

ENGD2104
ADVANCED LIGHTS CONTROLLER AND INTRODUCTION TO IoT PROJECT
2021-2022

Author: M A Oliver and T Allidina

Title: ENGD2104 Coursework Task 2.

1 Aims and objectives

This document provides an outline for the ENGD2104 Task 2 coursework.

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

- Extending the functionality of the existing Arduino/ESP8266 Resource Manager.
- Setting-up the ESP8266 as an Access-point (AP) with Webserver.
- Building a Web-based front-end to drive and monitor a physical system.

2 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.

Please see:-

<https://www.dmu.ac.uk/current-students/student-support/exams-deferrals-regulations-policies/student-regulations-and-policies/academic-offences.aspx>

for more details.

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.

3 The Task

The overall objective of this task is to design and implement a multi-functional lights system that can be controlled and monitored using a WiFi-enabled device such as a laptop or a cell-phone.

The Resource Manager from your previous assessment will form the heart of this project. This will be based around an Arduino and an ESP8266.

For the Arduino Nano, you will be required to write a number of code modules that are capable of running concurrently. In other words, they must each be capable of running in a non-blocking fashion. These modules will be:-

1. Resource Manager Finite State Machine (FSM)
2. A multi-functional traffic lights controller
3. Barrier crossing controller
4. Motor racing lights controller
5. A 3-colour LED controller
6. Two switch-debounce modules
7. A heartbeat module
8. A Tilt-Sensor and Thermometer based on the MPU6050
9. A simple Command Interpreter
10. Scheduler(s)

Your code from previous assignments should provide a solid start-point. For those who did not successfully complete Coursework 1, a model solution for the Resource Manager will be provided once all the vivas from Coursework 1 have taken place.

For the ESP8266, you will be required to write a number of code modules that are also capable of running concurrently. These modules will be:

1. A Resource Manager Finite State Machine (FSM)
2. A module for sending commands / requests to the Arduino to control the lights systems and to obtain status information.
3. A Web-based interface for driving the controller and displaying the current status of the physical system.

From a hardware perspective, you will be required to construct a circuit on breadboard. Using the parts from earlier exercises, the circuit will contain:-

- 1 Arduino Nano
- 1 ESP8266 NodeMCU module.
- 6 LEDs (2 x red, 2 x yellow and 2 x green) for the lights controller (with appropriate current limiting series resistors)
- 1 tri-colour LED (with appropriate current limiting series resistors)
- Current limiting series resistors (as outlined previously)
- 2 Pushbutton Switches
- 1 GY-521 Module with MPU6050 Accelerometer / Gyroscope
- 7-segment display with shift register.
- 3V3 protection circuitry for the ESP8266.

For this exercise you are free to choose whether to drive your GPIO lines using a bare-metal approach or to use `digitalWrite()`, `pinMode()` and `digitalRead()`

4 Multi-functional Lights Module

The Multi-functional Lights system will be configurable, via a command set, to perform the following operations:-

- Traffic Lights controller with Equal Priority
- Traffic Lights controller with Set 1 Priority
- Traffic Lights controller with Set 2 Priority
- Maintenance Mode
- Barrier Crossing Mode
- Motor Race Start Lights Mode

You will need two variables to keep track of which mode the multi-functional lights module is running. One to keep track of the current mode. One to keep track of the next mode that is to be run. Before a mode change can take place, the current sequence must be completely finished. In general, the traffic lights will perform their cycle then return to State 1 as per the tables below. Any transitions in mode will be applied as State 1 is being entered. For the Barrier Crossing and Motor Race modes, any transition in mode will occur in the Idle state.

Traffic Lights Controller



















The Traffic Lights controller will control a simple 2-way crossroads.

- Traffic Lights Set 1 control the North-South lanes.
- Traffic Lights Set 2 control the East-West lanes.

On reset, the Traffic Lights will start with both sets of lights having equal priority (i.e. both sets get green for the same duration). It will be possible to configure the priority with the following three options:

- Equal Priority,
- SET 1 Priority (i.e. Set 1 has a longer green period than Set 2), or
- SET 2 priority (i.e. Set 2 has a longer green period than Set 1).

The table below shows the timings for the traffic lights.

STATE	LIGHTS SET 1	LIGHTS SET 2	TIME (s) Equal Priority	TIME (s) SET 1 Priority	TIME (s) SET 2 Priority
0			2	2	2
1			1	1	1
2			1	1	1
3			<u>5</u>	<u>7</u>	<u>3</u>
4			1	1	1
5			1	1	1
6			1	1	1
7			<u>3</u>	<u>7</u>	<u>5</u>
8			1	1	1

State 0 is a safe state that will bring all traffic to order. This state will be run just once following a Power-on Reset. Once State 0 has been executed, the system will continually cycle-through States 1 to 8.

The system will need to monitor which set of lights is currently active:

- During States 1 – 4, Set 1 is considered active
- During States 5 – 8, Set 2 is considered active







Maintenance Mode

In Maintenance Mode, both sets of Traffic Lights will continuously flash Amber; off for 500ms, on for 500ms. Continuously flashing amber is usually interpreted as “proceed with caution”, i.e. traffic can still proceed cautiously while the lights are under maintenance.

Barrier Crossing Controller

For the Barrier Crossing Controller, SW1 will be used to count the trains in and SW2 will be used to count the trains out.

The table below shows the sequence for the barrier crossing.

STATE	LIGHTS SET 1	LIGHTS SET 2	Operation
Idle			Wait for SW1 press
0			5 seconds
1			0.5 seconds
2			0.5 seconds
3			0.5 seconds Returns to state 2 if SW2 presses have not been detected for every SW1 press. Returns to idle state if SW2 presses have been detected for every SW1 press.

The system will start in the idle state and remain there until SW1 is pressed (indicating an approaching train). At this point, the system will run States 0 and 1 just once (SW2 presses are ignored during these states.)

The system continually cycles between States 2 and 3, terminating once the same number of SW2 pushes as SW1 pushes were detected (indicating that all trains on the track have passed).

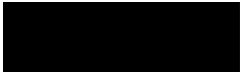

You are required to design this using FSM(s) before implementing the code.


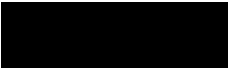

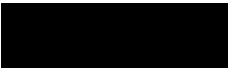

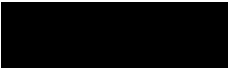








Motor Race Start-lights Controller

Ideally five red LEDs would be used to perform the start-lights sequence. With the LEDs available, some artistic license will need to be applied!



The table below shows the sequence for the motor race start-lights controller.

The system starts in the Reset state. The start-sequence commences with a press of SW1. States 1 through 5 are run; pressing SW1 at any time during the start-sequence aborts the start-up. (Once the start sequence is complete any press of SW1 from this point onwards causes the race to be red-flagged.) After completion of the start sequence, the lights go off for two seconds (State 6) and the race is underway. The system will remain in State 7 until a switch press is detected.



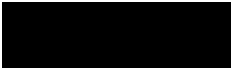
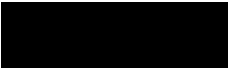
STATE	LIGHTS SET 1	LIGHTS SET 2	Operation
			

Reset			Wait for SW1 press
1			1 second
2			1 second
3			1 second
4			1 second
5			Random delay between 2 and 5 seconds.
6			2 seconds
7			Race underway: remain here until:- Pressing SW2 returns to Idle state. Pressing SW1 red-flags the race (jump to Red Flag State)

The table below shows the lights for a red-flagged race.

STATE	LIGHTS SET 1	LIGHTS SET 2	Operation
Red Flag			Red Flagged Race. Pressing SW2 returns to Idle state

The table below (and continued overleaf) shows the lights for an aborted start.

STATE	LIGHTS SET 1	LIGHTS SET 2	Operation
Abort 1			Aborted Start Sequence 0.5 seconds
Abort 2			Aborted Start Sequence 0.5 seconds Pressing SW2 returns to Idle state Otherwise return to Abort 1 state.

You are advised to design this using FSM(s) before implementing the code.

5 Tri-Coloured LED

A tri-coloured LED is used in this project to represent a flashing LED of a service vehicle. In this project the LED should repeatedly:-

- Increase in intensity from zero to maximum over the course of 1 second.
- Decrease in intensity from maximum to zero over the course of 1 second.

The tri-coloured LED needs to be connected to pins 9, 10 and 11 on the Arduino Nano. The intensity can be controlled by driving the LED using PWM using the `analogWrite()` function.

The following modes should be supported

- Off
- Amber (indicating hazard)
- Blue (indicating emergency)
- Green (indicating doctor on call)
- Red (indicating stationary police car)

6 The Pushbutton Switch Modules

In this exercise, there will be two pushbutton switches, SW1 and SW2.

The switches need to be individually debounced. A debounce time of 300ms is required.

7 The 7-Segment Display

The 7-segment display will be used to display the current mode of operation:

- Lower-case 't' for maintenance mode
- Upper-case 'E' for traffic lights, equal priority
- Number '1' for traffic lights, Set 1 priority
- Number '2' for traffic lights, Set 2 priority
- Upper-case 'B' for barrier crossing.
- Upper-case 'F' for motor racing lights.

8 Heartbeat

The d.p. element of the 7-segment display shall be used as a 'heartbeat' that will repeatedly blink on for 500ms and off for 500ms.

9 On-board LEDs

The on-board LED of the NodeMCU will be used to indicate "I am master" on the Resource Manager (on = master, off = not master).

10 Tilt and Temperature Module

The MPU6050 module, as well as containing an accelerometer and gyroscope, contains a temperature sensor.

In an ideal situation we want our traffic lights assembly to be perpendicular to the horizontal. However, sometimes wayward vehicles hit the traffic lights assembly. In this exercise, the accelerometer will be used to determine whether the traffic lights assembly is listing (as a result of a collision).

If the assembly is deemed to be tilting, the status should indicate Danger.

If the assembly is deemed to be not tilting, the status should indicate that the system is Good.

In this exercise, the temperature of the unit is required. The temperature may be obtained by reading registers 0x41 (TEMP_OUT_H) and 0x42 (TEMP_OUT_L) to form a 16-bit quantity TEMP_OUT. From this, the temperature in Celsius may be calculated from

$$T = (\text{TEMP_OUT} / 340.00) + 36.53$$

11 Command Set

The ESP8266 will have the ability to send simple commands to the Arduino. The Arduino will have to be able to interpret this simple command set.

ESP8266 to Arduino command requests

The ESP8266 will need to be able to send the following commands to the Arduino as single-byte opcodes. These are detailed below.

0x61:	Set Traffic Lights (equal priority)
0x62:	Set Traffic Lights (Set 1 priority)
0x63:	Set Traffic Lights (Set 2 priority)
0x64:	Set Traffic Lights (Maintenance Mode)
0x65:	Set Barrier Crossing.
0x66:	Set Motor Racing Start Lights.
0x67:	Set Tri-coloured LED to off.
0x68:	Set Tri-coloured LED to varying intensities of amber (Hazard).
0x69:	Set Tri-coloured LED to varying intensities of blue (Emergency).
0x6A:	Set Tri-coloured LED to varying intensities of green (Doctor).
0x6B:	Set Tri-coloured LED to varying intensities of red (Police Stopped).
0x6C:	Get status.

Arduino to ESP8266 Command responses

For each command received, the Arduino will respond with

- ACK ('A') or NAK ('N') (1 byte)
An ACK (Acknowledge) byte will be sent if the command opcode appears in the list above. A NAK (not-Acknowledge) byte will be sent if the command opcode is not listed.

In the case of 0x6B (Get Status) the ESP will send the following sequence:-

- ACK ('A') (1 byte)
- Advanced lights mode – current cycle (1 byte)
 - 0x81: Traffic Lights – Equal priority
 - 0x82: Traffic Lights – Set 1 priority
 - 0x83: Traffic Lights – Set 2 priority
 - 0x84: Traffic Lights – Maintenance
 - 0x85: Barrier Crossing
 - 0x86: Motor Race Start Lights
- Advanced lights mode – next cycle (1 byte)
 - 0x91: Traffic Lights – Equal priority
 - 0x92: Traffic Lights – Set 1 priority
 - 0x93: Traffic Lights – Set 2 priority
 - 0x94: Traffic Lights – Maintenance
 - 0x95: Barrier Crossing
 - 0x96: Motor Race Start Lights
- Current lights status (1 byte)
 - '1': (0x31) Traffic Lights – Set 1 currently active
 - '2': (0x32) Traffic Lights – Set 2 currently active
 - 'M': (0x4D) Traffic Lights – Maintenance
 - 'I' (0x49) Idle mode for Barrier Crossing or Motor Racing lights
 - 'A' (0x41) Active mode for Barrier Crossing or Motor Racing lights
- Tri-coloured LED status (1 byte)
 - 'O': (0x4F) Off / Not-Applicable
 - 'A' (0x41) Amber – Hazard / Service Vehicle
 - 'B': (0x42) Blue – Emergency State
 - 'G': (0x47) Green – Doctor on call
 - 'R': (0x52) Red – Police car stopped
- Temperature in whole degrees Celsius (1 byte)
(Please bear in mind the possible range of values for the temperature. Therefore it would make sense to employ a signed byte for this.)
- Tilt Status (1 byte)
 - 'G': (0x47) Good – no issues
 - 'D': (0x44) Danger – tilt detected.
- nano_active_time (4 bytes)

This will indicate the time that the Nano has been active in milliseconds.

Normal Action

- The ESP8266 will interrogate the Arduino every 650ms to acquire the latest status information. The interrogation will take place over the i2C bus in a similar way that accelerometer data was acquired in the previous assignment.
- The ESP8266 will issue commands to change the traffic lights mode only when requested via the Web interface.

12 Scheduler

The code will contain a lot of code modules. You will need a scheduler to activate and deactivate various code modules based on the last received command (mode). Depending on your design, you may wish to employ two schedulers.

On power-up the system should default to Traffic Lights (equal priority), with the 3-colour LED modes active but with the 3-colour LED off.

The main lights module (traffic lights, maintenance, barrier crossing and motor racing start) should run concurrently with the 3-colour LED.

13 Web Interface

The ESP8266 will run a Web-server and host a Webpage.

The Webpage will provide the controls for controlling the traffic lights system and for returning status information.

The Webpage will need to provide the interface for setting the traffic lights system to:-

- Traffic Lights – Equal Priority
- Traffic Lights – Set 1 Priority
- Traffic Lights – Set 2 Priority
- Traffic Lights – Maintenance Mode
- Barrier Crossing Mode
- Motor Racing Start Mode

The Webpage will need to provide the interface for setting the tri-coloured LED to:-

- Off
- Hazard / Service Vehicle (Amber)
- Emergency (Blue)
- Doctor on Call (Green)
- Stationary Police Car (Red)

It will also have the interface for the following functionality.

- Requesting current status

The Webpage will also need to show the following status information:-

- Current Lights Mode, i.e. traffic lights, maintenance lights, barrier crossing, or motor race
- Next Lights Mode, i.e. traffic lights, maintenance lights, barrier crossing, or motor race

- Current Lights Status, i.e. Set 1, Set 2, Maintenance, Idle (barrier crossing or motor race), Active (barrier crossing or motor race).
- Tri-colour LED Status, i.e. Off, Amber, Blue, Green, Red.
- Temperature
- Tilt Status (Good / Danger) – the use of red for danger could be beneficial
- Time that Nano has been active (to nearest 0.01s).

The Webpage will need to be updated every 10 seconds.

However, some of you might be familiar with frameworks such as AJAX, JQUERY, etc, for client-side interaction with a Webpage which could be used for a faster real-time updates. If you are familiar with these, you are free to use them. Please note that this not within the scope of this module and as such no teaching time / tutor time will be spent on these technologies.

The Webpage design can be subject to your creativity.

14 Implementation Strategy

As this project is somewhat involved, you are advised to tackle it in a modular fashion.

Work on one small section at a time. Compile your code frequently to identify and correct typos, missing brackets / braces / etc. Once you have implemented a module, test it and get it working before moving to the next stage.

The worst thing you can possibly do (aside from leaving it until the last minute!) is to type in several hundred lines of code then click the “Upload” option.

15 Submission

You are required to work individually and complete this exercise within your scheduled lab sessions, commencing Week 23. Your deadline will be the end of teaching Week 27, i.e. you will have approximately 5 teaching weeks to complete it. You are strongly advised to start working on this task as soon as possible. **The task at hand is significant and you are likely to run out of time if you underestimate the challenge.**

Once complete, you will need to provide a short demonstration of your system during your scheduled laboratory session in Week 30. You must submit your work via a corresponding submission Blackboard journal by Friday 8th April at 17:00pm.

You will need to submit:-

- Your Arduino Source Code
- Your ESP8266 Source Code
- A single PDF file. This file should contain:-
 - A photograph of your system (in .pdf, .jpg or .png format)
 - A scan of your Barrier Crossing FSM(s)
 - A scan of your Motor Racing Lights FSM(s)

- (Schematic. This is only required if you have changed from the recommended schematic. If you have made no changes, there is no need to include a schematic.)
- Video demonstration of your system.

Late Submissions

Unauthorized late submission within 14 calendar days of the deadline will be marked, but will result in a cap of the pass mark of 40%. Any late submissions after 14 days of the deadline will result in a mark of 0%.

16 Marking Scheme

This laboratory exercise will contribute towards 50% of your final module mark. You must work individually. Do not attempt to present somebody else's work as yours!

CRITERION (POINTS)	DETAILS
Documentation (20)	Quality of the breadboard layout (6) FSM for the barrier crossing (7) FSM for the motor racing start lights (7)
3-Colour LED modes (10)	Off (1) Red (1), Green (1), Blue (1) Amber, including quality (3) Fading (3)
Traffic Lights Modes (10)	Maintenance Mode (2) Correct sequencing of main lights (3) Priority sequencing (5)
Barrier Crossing (10)	Correct sequencing (5) Correct switch behaviour (4) Mode change only on idle state. (1)
Motor Racing Start Lights (10)	Correct start sequence (5) Aborted start (2) Red flagged state (2) Mode change only on idle state. (1)
Switch debounce (2)	SW1 correctly debounced (1) SW2 correctly debounced (1)
Seven segment (4)	Correct modes displayed on seven segment (2) Correct heartbeat without race conditions (2)
Web preliminaries (4)	Access point (2) 10-second refresh (2)
Web page (12)	Basic functionality (2) Displaying status data (3), tilt (2), temperature (1), time (2) Further creativity (2)
Ability to change between modes (18) A requirement for a pass is the ability to change between at least two modes using the web-interface. The mark will be capped at 38% if this is not achieved.	Maintenance (1) Traffic Lights: Equal priority (1) Traffic Lights: Set 1 priority (1) Traffic Lights: Set 2 priority (1) Barrier Crossing (1) Motor Racing Start Lights (1) 3-Colour LED: Off (1) 3-Colour LED: Red (1) 3-Colour LED: Blue (1) 3-Colour LED: Amber (1) 3-Colour LED: Green (1) Mode change on correct state of sequence (3) Mode change on main lights does not affect 3-colour LED (2) Mode change on 3-colour LED does not affect main lights (2)

A similar mark scheme will be applied to the demo/viva with questioning replacing the Documentation section.

Faculty of Computing, Engineering & Media (CEM)

Coursework Brief 2021/22

Module name:	Embedded Design for IoT		
Module code:	ENGD2104		
Title of the Assignment:			
This coursework item is:	Summative		
This summative coursework will be marked anonymously:		No	
This coursework is:	Individual		
This coursework constitutes 50 % of the overall module mark.			
Date Set:	11 March 2022		
Date & Time Due:	Fri 8 April 2022 at 17:00		
<p>Your marked coursework and feedback will be available to you on:</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> <p>Note that you should normally receive feedback on your coursework by no later than 20 University working days after the formal hand-in date, provided that you have met the submission deadline.</p>		<p>Thursday 12 May 2022 (on the proviso that all demos/vivas are complete).</p>	
<p>When completed you are required to submit your coursework via:</p> <p>1. Blackboard Submission Journal</p> <p>If you need any support or advice on completing this coursework please visit the Student Matters tab on the 'Computing, Engineering and Media' Blackboard shell.</p>			
<p>Late submission of coursework policy: Late submissions will be processed in accordance with current University regulations which state:</p> <p><i>"the time period during which a student may submit a piece of work late without authorisation and have the work capped at 40% [50% at PG level] if passed is 14 calendar days. Work submitted unauthorised more than 14 calendar days after the original submission date will receive a mark of 0%. These regulations apply to a student's first attempt at coursework. Work submitted late without authorisation which constitutes reassessment of a previously failed piece of coursework will always receive a mark of 0%."</i></p>			
<p>Academic Offences and Bad Academic Practices:</p> <p>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</p>			
<p>Tasks to be undertaken:</p> <ul style="list-style-type: none"> Implementation of the specification. 			

Deliverables to be submitted for assessment:

- **Arduino Source Code**
- **ESP Source Code**
- **Documentation (single PDF containing photo of breadboard and FSM diagrams)**

How the work will be marked:

- **50% of the mark will be based on a demo/viva**
- **50% of the mark will be based on off-line marking**

Module leader/tutor name:**Dr Mike Oliver****Contact details:****michael.oliver@dmu.ac.uk**