Super Senior Integration Project

Pierre Grabolosa | Jesse Himmelstein | Blaise Madeline

Imérir 3A, October 2015

Before we start… team up!

- Rules
  - teams of 3
  - the team must be mixed (student / apprentice)
  - 9 teams total

- Advice
  - balance your skills
  - required skills
    http based services, client-server programming, GNU/Linux systems, graphics programming, mobile programming (iOS/Android), graph theory/AI

The project

The assignment

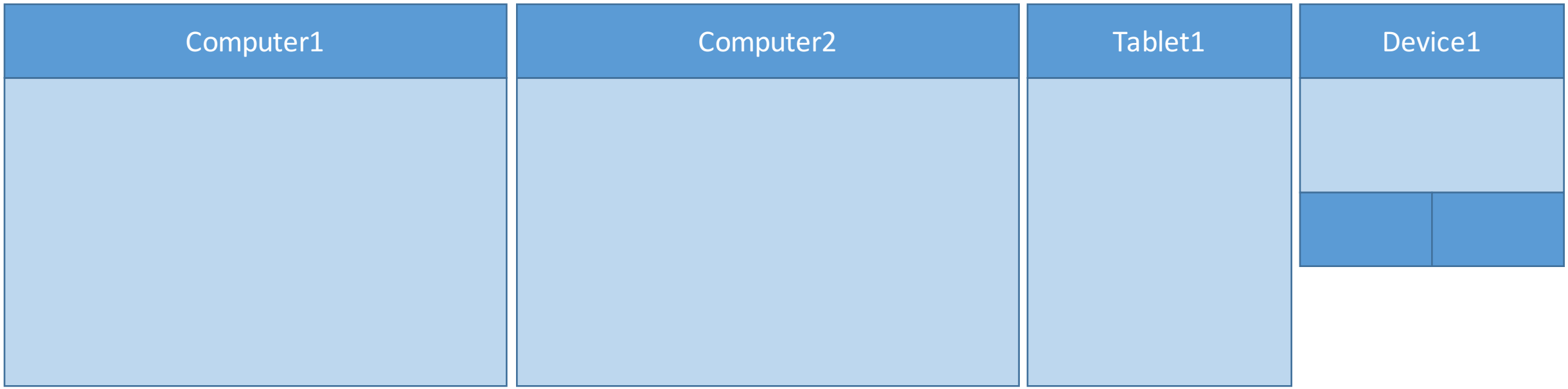
Cab Simulation

- There's a city
- There're cabs
- Cabs go from their current location to a requested location
- Multiple devices display and make this simulation possible

Cab Simulation

- City → Graph
- Cab→ Agent
- Going someplace → Navigation / Path finding
- Multiple devices → Networking / Notification system

Wireframe



Gobal architecture

Sequence » Loading

- Server starts
- Monitors register until server has enough monitors to represent all parts of the city → when a monitor registers it receives a description of its urban area
- Cab device registers → you may assume only one such device (BONUS: handle more than one)
- *Simulation* can start / starts

Sequence » *Simulating*

- Whenever a destination is picked (click or touch on a monitor), it is added to a queue of targets and the cab is notified
  - early simplification: closest vertex
  - BONUS: closest road point
- The cab device continuously displays:
  - its state (busy/free),
  - the distance travelled since the last course (early simplification: vertex count ; BONUS: actual distance) – Whenever the cab moves, the monitors update its location
  - the length of the queue
- Whenever the cab is free, the user may choose to accept or deny the next destination from the queue.
  - Whenever the queue changes, the monitors update their display

City Description

- The server loads a configuration file which contains the city description. Recommended format:

```
rootObject -> {
  "areas": [area...],
  "cabInfo": [cabInfo...],
  "cabQueue": [cabRequest...]
}

area -> {
  "name": string,
  "map": {
    "weight": {"w": #, "h": #},
    "vertices": [vertex...],
    "streets": [street...],
    "bridges": [bridge...],
  }
}

vertex -> {
  "name": string,
  "x": # between 0 and 1,
  "y": # between 0 and 1
}
```

```
street -> {
  "name": string,
  "path": [vertex.name...],
  "oneway": true|false (BONUS)
}

bridge -> {
  "from": vertex.name,
  "to": {
    "area": area.name,
    "vertex": vertex.name},
  "weight": #
}

cabInfo -> {
  "odometer": #,
  "destination": null | cabRequest,
  "loc_now":  locVertex | locStreet,
  "loc_prior":locVertex  | locStreet
}
```

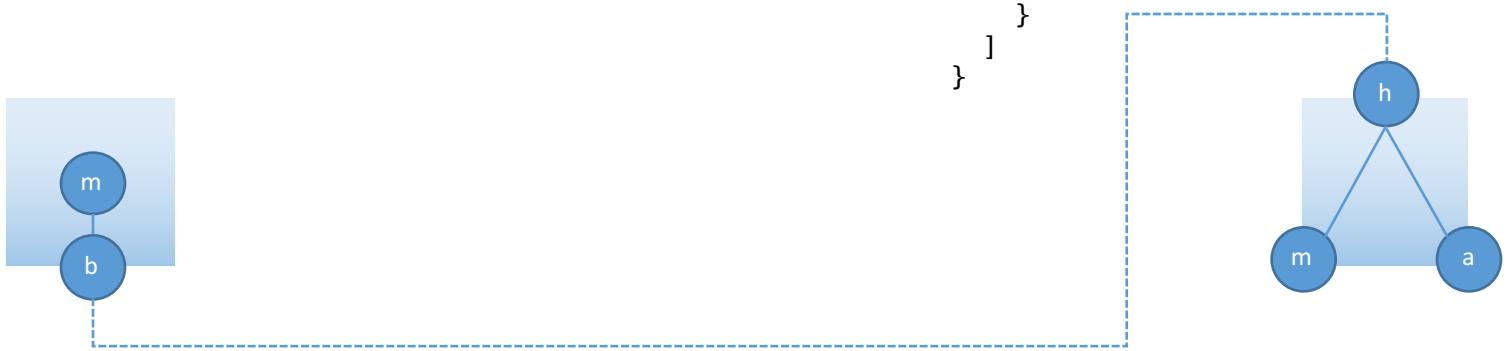
```
locVertex -> {
  "area": area.name,
  "locationType": "vertex",
  "location": vertex.name
}

locStreet-> {
  "area": area.name,
  "locationType": "street",
  "location": {
    "from": vertex.name,
    "to":   vertex.name,
    "progression": # between 0 and 1}
}

cabRequest -> {
  "area": area.name,
  "location": locVertex | locStreet
}
```

City Description Example

```
{
  "areas": [
    { "name": "Quartier Nord",
      "map": {
        "weight": {"w": 1, "h": 1},
        "vertices": [
          {"name": "m", "x": 0.5, "y": 0.5},
          {"name": "b", "x": 0.5, "y": 1}
        ],
        "streets": [
          {"name": "mb", "path": ["m", "b"], "oneway": false}
        ],
        "bridges": [
          { "from": "b",
            "to": {
              "area": "Quartier Sud",
              "vertex": "h"},
            "weight": 2
          }
        ]
      }
    },
    { "name": "Quartier Sud",
      "map": {
        "weight": {"w": 1, "h": 1},
        "vertices": [
          {"name": "a", "x": 1, "y": 1},
          {"name": "m", "x": 0, "y": 1},
          {"name": "h", "x": 0.5, "y": 0}
        ],
        "streets": [
          {"name": "ah", "path": ["a", "h"], "oneway": false},
          {"name": "mh", "path": ["m", "h"], "oneway": false}
        ],
        "bridges": [
          { "from": "h",
            "to": {
              "area": "Quartier Nord",
              "vertex": "b"},
            "weight": 2
          }
        ]
      }
    }
  ]
}
```

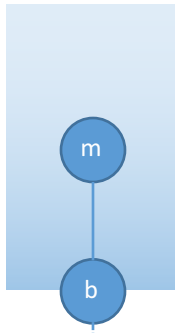
City Description Example

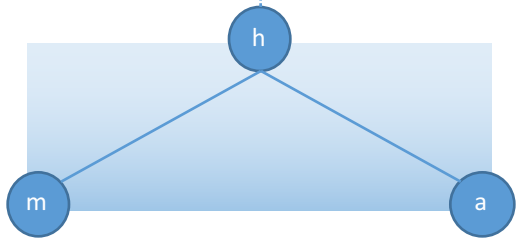
```
{
  "areas": [
    { "name": "Quartier Nord",
      "map": {
        "weight": {"w": 1, "h": 1},
        "vertices": [
          {"name": "m", "x": 0.5, "y": 0.5},
          {"name": "b", "x": 0.5, "y": 1}
        ],
        "streets": [
          {"name": "mb", "path": ["m", "b"], "oneway": false}
        ],
        "bridges": [
          { "from": "b",
            "to": {
              "area": "Quartier Sud",
              "vertex": "h"},
            "weight": 2
          }
        ]
      }
    },
    { "name": "Quartier Sud",
      "map": {
        "weight": {"w": 1, "h": 1},
        "vertices": [
          {"name": "a", "x": 1, "y": 1},
          {"name": "m", "x": 0, "y": 1},
          {"name": "h", "x": 0.5, "y": 0}
        ],
        "streets": [
          {"name": "ah", "path": ["a", "h"], "oneway": false},
          {"name": "mh", "path": ["m", "h"], "oneway": false}
        ],
        "bridges": [
          { "from": "h",
            "to": {
              "area": "Quartier Nord",
              "vertex": "b"},
            "weight": 2
          }
        ]
      }
    }
  ]
}
```



City Description Example » affichage %


```
{
  "areas": [
    { "name": "Quartier Nord",
      "map": {
        "weight": {"w": 1, "h": 1},
        "vertices": [
          {"name": "m", "x": 0.5, "y": 0.5},
          {"name": "b", "x": 0.5, "y": 1}
        ],
        "streets": [
          {"name": "mb", "path": ["m", "b"], "oneway": false}
        ],
        "bridges": [
          { "from": "b",
            "to": {
              "area": "Quartier Sud",
              "vertex": "h"},
            "weight": 2
          }
        ]
      }
    },
    { "name": "Quartier Sud",
      "map": {
        "weight": {"w": 1, "h": 1},
        "vertices": [
          {"name": "a", "x": 1, "y": 1},
          {"name": "m", "x": 0, "y": 1},
          {"name": "h", "x": 0.5, "y": 0}
        ],
        "streets": [
          {"name": "ah", "path": ["a", "h"], "oneway": false},
          {"name": "mh", "path": ["m", "h"], "oneway": false}
        ],
        "bridges": [
          { "from": "h",
            "to": {
              "area": "Quartier Nord",
              "vertex": "b"},
            "weight": 2
          }
        ]
      }
    }
  ]
}
```



WebServer

- Your choice among
  - Python/Flask
  - Node.js
  - PHP (Flight?)
- Data must be exchanged in
  - JSON
- Limitations?

Pub-Sub

- Publisher-Subscriber
  - Observer Design Pattern
- Why isn't HTTP well suited?
- WebSockets

Monitors

- Platorms:
  - Tablet (Native): Android or iOS (choice may not be yours)
  - PC: .Net, Java or HTML5? (your choice!)

- Draw the aread scaled to fill the screen/window
- Click/touch → Cab request queue

Cab Device

- Galileo + Shields

- Capabilities
  - Ethernet
  - LCD Display (Liquid Crystal)
  - Buttons

- It is ok to only display the current/next vertices.
  It is better to display the actual travelled distance.

Integration project

- 4 platforms => 4 code bases => integration of 4 software
  - 2 protocols (HTTP + WebSockets)

- Wider reach
  - Raspberry Pi challenges
    - Raw system: installation/configuration
    - Two networks: Wifi (PC/Tablets) & Ethernet (Galileo)
  - Graph → Graph theory / AI

- ~~Redmine, SVN~~ → Github

Evaluation

- 1 group code reviews @ mid project
- 1 individual code review @ end
- 1 demo @ end

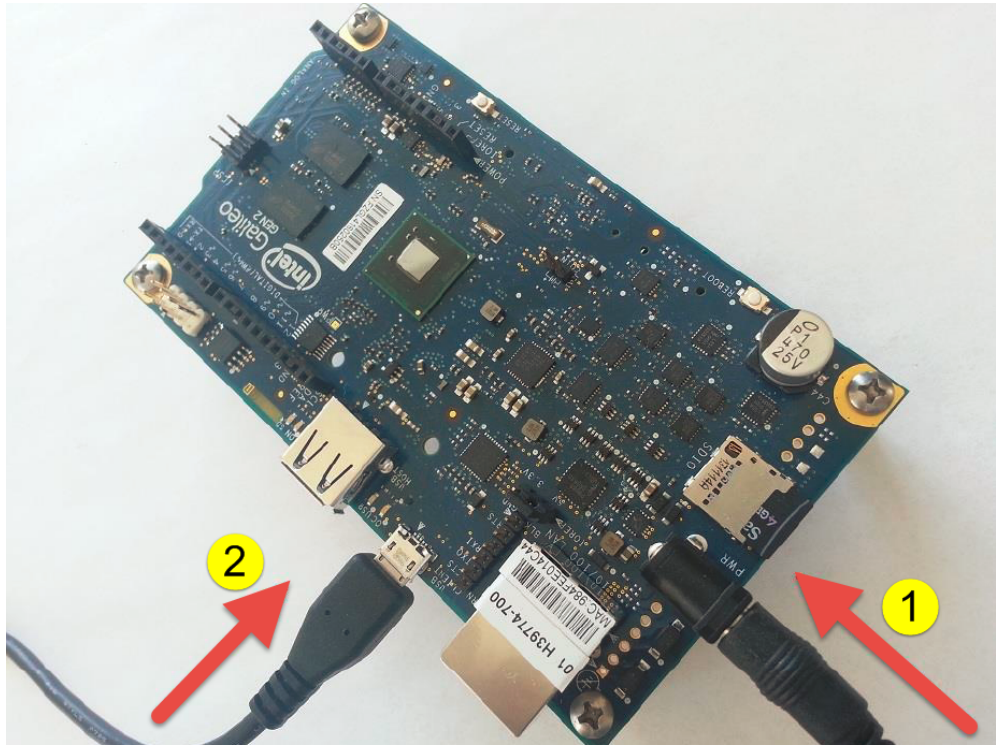
- SCM frequency/quality checked
- Code/Architecture quality matters
- Documentation matters

Evaluation » Passing grade

- To get a passing grade
  - Make the basic requirements work
  - Have a reasonably valid SCM/code quality/documentation record

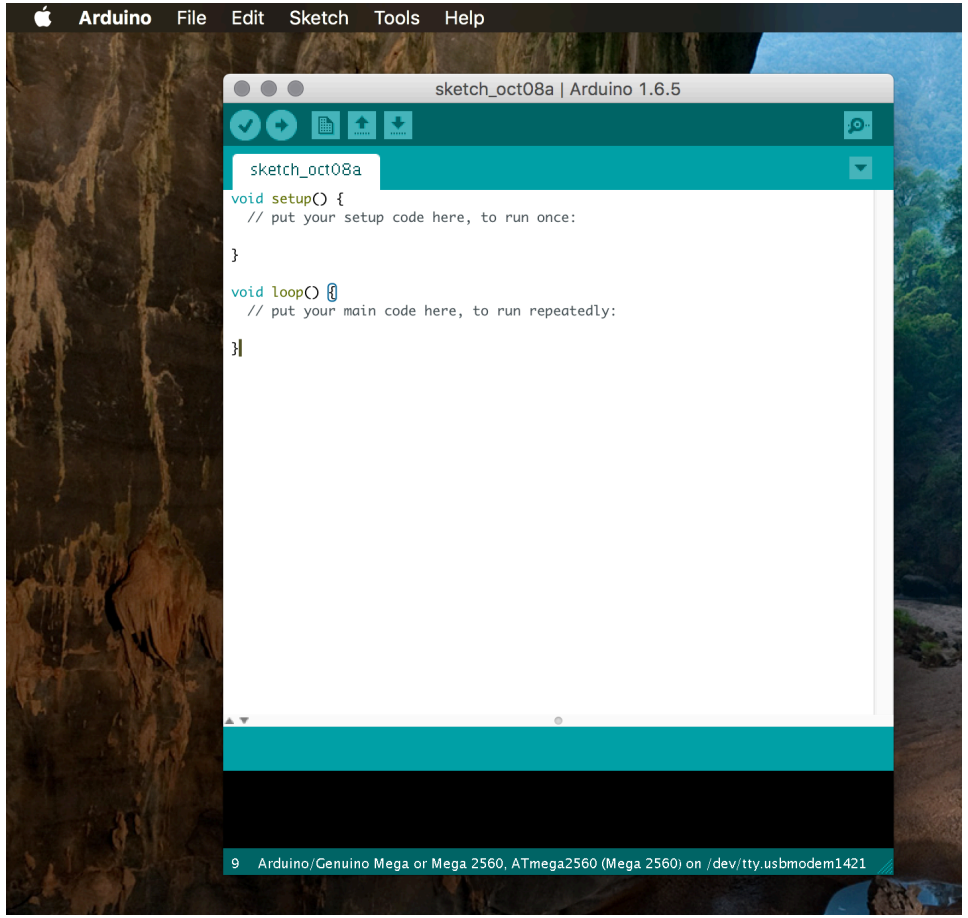
Galileo Setup

- **Always power the board before connecting USB!**

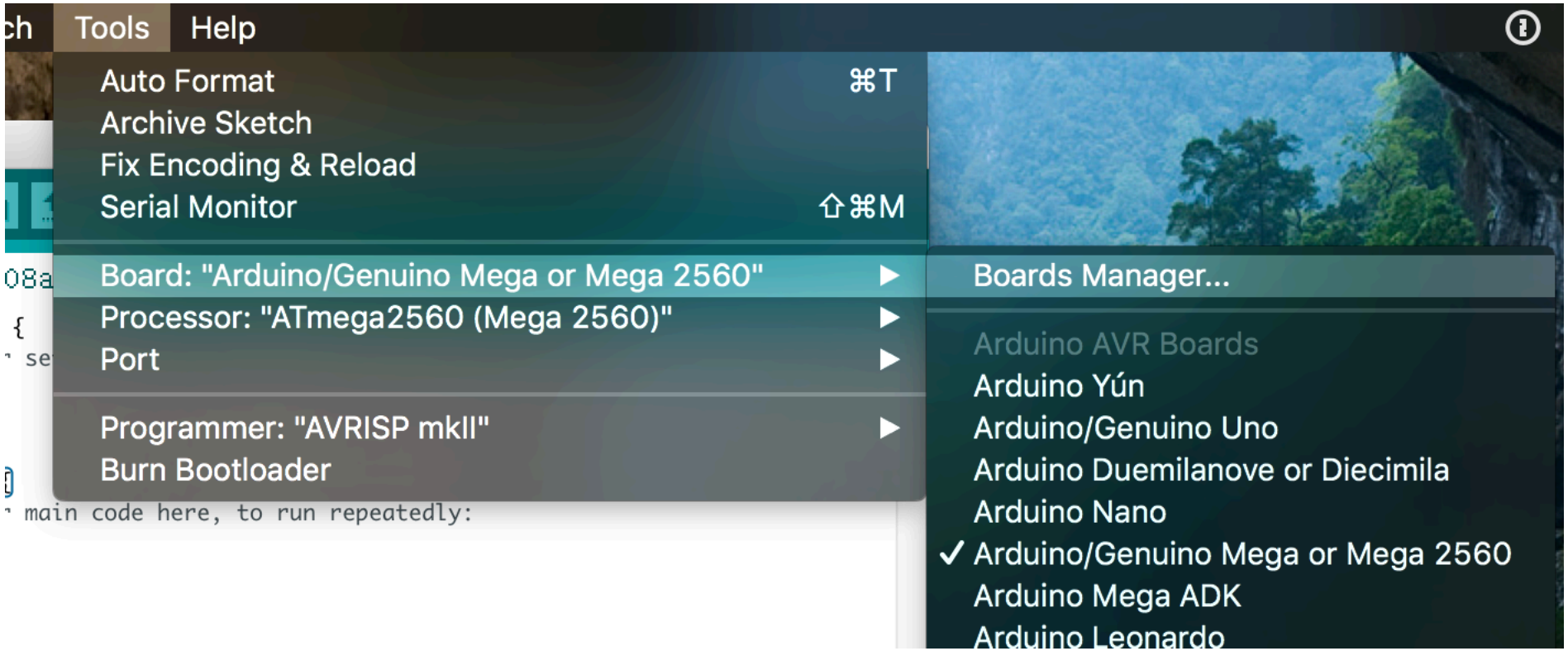


[Intel Website (Getting started, docs, examples…)]

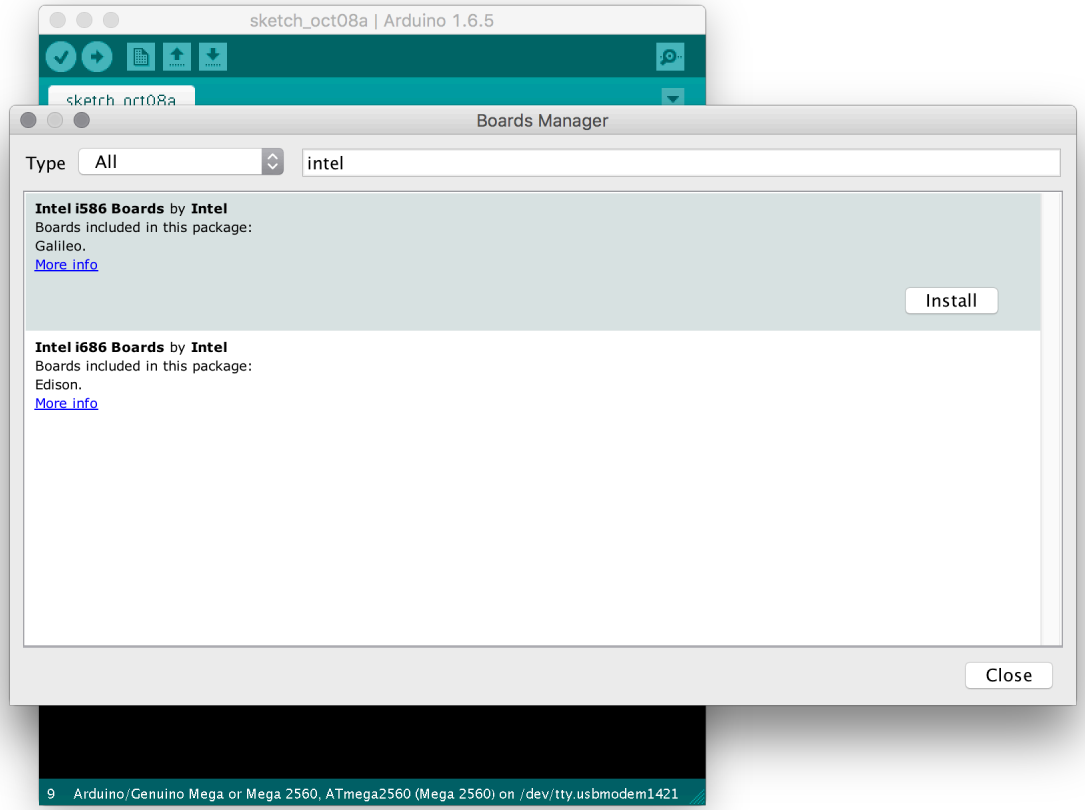
Galileo Setup » Arduino



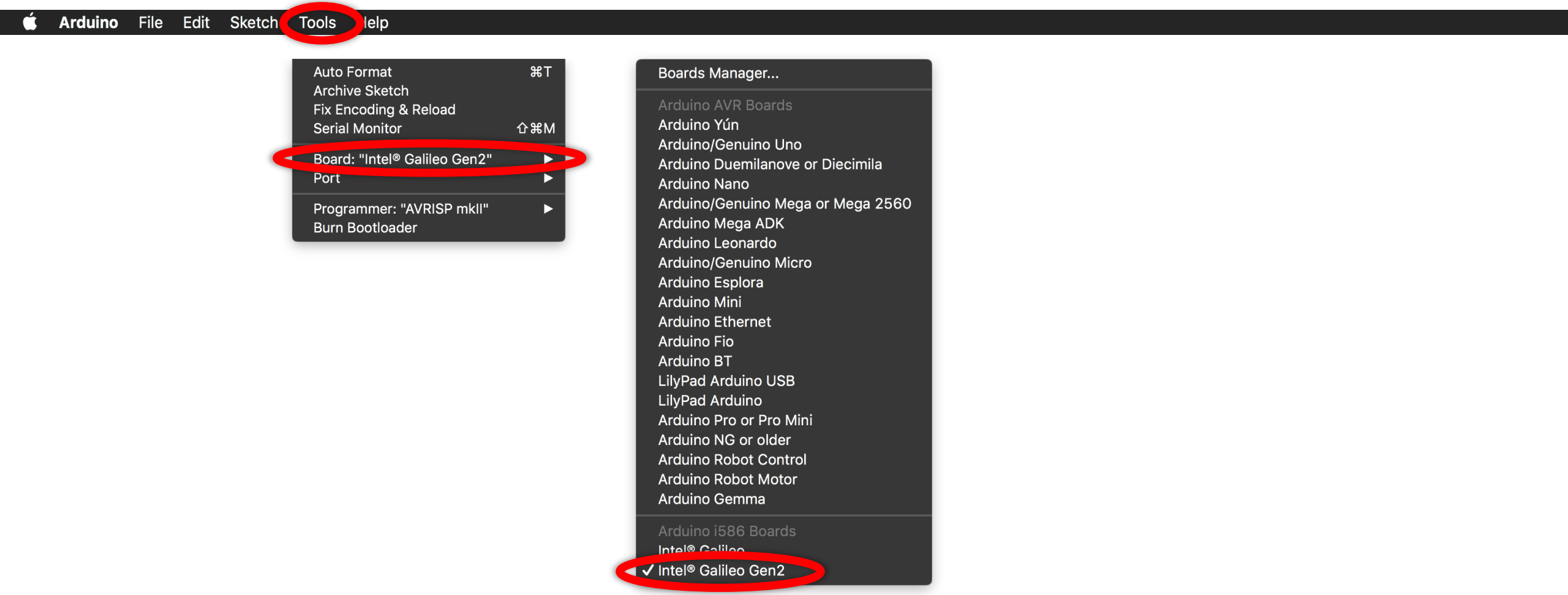
Galileo Setup » Arduino » Add the boards



Galileo Setup » Arduino » Install the definitions



Galileo Setup » Arduino » Set board type



Tight schedule

- D1 – Friday 9
- Monday 12 → Conferences
- D2 – Tuesday 13 … D9 – Thursday 22

- Not really 9 day, but 10, maybe 14…

- Today's priority: make sure the assignment is clear. Define and agree today on the protocols you'll be using. Set up your environments and make sure you get your devices working.
- Important: whenever you are stuck, ask questions → B14