Guidelines for your Hardware Abstract Layer

Context

- For all the semester, you have to make a team of 2 to 3 people. The goal of the semester is to build your own HAL. You will add the different features over this semester.
- Your code is in a GitHub repository (for all the HAL feature). You will add “felicityDorleans” as a collaborator, in order to assess your work.
- Your project is in Rust, and abstract the Atmega328p (Arduino Uno) and another target, as Tensilica LX106 (Esp8266), or Cortex-M7 (Teensy 4.1, STM32F7 series). You have to use the Atmega328p as your first target, but you are free of your choice for the second one.
    - Your HAL allows the user to use both target without knowing their technical specification, it hides all the registers and hardware details.
    - You can build your HAL the way you want. For example, there are multiple ways to adapt a code for multiple target.
    - You have to add a function (in your main.rs file) that act as an example. In this function, you will show quickly how to use your HAL. You will update this function throughout the semester (for the GPIO feature, you can show how to read the pin 2 for example).
- About the evaluation:
    - Each feature is assessed over the semester. You have two weeks after your lab to deliver the new HAL feature. Each delivery is assessed only with the new feature, therefore, all commits past the deadline won’t be assessed (each time). You can ask me to evaluate a project that is submitted past the deadline, but a penalty will be applied.
    - Your project (all the HAL features at once) will also be evaluate globally at the end of the semester (two weeks after your last lab). Then, if you commits on one HAL feature after their deadline, it can still be appreciate.

I. The GPIO feature

- Your GPIO feature have to abstract the Atmega328p (Arduino Uno). Other target (f.e. the Tensilica LX106 (Esp8266)) is not mandatory for now.
- Your code allow to control all the digital pin of the target(s), as input or output. More precisely, it can:
    - Configure any digital pin as input, or as output.
    - Read and write all digital pins.
    - If you want (but is not mandatory), you can abstract the pull-up resistor feature (Atmega328p and Tensilica LX106 targets support this option) and the pull-down resistor feature (the Atmega328p doesn’t support this option).

II. The USART feature

- The USART feature abstract the Atmega328p and another target (as Cortex-M7 f.e). You have to use the Atmega328p as your first target, but you are free of your choice for the second one.
- You can use the Transmitter channel and the Receiving channel of your target.
- Your project is well organized, which mean that it contains a Cargo.toml file, a src folder with multiple module (== don't put all your code in your main.rs file (or lib.rs file)).
- You need to be in a group of 2-3 people (for all exceptions, you must send me an email).
- Idea of application (bonus point) :
    - You can emulate two MCU communicate, thanks to their USART feature (even more bonus if you manage to make communicate two different MCU).