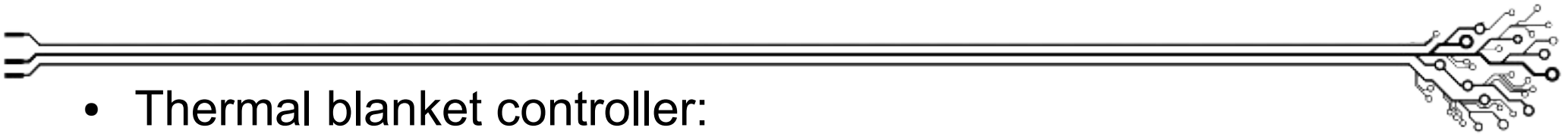
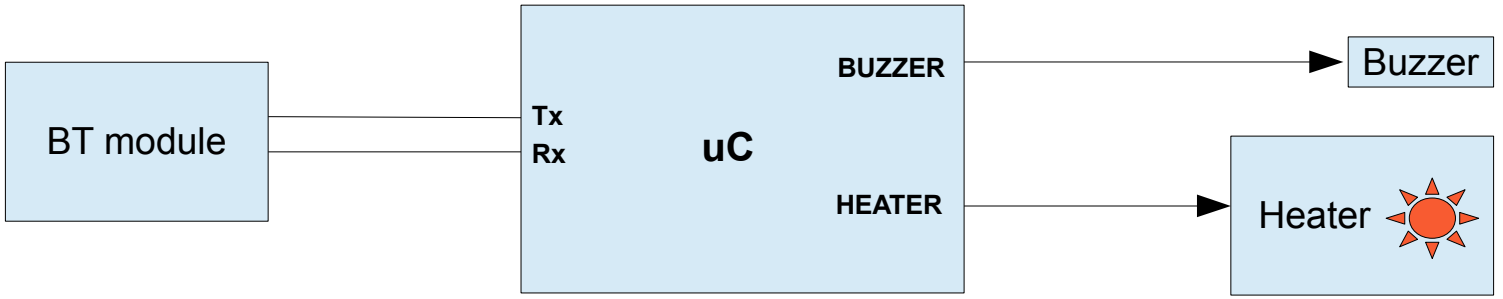
C4µC - Examination



- Thermal blanket controller:
  - You are asked to design a uC circuit to control a thermal blanket with the functionalities of a timer (auto-shutdown) and heat level regulation.
    The circuit will have the uC connected to a Bluetooth module (via UART), to a buzzer and to the output for the heater
  - The functionalities should be as following:
    - The circuit, when first powered, is in standby (STDBY) and the uC sends the string "OFF\n" every second on the UART (to the BT module)
    - When on the UART is received the string "ON\n", the output to the heater (HEATER) is activated and default system values are set and then sent via UART every second according to the following format: "Lx\n" and "TIMEmm:ss\n" where x is the power level applied on the heater [1-9] and mm are minutes, ss are seconds;
      - of course the level is the one currently applied and the time is decreasing,
      - the two messages are sent in sequence
      - the default system values are 30 minutes and a heat level of 5 (first init)
    - 10 seconds before the time is over (<=10), the buzzer (BUZZER) is activated
    - When the time is over (==0), everything is off and the system is in standby again
    - The timer can be increased by 30 minutes if the message "+TIME\n" is received on the UART
    - The heat level can be changed if a message "Lx\n" is received, with x the new level [1,9]
    - If the message "OFF\n" is received, everything is stopped and the system goes to stdby

C4µC - Examination

- The system is connected as in the following scheme:



- Recap of possible messages:

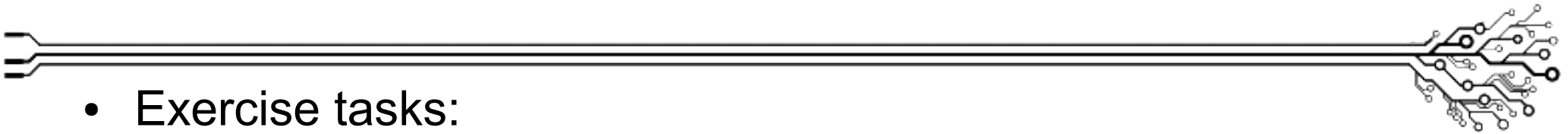
BT to uC	Notes
ON\n	Turn ON
+TIME\n	Increase 30'
Lx\n	Set level x
OFF\n	Turn OFF

uC to BT	Notes
OFF\n	Every sec if OFF
Lx\n	Every sec if ON
TIMEmm:ss\n	Every sec if ON

"\n" is line terminator

- The heat level is applied calling the given function:
  - **`heater_apply(uint8_t level)`** with level ϵ [1, 9]

C4µC - Examination



- Exercise tasks:
  1) Define inputs and outputs for the reference µC 328p [1 pt.]
  2) Draw the state-chart for the system [2 pt.]
  3) To implement all the functionalities described, write down the code for the init function (e.g. s*etup()*), the main function (or *loop()*) and other functions, if any, declaring all the needed variables [7 pt.]
  4) Implement the function **`heater_apply(...)`**, that uses a variable dutycycle to the output HEATER [2 pt.]:
     - Level 1 corresponds to 100ms HIGH and 900ms LOW
     - Level 2 corresponds to 200ms HIGH and 800ms LOW
     - And so on till level 9, that corresponds to 900ms HIGH and 100ms LOW
- General note:
  - Write any comment to justify a choice when the system requirements leave a degree of uncertainty
  - If any block of code is taken from internet, mandatory cite the source (link in comments)