

# Train seat management









I'm Stephanie Minne, I study New Media and Communication Technology (NMCT). At school we got the assignment to make a device with a webpage.

I have chosen to make a train seat with a sensor in it. The sensor will detect if there is a person sitting on the seat.

## Step 1: Required Materials

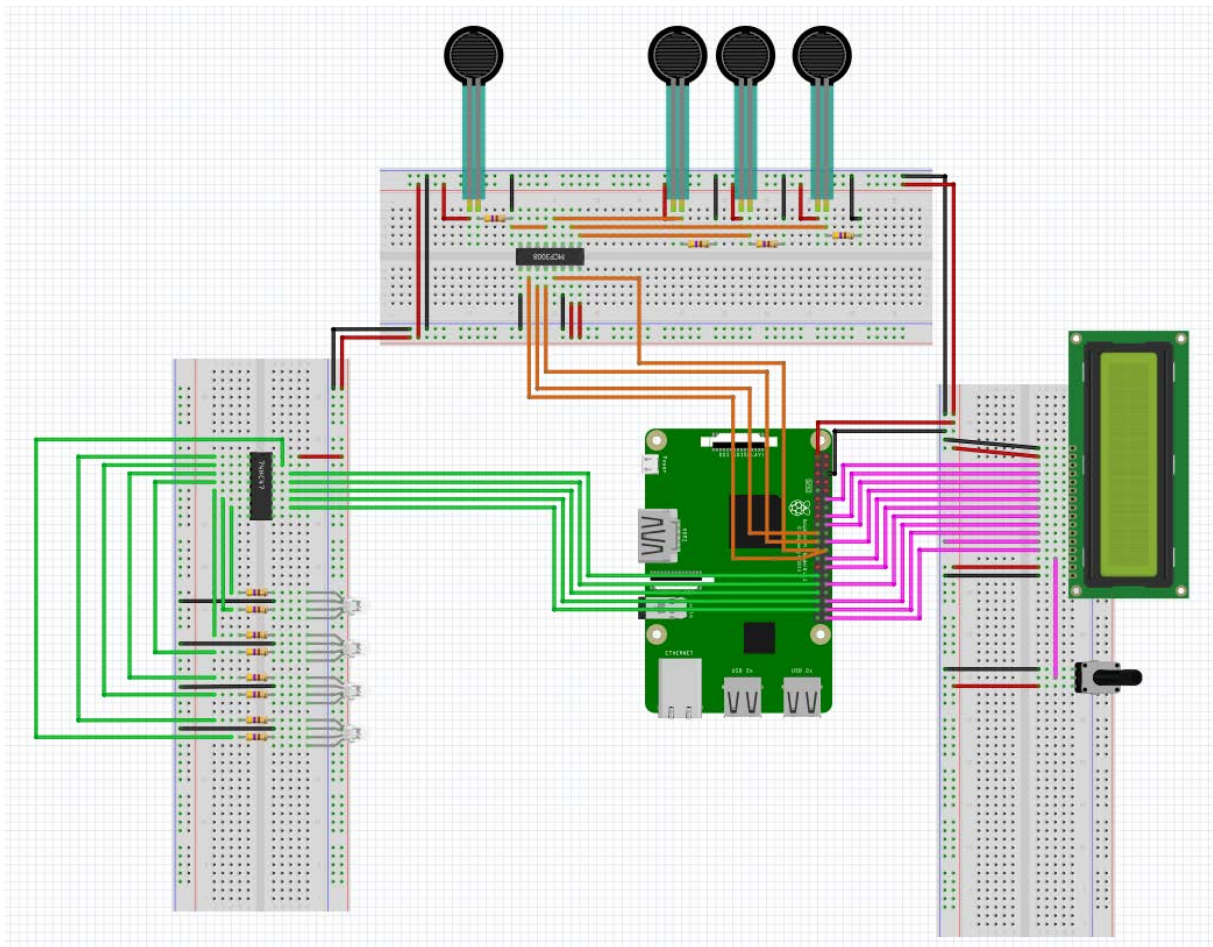
For this project several materials are needed.

Pressure sensor (fsr 400) = €9,20/unit	
Raspberry Pi 3 = €37,95	
RGB leds = €0.5 /unit	
Resistor (470 ohm) = €0.10 /unit	
LCD display = €2.44 /unit	
Mcp 3008 = €29,00 /unit	
Sn74hc595n = €1,20 /unit	

Potentiometre = €1,15



## Step 2: the circuit



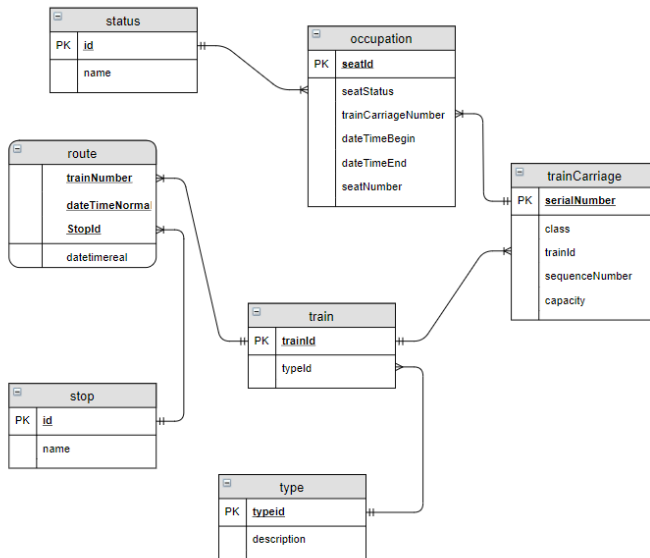
The hardware is pretty easy.

As can be seen on the schematic, I used a 74hc47 for my leds. The reason for that is because my raspberry Pi don't has a lot of pins. The disadvantage of this, is that you only can make the led red or green. You don't have a pwm signal to make different colors.

The raspberry Pi doesn't have analog input pins. So I need a mcp 3008. This converts an analog signal to a digital. The pressure sendor is put on the channels of the mcp 3008.

### Step 3: make a database

In this project a mysql was used as database server.



The table occupation has information about the **sensors**. It captures where a person sits on a train seat. From the moment a person sits on the seat the first data will be written in the table. All the data will be written in the table with the exception of the dateTimeEnd.

In the table route comes data about the different **routes** that a train does.

The table has two values: one for the normal datetime and another for the real datetime.

The datetime normal is the time that the train is at a stop theoretically. The datetime real is the time that the train is at a stop for real. This time is with the delay inclusive.

Every train has a clear and **unique reference**, composed by letters (p.e. IC stands for Intercity train) and numbers (p.e. 4565 is the train from Brussels to Antwerp).

And every type has its own **accommodation**. For example tables, rubbish bins.

## Step 4: code it!

The base code of the sensors is written in Python. There are 2 kinds of code. The first one is for the sensors. The second one is for the webpage. The webpage data is written in Python, Flask, JavaScript, HTML and CSS.

### The sensor

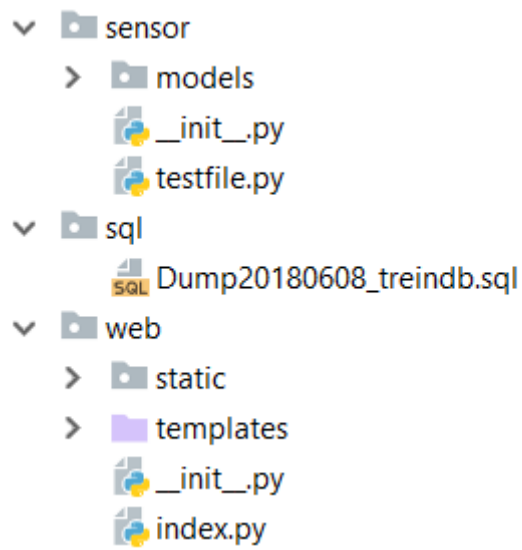
Every component has its own class. All the different classes are used in the test.py.

### The web

The web code is written in Python and Flask. For the header a template is used. The other piece of the page has its own html page.



## The file structure



- The test file can be found in the directory sensor.
- The classes are located in the directory models.
- The directory sql contains the dump of the train database.
- The directory web contains the index.py. This is the Flask page. This file has to be run in order to run the webpage.
- The templates directory contains the html pages.
- The directory static contains images, fonts and the css files.



## Step 5: make it beautiful

There are different ways to make a beautiful train. I have chosen to make a 3d print of a train.

### The seats.

A small hole has to be foreseen in the seat as the sensor needs to be put on the seat. The leds are fixed on top of the seat.

### The wagon.

There are a few important features in the wagon. At first there must be little holes in the floor where the seats come.

1. The sensor and the cales of the leds should go though these holes.
2. In the sidepanel there is a hole to fix the LCD display with the information on the IP address and the number of free seats.
3. The Raspberry Pi should be located on the back out of sight.

!Attention: the 3d print is printed a bit smaller so an extra 3 or 4 mm for the holes must be foressen!





