ENGD3051S ADVANCED EMBEDDED SYSTEMS

Coursework Briefing

Design of a Data Logger DC Motor Speed Controller

Module Leader: Mr. Pratheepan Gunaratnam
pratheepan.gunaratnam@dmu.ac.uk

Type of assessment	Component Marks	Date Set	Date Due
Code Submission	100%	18 March 2023	7 May 2023
Demonstration & Viva		18 March 2023	14 May 2023

Title:	*Design of a data logger DC motor speed controller*
Style:	*Individual work*
Assessed by:	*Firmware and viva/demonstration*
To be submitted by:	*Code to be submitted online by: 07/05/2023* *Viva/demonstration to take place: 14/05/2023*
Feedback available:	*At regular intervals throughout the year and in any scheduled laboratory session. Demonstration/viva feedback will be immediate and verbal.*
Marks available by:	*At the end of your demonstration*
Anonymously marked:	*No (individual project)*
Percentage of module:	*100%*

Table of Contents

PLEASE READ THIS CAREFULLY

The coursework for this module is a single project, **'Firmware design for a DC motor speed controller'**. This is an individual project, with a short demonstration of the final firmware you will produce. There are NO formal reports required. However, you will be encouraged to keep and maintain an online journal (a.k.a. a logbook) on the Blackboard shell to which you will be expected to make ***regular*** updates. This logbook is not assessed in itself, but is there to enable you to record your progress and thought processes as you progress through the project, and will help you to evidence your work during the viva/demonstration.

1 Aims and objectives

You are to design the firmware for a speed controller for a DC motor**.** This project is predominantly firmware. Most of the hardware will be provided in the form of the laboratory development boards. However, you will be expected to construct small add-on circuitry on breadboard.

1.1 Platform

The hardware platform you will use for your design is the DMU-designed ATMega328P Microcontroller Development Board. These are available in E & E Lab main building. This board is based around an ATMega328P microcontroller, and this Atmel AVR-based microcontroller is the target platform for your firmware. The board contains a DC motor attached to a small fan, and an infra-red (IR) sensor that will allow you to count the number of times the IR beam is broken by each of the three fan blades. The board also contains:

- a 3x4 keyboard matrix
- a rotary encoder switch
- a 7-segment LED display
- an accelerometer module (comprising accelerometer, gyroscope and thermometer)
- a number of individual coloured LEDs
- 2 pushbuttons
- a 2 line by 16 character LCD display
- expansion port

All of the above items are interfaced to, and therefore can be controlled by, the ATMega328P microcontroller

Not all of these features may be required in order to implement this specification.

1.2 Schematic

A schematic for the development board is available and may be downloaded from the Blackboard shell. This should be consulted, along with the datasheets for the associated peripheral ICs, in order to determine how the firmware should be constructed to manipulate the required hardware.

1.3 Kernel

A minimal low-footprint 'operating system' is provided. This provides memory management, timers, message queues and a round-robin task manager: common services provided by most embedded operating systems. It is there to help you, and the source code is provided. This should be made use of in your final design. Source code to the kernel is available and can be downloaded from Blackboard, along with a template project to get you started.

1.4 Features and specification

Your firmware must implement the following features:

1 Display (Main Operation)

1.1 The 2x16 LCD display should display two three-digit values, along with appropriate text, to indicate the demanded speed (what is set by the operator via the keypad) and the actual speed (as measured). The manner and position on the display of these two values and accompanying text is up to the designer, but at all times when the system is powered, the display should be legible and there should be no ambiguity between the values as displayed. Such a display could look like:

DEMAND SPD: 200
ACTUAL SPD: 200

1.2 Leading zeroes should **not** be suppressed.

1.3 When the operator changes the speed via the keypad, a cursor on the display over the demanded speed should allow the operator to visualize the entry of one digit at a time. This will include the backspace function (see4).

1.4 If the operator enters a speed outside of the range specified in section 5, the display should display the characters ‘ERR’ in place of the demanded speed for 2 seconds +/- 0.5 second. The display should then revert to the value displayed before the out-of-range speed was entered.

2 Keypad:

2.1 All keyboard switching should be fully debounced.

2.2 A key is accepted as ‘pressed’ at the point when the debounce is complete if it is determined that the transitions on the keyboard matrix are consistent with an actual keypress.

2.3 If more than one key is pressed at the same time, the first key to be detected as pressed is the one to be accepted.

2.4 Numerical values should be set by the operator via the keypad.

2.5 The ‘#’ key is an ‘enter’ key – the value entered by the operator will be accepted by the firmware once this key is pressed.

2.6 The ‘*’ key is the backspace key – this will delete the previous digit entered. If no digits are entered, this key will do nothing.

3 7-segment LED display

3.1 The 7-segment LED display should display the current key being pressed, for the duration for which it is pressed. '#' should be displayed as 'E', while '*' should be displayed as 'b'.

3.2 When no key is pressed, the decimal point should be illuminated.

3.3 The decimal point must be extinguished for the duration for which a key is pressed.

4 Analog pot

4.1 This is not used in this specification.

5 Push switches S1 and S2

5.1 These are not used in this specification but may be used if you wish for any additional purpose as desired by the designer (e.g. debugging).

6 Individual LEDs

6.1 Individual LEDs are not required for this specification, and thus may be used for any purpose as desired by the designer (e.g. debugging).

7 Control

7.1 On power up/reset:

7.1.1 The message "Starting…" should be displayed on the screen until the firmware is fully initialized. This period should not exceed 2 seconds, but may be less than 2 seconds.

7.1.2 The date/time may be set by hardcoding it during initialization. It does not need to be the actual date and time - it may be hardcoded to any *valid* date and time to allow testing of the datalogger

7.1.3 Once initialized for the first time, the display should behave as required by section 1

7.1.4 A valid date and time should be programmed into the real-time clock and the display should read in accordance with section 2.

7.1.5 the motor should be stationary (not rotating).

7.1.6 the 2x16 display should read zero for both demanded and actual speed.

7.1.7 the 7-segment display should be blank, with the decimal point illuminated.

7.2 Motor speed control:

7.2.1 Motor speed control should be achieved using open-loop control.

7.2.2 Maximum displayed speed error (displayed demanded value to actual value) is 10%. Some slack will be allowed in this number to account for the different motors in different development boards having substantially different characteristics.

7.2.3 Maximum actual speed error (as measured by an external tachometer) is 10%

7.2.4 Minimum motor speed is 50 revolutions per second (rps).

7.2.5 Maximum motor speed is given by the maximum speed possible when the motor is driven from a PWM source with a duty cycle of 1. This will have to be determined experimentally by the designer.

7.2.6 Operators shall not be able to enter a speed out of this range without the ‘ERR’ message being displayed (section 6) and the value being rejected.

8 Data Logging

8.1 Data is to be logged once per second when the open-loop speed controller is active, i.e. for demanded speeds greater than zero. The logging can be suspended for the duration where the demanded speed is zero.

8.2 A log entry will be made every 2 seconds, and additionally once at the point when a new speed value is entered by the keypad.

8.3 Each logged entry will record the day, month, year, hour, minute, second, , demanded speed, actual speed.

8.4 Each logged entry will be written to the external I^2C EEPROM.

8.5 Once the demanded speed is set to 0 RPS, the data logging will stop, and the logged data will be sent to the serial port. The date should be in a human-readable comma separated variable (CSV) format, while the demanded speed and actual speed may form part of the textual log message The contents of which should be able to be cut and pasted from the serial terminal and saved as a text file with .CSV extension. Such a file should be able to be imported into spreadsheet packages such as Excel.

8.6 The external EEPROM contains a finite storage space. Once the storage is filled, the datalogger should act like a flight recorder; i.e. ‘wrap around’ and overwrite from the beginning, but do so without causing corruption of the store.

9 Firmware restrictions

9.1 The firmware shall be constructed in such a manner as to facilitate ease of expansion with additional I/O.

9.2 The provided kernel must be used.

9.3 Languages:

9.3.1 C or C++ should be used. Some functions may be written in machine code using AVR assembler language where appropriate.

2 UKSPEC learning outcomes

SM1p: Knowledge and understanding of scientific principles and methodology necessary to underpin their education in their engineering discipline, to enable appreciation of its scientific and engineering context, and to support their understanding of relevant historical, current and future developments and technologies.

SM1m: A comprehensive knowledge and understanding of scientific principles and methodology necessary to underpin their education in their engineering discipline, and an understanding and know-how of the scientific principles of related disciplines, to enable appreciation of the scientific and engineering context, and to support their understanding of relevant historical, current and future developments and technologies.

EA1p: Understanding of engineering principles and the ability to apply them to analyse key engineering processes.

EA1m: Understanding of engineering principles and the ability to apply them to undertake critical analysis of key engineering processes.

EA4p: Understanding of, and the ability to apply, an integrated or systems approach to solving engineering problems.

EA5m: Ability to use fundamental knowledge to investigate new and emerging technologies.

EP2p: Knowledge of characteristics of particular materials, equipment, processes, or products.

EP2m: Knowledge of characteristics of particular equipment, processes, or products, materials and components.

EP3p: Ability to apply relevant practical and laboratory skills.

EP3m: Ability to apply relevant practical and laboratory skills.

EA4m: Understanding of, and the ability to apply, an integrated or systems approach to solving complex engineering problems.

3 Submission requirements:

Note that the primary assessment is the demonstration and viva. We ask for the code to be submitted ahead of this time in order that it can be scrutinized, but this forms part of the demonstration process and this is not a separately weighted component: **in other words if**

you submit the code but fail to attend the viva/demonstration, you will score a mark of zero. During the viva/demonstration you will be expected to answer technical questions relating to the code you have written. Hardware will be available during the demonstration if you wish to refer to this during the viva/demonstration. Failure to submit either will result in failure of the module.

Note: no written report is required. All assessment is determined from your code and your performance in the viva.

3.1 Firmware submission

A portal will be opened on Blackboard closer to the date specified in section 8. The firmware should be zipped up and a single **.zip** file submitted. Please note that .rar files are not acceptable. Also note the following:

- Please ensure you have **checked that the .zip file contains all the files** (and there will be more than one) of your firmware project. You do not need to include the provided kernel in your .zip file (unless you have modified it).
- Please ensure that you can:
  - **unpack the .zip file**
  - **that the firmware compiles** once you have unpacked it (i.e. the newly unpacked project will compile, not your original one).

There will be no leniency given if the firmware you have submitted fails to compile. You will be expected to have checked it.

Note that the purpose for submitting the firmware is to allow time for the assessors to scrutinize it prior to the demonstration. The actual assessment point is the demonstration, no marks are awarded for the code submission alone. You must demonstrate the same code as was submitted to Blackboard in accordance with the this section – the code may not be modified between submission to Blackboard and your demonstration. This is to ensure a level playing field (demonstration schedules may be spread across more than one week owing to student numbers and lab availability).

3.2 Viva/demonstration

The viva/demonstration will be held during the day indicated in section 8. As numbers of students and staff availability may dictate that this takes place on a day different to your scheduled sessions, we will communicate the exact day and time to all students registered via the Blackboard shell closer to the time.

The viva is the main point of assessment. **Failure to attend the viva will lead to failure of the project as a whole.** The vivas are intended to probe each of the learning outcomes with respect to the project you have submitted, as well as to determine authorship. If you have worked consistently and submitted your own work, you will have no problems with the viva.

4 Schedule of submission dates/deadlines:

Week/date	Description
07/05/2023, 11:59	Submission of firmware code via Blackboard

14/05/2023	Demonstration/vivas

5 Hardware Development:

5.1 Breadboard Construction

You will be required to construct some additional circuitry on breadboard. For those who studied ENGD2001 and/or ENGD2103, you already have breadboards and you are expected to bring one breadboard to your laboratory sessions. Components specific to the requirements of this module will be provided.

Your hardware will be interfaced to the expansion port by way of right-angle header pins on the development board.

5.2 Additional hardware

You are required to add a real-time clock to your breadboard. This is an I^2C device and will require a 32,768 Hz “watch-crystal” to maintain time. The IC you will be using is the MCP7940M-I/P and a datasheet is provided on Blackboard.

You are also required to add an ***external*** EEPROM device to the breadboard. This is also an I^2C device. ***Please note that the use of the internal EEPROM of the ATmega328P is strictly prohibited for this assessment.*** The part type you will be using is a 24LC512-E/P and a datasheet is also provided for this assessment.

5.3 Interfacing to the development board.

Your hardware will be interfaced to the expansion port on the development board. This is situated at the front of the board via a row of connectors and is easily visible.

6 The lab equipment:

6.1 Labs

We have a standard laboratory with oscilloscopes, signal generators and power supplies and Development boards.

7 Marking scheme:

Criterion:	Clear Fail <30%	Marginal Fail 30 - 39%	Pass 40 - 59%	Merit 60 - 69%	Distinction >70%
Style of provided source code	Shambolic, or very little original source code presented. No clearly logical execution path and code may not compile. No, or very few comments. No encapsulation, poor or no effective use of operating services.	Code is very unclear and can not be easily read to determine whether or not the code is capable of functioning as claimed (if errors were corrected). Very few or no comments. Indentation style makes code difficult to read. No encapsulation, poor use of operating system services	Code compiles, is reasonably clear in most places. Indentation style is generally acceptable in most places. There are enough comments to allow the basic principle of the design to be seen. Some attempt has been made at encapsulation in most places. Attention has been paid to where efficiency is required.	Good, readable code with a clear indentation style. There may be some convoluted areas but would in general be acceptable in industry. Encapsulation and use of operating system services are generally well applied. Coding style and choice of structure reflects requirements for efficiency	Excellent quality code, instantly readable with strong use of encapsulation and operating system services. Very few errors or unnecessarily convoluted structures. Good use of language features and choice of design methodology matches the requirements of efficiency.
Specification requirement (as listed in marking scheme)	Lack of or very limited advance on code given in the examples. No understanding of the code submitted (if any) and/or completely non-working when tested on hardware. Structureless and poorly designed. Could not be easily fixed to make it work.	An attempt has been made to implement the requirement, but this is deficient in too many ways to enable the design as it stands to be easily fixed to work as desired.	Specified requirement has been implemented and can be shown to work as per the specification with few or no compilation errors that were not easily fixed by the student at the viva. There may be a number of design or stylistic errors, but learning outcomes have been hit and the student clearly understands the code they have written, its strengths, and how it is deficient when compared with the specification	Specified requirement has been fully implemented and works well with no compilation or significant design errors. Meets the specification. Learning outcomes have been hit and the student clearly and fully understands the code they have written and how it was designed. There may be one or two minor errors, easily fixed and understood	Specified requirement fully implemented and completely meets the specification. Fully understood at the viva. Any errors are minor and do not significantly detract from the implementation of the specification.
Performance at viva with respect to Q&A	Does not understand the code they have submitted. Technical knowledge is substandard. Very little in the way of technical knowledge	Some technical knowledge and an attempt to answer wider questions in the subject area. There is some understanding of the code that has been submitted but this is weak.	Student performance is of an acceptable standard, it is clear that the student understands their own submission and that the learning outcomes are hit, but the wider knowledge may be weak. In general the assessors would feel confident if this student was employed in industry in a role involving embedded system design	Fully understands the code in their submission and can articulate it well. There may be some gaps in the wider knowledge of embedded systems, but in general their knowledge is sufficient for them to be able to develop a design independently if they had access to written materials	Very good wider knowledge, fully understands the code and design they have submitted and can articulate this fully. Would be able to easily develop an embedded system independently and would know where to find further information to broaden their knowledge.

8 Late submission of coursework policy

Late submissions will be processed in accordance with current University regulations.

Please check the regulations carefully to determine what late submission period is allowed for your programme.

9 Academic Offences and Bad Academic Practices:

These include plagiarism, cheating, collusion, copying work and reuse of your own work, poor referencing or the passing off of somebody else's ideas as your own. If you are in any doubt about what constitutes an academic offence or bad academic practice you must check with your tutor. Further information and details of how DSU can support you, if needed, is available at:

http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/academic-offences.aspx and

http://www.dmu.ac.uk/dmu-students/the-student-gateway/academic-support-office/bad-academic-practice.aspx

Marking guide (to be completed after viva)

ID: _____ Name: _____ Date: _____

Learning outcomes met overall: ☐ Yes ☐ Partially ☐ No

Note: If the submitted source code cannot be made to compile at all, and the student is unable to remedy the situation at the viva, then marks can only be awarded for sections 1 and 3 below. Section 2 marks should be recorded as zero.

	Criterion			*Weight (%)*	*First Marker mark (%)*	*Weighted mark (%)*	*Moderator mark (%)*	*Moderator weighted mark (%)*
1	Style of provided source code			10%		0.000%		0.000%
2	Requirements (as numbered in specification)	1	Display	15%		0.000%		0.000%
		2	Keypad	15%		0.000%		0.000%
		4	7 segment display	5%		0.000%		0.000%
		8.1	Power up/reset	10%		0.000%		0.000%
		8.2	Motor speed control	15%		0.000%		0.000%
		9	Compliance with firmware restrictions	5%		0.000%		0.000%
3	Performance at viva with respect to Q&A (to cover comprehension and a deeper under-standing of the code you have written).			25%		0.000%		0.000%
	Totals			**100.00%**		**0%**		**0%**

First marker: _____ Moderator: _____

Agreed mark _____

Further comments: