I²C interface

The model can be controlled via I²C (Two Wire). It operates as a I²C slave with the address *111*. It expects messages of up to four bytes. The first byte acts as a command byte that selects the function to be executed. The following one to three bytes are arguments for the choosen command.

Nearly all functions of the model are timer and interrupt based and therefore asynchronous. This implicates that none of the functions is transactional. Some will be executed immediately others might be executed with a varying delay. The approximate wait time for a command to be executed is described for each command as the `timing` property. Keep in mind that I²C communication interrupts the main processor and can therefore cause further delays when issued too frequent.

Each command must be transmitted within a single I²C transaction and thus be preceeded by a I²C START and commited by a I²C STOP sequence. Sample sequence:

```
START -> $CommandByte -> $DataByte1 -> $DataByte2 -> STOP
```

With `$CommandByte` beeing one of the available commands (see `Constants`) and `$DataByte1`, `$DataByte2`, ... beeing one of the available data bytes or a numeric value (see below).

The same sample as code:

```
twStart (111); // activate I²C slave #111
twWrite ($CommandByte);
twWrite ($DataByte1);
twWrite ($DataByte2);
twStop ();
```

All byte values are listed at the end of the document within the `Constants` chapter.

Available commands

Town hall RGB LEDs

Lights of the town hall can be set to any color within the RGB spectrum. It further features a demo mode that will slowly fade through all available colors. Demo mode is enabled by default when the model resets.

`TW_HOUSE_DEMO`

This command can be used to enable or disable demo mode. Demo mode wont be enabled/disabled automatically when other commands are issued. If demo

mode is not disabled sent commands will be overriden by demo mode commands. So keep in mind to disable it before issuing any other `TW_HOUSE_*` commands.

Command structure:

```
START -> TW_HOUSE_DEMO -> [TW_DISABLE|TW_ENABLE] -> STOP
```

Timing: up to 2.6 seconds delayed

`TW_HOUSE_RGB`

Set all three color channels at once. Make sure to disable the demo mode and to honor the possible delay.

Command structure:

```
START -> TW_HOUSE_RGB -> $RED -> $GREEN -> $BLUE -> STOP
```

With `$RED`, `$GREEN`, `$BLUE` being the actual values (`0x00`...`0xFF`) for each channel.

Examples:

```
START -> TW_HOUSE_RGB -> 0x00 -> 0xFF -> 0x00 -> STOP // set the color to blue
START -> TW_HOUSE_RGB -> 0xFF -> 0xFF -> 0xFF -> STOP // set the color to white
START -> TW_HOUSE_RGB -> 0x00 -> 0x00 -> 0x00 -> STOP // turn all lights off
```

Timing: instant

`TW_HOUSE_R`, `TW_HOUSE_G`, `TW_HOUSE_B`

Sets each corresponding color separately. Examples:

```
START -> TW_HOUSE_R -> 0x00 -> STOP // turn off red color channel
START -> TW_HOUSE_B -> 0x80 -> STOP // set blue color channel to 50%
START -> TW_HOUSE_G -> 0xFF -> STOP // set green color channel to full power
```

Timing: instant

Ferris wheel, Riesenrad

`TW_FERRIS_ENABLED`

Enables or disables the ferris wheel spinning around. It is enabled by default when the model resets.

Command structure:

```
START -> TW_FERRIS_ENABLED -> [TW_DISABLE|TW_ENABLE] -> STOP
```

Timing: instant

Metro

These commands control the two metro trains. It features a demo mode just as the town hall.

`TW_METRO_DEMO`

Wheter to enable or disable the metro demo mode. In demo mode both trains will arrive and depart in regular intervals from the station. Disable demo mode before issuing metro drive commands and honor the possible delay. If demo mode is not disabled commands sent will likely be overriden by consecutively commands issued by the demo mode. Demo mode is enabled by default when the model resets.

```
START -> TW_METRO_DEMO -> [TW_DISABLE|TW_ENABLE] -> STOP
```

Timing: up to 2.6 seconds delayed

`TW_METRO_1_DRIVE`, `TW_METRO_2_DRIVE`

Let the corresponding train depart or arrive at stations. Metro 1 is below the town hall and metro 2 is below the ferris wheel.

Command structure:

```
START -> TW_METRO_1_DRIVE -> [TW_INTO_STATION|TW_INTO_TUNNEL] -> STOP
START -> TW_METRO_2_DRIVE -> [TW_INTO_STATION|TW_INTO_TUNNEL] -> STOP
```

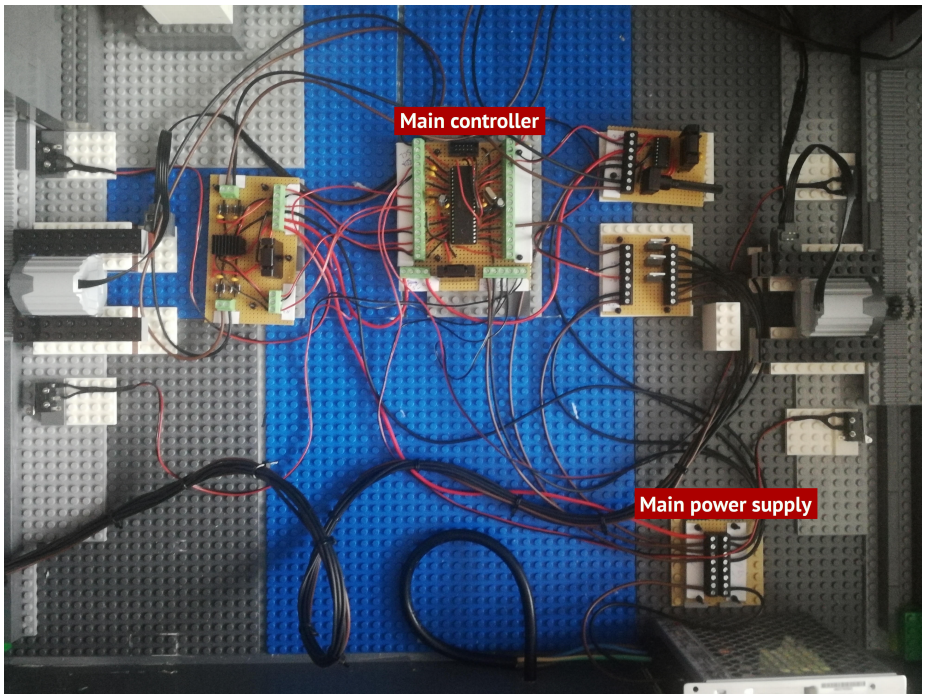
`TW_INTO_STATION` lets the train arrive at the station

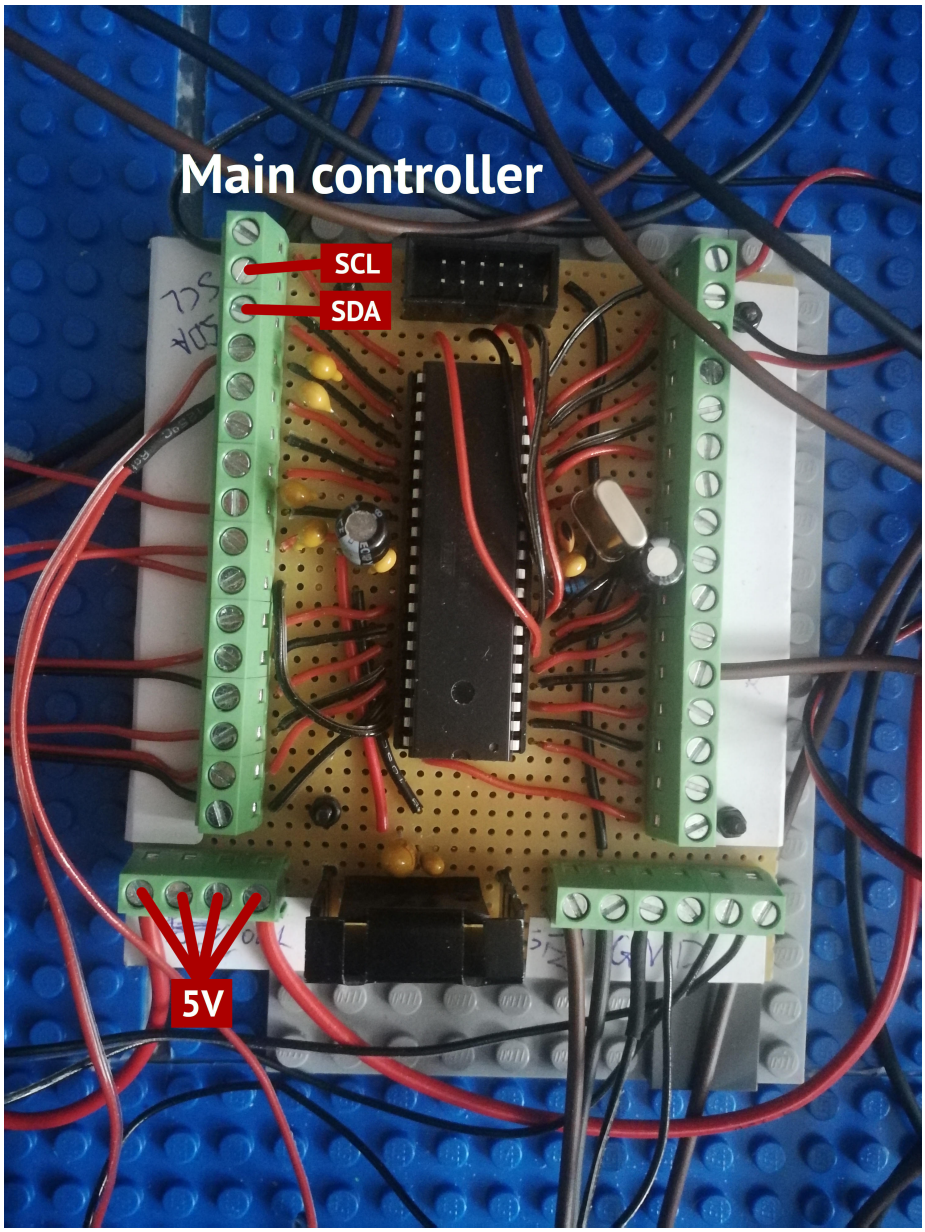
`TW_INTO_TUNNEL` lets the train depart from the station

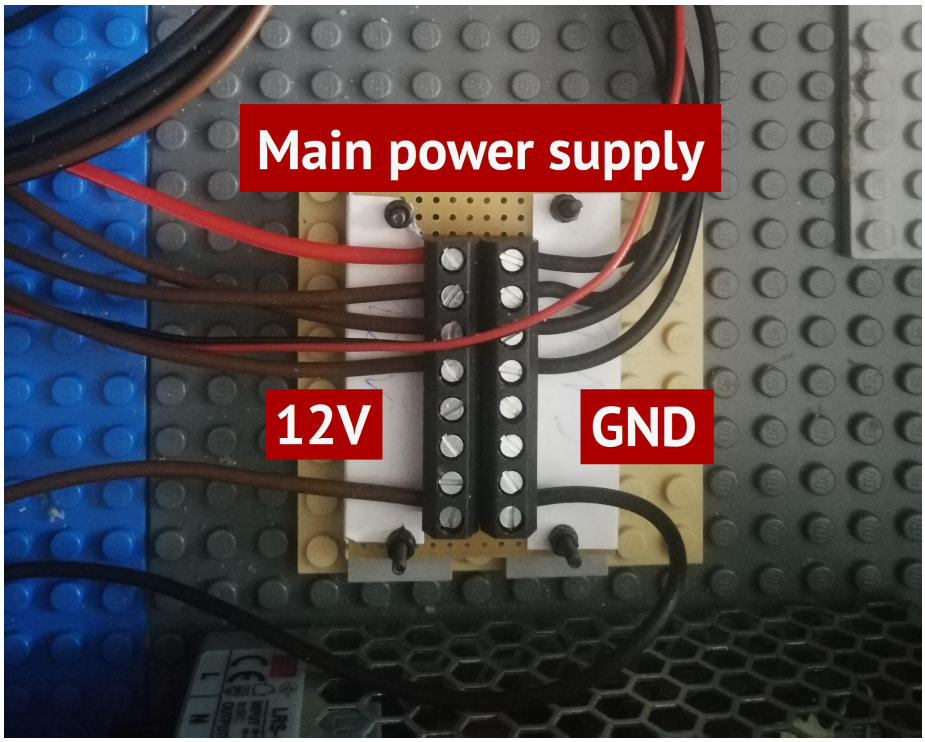
Timing: instant

Wiring

Connect with the I²C SDA and SCL pins on the main controller board. Further make sure to connect the grounds (main power supply) if you use an external power supply. Small loads like simple microcontrollers (ESPs, AVRs, PICs, not a Raspberry) can be powered via the 5V power supply located on the main controller board. Ground can be sourced from the main power supply board as well as 12 volts.


Main controller
Main power supply


Main controller
SCL
SDA
5V



Constants

Command bytes

```
#define TW_HOUSE_DEMO 0x20
#define TW_HOUSE_RGB 0x21
#define TW_HOUSE_R 0x22
#define TW_HOUSE_G 0x23
#define TW_HOUSE_B 0x24

#define TW_FERRIS_ENABLED 0x30

#define TW_METRO_DEMO 0x40
#define TW_METRO_1_DRIVE 0x41
#define TW_METRO_2_DRIVE 0x42
```

Data bytes

```
#define TW_DISABLE 0x00
#define TW_ENABLE 0x01
```

```
#define TW_INTO_STATION 0x00
#define TW_INTO_TUNNEL 0x01
```