Assignment 1 - Specification

MTRN3500 - UNSW School of Mechanical and Manufacturing Engineering

Alexander Cunio & Jay Katupitiya

September 4, 2024

Changelog

- 09/09/24: Initial release

1 Assignment Aims

This assignment has been designed to introduce you to how you might interact with existing on-the-market hardware. It is conducted through an experimental hardware setup we will call "the PLC System".

By the end of the assignment you will have:

- Interacted with the hardware by writing a high-level interface.
- Found the required methods of communicating by reviewing hardware documentation provided by a manufacturer.

2 Assignment Problem

This assignment requires you to develop an object oriented software package for a Programmable Logic Controller (PLC - and hence the PLC System) which in this case is used as a general purpose interface device. PLCs like this are commonly used to control various robotic systems in the industry as they have more functionalities such as process control loops (PID loops). They are equipped with various digital inputs and outputs (for controlling ON/OFF systems, sensing ON/OFF signals and for communicating with encoders) as well as analogue I/O (for dealing with amplifiers, actuators or other analogue equipment).

Twenty four of these units have been setup, along with peripheral equipment (encoders, volt meters, LEDs, motors etc.) in the Mechatronics labs at UNSW. You are required to write the necessary software for controlling the PLC. A header file `Galil.h` has been provided for you. It declares all the functions which you are required to implement. You must create a C++ file called `Galil.cpp` (NOTE: The file must have this name EXACTLY) and it must contain all the function definitions you will write. These are defined in `Galil.h`. You are allowed to add your own **private** member functions or data, but you must not change the existing **public** member functions and data. Having completed `Galil.h` and `Galil.cpp` you must be prepared to demonstrate the functionality of your object class by calling individual functions where appropriate in a `main` function.

For testing outside of lab sessions, a simulator has been provided. This emulates the functionality of the physical devices as a means of continuing development. Please note that the in-person assessments will run on the physical hardware within the lab so ensure you test on these during your lab sessions.

You will be coding the assignment in Visual Studio 2022 (a C++ IDE produced by Microsoft). Most of the development can be conducted on your own computer, so you will need the software installed. Running the assignment requires connection to the PLC manufactured by Galil. This can be done in one of two ways, either the physical in-lab hardware or the provided simulator. Further information on running the assignment can be found in the provided 'Galil Use' document (found on Moodle).

3 Your task

3.1 Part A: Interacting with the Galil PLC

Your task is to implement a set of functions that define a variety of means of communicating with the Galil PLC. Communication is conducted through different commands that are provided by the manufacturer in the documentation (`Galil-RIO47142-CommandReference` on moodle). You must implement the class member functions defined within the `Galil.h` (in the `Galil` class). The functionality of each of these is described within the header file itself. These functions each define a way to interact with the hardware, including the digital and analogue inputs and outputs and querying the PLC.

Each function that must be completed is marked with a '`TODO`' indicator within the header file. This implementation should be written in a file called `Galil.cpp`. You must thoroughly test your implementation not only on the simulator provided but also on the physical hardware within the lab classes (you are provided plenty of time within each lab class to do so).

Whenever you send a command to the Galil PLC (whether physical or in the simulator) it must be sent through the provided class `EmbeddedFunctions`. This has a number of functions defined within `EmbeddedFunctions.h` that you should review. This provides a wrapper around Galil-provided implementations and a simulator interface.

3.2 Part B: Direct Ethernet communication with hardware

Throughout part A you have interracted with the EmbeddedFunctions wrapper class in order to communicate with the Galil PLC. This functions to allow the computer to communicate through TCP/IP communication over Ethernet. This is a reliable means of sending data through a network connection which requires three main stages; connection (establish a new connection between the host [the Galil] and the client [the computer]), communicate (send and receive data) and closing (terminate the connection).

You must now implement your own modified version of this class using TCP communication principles taught in lectures. Your first task is to write the underlying code that permits this communication to take place so that useful commands can be sent to the PLC. Within a separate EmbeddedFunctions project, you have been provided with the header file `EmbeddedFunctions.h` (note this is different from the header file of the same name in the previous part of the assignment) that outlines a series of functions that you must implement (these will be implemented in the file `EmbededFunctions.cpp`).

- `void GOpen(String^ address, const int port)`: This establishes a new connection with the PLC by opening up a TCP socket to the prescribed address.
- `void GClose()`: Closes the existing connection to the PLC.
- `String^ GCommand(String^ command)`: This sends the query within `command` to the Galil and returns the response to the caller.

All these should be implemented using networking principles discussed within the lectures.

Your code should be written to account for any errors that may occur during execution.

4 Overview of the Supplied Files

You have been provided with a repository (accessible through GitHub classroom) that contains a Visual Studio solution ready for you to start coding. All dependencies have already been included and appropriately linked to the corresponding projects.

There are two projects within the solution which each correlate to one part of this assignment as described in Section 3.

The `Galil` project is for part A and contains the following important files:

- `Galil.h`: This header file declares all the necessary functions that you must write in your `Galil.cpp`. Read all the instructions carefully.
- `Galil.cpp`: This file is where you should implement all of the functions defined in `Galil.h`.
- `GalilMain.cpp`: This file should contain the main function for this project that you should use for testing your implementation.
- `EmbeddedFunctions.h`: This header file wraps the Galil commands in a class structure. Whenever you send a command to the board, send it through this class. Failure to do so will result in mark deduction.

- **Other Files**: `gclib.h`, `gclibo.h`, `gclib_errors.h`, `gclib_record.h`, `gclib*.lib`, `gclibo*.lib`, `gclib.dll`, `gclibo.dll`, `GalilControl*.lib`, and `Embedded_funcs*.lib`: These files contain all the dependencies required for the project. They include those provided by Galil for use of their library along with some developed for assignment components. They can be found throughout the provided repository and are already linked to your project ready to be used.

The `EmbeddedFunctions` project is for part B and contains the following important files:

- `EmbeddedFunctions.h`: This header file defines the functions you must implement yourself using Ethernet communication.
- `EmbeddedFunctions.cpp`: In this file you should write the implementation for all the functions defined within `EmbeddedFunctions.h`.
- `EFMain.cpp`: This file should contain the main function for this project that you should use for testing your implementation.

In addition to these, you also have been provided with the following that should be used while completing the assignment (they can be found on Moodle alongside this spec):

You can find the following in the Moodle site under Assignment 1.

- **Command Reference**: This provides a manual for all the commands that can be sent to the Galil PLC as provided by the manufacturer. For this assignment these commands must be sent through the EmbeddedFunctions interface which you will need to interact with throughout the assignment.
- **Galil Use Instructions**: This supplementary document provides additional details on running the assignment using both the physical hardware in the lab and the provided simulator used for testing at home.

It is expected that you review and read all provided files (including code files) as they form part of this assignment specification.

5 Marking Criteria

This assignment is worth 20 % of the total course mark. The breakdown of marks for this assignment is described in table 1.

Table 1: Mark allocation for assignment

Criteria	Weighting	Description
In-person assessment	40 % (8 marks)	As below in Section 5.1
Auto-marking	30 % (6 marks)	As below in Section 5.2
Style marking	30 % (6 marks)	As below in Section 5.3

5.1 In-person assessment

Your assessment will take place in Week 5, during your scheduled tutorial. It will constitute a short 15-minute assessment where you will be provided with a question that prescribes functionality that is desired from the PLC. Using your interface that you have already written (i.e. you Galil class implementation), you must call the functions from the public interface to make the PLC act as specified. Please see the submission section below for opportunities for bonus marks submitting early.

During this in-person assessment, you will also be tested on your implementation of the separate `EmbeddedFunctions` class (part B). You must have the class implemented as prescribed and the tutor will type in and attempt to send string commands to the Galil through the main function `EFMain.cpp`.

Warning: You must attend your designated weekly lab class to progress and get support with this assignment. **You will be assessed on the physical hardware in the lab classes**, so make sure you test your code here beforehand.

Information for the assessment:

- You will also be asked **two** theoretical questions related to the PLC and will be given 10 minutes to answer these questions after the completion of the first component. These will be submitted via an online form.
- It is really important to have all the class instantiation completed and be really prepared to call any of the member functions to complete the specified task. **Only one attempt is allowed.**
- You should have your implementation ready for this along with a `main()` function ready to use. Within this you can have written all objects instantiated ready for use but no other implementation is permitted.

- The functionality of this task will be assessed in-person by your demonstrator during your lab session directly after the 25 minutes is complete.
- **2 marks** will be awarded for the functionality of your program, **2 marks** for the questions asked, and **4 marks** for your `EmbeddedFunctions` implementation.

5.2 Auto-marking

Auto-marking will be carried out by running your submitted code (the `Galil` class) against a series of tests written to validate your implementation.

5.3 Style marking

Style marking will be broken down based on the following criteria.

- **Structure (2 marks)**:
  - Logical breakdown into separate files,
  - Well-reasoned selection of data and function members of the classes and non-member functions.
  - Good application of programming paradigms including DRY (don't repeat yourself) and KISS (keep it simple).
  - Usage of modern C++ principles where appropriate.
- **Layout (2 marks)**:
  - Order of programming statements,
  - Indentation,
  - Use of braces and parenthesis,
  - Consistent and well reasoned choice of constant/variable/function names that enhances readability.
- **Program constructs (2 marks)**:
  - Well reasoned choice of data types,
  - Proper choice of iterative loops,
  - Orderly use of other constructs such as `switch`, `break`, `continue`, etc
  - Achieving best program logic with least amount of coding.

6 Submission

The assignment is submitted in two components: in-person completion of prescribed task and file submission for style and auto-marking.

The first submission is conducted during your practical assessment in your lab class. This will be assessed in person by your demonstrator for the completion of the provided task. You must also submit the assignment files `Galil.h`, `Galil.cpp`, and `Main.cpp` to Moodle in the 'Assignment 1 - Practical Assessment' submission box directly after completion of your assessment time (you will be given five minutes for this).

Completion and submission of this component must be during your assigned week 5 lab time slot. Early submission for one bonus mark must be during your assigned week 4 lab time and early submission for two bonus marks must be during your assigned week 3 lab time.

The second submission will be for code style and auto-marking. Without zipping the files, submit your `EmbeddedFunctions.h`, `EmbeddedFunctions.cpp`, `Galil.h`, and `Galil.cpp` files to Moodle in the 'Assignment 1 - Code Submission' submission box. The submitted code will be first checked for similarity scores prior to the assessment outcomes.

Submission must be by 11.59 pm of Friday of week 5 **(11 October, 2024)**. However, early submission can be conducted to gain bonus marks. Submitting one week early (by 11.59 pm of Friday of week 4) will grant one bonus mark and submitting two weeks early (by 11.59 pm of Friday of week 3) will grant two bonus marks.

***Note**: if you choose to submit early, both components must be submitted early. Submitting **both** parts one week early will grant a total of one bonus mark. Similarly submitting **both** parts two weeks early will grant a total of two bonus marks.*

***Note**: bonus marks are added to the assignment, which is capped at 20 marks (you are not able to get more than 20 marks for the assignment).*

7 Plagiarism

If you are unclear about the definition of plagiarism, please refer to What is Plagiarism? — UNSW Current Students. You could get zero marks for the assignment if you were found:

- Knowingly providing your work to anyone and it was subsequently submitted (by anyone), or
- Copying or submitting any other persons' work, including code from previous students of this course (except general public open-source libraries/code). Please cite the source if you refer to open source code.

You will be notified and allowed to justify your case before such a penalty is applied.