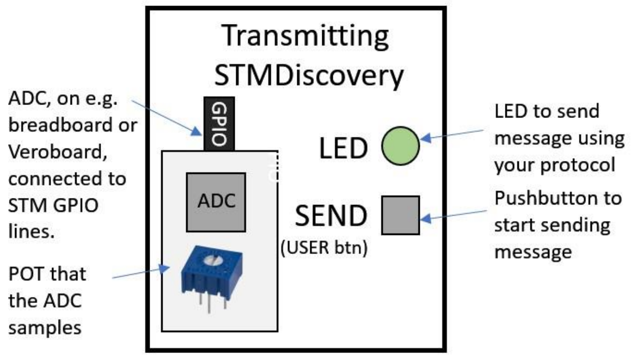
Embedded Systems EEE3096S/EEE3095S
Mini-Project Part A & B : 2022
Message by Light

The mini-project is a chance for you to use what you have learnt in this course to test your problem-solving skills. There are two parts to this project: Part A is for engineering and science students, registered for both EEE3096S and for EEE3095S. Part B is only for science student, registered for EEE3095S (although engineering students can also do this part of the project if they choose to do so, but will not receive additional marks for it).

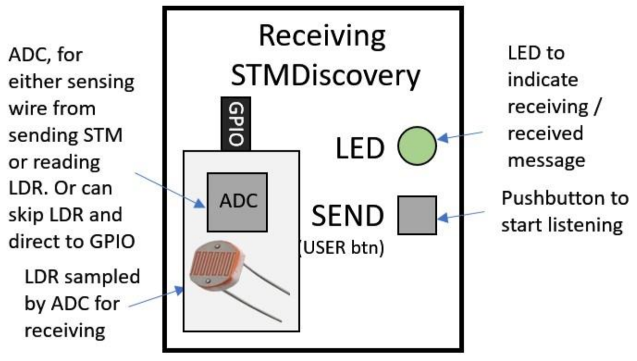
The projects use much of what was learned in the previous practicals in this course, and application of the STM32F0DISCOVERY development board. This can be done as a team project (team of up to four students).

Mini-Project Part A

This project involves development of two small embedded systems, requiring aspects of hardware/software interfacing, an ADC and developing a communications protocol by which to send out a reading from the ADC via light signals from the one, with the option to link this via wired connection to a second receiving STMDiscovery. The diagram below illustrates the configuration of the system involved. You need to connect up the components so that you can sample the POT via ADC from the transmitting STMDiscovery.



Transmitting STMDiscovery
ADC, on e.g. breadboard or Veroboard, connected to STM GPIO lines.
GPIO
ADC
POT that the ADC samples
LED
LED to send message using your protocol
SEND (USER btn)
Pushbutton to start sending message



Receiving STMDiscovery
ADC, for either sensing wire from sending STM or reading LDR. Or can skip LDR and direct to GPIO
GPIO
ADC
LDR sampled by ADC for receiving
LED
LED to indicate receiving / received message
SEND (USER btn)
Pushbutton to start listening

The onboard user pushbutton and onboard user LED can be used respectively to tell the sending system when to sample a value form the POT and then sent it as a light message using the LED. Similarly, you can use a pushbutton on the receive side to tell that STMDiscovery 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. 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 tramitter, 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 wither turning on the LED for 500ms if it is a 1 bit, or turning off the LED for 500ms if it is a 0 bit. To indicate end of text, flashing the LED for 1s, pulsing the LED at a 100ms 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 2s. 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 and Report

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

Please refer to next page for marking structure.

Part A Marking Guidelines

Table I: Project A marking Guide

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 don't 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

Mini-Project Part B

Part B is an extended version the Part A of the mini-project. Part B needs only be completed by CS students registered for EEE3095S.

This part of the project involves the addition of receiving a serial message from the STM board to the PC, the construction of a GUI or at least console display on the host, so that the PC can show the ADC readings that are being obtained by the ADC. The GUI application also needs to generate a CSV sampled recording log file of received ADC readings (via serial list). The GUI should also allow a function to activate the LoT tramission (as in activating the same transmission function that is provided when the user pushbutton is pressed). In terms of connecting to the PC and logging data, you can choose to do from the receive side – but it is not necessary to do so; as the Part B might be done by fewer students, than Part A, it would be quicker and enough to simply have the transmitter connect to the PC to send samples directly to the PC.

Your choice of graphics library to utilize for the GUI is flexible. You do not have to use a GUI, you can simply provide a menu-based console interface, e.g. which starts by displaying a welcome message, and then a list of keypresses that can be applied (e.g. A = Get and display ADC reading and add to log, E = Echo Test, to check that the serial comms is working, S = Do LoT Transmission, X = exit application).

If you would like to attempt to use a GUI library, suggestions are: Qt (available for C or Python); or PyQt5, Tkinter or just PySimpleGUI (these available for Python). Note that your GUI doesn't need to do anything too fancy, mainly plan it around ease of testing and remote access to your STM-based sampling system.

Part B Marking Guidelines

You need to have a separate submission for Part B, despite it being an extension of Part B. The aspects to do for Part A are marked according to the above Part A marking guidelines. The additions you provide in completion of the Part B tasks will be marked according to this guideline:

Please refer to next page for marking structure.

Table I: Project B marking Guide

Heading	**Report**
Part B Introduction	Provide a short introduction ($\sim \frac{1}{2}$ page) to your Part B of the project explaining your main design choices and what you have provided for the project additions (e.g. if using GUI tools or single text-based menu). **[15 marks]**
Requirements & GUI implementation plans	The requirements section should provide a description and diagram for the Part B additions. That can be an extended version of the diagrams you provided for this aspect in Part A. Provide accompanying text to clarify these requirements/additions. 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) describing the main operation and structuring of your Part B implementation additions (e.g. code modules/classes you may be using). You don't need to provide fine detail of the system, the diagram(s) can be e.g. at the level of functions. **[20 marks]**
Implementation	This section should give some snippets of important code and explanations for these in relation to your Part B additions (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 and usability of the host-based aspect of the system (e.g. how long does your system take to update the menu, respond to a menu selection, log data, etc). A snapshot / video (e.g. using smartphone as mentioned above) could be included as a demo and demonstration for the performance and validation testing. **[20 marks]**
Conclusion	Give a summary of the extent that the Part B additions of your 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 additional references if relevant. (make sure you indicate which references were added in relation to Part B aspects).
Total	100

end of assignment description