Module Guide for MECHTRON 4TB6

Team 25, Formulate
Ahmed Nazir, nazira1
Stephen Oh, ohs9
Muhanad Sada, sadam
Tioluwalayomi Babayeju, babayejt

April 5, 2023

1 Revision History

Date	Version	Notes
2023/01/18	1.0	Final Version

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	description
AC	Anticipated Change
M	Module
MG	Module Guide
MIS	Module Interface Specifications
OS	Operating System
FR	Functional Requirement
SRS	Software Requirements Specification
UC	Unlikely Change

Contents

1 Revision History **i**

2 Reference Material **ii**

2.1 Abbreviations and Acronyms ii

3 Introduction **1**

4 Anticipated and Unlikely Changes **1**

4.1 Anticipated Changes 1

4.2 Unlikely Changes 2

5 Module Hierarchy **2**

6 Module Decomposition **3**

6.1 Hardware Hiding Modules 3

6.1.1 mainArduino.ino (M1) 3

6.1.2 mainESP8266.ino (M2) 3

6.2 Behaviour-Hiding Module 3

6.2.1 ui_main.py (M3) 3

6.2.2 ui_functions.py (M4) 4

6.2.3 main.py (M5) 4

6.3 Software Decision Module 4

6.3.1 resource_rc.py (M6) 4

7 Traceability Matrix **5**

List of Tables

1 Module Hierarchy 3

2 Trace Between Requirements and Modules 5

3 Trace Between Anticipated Changes and Modules 5

3 Introduction

The following document explains the Module Guide for our Capstone project. Complementary documents include the System Requirement Specifications, (SRS), and the Module Interface Specifications, (MIS).

The rest of the document is organized as follows. Section 4 lists the anticipated and unlikely changes of the software requirements. Section 5 summarizes the module decomposition that was constructed according to the likely changes. Section 6 gives a detailed description of the modules. Section 7 includes two traceability matrices. One checks the completeness of the design against the requirements provided in the SRS. The other shows the relation between anticipated changes and the modules.

4 Anticipated and Unlikely Changes

This section lists possible changes to the system. According to the likeliness of the change, the possible changes are classified into two categories. Anticipated changes are listed in Section 4.1, and unlikely changes are listed in Section 4.2.

4.1 Anticipated Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision. The approach adapted here is called design for change.

AC1: The color scheme of the GUI which means a change in the stylesheet.

AC2: The layout of the pages and buttons in the GUI which means a change in the ui files and animation modules.

AC3: Information that is displayed on each GUI page.

AC4: The method of storing information in the database to accommodate for large amounts of data.

AC5: The fields required for creating an account.

AC6: The data structures for storing and manipulating data inside the software.

AC7: The method of decomposing front and back-end functionality of GUI.

4.2 Unlikely Changes

The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. Hence, it is not intended that these decisions will be changed.

UC1: Input device will be a keyboard and output device will be a screen.

UC2: The Qt library for creating the GUI

UC3: The hardware the GUI is running on

UC4: The programming language (python) for the GUI

UC5: Storing test and user information in a database

UC6: Hosting the database on an Azure SQL Server

UC7: Visually displaying stored test information in Power BI

5 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 1. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually be implemented.

M1: mainArduino.ino

M2: mainESP8266.ino

M3: ui_main.py

M4: ui_functions.py

M5: main.py

M6: resource_rc.py

Level 1	Level 2
Hardware-Hiding Module	mainArduino.ino mainESP8266.ino
Behaviour-Hiding Module	ui_main.py ui_functions.py main.py
Software Decision Module	resource_rc.py

Table 1: Module Hierarchy

6 Module Decomposition

6.1 Hardware Hiding Modules

6.1.1 mainArduino.ino (M1)

Services: Converts the input data taken from sensors into the data that is converted to data used by the UI. This function also provides the main program logic of the microcontroller of the board. It is used to initalize the microcontroller's pins, timers and other hardware components which all help in gathering data for our database.

Implemented By: [Arduino]

6.1.2 mainESP8266.ino (M2)

Services: An Arduino module establishes a Wi-Fi connection between the Arduino and PC to send data from the sensors and the Arduino to the PC. This allows us to collect and send data wirelessly through user application to our database wirelessly.

Implemented By: [Arduino, ESP8266]

6.2 Behaviour-Hiding Module

6.2.1 ui_main.py (M3)

Services: A PyQt designer generated module that sets up the application's window and design. This module has all functions and layout information for our user application. It also provides the code for the user interface components that make up the application.x''

Implemented By: [PyQt]

6.2.2 ui_functions.py (M4)

Services: A python module that imports all the necessary libraries for backend, contains class for UI functions and creates the connection to the database. This module also creates the functions that the user interacts while navigating through our user interface. It defines the function that the handle user inputs, such as button movements and mouse clicks

Implemented By: [PyQt]

6.2.3 main.py (M5)

Services: A python module that imports backend functions and frontend setup of GUI. This is also used to start and run the desktop application and can be considered the interpreter that reads and executes the code.

Implemented By: [PyQt]

6.3 Software Decision Module

6.3.1 resource_rc.py (M6)

Services: A python module generated by PyQt resource compiler and uses all the local resources to display images during runtime. It is used to embed binary resources, such as images, icons, or other data files, into PyQt.

Implemented By: [PyQt]

7 Traceability Matrix

This section shows two traceability matrices: between the modules and the functional requirements from our SRS document and between the modules and the anticipated changes.

Req.	Modules
FR1	M1
FR2	M1
FR3	M1, M2
FR4	M1, M2
FR5	M1
FR6	M3, M4
FR7	M4, M5
FR8	M4
FR12	M1, M2, M4
FR14	M1, M2, M3, M4, M5
FR18	M4, M5

Table 2: Trace Between Requirements and Modules

AC	Modules
AC1	M3
AC2	M3
AC3	M5, M4
AC4	M4
AC5	M3, M4
AC6	M4
AC7	M4, M5

Table 3: Trace Between Anticipated Changes and Modules