Super Senior Integration Project "Architecture & Platforms"

Imérir - www.imerir.com / Fall 2015

Instructors:

- Grabolosa, Pierre pierre.grabolosa@imerir.com
- Himmelstein, Jesse jesse.himmelstein@imerir.com
- Madeline, Blaise blaise.madeline@imerir.com

Context

Each **monitor** displays a graphical representation of an urban road system. Each **monitor** is responsible for the display of a specific region. Which region and its road system description are obtained from a WebService in JSON.

The regions are connected through *bridges* which let vehicles travel from one region to another.

Once all the monitors are setup and ready, we can simulate vehicles travelling the road system. More specifically, the vehicles are **cabs** which can be commanded to go to a specific location. Using algorithms learned in graph theory or AI courses, you must resolve a path for the **cab** to go from its current location to the requested location. (This assumes the **cab** is available; if already on route then the **cab** must first reach its current destination before acquiring a new one).

Cabs have an odometer (travelled distance, reset to zero upon acquiering a new destination) and a binary availability state (busy/free). A **cab** can only seek one destination at once. Each **cab** will have a dedicated device to report its state and the odometer's value. Provided the **cab** is available, this device will also be used by human operators to let the **cab** know it should acquire the next available destination (otherwise the **cab** just remains at its current location after reaching a destination).

Monitors will update continuously the display of the location of the **cabs**. To do so, instead of polling frequently the WebService, the **monitors** will be notified via a Pub/Sub implemented with WebSockets.

The simulation has as many **cabs** as there are counters connected.

Human end users have the following means of interaction with the system :

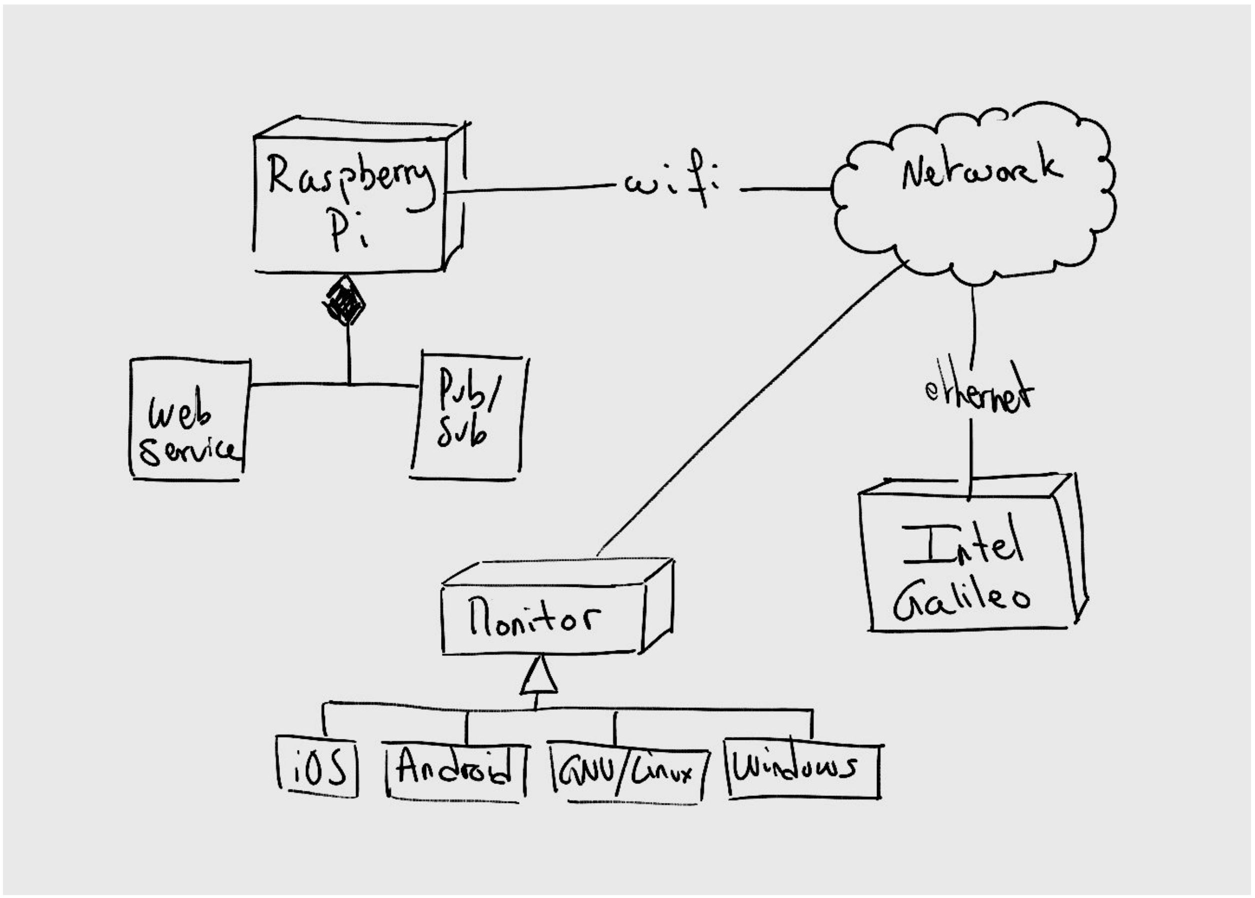
- request a new destination (through the **monitors** by click or touch),
- assign a **cab** the next destination in cue (provided the cab is available and using the Intel Galileo LCD shield buttons).

Technical Contraints

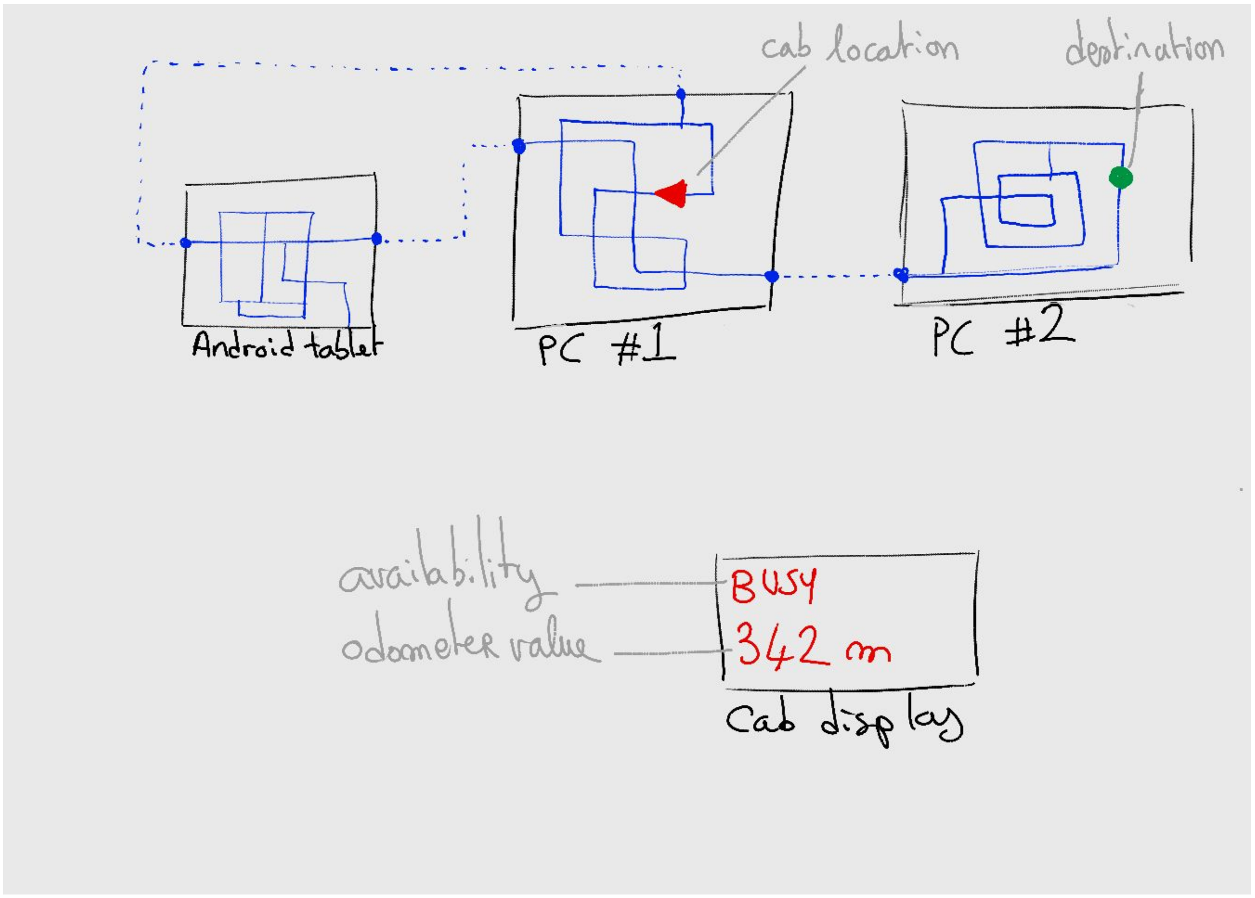
- Server : Raspberry Pi
    - HTTP WebService
        - Flask/Python or Apache HTTPD either mod_php or CGI Python.
    - Pub/Sub Service
        - Use the WebSockets protocol. Language? Your choice.
    - Database? Your choiec. Sqlite, mariadb/mysql are fine.
- Clients
    - **Cab device** : Intel Galileo (Arduino IDE)
    - **Monitors**; two kinds must be implemented:
        - mobile platform (iOS or Android; dependent on what will be assigned to your team at the start of the project)
        - PC (GNU/Linux or Windows): either Java, Qt or .Net

Illustrations

Global architecture



Mockup



Assigned Work

Web Service

You must design and implement a Web service. Document properly the URLs to access the resources and the formats used to describe them.

The WebService is used to:

- register clients (**monitors**, **cab** devices, ...), assign them an **id** and give them whatever information is needed to subscripte to the Pub/Sub service.
- retrieve the details of the urban road system,
- retrieve the details of connected devices,
- ...

The WebService may be used for supervision/debugging needs.

Pub/Sub

The only constraint: use WebSockets. Use this service to notify all subscribers of relevant events (change in location, for example).

Monitor

At least two platforms must be targetted.

- Mobile: iOS or Android (depending availability/attribution at the beginning of the project),
- Desktop computers: your choice of GNU/Linux or Windows.

Cab device

An Intel Galileo will be provided along with an LCD shield. You must use the Ethernet port of the Galileo to retrieve data from the WebService/WebSocket. A serial connection with the computer via USB may only be used for debugging.

SCM and Evaluation

Instead of Redmine and SVN, you will be using Github.

You do not have to produce a DOP or an oral presentation. Instead, you will demo your work on the last day and we will conduct two code reviews through our the project.

Instructors will evaluate the quality and the degree of completion of your work. You will also be individually interviewed during the code reviews. Lastly, your Github repository will be scrutinized: we will check authorship, frequency, comment quality, and volume of commits.

Despite lack of DOP requirement, you are still expected to document your project properly. Anyone should be able to retrieve your sources, compile them (as necessary) and deploy them to a working state. The code itself must be properly documented. This too is part of your final grade.

Logistics and evolving assignment

There will be nine (9) teams of three students each.

Instructors will be available in room B14 (see e-learning.imerir.com for a detailed schedule) to answer any question you may have regarding the project. In room B14 will be sheets that you must sign at 8h15 and at 17:45.

This document is intentionnally minimalistic (technically and functionnally) to give you room for thought. When in doubt, please check with the instructors.

This document (and therefore the assignment!) may evolve during the project, particularly to answer the most frequent questions (but not limited to!). Check regularly this document for updates.

Appendices

- e-learning.imerir.com
- presentation slides
- Arduino and Intel Galileo documentation.