Automatic Control Laboratory, ETH Zürich
Profs. M. Morari, J. Lygeros

Manual prepared by: J.Schäfer, C.Fischer, A.Liniger
Revision: September 11, 2014

IfA Fachpraktikum - Experiment 1.6:

Traffic Control

In this experiment you will program a control algorithm for a traffic intersection. The controller will be implemented in Stateflow. Stateflow is a development tool where state diagrams can be created graphically and get compiled directly afterwards. This facilitates to build the controller in a shorter development time than in comparison with for example a procedural language.

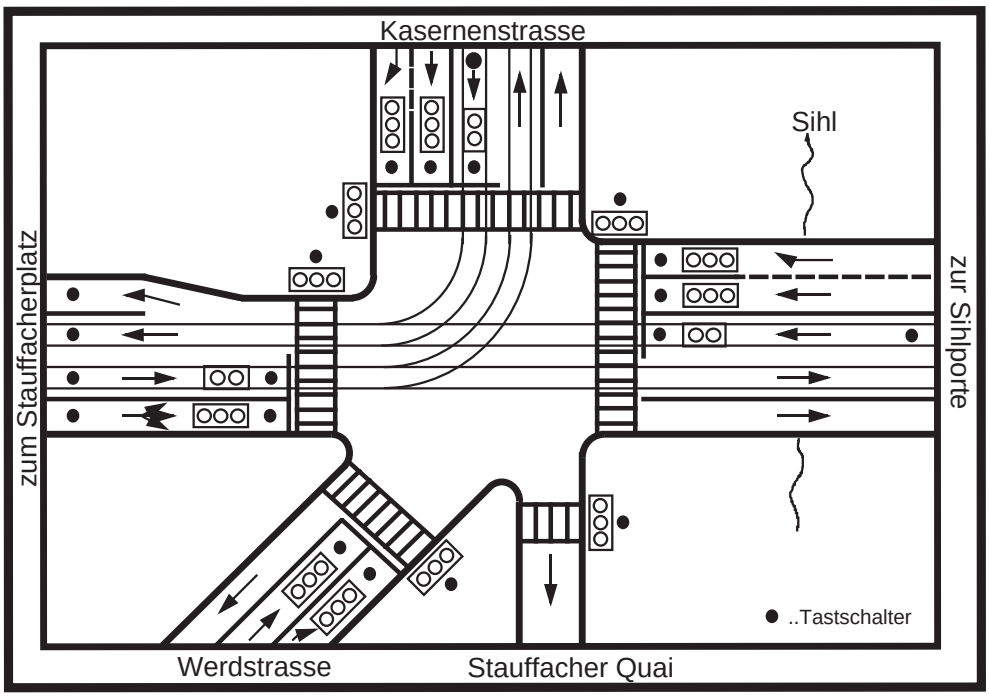

Figure 1: Model of the intersection Stauffacherplatz/Sihlporte.

For this experiment you need the *Einführung in Stateflow* manual which you can download from the web page of the experiment at `http://www.control.-ee.ethz.ch/~ifa-fp/`. To better understand the principles behind Stateflow, it is recommended to read the introduction to Stateflow and try out the example given therein before working through the description of this experiment.

An increasingly important task in the automation of industrial processes is the design of a sequence controller. The process is not controlled continuously, but after each control command, the reached state will be evaluated and based on this, the controller decides which control command will be employed next.

With Stateflow (see „Einführung in Stateflow“) we have a development environment in which we intuitively and graphically can build state flow charts and compile them directly. To work through this experiment in an efficient manner, it is recommended to read the separately available „Einführung in Stateflow“.

In this experiment Stateflow will be used for controlling the traffic lights of an intersection and to process the upcoming traffic automatically. After this experiment you will have

- designed a sequence controller and
- learn about the opportunities and limits of modern graphical software tools.

Contents

Chapter 1

Problem Setup and Notation

1.1 Description of the intersection and the hardware model

For the experiment there is a simple hardware model of the intersection between the Stauffacherplatz and Sihlporte available (Fig. 1.1). Each track as well as the pedestrian crossings and tram tracks leading to the intersection have their own traffic light. The corner Stauffacherplatz/Kasernenstrasse is interesting, because the road is so narrow there that tram and cars must drive on the same track.

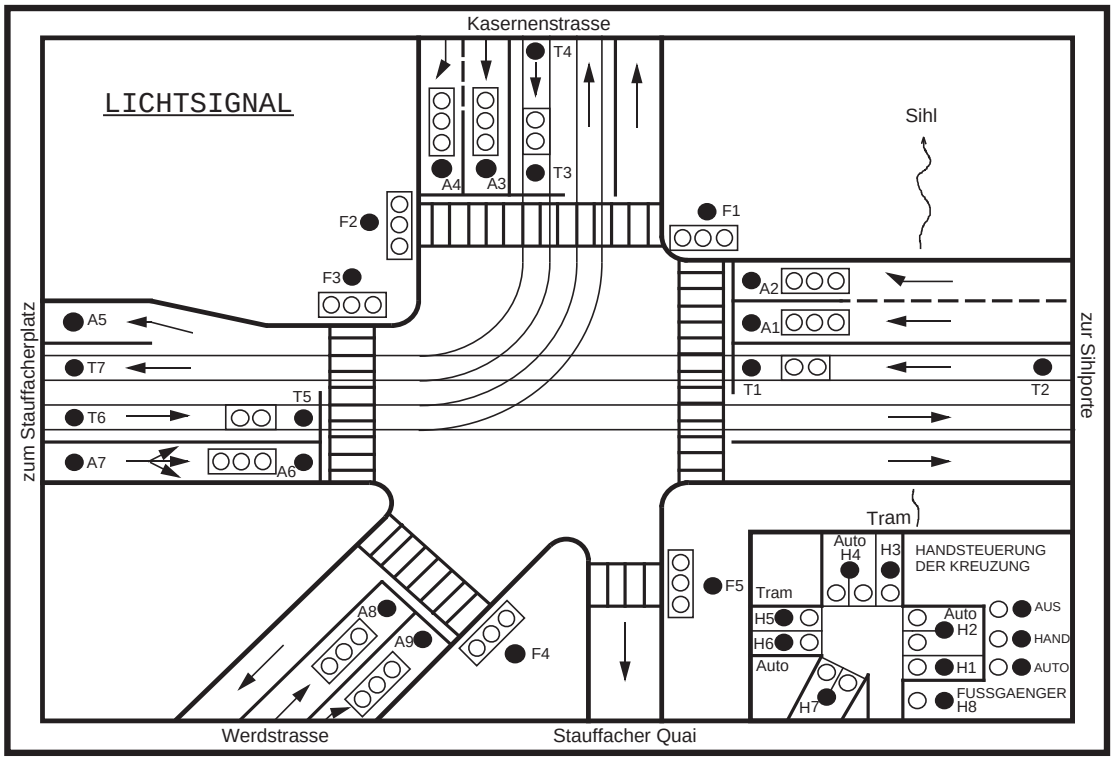

Figure 1.1: Front panel of the intersection model.

In this experiment there are three different operating modes: the manual mode, the automatic mode and the „Aus “mode where all lights are red. The manual control is for getting familiar with the system and to control the different green light situations, while the actual target of the experiment is the development of the automatic traffic light controller by means of Stateflow.

To sense the upcoming traffic, the traffic sensors are modeled by push buttons on the front plate which report the traffic to your controller. This controller should then regulate the traffic by means of predefined traffic light combinations.

1.2 Predefined Traffic Light Combinations

For the sake of simplicity there are only six possible green light situations implemented in the framework program of Simulink. Both in manual as well as in automatic mode one can distinguish only these six situations.

With these situations it is possible to allow each tram, car and pedestrian lane to cross the intersection at least one time. The transitions between the green and red state of a traffic light are controlled automatically. Additionally to these six combinations there is a situation where all traffic lights are red.

- Figure 1.2 shows the first green light situation.

  The traffic lights for the trams from the Stauffacherplatz and from the Kasernenstrasse are green, furthermore the cars from the Werdstrasse towards the Stauffacher Quai and Sihlporte and the cars which turn right coming from the Sihlporte are allowed to cross the intersection.

- In the second green light combination (Fig. 1.3) the same car traffic lights as in situation 1 are green, except that the tram traffic light from Sihlporte is green now.

- In the green light combinations 3 to 6 all tram traffic lights are red. In combination 3 (Fig. 1.4) all lights from Werdstrasse and Kasernenstrasse turn green.

- Combination 4 (Fig. 1.5) allows cars from Sihlporte and Werdstrasse towards Stauffacher Quai to cross the intersection.

- Fig. 1.6 shows the green light combination 5 in which the traffic light for the cars from Stauffacherplatz turns green.

- All pedestrian crossings are summarized to one green light combination, situation 6 (Fig. 1.7).

1.3 Framework Program

To be able to design a working program, one must be able to read the push button inputs and to set the traffic lights of the intersection (Fig. 1.1). In order to relieve you from the details of the hardware, there is an existing framework program provided in Stateflow/Simulink. This framework is presented in Fig. 1.8. The program contains already complete blocks for reading the push button inputs and setting the traffic lights. It is important that you get familiar with the functionality of the framework program before the execution of the experiment by reading the following description carefully. You will not need the program itself until the beginning of the experiment, since it does not work without the hardware intersection model.

1.3.1 Control of the Hardware

The reading and decoding of the inputs, i.e. the push buttons of the intersection model, is performed by the block `Decode Keys` (Fig. 1.8). The three variables `Off`, `Hand` and `Auto` choose the according operating mode. It is always only one of these variables set to 1 and they stay 1 even if the push button is released. A change takes place only when another mode button is pressed.

The push buttons of the manual control are mapped to the variables `H1` to `H8` (Tab. 1.1). The tram and car sensors are also mapped to the variables `T1` to `T8` and `A1` to `A9` respectively.

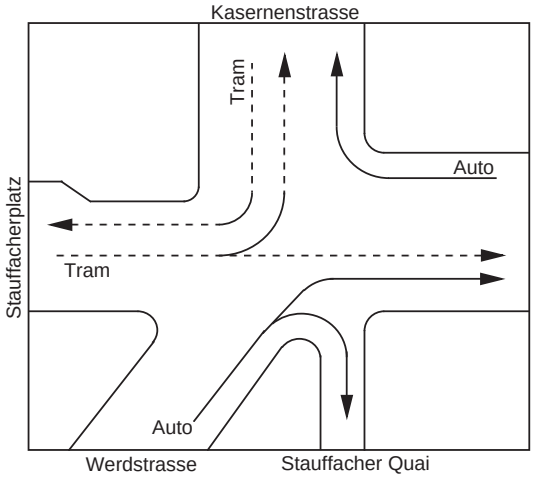

Figure 1.2: Green light situation 1.

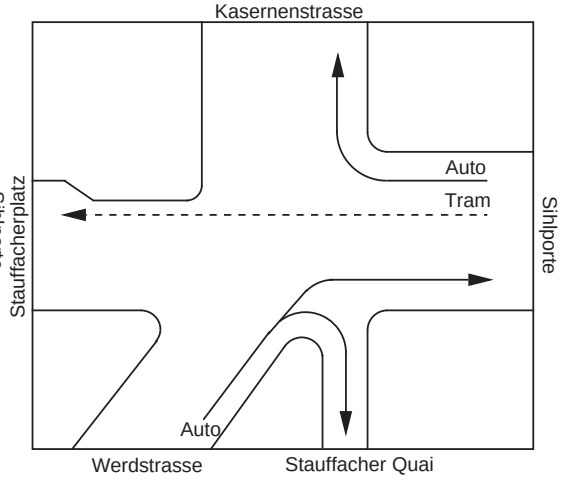

Figure 1.3: Green light situation 2.

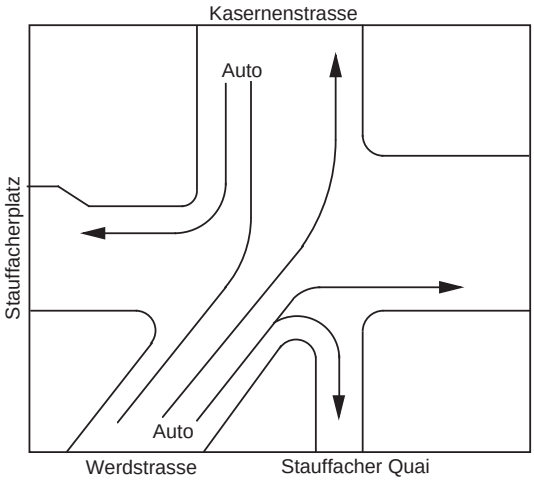

Figure 1.4: Green light situation 3.

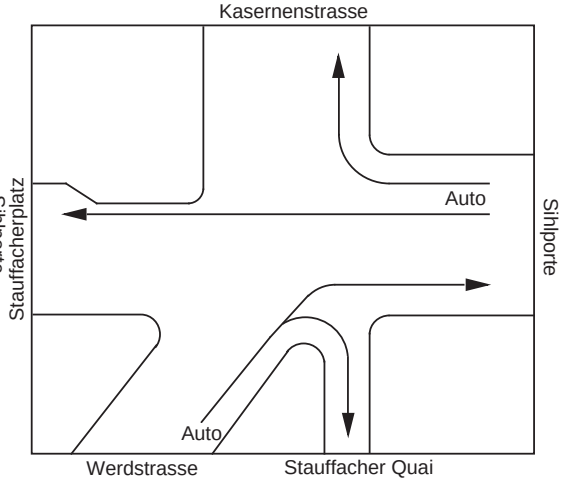

Figure 1.5: Green light situation 4.

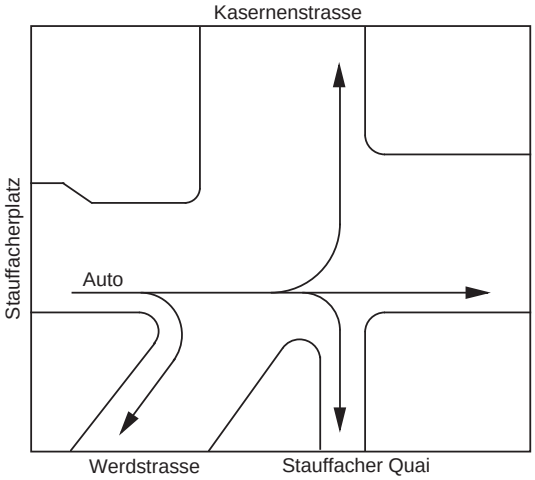

Figure 1.6: Green light situation 5.

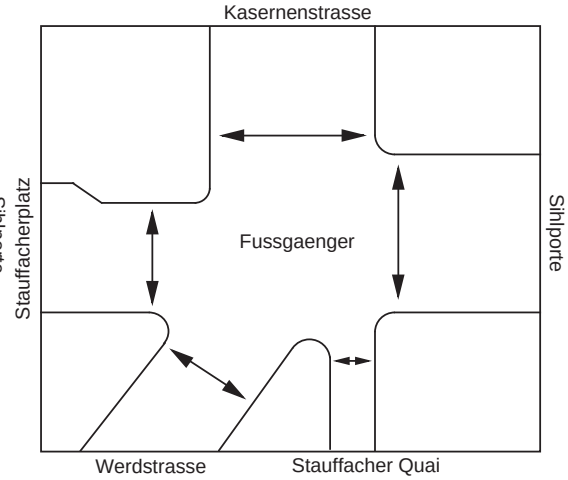

Figure 1.7: Green light situation 6.

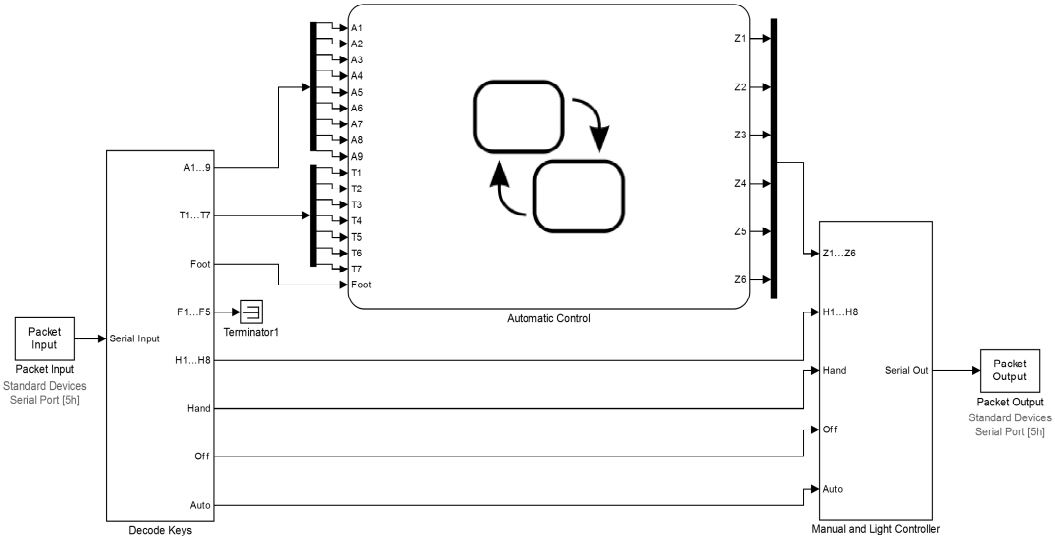

Figure 1.8: The framework program to control the intersection.

The variable `Foot` indicates if any of the pedestrian push buttons was pressed. The variables `F1` to `F5` are not used in this experiment. For this reason they are terminated in the framework program.

By pressing one of the push buttons on the intersection model, the corresponding variable will be set to 1. If the push button is then released, the variable stays on 1 until the next iteration step. For the control of the traffic lights and for the manual control there is a predefined block `Manual and Light Controller` in Simulink available.

The block `Automatic Controller` is the one in which you should implement your automatic control. It contains variables for the tram and car sensors as well as for pedestrians as inputs. The outputs are the variables `Z1` to `Z6` which represent the different green light combinations. To set one of the green light combinations, the corresponding output variable `Zi`, $i \in \{1...6\}$ has to be set. To subsequently reset the green light combination, the corresponding variable has to be reset to 0. The transition from green to red and vice versa happens automatically. Note that it is only permitted to set one of the variables `Z1` to `Z6` to 1 at the same time.

1.3.2 Functionality of the framework program

The control has three operating modes as already mentioned above. They are

1. shut down (**Off**),
2. manual control (**Hand**) and
3. automatic operation (**Auto**).

The switching between these different modes takes place by the mode push buttons. They are located in the lower right corner of the intersection model. If such a switching is performed, the

	Variables	Push buttons on the Simulations panel
Car Sensors	A1, A2, A3, A4, A5, A6, A7, A8, A9	A1, A2, A3, A4, A5, A6, A7, A8, A9
Tram Sensors	T1, T2, T3, T4, T5, T6, T7	T1, T2, T3, T4, T5, T6, T7
Push buttons for manual control	H1, H2, H3, H4, H5, H6, H7, H8	H1, H2, H3, H4, H5, H6, H7, H8
Pedestrians	Foot	F1, F2, F3, F4, F5
Push buttons to select operating mode	Aus, Hand, Auto	AUS, HAND, AUTO

Table 1.1: Provided variables in the prepared Stateflow diagram to determine the current traffic status of the crossroads.

Variables	Light combinations of the traffic lights
Z1	Green Light Combination 1 (Fig. 1.2)
Z2	Green Light Combination 2 (Fig. 1.3)
Z3	Green Light Combination 3 (Fig. 1.4)
Z4	Green Light Combination 4 (Fig. 1.5)
Z5	Green Light Combination 5 (Fig. 1.6)
Z6	Green Light Combination 6 (Fig. 1.7)

Table 1.2: Provided variables in the prepared Stateflow diagram to control the traffic lights

mode changes immediately. Depending on the operating mode either the block Handsteuerung or Automatiksteuerung (or even none of both) are „enabled“ and switched to the system. The block Handsteuerung is also already pre-implemented. The push buttons for the manual control are also located in the lower right corner of the hardware model inside of the panel for the manual control mode. They are numbered from H1 to H8. If the operation mode „Handsteuerung“ is chosen, first all traffic lights will be set to red. Then the system waits for a push button H1 to H8 to be pressed and turns the corresponding traffic lights to green according to Tab. 1.3. This state remains until another push button is pressed. Then the traffic lights which were green before will be set to yellow and after a small amount of time to red, before the new combination is set. If in the mean time a control mode button (AUTO, AUS) was pressed, the system switches into the corresponding operation mode. The realization of the manual controller can be found in Fig. 1.9. The diagram begins with the state Init. Subsequently, the „Inner Transition“ from the frame of the macrostate to the „Junction“ is executed over and over again. At the „Junction“ the system decides to which state it should change according to h1 to h8 (see *Einführung zu*

Button	Change to green light combination
H1	2
H2	4
H3	1
H4	3
H5	1
H6	5
H7	3
H8	6

Table 1.3: Push buttons of the manual control and the corresponding green light combinations

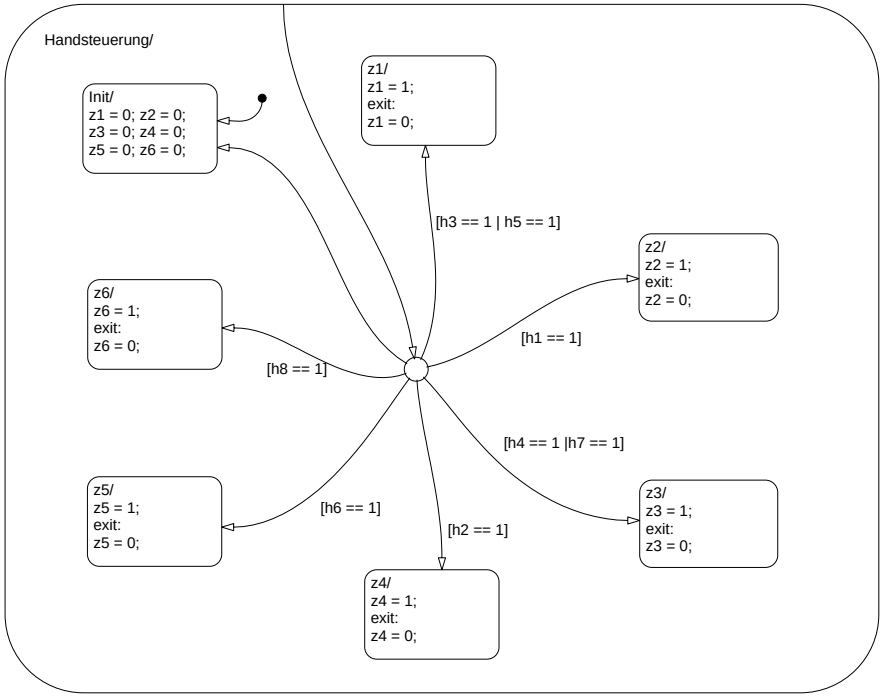

Figure 1.9: The Stateflow program for the manual control. The diagram begins with the state `Init`. Subsequently, the „Inner Transition“ from the frame of the macrostate to the „Junction“ is executed over and over again. At the „Junction“ the system decides to which state it should change according to `h1` to `h8` (see *Einführung zu Stateflow*)).

Stateflow)).

1.4 Specification of the automatic control

In this operation mode the traffic detection should take place over the corresponding sensors. These are simulated by the push buttons on the hardware model. We distinguish between the following sensors:

- Car sensors: With the car sensors the upcoming car traffic will be detected. There is a sensor placed in front of each stop line which registers the waiting cars. In the lane coming from the Stauffacherplatz there is an additional sensor (`A7`) placed in front of the intersection to sense the cars before they have to stop in front of the red light.
- Tram sensors: For the tram traffic there are built-in signal sensors similar to the car ones. But there are additional ones for each lane. The first sensor is located 140 m in front of the intersection and is for the announcement of the tram. The second sensor is situated directly in front of the stop line.
- Pedestrian sensors: By each pedestrian crossing there is a push button located at which the pedestrians can announce themselves.

Tab. 1.4 shows for each variable (i.e. sensor), which green light situation can be activated to allow the corresponding cars, trams and pedestrians to cross the intersection. Thereby it summarizes Fig. 1.2-1.7. Sometimes there are multiple possibilities to react on an announcement - here you can decide which green light combination you prefer. The sensors `A5` and `T7` are not needed in this experiment.

Sensor	Green Light Combination
`A1`	4
`A2`	1, 2 or 4
`A3`	3
`A4`	3
`A6`	5
`A7`	5
`A8`	3
`A9`	1, 2, 3 or 4
`T1`	2
`T2`	2
`T3`	1
`T4`	1
`T5`	1
`T6`	1
`Foot`	6

Table 1.4: Required green light situation depending of the traffic situation

1.4.1 Functionality

Your automatic controller should operate as follows:

1. The initial state should be red for all traffic lights. If a car waits in front of the intersection the traffic light should switch to green. The green time should be extended if the push button gets pressed multiple times (= much traffic). Define also a maximum green time to prevent starving in case of high traffic density (i.e. one green light combination never gets activated because of too much traffic on another lanes).

2. If one or more cars or pedestrians announce themselves while another traffic lights are green, they should be registered. After completion of the actual green light situation their traffic lights should become green. This is to ensure a fair allocation of green phases to prevent the effect of Starving. If there is no traffic, after a certain time all lights should set to be red.

3. If a tram registers itself at the first sensor (advance announcement), you should allow the tram to cross the intersection as soon as possible because they have a higher priority than cars and pedestrians. If the actual green phase was set because of a car, the tram traffic light should become green immediately. But if the green phase was set by another tram, you have to wait for the end of the current green phase (a tram has only a higher priority than a car in contradiction to another tram).

Chapter 2

Preparation@Home

In the end, the automatic controller should consist of two processes: the `Collector` and the `Controller`. The `Collector` should count the readings on the different sensors. The `Controller` decides which and for how long a green light situation will be set. The following tasks will guide you step by step through the design of the program.

You should prepare yourself for the lab session by working through the following tasks:

1. **Get familiar with Stateflow**
   Read through the introduction to Stateflow and try the example of the vending machine.

2. **Get familiar with the framework program, the variables and the functions**
   Familiarize yourself with the framework program (Fig. 1.8) and understand its sequence of events. Familiarize yourself especially with the variables you need for the automatic controller.

3. **Design of the state diagram of the collector on paper**
   The Collector process counts the announcements of the different sensors. To be able to do this you have to define a counter variable for each green light combination 1 to 6. When a car announces itself, the counter of the corresponding green light combination should be incremented (see Tab. 1.4)). The variables you should implement are `Anz1`, `Anz2`, `Anz3`, `Anz4`, `Anz5` and `Anz6` for the green light combinations 1 to 6. Use for this reason a state diagram with an „Inner Transition“. These should i.a. branch to an empty default state `Idle` to capture irrelevant signals.

Chapter 3

Lab Session Tasks

1. Turn on the computer and ensure that the hardware model is plugged in.

2. You can find an icon `IfA 1.6 Traffic Control` on the desktop of the computer. After a double click a script will be called which copies the required files automatically in the directory `C:/Scratch/Traffic_Control`. MATLAB starts and the framework program opens, compiles and starts too.

3. After each change in the Stateflow diagram you have to recompile and restart the system: The compilation takes place using the button `incremental build` in the Simulink window or with the shortcut *Ctrl+t*. To start the program you have to push `Connect to target` first and afterward `Simulation.Start` in Simulink (not in the Stateflow model). Now you can switch between the different operation modes and control the intersection manually as described above. If you now switch to the `Auto` mode, nothing will happen. It is on to you to program this part now according to the specifications in section 1.4.

4. Implement now the solution for the `Collector` which you have prepared and save it in Simulink with `File.SaveModel`. Do not forget to save your intermediate results regularly.

5. **Design of the state diagram for the controller**
   Now the actual controller should be developed. It is recommended to build the process in the following order:

   (a) There should be a state `Z1/` to `Z6/` for each green light situation 1 to 6. The states should turn the traffic lights of the corresponding light combination green and at the same time reset the counter of the situation which was incremented by the `Collector`. Afterwards set the traffic lights to red again.

   First concentrate only on the design of the states `Z1/` and `Z3/`. With these states you can control the green light situations 1 and 3.

   (b) Now try with these two states `Z1/` and `Z3/` to assemble the first elements of the automatic controller with an „Inner Transition“ based on the program for the manual control. Add now a default state `RED` which will be activated if the traffic situation does not require any green lights. Further the default transition should join the state `RED` (all traffic lights are red in the beginning). The counters `Anz1` and `Anz3` should be sampled for the transitions to the states `Z1/` and `Z3/`.

   (c) Now we will implement the time delay which ensures that a green light combination will be on for at least some given amount of time until the traffic lights become red again. This time delay should be at least a base value `basis` but should vary depending on the amount of traffic waiting (frequency of the announcements). For this reason, the total time delay `s_delay` should be extended by `verl` seconds per car waiting at the intersection.

Implement the variables `s_delay`, `verl`, `basis` as well as `tstart`. The „Inner Transisition“ should only be activated if `s_delay` seconds are elapsed. If the process is currently in the state `RED`, the „Inner Transisition“ should be always executed to react to announcements as soon as possible.

Test your program extensively on the hardware model to ensure that you have done everything right so far and that you are able to switch the green light situations 1 and 3 with the corresponding sensors. Check your time delay as well.

(d) Now you should implement the remaining states for the green light combinations `Z4/`, `Z5/` and `Z6/`. To sample the green light situations in a fair manner you should implement a modulo calculation in the default state `RED`. Add a new variable `nz` which should choose one of the four states `Z3/`, `Z4/`, `Z5/` and `Z6/` round robin based on a modulo calculation to get a fair distribution for all green light combinations.

(e) At last you should implement the third task of the functional specification: The controller should give priority to waiting trams even if the cars and pedestrians light are already green. Add this state `Z2/` for the green light combination 2 and ensure that the „Inner Transition“ is executed also if a tram announces itself.

Implement an exclusion mechanism by use of a variable `tram` so that a new registered tram has to wait in case that another tram got already green.

Test the new functionality by announcing some cars and pedestrians first and by pushing one of the tram buttons afterward. Also check that two trams cannot displace each other but their green phases are switched one after the other.

6. **Anti Starving Algorithm**

As you might have recognized, the problem of Starving is not completely solved with your previous implementation. Because if there are many cars waiting on a track, the following green phase can become extremely long. To get rid of this problem you have to implement a maximum allowable green time.

(a) Add a constant `tmax` with an appropriate value. Edit the current calculation for the variable `s_delay` to ensure that `s_delay` $\in$ (`basis`...`tmax`).

Now we have a useful implementation for the calculation of the time delay. But there is a new problem which shows up in our `Controller`. If there are too much cars waiting in front of a traffic light what requires to set `s_delay` to `tmax`, the variable `Anzi` will be set to 0 anyway. This causes the redundant cars to get forgotten.

(b) To solve this issue you have to edit the calculation of the variables `Anzi` $i \in \{1...6\}$ in the `Controller`. We expect that every `ext` seconds one car can pass the intersection. With this information we know that the `Anzi` variables are decreased by `s_delay`/`ext` per green light interval. Note that the variables can not turn negative.

Test your `Controller` by announcing more than `tmax` cars on a green light combination and check if the maximum green time is kept by your implementation.

Chapter 4

Lessons Learnt

In this experiment you have learnt how to implement a sequence controller in Simulink by means of Stateflow. You developed a functioning controller for a traffic intersection, which controls the green times and green light combinations automatically by analyzing the upcoming traffic.

Lesson Learnt 1: Suboptimal calculation of `s_delay`

1.1) This kind of implementation does not calculate the optimal traffic light green times. The calculated `s_delay` is always an upper bound for the needed green time for a green light combination. Explain why.

1.2) How could you improve your implementation to get rid of the problem in 1.1 and to be able to calculate a more efficient `s_delay`?

Lesson Learnt 2: Anti Starving

2.1) How did you prevent that cars starve in front of the intersection?

Task 1: Completion of the experiment

1.1) Please fill in the feedback form on the registration page at `MyExperiments`. Each student has to fill in his own feedback form. This helps us to continuously improve our experiments.

1.2) Discuss the experiment with your supervisor to get the attestation.