# Improvement of memory footprint

Below are two images showing the size of the object file of TheThingsNetwork library. The first image, Figure 1, illustrates the original file size without any modifications. The adjacent image, Figure 2, depicts the size of the same library object file after the applied alterations, which is named TheThingsNetwork\_IOT.cpp.o to make distinction clearer.

As depicted in the images, the size of TheThingsNetwork library's object file was approximately 146kB. However, following the implementation of adjustments and removals of unnecessary, unused functions and code segments, the size has been significantly reduced to 92.1kB. How the size has been reduced, is explained in the chapter below.

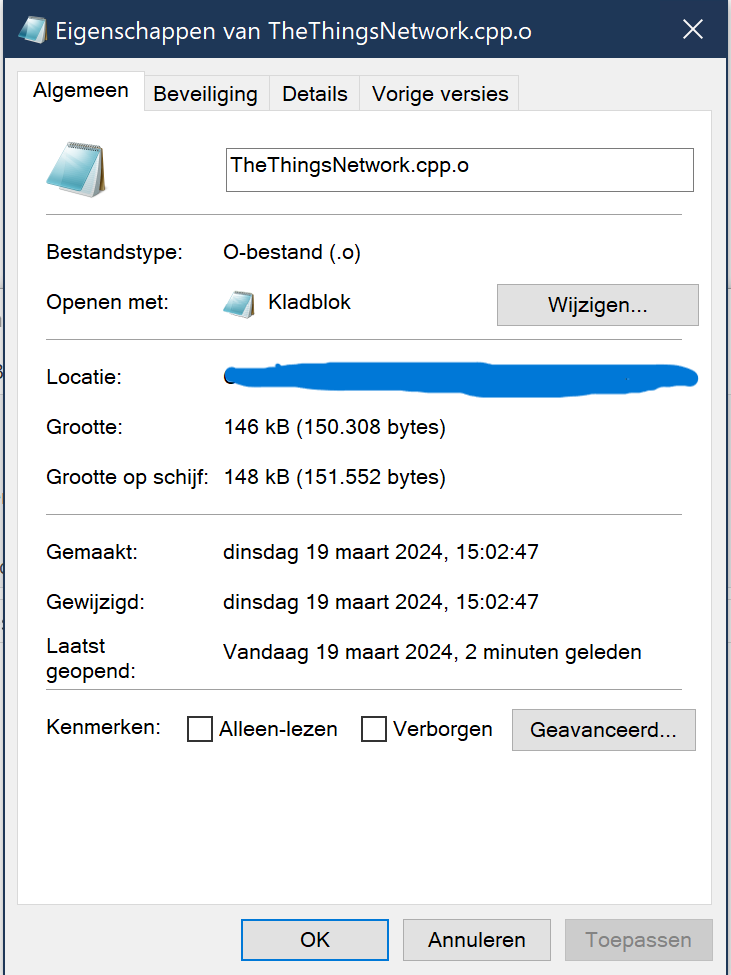
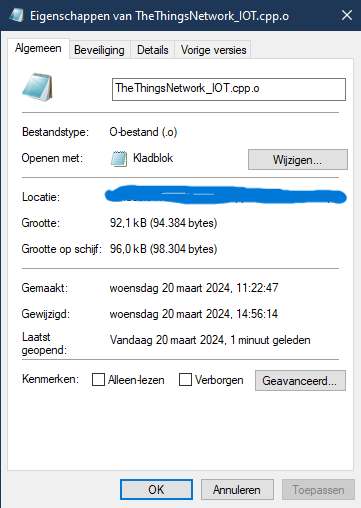


Figure 2: Adjusted footprint

Figure 1: Original footprint

Therefore, the size of TheThingsNetwork library's object file has been reduced by approximately 36.9%, while still preserving its full functionality. This substantial reduction underscores the effectiveness of the optimization endeavors, leading to a more streamlined and efficient codebase. Now, with more space available, developers have the freedom to implement additional features or integrate their own projects without concerns of exceeding flash space limits.

# Argumentation of deletions

## Regions

For this assignment, we assumed that the use of the RN2483 is confined to the current region only. Consequently, we were able to streamline various components within the codebase. This included the removal of an enum, configuration functions, and eliminating parameters from the TheThingsNetwork constructor. By restricting the scope to the current region, unnecessary code segments related to handling different regions could be safely removed.

## Debug

Further reduction of the footprint was achieved by disabling debug prints from TheThingsNetwork. During network joining, a considerable amount of strings are typically sent to the serial terminal, contributing to a larger footprint.

To exclude these functions, a #if defined directive was added at the top of the .cpp file. By default, when using this streamlined library, the variable is set to disable these debug prints. However, if necessary, users can easily enable them by adjusting the variable to ‘1’. It's strongly recommended to do this only in case of troubleshooting code issues. The library is essentially read-only, but it's easy for developers - who know what they're doing - to quickly make modifications.

Bovenkant formulier

## Radio

Drawing from the RN2483 LoRa Technology Module Command Reference User’s Guide, several functions related to radio frequencies have been removed from the library. In Chapter 2 of this guide, titled "Command Reference," various commands are listed that must be implemented. Those commands that are not specifically mentioned have been omitted from our library file. This selective approach ensures that only the necessary functions are included, aligning closely with the capabilities and requirements of the RN2483 LoRa module.

## Device classes

Class A devices are the most basic type of LoRaWAN devices. They operate with a simple "listen-before-talk" protocol, meaning they can only transmit data when prompted by the network, such as after receiving a downlink message from the network server. After transmitting, they open two receive windows for a short duration to listen for potential acknowledgments or additional downlink messages.

Classes B and C offer additional functionalities and features compared to Class A. Class B devices have scheduled receive windows, allowing them to listen for downlink messages at specific times, even if they haven't recently transmitted any data. Class C devices, on the other hand, have continuous receive windows, enabling them to listen for downlink messages almost all the time, except when they're transmitting data.

However, despite the additional features provided by Classes B and C, they are not mandatory for all LoRaWAN devices. including the device targeted by the library.Class A is the minimum requirement for LoRaWAN compliance. Therefore, in the library, all the functions regarding to Class B or C devices have been removed.

## Remaining adjustments

# References

Microchip. (2015). *The RN2483 LoRa Technology Module Command Reference User’s Guide.* Retrieved from <https://ww1.microchip.com/downloads/en/DeviceDoc/40001784B.pdf>