Exercise 10: Distributed Applications

Note: This exercise spans two sessions.

Overview

Until now, you have developed quite a large number of programs that were focused on reacting to human input, solving a specific task locally on the controller and producing the respective output. In this exercise, you will make your microcontroller communicate with the controllers of other teams to build a distributed application. The application that we will build is a traffic light controller that manages an intersection of a street (traffic light 1, TL1) and a pedestrian walkway (traffic light 2, TL2) (compare Figure 1 on page 7).

The exercise will be solved according to the following steps:

1. Every group implements the basic communication facilities using the microcontroller's two-wire interface (TWI), also know as I^2C. TWI is a rather complex, but powerful communication bus used in many applications.
2. Every group tests their communication facility with their neighbor group(s).
3. Four groups build a team. This means that we will have two teams in total. The groups of a team each implement exclusively *one* of the following functionalities:
   A) DCF77-based Time Server unit.
   B) State machine based Traffic Light Control Algorithm unit.
   C) Traffic Light Control unit.
   D) Traffic Light Fault Detection and Emergency Control unit.

The following terms will be used across this exercise:

- **Day mode:** This is the normal operation mode of the two traffic lights, in which TL1 and TL2 periodically switch between the red and green state. One period is shown in Figure 2(a) on page 7.
- **Night mode:** During night, the probability of pedestrians crossing the street is very low. This is why in night mode, TL1 is usually green unless there is a request from pedestrians waiting on TL2. If there is a request present, TL2 should become green for a short amount of time and then turn red again as shown in Figure 2(b) on page 7.

Introduction to TWI / I^2C

Exercise 10.1

Read section 21 of the ATmega168 manual and answer the following questions:

a) TWI is a Master/Slave based protocol. Explain the roles of the communication partners. In which four modes can any communication partner be at any time? What is their meaning?

b) Which pins are used for the TWI?

c) How is a specific device addressed? How many addresses are available? What is the meaning of the acknowledge bit while sending the address?

d) How are data transferred over the bus? What is the meaning of the acknowledge bit during data transmission?

e) Which interrupt is involved and when is it executed? What should the interrupt service routine do with the `TWI` flag in order to continue transmission?

f) How are multiple bytes submitted in a row?

Exercise 10.2

For the upcoming exercises, we assume the following TWI parameters:

- Use an SCL frequency of 10 kHz.
- The Master (not the Slave) should enable the internal pull-ups of the SCL and SDA pins.

Implement an application that initializes the TWI in Master Transmitter or Slave Receiver Mode (with slave address `0x01`) depending on a flag.

While in Master Transmitter mode, ask the user to enter a question on the serial console and store each character and in a buffer (maximum length 255 characters) until a newline character is received. Then, transmit the whole buffer including the newline character to the communication partner and switch to Master Receiver mode, waiting for a response. When receiving data, put them into the buffer until the buffer is full or you receive a newline character. Then, and print the buffer to the console after the last character has been received and switch back to Master Transmitter mode.

While in Slave Receiver mode, wait for characters being sent by the Master and store them in a buffer (maximum length 255 bytes) when they arrive. Print the buffer to the console after the last character has been received and switch to Master Transmitter mode. Wait for the user to enter a response and store each character in the buffer until a newline character is received. Then, transmit the whole buffer including the newline character to the communication partner and switch back to Slave Receiver mode.

Verify your program by asking your neighbor group(s) a question and let them answer and vice versa.

Exercise 10.3

Form teams of four groups. Make sure that all groups are at the same stage of implementation (i.e., they have solved exercise 10.2). If this is not yet the case, support your fellow groups so they can catch up.

Solve one of the following four exercises (named Part A, B, C or D), depending on your role within your team.

Figure 3 on page 7 shows the communication infrastructure required for solving the exercise.

The Traffic Light Circuit is a printed circuit board that shows the state of the respective traffic lights and also allows to simulate faults by switching on any of the traffic lights at arbitrary points in time. The layout of the board is shown in Figure 4 on page 8. The board allows to attach all four boards to a common TWI bus (headers `TWI_A`, `TWI_B`, `TWI_C` and `TWI_D`). For bus participant A and B, an additional `GND` (ground) header is provided. The ground lines for participants C and D are integrated in the `CONTROL_C` and `MONITOR_D` headers, respectively. The `CONTROL_C` header allows the Traffic Light Control Unit to set the state of each traffic light, while the `MONITOR_D` header allows the Fault Detection and Emergency Control Unit to sense the state of the traffic lights (for pinout, see Tables 1 and 2, respectively). Furthermore, the switches on the board can be used to perform the following actions:

- **Sim. TL2:** Switch on one of the red (R), yellow (Y, ignore for this exercise) or green (G) lights of TL2.

Listing 1: Time Information exchanged between Time Server and Control Algorithm

```
typedef struct
{
    uint8_t sec; // Current second (0..59)
    uint8_t min; // Current minute (0..59)
    uint8_t hour; // Current hour (0..23)
    uint8_t isDay; // Whether it is currently day (1) or night (0)
}
timePacket_t;
```

- **Sim. TL1:** Switch on one of the red (R), yellow (Y) or green (G) lights of TL2.
- **Request:** Simulate a request by a pedestrian (1) or car (2, ignore for this exercise).

Part A: DCF77-based Time Server (TWI Slave `0x01`)

Create a new project and modify your solution from exercise 6 (DCF77 decoder) according to the following requirements:

- Initialize TWI so that your program runs in TWI Slave Transmitter mode according to section 21.7.4 of the ATmega168 manual. Use TWI address `0x01`. Your program will serve time requests issued by the traffic light control algorithm.
- Whenever you receive a time request, send a packet with the structure shown in Listing 1 (4 bytes). The `isDay` member of the `timePacket_t` structure indicates whether the traffic light should be running in day or night mode.
- Since we can't wait for day/night cycles to arrive, you should implement your program to toggle the day/night flag on the press of a button.
- Furthermore, to speed up testing, make sure that you transmit the correct value for the `sec` member as soon as it is available, i.e., after your program has seen the first minute mark.
- Show all available information (second, minute, hour, day/night flag) on the LCD.

Part B: State Machine based Control Algorithm (TWI Master)

Create an application that implements the following behavior:

- Initialize TWI so that your program runs partially in TWI Master Receiver and Master Transmitter mode according to sections 21.7.1 and 21.7.2 of the ATmega168 manual. Your program will query the time server (at TWI address `0x01`) for the current time and the traffic light controller for pending requests, and implement a state machine based control algorithm for the two traffic lights.
- Make sure that the traffic light conforms to the specifications in Figure 5 on page 8 at any point in time.
- Implement a state machine that represents the mentioned constraints.
- Periodically perform the following actions:
  - Retrieve the current time from the time server. Do this by acting in Master Receiver mode on slave address `0x01`. You will receive a data packet with 4 bytes that corresponds to the structure shown in Listing 1. The `isDay` member of the `timePacket_t` structure indicates whether the traffic light should be running in day or night mode.

Listing 2: Request State exchanged between Traffic Light Controller and Control Algorithm

```
typedef struct
{
    uint8_t reqTL1; // Whether a request for TL1 has been issued (not used).
    uint8_t reqTL2; // Whether a request for TL2 has been issued (0 or 1).
}
requestPacket_t;
```

Listing 3: Traffic Light State exchanged between Control Algorithm and Traffic Light Controller

```
typedef struct
{
    uint8_t stateTL1; // State of TL1 (bits 0, 1, 2 = red, yellow, green)
    uint8_t stateTL2; // State of TL2 (bits 0, 2 = red, green)
}
trafficLightPacket_t;
```

  - Retrieve the current request status from the traffic light controller. Do this by acting in Master Receiver mode on slave address `0x03`. You will receive a data packet with 1 byte that corresponds to the structure shown in Listing 2.
  - Calculate the desired traffic light phase, which can be determined by inspecting the timing diagram for the respective day/night mode (compare Figure 2 on page 7) at the time of the `second` member of the `timePacket_t` structure modulus 10.
  - If the traffic light state needs to be changed, perform a state transition to the new state if it is safe with respect to the above mentioned constraints.
  - In Master Transmitter mode, send the new traffic light state to apply to the slave with address `0x03` according to the structure in Listing 3. Note that for the `stateTL1` member, bits 0, 1 and 2 correspond to the red, yellow and green indicators and for the `stateTL2` member, bits 0 and 2 correspond to the red and green indicators.

- Show all available information (state, day/night mode, pending requests) on the LCD.

Part C: Traffic Light Controller (TWI Slave `0x03`)

Create an application that implements the following behavior:

- Reserve port B for sending the output signal to the traffic light and for retrieving requests. Table 1 shows the pin assignment, which corresponds to the `CONTROL_C` header on the traffic light circuit (compare Figure 4 on page 8).
- Initialize TWI so that your program runs partially in TWI Slave Receiver and Slave Transmitter mode according to sections 21.7.3 and 21.7.4 of the ATmega168 manual. Use TWI address `0x03`. Your program will receive new traffic light states from the traffic light control algorithm (TWI Master) and report whether any requests are currently pending.
- When receiving a new traffic light state from the master (2 bytes) according to Listing 3, apply the new state to the traffic light (with no additional sanity checks).
- Use interrupts to listen for requests on `PB6` and `PB7`. When the interrupt is triggered, remember that there has been a request for the respective traffic light. When receiving a query from the Master, send a packet containing the request status (2 bytes) according to Listing 2. Reset the request status after successful transmission to the Master.

Pin	Name	Direction	Purpose
1	PB0	output	TL1 red
2	PB1	output	TL1 yellow
3	PB2	output	TL1 green
4	PB3	output	TL2 red
5	PB4	output	TL2 yellow (not used)
6	PB5	output	TL2 green
7	PB6	input	request for TL1 (not used)
8	PB7	input	request for TL2
9	GND	supply	ground
10	VTG	supply	supply voltage (+5V)

Table 1: Pin Assignment of Port B on Control Unit (CONTROL_C header)

Pin	Name	Direction	Purpose
1	PB0	input	TL1 red
2	PB1	input/output	TL1 yellow
3	PB2	input	TL1 green
4	PB3	input	TL2 red
5	PB4	input/output	TL2 yellow (not used)
6	PB5	input	TL2 green
7	PB6	input	ignore
8	PB7	input	ignore
9	GND	supply	ground
10	-	-	not connected

Table 2: Pin Assignment of Port B on Fault Detection Unit (MONITOR_D header)

- Allow group D to reset your microcontroller by connecting pin C6 to a digital output pin of that group. In case of an error, that pin will be used to keep your microcontroller in reset.
- Print all available information to the debug console.

Part D: Fault Detection and Emergency Control System (no TWI)

Create an application that implements the following behavior:

- Reserve port B for sensing the state of the two traffic lights. Table 2 shows the pin assignment, which corresponds to the MONITOR_D header on the traffic light circuit (compare Figure 4 on page 8).
- Periodically check the state of the two traffic lights (by reading port B) according to the rules specified in Figure 5 on page 8.
- When a violation is detected, switch to a safety mode as follows:
  - Assert that the Traffic Light Controller is kept in reset by forcing pin PC6 of that controller low. For this purpose, connect a digital output pin from your controller to that pin and make sure it is normally at a high level so that group C's microcontroller can actually run its program.
  - Let the yellow light on TL1 blink at a frequency of 0.5 Hz to signal that the traffic light is out of order.
- Print all available information to the debug console.

Exercise 10.4 (optional)

Enhance your applications as follows:

Part A: DCF77-based Time Server (TWI Slave `0x01`)

Make sure your program can handle receving a TWI request which is canceled prematurely by the remote Master Receiver. The fault detection unit will use this mechanism to query your program whether it is still alive.

Furthermore, implement a button that, when pressed, will disable interrupts and go into an infinite loop to simulate a fault.

Part B: State Machine based Control Algorithm (TWI Master and Slave `0x02`)

In addition to existing functionality, your program should be able to run in Slave Transmitter mode with slave address `0x02`. Using this mechanism, your program should be able to be queried whether it is still alive by the fault detection unit.

Make sure your program can handle receving a TWI request from another Master and send a byte of dummy data in response.

Part C: Traffic Light Controller (TWI Slave `0x03`)

Make sure your program can handle receving a TWI request which is canceled prematurely by the remote Master Receiver. The fault detection unit will use this mechanism to query your program whether it is still alive.

Furthermore, implement a button that, when pressed, will ensure the traffic lights are set to a legal state, disable interrupts and go into an infinite loop to simulate a fault.

Part D: Fault Detection and Emergency Control System (TWI Master)

- Initialize TWI so that your program runs in TWI Master Receiver mode according to sections 21.7.2 of the ATmega168 manual.
- Your program should periodically check for the presence of the Time Server (slave address `0x01`), the Control Algorithm (slave address `0x02`) and the Traffic Light Controller (slave address `0x03`). Your program should terminate transmission by sending a NOT ACK as soon as it has determined that a slave is there.
- Your program should tolerate being arbitrated by another TWI Master.
- When a unit does not respond accordingly, switch to safety mode as specified in part D of exercise 10.3.

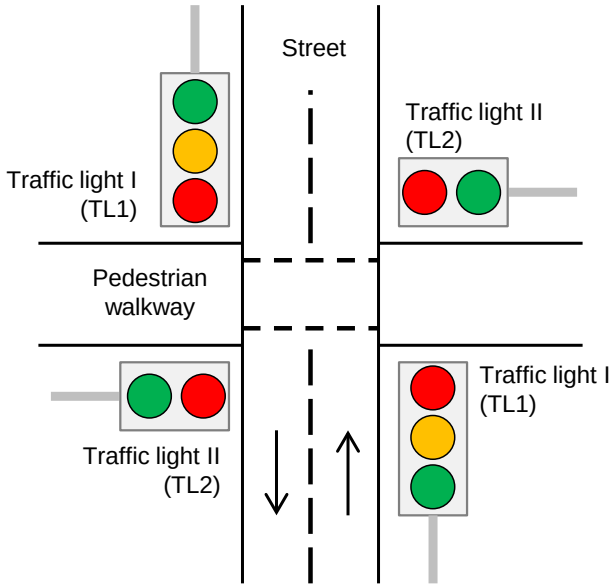

Figure 1: Traffic Light Setup

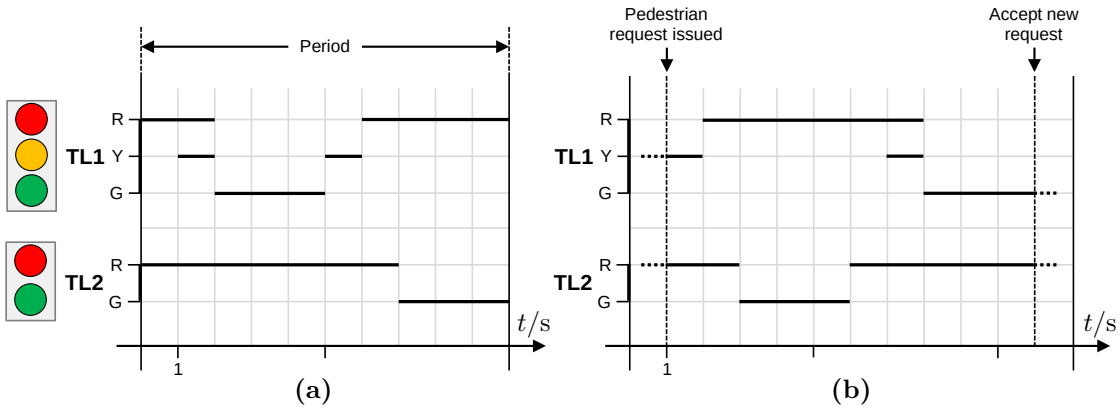

Figure 2: Traffic Light Timing Diagram; **(a)** Day Mode, **(b)** Night Mode

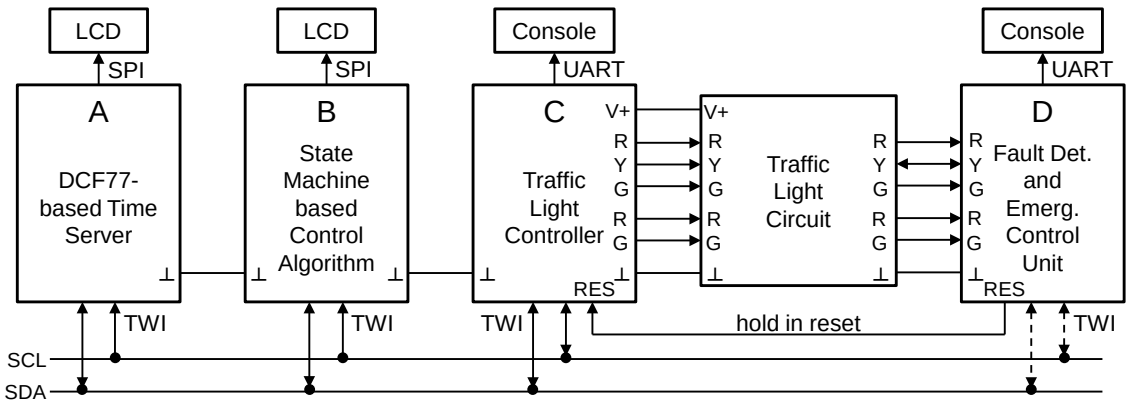

Figure 3: Communication Infrastructure

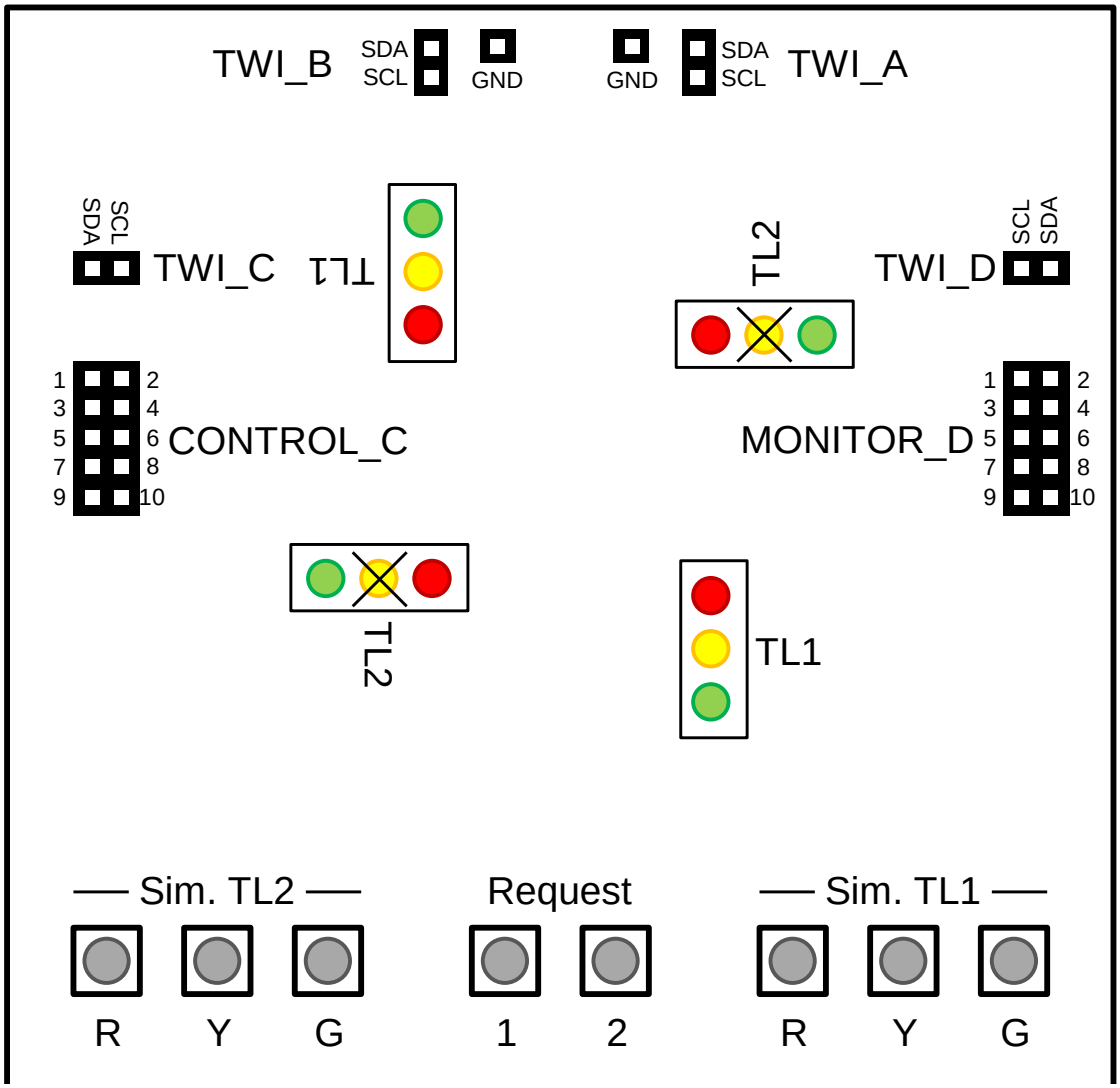

Figure 4: Traffic Light Circuit Board; see Tables 1 and 2 on Page 5 for Pinout of the `CONTROL_C` and `MONITOR_D` Headers.

- Each traffic light must show at least one of the colors at any time.
- Red and green must not be shown at the same time on any specific traffic light.
- Red and yellow must be shown together on TL1 during the transition from red to green.
- It is not allowed for TL1 to show yellow or green while TL2 shows green.
- The shortest possible green phase for TL1 and TL2 is 3 s. This must be guaranteed even if a state change occurs (e.g., from day mode to night mode) or when two pedestrian requests arrive in fast succession during night mode.

Figure 5: Traffic Light Specification