Implementation of a Server for a GPS/GSM Tracker

Nils Schlegel
Sebastian Romero Chavero

Abstract- "As part of an ever-increasing fleet management system, the location information of the various vehicles is playing an increasingly popular role."**[1] Because of this, a reliable way to store both GPS and GSM data is needed and so is a simple way to display it in a User-friendly interface.**

Index Terms – Localization, GPS, IoT

I. INTROCUCTION

A. Motivation

Localization is an increasingly popular application of vehicle fleet management. This can result in various advantages, such as increased security or simplified administration. Further savings in operating resources are possible because, among other things, fuel savings and automatic creation of the logbook are possible. GPS-based outdoor localization systems are a suitable solution for this. These systems provide a localization accuracy of up to 2m and form an economically scalable system.[1]

Such a system relays onto some kind of server in order to be able to access the location data from anywhere if needed. Therefore, such a server must be able to receive and manage the gathered data from the tracker attached to the desired object.

In this paper it is intended to implement such a server on the host PC, that can store the data produced by a GPS/GSM tracker with the use of MySQL and Apache. Both of those tools are publicly available and provide a detailed documentation [2-3] combined with a broad community of experts. Due to the possibilities of those servers combined with a web browser not only a powerful backend but also a visually appealing frontend can be created.

There are already some solutions [4-5] available online but most of these solutions do not fulfill the requirements for the system.

B. Structure of this paper

The remainder of this paper is structured as follows: Section 2 gives an overview of the localization system that is used. Section 3 outlines the structure and implementation of the server in PHP and SQL as well as the visualization of the data with HTML, CSS and JavaScript. The functionality of the server is also demonstrated here. Finally, Section 4 concludes the work and gives perspective on future topics.

II. LOCALIZATION SYSTEM AND SERVER OVERVIEW

A. Localization system overview

"The demonstrator who has been chosen is a tracker system for a car fleet, e.g., Uber cars. The idea of the 1 IMPLEMENTATION OF A GPS AND GSM MODULE […] TRACKING SYSTEM whole tracking system is shown in figure 1. Here we have the tracker consisting of a GPS receiver, a microcontroller, a power supply, and a GSM transceiver. The GPS module receives the data from the satellites and forwards the position data and the time stamp to a memory location. The GSM module is requested to collect the power data and the

corresponding base station IDs, which will be also stored in a memory. If a complete set of data has been collected, it will be sent by means of the GSM transceiver to a server. The server stores everything in a database. Before storing the data, the server can calculate the position on base of the GSM data by means of triangulation. This position is not as accurate as the position got by the GPS system, but the signal is more reliable in case of GPS signal loss. The position of any car can be requested by any mobile device."[1]

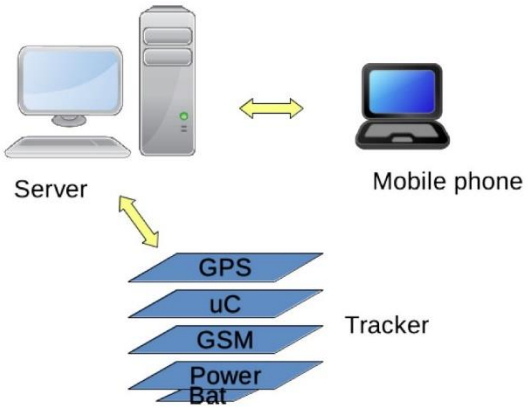

Figure 1. The complete tracking system

B. Server overview

A server is a piece of computer hardware or software (computer program) that provides functionality for other programs or devices, called "clients". This architecture is called the client–server model. Servers can provide various functionalities, often called "services", such as sharing data or resources among multiple clients, or performing computation for a client. A single server can serve multiple clients, and a single client can use multiple servers. A client process may run on the same device or may connect over a network to a server on a different device.[6]

A database server is a server which uses a database application that provides database services to other computer programs or to computers, as defined by the client–server model.[7] MySQL relies exclusively on the client–server model for database access.

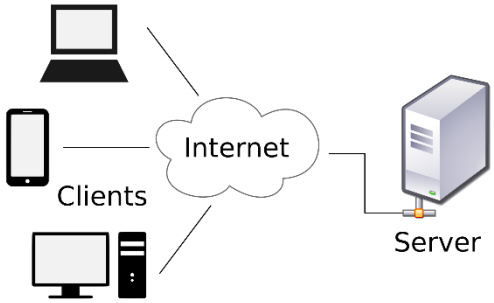

Figure 2: Client server computer module

Users access a database server either through a "front end" running on the user's computer – which displays requested data – or through the "back end", which runs on the server and handles tasks such as data analysis and storage.[8]

III. IMPLEMENTATION AND FUNCTIONALITY TEST

A. Server overview and setup

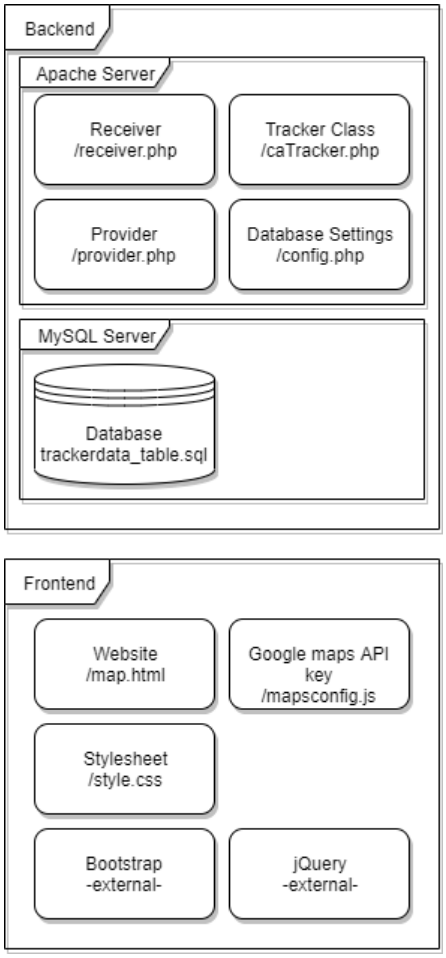

Figure 3. Structure/files of the system

As seen in Figure 3 the Server can be divided into two parts, the Backend and Frontend.

The implemented server uses a MySQL and Apache server running on a host pc. For simplicity we used the package XAMPP v7.4.3 to get those two mentioned servers up and running.

Due to the small amount of files used in this project all source files were put into the same directory. That allows a quick overview.

To provide a simple and easy setup of the server there are two files which must be renamed (postfix .sample must be removed) and edited:

`config.php.sample`
`mapsconfig.js.sample`

The first file contains the settings of the database, while the second file contains the Google maps API key.

B. Backend

The backend is responsible to receive tracker data, which is stored in a database, and providing this stored data again to be used by other applications.

a) Database

The database can have multiple tables with a simple structure, which contains only the necessary columns. To create a table with the needed structure the following command can be used:

```
CREATE TABLE `trackerdata_` (
 `id` INT(11) UNSIGNED NOT NULL AUTO_INCREMENT,
 `datetime` DATETIME NOT NULL,
 `gpstime` TIME NOT NULL,
 `lat` DOUBLE NOT NULL,
 `lon` DOUBLE NOT NULL,
 `gsm` TEXT NULL,
 PRIMARY KEY (`id`)
) ENGINE = InnoDB;
```

As rule for naming the tables we choose a combination of the serial number of the tracker and the prefix `trackerdata_`. Even though the serial number is not needed and can be left out it helps to determine which datapoints correspond to which tracker, in the case multiple trackers are used.

In the case of working with the database the file `config.php`, already mentioned above, is required once in a .php file. It contains all the necessary connection parameters.

b) Receiver

The receiver works by receiving a HTTP Get requests from the tracker device:

`/receiver.php?serial=&gpstime=&lat=&lon=&gsm=`

As one can see there are multiple parameters which can be set and sent to the server. For security reasons this input should be filtered and sanitized to prevent e.g., cross site scripting.

Due to the object-oriented programming approach (which helps to program modular) a class named Tracker is being used (`require_once('caTracker.php');`):[9]

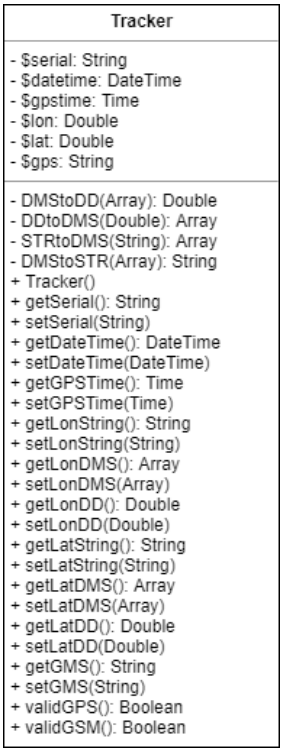

Figure 4. UML Diagram of the Tracker class

Every received parameter is converted and being stored into a Tracker object. The Tracker class provides helpful functions to handle the data. Especially the longitude and latitude information, which can be received as a string, must be converted into a double variable type.

The data of the object is stored into the database. The existence of required table is checked first and created if necessary.

The success or an error is returned at the end of the php script.

c) Provider

The provider in our backend can be called by:

`/provider.php?key=`

It creates a bridge between the received tracker data stored in the database with other applications (like the website - frontend). It returns all the data stored in the database of a given tracker in a Json encoding format.[10]

C. Frontend

To display the data, received from the tracker and stored into the database, a simple website is being used. This site can be called by:

`/map.html`

a) Design

The visual design is created with CSS and HTML. As base of the design the Bootstrap Library is used, which simplifies the development by providing a grid system. This allows a quick and more responsive web design.[11] Further CSS adjustments are being done in a separate stylesheet included in the `/map.html` called `/style.css`.

The site consists out of tree main sections. The first one is containing the Google maps map:

```
<!-- google maps -->
<div class="col-12" id="map"></div>
```

The second one contains a text input for the serial number of the tracker and a button to request the tracker data:

```
<!-- serial number -->
<div class="input-group col-12">
 <label for="serial" class="input-group-
text">Serial number</label>
 <input name="serial" id="serial_in" type="text
" class="form-
control" placeholder="serial number of tracker"
/>
 <input name="submit" id="submit_in" type="butt
on" class="btn btn-outline-
secondary" onclick="requestData()" value="Fetch
 Tracker Data" />
</div>
```

And finally, the third section contains the table with the tracker data:

```
<!-- table for data -->
<div class="col-12">
 <table class="table table-
fixed" id="datatable">
  <thead>
   <tr>
    <!-- column description -->
    <th class="col-1">#</th>
    <th class="col-2">Server DateTime</th>
    <th class="col-1">Satellite Time</th>
    <th class="col-2">Longitude</th>
    <th class="col-2">Latitude</th>
    <th class="col-4">GSM Data</th>
   </tr>
    </thead>
    <tbody id="data_items">
    <!-- sample item -->
    <tr class="item" id="proto_item">
    <td class="col-1">0</td>
    <td class="col-2">No Data</td>
    <td class="col-1">No Data</td>
    <td class="col-2">No Data</td>
    <td class="col-2">No Data</td>
    <td class="col-4">No Data</td>
   </tr>
  </tbody>
 </table>
</div>
```

b) Data management

The data management on the website is being done by JavaScript with the use of the library jQuery [12]. This library is lightweight with a "write less, do more" approach. Its purpose is to make it much easier to use JavaScript by offering simple methods, which normally require many lines of code.[13]

To fetch tracker data from the server the button 'Fetch Tracker Data' must be pressed. It calls the data provider `/data_bridge.php?serial=` of the backend and receives the Json encoded data of the tracker with stated serial number.

When adding the data to the HTML table `datatable` the placeholder item `proto_item` must be hidden first. Afterwards all the other code is being added by iterating over the acquired data of the backend and inserting it into the HTML table.

Also, the selection of the shown position markers on the Google maps map is being managed by a JavaScript function, which gets triggered when clicked onto an element in the table.

When data of a tracker with another serial number is requested, then the map as well as the table is cleared and set into the initial state.

c) Google maps

To be able to use Google maps a Google API key is mandatory.[14] For security reasons the request (contains the API key) to load the Google maps script is stored in the separate file `mapsconfig.js`. To have Google maps on the website it is necessary to not only load the Google maps script, but also implement the mandatory JavaScript function `initMap()`. It contains the basic options for displaying the map like the element name (in our case the name is `map`), where the map should be implemented on the site.

Position marker on the Google maps map (pin) are stored in an array and can have the following parameters: latitude, longitude and title.

By clicking on a data row of the `datatable` the GPS data gets either added to the `markers` array by a JavaScript function or removed. This `markers` array gets loaded onto the map and all the pins get reloaded. Additionally, the elements in the table change the background to identify which elements are currently shown on the Google maps map.

Further information regarding the Google maps JavaScript API can be found in their documentation.[15]

D. Functionality test

The functionality can be tested by simply launching the server and sending a GET request to the location described above. For easy testing, the `index.php` file contains the following examples (one for each test) which can be uncommented to be redirected to the link and executed.

a) Testing receiver

```
// Receiver example
header("Location:receiver.php?serial=&gpstime=1
11418.0&lon=9G38'43.8''E&lat=47G48'29.4''N&gsm=
5 45 1 0 E-Plus; MONI: N1 77 E720 E796 980 -
74dbm 25 19; MONI: N2 70 E720 98A4 685 -
75dbm 30 24; MONI: N3 74 E720 4B56 982 -
79dbm 20 14; MONI: N4 36 E720 4D2E 669 -
81dbm 24 18; MONI: N5 75 E720 9A7F 690 -
82dbm 23 17; MONI: N6 31 E720 8B39 984 -
87dbm 18 12;");
```

The code above should redirect to the `/receiver.php` with parameters and result into a successful database table insertion (Figure 5, 6).

Data received & stored

Figure 5. Result of the requested `receiver.php`

id ▲ 1	datetime	gpstime	lat	lon	gsm
1	2021-05-30 11:55:44	11:14:18	47.808'	9.6455	5 45 1 0 E-Plus; MONI: N1 77 E720 E796 980 -74dbm ...

Figure 6. Database entry

b) Testing provider

```
// Provider example
header("Location:provider.php?serial=");
```

The code above should redirect to the `/provider.php` with parameters and return the data of the table connected to the tracker with the given serial number (Figure 7).

[{"id":1,"datetime":"2021-05-30 11:55:44","gpstime":"11:14:18","lat":47.808166666667,"lon":9.6455,"gsm":"5 45 1 0 E-Plus; MONI: N1 77 E720 E796 980 -74dbm 25 19; MONI: N2 70 E720 98A4 685 -75dbm 30 24; MONI: N3 74 E720 4B56 982 -79dbm 20 14; MONI: N4 36 E720 4D2E 669 -81dbm 24 18; MONI: N5 75 E720 9A7F 690 -82dbm 23 17; MONI: N6 31 E720 8B39 984 -87dbm 18 12';"}]

Figure 7. Output of the requested `provider.php`

c) Testing website

```
// Show map
header("Location:map.html");
```

The code above should redirect to the `/map.html` and open the website. Now the data of a tracker can be fetched (Figure 8).

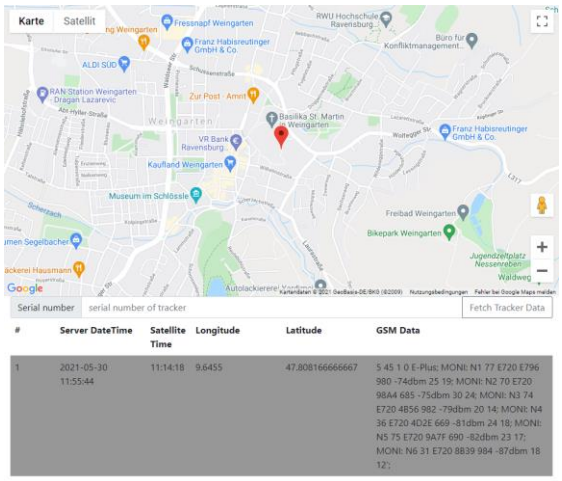

Figure 8. Website `map.html` with fetched data (here serial number = NULL) shown on the map

IV. CONCLUSION AND FURTHER WORK

A. Conclusion

So far, we have shown a server that displays the GSM information and the GPS location on a map, which uses data sent from GPS/GSM module through a HTTP GET request. All the capabilities shown were implemented on an Apache and MySQL server with help of PHP.

B. Further work – Backend

A full security check of the system would be necessary at some point to figure out and fix yet undetected safety concern issues, to provide stability and reliability. Because the Frontend could become a target of individuals/organizations trying to access the Backend, hence the Databases they contain. A design pattern focused on security most be approach to protect the valuable data. This could be achieved by understanding the risks and taking them into account while designing the system, this means implementing design mechanism against them.

C. Further work - Frontend

Further possibilities regarding the visualization are nearly endless. Some could e.g., connect multiple datapoints with lines to generate a route, with the help of Google Maps layers. Also, some settings to filter the data by time, date, region, etc. would be a neat and helpful feature. Those filters could be done completely by JavaScript or by the SQL request of data providing PHP site. The better choice needs to be determined and is mainly affected by the amount of datapoints.

D. Further work – GSM position localization

An interesting approach would be to use triangulation to get the position by the GSM data. To do this the cellphone tower location is needed and the strength between the GSM module and the tower. To get the location, the project OpenCellid, which offers an API, can be used. But to get the position of a cellphone tower the mobile country code (mcc), the mobile network code (mnc), the local area code (lac) as well as the base station id (cellid) is required and must be known.[16,17]

REFERENCES

[1] G. Benz, and A. Siggelkow, ”Implementation of a GPS and GSM module into a Zynq Z7 SoC based emulator tracking system“, MPC-WORKSHOP FEBRUAR 2020.

[2] https://httpd.apache.org/docs/, viewed 01 June 2021

[3] https://dev.mysql.com/doc/, viewed 01 June 2021

[4] https://github.com/maxux/gps-server, viewed 01 June 2021

[5] https://github.com/bmellink/GPS_Tracker_Server, viewed 01 June 2021

[6] *Windows Server Administration Fundamentals.* Microsoft Official Academic Course. 111 River Street, Hoboken, NJ 07030: John Wiley & Sons. 2011. pp. 2–3. ISBN 978-0-470-90182-3.

[7] Thakur, Dinesh. „What is a Database Server“, ecomputernotes, https://ecomputernotes.com/fundamental/what-is-a-database/what-is-a-database-server, viewed 01 June 2021

[8] “Database server”, Wikipedia, https://en.wikipedia.org/wiki/Database_server, viewed 01 June 2021

[9] “What is Object Oriented Programming? OOP Explained in Depth”, educative, https://www.educative.io/blog/object-oriented-programming, viewed 01 June 2021

[10] “JSON”, Wikipedia, https://en.wikipedia.org/wiki/JSON, viewed 01 June 2021

[11] https://getbootstrap.com, viewed 01 June 2021

[12] https://jquery.com, viewed 01 June 2021

[13] “jQuery Introduction”, W3Schools, https://www.w3schools.com/jquery/jquery_intro.asp, viewed 01 June 2021

[14] https://developers.google.com/maps/documentation/javascript/get-api-key, viewed 01 June 2021

[15] https://developers.google.com/maps/documentation/javascript/overview, viewed 01 June 2021

[16] https://opencellid.org, viewed 01 June 2021

[17] http://wiki.opencellid.org/wiki/API#Getting_cell_position, viewed 01 June 2021