**Network Communication Protocols  
  
Communication devices by order:** Arduino, Raspberry Pi and the Server.

Arduino to Raspberry Pi Protocols:

Send to Raspberry Pi the seats:  
s/e – Sending data OR error.  
L1 – Length of responsible line.  
R – Line number.  
L2 – Length of amount of chairs -> Length (Amount of chairs)  
A – Status of every seat. 0 or 1.  
  
String summary: s;L1;R;L2;A .   
In case of an error: e.

Example: s;1;1;2;10 (sending, len(line\_1),line 1, 2 chars of data, 10)

Raspberry Pi to Server Protocols:  
Similar to the Arduino to Server Protocols style but changing the first letter.  
Send to the Server the data:  
‘u’ or ‘c’ – update, close (socket) – if c is the first letter, no need in the rest of the msg.  
t - Type of vehicle.  
C - Company of vehicle.  
n - Number of vehicle  
L1 – Length of amount of chairs -> Length (Amount of chairs)  
A – Status of every seat. 0 or 1.  
Because we have a couple of lines ( of seats ), we send a whole line of flags ( of the seats ), and then separating every line with the character: ‘\_’ .  
  
String summary: u/c;t;C;n;L1;A

Example:  
Updating seats: u;bus;egged;263;len(0111\_1111\_1111\_1111);0111\_1111\_1111\_1111   
Closing socket: c (and that’s all).

Server to Raspberry Pi Protocols:

r – received ( when the pi sends data to server, returns ‘r’ for ack. If RPI doesn’t get ‘r’, it doesn’t continue ).  
e – error ( asks the RPI to send again the string ).

Client to Server Protocols:  
l – login (username, password, email).  
r – register (username, password).  
c – close (socket).  
v – view (vehicle information).  
g – get seats (inside the vehicle information section).  
E – exit from server log (protects the data of the server).   
  
**How to identify a vehicle: vehicle\_type + vehicle\_company + vehicle\_number**  
  
Login: l;length(username);username;length(password);password  
  
Register: r;length(username);username;length(password);password;length(email);email  
  
View Vehicle: (first screen of the vehicle whenever a client requests a vehicle) v;len(vehicle\_type);vehicle\_type;len(vehicle\_company);vehicle\_company;len(vehicle\_number); vehicle\_number  
  
Get Seats Data: (a button inside the vehicle information – view vehicle to get the seats) g;len(vehicle\_type);vehicle\_type;len(vehicle\_company);vehicle\_company;len(vehicle\_number); vehicle\_number  
  
Exit: E;len(username);username

Server to Client Protocols:

Login: l;success\_flag(0 or 1) .  
  
Register: r;success\_flag(0 or 1) .  
  
Get Seats Data:  
g;len(data);1\_010111|2\_011111|3\_0111 (number of line and after comes the status)  
(\_ indicates the same line inside db but to another row)  
(| indicates moving to the next line )  
  
View Vehicle: **NOTICE THIS** – the server will return the delay given from the assigned time, if the user wants a bus from 8:00, and the delay is 0-5-20, then on the client’s screen the data will be showed like this: 8:00, 8:05, 8:20.  
v;len(total startTime\_EndTime); 8:00\_8:30|8:35\_9:05;len(data);Rabin\_0|Big\_25  
(data here is the delay table).