GPIO challenge

Semester 3 Embedded Systems

1 Introduction

1.1 Note for Bachelor / Associate degree students

This challenge is for both bachelor (BA) as associate degree (AD) students. BA students in general will start with research which will eventually result in a complete design. AD students do not have to carry out / document the research steps which would normally lead to a global design. Instead AD students will be given a global design. AD students will still have to fill in the detailed design of the separate components (UML).
In this document the phrases "**BA only**" or "**AD only**" will be used to distinguish between the two.

1.2 Background

Embedded systems are often connected to the outside world through available in- and output pins, or GPIO. This way sensors and actuators can be connected. To enable an user to interact with an embedded system it is possible to connect a Manual Control Panel (MCP) containing buttons or a rotary encoder and LEDs or similar. Due to housing limitations and user friendliness it is often desired to limit the number of buttons and LEDs. To still let the user control a certain amount of functions the buttons can be pressed in different ways, e.g. a short press, long press or quick double press. To be able to operate the MCP on batteries, low power design is important.

1.3 Requirements

In the remainder of the challenge we will use the acronym MCP to address the Manual Control Panel.
The MCP described here is only a proposal and you are challenged to create your own MCP! In this case consult your teacher to check if the learning goals can be covered with your MCP.
The proposed MCP contains two buttons and two LEDs. A short press has a duration between 20 and 500 ms. A long press has a duration longer than 500 ms.
Instead of the buttons one can also use a rotary encoder or another input control.
The letters in front of the requirements refer to the MoSCoW method.

1.3.1 Functional

1. **BA Only:** (M) Short and long presses on the buttons or left and right turns of the rotary encoder must be distinguishable by the LED lighting pattern.
2. **BA Only:** (M) Short and long presses on the buttons or left and right turns of the rotary encoder must be distinguishable on the serial monitor.
3. **AD Only:** (M) A press on a button must toggle the corresponding LED.
4. **AD Only:** (M) A press on a button must be visible on the serial monitor.

1.3.2 Non-functional

1. (M) The MCP must function properly with buttons or a rotary encoder with a bouncing time of less than 20 ms. When using capacitors this requirement is fulfilled.
2. (M) Button presses or rotary encoder turns must never be missed i.e. detecting presses or turns must be done without polling.
3. (M) Your solution must be scalable i.e. adding more buttons or LEDs should be easy.
4. (S) When no button is pressed or the rotary encoder is not turned, the power consumption of the MCP must be minimal.

2 Challenge

Guideline: ± 4 days of work.

2.1 Challenge details

- You can use the challenge as described in the previous section, or a challenge of similar complexity you created yourself (please consult your teacher beforehand).
- **BA only**: Start the challenge by carrying out research. Investigate what is needed to implement the requirements of the challenge. Motivate the choices you make. This includes algorithms, needed internal MCU components, configurations and so on. Carry out the research by using the DOT framework strategies and methods, see http://ictresearchmethods.nl/Main_Page.
- **AD Only**: In the appendix A below a global design is given which you can follow.
- Design all the flowcharts (C) or UML (C++) diagrams needed for the operation of the MCP.
- Conduct POC's (Proof of concepts) which handle the configuration of the needed internal components of the MCU. POC's are experiments which you carry out to test critical parts of your design. Please document these POC's and use pictures / diagrams to clarify your setup and measurements. Note that a POC can be seen as a research method.
- Write and integrate all necessary code such that all requirements are implemented. Assess, using tests or testcases, that all requirements are implemented properly.
- Write a report, see details below.

2.2 Additional information

- Make use of CMSIS for the implementation.

2.3 Hints

- To measure a time delay you may use the HAL_GetTick() function to get the elapsed time in ms.
- As MCP a Danger Shield can be used. Make sure it is the right version (with the Cap Sensor and not with the Knock sensor). In the Toolbox section of the Canvas course the circuit diagram of this board can be found in 'Danger_Shield-v16.pdf'. The buttons and LEDs on this Danger Shield are only connected to GND (and not to 5V) so it can be safely used with the 3.3V STM boards.
- Achieving really low power consumption can be challenging. Note that you have a timer tick interrupt (used by HAL_GetTick()) which will wake up the MCU every 1 ms. For this assignment showing that you can achieve (some) power reduction is sufficient.
- Although the Nucleo board is Arduino shield compatible, the analog pins used by the Danger Shield are not 5V tolerant. So do not plug the Danger Shield into the NUCLEO board but connect it with jumper wires.

3 Hand in to Canvas

Important note:
The grading for this challenge will for a great deal be based on your documentation. The reader should be able to verify the design process and the design choices based on objective arguments and measurements. Use pictures and diagrams to clarify your setup / measurements and design. Fully functional code without the required documentation has not much value.

Please hand in:

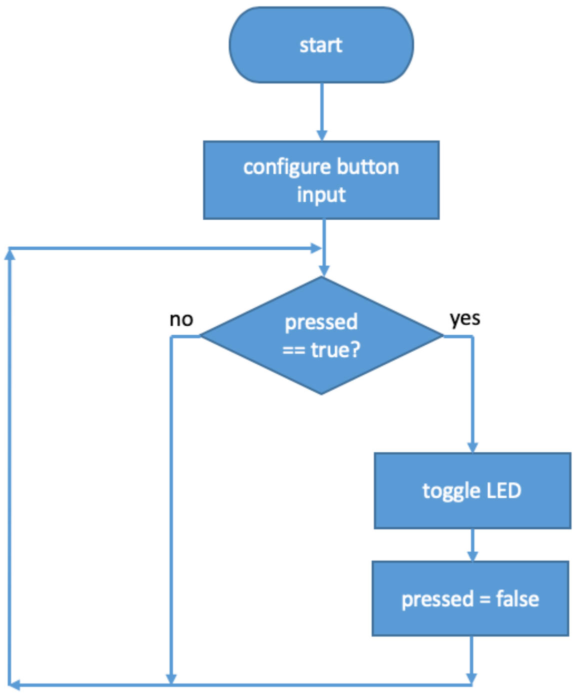
1. All code.
2. Proper documentation consisting of:
    a. Title page with date and name.
    b. Short introduction.
    c. **BA only:** Your research: Show all relevant research steps (DOT framework) and elaborate on the choices made.
    d. Your design: all diagrams with accompanying texts and research.
    e. How you tested your implementation including proof (how you validated the results and so on).
    f. Reflection on what you have learned.
    g. Bibliography (sources).

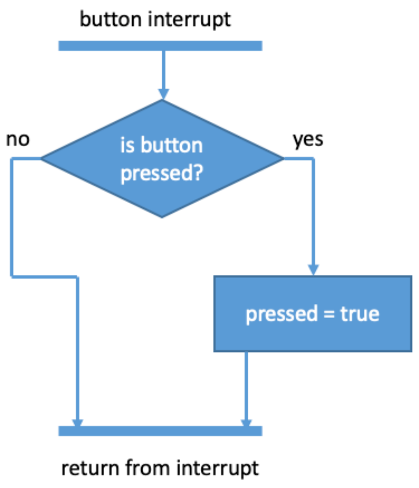
Appendix A: Design proposal for AD

Below a global design proposal is given for one button and one LED. When pressing the button the LED will turn on. When pressing again the LED will turn off. In the interrupt handler the button state is read (polled) and stored in global accessible variable or similar. This variable can be used in the main program.

Notes

- The global design is at a high abstraction level and the implementation may vary.
- Only one button and LED is depicted, but you need to implement at least two buttons and two LEDs or similar.
- Having global variables accessible from all over your code is not a good idea and goes against the scaling requirement. Try to use some form of encapsulation.
- If you want you can deviate from the design, but please document the arguments.


start
configure button input
no
pressed == true?
yes
toggle LED
pressed = false


button interrupt
no
is button pressed?
yes
pressed = true
return from interrupt