2 Tendril Repository 21
2.1 What's the Tendril Repository? 21
2.2 What's a wireless mesh network? 21
2.3 What does the network allow me to do? 21
2.3.1 Features 21
2.3.2 Scenarios 22
2.4 How can I make one? 23
2.4.1 Designing the network 23
2.4.2 Manufacturing the network 33
2.4.3 Deploying the network 65

A THESIS ON NETWORK INFRASTRUCTURE
AND SELF-PRODUCTION

TENDRIL

FRANCESCO ZURLO FEDERICO BASSI ALESSANDRO IANNIELLO

2.1 What's the Tendril Repository?

The Tendril Repository is an entirely open-source collection of resources to design, manufacture and deploy your own ad-hoc wireless mesh network for P2P messaging, serving as a full-*stack* learn-by-doing experience aimed at regaining agency over the contemporary.

The repository functions as an easily approachable DIY project with comprehensive documentation on all of its phases, offering an iteratively scalable system for any further development without compromising beginner's accessibility. Go to the Tendril Repository GitHub[13] page to download all the above-stated resources by clicking on Code, then Download ZIP.

13

2.2 What's a wireless mesh network?

A mesh network, meshnet for short, is a decentralised network topology in which there is no hierarchy between network infrastructure, meaning that *nodes* are allowed to directly transmit information to every other *node* on the network to ultimately cooperate in routing data between clients.

Contrary to standard internet infrastructure, meshnets can dynamically self-organise and self-configure, reducing overhead and improving fault tolerance—to put it simply, as every *node* plays a small part within the larger infrastructure, meshnets are cheap and can be built with incremental complexity while also displaying remarkable resilience in the eventuality one or more *nodes* should fail.

2.3 What does this network allow me to do?

2.3.1 Features

The ad-hoc wireless mesh networks the Tendril Repository allows you to build employ an upper layer that focuses on peer-to-peer messaging. Meaning that thanks to a physical layer composed of a varying number of long-range radio transceivers called *nodes*, it becomes possible for network clients to share text messages. Being the networks ad-hoc, they provide no on-the-grid connectivity,

so, as much as they cannot access the internet, they also do not rely on it. Clients, once they access the network by interfacing with a *node*, can begin sharing low-data packets of information with other clients currently connected to the same network, regardless of internet coverage.

To put it simply, by pairing the *nodes* with a companion app, you can send and receive encrypted messages from your phone, tablet or computer, to and from other networked devices.

2.3.2 Scenarios

An ad-hoc wireless meshnet for P2P messaging has the potential to give coverage wherever internet providers cannot or do not want to, as well as offer an off-the-grid and fully encrypted alternative to standard messaging solutions.

An ad-hoc wireless meshnet for P2P messaging has the potential to give coverage wherever internet providers cannot or do not want to, as well as offer an off-the-grid and fully encrypted alternative to standard messaging solutions.

Additionally, the physical layer of the network can be configured to meet demands that go beyond the topic of P2P telecommunication, as *nodes* can be programmed and customised to suit many other purposes, and even combined with different non-mesh-like network infrastructure to employ alternative upper layers. This flexibility carries throughout all the project's phases, as these networks can be designed, manufactured and deployed with incremental levels of complexity in each and every step.

The application scenarios for these networks are therefore many and entirely up to whoever decides to make one, greatly varying according to how much resources and effort gets put into them. To provide a couple of examples:

Pursuing Privacy—In this scenario, a meshnet gets designed, manufactured and deployed for fully encrypted messaging between two or more clients desiring undisturbed privacy when communicating with each other.

This application would be best suited for both journalists and activist organisations looking for a safe and under-the-radar alternative to normal messaging solutions.

Jakub Geltner's 2015 sculpture *Nest 05*

Filling Coverage Gaps—In this scenario, a meshnet gets designed, manufactured and deployed to curb internet providers' shortcomings in areas that lack any internet or cellular coverage.

This application would be best suited for hikers, skiers and other outdoorsy people looking to communicate with each other, camps, shelters and—in the unfortunate case they were required—emergency service units for rescue.

Picture by Lucia Galiotto

Providing Fallback Infrastructure—In this scenario, a meshnet gets designed, manufactured and deployed to quickly intervene in the eventuality main infrastructure should fail.

This application would be best suited for extreme emergency situations in which the rapid deployment of a network could allow for critical communications to run unbroken, or at least until primary infrastructure goes back to working order.

Picture by Sadatsugu Tomizawa during the 2011 Tohoku earthquake

2.4 How can I make one?

2.4.1 Designing the network

In this phase—we understand how *nodes* communicate with each other and apply what we learned about network topology by designing a custom meshnet architecture.

Nodes are radio transceivers that make up the physical layer of the network and together define its topology.

Nodes to interface with client devices like phones, tablets and computers, use Bluetooth Low Energy (BLE), whereas, to transmit data between each other, employ LoRa: a radio transmission technique for LPWANs that operates within licence-free ISM radio bands. LoRa offers long-range connectivity up to a declared

maximum of 5 km in urban areas, and, provided there is a line of sight (LoS) between the *nodes*, up to 15 km in rural areas, achieving data rates between 0.3 kbit/s and 27 kbit/s.[14]

14 Semtech Corporation, 2022

The number, position and class of *nodes* determine how large the network is, how many clients it has, and, ultimately, its purpose. By determining how many, their projected position, and the most suitable combination between their available classes, with *nodes* is possible to cover a wide array of use cases, as well as imagine new and exciting scenarios in which a custom meshnet could be useful. The network designing phase is divided into three steps: Numbering, Positioning and Classifying.

Works cited:

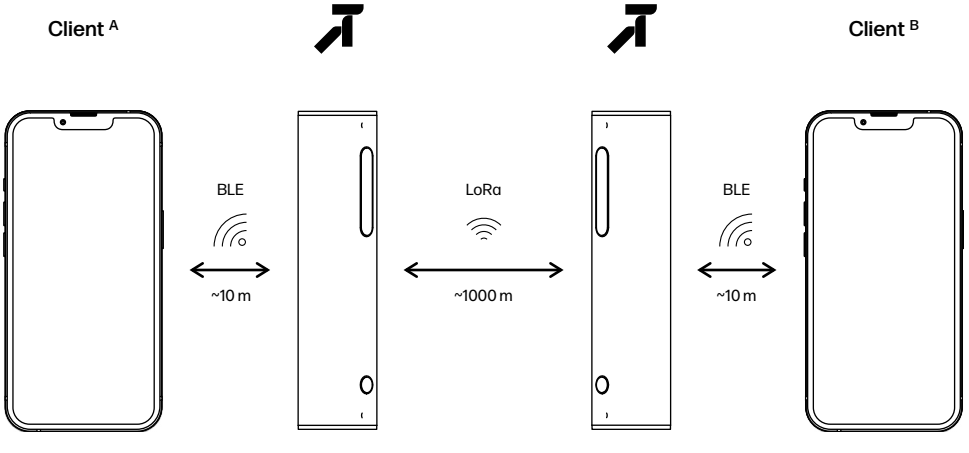
Semtech Corporation. (2022). LoRa and LoRaWAN: Technical overview. Retrieved from https://lora-developers.semtech.com/documentation/tech-papers-and-guides/lora-and-lorawan/

Numbering

In this step—we determine the number of *nodes* in our custom meshnet architecture.

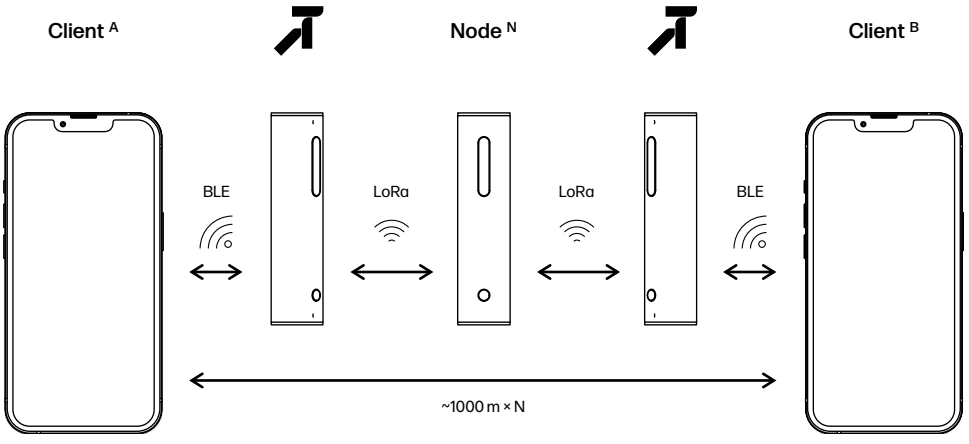
The number of *nodes* on the network affects two things: the number of active network clients and the overall network coverage.

The number of active network clients gets affected by the number of *nodes* as the clients themselves are required to be near one to operate the network (within Bluetooth range), making it safe to assume the number of *nodes* always being greater than or, at least, equal to the number of active clients on the network.The overall network coverage gets affected as meshnet topology extends its range cumulatively—meaning that, given how *nodes* can cooper-



An example of messaging between 2 clients via 2 *nodes*

ate in routing data between clients: the more the *nodes*, the larger the network, and the further the data can travel.
For a network to function, the minimum required number of *nodes* is 2 and the maximum, as of 2022, is 32.

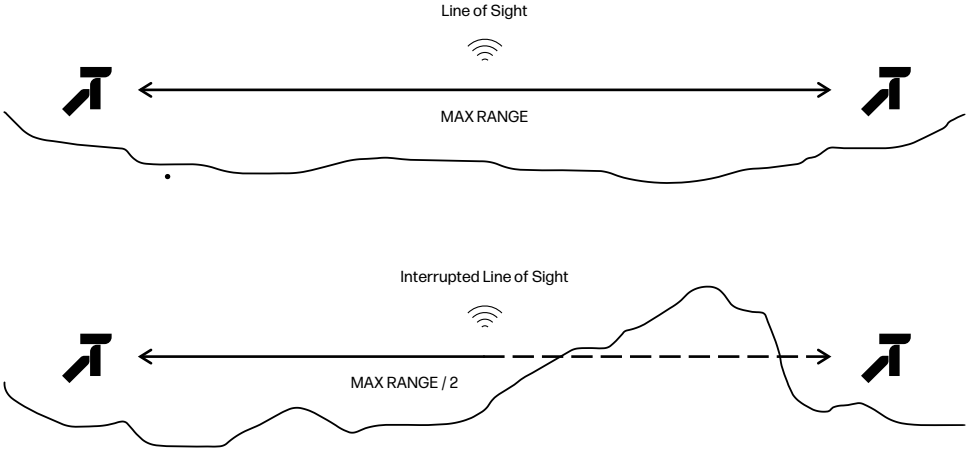


An example of messaging between 2 clients via an N number of *nodes*

Positioning

In this step—we determine how the *nodes* in our custom meshnet architecture could be positioned.

How *nodes* are positioned within the network fundamentally impacts its size, shape and degree of mobility.



An example between a clear LoS transmission and a, less ideal, interrupted one.

Nodes are all equipped with omnidirectional antennas—meaning that, in three-dimensional space, they display a radiation pattern that can be simplified to a sphere varying in size according to their position, as these spherical radiation patterns have a gradient of effectiveness that gets negatively affected by distance and interference (e.g. buildings, trees and terrain).

Cleverly positioning *nodes* in three-dimensional space for both maximising coverage and LoS between them, can therefore greatly affect the performance of a network.

Whenever it is not possible to do so, increasing the number of *nodes* can always help mitigate this problem.

Classifying

In this step—we determine to which combination of classes the *nodes* in our custom meshnet architecture belong.

Nodes are modular and, as of now, come in four classes that can be freely upgraded or downgraded from each other with minimal loss of parts.

Each of them tries to cover a different role within the network's architecture, but how they will get employed, ultimately comes down to your preference and imagination.

* Coverage is indicative, omnidirectional and with no LoS unless stated otherwise

** Cost is indicative and excludes tooling, however, it includes a generic tripod for RES *nodes* and a generic drone for AIR *nodes*

*** Coverage with optional 3dBi SubG Antenna Radio Module

**** USB-C powered

Coverage*: ~500 m
Cost:** ~25 $
Board: Standard, GNSS GPS (optional)
Radio Module: Standard
Power Module: N/A****
Cover: Ø 35 × 72 mm
Attachments: N/A

Developer *node* (DEV)—USB-powered entry-level *node* variant meant for beginners and developers.

DEV *nodes* offer a cheap introduction to LPWAN mesh networking as well as a reliable starting point to further build upon.

This class of *nodes* is recommended for all *makers* wanting to dip their toes into the Tendril Repository without any substantial overhead costs.

Coverage*:	~500 m, >1000 m***
Cost:**	~50 $
Board:	Standard, GNSS GPS (optional)
Radio Module:	Standard, External 3dBi SubG Antenna (optional)
Power Module:	Rechargeable Battery
Cover:	Cover: Ø 35 × 142 mm
Attachments:	Salomon Quicklace Netting (optional)

Nomad *node* (NMD)—Battery-powered *node* variant meant for roaming network clients.

NMD *nodes* offer versatility and performance that make them suitable for most mesh networking use cases.

This class of *nodes* is recommended for all *makers* wanting a straightforward personal *node*, or those looking to simply increase the number of clients on their network.

NMD "PSY"

01

TENDRIL
NMD "PSY"
01

Coverage*:	LoS ~3000 m
Cost:**	~150 $
Board:	GNSS GPS
Radio Module:	Standard
Power Module:	Rechargeable Battery
Cover:	Ø 35 × 142 mm
Attachments:	× 2 Voltaic Systems 1 Watt 6 Volt Solar Panels, Hi-vis *node* ID, ¼" Mounting Support for Tripods

Resident *node* (RES)—Solar-powered *node* variant meant for permanent stationary deployment.

RES *nodes* can solidify a network's topology by extending coverage for all clients, performing best when positioned with LoS between them.

This class of *nodes* is recommended for all *makers* wanting their mesh network to permanently and reliably cover a specific area of land.

04\05

Coverage*:	LoS >10000 m
Cost:**	>1000 $
Board:	Standard
Radio Module:	External 3dBi SubG Antenna
Power Module:	Rechargeable Battery
Cover:	Ø 35 × 142 mm
Attachments:	¼" Mounting Support for Drone Camera Brackets

Airborne *node* (AIR)—
Drone-mounted *node* variant meant for emergency deployment.

AIR *nodes* can function as temporary RES *nodes* whenever LoS is not immediately available.

This highly specialised class of *nodes* is recommended for all *makers* wanting to quickly deploy a large-sized network over a specific area of land in the eventuality the situation called for extreme levels of urgency.

TENDRL
ARF

TENDRL

In this phase—we understand how *nodes* are made and apply what we learned by manufacturing the required amount to satisfy our custom meshnet architecture.

Nodes, to both suit even the most demanding needs as well as keep up with tech advancements in the LoRa ecosystem are designed with modularity and customization in mind.

To reflect this, *nodes* are built around the RAKwireless' RAK WisBlock family of products for IoT applications: an industrial-grade modular system of baseboards, MCUs, modules and accessories, that includes an nRF52 MCU for LoRa radio transmission. The ever-evolving RAK WisBlock ecosystem offers a straightforward and extensively documented *maker* experience that does not require any soldering or electronic engineering knowledge, lowering entry barriers without compromising any later and more advanced experimentation.

To further reinforce this, the FDM printed *node* enclosures are also modular and are designed for fast assembling and disassembling with zero fasteners—as snap fits and Lego components allow for a friendly and tool-free approach to *node* manufacturing. Alongside RAKwireless and Lego, *nodes* also make use of parts coming from a series of other trustworthy suppliers like Voltaic Systems, Adafruit and even Salomon.

Together FDM fabricated parts and components sourced from this combination of suppliers establish a familiar design language that runs across all *node* classes and accessories, encouraging all your further hardware-related endeavours.

The network manufacturing phase is divided into three steps: Sourcing, Fabricating and Assembling.Before starting the manufacturing phase, go to the Tendril Repository GitHub[15] page and download all the required materials by clicking on Code, then Download ZIP. Also, make sure you have access to the following equipment:

15

- Computer with internet access and 3D printing slicing software, as well as a vector graphic editor for making Labels and Node IDs.
- FDM 3D Printer.
- Soldering iron with solder, small heat shrink sleeves and splicing scissors (not needed for DEV *nodes*).
- Generic set of tools that must include a compatible screwdriver for M1.2 RAK WisBlock screws.

Sourcing

In this step—we source all the required components for the given number and class of *nodes* in our custom meshnet architecture.

Open the Sourcing/Fabricating Google Sheet listed in the GitHub repository to figure out what components you will need to source. Then, place all the required orders.

Please note that:

Components are sourced internationally, checking local suppliers before placing orders can help mitigate shipping costs.

Lego components are sourced from BrickLink. Follow BrickLink's New User Tutorial[16] for a quick rundown on how it works.

16

Be careful of ordering the Arduino version of the RAK4631.

RAK4631s and 3dBi SubG Antennas have to be ordered specifying their frequency band, so be mindful of choosing the one that is compliant with your national ISM regulations and sticking to it across all the *nodes* in your network.

Europe	**EU433**	Europe	**EU868**	Korea	**KR920**
China	**CN470**	Canada	**US915**	Asia	**AS923**
India	**IN865**	Australia	**AU915**	Russia	**RU864**

Fabricating

In this step—we fabricate all the required parts for the given number and class of *nodes* in our custom meshnet architecture.

Open the Sourcing/Fabricating Google Sheet listed in the GitHub repository to figure out what parts you will need to fabricate. Then, slice the STLs with a 3D printing slicer software of your choice and begin printing.

Please note that:

All STLs have been designed for FDM 3D printing with Autodesk Inventor, sliced with Prusa Slicer and have been tested on a stock Creality Ender 3 V2 using SUNLU PLA+.

Each part is to be fabricated following the notes stated in the Google Sheet.

All STLs are to be printed with a 0.2 mm layer height, 3 outer perimeters, with no supports, in the orientation they enter the slicer with (unless stated otherwise in the Fabrication Notes). All omitted slicer-related information is entirely up to your own preferences.

Before fabricating the 3D-printed parts, printer calibration is recommended. To do so, refer to Teaching Tech's "3D Printer Calibration" GitHub page[17].

17

Print Lego_Test_Pin and Lego_Test_Axle to check your tolerances. If both of them behave like Lego when connected with Lego Pins and Axles proceed with printing all other parts. If the connections are too tight, consider recalibrating your printer and try again until you meet the expected behaviour. If the connections are too loose, consider increasing your extrusion multiplier in 5% (+0,05) increments and try again until you meet the expected behaviour. Printing the outer perimeters first can help with improving tolerances across all components, check to see if your slicer can enable this feature.

Labels and Node IDs will require you to outsource printing & laser-cutting processes. Edit the two .ai and export files for printing and cutting. Alternatively, use/reference the two .pdf to make your own designs.

Assembling

In this step—we assemble all *nodes* in our custom meshnet architecture.

Refer to the following steps to figure out how to assemble your *nodes* according to their class.

Please note that:

No matter what, avoid powering the boards if they are not fully assembled (RAK19003 Mini Base Board + RAK4631 LPWAN Module with both LoRa and BLE antennas firmly plugged in. If you have a RAK12500 GNSS GPS Location Module installed on your Base Board make sure the GPS antenna is firmly plugged in too.

18

You can refer to the RAK Documentation Center[18] for any further guidance throughout the board assembly process.

RES *nodes* do not have external USB-C access, meaning that, to flash the boards you will be required to take them apart. For the sake of simplicity is therefore recommended to fully assemble all the boards you have beforehand and jump to the Flashing and Configuring steps of the deployment phase before proceeding any further.

Developer *node* (DEV)

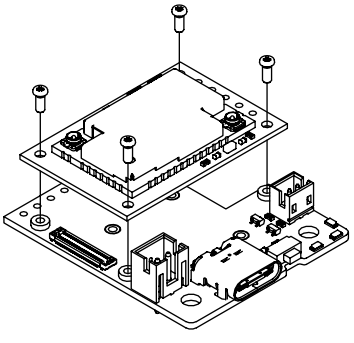
A Assembling the Board

Before proceeding, read the following documents:

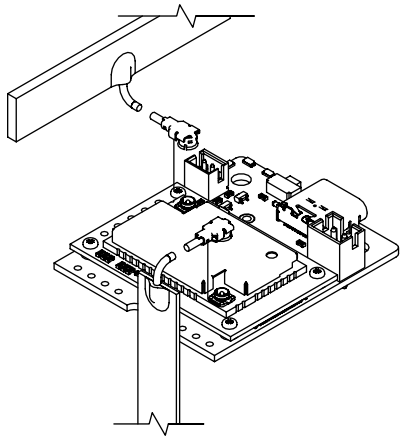
- WisBlock Quick Start Guide[A]
- RAK19003 Quick Start Guide[B]
- RAK4631 Quick Start Guide[C]
- RAK12500 Quick Start Guide[D] (if you plan on installing one)

A B C D

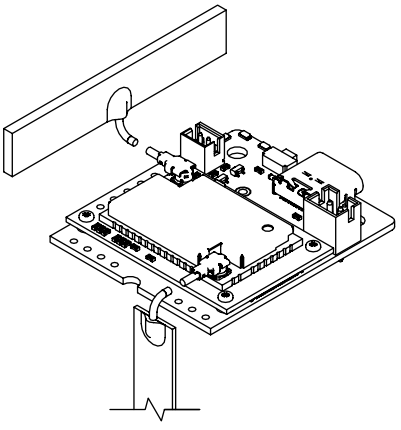
1/1



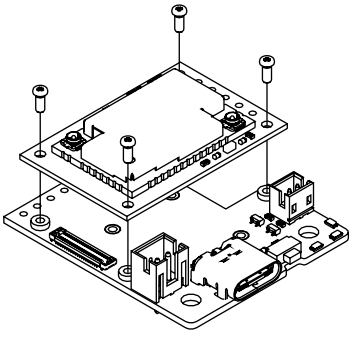
(Standard) 1



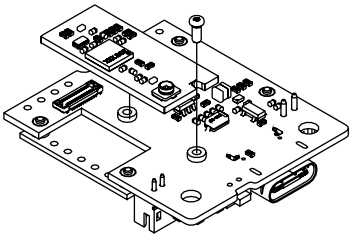
2



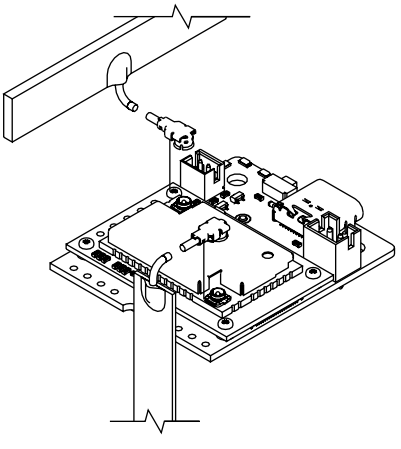
3



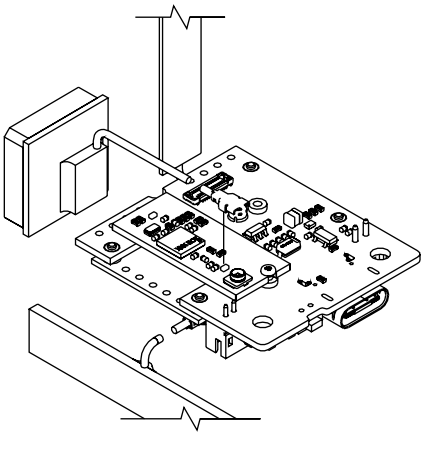
(GNSS) 1



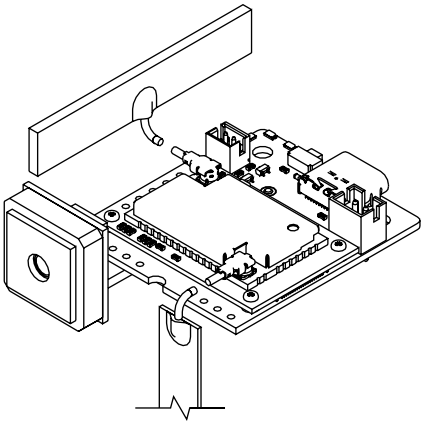
2



3

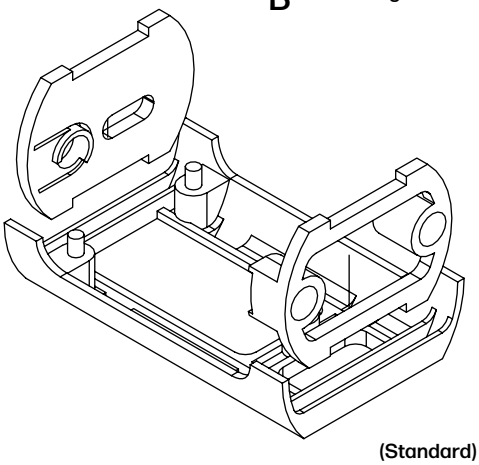


4

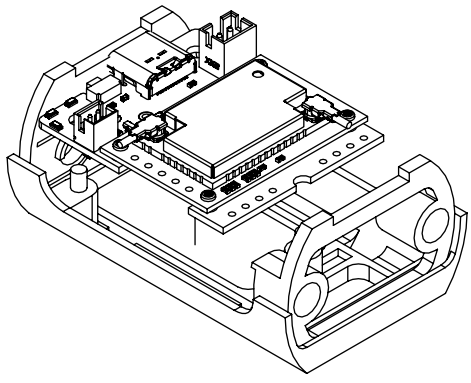


5

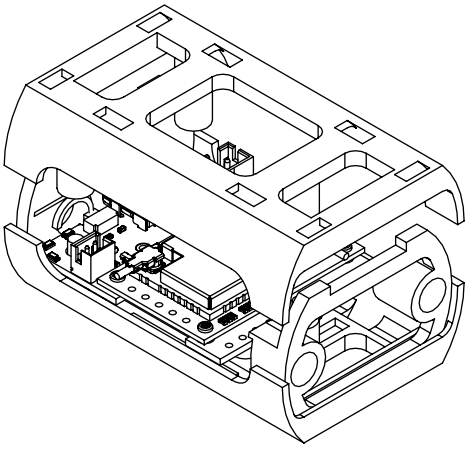
B Assembling the Radio Module



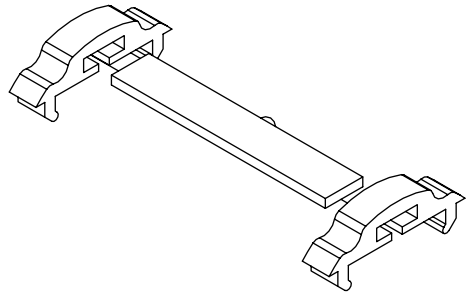
(Standard) 1



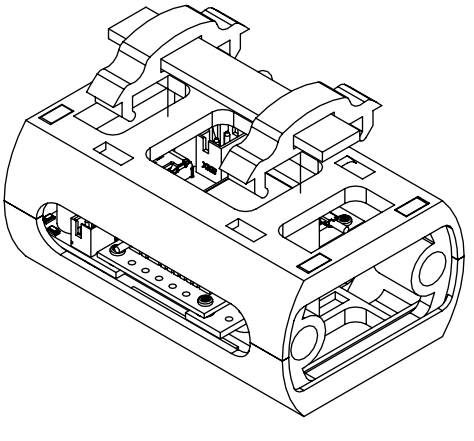
2



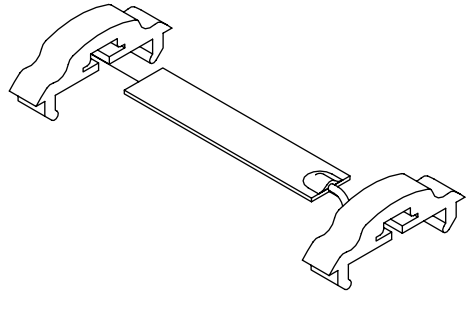
3



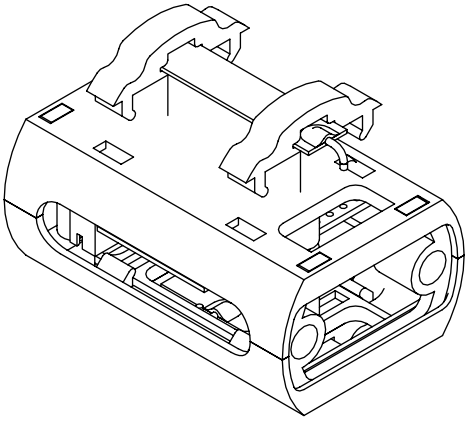
4



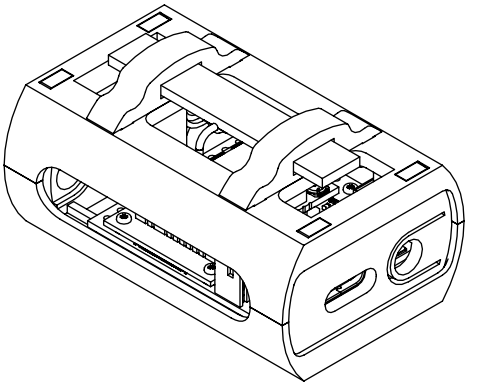
5



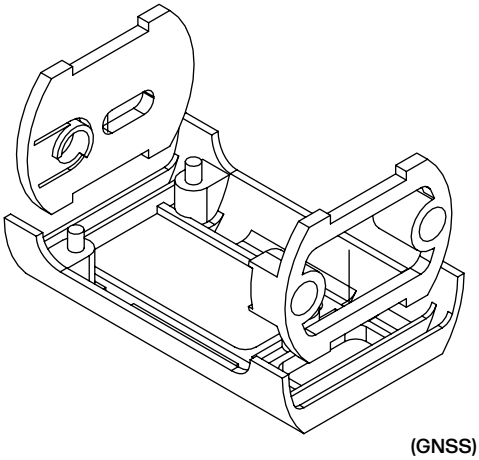
6



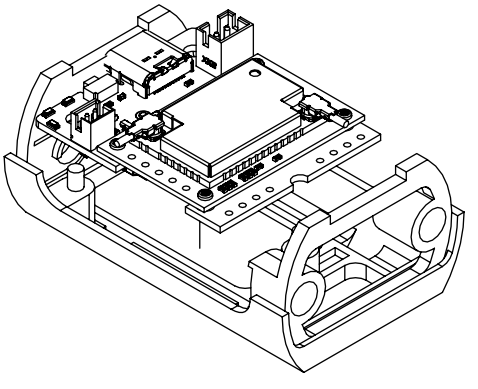
7



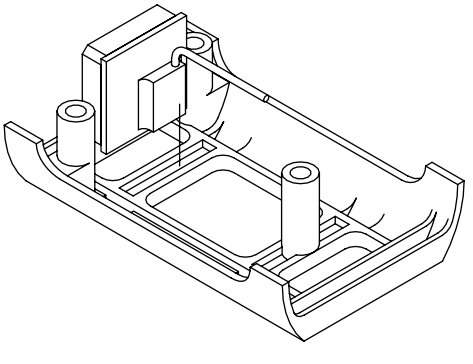
8



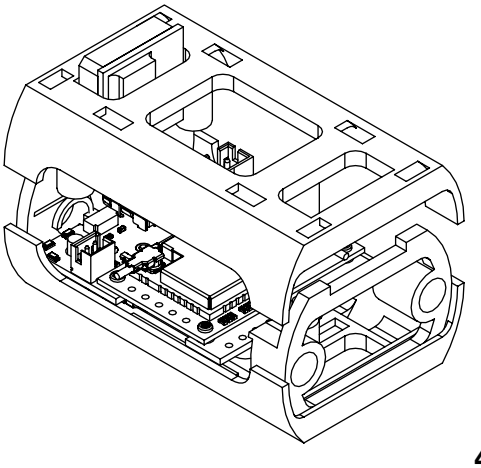
(GNSS) 1



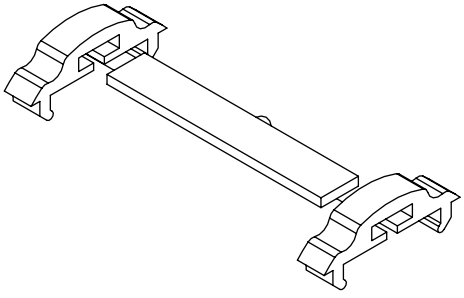
2



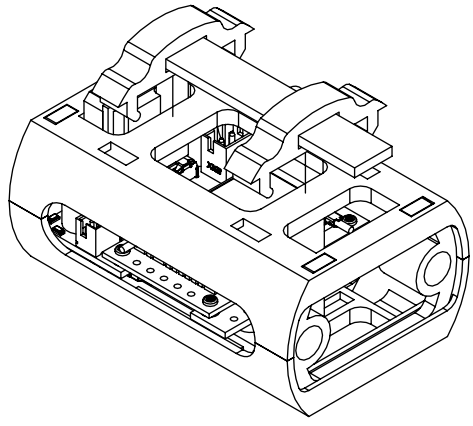
3



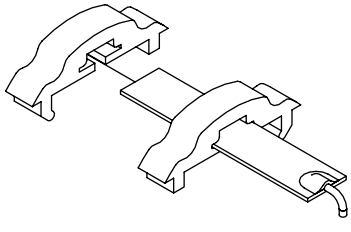
4



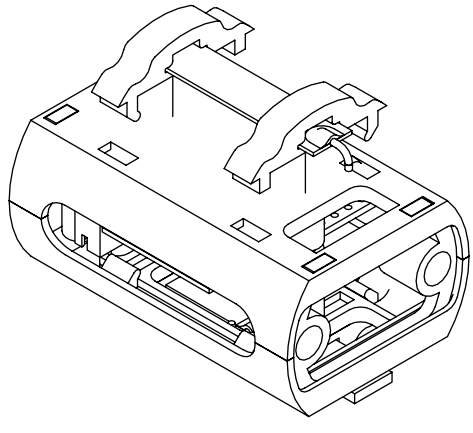
5



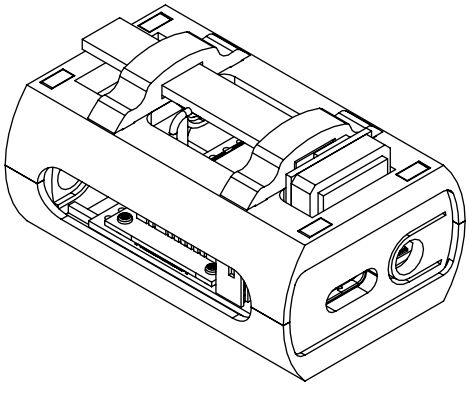
6



7



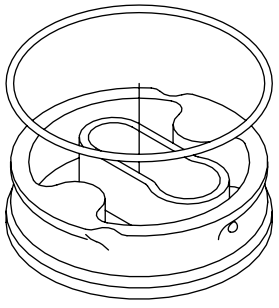
8



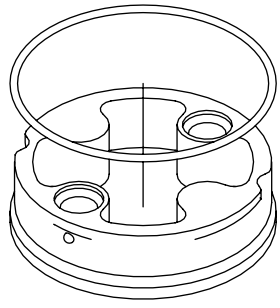
9

1/1

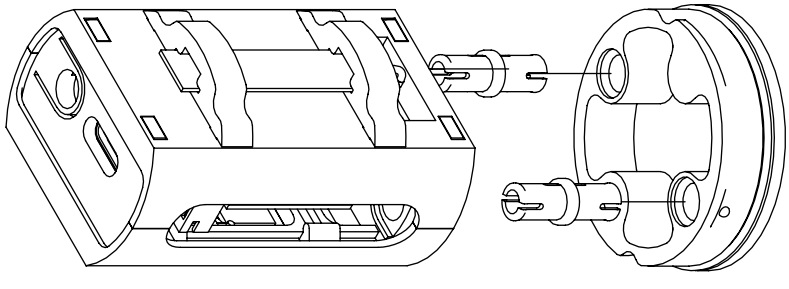
C Assembing the Cover



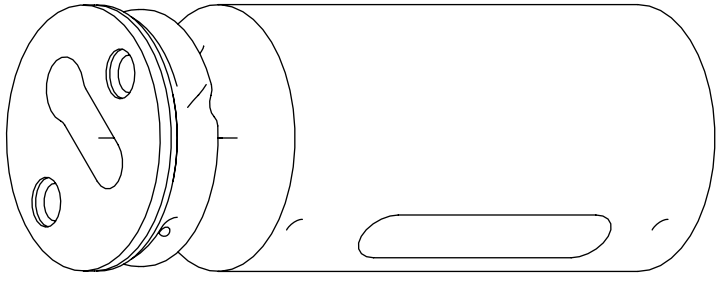
1



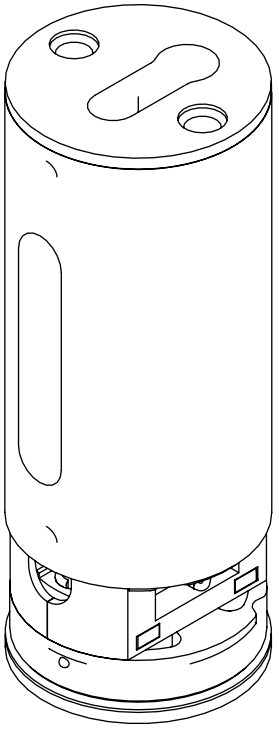
2



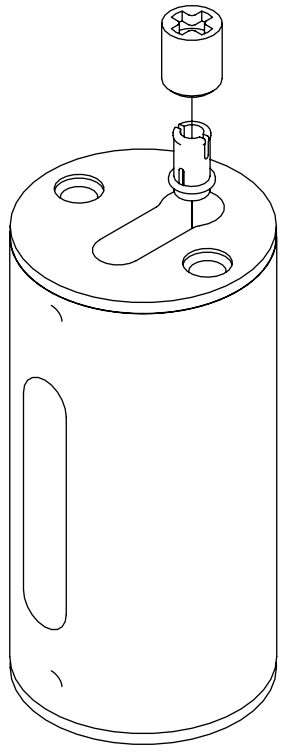
3



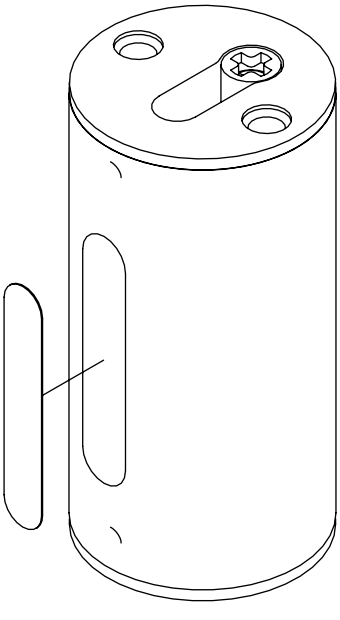
4



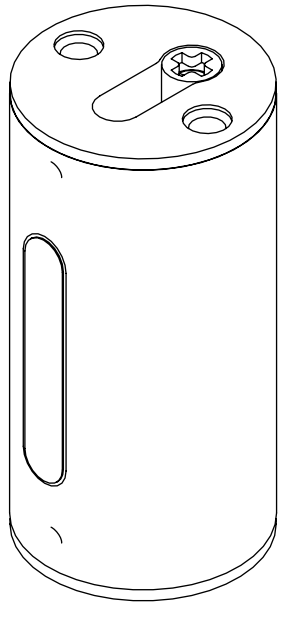
5



6



7



8

Nomad *node* (NMD)

A Assembling the Board

Read the following documents and refer to point A in the Developer *node* assembly step until the Board is done.

- WisBlock Quick Start Guide[A]
- RAK19003 Quick Start Guide[B]
- RAK4631 Quick Start Guide[C]
- RAK12500 Quick Start Guide[D] (if you plan on installing one)

A B C D

After having assembled one board, it is recommended to do the same for all the other *nodes* in your network and jump to the Flashing and Configuring steps of the deployment phase.

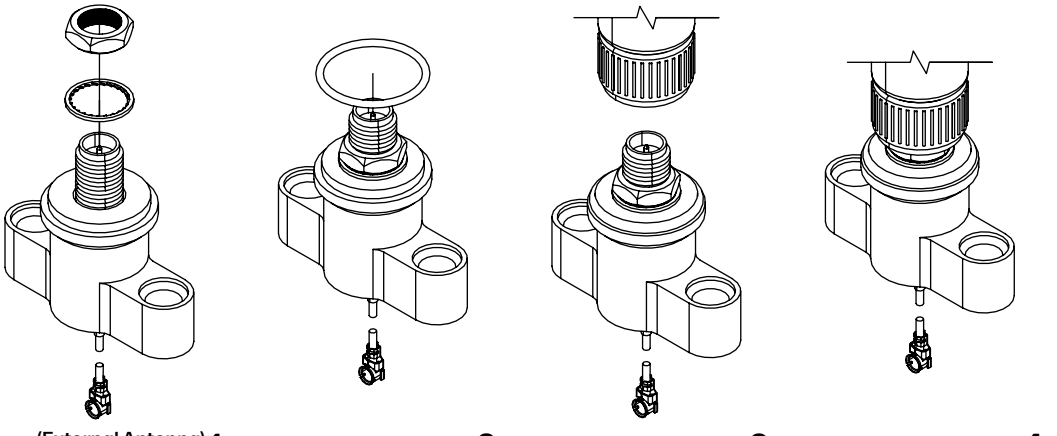
Only proceed after having made sure everything is in working order.

B Assembling the Radio Module

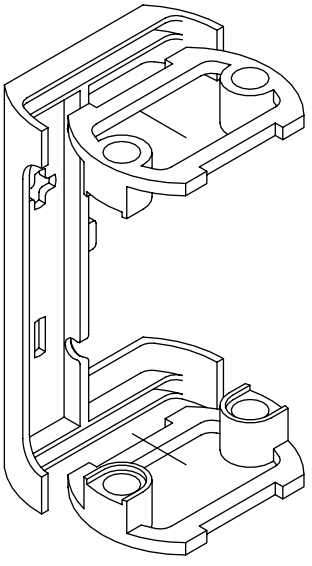
If you plan on installing an external antenna refer to the following instructions, then refer to point B in the Developer *node* assembly step until the Radio Module is done.

If not, refer straight away to point B in the Developer *node* assembly step until the Radio Module is done.

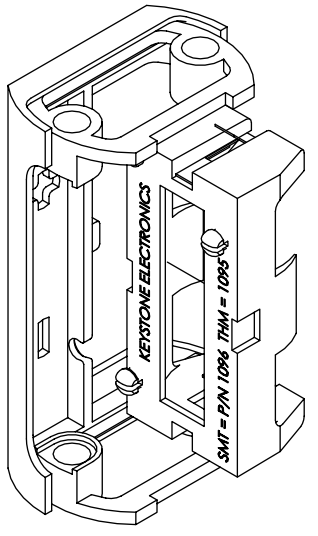
1/1



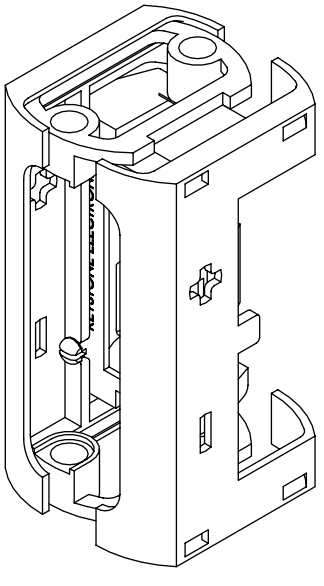
C Assembling the Power Module



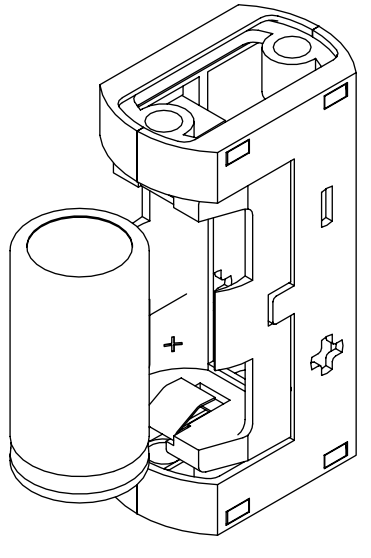
1



2



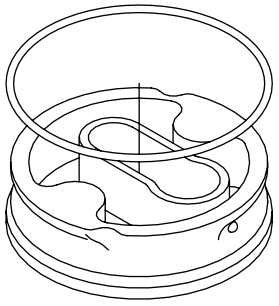
3



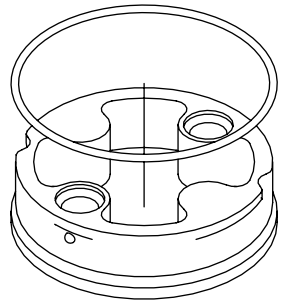
4

1/1

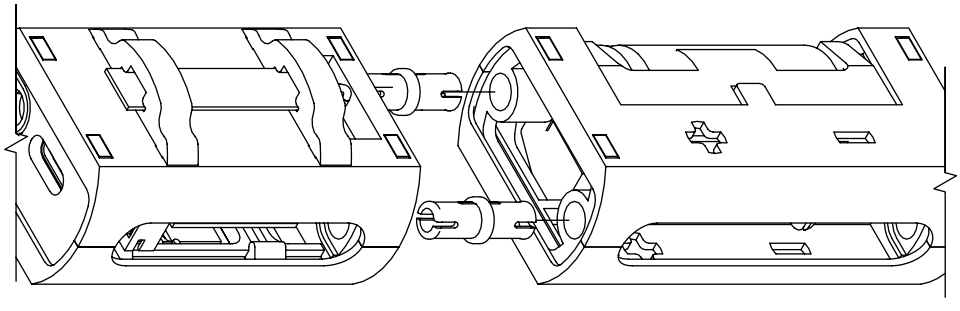
D Assembling the Cover



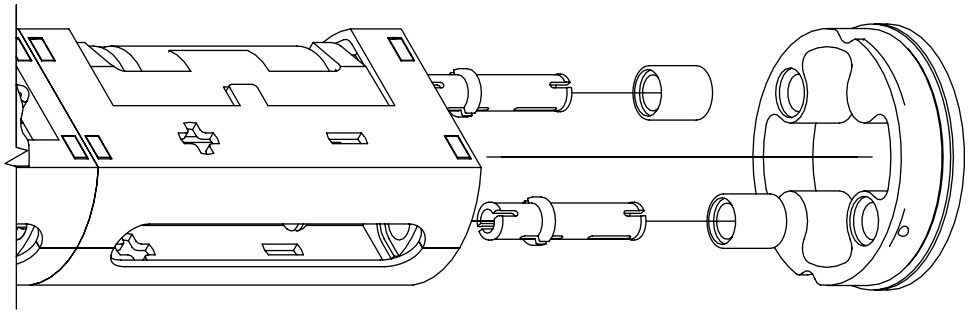
(Standard) 1



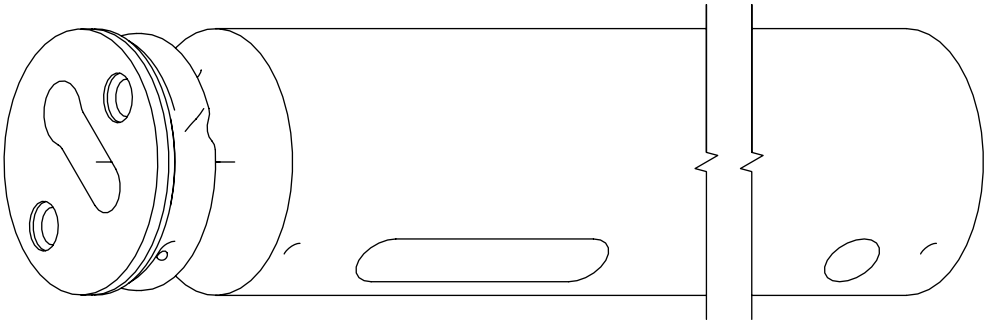
2



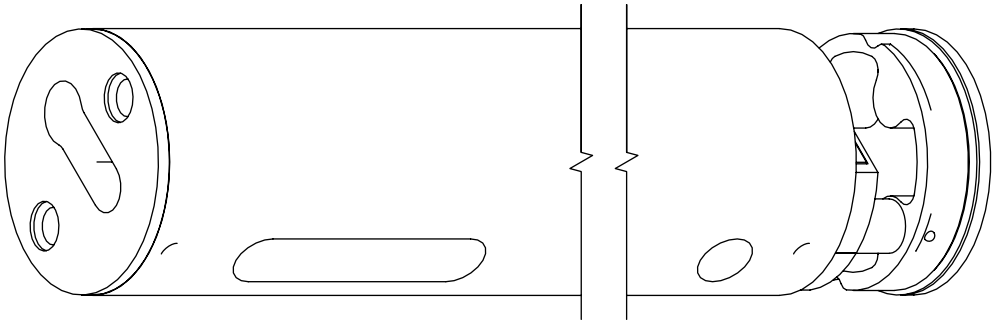
3



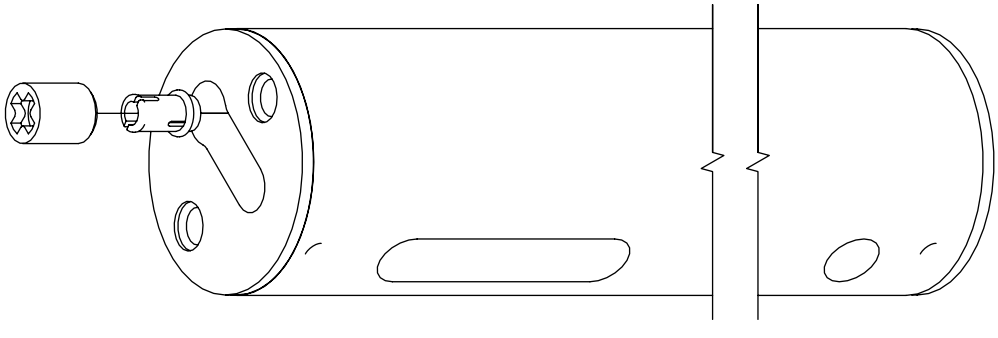
4



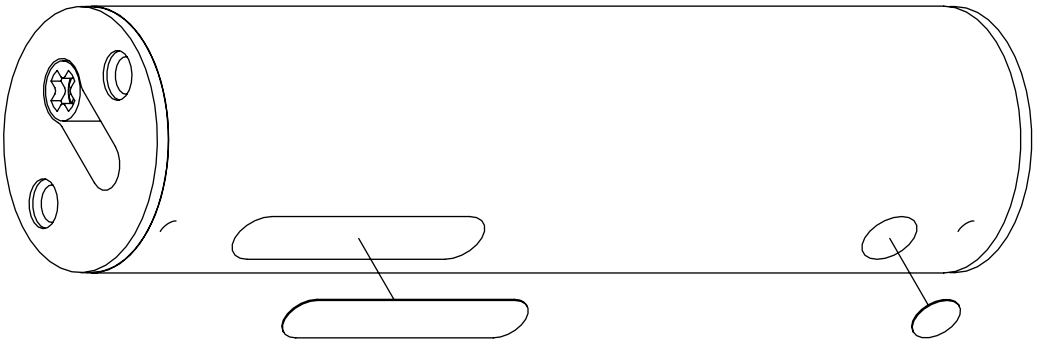
5



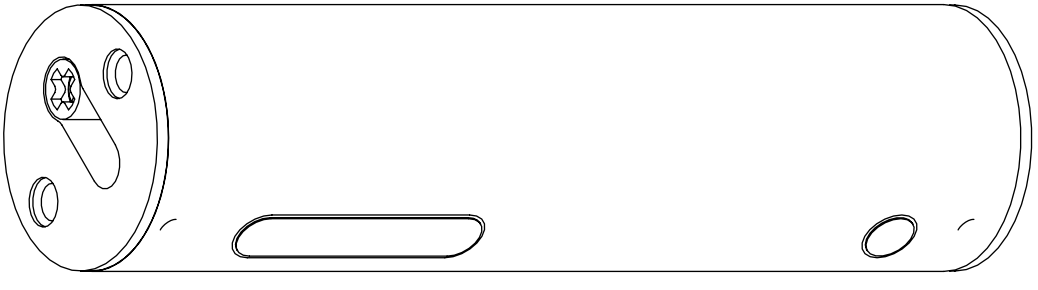
6



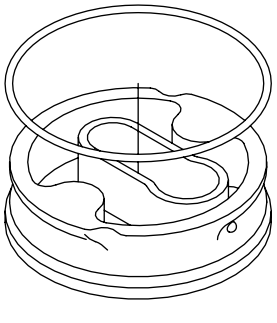
7



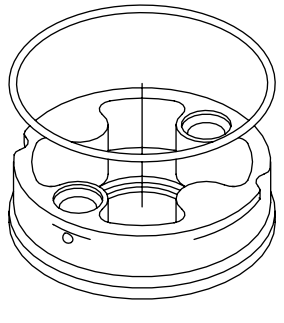
8



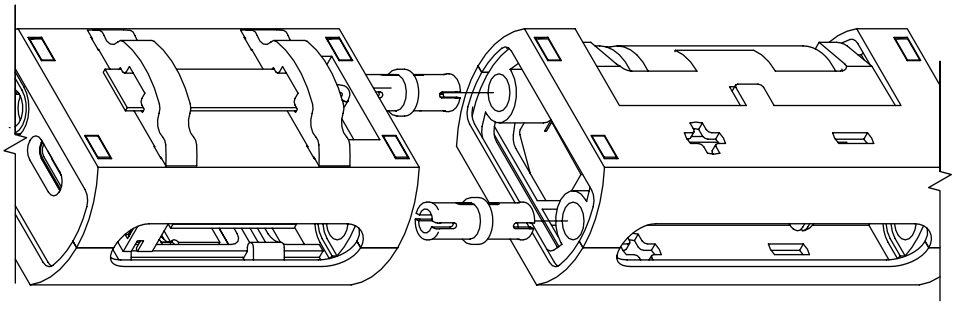
9



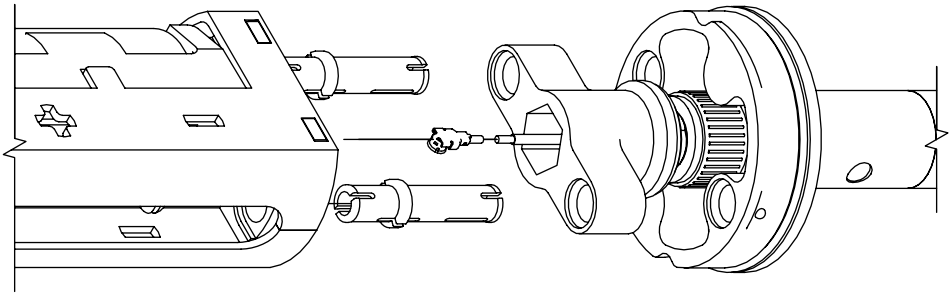
(External Antenna) 1



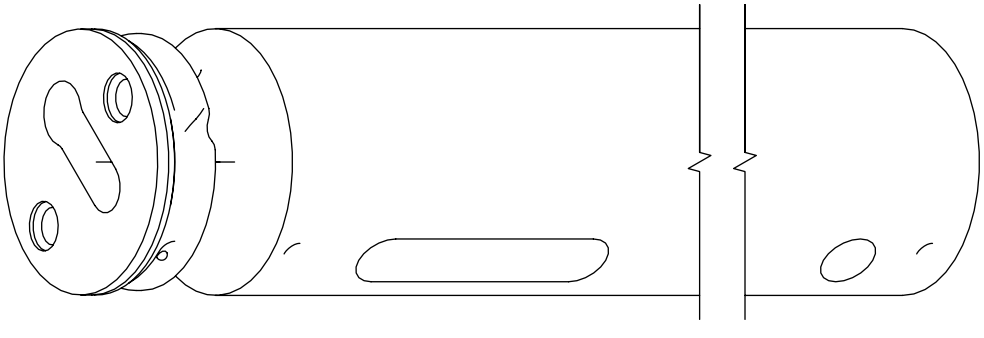
2



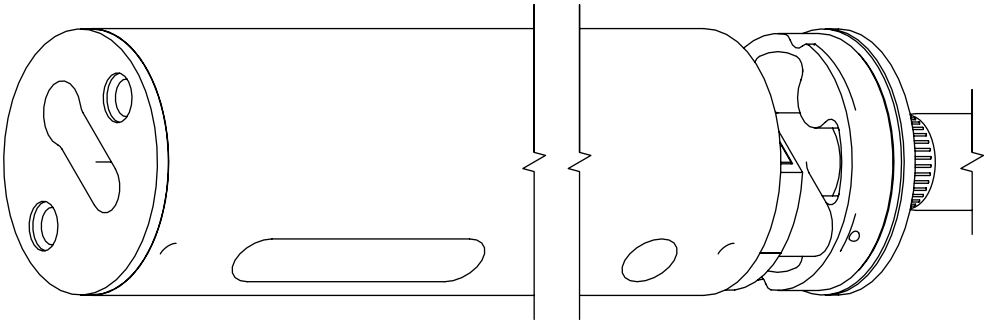
3



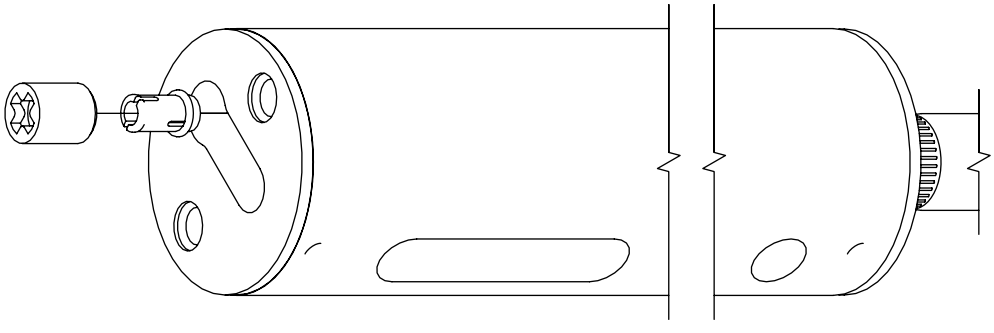
4



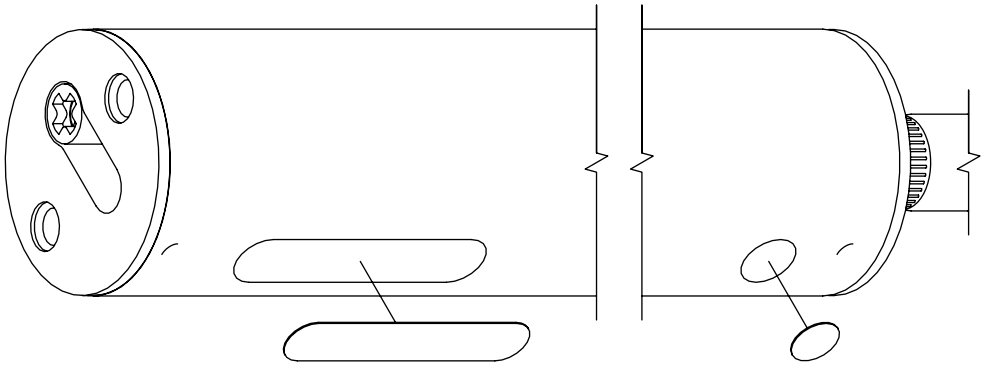
5



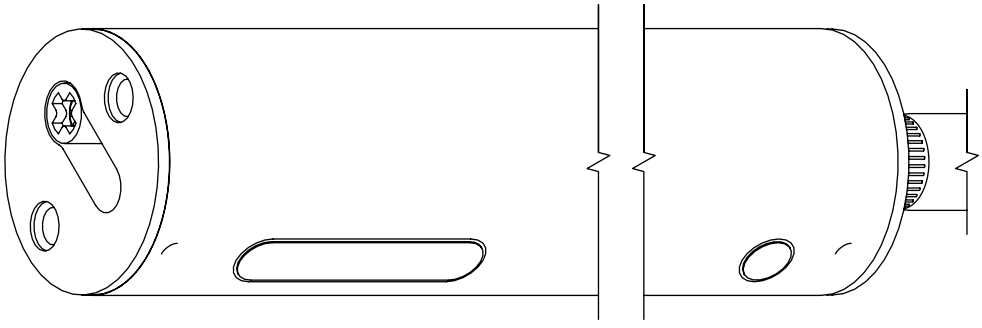
6



7



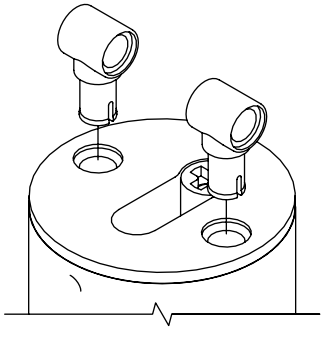
8



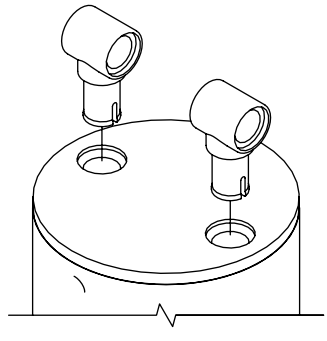
9

1/1

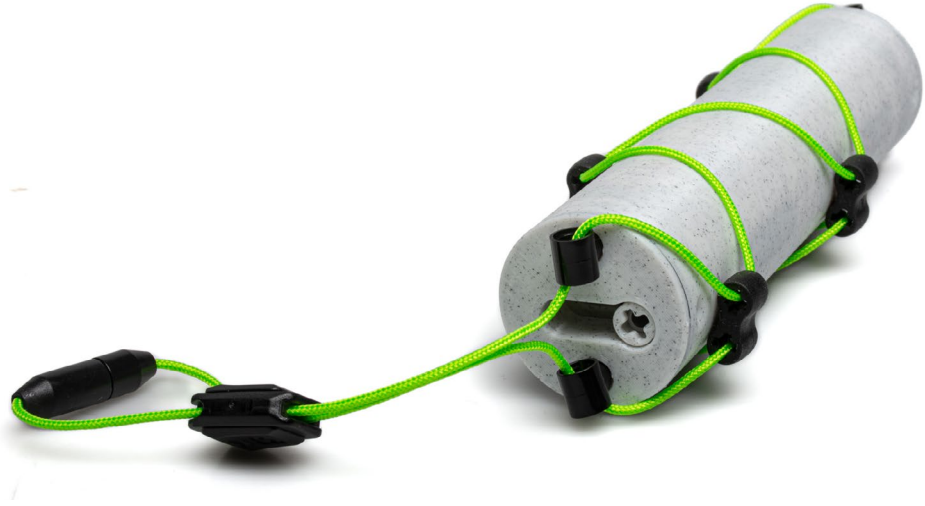
E Assembling the Attatchments



1



2



3

Resident *node* (RES)

A Assembling the Board

Read the following documents and refer to point A (GNSS) in the Developer *node* assembly step until the Board is done.

- WisBlock Quick Start Guide[A]
- RAK19003 Quick Start Guide[B]
- RAK4631 Quick Start Guide[C]
- RAK12500 Quick Start Guide[D]

A B C D

After having assembled one board, it is recommended to do the same for all the other *nodes* in your network and jump to the Flashing and Configuring steps of the deployment phase.

B Assembling the Radio Module

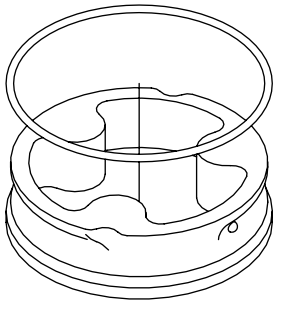
Refer to point B in the Developer *node* assembly step until the Radio Module (GNSS) is done.

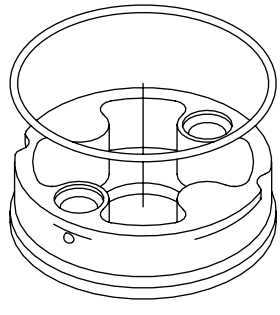
C Assembling the Power Module

Refer to point C in the Nomad *node* assembly step until the Power Module is done.

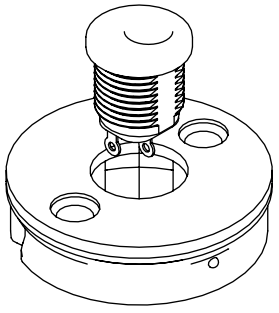
1/1

D Assembling the Cover

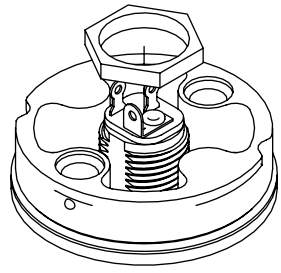




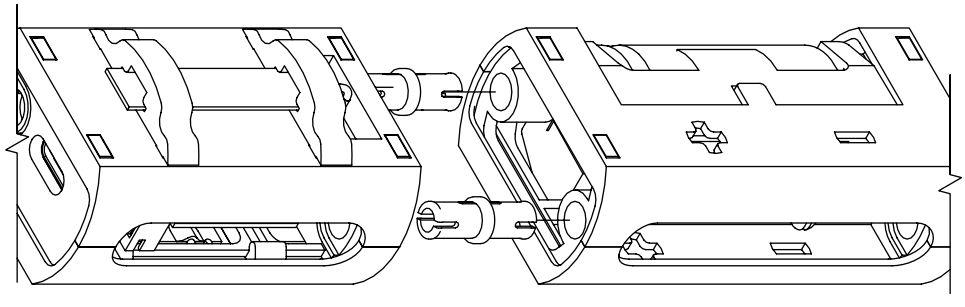
1 2



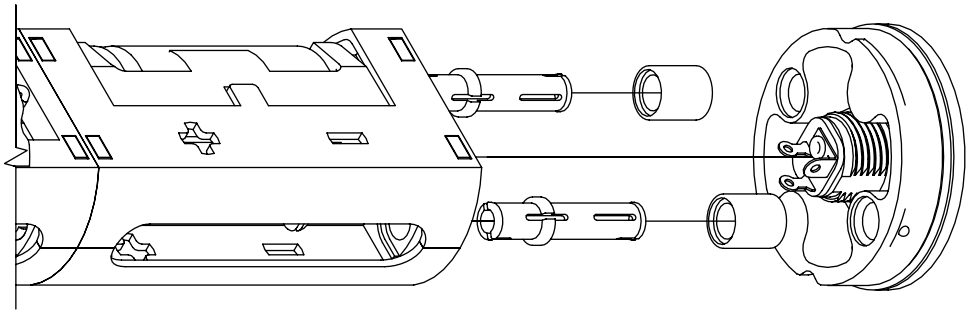
3



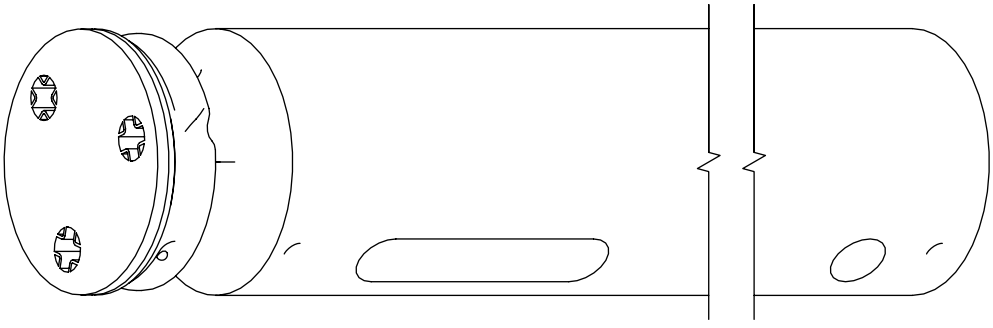
4



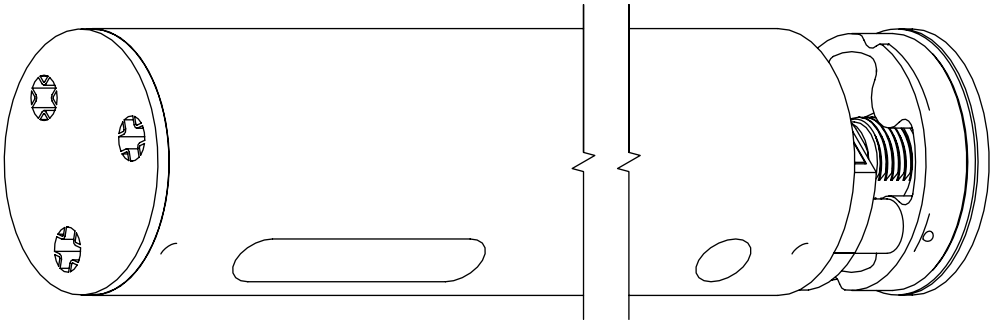
5



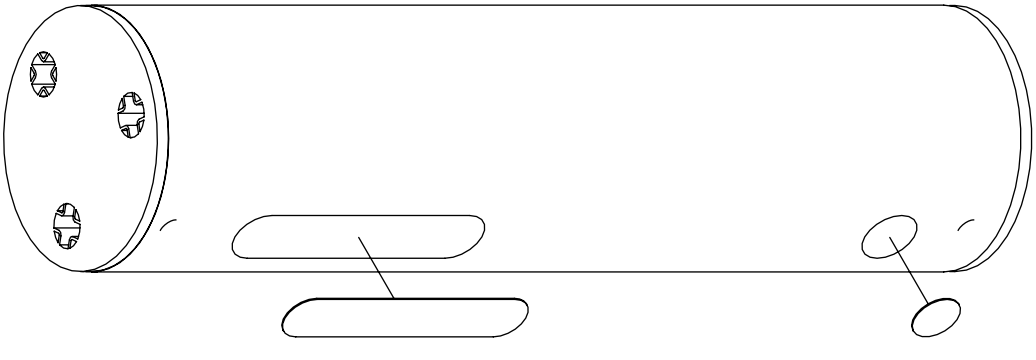
6



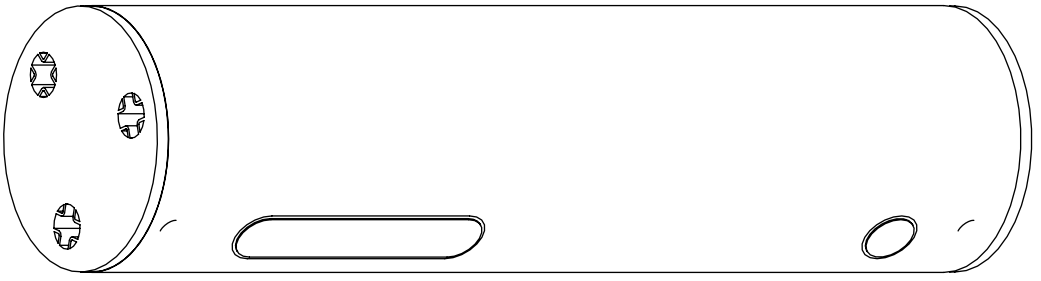
7



8



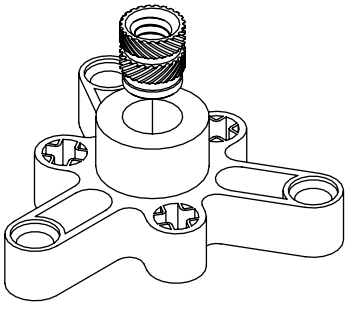
9



10

1/1

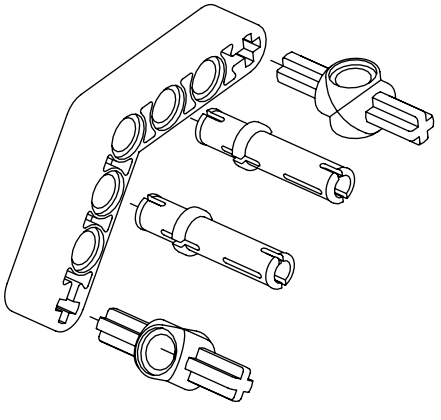
E Assembling the Attatchments



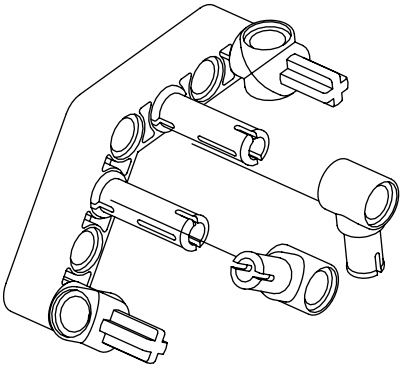
Use the soldering iron to press the Threaded Inserts in. In doing so, keep the soldering iron vertical and the Threaded_Base stable on a flat surface.

When the tip hits the surface (through the bottom hole) the Threaded Insert should be fully seated.

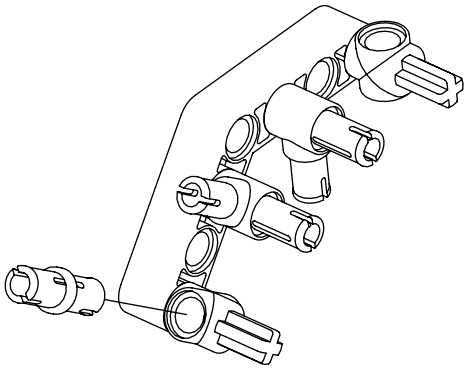
1



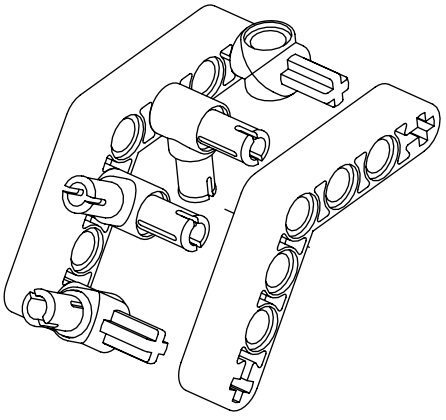
2



3

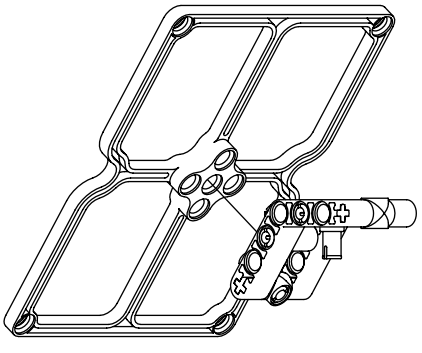


4

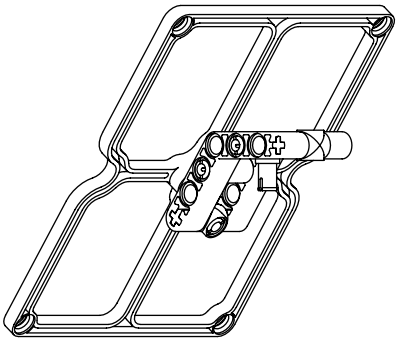


5

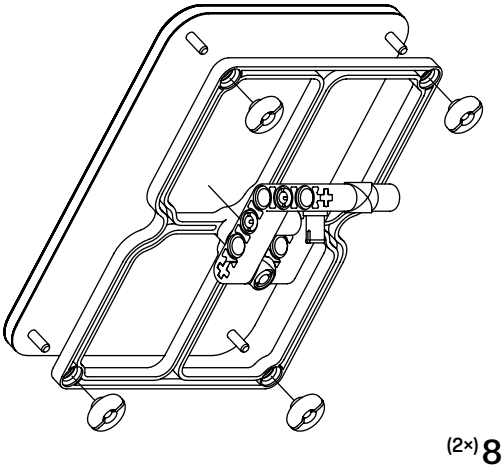
½



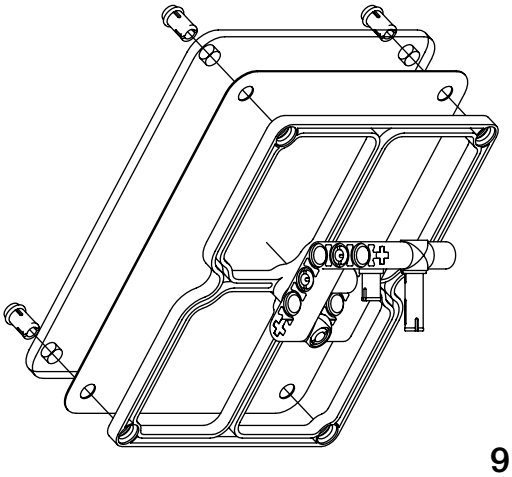
6



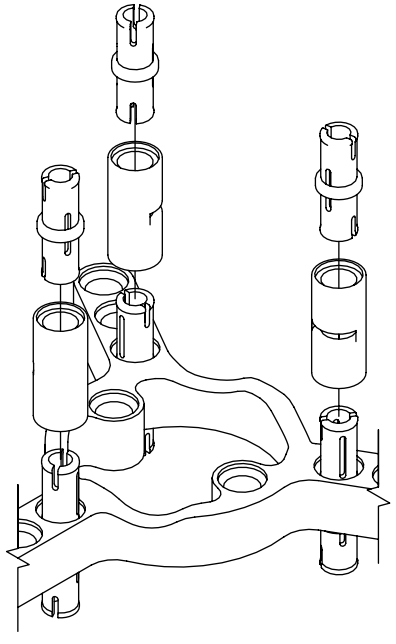
(3×) 7



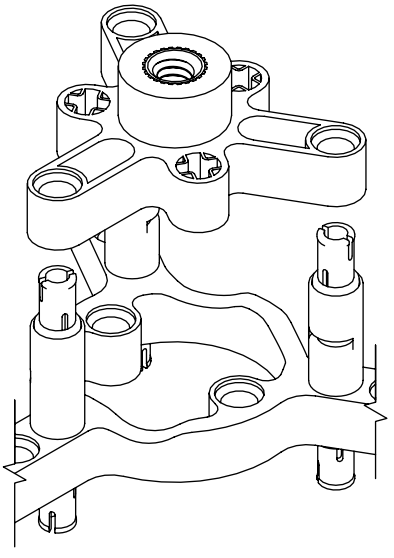
(2×) 8



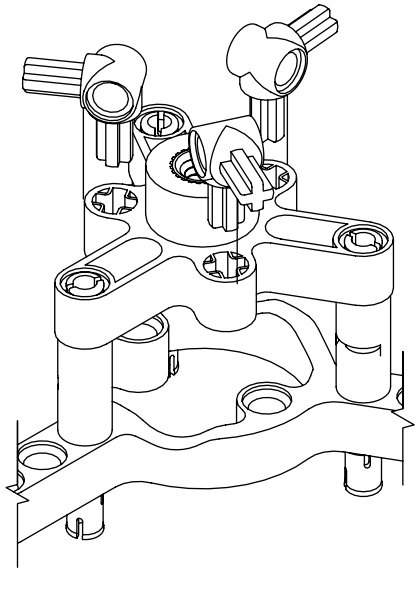
9



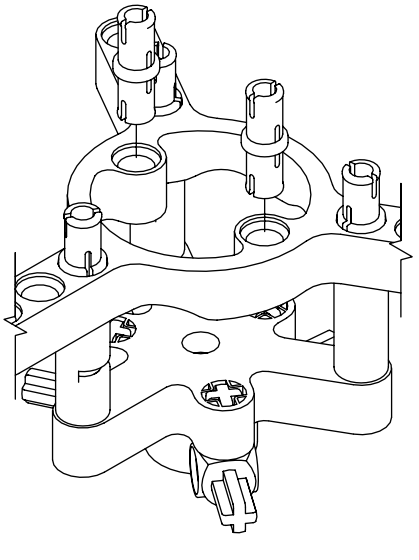
10



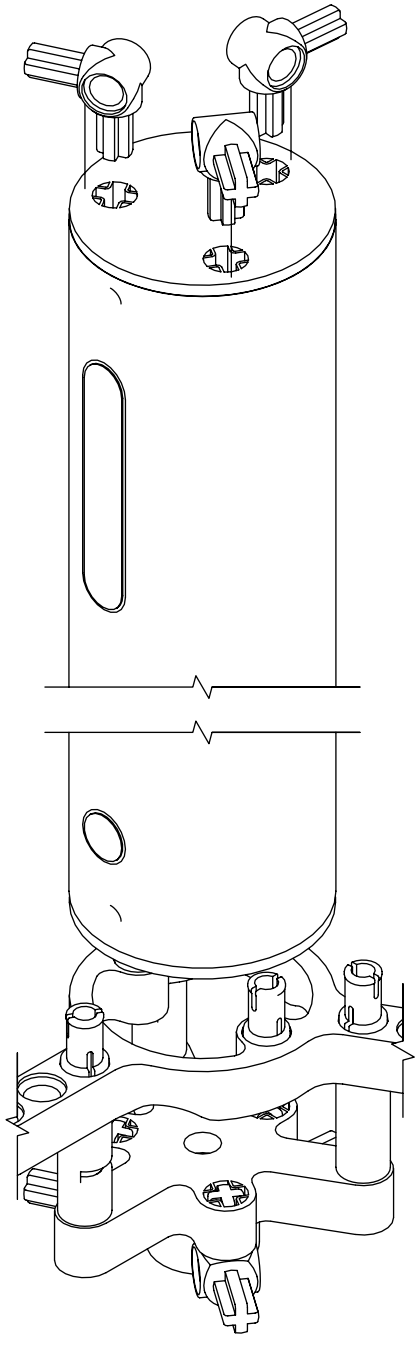
11

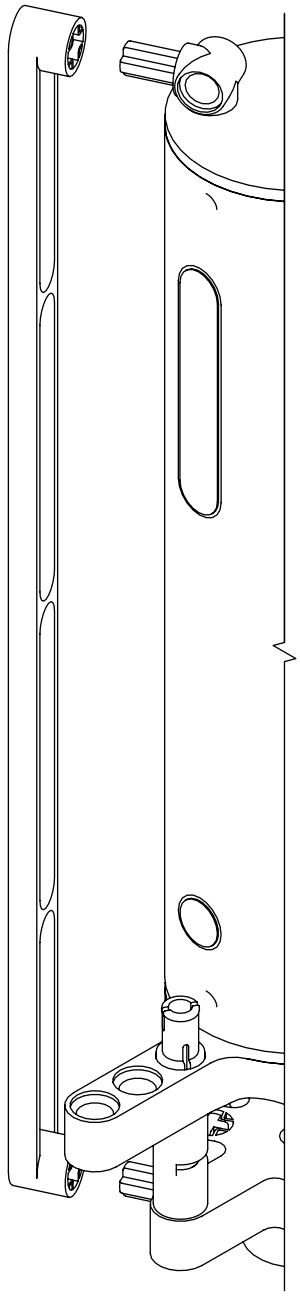


12



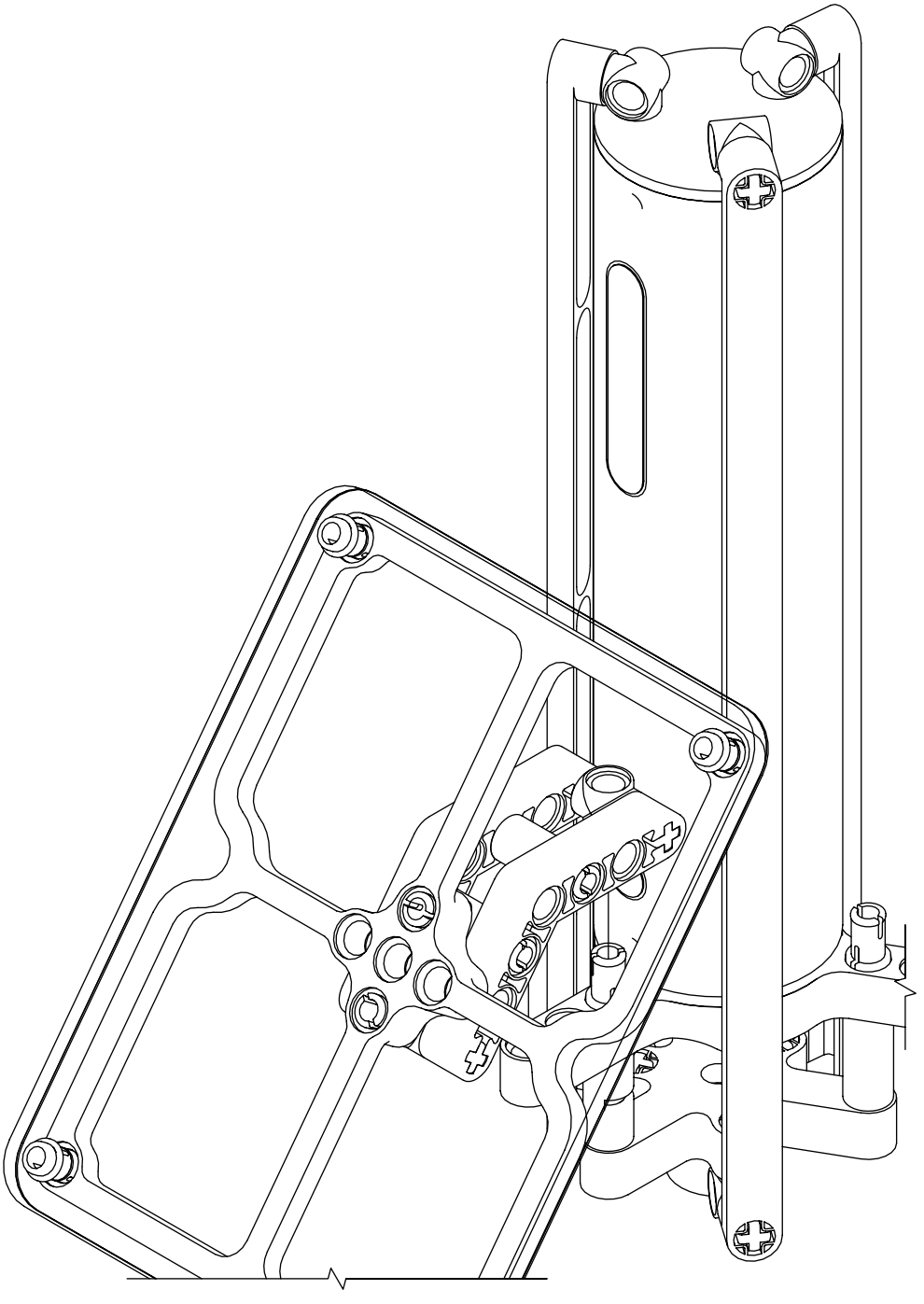
13

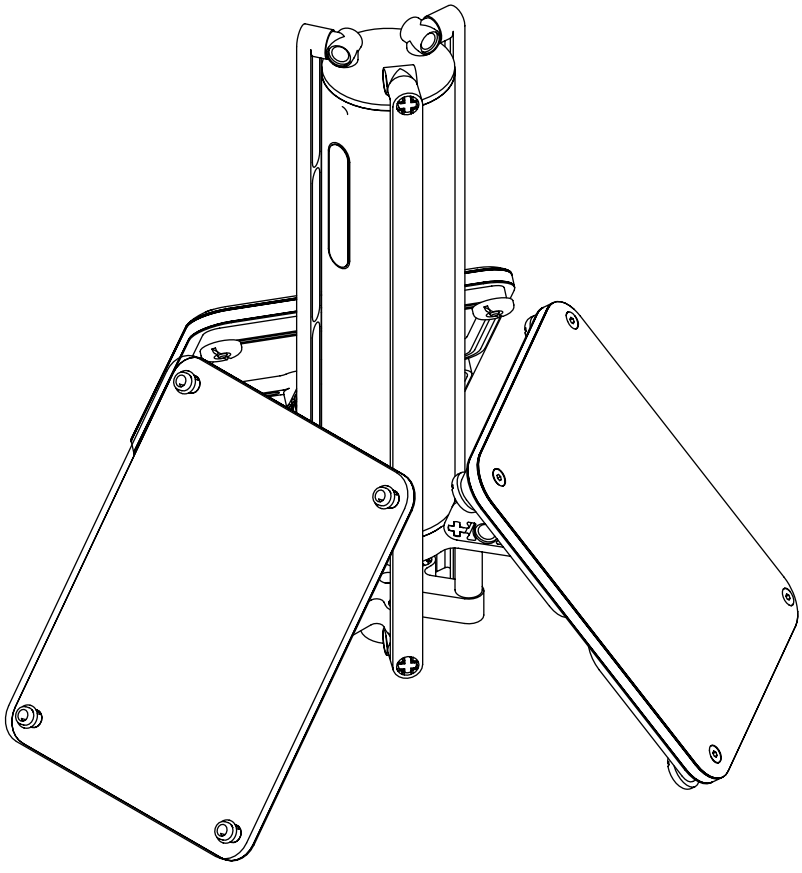




14

(3×) 15





17

Airborne *node* (AIR)

A Assembling the Board

Read the following documents and refer to point A (Standard) in the Developer *node* assembly step until the Board is done.

- WisBlock Quick Start Guide[A]
- RAK19003 Quick Start Guide[B]
- RAK4631 Quick Start Guide[C]

A B C

After having assembled one board, it is recommended to do the same for all the other *nodes* in your network and jump to the Flashing and Configuring steps of the deployment phase.

B Assembling the Radio Module

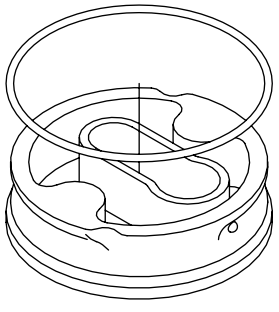
Refer to point B in the Nomad *node* assembly step until the Radio Module (External Antenna) is done. Then, refer to point B in the Developer *node* one (Standard).

C Assembling the Power Module

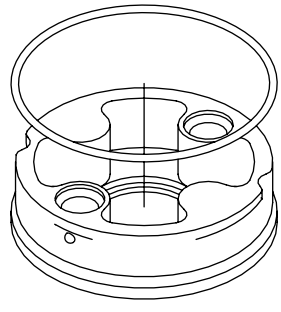
Refer to point C in the Nomad *node* assembly step until the Power Module is done.

1/1

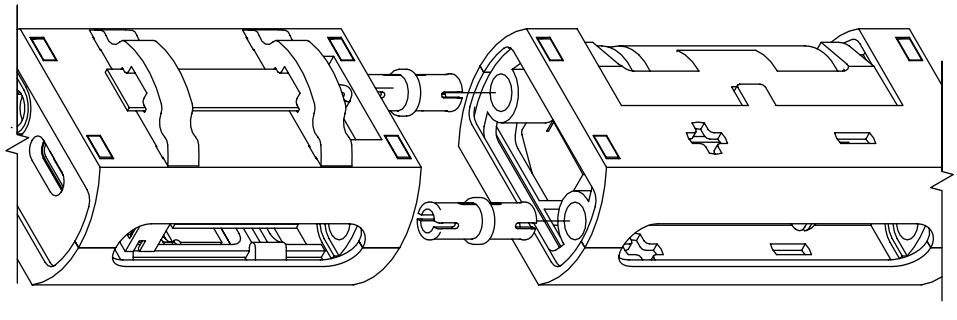
D Assembling the Cover



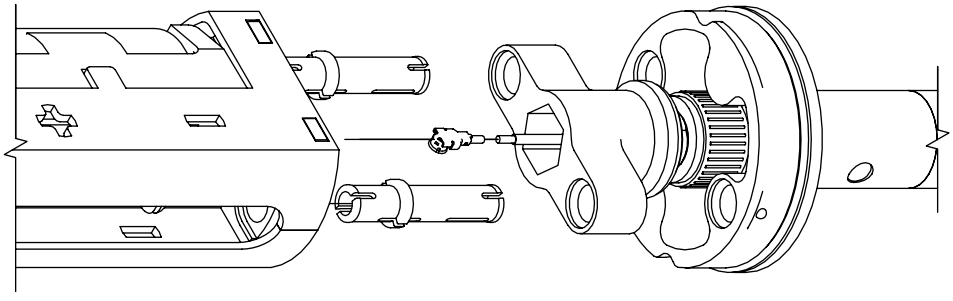
1



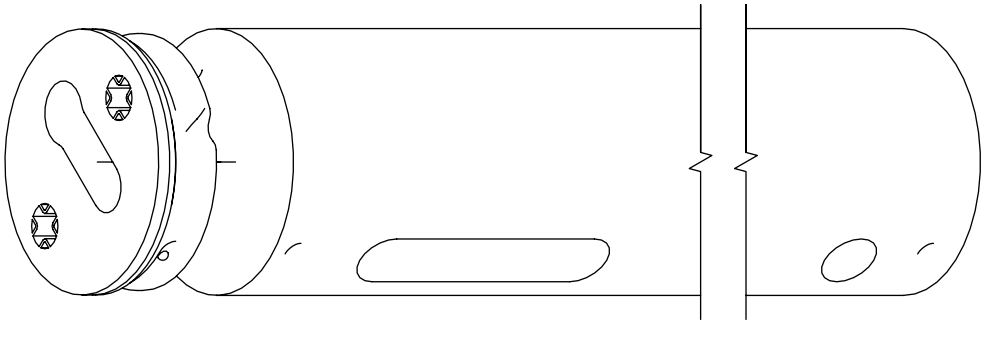
2



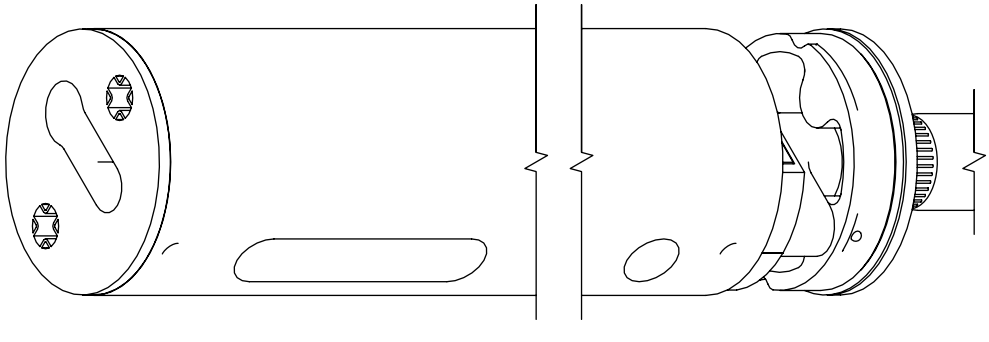
3



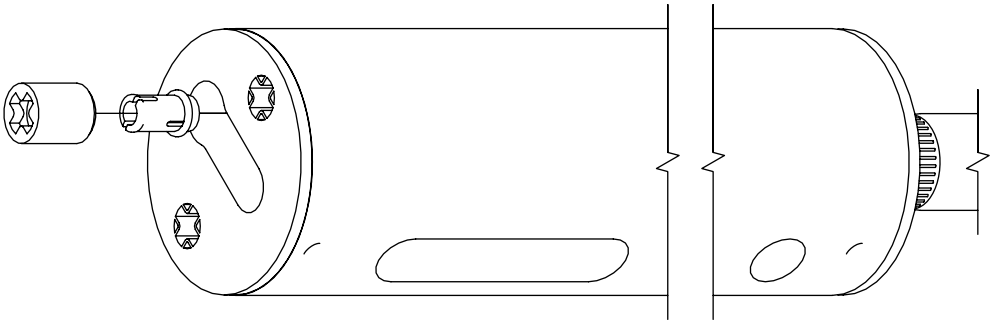
4



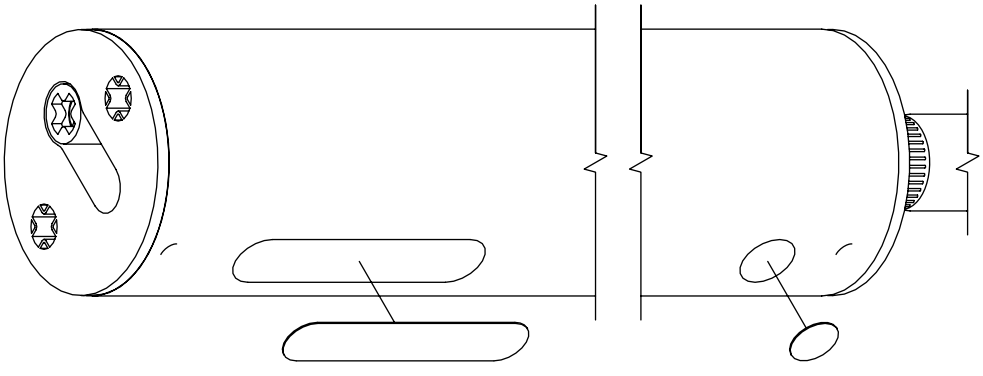
5



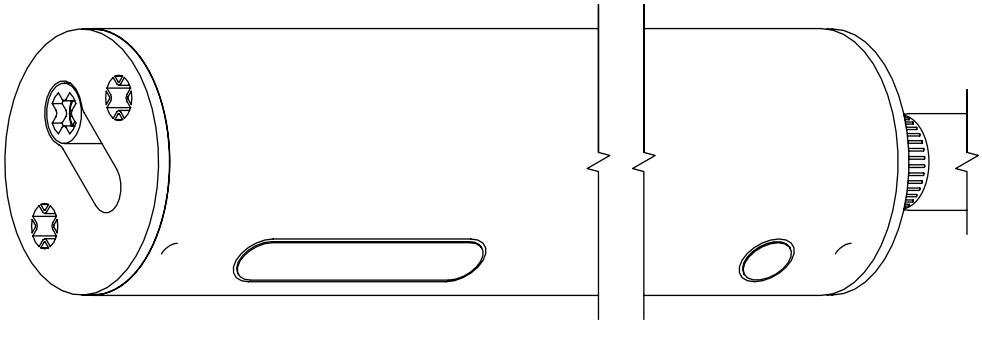
6



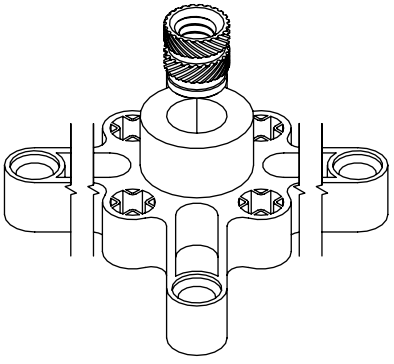
7



8



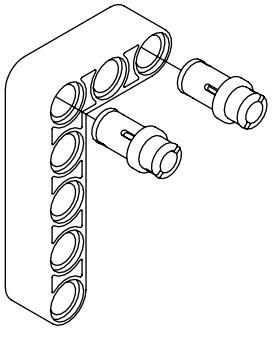
9



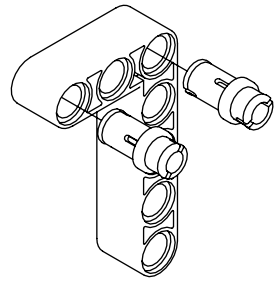
Use the soldering iron to press the Threaded Inserts in. In doing so, keep the soldering iron vertical and the Threaded_Base stable on a flat surface.

When the tip hits the surface (through the bottom hole) the Threaded Insert should be fully seated.

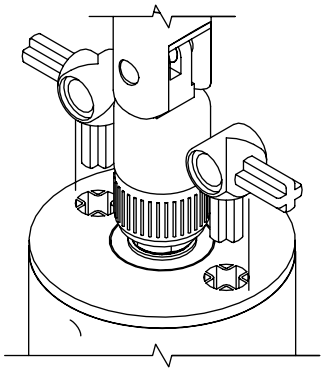
1



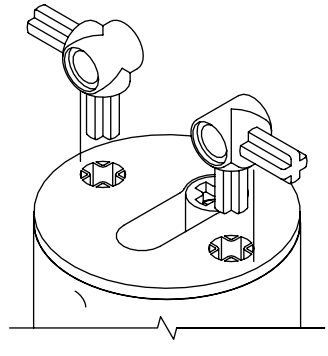
(2×) 2



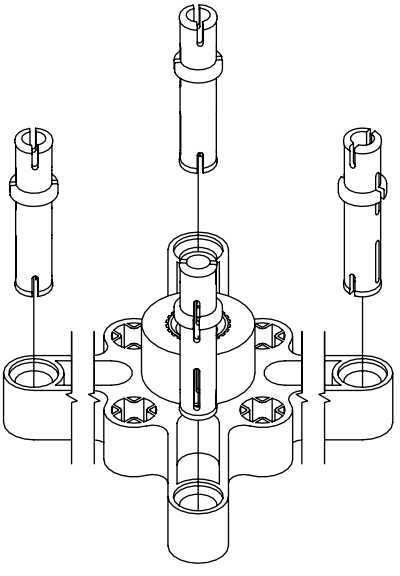
(2×) 3



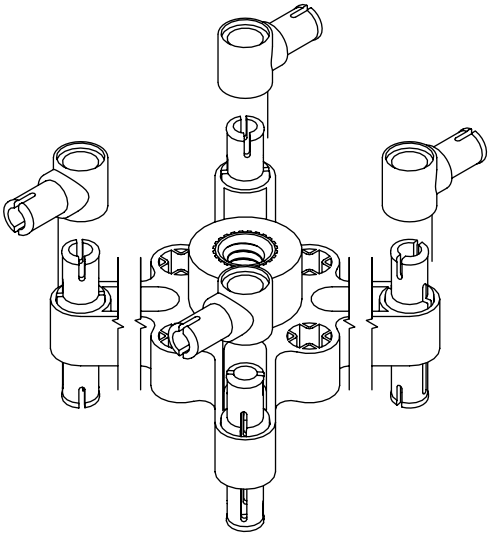
4



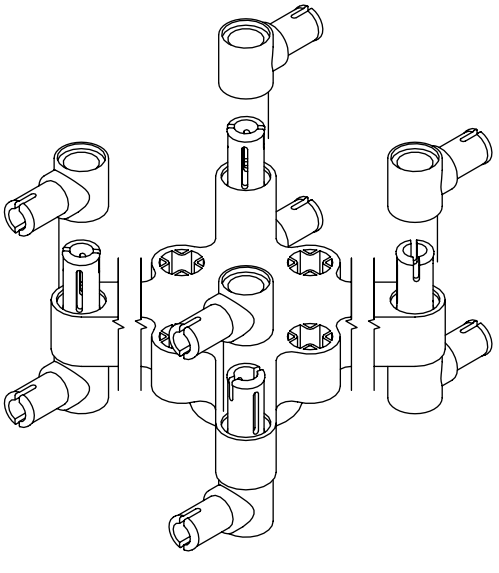
5



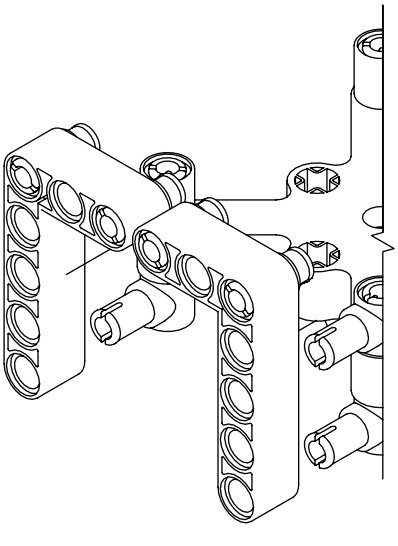
6



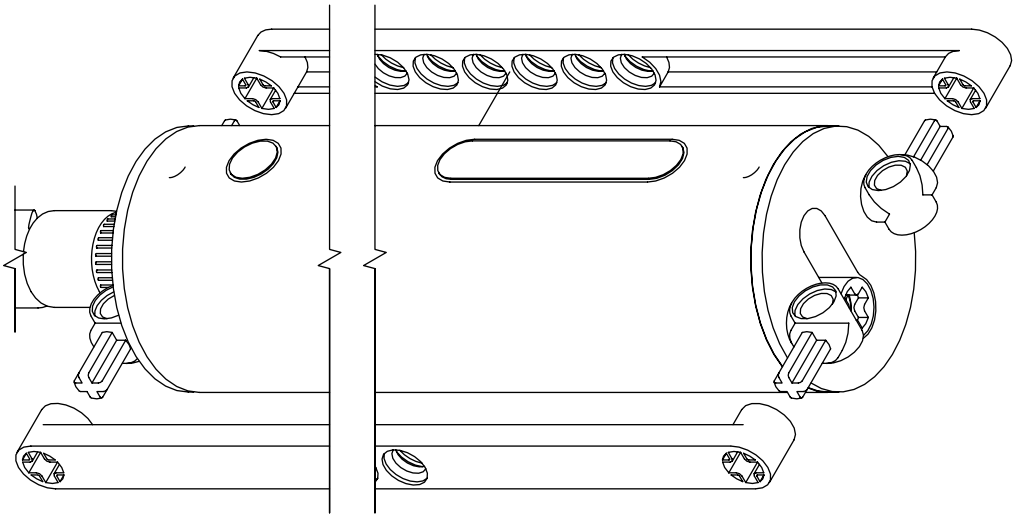
7



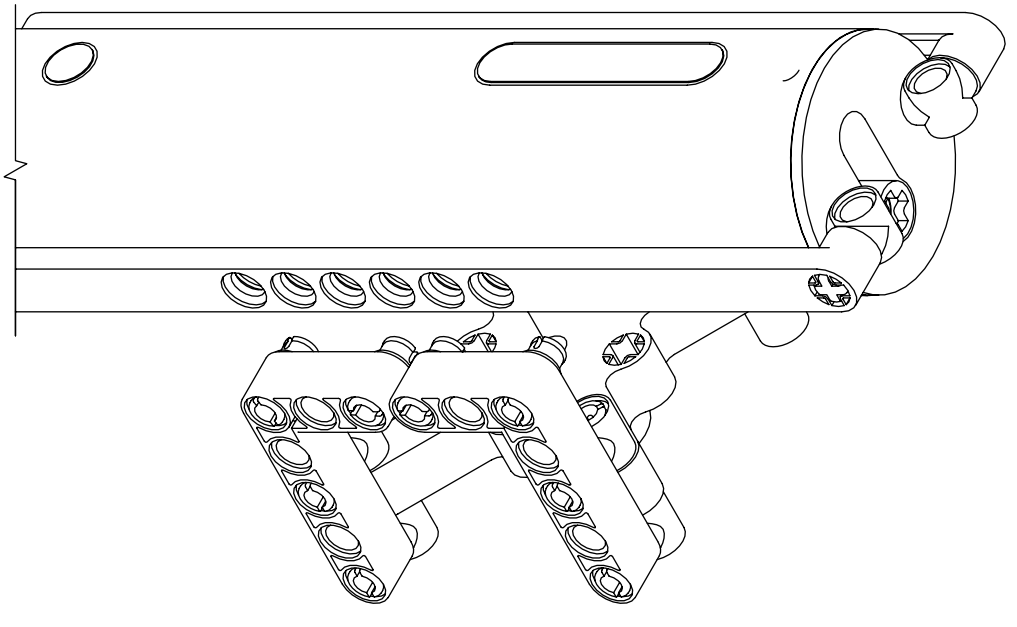
8



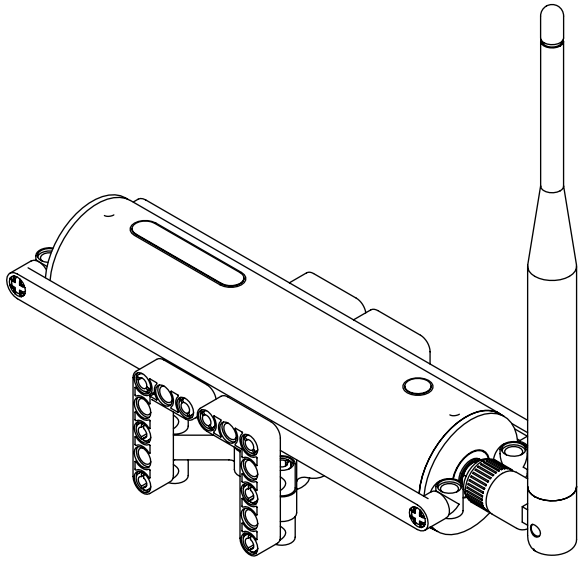
(2×) 9



10



11



12

2.4.3 **Deploying the network**

In this phase—we understand how *nodes* can be used and apply what we learned by flashing them with the required firmware, interfacing them with your smartphone, tablet or computer and deploying our custom ad-hoc wireless meshnet.

19

To enable P2P messaging with the LoRa radio transmission technique, the Tendril Repository Networks, rely on Meshtastic[19], an open-source upper-layer solution that comprises *node* firmware, back-end programming software and even a family of front-end companion apps for iOS & macOS, Android and a web-based one for Chromium browsers.

This ever-evolving ecosystem of resources offers an extensively documented upper-layer implementation process with plenty of community support and recurring updates.

Employing Meshtastic offers a welcoming introduction to online *maker* communities—comprehensive learning environments that function as both nurturing havens for beginners and electrifying launchpads for experts—nests in which "makers" can safely exercise self-production before setting out on their own. So, before starting with the deployment phase, feel free to join the

20

Meshtastic Discord[20] to meet the community and keep tabs on any development progress coming both from the Meshtastic ecosystem and LoRa's at large.

The network deployment phase is divided into three steps: Flashing, Configuring and Operating.

Flashing

In this step—we flash all *nodes* in our custom meshnet architecture with the Meshtastic firmware.

Please note that:

You can refer to the Meshtastic Docs[21] for any further guidance throughout the flashing process.

21

The devices are going to be flashed following Meshtastic's recommended "drag & drop" method described in the aforementioned docs.

If you have erroneously sourced a RAK4631-R instead of a normal one, refer to this guide[22] to convert it to the required Arduino version.

22

To reset to factory settings, drag and drop the Meshtastic factory reset UFL file from the latest firmware folder into the RAK4631 drive after having put the device in bootloader mode.

1. Check to see if the board is assembled properly (RAK19003 Mini Base Board + RAK4631 LPWAN Module with both LoRa and BLE antennas firmly plugged in.

   If you have a RAK12500 GNSS GPS Location Module installed on your Base Board make sure the GPS antenna is firmly plugged in too.
2. Download and unzip the latest firmware folder from the Meshtastic Downloads[23] page.
3. Plug the *node* with the provided USB-C cable into your computer.
4. Double-click the button to put the device into bootloader mode.
5. Notice a new drive will be mounted on your computer (Windows, Mac, or Linux).
6. Open this drive and you should see three files: `CURRENT.UF2`, `INDEX.HTM`, and `INFO_UF2.TXT`.
7. Drop the appropriate firmware file (`firmware-DEVICE_NAME-vx-.x.x-xxxxxxx.uf2`) from the release onto this drive.
8. Once the file has finished copying onto the drive, the device will reboot and install the Meshtastic firmware.

23

Configuring

In this step—we configure all *nodes* in our custom meshnet architecture through Meshtastic's command line interface to function together on the same network.

Please note that:

ou can refer to the Meshtastic Docs[24] for any further guidance throughout the configuring process.

The devices are going to be configured with Meshtastic's Python CLI interface as described in the aforementioned docs.

This step details configuring on a Windows machine using a standalone executable, refer to this guide[25] if you have a different machine and/or wish to proceed differently.

24

25

26

1. Download the latest Meshtastic standalone executable from the releases GitHub page[26] by clicking on "`assets`" and then "`meshtastic_windows`".
2. Rename "`meshtastic_windows`" to "`meshtastic.exe`"
3. Open the Command Prompt or another CLI of your choice and navigate to the directory in which the meshtastic.exe executable is located. (e.g. if the executable is on the desktop, run "`cd desktop`" so that the path changes to "`C:\Users\username\Desktop>`")
4. Run:

   ```
   meshtastic.exe
   ```

   and check for an affirmative return message.
5. Open the Device Manager and, under Ports (COM & LPT), check for the port your radio is connected to.
6. In the CLI, connect to the radio by running:

   ```
   meshtastic --port COM3
   ```

   (COM3 is an example, check your own in the Device Manager) and check for a "`connected to radio`" return message.
7. Run:

   ```
   meshtastic --export-config > example_config.yaml
   ```

   and export a configuration file.
8. Duplicate the configuration file for all the *nodes* in your network (so that the "`channel_url`" stays the same across all nodes) and, if you want, rename all the resulting files to a name of your liking (e.g. `nomad1.yaml`, `nomad2.yaml`, `resident1.yaml`, etc.).
9. One by one open the configuration files with a software of your choice and change the "`owner`" and "`owner_short`" of your *node* to a name of your liking and then proceed to save the file.
10. Once you are done, in the CLI, run:

    ```
    meshtastic --configure example_config.yaml
    ```

    to load the configuration file of your choice into the *node* that is currently plugged in and check for confirmation.
11. To make sure everything went as planned run:

    ```
    meshtastic --port COM3 –info
    ```

    (COM3 is an example, check your own in the Device Manager) and check for the returning report.
12. To see all Meshtatic CLI commands, run "`meshtastic –help`".

Operating

In this step—we operate our custom meshnet architecture through Meshtastic's front-end applications.

Please note that:

You can refer to the Meshtastic Docs[27] for any further guidance throughout the operating process.

This step details operating on an iOS device, some descriptions may vary for other companion apps.

27

28

1. Download the appropriate Meshtastic application for the device you wish to interface with the network from the Meshtastic Downloads[28] page.
2. Launch the Meshtastic application.
3. Make sure you are in Bluetooth range from a powered *node*.
4. Navigate to the Bluetooth page to pair with it.
5. If it is your first pairing enter "123456" when asked for a pairing code. Also if it is your first pairing, specify your region-specific frequency band and save (make sure all meshnet *nodes* share the same one), pair again if needed and you are set to go.

- On the Messages screen, you can group-chat with all network clients as well as message them directly by selecting the *node* to which they are connected.
- On the Bluetooth screen, you can pair with all *nodes* within the Bluetooth range of your device.
- On the Nodes screen, you can see all the *nodes* belonging to the meshnet, allowing you to keep tabs on their activities, metrics and battery life.
- On the Mesh Map screen, you can see the latest GPS position of all the *nodes* belonging to the meshnet. This feature, if enabled (Go to Settings > App Settings > Phone GPS and turn on "Provide location to mesh"), allows your device (e.g. phone, tablet, computer, etc.) to regularly transmit its GPS position (also check the App Settings to establish how often) across the meshnet to all other network clients. If the *node* equips a RAK12500 GNSS GPS Location Module, the GPS position will.

FRANCESCO ZURLO FEDERICO BASSI ALESSANDRO IANNIELLO