UART

Exercise 1 – Read version info and DevEui from LoRa module

Implement a program that verifies that UART connection to LoRa module works and then reads and processes EUID from the module. The program should work as follows:

1. Program waits for user to press SW_0. When user presses the button then program starts communication with the LoRa module.
2. Program sends command "AT" to module and waits for response for 500 ms. If no response is received or the response is not correct the program tries again up to five times. If no response is received after five attempts program prints "module not responding" and goes back to step 1. If response is received program prints "Connected to LoRa module".
3. Program reads firmware version of the module and prints the result. If no response is received in 500 ms program prints "Module stopped responding" and goes back to step 1.
4. Program reads DevEui from the device. If no response is received in 500 ms program prints "Module stopped responding" and goes back to step 1. DevEui contains 8 bytes that the module outputs in hexadecimal separated by colons. The program must remove the colons between the bytes and convert the hexadecimal digits to lower case.
5. Go to step 1

See page 10 in LoRa-E5 AT Command Specification_V1.0.pdf in the Oma workspace/Documents/Project for information about the structure of the commands. See page 24 for details of the commands needed in this assignment.

Processing of DevEui is needed for registering the module to Metropolia LoRaWAN network. The server needs the DevEui in the specified above format to register the device to the network. In the project you need you register your device LoRaWAN to send data.

For example:

- Module sends DevEui:
  ```
  +ID: DevEui, 2C:F7:F1:20:32:30:A5:70
  ```
- Program prints:
  ```
  2cf7f1203230a570
  ```

See hints on the next page.

Hints:

First implement a function that reads a string from UART with a timeout. The function should wait until linefeed is received or timeout occurs. See implementation of `uart_is_readable_within_us` in PicoSDK for the principle of implementing a timeout.

You can test that your software detects non-responding LoRa-module by running it with the module disconnected.

Steps described in the assignment can also be considered states of a state machine. For example, the state machine could be implemented as shown below:

```
switch(state) {
    case 1:
        // do what needs to be done in state1
        // to change state assign a new value to 'state'
        break;
    case 2:
        break;
    case 3:
        break;
    case 4:
        break;
}
```