

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 (optional)
- Webots **R2023a**

General information

This project will be carried out in teams of **three** students. The goal is for you to work autonomously both individually and as a team to solve the presented problem on your own. The project is divided into **three independent Work Packages (WP)**. Each WP is implemented individually by a single student (individual effort) and then the team is responsible to bring together these WPs to implement the full project (group effort). In case where the team is only constituted of two students, only two out of the three WPs must be implemented (students can choose which two WPs to implement). Each WP will be graded individually and will therefore yield a different grade for each student (individual grades). A shared grade will be attributed to the team for the integration of the WPs into a comprehensive final solution (group grade). As a result, each student of the team may receive a different grade at the end of the project, resulting from the weighted combination of the individual and group grades obtained. Despite each WP being graded individually, we recommend the students to collaborate while implementing them as each WP will have to be integrated with the remaining WPs. Early coordination will be beneficial.

Teams have been defined by considering student preferences and eventually approved centrally by the Head TA of the course. Each team will proceed with a single submission where individual contributions are clearly highlighted (more details below). 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.

You will be able to get assistance from the teaching assistants during the dedicated Thursdays lab sessions, **exclusively in the third hour** of the lab session when a regular lab is taking place. During Week 11 and Week 13, all the three hours of the lab session will be dedicated to the course project assistance.

Collaboration policy

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.

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 (ensure you work locally on the lab computer!).

You will find additional technical information useful for completing this project in the accompanying document `SIS_25-26_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 lights in good conditions to 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:

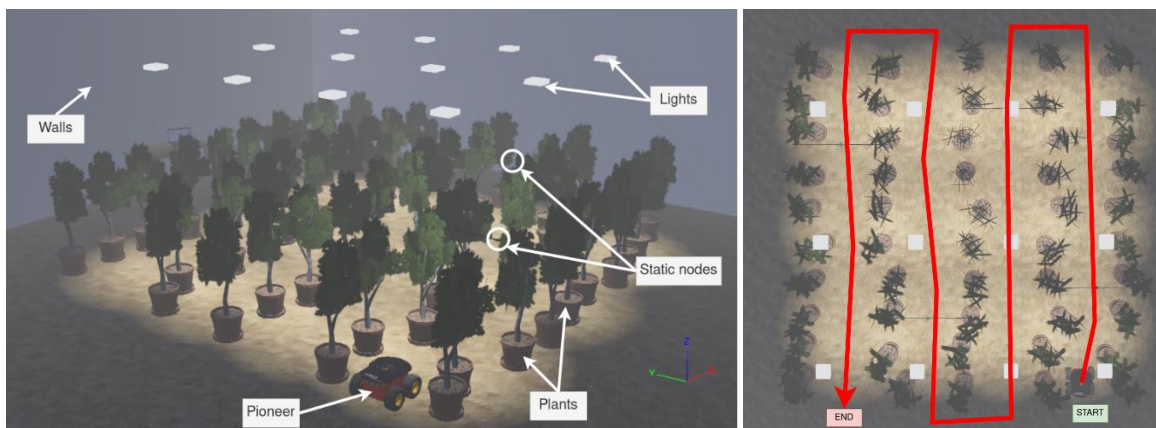
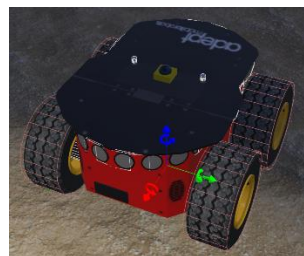


Figure 1. Greenhouse overview with robot starting location and required robot path.

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/s²]**.
- 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 and/or adapt your odometry model!

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: $>5\text{m}$ \rightarrow measurement: 0
- Distance: 0m \rightarrow 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 characteristics

The greenhouse thermal model is defined as follows:

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

where q_{heater} is the heat rate (power) produced by the heating system, $T(t)$ [°C] is the temperature in the greenhouse, T_o [°C] is the outdoor temperature (assumed constant), R [°C/W] is the thermal resistance and C [J/°C] is the thermal capacitance of the greenhouse.

The following characteristics are known:

- $C = 0.25$ [J/K]
- $q_{heater} = 0.1$ [W]

The ideal desired temperature in the greenhouse is $T^* = 21.5$ °C. You will need to collect data from the static nodes to learn about the temperatures $T(t)$ and T_o , with $T(t \leq 0) = T_o$, and assess whether the insulation R of the greenhouse is suitable to ensure the desired temperature T^* inside the greenhouse. The heater is turned on as soon as the simulation starts and can be assumed to immediately generate the given power (perfect step behavior). Make sure you collect sufficient data points for $T(t)$ and T_o from the static nodes so that you understand their evolution through time and can assess the thermal insulation of the greenhouse.

Note: the numerical values introduced here are not realistic but allow us to have a model easier to handle within the scope of the considered scenario. Do not rely solely on your intuition and validate your assumptions with the data you collect!

Greenhouse light system

The greenhouse is equipped with twelve lights disposed across its ceiling. In principle, these lights emit a rather constant light intensity when they are in good shape but the robot should make sure of it. The lights are known to display three types of behavior:

1. Nominal: healthy light, overall constant light intensity.
2. Flicker: high-rate fluctuations in light intensity.
3. Defect: turns on and off periodically.

Work Packages (WP)

Each student is attributed one of the following WPs listed below. We list here the detailed requirements in terms of implemented features and deliverable for each WP. We further distinguish between **real-time** (happens while the robot moves through the greenhouse, showcased during the demo) and **offline** (a posteriori analysis using data collected by robot, reported in the report) tasks.

The individual WPs are as follows (one WP per student), and we *highlight* elements we expect in your report:

- **WP1:** Robot Navigation and Temperature Analysis.
 - **(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. Implement a suitable strategy to achieve this objective. Only **onboard sensor data** may be used. **It is strictly forbidden to provide ground truth information to the robot!**
 - **(offline)** Add a *single plot* of the robot trajectory (using the pose ground truth, logged by the supervisor) when running your navigation solution, where the x-axis and y-axis are the x and y coordinates in meters in the greenhouse, respectively (line plot). Report collisions (if any) with the environment logged by the supervisor (scatter plot) and include the total number of collisions and total time in seconds it took the robot to finish travelling through the greenhouse in the title of the plot (note that some false positives may occur occasionally, no need to worry if this happens).
 - **(offline)** Identify the thermal resistance (isolation) R of the greenhouse offline based on the data logged by the robot when receiving messages from the static nodes. Compute the ideal value R^* , that would enable the indoor temperature to be maintained at the desired value T^* with the provided heating system. For this, give the equation of the *step response* of the indoor temperature $T(t)$ in the time domain that describes the observed evolution of the indoor temperature as a function of the defined thermal parameters.
- **WP2:** Robot Localization.
 - **(real-time)** Run an Extended Kalman Filter (EKF) to estimate the pose of the robot in the greenhouse (x, y, heading), relative to the origin where the robot starts from. Use the method `robot.set_state_estimate_marker(...)` each time you update the state estimate to visualize it in real time in the simulation (shown as a green arrow)! Think of the information available to the robot (onboard sensors, knowledge of the greenhouse's shape, data received from the static nodes, etc.). Use as much information as possible and motivate why you use certain pieces of information or not. **It is strictly forbidden to provide ground truth information to the robot!**
 - **(offline)** Provide a *single plot* showing the **estimated** robot position (x, y) (logged by the robot), where the x-axis and y-axis are the x and y coordinates in meters in the greenhouse, respectively (line plot). Plot the ground truth (logged by the supervisor) as well and label both lines clearly. In *another plot*, report the position error in x and y (solid lines), as well as the x and y uncertainty (dashed lines with matching color). Adjust the scale of the uncertainty accordingly for good visualization, and mention the scale used. In a *final plot*, report the heading error and the heading uncertainty similarly to the x and y error plot.
- **WP3:** Light Analysis.

- **(real-time)** Using the data collected by its sensors, the robot must print in the terminal each detected light when encountered, its status (whether it is good or its identified defect type with the help of the Fourier transform), and its estimated location. For instance: `Detected light n°2, status: good, location: (3.5,2.8) m.` Make sure *not to log too much information* to not clutter the terminal and make it unreadable for the final demo!
- **(offline)** 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 ground truth trajectory (line plot), the position of all detected lights (scatter plot), and their defect classification (e.g., using color codes on the scatter plot). Do not forget to add all relevant labels to the plot.

Finally, the team needs to also report on the integration of the individual WPs into a comprehensive solution:

- **WPG**: WP for the Group (WPG), details of WPs integration.
 - **(offline)** Detail which *interactions between WPs* had to be implemented (for instance, one WP may require the navigation WP1 to pause in some cases, or the navigation WP1 may want to get the robot's current position from WP2, etc.). How did you *collaborate* to ensure everything would work as a whole in the end? Mention *whether any WP is not (fully) functional*.

To help decouple the development of the various WPs, we provide a keyboard controller that allows the user to move the robots manually through the greenhouse. This is in particular useful when working on **WP2 and WP3**, until WP1 becomes functional.

The robot's ground truth pose can be obtained as well by the robot, which can be actively used when working on **WP3 only (when the localization estimation of WP2 is not integrated yet)**! Note that if used, a message is displayed in the 3D window of the simulator.

Note that in the end, neither of these features should be used for WPG, as we expect the robot to perform the entirety of its mission autonomously!

Submitting your project

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

- **SIS_25-26_Project_Group<GroupN°>.zip**, containing:
 - **material_Group<GroupN°>**
 - **SIS_25-26_Project_Report_Group<GroupN°>.pdf**

For instance, for group n° 42 (do **not** leave the “< >”!!):

- **SIS_25-26_Project_Group42.zip**, containing:
 - **material_Group42**
 - **SIS_25-26_Project_Report_Group42.pdf**

The project is due on **Mon May 25, 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 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.

Ensure that you write the name of the student(s) who worked on each file as a comment at the beginning of the file:

```
# Implemented by: <students' name>
```

Rename the `material` folder where you implemented your solution as (without the “< >”!!):

- **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 GR B0 01 computers (Ubuntu 20.04 and Webots R2023a).**

Report guidelines

The final report with all explanations, figures and results must be at most **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)

Do **not** alter the margins or any styling provided by the template!

Hand in the report in **PDF format**. It must be named (without the “< >”!!):

- **SIS_25-26_Project_Report_Group<GroupNo>.pdf**

Your final report should be structured as follows:

- Page 1: Project title (**SIS Project Report - Group <GroupNo>**), authors name (**Student1 (SCIPER), Student2 (SCIPER), Student3 (SCIPER)**), project abstract and one section entitled **WPs integration**, with additional subsections as relevant (evaluated **jointly**).
- Page 2: WP1 implementation and results (evaluated **individually**).
- Page 3: WP2 implementation and results (evaluated **individually**).
- Page 4: WP3 implementation and results (evaluated **individually**).
- Page 5: References (optional).

Add a page break at the end of each page if necessary to ensure the next WP starts on the next page. For each WP, start with a new **section**. Each WP page must be structured as follows:

- *Title*: Section’s title, same as the title defined in the Work Packages section of this document. Add the following sentence directly below the title: “*Implemented by <student name>, <SCIPER>.*” The following **subsections** must be included:
 - *Introduction*: a brief description of the WP, 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? You should answer questions here such as: - 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, diagrams) 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. Refer to the WP description to check what plot(s) are expected. Feel free to add more plots if relevant and if space allows.
 - *Conclusion*: summary, overall evaluation of solution performance, bottlenecks/difficulties, future improvements (for instance, what you might not have had time to do but would have been a nice addition).

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 computers usually deployed in GR B0 01. 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 are located differently from the world you were provided for developing your solution. The plant pots may be moved slightly as well.
- Your robot should demonstrate the “**online**” requirements in the WP descriptions and run fully autonomously. If not, you will be able to run your WP(s) manually to demonstrate the implemented functionalities if relevant.
- 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 **Thu May 28**. 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:

- Individual: 60%

Work performed and achievement of goals	40%
Quality of individual WP report	20%

- Group: 40%

Teamwork	20%
Quality of group WP report	10%
Quality of final demonstration	10%