The Night of the Living Sensors: How to Tame the MKR WAN 1310 Without Fear

Version: 1.0, 2024/10/16
Author: Marco Rainone

Introduction

This document outlines the criteria used in the development of an IoT system designed to monitor water quality.

Using an **Arduino** microcontroller and a series of sensors, the system measures key parameters such as temperature, conductivity, pH, dissolved oxygen, dissolved solids, and turbidity. The collected data is sent wirelessly via **LoRaWAN** and transmitted to a server using MQTT to the **ThingsBoard** platform, enabling real-time monitoring and analysis.

Designing a data acquisition system involves combining hardware and software components, integrating different technologies that need to work together efficiently and seamlessly.

This approach allows for the creation of a reliable, easy-to-use system capable of continuously monitoring important parameters without the need for manual intervention.

The choices made in this project are based on general criteria that can also be applied to other IoT projects, making it versatile for various applications, such as environmental, agricultural, or industrial monitoring.

A new project: what to watch out for

When using a microcontroller like Arduino for a new project, there are several things to keep in mind to ensure everything works well.

Here are the key points to focus on:

1. Project goals

- **Define your goal**: Before starting, you need a clear idea of what you want your project to do. Which sensors or devices do you want to control? What functions should the system have?
- **Think ahead**: If your project might grow in complexity, consider whether Arduino will be enough, or if you might need a more powerful board.

2. Choosing the right board

- **Hardware compatibility**: There are different Arduino models (Uno, Mega, MKR, ESP32, STM32, etc.), each with different features. Make sure to choose one that has enough memory and pins for all the devices you need to connect.
- **Connectivity**: If your project needs Wi-Fi, Bluetooth, or LoRaWAN, choose a board that already has these features built-in, or that supports compatible external modules.

3. Power management

- **How to power Arduino**: You need to decide if you'll power the project with a battery, an adapter, or via USB. It's important that the power supply can provide enough current to run all your peripherals.
- **Energy-saving**: If you're using a battery and the project needs to run for a long time, try to optimize energy consumption. For example, you can put Arduino into sleep mode when it's not needed.

4. Managing inputs/outputs (I/O)

- **Number of pins**: Make sure the Arduino board has enough pins to connect all your sensors and devices. If not, you can use tools to expand the available pins.

- **Pay attention to voltage**: Some Arduino boards run on 5V, while others run on 3.3V. Sensors can also require different voltages. If you use components with different voltages, you might need a converter to avoid damaging them.
- **Protecting the pins**: If you’re connecting external devices, it might be useful to protect the pins with resistors or other components to avoid short circuits.

5. Memory management

- **Limited memory**: Arduino has limited memory for both code and data. Make sure your program doesn’t use too much memory, or it might crash.
- **Saving data**: If you need to save data even after Arduino turns off, you can use non-volatile memory like a microSD card or an EEPROM (but keep in mind EEPROMs have a more limited number of write cycles).

6. Connecting sensors and actuators

- **Compatibility**: Check that the sensors and actuators you want to use are compatible with the Arduino board in terms of voltage and communication protocols (I2C, SPI, UART, etc.).
- **Pre-made libraries**: Many sensors have ready-made libraries for Arduino that simplify their use. Make sure the library is compatible with the Arduino model you're using.

7. Testing and troubleshooting

- **Serial monitor**: Use Arduino’s Serial Monitor to see real-time data and troubleshoot any issue.
- **Step-by-step testing**: Don’t try to do everything at once. Test one component at a time, then combine everything after ensuring that every part works.
- **Error handling**: In your code, add checks to handle errors, such as a sensor’s failing, so the system becomes more reliable.

Analysis of project components and application of principles

The Arduino MKR WAN 1310

The **Arduino MKR WAN 1310** is a microcontroller board designed for developing **Internet of Things (IoT)** systems, capable of collecting and transmitting data over long distances using **LoRaWAN** technology. This technology is particularly suited for applications where minimizing energy consumption and ensuring long-range coverage are crucial.

Whether for environmental, agricultural, or industrial monitoring, the MKR WAN 1310 offers a simple yet powerful solution for gathering data in remote areas where Wi-Fi or cellular connections are unavailable or impractical.

The **input/output (I/O) pins** and **analog ports** on the MKR WAN 1310 operate at **3.3V**, meaning sensors and devices connected to the board must be compatible with this voltage level to avoid damaging the board.

The MKR WAN 1310 is ideal for those looking to create monitoring and data collection systems across various scientific fields, from ecology to agronomy and industrial engineering.

It is designed to connect sensors and devices to a LoRaWAN network, enabling data transmission from isolated locations such as remote weather stations, water quality monitoring systems, or distributed agricultural applications.

Table of Main Features:

Feature	Description
Microcontroller	SAMD21 Cortex-M0+ 32-bit, manages system processing and operation
Connectivity	LoRaWAN, enabling long-range communication with low power consumption
Memory	256KB flash memory, 32KB RAM, for handling code and data
Power Supply	Powered via USB, Li-Po battery, or Vin pin
Power Consumption	Optimized for low-power operation, ideal for long-duration battery-powered projects
I/O Interfaces	Digital and analog pins at **3.3V** for connecting sensors and other devices; logic level at 3.3V
Communication Ports	**UART (Serial), I2C, SPI**, for communication with a variety of sensors and peripherals
Size	Compact design, easy to integrate into field installations
Battery Charging	Includes a charging circuit for Li-Po batteries, making it suitable for mobile or remote projects
Programming	Programmable via the Arduino IDE, making it accessible even to users with little programming experience

The other components of the project

Below is the list of peripherals that need to be connected to the Arduino MKR1310 microcontroller in this project:

- **KIT0021** - Digital DS18B20 kit Arduino (temperature sensor)
- **SEN0169-V2** - Analog pH Meter Pro Kit V2 (pH sensor)
- **SEN0189** - Analog Turbidity Sensor (turbidity sensor)
- **SEN0237-A** - Analog Dissolved Oxygen Sensor (dissolved oxygen sensor)
- **SEN0244** - Analog TDS Sensor (Total Dissolved Solids sensor)
- **SEN0451** - Conductivity Sensor (K=1) (electrical conductivity sensor)
- **DFR0641** - Precision DS3231M MEMS Real Time Clock
- **TEL0132** - GPS + BDS BeiDou Dual Module

The following table summarizes key information about the peripherals, details in the links provided:

Device code	Use	Voltage range
KIT0021 - Digital DS18B20	Waterproof temperature sensor	3.0V - 5.5V
SEN0169-V2 - Analog pH Meter	pH measurement with analog output	3.3V - 5.5V
SEN0189 - Analog Turbidity Sensor	Measures water turbidity	5V
SEN0237-A - Dissolved Oxygen Sensor	Measures dissolved oxygen in water	3.3V - 5.5V
SEN0244 - Analog TDS Sensor	Measures Total Dissolved Solids (TDS)	3.3V - 5V
SEN0451 - Conductivity Sensor (K=1)	Measures water electrical conductivity	3.3V - 5V
DFR0641 - DS3231M MEMS RTC	Low-power, battery-backed real-time clock	3.3V
TEL0132 - GPS + BDS Module	GPS and BeiDou satellite positioning	3.3V

Characteristics of the sensors used in the project

As designers, we did not participate directly in the selection of the sensors used for the project. However, after reviewing the documentation, we found that the selected sensors vary in quality and features. These can be categorized into three main types: **professional sensors**, **laboratory sensors**, and **non-professional sensors**.

- **Professional sensors** are more reliable and durable, suitable for challenging environments like outdoor installations, such as near a river.
- **Laboratory and non-professional sensors** are better suited for controlled environments or educational projects, but offer less precision and durability.

This variety in sensor quality can impact the precision of measurements and long-term system reliability. It is essential to consider these differences during development to ensure that the system meets the project's goals.

Sensor table

Sensor	Category	Reason
SEN0237-A - Analog Dissolved Oxygen Sensor	Professional sensor	Ideal for measuring dissolved oxygen, suitable for field monitoring and serious applications.
SEN0451 - Gravity Conductivity Sensor (K=1)	Professional sensor	Used for professional water quality analysis, designed to withstand real-world conditions.
SEN0169-V2 - Analog pH Meter Pro Kit V2	Professional sensor	High-precision pH sensor, suitable for continuous use in professional applications.
KIT0021 - Waterproof DS18B20 Temperature Sensor	Laboratory sensor	Waterproof sensor for controlled environments or educational projects.
SEN0244 - Gravity Analog TDS Sensor	Laboratory sensor	Sensor for measuring dissolved solids, suitable for laboratory experiments, not for harsh conditions.
SEN0189 - Analog	Non-professional	Low-end turbidity sensor, ideal for prototypes

Sensor	Category	Reason
Turbidity Sensor for Arduino	sensor	and educational projects, not suitable for professional use.

How to solve this?

1. **Flexible software configuration**: The software can be configured to activate or deactivate sensors based on the operating environment (field or laboratory).
2. **Modularity**: The system can be designed in a modular way, allowing easy sensor replacement based on the context.
3. **Calibration**: Implement different calibrations for sensors, depending on the type and the environment in which the system is used.

In this way, the same system can be used in different contexts, optimizing both performance and costs, without compromising the quality of the collected data.

Voltage level compatibility between the microcontroller and sensors

The Arduino MKR1310 operates at **3.3V**, and most of the sensors are compatible with its voltage. However, the SEN0189 **- Analog Turbidity Sensor** requires **5V** to operate, so an external 5V power source or a logic level converter will be needed to handle its analog signal properly.

Connecting the SEN0189 to the MKR1310

The **SEN0189** sensor requires a **5V** power supply, while the MKR1310 operates at **3.3V**.

To interface them correctly, two things need to be addressed:

1. power supply and
2. signal handling.

1. Powering the sensor:

- The **SEN0189** needs **5V** to work. Since the MKR1310 doesn’t provide 5V directly, an external **5V** power source (such as a USB port or a separate power supply) must be used.

2. Handling the analog signal:

- The **SEN0189** outputs an analog signal ranging from **0V to 5V**, but the MKR1310 can only read signals up to **3.3V**. To prevent damaging the microcontroller, a **voltage divider** must be used to reduce the signal.

Voltage divider:

A voltage divider is a simple circuit made with two resistors in series. The output signal from the sensor passes through these resistors, reducing the voltage to a safe level for the MKR1310.

$$V_{out} = V_{in} \times \frac{R2}{R1 + R2} \quad (1)$$

For example, to reduce the signal from **5V** to **3.3V**, the following resistor values can be used:

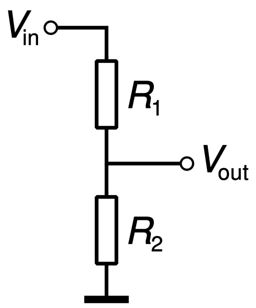
- **R1**: 62kΩ
- **R2**: 120kΩ

With Vin=5V Vout results 3.296V

To calculate resistor values using standard commercial values, a very useful tool is this one: https://damien.douxchamps.net/elec/resdiv/

Wiring diagram:

- Connect the output pin of the **SEN0189** sensor to **Vin** of resistor **R1**.
- Connect the other end of **R1 (Vout)** to the **analog pin** of the Arduino MKR1310.
- Connect **R2** between the junction of **R1** and **A0**, and **GND** of the Arduino.



3. Final wiring setup:

- **VCC (5V)**: Connect the sensor to an external **5V** power source.
- **GND**: Connect the sensor's **GND** to the **GND** of the MKR1310.
- **OUT (analog signal)**: Connect the sensor's output signal to the voltage divider, then to the **A0** pin of the Arduino.

4. Reading the signal in Arduino:

To read the sensor's signal with the Arduino MKR1310, use the following code:

```
#define ANALOG_PIN_TURBIDITY  A0
...
int turbidityValue = analogRead(ANALOG_PIN_TURBIDITY);
```

After obtaining the analog value, remember to calibrate the sensor and convert the value to a real turbidity measurement according to the sensor's documentation.

Power Supply Structure in the Project

The system is powered by a **12V battery** recharged by a **solar panel**. The battery is managed by a **solar charge controller**, which has two USB ports providing power to the various electronic components.

For this project, two voltage levels are required:

5V to power the **Arduino MKR1310** microcontroller and **3.3V** for most of the sensors.

Below is a step-by-step explanation of how to properly set up the power supply.

Power Structure

- **Solar panel**: Converts sunlight into electrical power, which is sent to the **solar charge controller**.
- **Solar charge controller**: Charges the 12V battery and provides regulated **5V** to the USB outputs to power the system.

Reducing the voltage for the sensors

The microcontroller provides a 3.3V output from its onboard regulator, but it is important to check if this regulator can supply enough current for the sensors. If not, another **voltage regulator** is needed to reduce the voltage from **5V** to **3.3V**.

Current Consumption Estimate During Operation

Below is an updated table that shows the sensors and additional components being used, their operating voltage, and the current they consume based on the manufacturer’s specifications.

This includes the **SD Proto Shield** and a **32GB microSD**.

Component	Operating Voltage (V)	Current Consumption (mA)
DFR0641 - DS3231M MEMS RTC	3.3V	1-2 mA (active mode)
KIT0021 - Digital DS18B20	3.3V - 5V	1.5 mA (active mode)
SEN0169-V2 - Analog pH Meter	3.3V - 5V	5-10 mA
SEN0189 - Analog Turbidity Sensor	5V	30 mA
SEN0237-A - Dissolved Oxygen Sensor	3.3V - 5V	40 mA (up to 50 mA during readings)
SEN0244 - Analog TDS Sensor	3.3V - 5V	3-6 mA
SEN0451 - Conductivity Sensor (K=1)	3.3V - 5V	3-5 mA
TEL0132 - GPS + BDS Module	3.3V	30 mA (up to 40 mA during GPS

Component	Operating Voltage (V)	Current Consumption (mA)
		search)
MKR SD Proto Shield	3.3V	3-5 mA
32GB microSD	3.3V	100 mA (approximate)

Estimated Total Current Required

- **DFR0641 - DS3231M MEMS RTC**: 2 mA
- **KIT0021 - Waterproof DS18B20**: 1.5 mA
- **SEN0169-V2 - Analog pH Meter**: 10 mA (maximum estimated)
- **SEN0189 - Analog Turbidity Sensor**: 30 mA
- **SEN0237-A - Dissolved Oxygen Sensor**: 50 mA (during readings)
- **SEN0244 - Analog TDS Sensor**: 6 mA
- **SEN0451 - Conductivity Sensor (K=1)**: 5 mA
- **TEL0132 - GPS + BDS Module**: 40 mA (during GPS search)
- **MKR SD Proto Shield**: 5 mA
- **32GB microSD**: 100 mA

Total Calculation:

Summing up the maximum current consumption of all components:

Total Current=2+1.5+10+30+50+6+5+40+5+100=249.5 mA

So, the estimated total current to power all components, including sensors, the SD Proto Shield, and the microSD, is about **250 mA**.

Additional Considerations:

- This estimate still doesn’t include the **Arduino MKR1310** microcontroller, which consumes about **32 mA** in active mode, with significant increments during wireless communication.
- It’s crucial to factor in the additional load from the **microSD card**, which draws around **100 mA** during operation.

Final Estimate:

When accounting for the **Arduino MKR1310**, the total current requirement for the system would be around **282 mA** (250 mA from components + 32 mA from the MKR1310).

A well-designed power supply and battery should account for this total to ensure smooth operation, even during peak demand.

Note on microsd consumption

The power consumption of microSD cards can vary based on their capacity, speed class, and usage mode (read/write/idle). Typically, a microSD card will consume:

- **Idle mode**: 0.5 mA to 2 mA

- **Read operations**: 20 mA to 40 mA
- **Write operations**: 30 mA to 100 mA (or more, depending on the card's speed class and capacity)

For a **32GB microSD card**, the typical power consumption might be around:

- **Idle**: 0.5 mA - 2 mA
- **Reading**: 20 mA - 30 mA
- **Writing**: 30 mA - 80 mA

The 100 mA I used earlier would be on the higher side, possibly during intensive write operations on a fast-speed class card (e.g., UHS-I or higher).

For most average use cases, the card would draw closer to **30-50 mA** during normal read/write operations.

Current Limits in the MKRWAN 1310 specifications

In the documentation for the Arduino MKRWAN 1310 (available here), there are important details about current limits:

1. **Maximum source current: 46 mA**
    - This means the MKRWAN 1310 microcontroller can supply a maximum of **46 milliamps (mA)** at **3.3V** in total to power connected devices. Since our connected peripherals consume more they have to be externally.
2. **Maximum sink current per pin group: 65 mA**
    - "Sink" refers to the Arduino receiving current from external devices. Each pin group on the Arduino can absorb up to **65 mA** in total. This limit is important to prevent overloading the Arduino.
3. **Maximum current per pin: 7 mA**
    - Each individual pin can handle a maximum of **7 mA**, either in or out. So, if you connect a device or sensor to a pin, it shouldn't draw or send more than **7 mA**, or you risk damaging the pin or the connected device.

Consequences for the sensors:

These limits mean that you **cannot power sensors directly through the I/O pins** of the Arduino MKRWAN 1310 (like the digital or analog pins), because the maximum current per pin is only **7 mA**, and the total output current limit of **46 mA** is not enough to power all the sensors at the same time.

Solution: External power supply for the sensors

To power the sensors properly, you need to use an external system that can provide the required amount of current.

What to do:

1. **Use an external voltage regulator**
    - You'll need a voltage regulator to convert the **5V** (from the solar controller or another source) to **3.3V** for the sensors. This regulator should be able to supply **at least 400 mA**, to give a safety margin, as the estimated total consumption is around **280 mA**.
2. **Separate the power supply for the sensors and the microcontroller**
    - Connect the sensors that need **3.3V** directly to the output of the external regulator.
    - Use the **5V** from the solar controller to power the MKRWAN 1310 through the **Vin pin** and the **SEN0189** (Analog Turbidity Sensor) which requires **5V**.

In summary, the MKRWAN 1310 internal regulator cannot provide enough current to power all the sensors in the project.

An external voltage regulator is needed to supply **3.3V** and enough current to the sensors, while the microcontroller can be powered separately using the **5V** from the solar charger.

General power supply setup

1. **Battery:** Charged by the solar charger, acts as the energy reservoir.
2. **Solar charger controller (USB):** The **5V** output from the solar controller via the USB ports can be directly connected to the **Vin pin** of the **MKR1310**, which is designed to accept **5V**.
3. **5V to 3.3V voltage regulator**: This component converts **5V** into **3.3V** to power the sensors. There are two main options:
    1. **LDO (Low Dropout Regulator)**: If the system requires less than 500mA, an LDO regulator like the **AMS1117-3.3** is a simple solution.
    2. **Step-down converter (buck converter)**: If more current is required or higher efficiency is needed, a **step-down converter** like the **MP1584** is better because it generates less heat.

Battery monitoring

A potential future upgrade for the project could be adding a voltage sensor to monitor the battery charge level and send the data to the microcontroller.

This allows for efficient energy management based on the available power.

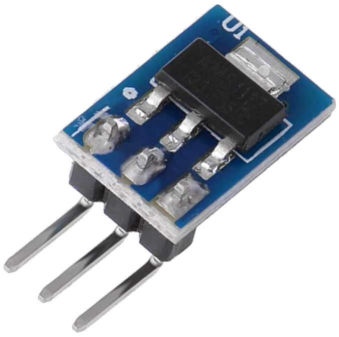
Final considerations

- **Energy efficiency**: For long-term use, a **step-down converter** is ideal to minimize energy loss.
- **Power saving**: The microcontroller can be set to **low-power mode**, and sensor readings can be scheduled at intervals to conserve battery power.

Purchased voltage regulator

The voltage regulator model we have purchased is the **AMS1117-3.3 DC 4.75V-12V to 3.3V Voltage Regulator Step Down Power Supply Buck Module**, with a maximum output current of **800mA**.

https://www.amazon.com/Anmbest-AMS1117-3-3-4-75V-12V-Voltage-Regulator/dp/B07CP4P5XJ/



Features:

- Output current: 1A
- Line adjustment rate: 0.2% (maximum)
- Load regulation: 0.4% (max.)
- Package Type: SOT-223
- Operating junction temperature range: -40 to 125°C
- Welding temperature (25 seconds): 265°C
- Storage temperature: - 65~150°C
- Output voltage: 3.267 to 3.333V (IOUT: 0 to 1A, VIN :4.75V to 12V)
- Line adjustment (maximum): 10mV (VIN :4.75V to 12V)
- Load regulation (maximum): 15mV (VIN = 5V, IOUT: 0 to 1A)
- Voltage difference (maximum): 1.3V
- Current limit: 900 ~ 1500mA
- Quiescent Current (Max): 10mA
- Ripple suppression (minimum): 60dB
- Size:8.6mm x 12.33mm

Verifications and Adjustments of Software Libraries

After working on the electronics part, it was necessary to check whether the software libraries provided by the manufacturer of the sensors could be used as-is or if modifications were needed.

It was found that all the libraries provided by DFRobot for analog sensors were designed for **Arduino Uno**. To make them work correctly with the **MKR1310**, there are some key differences to consider, such as operating voltage, analog pins, and the behavior of input/output functions.

Below is a table that summarizes the main differences between **Arduino Uno** and **MKR1310** regarding power and analog signal management:

Feature	Arduino Uno	Arduino MKR1310
Operating Voltage	5V	3.3V
Pin Logic Voltage	5V	3.3V
Maximum Pin Voltage	5V	3.3V (higher voltages can damage the microcontroller)
Number of Analog Pins	6 (A0 - A5)	7 (A0 - A6)
ADC Resolution (Analog-to-Digital Conversion)	10-bit	12-bit (more precision for analog signals)
Maximum Current per I/O Pin	20 mA	7 mA

Feature	Arduino Uno	Arduino MKR1310
Total Maximum Current for All Pins	200 mA	46 mA
Vin (Power Input Pin)	6V - 20V	5V
Vcc Output Pin	5V	3.3V

What to Do:

1. **Power Supply Compatibility**
    - Modify any references in the libraries to the reference voltage (usually 5V for Arduino Uno) to **3.3V** for the MKR1310.
2. **Modifying Analog Readings**
    - Check the use of **analogRead()** in the libraries. Arduino Uno uses a **10-bit** resolution (values from 0 to 1023) for analog pins, while the MKR1310 uses a **12-bit** resolution (values from 0 to 4095). You can either maintain the 10-bit resolution for compatibility or take advantage of the 12-bit resolution of the MKR1310.
    - To set the resolution to **10-bit** on the MKR1310, use:

    ```
    analogReadResolution(10);
    ```
3. **Modifying Reference Voltages (AREF)**
    - On Arduino Uno, the analog input reference is **5V**, while on the MKR1310, it's **3.3V**. Check the libraries for calculations based on **5V** and adjust them to **3.3V**.
4. **Adjusting ADC Readings**
    - Since the MKR1310 has a **12-bit** ADC resolution, you can decide to use the higher resolution for better precision. If you prefer to keep the **10-bit** compatibility, use the **analogReadResolution(10)** function as described above.
5. **Timing and Clock Functions**
    - If the libraries use timing functions or hardware timers specific to the **Arduino Uno** architecture (like **millis()** or **micros()**), ensure they are compatible with the ARM **Cortex-M0** architecture of the MKR1310.
6. **Serial Communication and Other Peripherals**
    - If the libraries use serial communication, verify the port usage. On the MKR1310, the standard hardware serial port is **Serial** (for USB), while other ports (e.g., **Serial1**) can be used for UART communication. **SoftwareSerial** is not supported on MKR1310, so you may need to adjust the code.
7. **External Libraries or Dependencies**
    - Check if the libraries rely on external libraries designed for the **Arduino Uno** architecture. Some of these may not be compatible with the **SAMD** architecture of the MKR1310. In that case, look for alternatives compatible with the **SAMD21**.
8. **Testing and Calibration**

- After making the adjustments, run practical tests to ensure the sensors work correctly. If the sensor requires calibration (such as pH, TDS, or dissolved oxygen sensors), recalibrate as the change in reference voltage could affect the readings.

Key Modifications:

- Adjust the reference voltage (AREF) from **5V** to **3.3V**.
- Update analog readings (possibly setting the resolution to **10-bit** for compatibility).
- Check for any **Arduino Uno specific** timing or register references and modify them for the **SAMD** architecture of the MKR1310.
- Use hardware serial ports, as **SoftwareSerial** is not supported.

System Electronics: Choosing Between Printed Circuit Board (PCB) or Solderable Breadboard?

In the project, we needed to create three prototypes of the system.

At this point, we had to decide whether to use a **printed circuit board (PCB)** or **solderable breadboards** to assemble the electronic components.

This choice depends on several factors. Here are the pros and cons of both solutions to help determine the best option:

Solderable Breadboards

Pros:

- **Easy to modify**: You can easily change the circuit, add components, or correct errors.
- **Cost-effective**: They are inexpensive and readily available.
- **Quick to use**: You don’t have to wait for a PCB to be created, so you can start building the prototype right away.
- **Great for quick tests**: Perfect for rapidly testing the circuit.

Cons:

- **Messy connections**: The wires used to connect components can make the circuit look cluttered and unclear, especially in more complex projects.
- **Less reliable**: Wires and solder joints can loosen over time, making the circuit less stable.
- **Difficult to replicate**: Making exact copies of the circuit can be complicated and time-consuming.

Printed Circuit Board (PCB)

Pros:

- **More reliable**: Components soldered directly onto a PCB are more stable and durable.
- **Clean and professional layout**: The circuit is well-organized, ideal if you need to create multiple copies of the same design.

- **Easy to assemble**: Once designed, assembling the components is straightforward and free of wiring errors.
- **Ideal if you need to create multiple identical prototypes**

Cons:

- **Higher costs and longer lead times**: Designing and producing a PCB takes more time and money compared to solderable breadboards.
- **Less flexibility**: Once the PCB is printed, it's hard to modify the circuit. If there are errors, a new PCB needs to be made.

When to Use Solderable Breadboards:

- If you want to build a prototype quickly and possibly make changes.
- If the project is simple or not too complex.
- If you want to keep initial costs low.
- If you expect to make frequent adjustments during development.

When to Use a PCB:

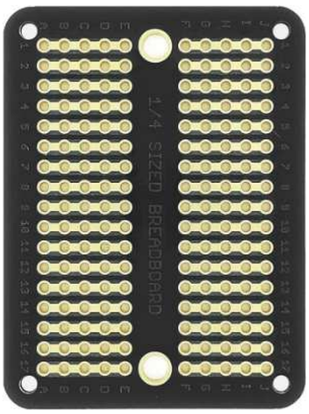
- If the project is more complex and you want a clean, precise layout.
- If you need to make multiple identical copies of the circuit.
- If you want a stable, durable, and professional circuit.
- If the design is already finalized, and you don't expect to make any changes.

Since we needed to build the prototypes quickly and the electronic design seemed not too complex, we decided to go with solderable breadboards.

There are different types of layouts for solderable breadboards.

The most common are:

- **Single holes**: Each hole is isolated, so you need to connect components with wires or jumpers.
- **Holes connected in rows or columns**: Some breadboards have holes already connected in rows or columns, similar to regular breadboards. This makes it easier to connect common components, like power lines or GND.

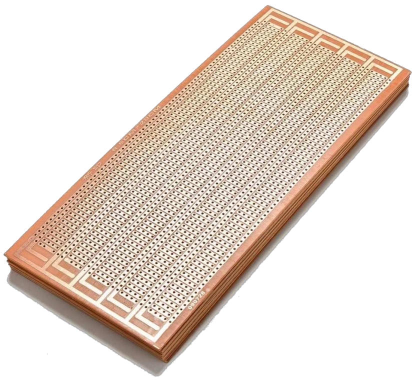


It's important to ensure that the solderable breadboard matches the pin spacing (pitch) of the components you're using.

For example, modules like the MKR1310 have a 2.54 mm pin spacing, so a breadboard with the same spacing will allow for easy mounting of the components.

Solderable breadboards used in the project

https://www.amazon.it/Hongtian-Universal-Matrix-circuiti-prototipo/dp/B06XXPT7F1



Hongtian Universal Matrix - Circuit Boards, DIY, Prototype PCB, 10 pieces
Brand: **HONGTIAN**
Hole spacing: **standard 2.54 mm**
Approximate dimensions: **length 8.5 x width 20 cm**

Criteria for Choosing the Enclosure

For this project, the water quality measurement system will be installed near a river. It is crucial to choose an enclosure that adequately protects the electronics and sensors from harsh environmental conditions, such as humidity, dust, temperature variations, and direct contact with water.

Additionally, we have decided to place the LoRaWAN and GPS antennas inside the enclosure. This decision offers several advantages:

1. **Physical protection**: The antenna will be shielded from rain, dust, tampering, and vandalism, increasing its lifespan and reducing the risk of damage.
2. **Simplified installation**: Fewer external cables and easier assembly, which is especially useful for mobile or temporary projects.
3. **Integration in controlled environments**: Ideal for industrial or laboratory settings, where order and safety are important.
4. **Reduced wear and tear**: The antenna is less exposed to mechanical stress or vibrations, prolonging its lifespan.

Key aspects when choosing the enclosure:

1. IP Protection Rating

The enclosure must have a high IP (Ingress Protection) rating against water and dust. For an installation near a river, aim for an enclosure with at least an IP65 rating or higher:

- **IP65**: Protected from dust and water jets from all directions.
- **IP66**: Protected from strong water jets.
- **IP67**: Protected from temporary immersion in water (up to 1 meter for 30 minutes).
- **IP68**: Protected from continuous immersion in water (suitable for prolonged submersion at a specified depth).

2. Enclosure material

Since the LoRaWAN antenna will be placed inside the enclosure, the material must be transparent to radio waves. Here is a classification of materials:

Material	Radio Wave Transparency	Notes	Additional Properties
Polycarbonate	High	Non-metallic plastic material, transparent to radio waves.	Lightweight, corrosion-resistant, UV-resistant, often used for industrial enclosures.
ABS	High	Widely used plastic material, transparent to radio waves.	Lightweight, durable, non-conductive, ideal for outdoor installations, UV protection, and corrosion resistance.
Reinforced Polypropylene	Moderate	Transparency depends on the type of reinforcement. If it contains metals, it may interfere with the signal.	Resistant to chemicals and impacts, ideal for corrosive environments like near bodies of water.
Stainless Steel	Low	Metal material that significantly blocks or attenuates radio waves.	Robust and corrosion-resistant, ideal for extreme environments, but more expensive and heavier than plastic materials.
Anodized Aluminum	Low	Metal material that blocks radio waves, not ideal for internal antennas.	Lightweight and corrosion-resistant, suitable for outdoor use, but not ideal for humid environments without proper treatment.

Conclusion:

- **Polycarbonate** and **ABS** are ideal materials for housing the antennas, as they are transparent to radio waves and resistant to environmental elements.

3. Seal quality

Ensure the enclosure has silicone or EPDM rubber seals around the lid and cable entries to guarantee a waterproof seal. Seals prevent water ingress even in the case of splashes or brief submersion.

4. Waterproof cable glands

Since cables (e.g., for solar panels or sensor connections) need to pass outside the enclosure, use waterproof cable glands with an IP67 rating or higher. These glands maintain the enclosure's integrity and prevent moisture from entering through cable openings.

5. Mounting and accessories

The enclosure should be mountable on poles or vertical surfaces. Look for enclosures with bracket mounting options or pre-drilled holes for easy installation on existing structures.

6. Impact and tamper resistance

Consider using an enclosure with locks or anti-tamper systems to prevent unauthorized access. Also, choose an impact-resistant enclosure to protect internal components from accidental damage.

7. Anchoring and securing the sensors

For sensors that need to be submerged in water (such as the turbidity, dissolved oxygen, and pH sensors), plan a system to safely submerge them while keeping them stable and easily accessible for maintenance and calibration. Consider using protective tubes or submersible cases to shield the sensors while submerged and ensure they remain properly positioned in the water flow.

Ideal enclosure choice:

- **Material**: ABS or polycarbonate plastic.
- **Protection rating**: IP67 or IP68.
- **Seals**: Silicone or EPDM rubber.
- **Cable glands**: Waterproof, IP67 or IP68.
- **Mounting**: Bracket or pole mounting options.

Practical option:

An ABS IP67 waterproof enclosure with silicone seals and sealed cable glands is a common choice for similar projects.

Characteristics of the enclosure used in the project

Protection rating: IP 65.
Case material: Acrylonitrile-Butadiene-Styrene (ABS).
Lid: Hinged on the long side with 2-point closure, gray or transparent color.
Color: Light gray - RAL 7035.
Operating temperature: -25 °C to +60 °C.
Impact resistance: IK8.
Package contents: Electrical enclosure, door with foam gasket, mounting plate (galvanized steel sheet), 4 wall mounting tabs, and lock.
Internal dimensions: 352 x 245 x 159 mm.
Weight: 3 kg.

The need to extend the cables for the sensor probes

The sensor probes used in the project are equipped with cables that need to be connected to their respective signal conditioning boards. The length of these cables may vary, and it might be necessary to extend them to reach the measurement points.

Extending the sensor cables can lead to several technical issues, especially in projects that involve analog and digital sensors for measuring environmental parameters. Here are the main problems that may arise and some possible solutions:

1. Signal degradation

- **Problem**: In analog sensors, extending the cable can degrade the signal, resulting in inaccurate or noisy readings.
- **Solution**: Use shielded cables to reduce interference and maintain signal integrity. Additionally, consider using signal amplifiers or analog-to-digital converters (ADC) closer to the sensor to minimize the distance the analog signal needs to travel.

2. Voltage drop

- **Problem**: Longer cables increase resistance, leading to a voltage drop that can affect the performance of powered sensors.
- **Solution**: Use cables with appropriate gauge to reduce resistance, or supply the sensors with a slightly higher voltage to compensate for the drop.

3. Electromagnetic interference (EMI)

- **Problem**: Longer cables can act like antennas, picking up electromagnetic noise, especially in industrial environments.
- **Solution**: Use shielded or twisted cables to reduce interference, and position the cables away from sources of electromagnetic noise.

4. Sensor compatibility with long cables

- **Problem**: Some sensors may not be designed to function with long cables, and their reliability could be compromised, especially in analog sensors.
- **Solution**: Check the manufacturer's specifications for any cable length limits. In some cases, signal amplifiers or converters may be needed.

5. Latency and response time

- **Problem**: Extending the cables can affect the response time of some sensors, especially digital sensors that use low-speed communication protocols like 1-wire (e.g., the DS18B20).
- **Solution**: Ensure that the sensor’s communication protocol supports long distances. For the DS18B20 sensor, it’s recommended to limit the cable length or use signal repeaters.

6. Corrosion and environmental conditions

- **Problem**: In corrosive or humid environments, extending the cables may reduce their lifespan due to corrosion or moisture ingress.
- **Solution**: Use waterproof cables or those protected by external coatings that resist corrosion, especially for sensors used near water.

7. Sensor-specific issues

- **SEN0237-A (Dissolved Oxygen), SEN0451 (Conductivity), SEN0169-V2 (pH)**: These sensors require high precision, and extending the cables can compromise the accuracy of the readings. In these cases, it is crucial to maintain signal integrity with high-quality, shielded cables.
- **KIT0021 (DS18B20)**: This is a digital sensor, but the 1-wire protocol can be sensitive to cable length. It is recommended not to exceed 30 meters without using repeaters.
- **SEN0189 (Turbidity) and SEN0244 (TDS)**: Although less sensitive than others, signal integrity and the quality of cables used can affect measurement stability.

In summary:

- **Analog sensors** require extra care to avoid signal degradation; shielded cables and signal amplifiers can help mitigate these issues.
- **Digital sensors** like the DS18B20 may require repeaters or limited cable lengths.
- Shielded, appropriately gauged, and corrosion-resistant cables are essential to ensure system durability and measurement accuracy.

Finally, it is crucial to test the cable extensions during the development phase to identify and address potential issues.