ELEC3227 Embedded Networked Systems

Coursework 2023/24

Assignment Set:	Tuesday 24th October 2023
Assignment Type:	Individual coursework, but working in teams
Submission Deadline(s):	Fri 17th Nov 2023, 11:00 (*standard document*) Fri 15th Dec 2023, 12:00-17:00 (*practical demonstration*) Mon 08th Jan 2024, 16:00 (*written report*)
Feedback:	You will receive oral feedback on your design at the Coursework Review Meeting. There will also be a Q&A drop-in session. You will receive a mark and written feedback within four weeks of the report deadline[1].
Mark Contribution:	This assignment is worth 40% of your mark for ELEC3227
Required Effort:	You should expect to spend up to 60 hours on this coursework
Examiners:	Professor Geoff Merrett and Dr Alex Weddell

Learning Outcomes

Having successfully completed this coursework, you will be able to:

- demonstrate knowledge and understanding of layered networking models
- interpret standardisation documents
- communicate your technical work
- design, implement and test embedded networking protocols on practical hardware

Academic Integrity

Please ensure that you are aware of the University's regulations and guidance on academic integrity, particularly on collaboration vs. collusion. While you will be working in *teams*, this is an individual coursework and the report that you submit must be your own work[2]. Any use of AI-based text- or code-generation tools, or collaboration with any other individual, must be clearly declared and is likely to result in a reduced mark. While you are expected to discuss (and standardise) the protocol for your layer(s) with your *peer-teammate(s)*, these discussions should only be at the abstract level, i.e. not discussing the implementation, code, etc. To avoid any potential issues with recycling, any student who is repeating the coursework must not attempt the same layer(s) again (i.e. you should pick a different layer to design and implement this year).

Disclaimer: *We try to make this coursework engaging, practical and useful (and hopefully fun). However, this makes it quite intricate and complex. As such, we may need to make minor tweaks and modifications to the specification as the coursework develops to ensure fairness. Apologies if this happens, and please be patient with us!*

[1] *Optionally*, if you submit your report by 16:00 on Fri 15th Dec 2023 AND email gvm@ecs.soton.ac.uk to inform us it is your final submission (hence waiving the right to make any further submissions), we will endeavour to return your provisional mark and feedback to you before the start of the exam period.
[2] An explicit exception to this is your Standard Document. It is expected that this will be the same for *peer-teammates*; **it is inherently collaborative and is allowed to be identical**!

Individual or Team?

This coursework is individually assessed, but you will be working in *teams*, as below:

Team A1	Team B1	Team C1	Team D1	Team E1
Enrico, O Castrelo Brignell, X Clark, W	Osborne, P Betson, J Bedwell, J	McGregor, O Aries, R Cai, H	Szabo, E Davies, J Antoniou, A	Yusri, M Vaux, J Zulkepli, S
Team A2	**Team B2**	**Team C2**	**Team D2**	**Team E2**
Vu, G Chu, C Sedhai, S	Wong, G Kwek, A Thadathil, J	Goh Yeh Wei, J Tan, S Yap, H	Lafi, M Chia, C Toland, W	Leung, Y Tang, W Law, S
Team A3				
Turner, J Hudson, K Couchman, H				

Some **important** definitions:

- Individuals that are in the same *team* (e.g. everyone in Team A1) are *teammates*.
- *Teams* that share the same letter (e.g. A1, A2) are *peer-teams*.
- Individuals working on the same layer(s) in *peer-teams* are *peer-teammates*.

This is an individual coursework. If you can't get a response from a *teammate* or *peer-teammate* for an extended period, please notify Prof Merrett and Dr Weddell. In this situation, you should continue to work on the coursework/submission(s) alone.

Your Task

Your *team's* task is to design and implement a full, working, network architecture capable of supporting a smart lighting application(s). Your architecture should be designed to support multiple 'inputs' (e.g. light switches) and multiple outputs (e.g. lights) in the network, where each input can control one or more outputs. The architecture should support a distributed application, rather than requiring a centralised 'hub'.

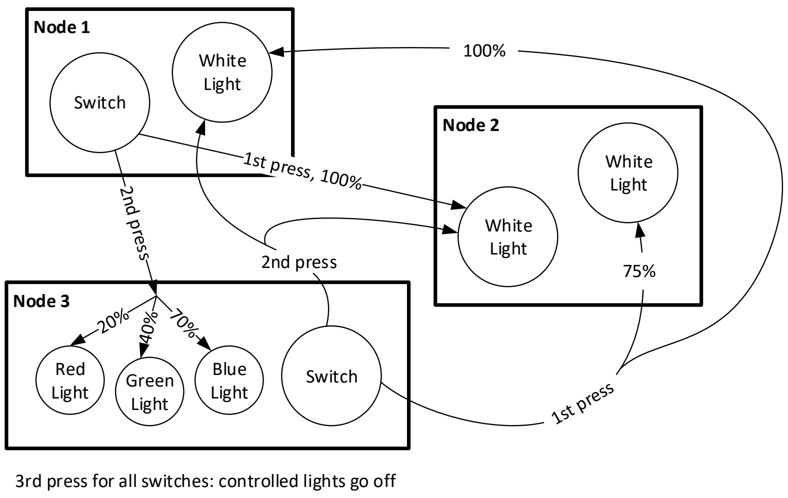

Figure 1: an example of a smart lighting application. Note the arrows indicate functionality, not necessarily direct communication: for example, Node 2 could be out-of-range of Node 1, and hence packets need to be routed via Node 3.

Figure 1 shows an **example**[3] of this, showing three nodes (also referred to as 'stations' or 'devices'). Different presses of the switches cause the lights on other nodes to change to different states. How your *team* interprets the task, the services that you offer in *your* protocols, and how you elaborate upon the brief are entirely up *you* and your *teammates* to decide. For example, supporting multiple applications on nodes (which requires the use of multiple ports), or more elaborate payloads (such as specifying a sequence of times, brightnesses and colours that the lights should go through).

The coursework requires your *team* to implement a protocol stack, including the *application layer* (APP). However, marks are not awarded for *application software* – don't waste time on designing and implementing a complex application. Interfacing with the APP via the Il Matto's UART or switches/LEDs will suffice. However, the *teammate* responsible for the APP will need to consider the interface an application has with it; the APP does not care about button presses or LEDs, where lights or switches are located within a building, etc. Consider other APP protocols you may be familiar with (e.g. HTTP, FTP, SMTP). Also, your protocol shouldn't be fixed to restrictions of your testing scenario (e.g. only supporting 3 or 6 devices).

Splitting the Task across your Team

Although you are working in *teams*, you must each work on unique and clearly defined tasks. The recommended way to implement this would be to give each *teammate* different layer(s) from the 5-layer reference model considered on this module (PHY, DLL, NET, TRAN, APP), where everyone should have at least one layer to work on[4]. However, in some cases, it might be necessary to split a layer into sub-layers (e.g. the MAC and LLC sub-layers of the DLL) in order to appropriately divide tasks. Whatever you choose, it should be clearly defined, agreed amongst *teammates*, and in-line with the spirit/principles of protocol layering.

When you settle on your division of labour, one of the first things that you need to do is agree on the (initial) functionality of each layer and then define the software interfaces. That is, you need to agree on a prototype for the function to call when one layer needs the services of another. Keep in mind that the end product is one integrated program, which needs to be small enough to fit onto an Il Matto. Although you are working in *teams*, this is an individual coursework. You should only interact with your *teammates* through clearly defined and agreed interfaces (you may also need to discuss the limited resources, e.g. RAM, CPU cycles, etc).

The Networking Architecture

You are not being asked to implement an existing network architecture[5] (e.g. ZigBee); in fact we want you to create your own! However, there is no requirement to invent new algorithms (e.g. you can use existing error control, routing algorithms, etc). Your architecture **must** be implemented using the Il Matto + RFM12B[6]. Your team should have three (or four) of each; hence you should *plan* to be able to demonstrate your network using all of these.

[3] You are free to interpret the task as you wish, and don't have to mirror this functionality

[4] If you are in a team of 4, we recommend two of you independently work on the DLL. While both will need to liaise to collectively agree the standard, they should otherwise not work together. The standard should include optional functionality allowing both teammates to focus on different things.

[5] You are being asked to implement a network architecture, not a protocol stack. Your *team* could choose to implement multiple protocols at a particular layer (e.g. both a reliable and an unreliable DLL); but there is no necessity to do so and your network architecture may be a single protocol stack.

[6] The coursework requires you to implement your architecture using the Il Matto and RFM12B hardware. **Don't use something else**. We recommend developing on the Il Matto from the outset.

TRAN:	Control [2]	SRC Port [1]	DEST Port [1]	Length [1]	APP Data [1-114]	Checksum [2]

NET:	Control [2]	SRC Address [1]	DEST Address [1]	Length [1]	TRAN Segment [8-121]	Checksum [2]

DLL:	Header [1]	Control [2]	Addressing [2]	Length [1]	NET Packet (or part of) [1-23]	Checksum [2]	Footer [1]

Figure 2: Segment/packet/frame field structures defined by the coursework

The segment/packet/frame field structures in Figure 2 are defined by the coursework and are non-negotiable (please note that management, rather than data, segments/payloads/frames do not need to adhere to this). We do not specify a field structure for the APP layer. The numbers in square brackets refer to the field size in bytes; if you do not want to use a field (or some of the bits within), you should fill the field's bits with zeros. You have flexibility in how you use some of these fields. For example, the use of the bits in the Control fields; the Checksum field does not have to be a true 'checksum', and could be a parity bit(s), CRC, etc.

As can be seen in Figure 2, If any NET packet is >23 bytes, it will need to be split into multiple frames. Likewise, if the APP data is >114 bytes, it will need to be split into multiple TRAN segments. You may wish to start by designing your system to work with small amounts of data (<9 bytes, so that it all gets encapsulated into a single DLL frame), and then expand it later.

You'll notice there are addressing fields in the DLL (often referred to as the 'MAC address'). You may use these to help route data through the network, by the NET asking the DLL to transmit a packet to a particular next hop, addressed by the addressing field. This is similar to the approaches that you will learn about in the lectures. However, you may use broadcasting in the DLL, whereby a node broadcasts the packet to all neighbours that can hear it. A sensible approach would be to assume that a destination MAC address of 0x00 means that it's broadcast (this is something that should be included in your standard). The NET then has to decide whether to rebroadcast (or drop) packets. This is an approach often adopted by epidemic approaches (e.g. packet flooding) – but you need to constrain it from going on forever (e.g. the same node doesn't keep receiving and rebroadcast the same packet).

Standardising your Network Architecture

A proprietary network architecture is fine for communicating within your own network, but limits the potential of a device. We would therefore like you to plan for your *team's* network to be able to successfully communicate with the nodes in your *peerteam's* network. To enable this, you must also agree a standard[7], which defines:

- The services, and interfaces that they're accessed through, offered by your protocol(s). This should be agreed with the *teammate* who is developing the layer(s) above;
- The functionality (not implementation) of the protocol. This should be agreed with *the peer-teammate* who is working on the same layer(s) as you.

[7] If you are not sure what a standard looks like, look at http://standards.ieee.org/about/get/. Yours, however, should be a maximum of two pages long!

Your standard should provide enough information for someone to communicate with your stack, even if it's implemented differently. Although you will be working in *teams*, this is an individual coursework. You should interact with your *peer-teammate(s)* only to define and agree protocol(s). No other collaboration should be necessary (in fact, you will be penalised if it occurs). You do not need to subdivide the tasks in the same way as your *peer-team(s)* (i.e. there does not have to be a 1:1 mapping between the tasks undertaken by *peer-teammates*).

Your agreed standard may include some optional functionality that is not necessarily implemented by both/all *peer-teammates*. For example, your DLL might include a bit in the Control field which indicates whether the Checksum field is using a single even parity bit (core functionality) or a 32-bit CRC (optional functionality). However, the presence or absence of these optional functionalities should not stop the 'core' (non-optional) functionality from working. This ensures no-one is limited by the capabilities/aspirations of *peer-teammate(s).*

Implementation and Testing

You have choice over the level of ambition/functionality that *you* are attempting. We would prefer to see something working and well analysed/evaluated, rather than something complicated that is only part-implemented and not thoroughly evaluated. You are <u>strongly</u> encouraged to include some simple core functionality in each standard, alongside any more ambitious approaches that you wish to attempt (which might be optional in your standard). Implement and evaluate this first, before attempting more ambitious parts. The Il Mattos are constrained devices: it's a much better (and safer) idea to get a simple stack working first, and then expand functionality. Think about the resource implications of your design decisions.

Think about ***how*** you're going to demonstrate and evaluate your layer(s) working.

- You may choose to try to integrate your layer(s) with some of your *teammates* (for example someone responsible for the NET might demonstrate/evaluate it working across the wireless channel via a *teammate's* implemented MAC and PHY). However, ensure that you can 'isolate' *your* layer(s) to demonstrate and evaluate what it is doing.
- You should ensure that you are not reliant on the success of your *teammate(s)*. You may choose to build a test-harness allowing your layer to communicate directly with itself. For example, if you were working on the DLL, you might demonstrate that you pass it a payload of data, show that it splits it into multiple frames with sequence numbers, adds error detection etc, and then – instead of passing it to a real PHY, the test harness loops the same frame back into the DLL, showing that it can check for errors, recombine all of the frames, and extract and return the payload.

Think carefully about how to evaluate your implementation, rather than just including a screenshot of debug output showing that it worked once. We want to see results ('*how well does it work*'), not debug outputs (typically '*does it work*') or raw data. For example, for an error detection/correction algorithm, can you run a large number of data + errors through it, evaluating its performance for different types of errors? For a MAC, you could investigate the number of collisions/backoffs and failed transmissions across a large number of frames?

Deliverables and Marking Scheme

This coursework is marked out of 100, and contributes 40% of the credit for this module.

The coursework has deliverables as listed below. Where maximum page-limits or time-limits are stated, anything exceeding these will not be marked. Written deliverables MUST be on A4 with 1 inch (2.54 cm) margins, using single-column single-spaced 12pt Times New Roman.

1) Coursework Review Meeting [total of 20 marks]
(held virtually on Teams, during the lab slot on Friday 17th November 2023)[8]
The examiners will meet with all *peer-teams*; refer to the table below to find your session. The link in the right-hand column should be used to join the MS Teams meeting. Meetings are deliberately time-constrained, so make sure you have a plan for best using the time.

Team	Date	Time	MS Teams Link
Team A1	Friday 17th November 2023	12:00	*You will receive an invitation via email containing the link/see your calendar*
Team A2			
Team A3			
Team B1		13:00	
Team B2			
Team C1		14:00	
Team C2			
Team D1		15:00	
Team D2			
Team E1		16:00	
Team E2			

You will receive some feedback during the Coursework Review Meeting, and further feedback and your mark at the same time as your Final Report mark.

- **Draft Standard Document** [10 marks; maximum 2 pages]:
  Before the meeting (11:00 on the same day), every *teammate* must submit, via handin, a draft of the standard document for their protocol(s) which they've agreed with their *peer-teammate(s)*. It also specifies the interfaces to *teammates'* (neighbouring) layers. Standards are assessed for consistency, completeness, functionality, technical correctness.
- **Protocol Presentation** [10 marks; maximum 5 minutes]:
  Pairs of *peer-teammates* will explain the protocol logic, services, and interfaces of each standard, as per their submitted standard document. You should jointly prepare a 5-minute presentation, and both be actively involved in its delivery. You will be assessed on your explanation, justification and understanding of the protocol(s).

[8] If you do not attend your scheduled session, you will receive a mark of zero for this element.

2) Practical Demonstration [total of 25 marks]

(assessed on Friday 15th December 2023)[9]

Teams should attend their allocated demonstration session, as indicated in the table below. Demonstrations will take place in the Zepler Level 2 Laboratories, unless otherwise stated. At the end of your demonstration, we will collect in your RFM12B radio module.

Team	Demonstration Date	Time
Team A1		12:00
Team A2		12:25
Team A3		12:50
Team B1		13:15
Team B2		13:40
Team C1	Friday 15th December 2023	14:05
Team C2		14:30
▨		▨
Team D1		15:20
Team D2		15:45
Team E1		16:10
Team E2		16:35

You should demonstrate[10] the functionality of the layer(s) that you designed and implemented, to evidence your achievements. Be creative, and think how you can effectively demonstrate the functionality that you've implemented. You should take care to sufficiently and succinctly explain to us what you are showing. You will be assessed on the achievement and level of challenge associated with the layer(s) that you implemented, how technically sound they are, and how you clearly and comprehensively you were able to demonstrate their functionality.

If you are able to demonstrate your layer(s) as part of 1) a your *team's* stack (or subset), or 2) your *peer-team's* network, we would like to see this too. No marks are directly allocated to this, but it will give context and evidence to support claims that you include in your report.

---

3) Individual Report [total of 55 marks]

(due by 16:00 on Monday 08th January 2024, online hand-in)

Your report should be structured to include the following sections (and ONLY these).

- **Introduction** [0 marks[11], maximum 1 page]:
  State your name, student ID, and team letter/number. The only other thing on this page should be a simple and clear diagram of the protocol stack your *team* adopted, with clear annotations on the layers *you* contributed to.

[9] If you do not attend your scheduled session, you will receive a mark of zero for this element.
[10] We only want to see what you can demonstrate; we'll read about everything else in the report!
[11] This carries no marks. However, it helps us understand and interpret the remainder of the report.

- **Standard** [0 marks[12]; maximum 2 pages]:
  A standard document for the layer(s) *you* worked on, so we know ***what*** you were designing/implementing. It may be identical to the version submitted for the Coursework Review Meeting (if nothing changed), or may be updated to take account of modifications. We expect it to align with the functionality you ultimately designed/implemented.

- **Design** [20 marks, maximum 2 pages]:
  Provide a clear justification for ***why*** you designed *your* protocol(s) the way that you did. Justify your choice of protocols for the target application scenario. Explain how they work (not how they were implemented), e.g. using a flowchart, sequence diagram, etc.

- **Implementation** [10 marks; maximum 1 page]:
  How did you ***implement*** your protocols in C on the Il Matto? How did you write your code such that it would work on a single-threaded micro-controller that also needed to execute other layers of the stack? How did you structure your code to ensure that you followed the principles of a layered architecture (i.e. interfaces, services and protocols)?

- **Results and Analysis** [20 marks; maximum 2 pages]:
  How well do *your* protocol(s) work? Do they operate effectively under all conditions/scenarios? What is the performance? What are the code size/memory requirements? What data/evidence to you have? Avoid including screenshots of debug output; think creatively about the clearest way to analyse your data, and present it using appropriate plots and figures. How did you test and evaluate your protocol(s)? Were you able to test it working with *teammates'* layers, and/or *peer-teammates* layers?

- **Critical Reflection and Evaluation** [5 marks; maximum ½ page]:
  What are the weaknesses of your design/implementation? Be ***critical*** of your protocol choices, design, implementation. How would you resolve or improve them? What would you do differently if you did it again? What problems did you encounter working in a *team*, agreeing on interfaces, and agreeing your standard with your *peer-teammate(s)*?

- **References** [0 marks; no page limit]:
  Include references to any material used, using the IEEE style[13].

- **Appendices**:
  No appendices or additional pages are permitted.

- **Design Archive** [0 marks]:
  Please also upload a ZIP file to Handin, containing all the code you wrote. This is not assessed, but may be looked at by the examiners if they have any concerns regarding academic integrity (and we may run it through code similarity checking tools).

The marks for the demonstration will be based on what is demonstrated at the practical demonstration. The marks for the individual report will be based on what is documented in the report when submitted; but we expect this to correlate with the functionality we observed during the demonstration. RFM12B Radio modules will be collected in after your demonstration, and no development/experimental work should be conducted after this.

[12] This carries no marks, as a previous version was already assessed at the coursework review meeting.
[13] https://ieeeauthorcenter.ieee.org/wp-content/uploads/IEEE-Reference-Guide.pdf