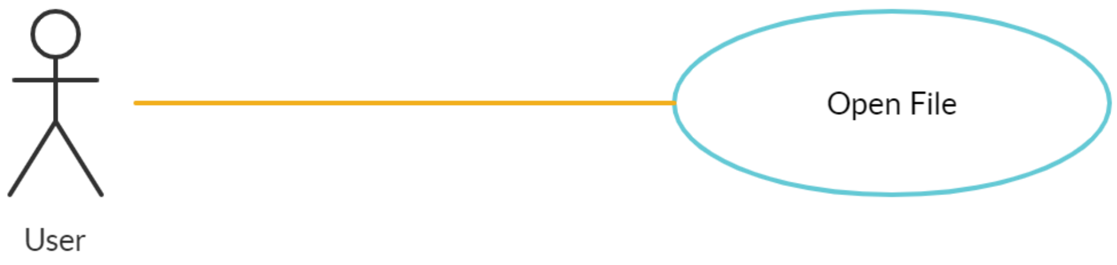
Description	The user should be able to use break points to debug his python code.
Primary Actor	The user.
Main Flow	1. User selects a line of code and adds a break point to it. 2. User specify the running port. 3. User runs the code. 4. A new panel that shows used variables values should appear to the user.

4. **Run Code**



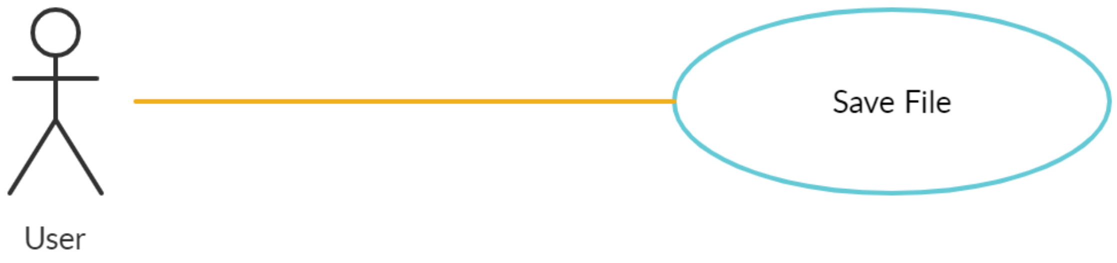
Description	The user should be able to run his python code on the micro-controller.
Primary Actor	The user.
Main Flow	1. User specify the running port. 2. User chooses to run the code. 3. A feedback message should appear to the user whether it's a successful run or not.

5. Open File



Description	The user should be able to open an existing file using the tree view panel or the file menu.
Primary Actor	The user.
Main Flow	1. User opens the file menu. 2. User selects the “Open” option. 3. User selects the desired file to open using the file explorer.

6. Save File



Description	The user should be able to save his current progress.
Primary Actor	The user.
Main Flow	1. User opens the file menu. 2. User selects the “Save” option.