# Lab 10: Introduction to the DISAL Arduino Xbee Sensor Node

This laboratory requires the following software (installed in Ubuntu Linux in GR B0 01) and hardware (provided to each group for this exercise session only):

- Arduino IDE
- 2 Arduino Mega boards with Xbee shield
- 1 USB cable

The laboratory duration is approximately 3 hours. Although this laboratory is not graded, we encourage you to take your own personal notes as the final exam might leverage results acquired during this laboratory session. For any questions, please contact us at **sis-ta@groupes.epfl.ch**.

### 1.1 Information

In this assignment, you will find several exercises and questions.

- The notation $\mathbf{Q_x}$ means that the question can be answered theoretically or with simple commands in the Linux operating system, without implementing or running any code.
- The notation $\mathbf{S_x}$ means that the question can be solved only by compiling and running a piece of code or an additional simulation.
- The notation $\mathbf{I_x}$ means that the problem has to be solved by implementing, possibly compiling, and running a piece of code.
- The notation $\mathbf{B_x}$ means that the question is optional (bonus) and should be answered if you have enough time at your disposal.

# Set up: The Arduino Xbee log and communication kit

The Arduino Xbee log and communication kit was developed at DISAL in 2019. It is composed by an Arduino Mega 2560 board, a custom shield board designed at DISAL and an Xbee module for communication. The custom shield board allows the Arduino Mega to be interfaced with several sensors and an easy-to-use communication module.

## 1 Arduino Mega board

Arduino is an open source electronic platform that provides easy-to-use hardware and software. Arduino boards can be used for a wide variety of development projects, from using sensors, to controlling a motor, to creating your own greenhouse.

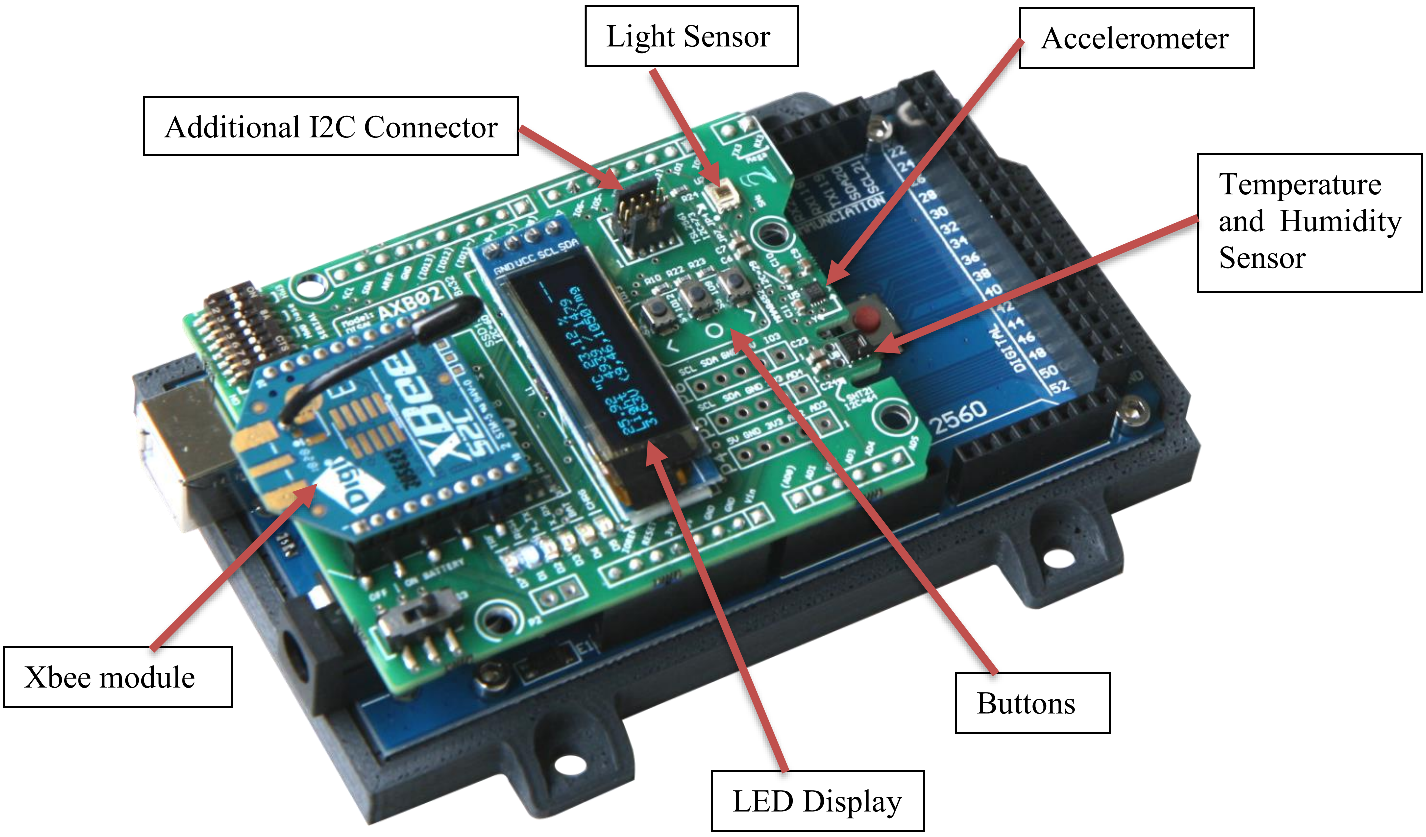
The Arduino Mega board was designed by the Arduino company. With its ATMega2560 microcontroller, it is a more powerful board than the company’s most renowned Arduino UNO. The Mega board is also bigger with 54 digital I/O pins and 16 analog pins, making it ideal for more complex development projects.

## 2 Arduino Xbee shield

The Arduino Xbee shield was designed at DISAL in 2019. This shield board can be plugged into an Arduino Mega or UNO and hosts several sensors and a communication module. Having a pluggable shield board, rather than wiring the sensors to the Arduino Mega by hand, makes it easier and more robust to use sensors. The shield board also provides additional pins to easily connect sensors that are not already present on the board. Moreover, the shield is powered with a battery, which is recharged when the board is connected to a computer.

The Arduino Xbee shield hosts the following modules:

- Light sensor (TSL2561-T, ams)
- Humidity and Temperature sensor (SHT20, Sensirion)
- Digital Accelerometer (MMA8652, NXP)
- Xbee connector
- LED display
- Buttons
- Additional connector that supports the I2C protocol




## 3 Arduino IDE and code structure

The Arduino Integrated Development Environment (IDE) is an open source software that allows for easy code development for Arduino boards. The Arduino IDE's key functionalities allow to write code, compile it, upload it and communicate to your board.

More information and resources about the IDE can be found on the Arduino website (www.arduino.cc/en/Guide/Environment).

Each Arduino program is constituted by two functions: setup() and loop(). The setup() function is called at each power-up or reset of the Arduino board and runs only once. It is used to initialize variables, configure pins, start using a library etc. The loop() function contains the main body of the program. As the name suggests, it keeps looping consecutively.

# Lab: Getting familiar with sensor nodes

## 1 Testing the hardware/software setup

Start by verifying that all the tools we provide you with are working properly

1. Extract the provided tools:

   ```
   tar -xzf lab10.tar.gz
   cd Lab10
   ```

2. Start the Arduino IDE software on your computer (it is already installed in the computer room)
3. Connect one of the two Arduino Mega board to the computer using the provided USB cable
4. In the Arduino IDE, select the type of board you want to upload your code to by going to *Tools → Board (Outils → Type de carte)* and selecting the correct board (Arduino/Genuino Mega 2560 for this lab)
5. Select the port that the board is connected to by going to *Tools → Port* (e.g. dev/ttyUSB0). The name of the port to which the board is connected is important, as it identifies the serial line that will be used to communicate from the computer to the board. If you have multiple boards connected to the computer, you will see multiple port names, each one corresponding to one port.
6. In the Arduino IDE, open the file *helloworld.ino* which is in the folder *part1/helloworld.* If you click on the file containing the code and start the IDE this way, you will not be able to upload the code to your board due to some permission restrictions.
7. Look at the code. On the top you will see some libraries that need to be added to your Arduino environment. To do this go to *Sketch/Include Library/ Manage Libraries…(Croquis/Inclure une bibliothèque/Gérer les bibliothèques)* and use the search bar to look for the libraries you need to add. Add each of them by clicking on the *Install* button. You will need to add the libraries only once to your Arduino environment, since they are saved in your local configuration.
8. Verify your code by clicking on 

9. If there are any mistakes detected by the verification step, address them and verify again until no error is shown.
10. Upload the code to your board by clicking on 

**$S_1$**: Now look at the board, what is happening?

**$S_2$**: Try pressing the buttons (look at the figure on page 2 to see where the buttons are) , what happens? Can you find the lines in the code that allow this behavior?

**$Q_3$**: Look at the code and in particular focus on the content of the functions setup() and loop(). Describe in detail what happens in each of them.

## 2 Local Sensing with one node

In this section of the lab, you will use the sensors embedded in the shield to monitor the environment.

From the Arduino IDE, open the file *ambientlight.ino* which is in the folder *part2/ambientlight.*

**$Q_4$**: Read the code and try to understand what is happening. What do you think will happen when you upload the code on the board?

**$S_5$**: Upload the code on the board (follow again the steps described in Part 1) and verify your assumptions.

Focus on the command "*Serial.print(format("Tsl2561 (full / ir) = %5u /%5u\n", full, ir));*" This command is used to print data to the serial line that is connected to the computer via USB cable. To visualize what is printed open the Arduino serial monitor (*Tools/Serial Monitor*). Make sure that the baud rate is set to 9600 (you can change it in the bottom right corner), which is the same as the one set on the board and used for communication. The baud rate represents the amount of data that is transmitted in a second. You can see that the baud rate is set in the first lines of the *setup()* function in the *ambientlight.ino* code. If more than one Arduino board is connected to your computer, remember to select the right serial port (as seen in Part 1) to upload the code to the correct board.

**$S_6$**: What gets printed on the serial monitor? How many samples do you get in 20 s? What is the frequency of data transmission?

Close the serial monitor and now let's get acquainted with another very useful tool of the Arduino IDE: the Serial Plotter. With the same code running on the board, open the serial plotter (*Tools/Serial Plotter, Outils/Traceur serie*).

**$S_7$**: What do you see? What are these values? Why do you think that there are variations in the signal even when the light conditions do not change?

**$S_8$**: Now orient the light sensor on the board towards a more intense light source and then cover it with your finger, repeat these actions a few times. How does the plot change? Write down some examples of values when the light is intense, low and medium range.

**S9**: In the code, understand how the frequency of the data transmission in determined. Change the frequency to receive 1 sample every 5 seconds, upload the code to the board and validate your implementation by looking at the serial monitor. Now take a look at the serial plotter. How does the plot compare to the previous questions? What differences do you see in the plot when the light conditions change?

Set the frequency back to 1 Hz. We now want to monitor the ambient light and send a warning to the user of this sensor when the light is too low. Based on the previous question, pick a threshold for which you consider the light to be too low (for example, you should have a warning when you place your finger on the sensor).

**I10**: Go to the loop() function and add a few lines of code to first check if the light value is lower than the threshold you selected and then send a warning message on the serial line when this happens. Upload your code to the board, open the serial monitor and check that, when your finger is placed on the sensor, the warning is triggered correctly.

From the Arduino IDE, open the file *environmentalmonitoring.ino* which is located in the folder *part2/environmentalmonitoring*.

**S11**: Upload the code to the board and describe what happens. Which sensors are used?

**I12**: Go to the loop() function. Add a few lines of code to allow the user to receive data on the serial monitor from a sensor that they can select using buttons. You should print the value that the user is requesting by writing to the serial line and you should change the value of the variable btn to display on the screen which sensor data is sent back to the computer.
Change the value of the variable btn according to this schema:

- btn="T" when the board is sending temperature and humidity values
- btn="G" when the board is sending the accelerometer values
- btn="L" when the board is sending light values

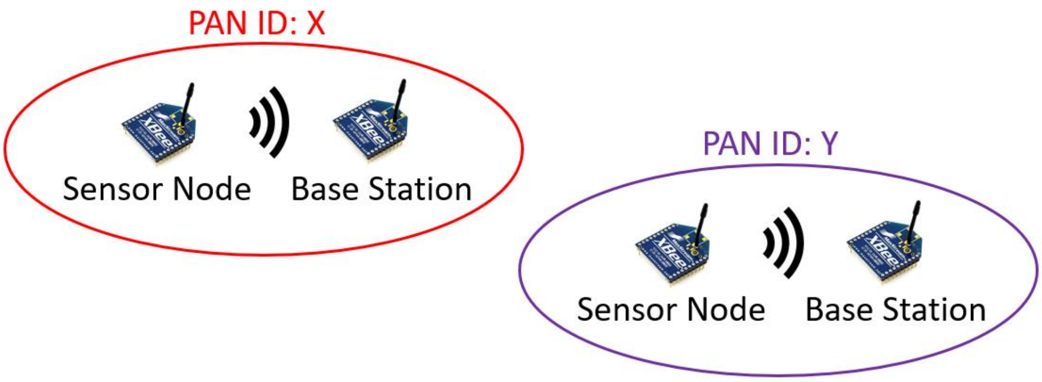
Verify your implementation by opening the serial monitor and checking that, when you press a button, the corresponding sensor value is sent on the serial line and the letter T,G or L is displayed on the screen to show which sensor value is sent.

# 3 Remote Sensing with two nodes

You will now simulate a remote sensing station using two Arduino boards. In this scenario, one board, the remote sensor node, is powered with a battery and can be displaced freely in the room, while the other board is connected to the computer and acts as a base station to receive data from the remote node.

The shield mounted on the Arduino boards uses Xbee modules for communication. These modules provide a low-cost and easy-to-use solution for indoor short-range communication. They also consume little power, which makes them ideal for devices powered with a battery. They are compliant with the IEEE 802.15.4 and Zigbee

standards for short range, low power networks. These standards allow to build reliable networks of different topology very easily.
In this laboratory, you will program the remote node to send data to the base station. To accomplish this and make sure that you are sending data to your base station and not to someone else's, you have to configure a parameter called PAN ID. Your Xbees will broadcast data to all Xbees with the same PAN ID, creating a network.

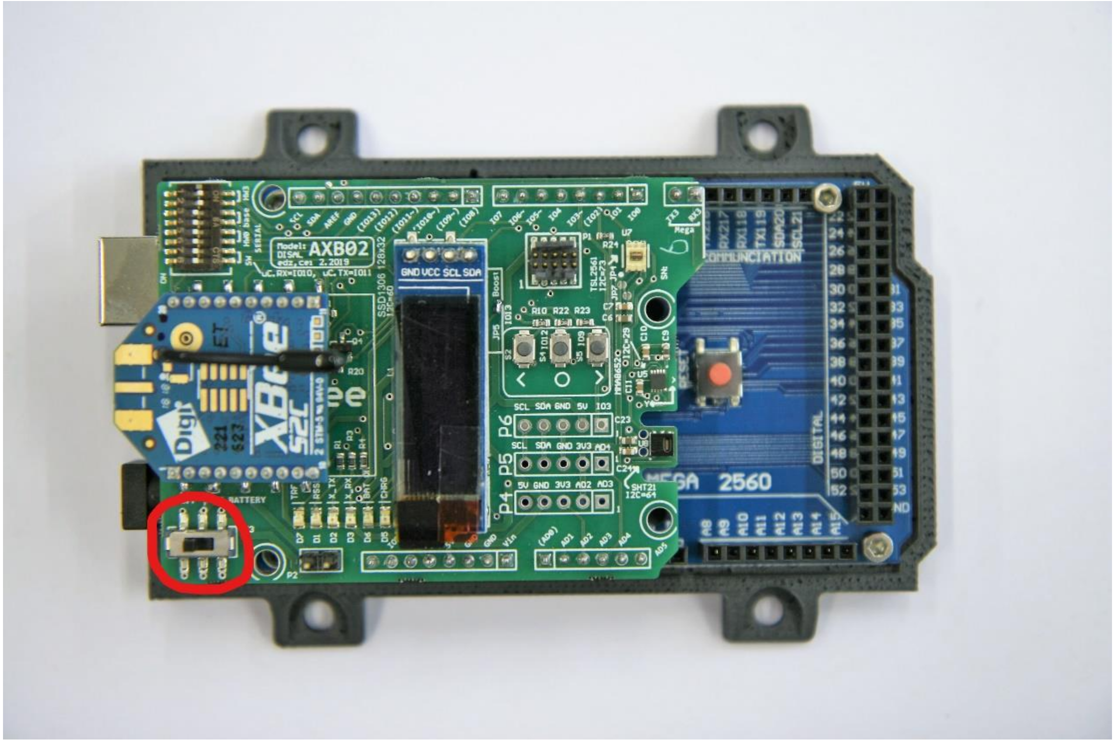



In the Part 3 folder, open the file receiver.ino. Take a look at the code and notice that another serial line is initialized in the setup (Serial3). This serial line is connected to the Xbee and is used to transmit and receive data through wireless communication. At line 33 **change the PAN ID by substituting the number 2019 with the number of your computer** in the computer room (eg., Serial3.print("ATID 3\r"); ). Upload the code to the board that you are going to use as base station.

**S13**: Look at the loop() function. Can you explain what is happening?

Your base station is now ready to be used. Disconnect it from the computer and connect the board that you will use as a remote node.

Open the file transmitter.ino in the part3 folder. At line 48 **change the PAN ID by substituting the number 2019 with the number of your computer** in the computer room (eg., Serial3.print("ATID 3\r"); ). This number has to be the same as the one in the receiver.ino file. Upload the code to the board and disconnect it from the computer. Turn on the battery switch (circled in red in the figure below), the board should turn on.




**S14**: Keep the transmitter.ino code open and connect the base station to the computer. Open the serial monitor (if the serial monitor does not open, try to select the correct Port of the receiver board from the Tools menu). It might take up to 30 seconds for the communication to start. What do you observe? Which sensor is used for data transmission? What is the frequency of transmission?

**S15**: Open the serial plotter and observe the data. Let the board send data for about 30 seconds, then place your finger on the temperature sensor on transmitter board and wait for about 30 seconds. Remove your fingers and wait for 60 seconds. How does the data plot vary? Is the sensor responsive? Is it noisy? How does it compare to the response time of the light sensor used in Part 2?

**S16**: In the transmitter.ino file, change the value of the SAMPLE_FREQ variable to allow faster sampling, upload the code to the remote node and repeat the experiment described in S2 (pay attention to upload the code to the remote node and not to the base station board!!). Do you observe any differences in the plot? What would happen if you decrease the sampling frequency?

## 4 Energy-aware communication

In this part of the lab, you will put together everything that you have learnt so far to create a remote sensing station that is energy-aware. This means that your remote node will transmit information only in some occasions to efficiently use the power of the battery.

You are going to recreate this situation: your remote node is monitoring a greenhouse with several plants that need stable temperature to thrive. In normal operation mode, your system sends temperature data with a frequency of 1 Hz. However, when the temperature in greater than 28 degrees Celsius, you want to send data at 10 Hz to improve the sensing and have more information about this critical condition.

Open the file *energyawaretransmitter.ino* in the Part 4 folder.

**I17**: Implement the scenario above by detecting when the temperature reaches a certain stage and changing the value of the frequency of data transmission accordingly. **Do not forget to change the PAN ID** to the number that you have used in previous questions. Upload your code on the remote node, then connect the base station to the computer and open the serial plotter. Test your implementation by placing a finger on the temperature sensor to increase its temperature.

## IMPORTANT: What to do before handing the board back to the TAs

Before handing the boards back to a TA, go to the folder *maintenance* and upload the code you find inside on both boards. This code reads the battery voltage and prints it on the OLED screen. This way, we can quickly turn the boards on and see if they need to be recharged. Thank you for your help!