SLP-assignment #5: traffic light

(15 points, groups of two)

Implement a traffic-light system controller with a pedestrian passing in the file `ampel.c`. Normally, the lights for the cars should be green and red for the pedestrians. If a pedestrian wants to cross the street, they can request a switch by pressing a button. After the request, the traffic light for the cars is set to red step by step while the pedestrian lights switch to green. The pedestrian has some time to cross the street before the controller switches back to its normal mode of operation.

This assignment can be divided into two parts: Part a) implements the basic traffic-light control and part b) introduces an error state for the system. If the cause of the error is no longer present, the controller should – without endangering pedestrians – switch back to its normal state. In the following, both parts are described in detail.

Part a: Traffic Light Controller (11 points)

The traffic lights for the cars is represented by the LEDs `RED0`, `YELLOW0` and `GREEN0`, the pedestrian lights by `RED1` and `GREEN1` (there is no yellow for pedestrians). By pressing `BUTTON0`, a pedestrian can request a switch of lights. The LED `BLUE1` signals that a request has been accepted.

In detail, the controller should work as follows:

- A request is triggered by pressing `BUTTON0`. By switching on the LED `BLUE1`, the request is accepted ("signal is coming soon"). This LED gets switched off as soon as the pedestrian light is set to green. Further activations of `BUTTON0` are ignored until the pedestrian light shows red again.
- After a successful request in the normal mode (cars: green, pedestrians: red), the traffic-light systems counts down 8 seconds on the 7-segment display to indicate, how long the pedestrian has to wait; in all other cases, the 7-segment display is switched off. For the first 5 seconds of the countdown, there is no change of any lights, then the light for the cars changes to yellow for one second before turning red. After two more seconds with both lights showing red, the lights for the pedestrians changes to green.
- The pedestrian light should be green for exactly 5 seconds, then it should switch back to red. After another second, the traffic lights for the cars show red-yellow and after yet another second it switches to green, the normal mode.
- Initially, both lights should be set to red and the traffic light for the cars switches to red-yellow and then to green in the same speed as above while the pedestrian light remains red (however, requests can already be accepted).
- A request can be accepted as soon as the pedestrian light is set to red. This means that request can be accepted even before the traffic light for the cars shows red-yellow (i.e., both lights are red) – while the LED `BLUE1` immediately signals the accepted request, the countdown will not start before the traffic light for the cars shows green.

Make sure that the microcontroller enters a sleep mode whenever no calculations are required. This can be done using functions from `avr/sleep.h`.

Always remember the correct usage of the `volatile`-keyword. For each declaration of a `volatile` variable, add a comment explaining why the keyword is needed there.

Part b: Error State (4 points)

The traffic light should be extended to include an error state (light for the cars blinks yellow, pedestrian light and 7-segment display are switched off). As an indicator for an error, the external interface `EXT` should be used. Since it is wired to the same pin as `BUTTON1`, the button can be used to test the error state.

- During low level (`PD3` connected to `GND`), corresponding to `BUTTON1` being pressed, the error state is activate.
- The blinking period is one second. Between switching the indicator light on and off, the CPU has to enter a sleep mode as always.
- The error state is ended as soon as a high level is present at the pin. To prevent any accidents, the controller should initialize the traffic light with red for both lights and then switch to green as explained before.
- The edge case, where the error state is present at the start of the traffic systems, can be ignored.

Hints:

- Use the modules `led` and `7seg` of the `libspicboard` for all outputs.
- However, do *not* use the `button` and `timer` module of the `libspicboard`!
  - Instead, you have to configure the interrupt handling and -handler for `BUTTON0` and `BUTTON1` directly. They are wired to pins `PD2` (`BUTTON0`) and `PD3` (`BUTTON1`) respectively and therefore connected to the external interrupt sources `INT0` and `INT1` of the ATmega. However, consider that the interrupt detection could be different for these two buttons.
  - For the timing, you should use `TIMER0`. You may use the overflow interruption `OVF` (errors up to 50ms can be tolerated). Choose the most resource efficient `prescaler`. When switching from and to the error state, waiting phases of less then 1s are tolerated (the timer does not have to be reset).
- Design your program in such way that `main()` implement only once the logic for entering a sleeping state. In particular, do not call or implement the sleep state for each change of state individually.
- Always give a reason why you use the `volatile` keyword. If the same reasoning holds for multiple variables, you can justify them together.
- In the directory `/proj/i4spic/pub/aufgabe5/`, you can find the file `ampel.elf` which contains a reference implementation.

Deadline

T01	16.06.2024	18:00:00
T02	16.06.2024	18:00:00
T03	17.06.2024	18:00:00
T04	18.06.2024	18:00:00
T05	18.06.2024	18:00:00
T06	19.06.2024	18:00:00
T07	19.06.2024	18:00:00
T08	20.06.2024	18:00:00
T09	17.06.2024	18:00:00