SIS Project Assignment

This project requires the following software and is meant to run on **Ubuntu 20.04** (as provided in the computer rooms):

- C development tool (gcc, make, etc.)
- Python3 (optional)
- Matlab R2021b
- Webots R2023a

General information

This project will be carried out in teams of **three** students. The goal is for you to work autonomously as a team and solve the presented problem on your own. You will be able to get assistance from the teaching assistants during the lab sessions on Thursday.

Collaboration policy

This project aims to foster the collaboration capabilities of students to work as a team. Teams have been defined by considering student preferences and eventually approved centrally by the Head TA of the course. Therefore, by default, the submitted deliverables will be considered a product of the team, and students belonging to the same team will receive the same grade. We therefore highly encourage all group members to be fully involved in all parts of the project.

While intra-team collaboration is promoted, any inter-team collaboration is not allowed and, **in case of evidence of copies between teams, there will be severe penalties for all parties involved**. Additionally, we will **use automatic tools to detect plagiarism for all deliverables (code and report) compared to what was submitted in the current and previous editions of the course**. In case we detect any issue, there will be severe consequences, according to EPFL's plagiarism policy.

In case of issues within a group in terms of contribution fairness or disenrollment of a group member from the course, please contact the Head TA as soon as possible to find a solution (e.g., splitting the team or grading the individual contributions separately).

Getting Started

To start with the project, you will need to download the material available on Moodle. Download `material.zip` and extract it in your home directory. We recommend you use the provided files as a basis to get started with the project, although you are free to adapt, add, or change any file you want in the `controller` and `scripts` directories. Note that the files provided should offer you a suitable structure to start from.

You will find additional technical information useful for completing this project in the accompanying document `SIS_23-24_project_documentation.pdf`.

Project description

The task at hand consists of automating the inspection of a greenhouse with a ground robot. The greenhouse is composed of five rows of plants forming four alleys. The robot must acquire the following information:

- Light data from the lighting system.
- Indoor and outdoor temperatures broadcasted by static nodes, communicated over a serial infrared channel.

From this collected data, the robot must identify **in real-time** whether some lights present any defect, and classify the type of defect encountered. Note that we expect light in good condition will provide a continuous and stable lighting intensity. Also, the isolation of the greenhouse must be identified from the collected temperature data to assess whether it is in a good state, which can typically be done once the robot is done navigating the greenhouse.

To collect these data, the robot must be able to navigate the greenhouse, which is a cluttered environment. The robot should navigate from one alley to the next and drive along them, in a lawnmower fashion (see figure below). While moving around, the robot needs to estimate its position and orientation to localize the lights presenting faulty behaviors.

The two pictures below show the greenhouse and the path the robot is expected to take:

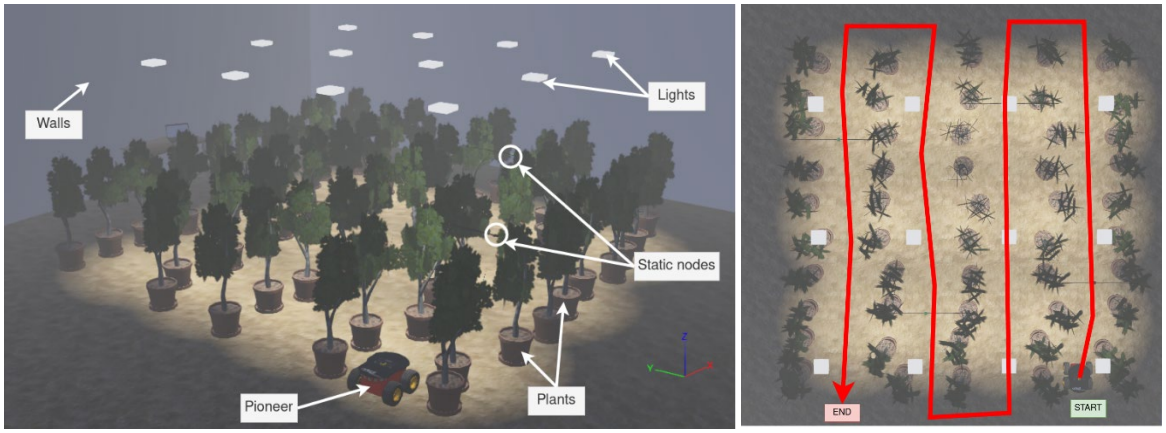
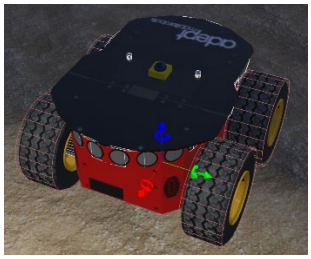


Figure 1. Greenhouse overview with robot starting location and expected robot path.

You will use the Pioneer 3-AT robot simulated in Webots:



It is equipped with the following sensors:

- Fifteen sonars (acting as proximity sensors); see section below for more information.
- Accelerometer (x: Forward, y: Left, z: Up), with standard deviation **0.05 [m/s^2]**.
- Gyroscope (same orientation as the accelerometer), with standard deviation **0.025 [rad/s]**.
- Light sensor (upward looking, hence along robot z-axis), with zero standard deviation.
- Wheel encoders, with zero standard deviation.

The robot starts at the origin of the world with zero heading; hence its initial pose is $(x, y, \theta) = (0,0,0)$, where θ is the robot's heading. Positions are expressed in meters, while the heading is expressed in radians.

The walls encountered at each end of the alleys are located at -1.3 (behind the robot when starting) and 7 (in front of the robot when starting) meters along the x-axis of the world, respectively.

The robot is operated as a differential drive robot (similar to the E-Puck you have seen in the labs) despite being equipped with four wheels. Hence, the wheels on each side always turn at the same rate. Note that despite having wheel encoders that exactly read the wheels' position, the wheels can slip. Make sure to include a certain degree of uncertainty in your odometry.

Have a look at the robot documentation for more details: https://www.cyberbotics.com/doc/guide/pioneer-3at?version=cyberbotics:R2023a.

Proximity sensor response

The proximity sensors allow the robot to detect obstacles all around it. Each sensor follows the response:

- Distance: `>5m` → measurement: `0`
- Distance: `0m` → measurement: `1024`

The response is linear between 0 and 5 meters. The measurement noise follows a standard deviation equal to **1%** of the current measurement value.

Serial communication

The greenhouse is equipped with four static nodes that broadcast information about the greenhouse. They are installed at fixed locations above the alleys (one node per alley, at an arbitrary location along the alley). Each static sensor broadcasts five values (of type double) in each packet:

- **sensor id** (integer converted to double)
- **x** location in the greenhouse in meters
- **y** location in the greenhouse in meters
- current **indoor temperature** $T(t)$ in degrees Celsius
- current **outdoor temperature** $T_O(t)$ in degrees Celsius

The nodes have a downward-looking cone of communication within which the robot can receive data packets. All nodes are located 1 meter above the ground, and their communication cone has an aperture of 1 radian. Note that the robot's light sensor is located on top of the robot, and the robot is 0.277 m tall. Furthermore, the robot can measure the signal intensity, which is inversely proportional to the square of the distance to the emitter.

Data logging

We provide you with some logging solutions to store data from the simulation into csv files conveniently. You can find more information about this in the project documentation.

Data will be stored in two main locations:

- `controllers/controller/data`: folder where your controller will log all relevant data you need for your analysis.

- `controllers/supervisor/data`: ground truth pose information is automatically logged by the supervisor, which you can use to compare your pose estimate pipeline. The collision count between the robot and the environment is also logged there.

Greenhouse thermal model

The greenhouse thermal model is defined as follows:

$$C\frac{dT(t)}{dt} = q_{heater} - \frac{T(t) - T_O(t)}{R}$$

where q_{heater} is the heat rate (power) produced by the heating system, $T(t)$ is the temperature in the greenhouse, $T_O(t)$ is the outdoor temperature, R is the thermal resistance and C is the thermal capacitance of the greenhouse.

The following characteristics are known:

- $C = 0.1[J/K]$
- $q_{heater} = 0.1[W]$

You will need to collect data from the static nodes to learn about the temperatures $T(t)$ and $T_O(t)$.

Note: these values are not realistic but allow us to have a model easier to handle within the scope of the considered scenario.

Minimal requirements

You must ensure the following features are implemented. Feel free to extend these minimal requirements with additional features, plots, etc., if you feel it would improve your solution or provide better insights into your results:

- **Online** (i.e., the robot must perform these actions in real-time):
    - The robot must be able to traverse each alley of the greenhouse without any collision and stop once all alleys have been traversed at least once.
    - The robot must print in the terminal each detected light when encountered, its status (whether being good or presenting a defect and its identified defect type), and its estimated location. For instance: `Detected light n°2, status: good, location: (3.5,2.8) m` Make sure, therefore *not to log too much information* in the terminal to not clutter the terminal and make it unreadable for the final demo!
- **Offline** (e.g., by running a script visualizing/analyzing the data logged by the robot once it is done navigating):
    - Provide one or several plots that adequately compare the ground truth trajectory with the one estimated by the robot. Visualize the uncertainty of the pose estimate as well. Report in a table the average pose estimate error along the whole trajectory for each state dimension, hence x, y and heading.
    - Provide a single plot where the x-axis is the x coordinate in the greenhouse, and the y-axis represents the y coordinate in meters. The plot must include the estimated and ground truth trajectories (solid and dashed lines, respectively), the position of all detected lights (as a scatter plot), and their defect classification (e.g., using color code on the scatter plot). Finally, report any collision position during the exploration (as a scatter plot). Do not forget to add all relevant labels to the plot.

- Identify the thermal resistance (isolation) of the greenhouse offline based on the data collected by the robot when receiving messages from the static nodes, as well as estimate the ideal thermal resistance for the given parameters and measurements. Compute the ideal value R^*, so that the temperature can be maintained given the provided heating system (assume constant heat input). Give the transfer function of the indoor temperature (i.e., the system's output) using the Laplace transform and report the step response for both R and R^*, where the input is the heat produced by the heater.

Submitting your project

You need to submit a **report** and all your **implemented code** for this project via Moodle as a **single** zip file named:

- **`SIS_22-23_Project_Group<GroupNo>.zip`**, containing:
  - **`material_Group<GroupNo>`**
  - **`SIS_23-24_Project_Report_Group<GroupNo>.pdf`**

The project is due on **Monday, December 18th, 23h59**. It must be uploaded on Moodle, at the designated location. You will be able to submit your project at any time and update the submitted material at any time until the deadline.

Code submission guidelines

We provide you with the project material in the `material.zip` archive. It is structured as follows:

- `material`
  - `controllers`
    - `controller` (robot controller, to implement)
      - ***`braitenberg.hpp`***
      - ***`controller.cpp`*** (this is the main file)
      - *`data/`* (folder in which the controller logs data as csv files)
      - ***`FSM.hpp`***
      - ***`kalman.hpp`***
      - *`Makefile`*
      - ***`odometry.hpp`***
      - ***`serial.hpp`***
      - ***`signal_analysis.hpp`***
    - `static_node`
      - *`static_node`* (static node controller, compiled)
    - `supervisor`
      - `data/` (folder in which the supervisor logs data as csv files)
      - *`supervisor`* (supervisor controller, compiled)
  - `libraries`
    - `kiss_fft` (C library for FFT)
    - `pioneer_interface` (robot API, included by `controller.cpp`)
    - `utils` (contains a utility to log data as csv file, included by `controller.cpp`)
  - `protos` (Webots specific files)
  - `scripts`
    - `matlab`
      - ***`load_data.m`***

        - `python`
            - ***`load_data.py`***
    - `worlds` (Webots world file)
        - `greenhouse.wbt`

You are expected to submit the **exact same directory structure**. You can edit and/or create files in the folders `controller`, `scripts/matlab` and `scripts/python` only! The files you can edit are indicated in ***bold*** in the list above.

Rename the `material` folder where you implemented your solution as:

- **`material_Group<GroupNo>`**

Include it in the submitted zip file, as stated at the beginning of this section. **Caution: the code must be able to build and run on the lab computers (Ubuntu 20.04 and Webots R2023a).**

Report guidelines

The final report with all explanations, figures and results must not exceed **4 pages**. Note that this is a hard limit, and longer reports will be penalized. If you want to add references to your report, they can exceed the fourth page.

Please use one of the following templates in 2-column format (note these are the templates for IEEE publications):

- http://ras.papercept.net/conferences/support/tex.php (ieeeconf.zip, for LaTex)
- http://ras.papercept.net/conferences/support/word.php (ieeeconf_A4.dot, for MS Word)

Hand in the report in **PDF format**. It must be named:

- **`SIS_23-24_Project_Report_Group<GroupNo>.pdf`**

Your final report should include the following sections:

- *Abstract*: short and concise description of the project and results, usually one or two paragraphs at most.
- *Introduction*: a brief description of the project, what has to be done, what is challenging in your opinion, brief mention of the solutions you implemented.
- *Methods*: what methods are used, and what choices are made and why (for controller design, data processing, etc.). What are the sources of noise, limitations, and constraints in your work, and how do you deal with them? What is your strategy for making your approach robust to changes in the environment and noise? You should describe your algorithms with appropriate abstractions (e.g., flowchart, pseudocode) and avoid using code in your report. Make use of appropriate visual aids (e.g., drawings) to explain your ideas in a precise and concise way. Report all relevant equations that you used or implemented.
- *Results*: You should include tables or plots here. Make sure that the tables, plots, etc., are readable, e.g., not too cluttered, and contain all the relevant information, such as the correct labels, units, etc.
- *Conclusion*: summary, overall evaluation of solution performance, bottlenecks/difficulties, future improvements (for instance, what you might not have had time to do but would be a nice addition).
- *References*: relevant past work and sources that you used, i.e., that you did not come up with yourself but used in your project.

You can add subsections to further structure your report if needed.

Final demonstration

On top of submitting your project on Moodle (report and code), you will present your solution during a **live demonstration**, where you will be able to showcase your implemented solution. During the demonstrations, the following procedure will be applied:

- Your submitted code (without any change) will be built and run by the TAs on one of the lab computers. Make sure to include all required files for us to build and run your project, including makefiles. Anticipate that we may use a Webots world where light intensity patterns interchange among lights. For example, the light labeled as 'number 2' could exhibit an intensity pattern typically associated with 'number 3'.
- Your robot should demonstrate the "Online" requirements given in the "Minimal Requirements" sections.
- You will also be asked some specific questions about your implementation, such as the explanation of your strategy, justification of your results, possible improvements and clarification about your report, etc.

The demonstrations will take place on **Thursday, December 21st**. The exact schedule will be communicated in a timely fashion.

Grading

The final grade for your course project will be based on the following criteria:

Initiative, rigorousness, teamwork	25%
Actual work performed and achievement of goals	40%
Quality of final report	25%
Quality of final demonstration	10%