ENGLISH

USER MANUAL

RSE Probe-Bot

Mini-Rover and PC application
for data acquisition
in hazardous environments

Final Project
Computer Science and Mathematics (200.C0)
College of Bois-de-Boulogne
Winter 2018 Semester

By :
Katherine Zamudio-Turcotte,
Arthur Van Betsbrugge
and Charles-Éric Langlois

V. 2.0.2 - 26/04/2022

Abstract

This project was conducted as part of the Integrative Project course in the Computer Science and Mathematics program (200.C0), also known as “SIM”.

We are three students who studied SIM from fall 2016 to winter 2018 at College of Bois-de-Boulogne, in Montreal, Canada. Two of us are now studying Computer Engineering at Polytechnique Montreal and are developing their passion for embedded systems and robotics. The last one of us is now studying Software Engineering and is hoping to pursue a career in the videogame industry. We are still extremely proud of this project, which we started not too long after learning the basics of programming in Java. This project allowed us to understand our interests better and to confirm our university program choice. To this day, RSE is used in youth workshops and recruiting activities in different robotics clubs. It inspires people of all ages to discover STEM and to try something new. We hope that RSE will bring as much excitement to you as it did to us while we created it.

Katherine Zamudio-Turcotte – https://www.linkedin.com/in/katherine-zamudio-turcotte/
Arthur Van Betsbrugge – https://www.linkedin.com/in/arthur-van-betsbrugge/
Charles-Éric Langlois – https://www.linkedin.com/in/charles-éric-langlois-b619a2193/

Picture of RSE now used to promote the SIM program
on the College of Bois-de-Boulogne website

I. Required pieces

In order to build an RSE robot just like ours, the following items are required :

- Robot structure

  1x Arduino DFRobotshop Rover V2 kit[1]
  1x Expansion plate for the DFRobotshop Rover V2 kit[2]

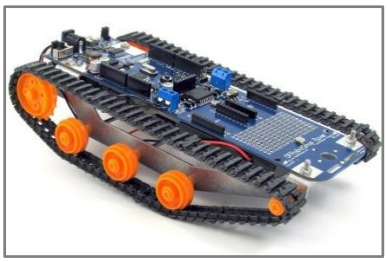
- Electronics

  1x Arduino Mega 2560
  1x Si7021 Breakout Board from Adafruit[3]
  1x MQ-2 Analog gas sensor from DFRobot[4]
  1x MQ-7 Analog carbon monoxide sensor from DFRobot[5]
  1x Analog ambient light sensor from DFRobot[6]
  1x HC-06 Bluetooth Module[7]
  1x HC-SR04 Ultrasound sensor[8]
  1x Mega sensor shield V2.4 from DFRobot[9]
  1x LED, preferably 10 mm wide
  1x 560 ohms resistor

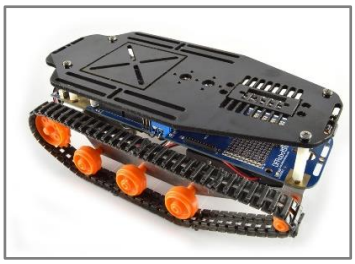
- Cables and batteries

  1x Male USB A to male USB Mini-B (5 pin) cable[10]
  1x Male USB A to male USB B cable
  1x LiPo battery 7.4V 1Ah[11] (and charger[12]) **or** 9V batteries (and 2.1mm connector[13])
  4x AA batteries
  9x Male-female junction wires of 6 inches[14]

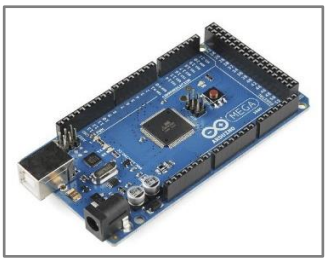
[1] https://www.robotshop.com/ca/en/dfrobotshop-rover-tracked-robot-basic-kit.html
[2] https://www.robotshop.com/ca/en/dfrobotshop-rover-expansion-plate.html
[3] https://www.adafruit.com/product/3251
[4] https://www.robotshop.com/ca/en/analog-gas-sensor-mq2.html
[5] https://www.robotshop.com/ca/en/gravity-analog-carbon-monoxide-sensor-mq7.html
[6] https://www.robotshop.com/ca/en/gravity-ambient-light-sensor.html
[7] https://www.robotshop.com/ca/en/hc-06-bluetooth-module.html
[8] https://www.robotshop.com/ca/en/hc-sr04-ultrasonic-range-finder-tys.html
[9] https://www.amazon.com/DFRobot-Sensor-Shield-Compatible-Arduino/dp/B00CK5B06Q
[10] https://www.digikey.ca/en/products/detail/qualtek/3021009-06/1531292?utm_adgroup=USB%20Cables&utm_source=google&utm_medium=cpc&utm_campaign=Shopping_Product_Cable%20Assemblies&utm_term=&productid=1531292&gclid=CjwKCAjwr7X4BRA4EiwAUXjbt65XCAZOP8NOqrd86J_lS_fPV9AvK9S8vgT-ucKDt8MeqIS70I4P8hoCpjgQAvD_BwE
[11] https://www.robotshop.com/ca/en/74v-1000mah-5c-lipo-battery.html?utm_source=google&utm_medium=surfaces&utm_campaign=surfaces_across_google_caen&gclid=CjwKCAjwr7X4BRA4EiwAUXjbtwidqZAyDK-TNWFeJLfdGyVUONkcZ7hqjjN8WXhJsFVqxWl0OudAihoC33wQAvD_BwE
[12] https://www.robotshop.com/ca/en/74v-lipo-battery-charger.html
[13] https://www.digikey.ca/en/products/detail/kitronik-ltd/4132/8635470
[14] https://addison-electronique.com/en/products/electricity/pjwmf-6-male-to-female-premium-jumper-wires-pq10/



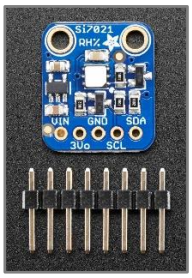
DFRobotshop Rover V2



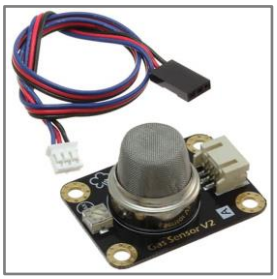
Expansion plate



Arduino Mega 2560



Si7021



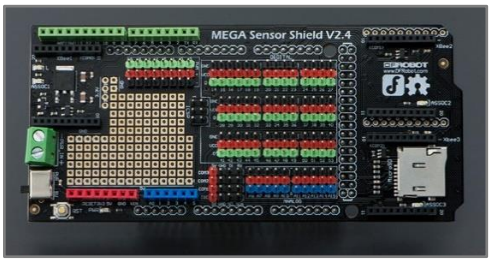
MQ-2

MQ-7

Ambient light sensor

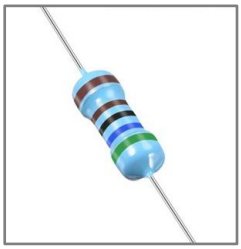
HC-06

HC-SR04



Sensor shield V2.4

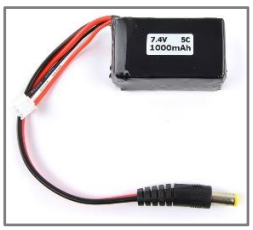
10 mm LED



560 ohms resistor

USB A - USB Mini-B (5 pin) cable

USB A - USB B cable



LiPo battery

4 AA batteries

9 junction wires

Price estimate : 365$ CAD + shipping

You may also need additional wires, scotch tape, feltac and some other electronics pieces.
The following sections of this manual assume you already have built an RSE robot.

II. Quick building procedure

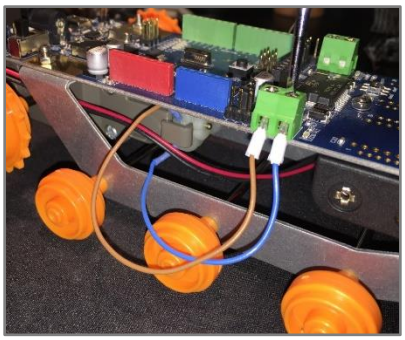
1. Stack the Mega sensor shield on top of the Arduino Mega.

2. Connect the breakout board, the gas sensors and the light sensor on the shield using their documentation (that can be found easily on the web) :
    a. analog port A6 for the MQ-2 sensor
    b. analog port A8 for the MQ-7 sensor
    c. analog port A10 for the light sensor
    d. I2C pins for the breakout board

3. Connect the Bluetooth module to the shield using COM1 :
    a. module's VCC to shield's 5V
    b. module's GND to shield's GND
    c. module's TXD to shield's RX
    d. module's RXD to shield's TX

4. Build the DFRobotshop Rover V2 kit as instructed, with motors in the C position.

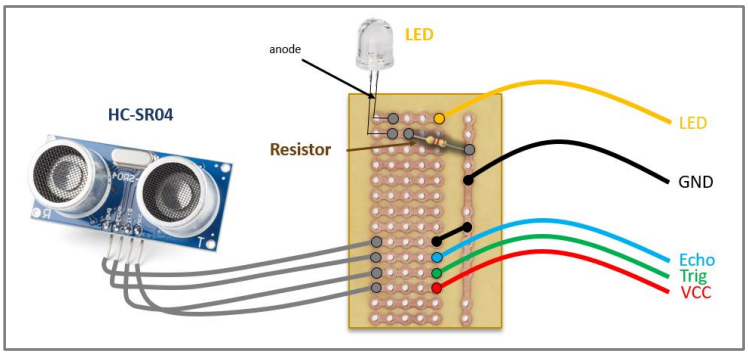
5. Use wires to connect the twin motors to the green terminals on each side of the rover's PCB :



Connecting the twim motors to the Arduino Uno

6. Put AA batteries in the battery pack and plug the power cable in the Arduino using a white terminal next to the power switch.

7. Make this circuit for the HC-SR04 distance sensor :



HC-SR04 circuit with LED

8. Connect the Echo wire to digital pin 3, the Trig wire to digital pin 4 and the LED wire to digital pin 2. GND goes on GND and VCC goes on 5V. Place the circuit so that the HC-SR04 sits at the front of the Arduino Uno PCB and keep it in place with scotch tape (you can level the circuit with feltac).

9. Use a junction wire to connect digital pin 13 to the pin controlling the blue LEDs on the rover, which is one of the pins located next to the power switch (specifically the pin closest to the edge).

10. Install the expansion plate.

11. Place the Arduino Mega, the sensors and the Bluetooth module on the plate (secure with tape and level with feltac). We recommend placing the Arduino near the back of the robot to balance weight.

12. Use a junction wire to connect the Uno's TX pin to the Mega shield's COM2 RX pin.

13. Use a second junction wire to connect the Uno's RX pin to the Mega shield's COM2 TX pin.

14. Plug the LiPo or 9V in the Arduino Mega and power up the Arduino Uno with the power switch.

III. Code uploading procedure for the Arduino controllers

15. Download Arduino IDE if you don't already have it :
    https://www.arduino.cc/en/main/software
    Note : Download the latest version to make sure the DFRobotshop Rover V2 Arduino is recognized. It should be recognized as an Arduino Uno, even if the PCB is different.

16. Clone our GitHub repo :
    https://github.com/KatherineZT/RSE-Probe-Bot

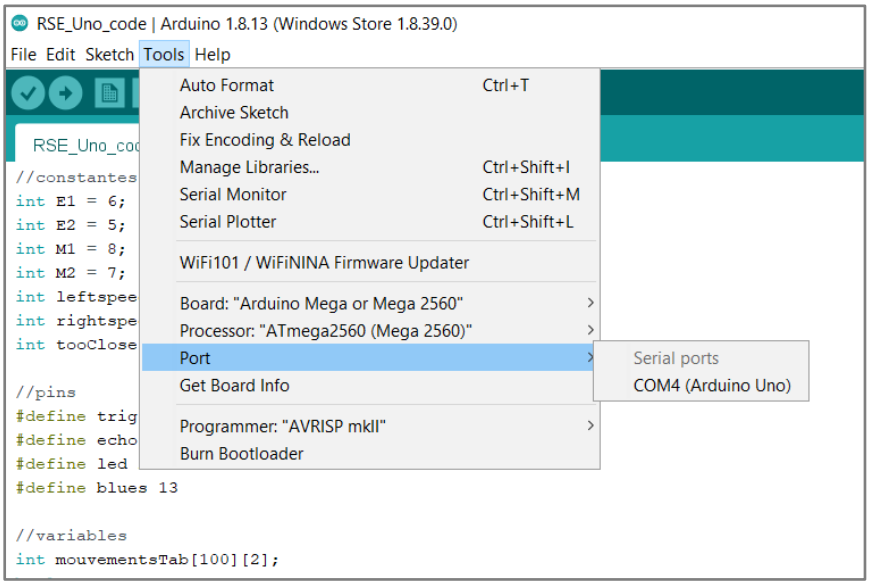
A. Arduino Uno

17. Navigate to *Arduino files* → *RSE_Uno_code*.

18. Double-click on *RSE_Uno_code.ino* to open the file in Arduino IDE.

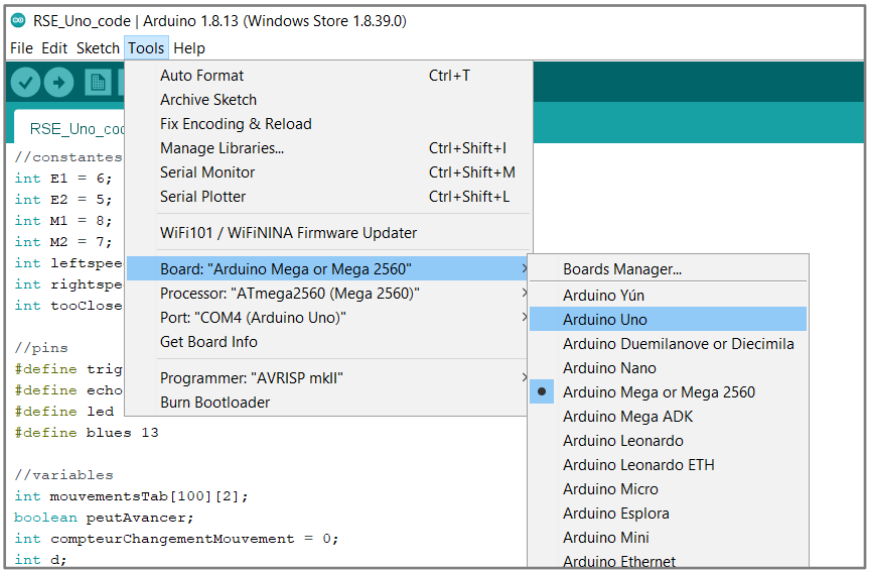
19. Make sure the Arduino is turned off with the power switch and plug it into your computer using the USB A to Mini-B (5 pin) cable.

20. In Arduino IDE, open « Tools » in the task menu. Hover over « Port » and select the COM port associated with an Arduino Uno :



COM port selection for the Arduino Uno

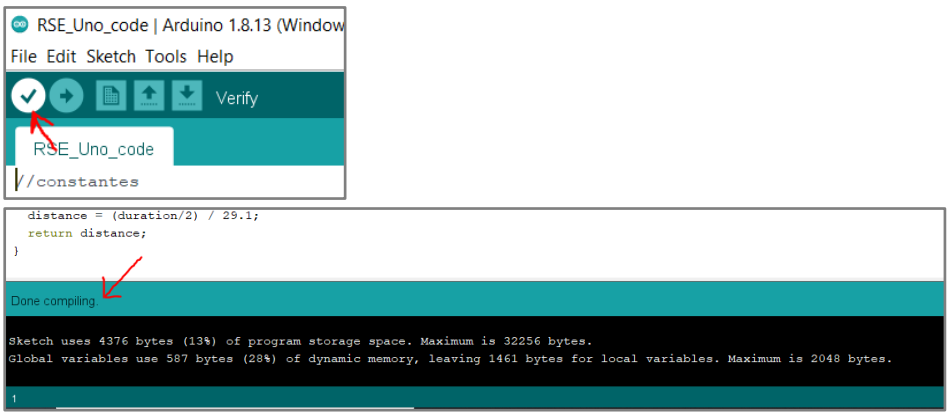
21. Open « Tools » again and select « Arduino Uno » as the board :



Board selection for the Arduino Uno

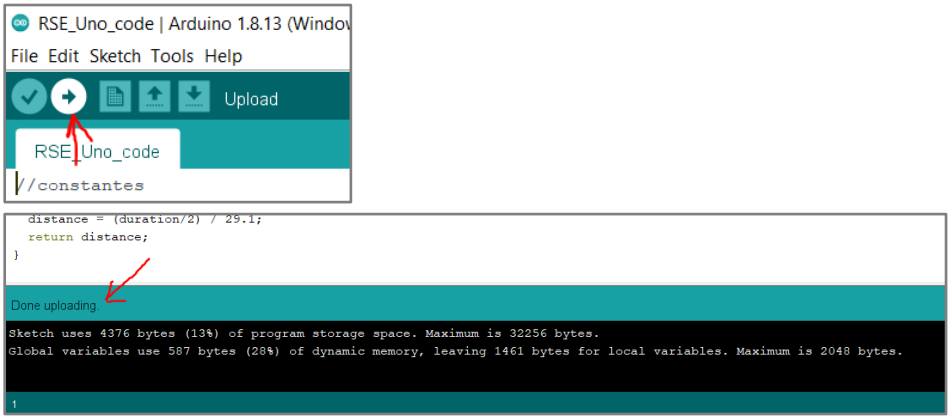
22. Find COM2 on the V2.4 Mega sensor shield. Unplug the two junction wires connected to it, since they can interfere with the upload of the code.

23. Click on « Verify » to compile and make sure no errors are detected :



Verification successful

24. Click on « Upload » to upload the code into the board :



Upload successful

25. Unplug the robot from your computer. Power it up and do not put back the COM2 cables just yet.

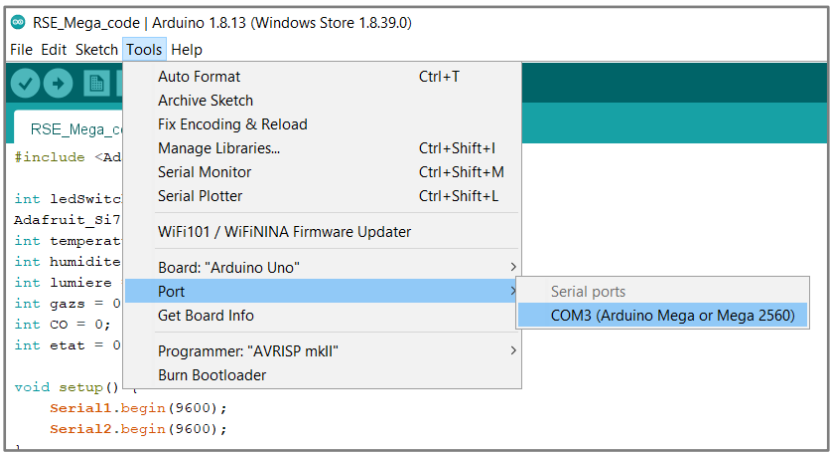
B. Arduino Mega

26. Unplug the LiPo battery or the 9V batteries.

27. Navigate to *RSE_Mega_code* directory and double-click on *RSE_Mega_code.ino* to open it.

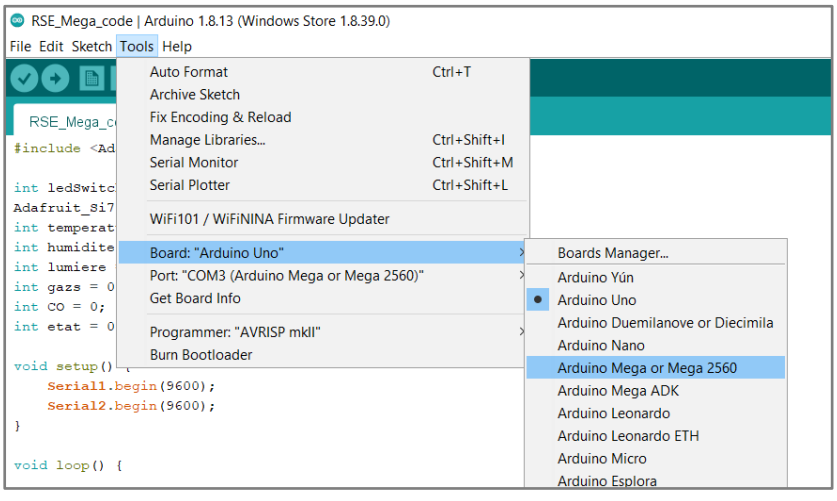
28. Use the USB A to USB B cable to connect the Arduino Mega to your computer.

29. In Arduino IDE, open « Tools » in the task menu. Hover over « Port » and select the COM port associated with an Arduino Mega :



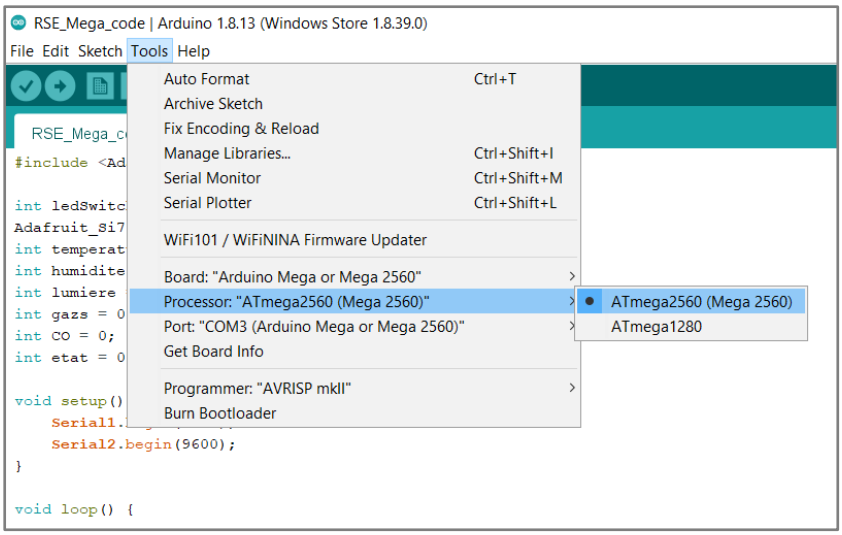
COM port selection for the Arduino Mega

30. Open « Tools » again and select « Arduino Mega or Mega 2560 » as the board :



Board selection for the Arduino Mega

31. Open « Tools » again just to verify that the processor is set to « ATmega2560 (Mega 2560) » :



Processor selection for the Arduino Mega

32. Click on « Verify » and « Upload ».

33. Put back the COM2 cables on the V2.4 Mega sensor shield the way they used to. Now the two Arduinos can communicate to each other using this port.

34. Unplug the Arduino Mega from your computer and plug the batteries again.

The RSE is now ready to be used. The only step left is to open the data visualization and mission control application.

IV. Application startup

1. The application file is the JAR file (RSEApp.jar) that's in the RSEApp directory of the GitHub repository.

2. Make sure that the red light on the HC-06 Bluetooth module is blinking.
   Important : If the light isn't always blinking or is off, change the LiPo or 9V batteries.

3. Also make sure that the Arduino Uno's power switch is ON and that the green and yellow LEDs on the board are lighting up.
   Important : Otherwise, change the AA batteries.

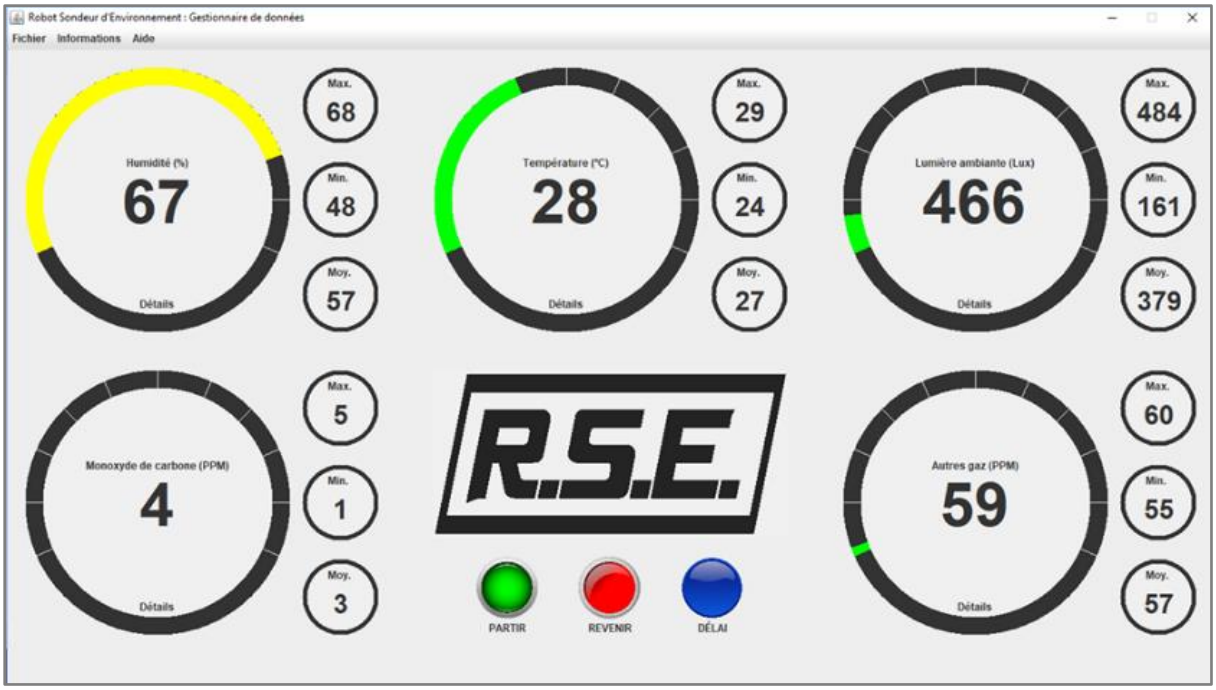
4. Launch the JAR executable by double-clicking on it.
   Note : If a security warning is shown, you can safely ignore it.
   Note : If you move the JAR file to execute it in another directory, the application will work but the images won't be found.

5. The app should now open. The Bluetooth module should stop blinking and stay on. The real-time sensor values should start refreshing every second or so.
   Note : Close the window and launch the JAR again in case there is a problem.

Important : The application tends to not work as well when the RSE is off or too far (a Bluetooth connection works well below 10 meters).

You may want to use the Eclipse project instead of the JAR file. In this case, go see Annex A.

V. Application tour



Note : The dashboard may not look like this picture. We still need to fix the positions and scaling. With the JAR, the results might be worse. These pictures represent how the application looked when we did the demo of our project in class.

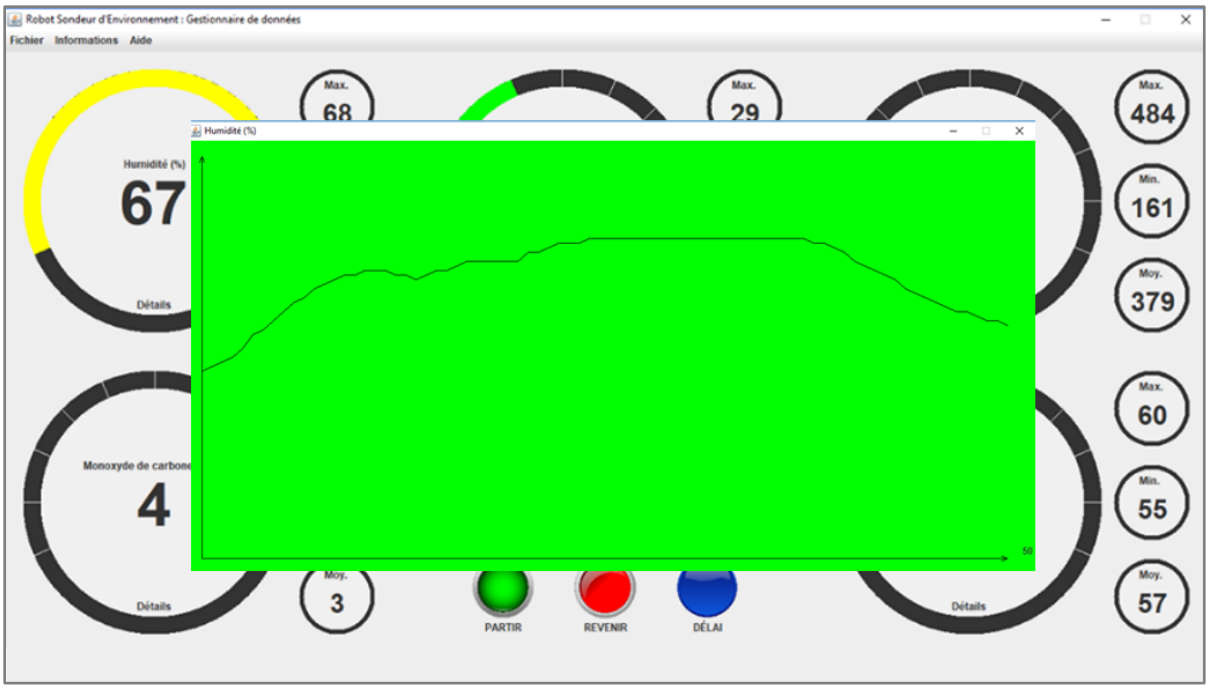
The dashboard shows 5 **dials**. Each one represents a measured physical quantity in the surrounding environment : humidity, temperature, ambient light, carbon monoxide, and combustible gas concentration (LPG, butane, propane, methane, alcohol, hydrogen, smoke).

In the center of each dial, there is a value representing the real-time quantity that is measured. The colored arcs which form the outline of each dial have a color code that represents the danger level :

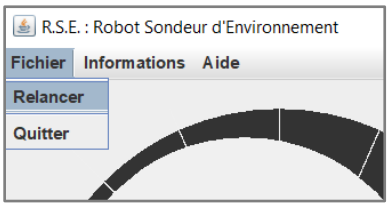
Green → none or little danger
Yellow → moderated danger (avoid staying in that area for too long)
Red → life-threatening

For example, insufficient lighting will be categorized as yellow, but a life-threatening concentration of carbon monoxide will be tagged as red.

To the right of each dial are 3 smaller dials which indicate the maximum, minimum and average values since the beginning of the mission for each physical quantity. At the bottom of each large dial, there is a clickable "**Détails**" (details) label, which opens a **chart** that shows the values' evolution over time. The chart is shown in the same color as the current value's danger level. The chart does not refresh on its own : it is like an image. It needs to be closed and re-opened to be refreshed. The chart's axes are not graduated, but the value corresponding to the mouse cursor's position on the x axis is shown at all times next to the cursor.

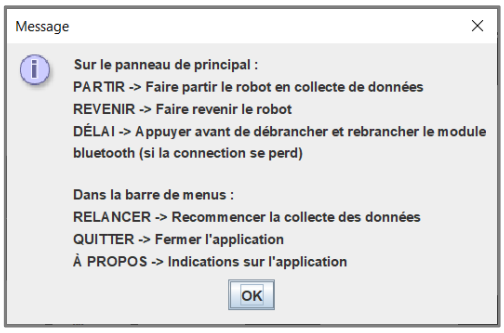


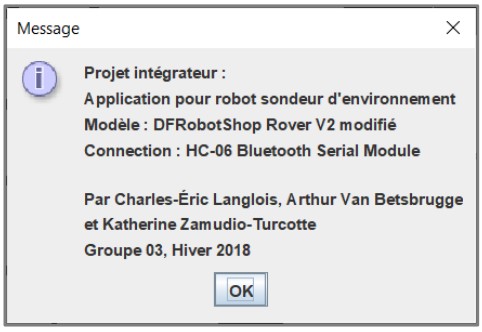
To restart the acquisition of sensor data and forget previous measurements, the user can use the « Fichier » → « **Relancer** » (restart) menu option. To close the application and end the Bluetooth data flow, the window can be closed or the « Fichier » → « **Quitter** » (quit) option can be used.



In the « Informations » menu, « **État du robot** » (robot state) tells the user if the robot is moving and if it has been called back to its starting point.

In the « Aide » (help) menu, the « **Utilisation** » option gives a reminder (in French) of the commands that have been explained just now and « **À propos** » (about) shows basic information about the robot model and the Bluetooth module model.





On the dashboard, 3 **buttons** can control the robot. They are all in a « ready » state when the application opens.

« Partir » (start) tells the robot to start a mission. The robot starts to move and to avoid obstacles on its own.

« Revenir » (return) tells the robot to turn around and come back to its starting point.

The « Délai » (delay) button enables the user to stop the Bluetooth connection momentarily without disturbing sensor data acquisition. The user can click on this button before unplugging and replugging the Bluetooth module on the robot to restore a normal Bluetooth connection, and click on « OK » once it’s done.

Annex A – Open the app with Eclipse

1. Download Eclipse IDE : https://www.eclipse.org/downloads/.

2. Start the IDE.

3. In the options menu, select « File » → « Open Projects from File System… ».

4. To the right of « Import Source : », click on « Directory… ». Choose the RSEApp directory from our GitHub repository and select « OK ».

5. Click on « Finish ».

6. In the Package Explorer, the RSEApp should've opened. Right-click on the project name, hover on « Build Path » and select « Configure Build Path… ».

7. Open the « Libraries » tab. Select the « bluecove-2.1.1.jar » library and click on the « Edit… » button. Choose the « bluecove-2.1.1.jar » file that you will find in the RSEApp directory of our GitHub repository. Click on « Open » and then on « Apply and Close ».

8. Use the green « Run » button in the action menu on top of the screen or the Ctrl+F11 shortcut to launch the application.