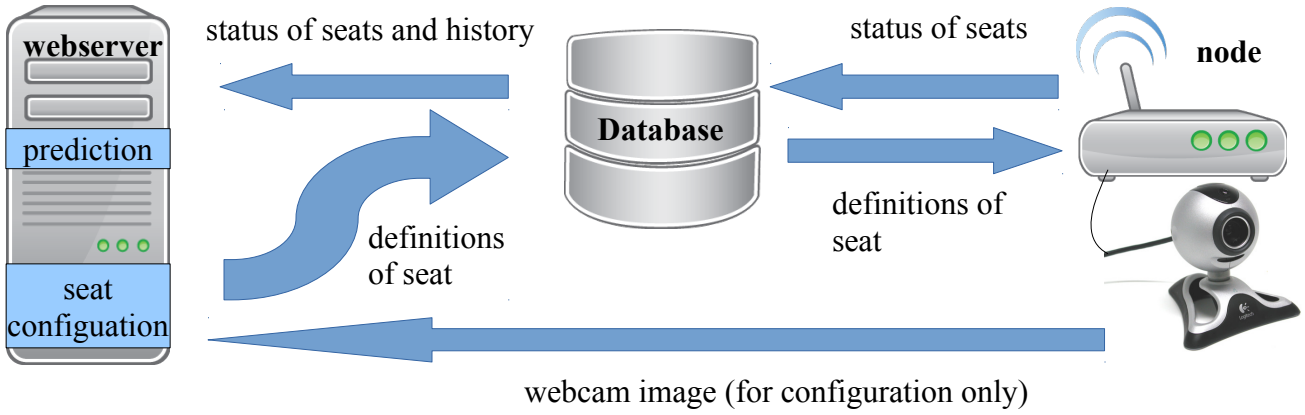
TK3 final project Group E

Project idea
Our porject aimed at implementing a seat monitoring system. It should detect whether seats in different rooms are free or used at the moment and also enable prediction for this. To archieve this, a node with a webcam is places in the room performing motion detection algorithms. The data (for all rooms/nodes) can then be accessed from a central webserver which also performs the prediction algorithms.
With this system the rooms get smart and know their current utilization. The user no longer needs to go to all rooms to check for available seats but can check this in advance.
Possible use cases are all scenarios with shared seats like computer pools at a university, libraries or also public transportation.
Since different work places may include different equipment, "tags" are used to define these. With this it is also possible to e.g. look for working places equipped with a PC.

System architecture
Our system consits of three main parts:

- nodes with webcams in each room
- central database
- web-interface to access the data and also for seat configuration

Basically the interaction between these is pretty straight forward. The nodes senses for motions on the defined seats and write the "sensed" data to the database. The webserver then accesses this data from all nodes, performs prediction algorithms and displays it to the user. However, there are some minor details which will be discussed in the different sections of this document.
The following illustration shows an overview of our system, for better overview with just one node:



Motion Detection:
One of the cores of our implementation is the motion detection. Therefore we used a capturing device, i.e. a webcam or camera and some third-party software to process the data that our device provides. For this image processing we used the OpenCV library which is written in optimized C/C++.
For the motion detection we basically have to place the camera in a fixed spot where we have full vision over the complete area we want to observe. After that it has to be defined where the seats are in

this view. This is done manually in the webbrowser. Once all coordinates of the seats are properly defined, the motion detector can be activated. It captures frames in a predefined time interval and compares every frame with its predecessor. We used the simplest approach where we just calculate the difference between two frames, because this approach fully meets our condition. But in order to get more reliable results, some filters are applied to reduce noise interference. First we convert the colors of every frame from RGB into a grayscale. After that the absolute difference of the two frames will be calculated. To get best results we eliminate noise by appling a threshold function on the frame. We use a binary threshold, so if the intensity of the pixel is higher than a certain threshold, the pixel is set to the maximum value. Otherwise the pixel is set to 0. So finally we have to count the number of white pixels in a certain seat to decide whether there is a motion (occupied) or not. For better performances all the pixels that are inside one quadrangle (seat) will be precomputed and so we also do not have to iterate over all pixels of the frame, but only where a seat is.

Logging the Seat Status:
Whenever the Motion Detection is activated due to its interval, it calls the Database Connector to write the status for the seats into the central database appropriate tables. These are the 'seats' table, storing the current status, and the 'history'-table, which is utilized in the seat status prediction.
Apart from writing the current occupation for the given seat in the node's room into the database, the connector is also called once by the Motion-Detection module, to receive the predefined positions of seats in the webcam's videostream.
This part utilizes the mySQL -connector c++ framework.

Web Frontend:
Index page contains a table contain a list of all the monitored rooms. In every row there is a button which will show the details of each room.
In the each room details page list of all available seats can be found along with the current status of the seat, type of the seat and a link to history of that particular seat.
In the seat history page, seat details and seat occupancy history information is there.
Apart from these there is a room search option where users can search for any seat at any availible room and can check whether the seat is empty or not.

Seat configuration front-end
The web front-end includes an interface to define the seats in a room (*tk3_web_frontend/seat_config*). The user interaction is handled in JavaScript. For the connection to the database server side php-scripts are accessed over AJAX requests and JSON strings are used as return values.
The interface allows to create new seats, edit and delete existing seats. This also includes defining tags for the different seats.
For the visualisation and drawing of sests a HTML5 canvas is used. Since the user wants to see where the seats are located, the webcam image of the node has to be displayed. As the seat configuration runs on the webserver and not on a node with the webcam the image has to be fetched from the node. To archive this the node registers with its IP address and a TCP port at the database. We implemented a simple HTTP server (*ImageServer.cpp* in MotionControl code) running on this address which answers requests with a jpg image from the webcam.

TK3 final project Group E

Seat status prediction

Checking if a seat will be occupied or free depends on the actual time a request is entered and on the previous seven days. The code is implemented in Python with use of the mysql.connector and datetime. A request is made by calling the function *check_occupied(req_room_id, req_seat_id)*. This arguments passed are the requested room id and and the request seat id for the room. In this function the actual hour of the request is extracted in 24h format. Now a range to check for is generated, depending on the actual time. The range includes the actual hour and the two previous hours.
After a connection to the MySQL-Database has been established, the SQL-Query is filled the requested room and seat id. The query will look like:

SELECT times, occupied FROM history WHERE seat_id=1 AND room_id=1 ORDER by room_id DESC;

As a result the query returns all timestamps and occupied values (0 if not occupied, 1 if occupied) for this seat in the room stored in history-table. These results (about 10080 entries per unique seat) will now each checked if its hour from the timestamp is in the previous defined range. If true, the occupied value will be added to an overall score and a counter will be incremented. As soon as all entries are checked the overall-value will be divided through the counter to get the ratio of occupied entries to total entries for this unique seat in the chosen range. If this score is greater than 0.5 we assume a seat is occupied, otherwise it is free (under the assumption, that a person is regular studying at his favorite seat for a longer period of time). The result will be returned as a String value

Install/run/compile instructions

As our system consists of many different loosley coupled parts, there are some steps required to set up everything:

1. Set up a SQL server (MySQL) with the database found in *database/tk3_final_project.sql* in the database "tk3_final_project" and a user "tk3" with password "tk3".
2. Install a webserver (e.g. Apache) and copy the files in *tk3_web_frontend* to it. If it is running on a different system than the database you have to adapt the settings in *server_config/config.php*
3. Compile the node (MotionDetection) code. It depends on some libraries, so you have to install them first. For debian/ubuntu everything can be found in the default repositories: *opencv_core, opencv_imgproc, opencv_highgui, pthread, mysqlclient, mysqlcppconn.* If the mySQL c++ connector is not already installed on the node, it needs to be built. Certain required header files for that, like the mysqlconnector c, and the boost libraries should come in the src\include directory together with this.
   Attach a webcam to the node if it does not already has one. Start it AFTER step 4 with the number of the room for this node and (propably) the DB server data:
   usage: *MotionDetection ROOM_ID [DB_SERVER_IP] [DB_SERVER_PORT]*
4. Configure youre seats over the web interface, start the node AFTER this and never again walk to a room without free seats!