Auckland University of Technology

School of Engineering, Computer and Mathematical Science

ENEL712 Embedded Systems Design

Project Report

Submitted by: Josiah Brough
Student ID: 22160417

18 May 2023

Table of Contents

1 Introduction

The project aims to develop a Graphical User Interface (GUI) using C# for controlling the functionalities of the AUT AT90USB Applications Board from a PC. The GUI on the PC will have multiple tabs to control different aspects of the board, communicating via USB Serial and saving time and temperature values to a local database server.

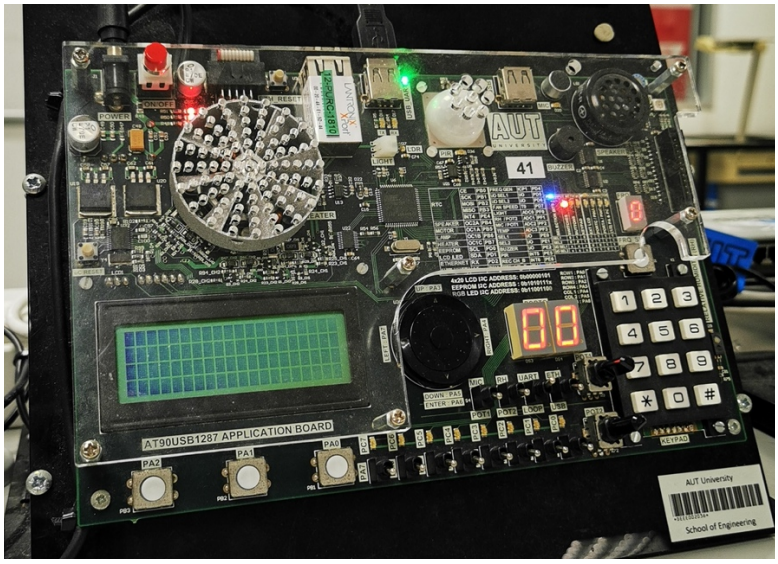

Figure 1: The AUT Lab Board

2 Objective

2.1 Microcontroller

The microcontroller of choice is the Atmel AT90USB1278 and is programmed in the C programming language. The program will require the ability to parse up to 5 bytes of data: a start byte, an instruction byte, two data bytes and a stop byte. When an instruction is received the microcontroller should perform a task and report back an acknowledgement or data. These tasks will be explained in the following sections.

2.2 Graphical User Interface (GUI)

The GUI will be a windows form application and will consist of four tabs each with their own purpose and functionality. Each of these tabs will also include open-source windows form components such as LEDs[2], gauges[3] and seven-segment[4] displays.

The setup tab will allow the user to select the USB-Serial COM Port and Baud Rate through two combo boxes. Upon clicking the Connect Button, the application should initiate the selected serial port, establish communication with the applications board, and activate the Serial Port Status LED, signifying successful communication. Similarly, clicking the Database Connect Button should establish a connection to the specified database using the provided log-in credentials. Once the connection is established, the Database Connection

Status LED will be activated, indicating a successful connection.

The digital I/O tab will enable the user to be able to see the current switches enabled on PINA of the lab board. Similarly, they will also be able to set the LEDs lights on PORTC of the lab board, with the value displayed on the seven segment displays in the GUI.

The Pot-lights tab will serve as a platform for the user to view the current voltage level of the two potentiometers on the lab board, updating them as they change. Additionally, the user will have the ability to adjust the lamp brightness directly from the GUI, with its brightness level being read from the LDR on the lab board and displayed on a gauge within the GUI.

The last tab is for the temperature control, this tab allows the user to read the current temperature and motor speed from a real-time graph. This implement a PI controller[5] allowing the user to adjust the K_p and K_i values. These values are graphed in real time and a button allows these values to be automatically stored in a locally hosted database server.

2.3 Database

Using the MySQL libraries within visual studio the user will be able to connect and log time and temperature data on the locally hosted server both manually and automatically. This will require the initialization of a new user, database and table whose credentials are submitted within the GUI.

3 Methodology

3.1 Microcontroller C program

The problem was broken down into smaller more achievable tasks, starting with the input-output devices and then working on the communication protocol, then finally linking the two together for a system that executes based off a given instruction. This approach is called the bottom-up approach where the low-level components are developed before the full system is designed.

Code from previous work was used to help initialize and test each component required for the project, thus ensuring each peripheral worked in a predictable manner. This was very valuable as a lot of the AUT lab boards have malfunctioning equipment, which this confirmation step helped identify hardware faults quickly.

The next step was programming the communications protocol using UART which was setup up with a baud rate of 38400 and in the format 8N1. The protocol was achieved by waiting for a start byte (0x53) to be received in the buffer of the UART, triggering an interrupt. If a start byte is received, the program would then move to a read state and begin saving bytes to an array. With each new byte, the program checks if it is at the end of the array or if the current byte is a stop byte (0xAA). If either of these are false, the program continues to save the byte and increases the pointer to the next address in the array. If it is

true, then the instruction and data is saved into their corresponding variables, the array is cleared, and the state is set back to the idle state where it waits for another start byte. The state diagram of this can be seen in figure 2.

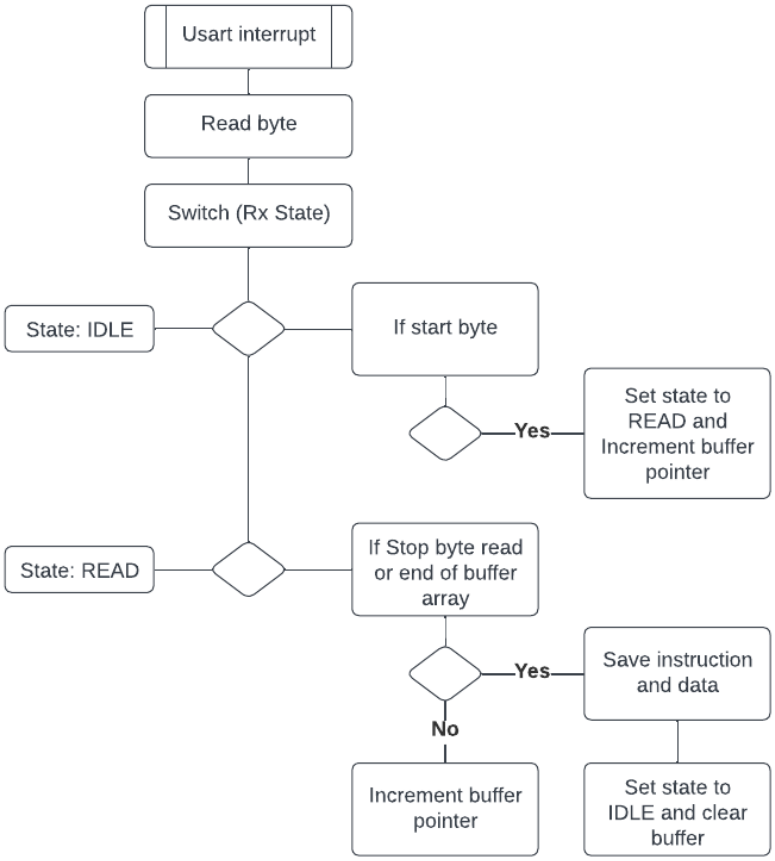

Figure 2: MCU Parsing Incoming Communications Algorithm

3.2 C# Graphical User Interface (GUI)

The Windows form functionality found in Microsoft Visual Studio was used to build the GUI, which allows for quick development of a windows application. The main programming paradigm used here is object orientated using events and event handlers. This is the standard format for windows forms and allows the program to react to certain user input. All the elements in the form toolbox were drag and drop which also included the automatic generation of event handlers. Once the GUI had been formatted using the drag and drop system, it was then a matter of programming each of the forum elements' interactions and functionalities. Starting with the LED tab it was first ensured that when an instruction was sent, that the correct reply was received with the help of debugging. Once this was completed it was simply a matter of repeating this for other functionalities, relying heavily on the debugger in both the C and C# program to understand the flow of data. The last tab implemented the graphing of data, database data logging and the PI portion of a PID controller for temperature control. Pseudo code was found on Wikipedia[5] in order to

implement the PID control. Suitable prototyping values were found after some experimental adjustment. The correct values for scaling the temperature were found in the lab-board schematic being 50mV/°C with a max of 80°C and therefore the calculation in (1) shows the maximum voltage from the temperature sensor.

$$50mV \times 80 = 4V \tag{1}$$

Because the max input voltage to the ADC was 5.15V this allowed the calculation of the percentage of max temperature reading in volts to the max ADC input (2). From this the maximum digital value for the temperature was found with equation (3).

$$\frac{4}{5.15} = 78.125\ \% \tag{2}$$

$$0.78125 \times 256 = 200 \tag{3}$$

Following this, the equation (4) was developed to convert to and from the digital reading and degrees Celsius.

$$\frac{T_{digital}}{200} = \frac{T_{Celsius}}{80} \tag{4}$$

3.3 Database Connection

Xampp[1] was used for the local server which uses Apache and MySQL. This was setup using a YouTube guide[8] which showed how the MySQL library functions worked and how to connect to a local server. A password protected account with read and write privileges was created so that it could issue SQL queries. A new database and table were also made, which then made it possible to use the MySQL library functions to connect to the server and upload some test data. Once this was achieved it was simply a matter of automating the data-logging process connecting it to the timer event handlers.

3.4 System Design

In figure 3 we can see the overview of the connected hardware devices that entails the system. From the peripherals of the microcontroller all the way through to the local sever.

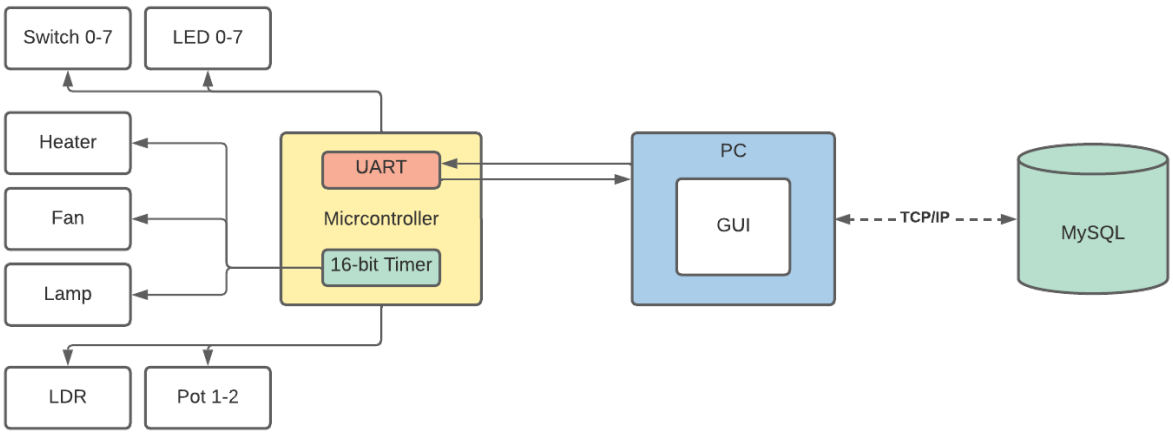

Figure 3: Flowchart of Connected Devices

4 Results

Running simulations was simple, by having the lab-board flashed with the firmware solution and connected to the PC over USB serial; the C# program was run and had all the functionalities tested and executing as expected.

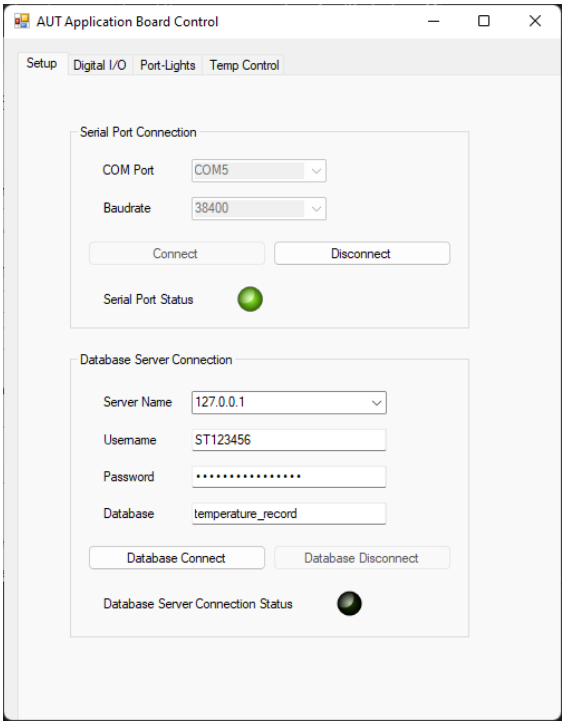

Figure 4: Setup Tab

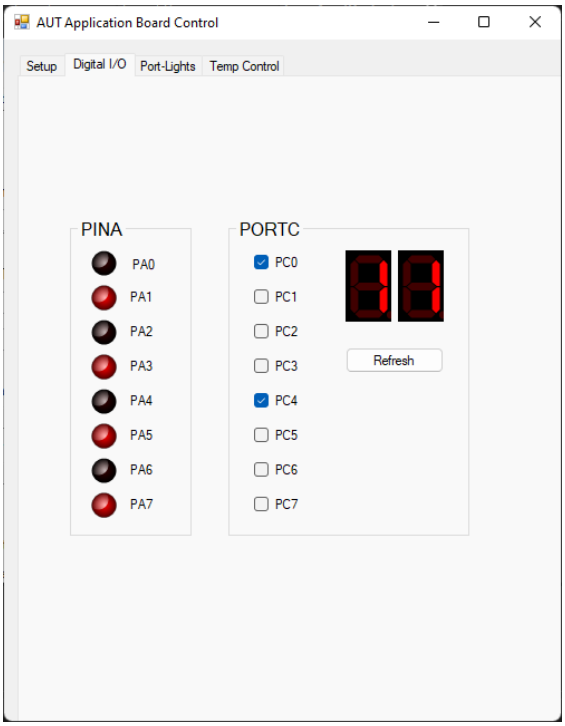

Figure 5: Digital I/O Tab

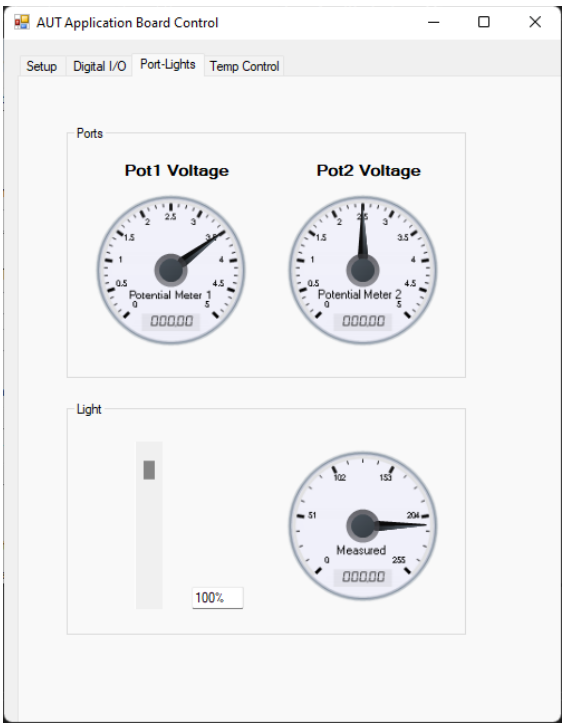

Figure 6: Pot Lights Tab

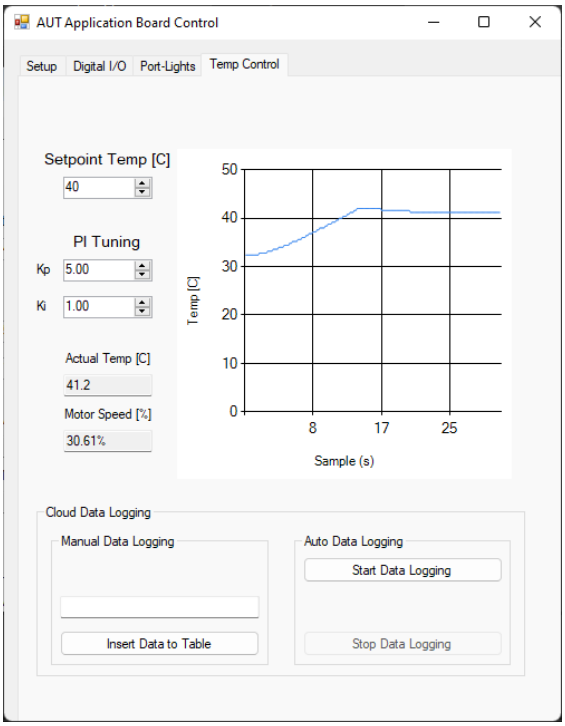

Figure 7: Temperature Control Tab

timestamp	temperature	remark
9/05/2023 8:51:48 pm	33.6	ST123456
9/05/2023 8:51:49 pm	34	ST123456
9/05/2023 8:51:49 pm	34	ST123456
9/05/2023 8:51:49 pm	34.4	ST123456
9/05/2023 8:51:50 pm	34.4	ST123456
9/05/2023 8:51:50 pm	34.8	ST123456
9/05/2023 8:51:50 pm	34.8	ST123456
9/05/2023 8:51:51 pm	35.2	ST123456
9/05/2023 8:51:51 pm	35.2	ST123456
9/05/2023 8:51:52 pm	35.6	ST123456
9/05/2023 8:51:52 pm	36	ST123456
9/05/2023 8:51:52 pm	36	ST123456

Figure 8: Example of Logged Data into Database

5 Discussions

The serial communications over UART proved to be the biggest challenge as it was difficult to see how external software parsed the bytes I was sending and receiving. The first program that I used was puTTY[6], which allowed a basic serial connection with

keyboard input. This was the tool I used the most and worked out to be very inefficient. This was because there is a lack of feedback as to how the bytes are being sent and received. This was later replaced with another program called realterm[7], which I found to be an extremely well rounded tool, very explicit in showing what is being sent and received. After finding out about this tool I was able to quickly debug my C program to ensure it is working in a predictable and reliable manner.

When creating the GUI, I was asked to make a separate class to keep all the AppBoard methods in but when I did this, I was unable to access the form controls because they were private. I later read that it is not recommended to access form controls from an external class. A compromise was made to keep all the code within the Form1 class. After I had completed this project, I had found out that the best way to do this was to have private instance variables that save the states and values in the Form1 class that the AppBoard class could access through getters and setters. These states and values would be updated from the event handlers throughout the program allowing for the two classes to be fully encapsulated.

Regarding the PI controller it could be further tuned through more research on the topic. As it is not a simple task. It will require a formidable amount of time to fully understand and calculate correct K_p and K_i values, that of which is out of the scope for this prototype.

An adjustment was also made with the prescaler for the 16-bit timer, because when having no prescaler the motor would stop when OCR1A was set at around 180 of 399. By the dividing the clock by eight meant that the top value was 49 to achieve 20kHz for the motor. Changing the setup to this resolved the problem by reducing the motor speed step resolution.

With the database user account, when I made it without a password, I was able to connect and make SQL queries, but when I used a password, I was then unable to connect. This was because as well as the database privileges, the user required global privileges to write to the database. Once Identified this issue was easily resolved.

6 Conclusion

Overall, the project was very successful, and all the requirements were met. A chance for further development was discussed with the PI controller as it required fine tuning and expert knowledge.

7 References

[1] 'XAMPP Installers and Downloads for Apache Friends'. https://www.apachefriends.org/ (accessed May 08, 2023).
[2] 'A Simple Vector-Based LED User Control - CodeProject'. https://www.codeproject.com/Articles/114122/A-Simple-Vector-Based-LED-User-Control (accessed May 08, 2023).
[3] A. Thirugnanam, 'Aqua Gauge', *CodeProject*, Sep. 04, 2007.

https://www.codeproject.com/Articles/20341/Aqua-Gauge (accessed May 08, 2023).
[4] D. Brant, 'Seven-segment LED Control for .NET', *CodeProject*, Jul. 01, 2009. https://www.codeproject.com/Articles/37800/Seven-segment-LED-Control-for-NET (accessed May 08, 2023).
[5] 'PID controller - Wikipedia'. https://en.wikipedia.org/wiki/PID_controller (accessed May 08, 2023).
[6] 'Download PuTTY - a free SSH and telnet client for Windows'. https://www.putty.org/ (accessed May 08, 2023).
[7] 'RealTerm: Serial/TCP Terminal', *SourceForge*, Oct. 13, 2017. https://sourceforge.net/projects/realterm/ (accessed May 08, 2023).
[8] 'How to connect phpMyAdmin MySQL Database in Visual Studio .NET Core C# Windows Form Application - YouTube'. https://www.youtube.com/watch?v=M6JJWrz2J9A (accessed May 08, 2023).