

ENGD2103
FORMATIVE ASSIGNMENT
2022 - 2023

Author: M A Oliver and T Allidina

Title: *Formative Assignment Specification.*

V1.0 First Release

Aims and objectives

This document provides the specification for the Formative Assignment. *This does not count towards the final grade for the module.* The purpose of this assessment is for assessing progress for qualitative feedback.

You should already have the necessary components for this assignment including Arduino and breadboard, and you should already be close to completing the hardware construction.

The purpose of this assignment is to provide you with the experience of constructing, driving and testing simple digital subsystems.

This formative assignment will be based around three short tasks. For each task, a piece of code will need to be written. A Hardware Abstraction Layer (HAL) module will evolve as the project progresses. Having a complete HAL will simplify one of the aspects of your summative assignment later.

These subsystems will form part of a larger system that will be used in your summative assignment later on in the term.

The schematic from Week 1 covers all three tasks in this formative assignment and will be used for the summative assignment later on in the term.

Additionally, it is recommended that you have a box to transport your circuitry to prevent it from being damaged.

Academic Practice Offences

Any attempt to obtain a pass by unfair means is a very serious offence. Submissions will be checked for plagiarism, cases of copying the work of another student, supplying solutions to fellow students (whether intentional or not), or the use of other unacknowledged material will normally result in a fail grade of zero marks for the entire assignment. You should be aware that formal disciplinary action, as described in the University Regulations, may also be taken against individuals who fall foul. The disciplinary panel has the power to inflict serious penalties including expulsion from the university.

















The bottom line is that you will need to submit your own work, and present your own work. You should know your own work intimately and be able to answer any question about it during the presentation.

1 Task 1: Traffic Lights Controller

For this task, you will be required to implement a traffic lights controller.

You will be required to:-

1. Assemble the schematic from Week 1 on breadboard. Once the breadboard assembly is complete, you will have one arrangement of Red, Amber and Green LEDs representing Traffic Lights Set 1, and another arrangement of Red, Amber and Green LEDs representing Traffic Lights Set 2. Consider a crossroads where Set 1 controls North-South traffic and Set 2 controls West-East traffic.
2. Write the code for controlling the traffic light sequencing, ensuring that your timing sequence complies with the requirements shown below:-

STATE	LIGHTS SET 1	LIGHTS SET 2	TIME (s)
1			1
2			1
3			5
4			1
5			1
6			1
7			3
8			1

3. Create a Hardware Abstraction Layer (HAL) code module for all platform-specific functionality.

2 Task 2: Counter

The purpose of this task is to give you experience of:-

- Interfacing a 7-segment LED display to a microcontroller via a shift register.
- I/O port expansion (i.e. dealing with situations where there are insufficient port-pins)
- 'Bit-banging' a simple protocol.
- Debouncing a switch

2.1 Task Requirements

In this task, you will be required to implement a counter that counts the number of times a push-button switch is pressed. The least-significant nibble (4-bits) of the count will be displayed on a 7-segment digital display.

The Arduino Nano resides at the heart of the system. The 7-segment digital display is a Common Cathode device. This will be connected to a 74HC595 shift register. Three lines (CLOCK, DATA and LATCH) of the shift register are connected to three digital pins on the microcontroller (as per the schematic from Week 1).

A push-button switch connects between Arduino line A0 and GND. This push-button will require a pull-up resistor. However, it is recommended that you use an internal pull-up.

For this exercise, you will be required to write code for maintaining the count, driving the shift-register and debouncing the switch.

For the code, you will be required to:-

1. Create a table (array) of byte values for displaying the characters '0' to '9' and 'A' to 'F' on the 7-segment digital display. The character set is shown below:-



2. Create a global variable called `counter` and initialize it to zero.

3. Write a function such as:-

```
void writeToShiftRegister(byte value)
```

This function will write the byte parameter `value` most significant bit first into the shift register. A bit-banging approach will need to be used to drive the DATA, CLOCK and LATCH lines.

4. Write a function:-

```
void waitForKeypress()
```

This function will wait until a debounced keypress of 500ms is detected.

5. The main `loop()` will perform the following operations:-
 - Wait for a debounced keypress
 - Increment the value of `counter`
 - Use the function `writeToShiftRegister()` to display the least-significant nibble of `counter`.
6. **Add** platform specific functionality to your ***existing*** Hardware Abstraction Layer (HAL) module.

3 Task 3 Accelerometer Interfacing Exercise

The purpose of this task is to give you experience of:-

- A 'lightweight' introduction to I²C interfacing using the 'Wire' library on the Arduino platform
- Driving and interrogating an I²C device; in this case an accelerometer
- Interpreting the data from the accelerometer

3.1 The Scenario

Consider a typical smartphone. When the phone is held in landscape orientation, the contents of the screen can rotate such that they appear in landscape form with correct orientation. Now, rotate the phone to portrait orientation. The contents of the screen will now rotate such that they appear in portrait form with correct orientation.

A typical smartphone contains an accelerometer that measures gravitational acceleration along x-, y- and z-axes. Firmware inside the smartphone uses this data to determine the orientation of the phone. Once the orientation is known, firmware inside the phone will rotate the contents of the screen for correct viewing.

In this exercise, you will be using an accelerometer and writing firmware to determine the orientation of your breadboard. Your solution will be effectively performing an equivalent operation to that of a smartphone.

3.2 Orientation Detection Task

From the assembly in Week 1, you will have already interfaced an accelerometer to your Arduino Nano module. The accelerometer is contained in an MPU-6050 integrated circuit that appears on your GY-521 module.

You will be required to write firmware for the Arduino Nano that samples gravitational acceleration, every 333ms, along x-, y- and z-axes.

From the x, y and z values of gravitational acceleration, your firmware will be required to determine the orientation of the accelerometer.

Firstly, you should design your firmware such that it writes the x-, y- and z-values of gravitational acceleration to the serial port. Using the outputted values, select threshold values for determining the orientation. The choice of threshold values should be documented in your code. **Hint:** look for the dominant component. Also consider whether **relative** values rather than absolute values may provide a better solution.

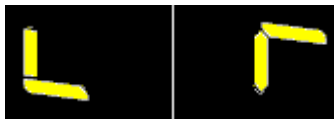
The results of the orientation need to be displayed on the 7-segment LED display introduced in the previous exercise. The LED display should display:-

'F'	when the board is lying flat
'b'	when the board is lying flat, with its base facing upwards
'L'	when the board is held in landscape
'U'	when the board is held upside-down in landscape
'l'	when the board is held in portrait (to the left)
'r'	when the board is held in portrait (to the right)

The following image shows how the upper-case characters 'F', 'b', 'L' and 'U' are to appear on the 7-segment display.



The following image shows how the lower-case characters 'l' and 'r' are to appear on the 7-segment display.



All these characters need to be displayed so they can be read normally according to the current orientation of the breadboard.

To summarize, for this task, you will be required to:-

1. Write the Arduino code for reading data from the accelerometer and displaying the perceived orientation on the 7-segment display. If, in a particular iteration, none of the thresholds are exceeded, the previous value should be displayed.
2. Document the threshold values used for determining each orientation of the board, (i.e. Flat, Flat (base-up) Landscape, Upside-down, Portrait-Left and Portrait-Right).
3. **Add** platform specific functionality to your Hardware Abstraction Layer (HAL) module. This should contain all platform-specific functionality for all three exercises.

4 Submission

The deadline for submission is 12:00 on Monday 31st October (Week 5).

For the submission process, you are required to upload five items to a submission journal on Blackboard:-

- Photograph of your breadboard (please avoid proprietary formats such as .HEIC as they cannot universally be opened)
- Source code for Task 1 (Traffic Lights)
- Source code for Task 2 (Shift register / 7-segment display / counter)
- Source code for Task 3 (Accelerometer interfacing)
- Source code file(s) for the final HAL module.

Brief demonstrations of your work will take place during the scheduled laboratory sessions in Week 5 to attribute ownership of the submitted code.

Faculty of Computing, Engineering & Media (CEM)
Coursework Brief 2022/23

Module name:	Embedded Systems Fundamentals		
Module code:	ENGD2103		
Title of the Assessment:	Formative Assignment		
This coursework item is: <i>(delete as appropriate)</i>		Formative	
This summative coursework will be marked anonymously:		No	
The learning outcomes that are assessed by this coursework are: <ol style="list-style-type: none"> 1. Basic interfacing 2. Bit-banging a simple protocol 3. Interpretation of data from a sensor 4. Structured code development 			
This coursework is:	Individual		
This coursework constitutes 0% of the overall module mark.			
Date Set: 14/10/2022			
Date & Time Due (the deadline):	31/10/2022 at 12.00 noon		
In accordance with the University Assessment and Feedback Policy, your marked coursework and feedback will be available to you on:			21/11/2022
<p>You should normally receive feedback on your coursework no later than 15 University working days after the formal hand-in date, provided that you have met the submission deadline</p> <p>If for any reason this is not forthcoming by the due date your module leader will let you know why and when it can be expected. The Associate Professor Student Experience (CEMstudentexperience@dmu.ac.uk) should be informed of any issues relating to the return of marked coursework and feedback.</p>			
When completed you are required to submit your coursework via: <ol style="list-style-type: none"> 1. Blackboard submission journal <p>If you need any support or advice on completing this coursework please visit the Student Matters tab on the CEM Blackboard shell.</p>			
Late submission of coursework policy: <p>Late submissions will be processed in accordance with current University regulations.</p> <p><i>Please check the regulations carefully to determine what late submission period is allowed for your programme.</i></p>			
Academic Offences and Bad Academic Practices: <p>Please ensure you read the section entitled “Academic Offences and Bad Academic Practice” in the module handbook or the relevant sections in this link: BaseCamp Link: Overview: Assessment and Good Academic Practices</p>			

Tasks to be undertaken: Construct the breadboard. Write the three pieces of code	
Deliverables to be submitted for assessment: <ul style="list-style-type: none"> • Photograph of breadboard (.JPG or .PNG only. NO .HEIC FILES) • Final HAL code • 3 main code files 	
How the work will be marked: By face-to-face demo. Qualitative feedback to be provided.	
Module leader/tutor name:	Dr Michael Oliver
Contact details:	michael.oliver@dmu.ac.uk

Should you need any further information or advice please email
cemadvicecentre@dmu.ac.uk