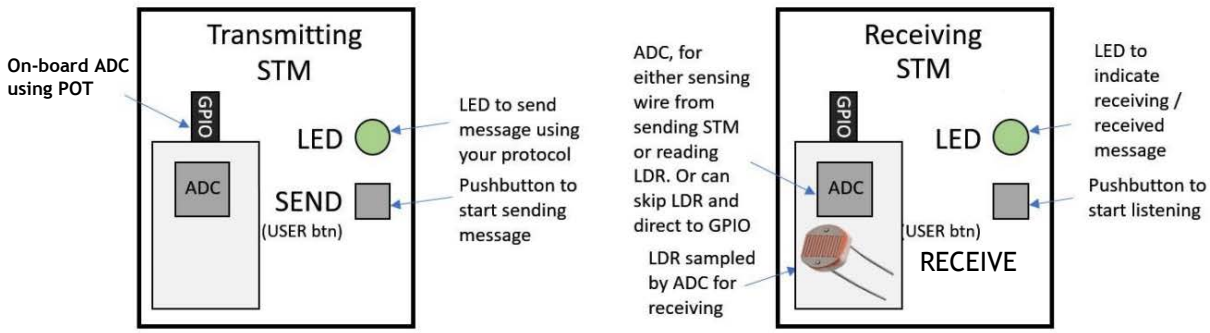
EEE3095S: Embedded Systems II
Course Project 2023
Messaging by Light!

The course project is a chance for you to use what you have learned in this course to test your problem-solving skills. **Only EEE3095S students are required to complete this project** to make up for the additional two credits of their version of the course. This can be done as a team project in groups of two or four as two UCT STM development boards will be required.

The project involves the development of two small embedded systems, requiring aspects of hardware/software interfacing and the use of ADCs and embedded communications to send out a reading from the ADC via light signals from one STM to another that is on the receiving end. The diagram below illustrates the configuration of the system involved. You need to connect up the components so that you can sample one of the on-board POTs via ADC from the transmitting STM.



An on-board push-button and LED(s) can be used respectively to tell the sending system when to sample a value from the POT and then send it as a light message using the LED. Similarly, you could use a push-button on the receive side to tell that STM to start listening for an incoming message.

On the receiver side, you have more flexibility. The minimum requirement is simply that it can listen to an GPIO pin to receive messages from the transmitting STM platform. In that case, the signals that are sent to the LED on the transmit can be duplicated to another GPIO pin that is simply wired over to the receiving STM. Note that you are not using RS232 or UART for this, you can simply 'bit bang' the GPIO while sending the message and flashing the LED. You can choose to use an LDR and ADC on the receive side, having the messages sent over light from the user LED (or other, e.g. a laser LED if you choose to hook one up), which is more complicated and time-consuming than simply listening to a GPIO line. But in either case, a video showing footage of sending and receiving a message is desired as a demonstration that you got the system working.

On the transmitter, you need to develop a Python or C program that will await for the pushbutton to be pressed. Once the pushbutton is pressed, the POT should be sampled. The sampled value should then be converted to a binary data packet and sent by the LED (and if needed, to get the message over to the receiver, duplicated to another GPIO).

You need to decide how the sampled POT value is converted to a binary message that is sent out the LED. At least have an option for slow human-viewable, option, which you can then push a button or change a setting in the code and recompile to make it work faster. For example: if the value sampled is X, you can simply send this out as a binary sequence as follows: turning on the LED for 1s to indicate start of text. Convert X into a Little Endian sequence, for each bit in sequence either turning on the LED for 500 ms if it is a 1 bit, or turning off the LED for 500 ms if it is a 0 bit. To indicate end of text, flashing the LED for 1 seconds, pulsing the LED at a 100 ms interval. You can utilize a smartphone to capture a video recording to show the POT level, the ADC reading (what can be sent out the serial line or shown as a brightness level of the LED), and then the flashing message and for demonstrating the operation of your system.

The transmitter and receiver should keep track, respectively, of samples sent and received. The transmitter it should have a counter for samples sent, and the receiver a counter for samples received. Besides the send ADC sample message, the transmitter should have another message 'checkpoint' that sends the *number* of samples it has sent to the receiver. When the receiver senses this message it should check if the number indicated in the checkpoint message is the same as that which it has received.

If the numbers differ then it should set its receiver count to the checkpoint count and indicate a missing samples alert (which can simply be flashing its LED quickly for two seconds). Else, if the checkpoint and received samples is the same, then the receiver indicates the 'all received' alert which can just be having its LED turn on, steady non-flashing, for 2 seconds. Clearly, you might think of ways that you can check that the checkpoint function is working properly.

Requirements to be completed

A. Planning and System Design Deliverables

1. List your planned activities with the project and allocation to team members

2. Description of your 'light-of-things' (LoT) message protocol (you can show a rectangular message structure illustration, and can supplement this with a timing diagram and textual description elaborating these). You need to cater at least for the means to send an ADC sample and a different checkpoint message.

3. A circuit diagram for the system (s) (i.e., you might have some non-trivial circuitry on the receiver side if you decide to use a LDR and ADC for receiving signals by light instead of just using a direct GPIO linked wire).

4. A choice on how you plan to approach the project (can connect with diagram deliverable below) and justification/motivation for design choices.

5. Design Diagram(s): hardware and software design aspects, flow chart or finite state machine diagram for software (or A flowchart detailing how system operates).

B. Code

The final hand in will be the report and your code submission. Details on these can be found in the marking guidelines below.

Project Marking Guidelines

Heading	Report
Introduction	Provide a short introduction ($\sim \frac{1}{2}$ page) to your project explaining your main design choices and how the project activities were allocated (step 1) and report has been structured. **[15 marks]**
Requirements & LoT Message Protocol	The requirements section should provide a description and diagram of the system, according to your implementation, and any accompanying text that is needed to clarify the requirements (see step 2). Highlight any departures or additions that you may have made compared to the original project description given in this document. **[15 marks]**
Specification and Design	This section should provide diagrams (flow chart or other suitable diagram) see Step 3 and 5 describing the main operation and to indicate the structuring of your implementation (e.g., code modules/classes you may be using). You do not need to provide fine detail of the system, the diagram(s) can be e.g., at the level of functions. You should also include a circuit diagram (step 3). **[20 marks]**
Implementation	This section should give some snippets of important code and explanations for this (or referring to particular functions in code files). The point here is elaborating any parts of the design diagrams that are not so straightforward to turn into code. **[20 marks]**
Validation and Performance	Provide at least a paragraph or two explaining the performance of the system. A snapshot / video (e.g. using smartphone as mentioned above) could be included and you could show test cases where you have tested that the system works reliably (e.g., using a voltmeter to indicate what ADC is reading). **[20 marks]**
Conclusion	Give a summary of the extent that the system was found to be successful. Discuss if you think that a system working in this way might be considered a potentially useful product. **[10 marks]**
References	Provide a few references if relevant.
Total	**100**