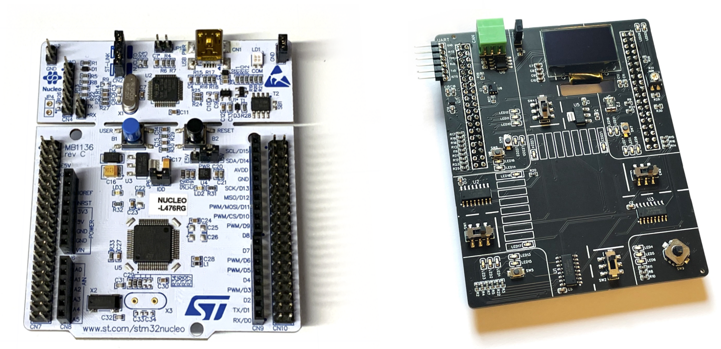
Matthias Becker December 2, 2024

Project Task IS1300 HT24

We us the traffic light shield together with the Nucleo-L476RG development board as hardware platform of the project task (see Fig. 1). No additional hardware (except the USB-cable) is needed. Serial communication to the PC (if needed) can be realized via UART2 which is connected to the ST-Link debug-module and accessible as virtual com-port on your PC.
The project is designed in a way that you have flexibility to select which of the provided implementation tasks you want to realize. You must implement all 3 of proposed implementation tasks to get full points for the project complexity. A detailed explanation is provided further down below.



(a) Nucleo-L476RG (b) Traffic Light Shield

Figure 1: Main hardware used for the project

Project Timeline

The project is to be completed over a period of +5 weeks. During this time period, you are responsible *yourself* to allocate time for the project. The project counts for 4.5 CP which translates to approximately 120 hours of work. Please make sure to devote sufficient time.
Lab sessions are useful to ask questions and get direct feedback from course staff. During the project time we will have 3 lab sessions (3 x 4h each) when you can work on the project and get support from us where needed. Please utilize those occasions.
The project concludes with the submission of source code, documentation and written report. Submission deadlines are listed in the table below. There will be no dedicated demo at the end of the project, we will test your submissions offline.

Submission Type	**Deadline**
Source Code	20. December 2024
Source Code Documentation	20. December 2024
Written Report	20. December 2024

Overview Implementation Tasks

Here we describe the different implementation tasks that you can select for the project. For each implementation task a short description is provided followed by a list of requirements. The short description is intended to introduce the task while the requirements are what the project client expects. All implementation tasks that you select shall be implemented *together* as part of the *same* project. This means if, for example, implementation task 1 and implementation task 2 are realized the final program meets all requirements of the implementation tasks together.

Implementation Task (1)

This implementation task focuses on the upper pedestrian crossing of the traffic light shield. Only the traffic lights shown in Fig. 4 are considered in this task. Cars are simulated by the respective switches on the road.

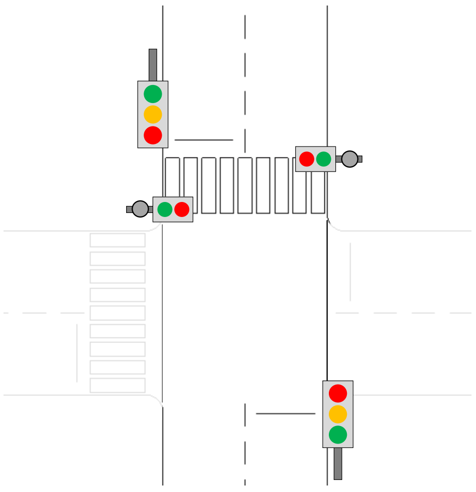


Figure 2: Sketch of the active components in implementation task 1.

The concrete requirements of this implementation task are as follows:

R1.1 At initialization, the pedestrian crossing shall be red and the car signal green.

R1.2 After a user pushed the button of the crosswalk, the indicator light shall toggle with a frequency of `toggleFreq` ms until the pedestrian crossing is green.

R1.3 All car signals of each active lane that cross the crosswalk shall be red `pedestrianDelay` ms after the pedestrian pushed the button.

R1.4 The pedestrian signal stays green for `walkingDelay` ms.

R1.5 The pedestrian signal must be red when the car signal of each active lane that crosses the crosswalk is either green or orange, and green otherwise.

R1.6 A car signal transitions from red to orange to green, or from green to orange to red, remaining orange for `orangeDelay` ms.

Implementation Task 2

This implementation task focuses on the road crossing of the traffic light shield, ignoring the pedestrian crossings. Cars are simulated by the switches on the road. For simplicity, we assume a car is only allowed to drive forward to take a right turn.

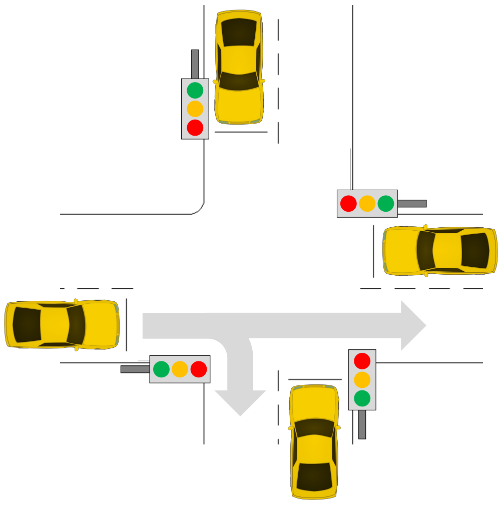


Figure 3: Sketch of the active components in implementation task 1.

The concrete requirements of this implementation task are as follows:

R2.1 Cars from each direction can either go forward, or turn right. Left turns are not allowed.

R2.2 Traffic lights must be set in a way that no two cars have overlapping possible paths.

R2.3 A car signal transitions from red to orange to green, or from green to orange to red, remaining orange for `orangeDelay` ms.

R2.4 If there is no active car in any direction the allowed (i.e. green) direction changes every `greenDelay` ms.

R2.5 A traffic light remains green if there are active cars in either allowed direction and no active cars are waiting on red traffic lights.

R2.6 If a car arrives at a red light and there are active cars in either allowed direction it waits `redDelayMax` ms until the signal has changed to green.

R2.7 If a car arrives at a red light and there are no active cars in either allowed direction the signal transitions immediately to green.

R2.8 At initialization the vertical lane shall be green and the horizontal lane shall be red.

Implementation Task (3)

This implementation task combines the results of implementation task (1) and implementation task (2) to manage the complete traffic crossing, including both pedestrian crossings. Cars are simulated by the switches on the road. A modular design of the previous implementation tasks will allow for an easy integration of the different parts. In addition to the control of the traffic lights, this task requires to control the shift registers using the SPI interface.

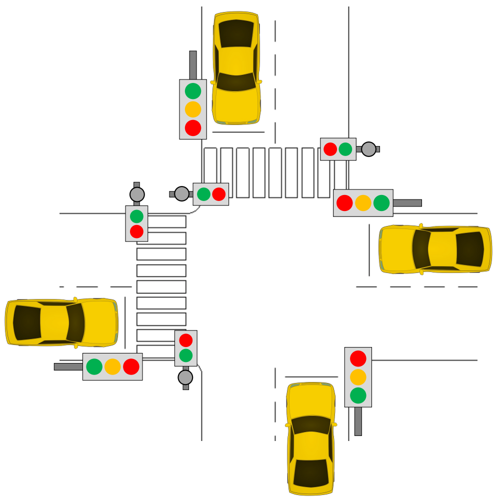


Figure 4: Sketch of the active components in implementation task 3.

The concrete requirements of this implementation task are as follows:

R3.1 Requirements of Task 1 for each crossing.

R3.2 Requirements of Task 2 for car crossing.

R3.3 Only one pedestrian crossing is green at a time.

R3.4 Shift register shall be controlled with the SPI interface of the microcontroller.

R3.5 A car is allowed to turn right when a crossing on the right lane is green.

General requirements

R0.1 The name of your STM32CubeIDE project name must be set based on your name in Canvas. White space is replaced by '_' and 'PRO1_' is added at the beginning. Example: Canvas name 'Matthias Becker' results in the project name 'PRO1_Matthias_Becker'. This is a hard requirement as parts of the evaluation process are automated. Different names will lead to a failed grade.

R0.2 In order to pass PRO1 each evaluation category must reach at least 1 point.

R0.3 In order to pass PRO1, the following documents must be submitted

- Written project report
- Source Code of the project
- Compiled binary of the project

Points for the Project and Mapping to Grades

The following describes how the points for the project PRO1 are determined. Table 1 gives an overview of all evaluated categories that give points and Table 2 gives the mapping of points to grades. Besides the extra point from the exam, you need *at least 1 point* from each category to pass PRO1. The following discusses each evaluation category in more detail.

Table 1: Points for the project

Category	**Points**
Planning, Architecture, Structure and Testing	0, 1, 2
Implementation and Complexity	0, 1, 2, 3
Report and Documentation	0, 1, 2
Real-Time Tasks	0, 1, 2, 3
Extra possible point from the exam	0, 1

Table 2: Grades Map

Grade	**Points**
A	11
B	9 - 10
C	7 - 8
D	5 - 6
E	4

Submissions that are *within the deadline and reach a passing mark* (in all categories) receive +1 point.

Planning, Architecture, Structure, Testing [0, 1, 2]

For full points, a well thought trough test strategy is needed. Tests shall not only test the basic case but be able to detect corner cases as well. Functions that are tested shall be the same that are also used in the project. All tests shall be executed from a dedicated file (similar to the PingPong lab) and be included in the final submission of the source code. Testing shall follow the Test Driven Development approach. (**1 Point**)
The hardware as well as software architecture shall be documented with adequate diagrams and described in text in the report. (**1 Point**)
Both parts must be addressed to receive at least one point!

Points for Implementation and Complexity [0, 1, 2, 3]

The points obtained for program complexity are a result of the implementation tasks that you select and complete. Please be aware that not all combinations of implementation tasks are possible. For example, to obtain one point for program complexity you can select either implementation task ① or ②. For two points you are required to complete implementation task ① and ②. And for full points you complete task ① and task ② and task ③. A description for all points is provided in the table below.

Nr. of Points	Required Impl. Tasks
1 Point	① OR ②
2 Point	① AND ②
3 Point	① AND ② AND ③

Report and Documentation [0, 1, 2]

The report shall be well written, well structured and contain relevant and correct references. See the page on '*Report Writing*' in the Canvas module '*Project*' for detailed information on expected sections and content. It is *expected* that the report contains different diagrams (see also 'Planning, Architecture, Structure, Testing'). (**1 Point**)
The interface between different code modules shall be described in the report and the code must be commented sufficiently. (**1 Point**)

Real-Time Tasks [0, 1, 2, 3]

The completion pf the RTOS-Lab is mandatory to complete this part. (**1 Point**)
For 2 or 3 points, the RTOS lab must be completed and additionally FreeRTOS shall be used to solve the project tasks. A proposal on how you plan to use the RTOS in the project shall be prepared and discussed with the teacher *before* you start implementing. Depending on the selected implementation form we will decide at this time if one or two additional points will be awarded for this grading category (if completed).

Code of Honour

The project is to be completed individually, group work is not allowed. For the project we follow the KTH Code of Honour[1].
Each student writes own code. Submission of the same code from several students is not accepted. Plagiarism of code may be subject to notification to the disciplinary committee.

[1] `https://www.kth.se/en/eecs/utbildning/hederskodex/inledning-1.17237`