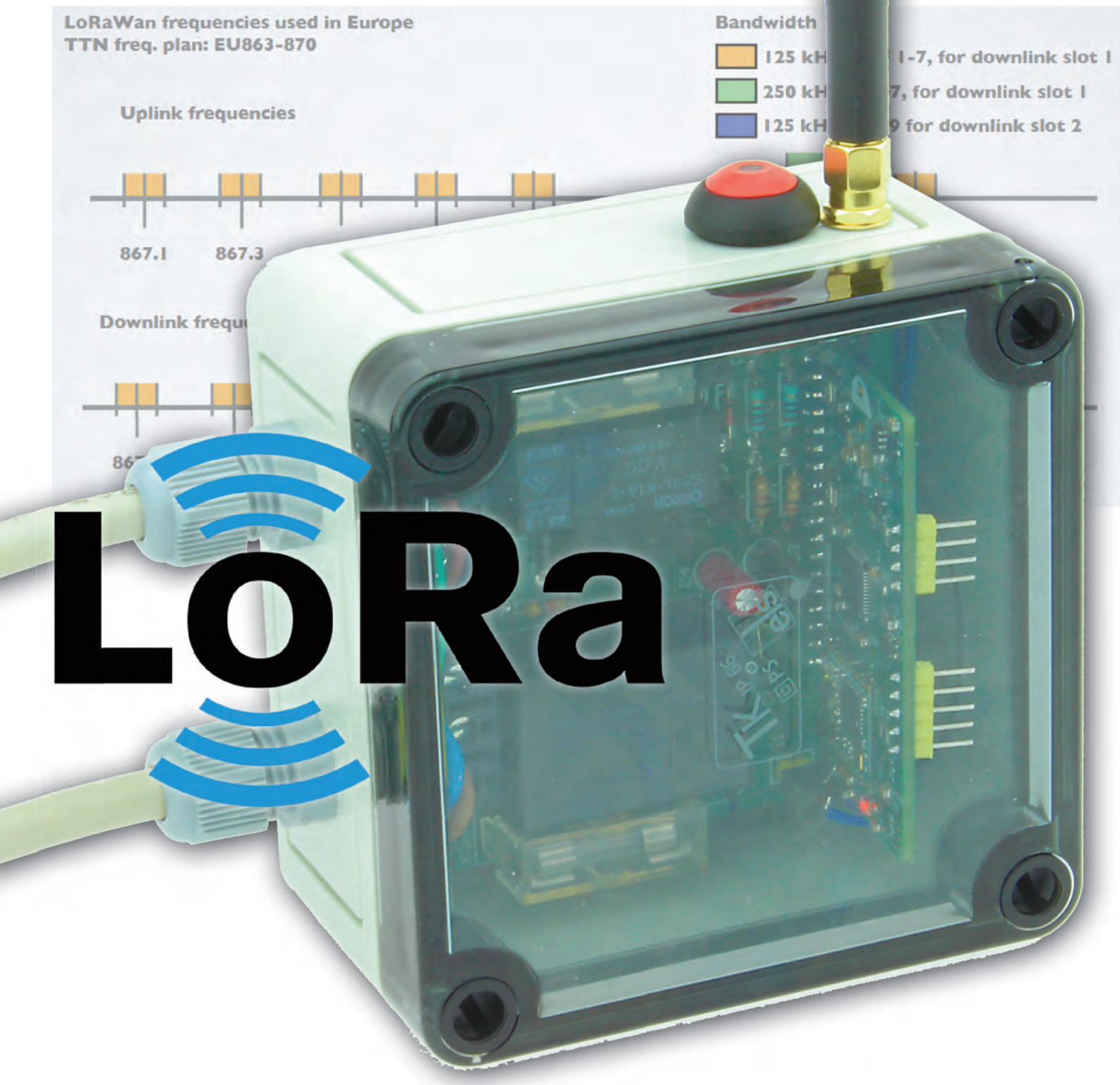
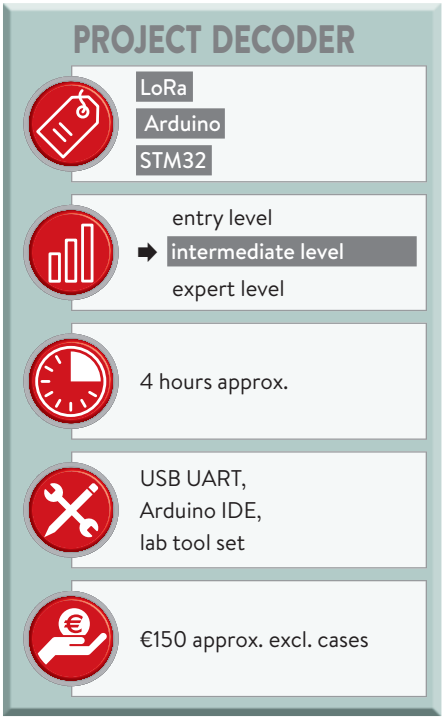
eLABS PROJECT

The Elektor LoRa Node

Versatile, long-range, 868-MHz remote control with state feedback and STM32 Inside





By **Mathias Claußen** and **Luc Lemmens** (Elektor Labs)

Convinced of the combined power of open-source hardware and software, Elektor smashes the myth that LoRa is for pros only. This article not only tells the story of persistence in electronics design in the face of real challenges, but also of a 3-element LoRa Node you can easily replicate for reliable on/off control with status feedback and covering distances 10 to 20 times those of consumer-level WiFi and 433-MHz ISM LPR.

The Elektor LoRa Node was born from the idea to have a compact PCB with support for rechargeable as well as non-rechargeable batteries, a common microcontroller like the LQFP48-cased STM32, and a type RFM95 LoRa module, also, the PCB was to fit in an enclosure that's easily sourced from one of the bigger distributors, also, for the STM32 compiler, we… **STOP!**

As the enclosure will co-determine the space available or the energy source, we need to discuss this first. The initial idea was to power the device from non-rechargeable batteries. This was rejected since Elektor wants its contribution to battery waste to be as small as possible. So, a rechargeable battery solution was sought, as well as ease of replacement by the user. This would result in a small, serviceable, water-resistant node capable of operating not only in Elektor Labs' comfy conditions, but also "out there in the wild" and conveying real-life environmental data. Further design aspects heavily debated during the project development period were the battery lifetime and the sensors to accommodate on or off the board. Clearly, we aimed to have a flexible and useful platform to explore LoRa and potentially LoRaWAN, not for 'academic' use in the airco'd and carpeted lab but ready to use in any (rough) place out there for the purpose of remote device on/off switching. From here, the challenge is the 'squaring of the circle' in terms of flexibility, power consumption, and size.

Read here how we did it. The story that follows is kind of chronological, since with this project we like to share not just the final product and a quality write-up, but also the real-life problems we ran into, and the tools used to develop the project. We do so at the request of many readers.

What's being described

In this article we cover the three elements mentioned above:

- **LoRa Node**. Board no. 180516-1, the LoRaWAN Node Experimental Platform, stuffed for 'local' LoRa only.
- **LoRa Button**. Board no. 180666-2. Two required for this project.
- **LoRa AC Power Switch**. Board no. 180666-1.

Together the three elements form not only a remote control with a range of 10 to 20 times that of a consumer-level WiFi link on 2.4 GHz, but also a LoRa development platform with potential for LoRaWAN. We kick off with the main

Quick Features

Lora Node

- LoRaWAN Experimental Platform board with minimum parts stuffed
- Li-Ion battery powered
- User changeable cells
- STM32F072C8T6TR ARM Cortex-M0 MCU
- SPI F-RAM or Flash (optional)
- Crypto co-processor (optional)
- GPS module (optional)
- USB interface (optional)

LoRa AC Switch (slave device)

- Relay state feedback to Node central
- Switch contacts rated 5 A (1000 W at 230 VAC; (500 W at 115 VAC)

LoRa Button (master device)

- Low-energy design
- Optional OLED display
- Integral helical coil antenna
- 100–500 m range to LoRA slave

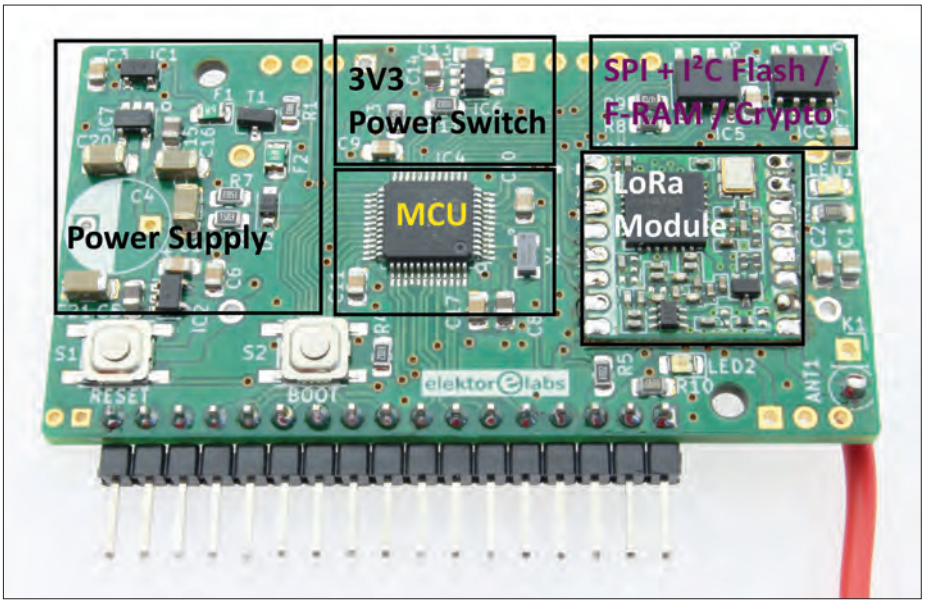


Figure 1. The LoRa Node control board with 'real estate' allocated to functional blocks. Actually, you are looking at a minimally-stuffed LoRaWAN Node Experimental Platform.

element: the LoRa Node.

The early beginnings

The PCB was designed using KiCad, a free, open source, multi-platform, electronics design tool that's gaining users every day, and not just hobbyists. If you'd like to start out with KiCad we have a fine book available in the Elektor Store [1].

After a false start with "another" processor, and some more stumbling in the dark, a board for the STM32F072 in LQFP48 case was designed, benefiting from that MCU's Arduino IDE compatibility. This makes development pretty easy as you can now use most of your well-known Arduino libraries. As we didn't have any incarnation of a quartz crystal on the board we resorted to the internal oscillator options. Here, this defaulted to 8 MHz for the MCU in order to reduce the power consumption, but ready for cranking up to 48 MHz if desired.

The modular approach

The Elektor LoRa Node board pictured in **Figure 1** (early version) is not only the basis for the present project but will be used in spin-off projects to be published later this year in *Elektor magazine*. That's why we will first take a tour of the components and other options that may be assembled on the board. The hard-core electronics fans among you may want to do a parallel tour by way of the circuit diagram printed in **Figure 2**.

To meet certain design aspects and requirements we stuck to thinking 'modular'. With the full complement of components fitted you'd have a LoRaWAN device with a GPS module, crypto coprocessor, SPI F-RAM or Flash storage. All features are "nice to have" but you may not need them. It also implies that components not populated on the board will save money in the basic version, and no need to pay for features you'll never need. As the pinout for the SPI F-RAM and SPI Flash modules is standardized, you can add your favourite module. The same for the crypto coprocessor, an ATECC608a: you can also choose to just add an I²C-EEPROM or an F-RAM chip, meaning you have a very flexible and use-configurable device available.

The LQFP48-cased STM32F072 MCU now on the PCB is a Cortex-M0 based device with 64 kB of Flash and 16 kB of RAM, allowing you to even have USB for firmware updates or for use in your own applications.

The nice thing about the STM32 series is the utter pin compatibility, meaning you can change the MCU without changing the board layout. For example, the even lower power STM32L072, or an STM32L151 in case you need more computation power and Flash. The STM32F103 you commonly find on the 'Bluepill' board can be fitted on the PCB, too.

The **Quick Features** box is just an attempt to show the versatility of the board.

PCB design and challenges

After building the first prototypes some adjustments seemed in order. For the GPS module we needed some tweaks to get better reception. For sure, you can build a miniaturized antenna but you need to take the ground plane into account. The GPS module datasheet states a minimum PCB size for the antenna to work. Still, we were not satisfied with the signal reception.

After Version 1 of the board we ditched the Quectel L96 GPS module and replaced it with the GPS Module from the Elektor Store [2] as this was both cheaper and, easier to connect to the board as a quasi-external part. This caused further redesigns for the connectors used, so everything is now in a 0.1-inch raster allowing you to plug the module onto a breadboard.

Then we have the RFM95W LoRa module that needs an antenna for sure. With the size and enclosure we selected, we wanted to use a DIY helical coil antenna for 868 MHz. If you desire even better reception, an external antenna is an option. We realized the design was a compromise between space, functionality and RF performance. Lessons learned.

We already mentioned the use of Li-Ion batteries in 10440 (AAA) format, as found in e-cigarettes. These are easy to replace by the user and charged outside the device with a proper Li-Ion charger. To protect the cells from deep discharge a voltage monitor IC was included switching off the LDO for the 3.3-V supply.

Also added at this point was a 2-mm pinhole part to attach LiPo battery packs commonly used for drones. These small and flat cell packs come with a Molex 510005 or JST connector and offer a fine capacity/space ratio. This makes the whole design more flexible and allows the use of off-the-shelf rechargeable batteries that can be sourced from big warehouses starting with an A.

Reverse-current protection was implemented using a 'loadswitch' device. Eventually we used the MAX40200 'ideal diode' for this purpose, as it will disconnect the supply voltage if it passes a certain amount of reverse current. A second one was added for the power supply towards the GPS module (K4), enabling the MCU to cut the power to the module in order to extend battery life. If

you forfeit the GPS module you can have the MCU control that 3.3-V rail to power or disable other external circuitry.

Powering the device from NiMH cells would be feasible but needs additional circuitry, in this case a DC/DC boost stage. As the boost stage will consume power all the time this may reduce runtime, and we also need to reconsider the under-voltage lockout.

With our home-made antenna transmitting fine, the biggest challenges at this

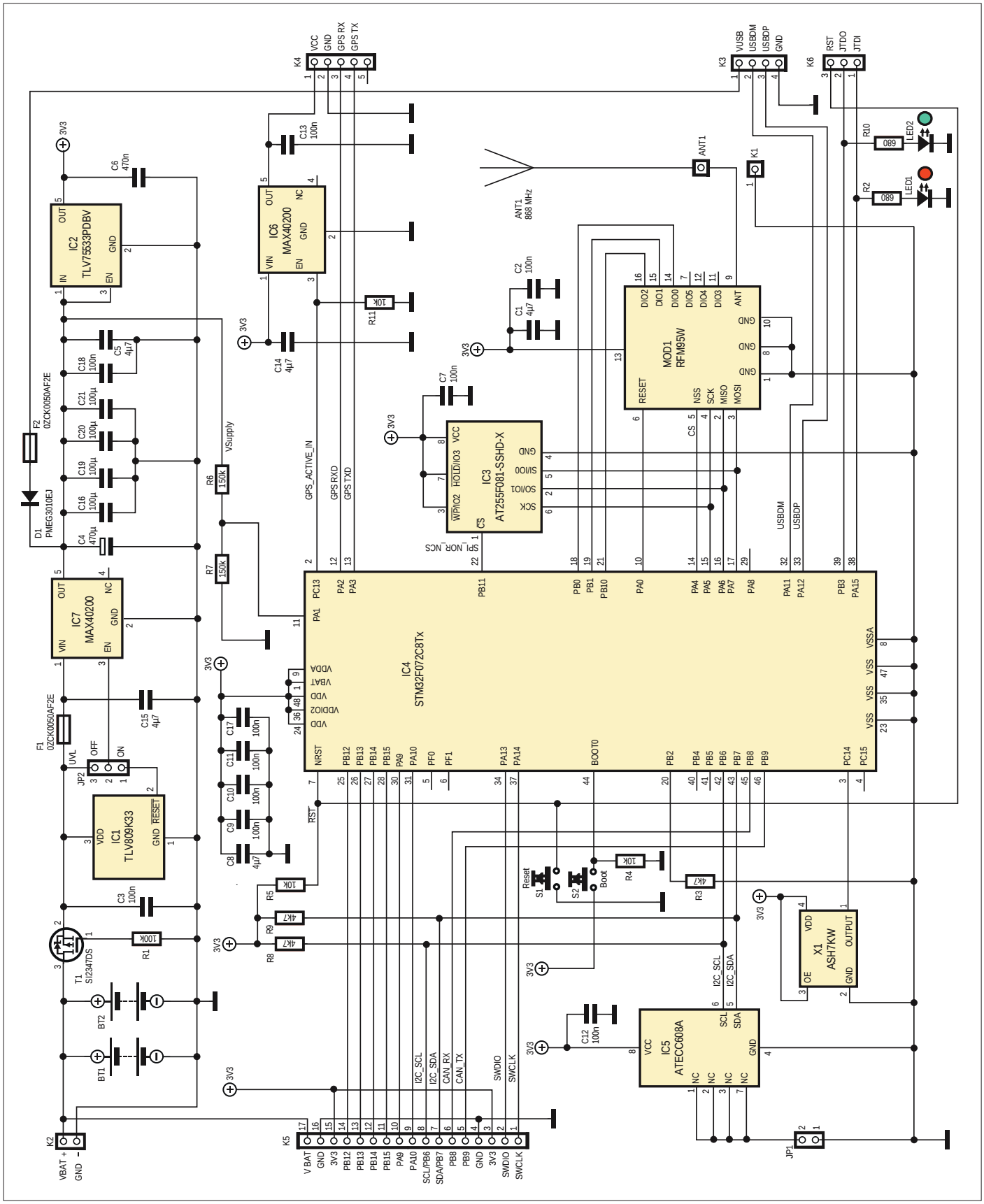


Figure 2. Circuit diagram of the LoRa Node control board. This is basically the LoRaWAN Node Experimental Platform board with a specific outfit of ICs.

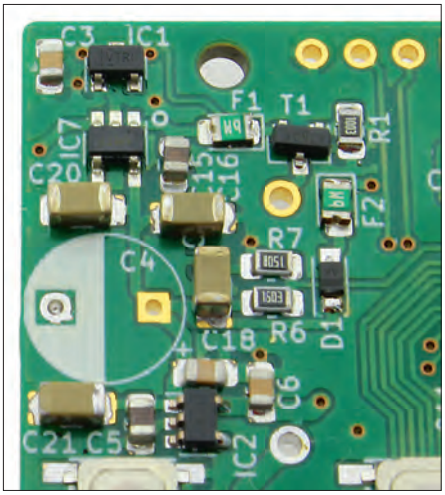


Figure 3. Board space reserved for electrolytic capacitor C4, one of the largest parts on the board.

Figure 5. A special method for mounting the LEDs is applied on the LoRa Button board.

point appeared to be within the project software. Undeterred though we ran point-to-point communication in the RAW mode offered by the LMIC library.

Down at the component level a few problems arose, including one with a 470-µF capacitor previously ahead of the undervoltage lockout device, IC1. If the input is below 2.97 V, it disables the HS switch IC7 and the supply voltage is cut as well for a period of 20 ms.

Without a large buffer capacitor, high currents at start-up upset the UVLO and its input voltage will drop below 2.97 V, resulting in a permanent loop of enabling and disabling the LDO. Adding a 'big' electrolytic like C4 (**Figure 3**) made the power path work as expected. Fit it only if you find the two 100-µF MLCC solid caps too expensive. Also according to the datasheet, we don't have a FET or diode protecting the batteries from being charged from the V_{USB} side. This job is done within the MAX40200 — if a reverse current flows the chip will cut power. This is also true if the voltage coming from V_{USB} exceeds that provided by the system batteries.

The two LEDs on the board flag the board status reported by the MCU on port lines PB3 and PA15 a.k.a. 'JTDO' and 'JTDI' on pinheader K6.

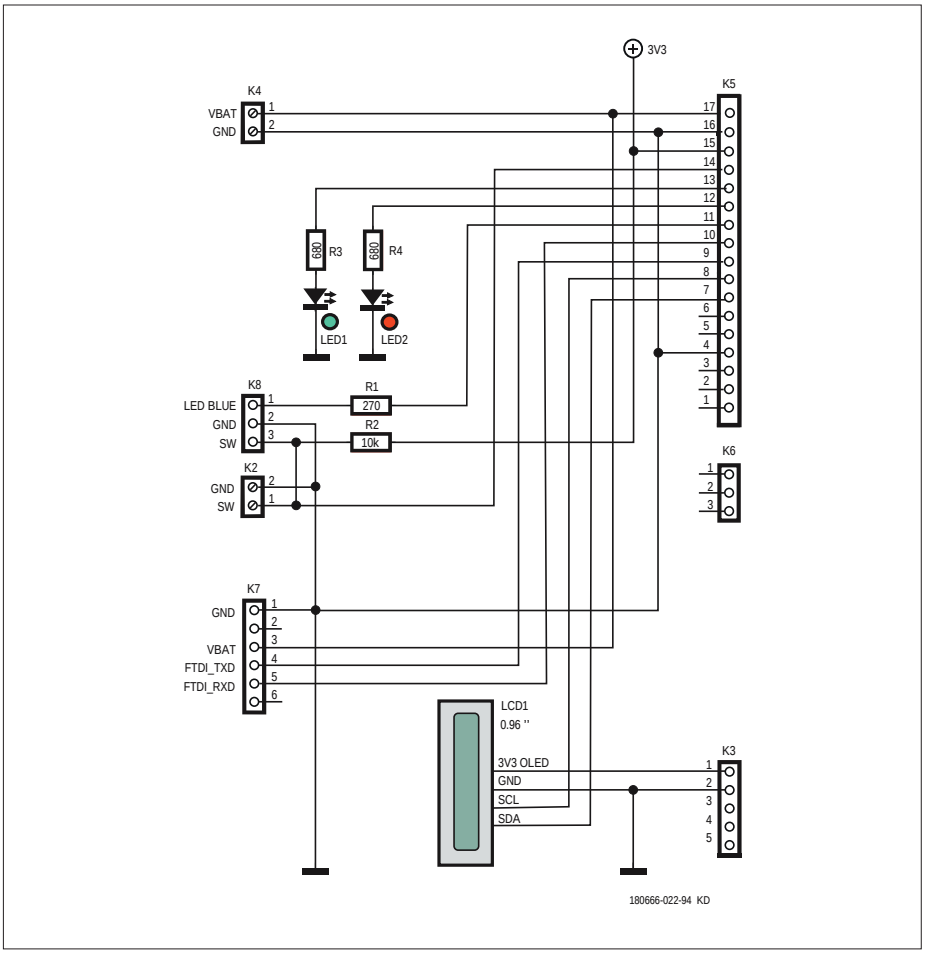


Figure 4. LoRa Button schematic. This is the 'sending' or 'master' part of the Elektor LoRa Node. The OLED display is optional and its use requires deep thought about energy use!

Software: the groundwork

Although system software increasingly defines hardware functionality, to some it's just black magic that needs to be 'flashed' into a chip. A comparison of lab time spent on software *versus* hardware shows many more hours dedicated to the former. How come?

As already mentioned, you can program the board with your favourite Arduino IDE. For this the *stm32duino* project [3] did especially well in supporting a wide range of STM32 MCUs and STM32 boards. Formally adding the Elektor LoRa Node Experimental Platform was the first 'clerical' task that needed to be done. In the Arduino core for the STM32 we had to edit a few files and add the board definition. After some errors and unexpected software behaviour we got the template running and were able to start coding the first few lines.

One thing you normally don't care much about is the way the MCU generates the clock. On the older AVR chips you had a crystal and that was it. On the more recent MCUs though you have more options to generate the MCU main clock, often from internal PLL / FLL sources or high-speed internal RC oscillators. We decided to use the internal clock sources. They're not great for frequency stability, but good enough to get us going. By allowing us to change the clock speed during runtime, going 'slow' saves power and going 'fast' yields higher throughput. The default value of 8 MHz for the core

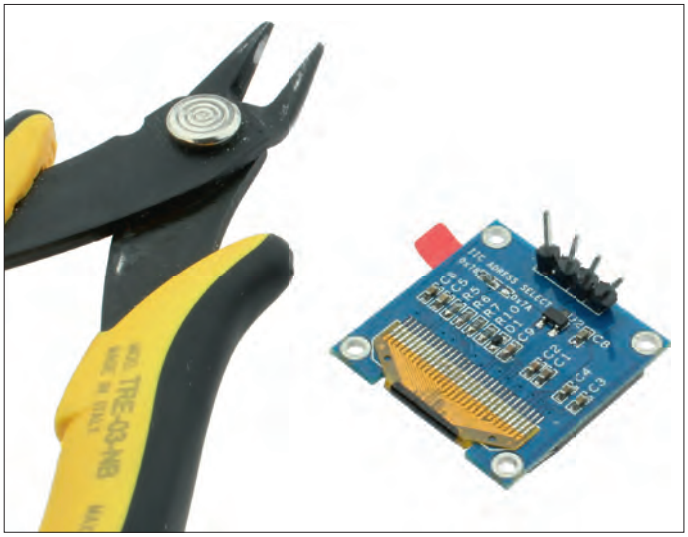


Figure 6. Use sharp cutters to snap into the plastic material between the pins until they are no longer held together.

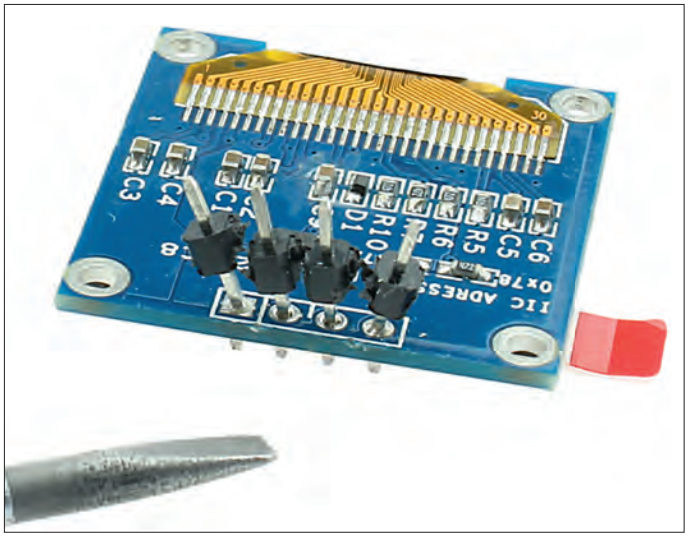


Figure 7. Using a flat screwdriver blade, carefully lever up the plastic remnants surrounding the pins, and remove them.

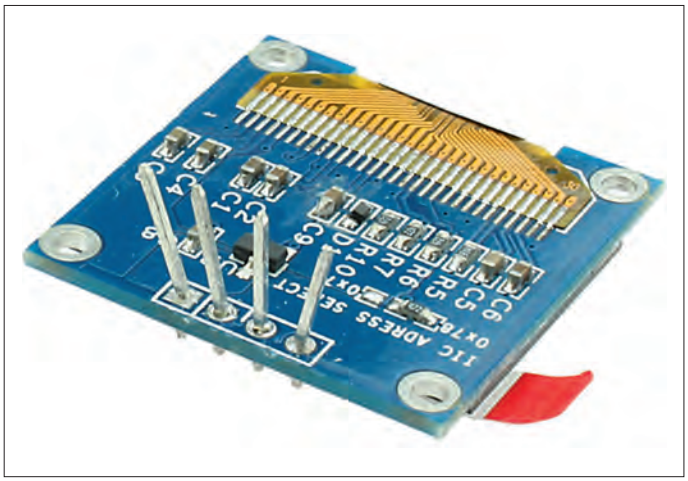


Figure 8. You now have a 'naked' pin row.

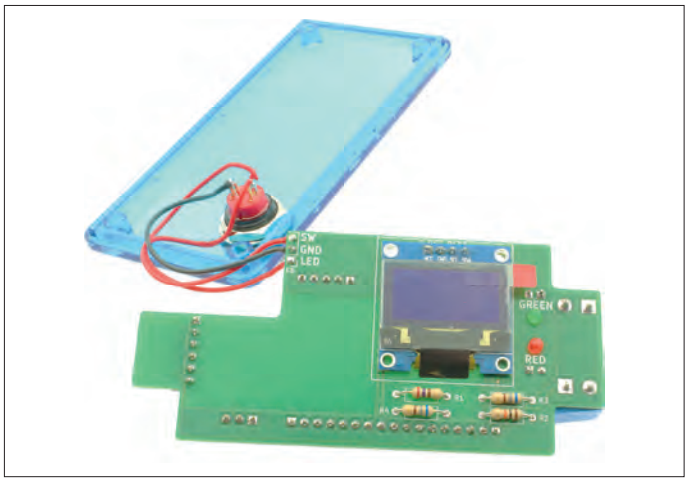


Figure 9. The almost completed LoRa Button board.

and peripherals keeps power consumption modest yet is fast enough to process the LoRa stack.
For accuracy in the RTC domain a 32.768-kHz oscillator was added (X1). It is optional. At the cost of a µA of current we gain UARTs that can be clocked from the crystal as well as an RTC that can wake up the MCU after a given timeframe.

The LoRa Button

For a simple input device to convey its state over radio and learn about things we propose the second component in the project, the 'LoRa Button'.
The schematic in **Figure 4** shows that the LoRa Button has just a few resistors, a pair of LEDs, PCB screw terminal blocks, pinsockets, and pinheaders. These parts are essential for the basic version of the Elektor LoRa Node. The OLED display from the Elektor Store [3] (component: LCD1) is optional.
When assembling the board, there are a few things that need to be considered. The LEDs, for example, have their bodies bent over and inserted in PCB holes so their faces are at the board underside, see **Figure 5**.
For the OLED display more delicacy is in order because you either have to unsolder the entire pinheader block or bodge a bit by removing the plastic shrouding around the pins in the pinheader. To remove the excess plastic, first bite into the plastic part between the pins (**Figure 6**), then slowly lever the plastic upward over the pins using a screwdriver blade (**Figure 7**). If everything goes well, the metal pins will be exposed (**Figure 8**).
The OLED display can then be inserted flat into the intended position, followed by the pinheader sockets, the angled pinheaders, and the PCB screw terminal blocks. This almost completes the construction, less the real button of course (**Figure 9**).

Interlude: a DIY 868-MHz antenna

The LoRa Button now needs an antenna for transmitting and receiving at 868 MHz. A quarter-wave 'Marconi'

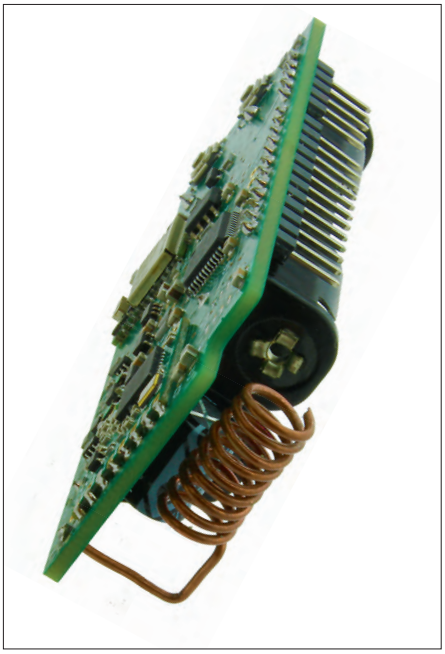


Figure 10. The feedpoint of the home-wound, quarter-wave antenna for the 868-MHz band is soldered to the ANT pad on the LoRa Button board (early prototype shown). Note the coiled construction of the antenna.

antenna made from 1-mm diameter enamelled copper wire (AWG #18) is a good and low-cost option. Assuming a ground plane is present the wire should have a length of 8.635 cm in a vacuum but 8.2 cm in practice! That's because of the *air velocity factor* of 0.95 [4]. Depending on the space available, the antenna wire may be left straight to equal a rod antenna or wound into a coil and compressed with 1-mm spaced turns. Either way, the wire is soldered to the ANT connector **(Figure 10).**

The LoRa AC Switch Slave

As an example of a slaved "output device'' controlled by the Elektor LoRa Node as well as 'listening' to the LoRa Button, a power switch was developed for mains operated equipment consuming up to 1 kilowatt (kW). An optocoupler returns the (on/off) state of the AC Switch Slave to the Lora Node. The board also houses a 100-230 VAC to 5 VDC power converter allowing the Lora Node and the AC Switch Slave to be powered directly from an AC power outlet.

The real intelligence of the LoRa Switch Slave resides in the LoRa Node hardware and software, which is connected as a plug-in board. None the less, the Switch Slave provides these functions:

- a 5-VDC supply rail;
- load switching dimensioned for 115/230 VAC, 1 kW max.;
- switch state feedback;
- a local control (on/off) pushbutton.

The schematic of the LoRa Switch Slave is given in **Figure 11**. We'll look at the main components and the construction.

Power supply. A 5-V, 2-W, AC/DC converter from Mean Well proved to be a compact, safe and relatively inexpensive solution for the project. Although the documentation with the SMPSU does not mention the need for a fuse at the AC side, we added F1 just to be sure. NTC (negative temperature coefficient) resistor R1 limits the supply inrush current, which can soar in these inverters. C1, C2 and common-mode filter L1 at the output of the AC/DC converter may appear overkill, but we did not rely on Mean Well's statement that the module contains enough internal EMI interference suppression.

Power switching. For this application a device from Omron's G5RL-K1 series of bistable relays was chosen. These have a Set and a Reset coil, and a short excitation of either is enough to toggle the contacts. Although the relay contacts are rated for a maximum current of 16 A, fuse F2 melts at about 5 A to protect the copper tracks on the circuit board from burning out.

Feedback. Theoretically the microcontroller on the LoRa Node board should be able to keep track of the last action sent to the Slave device, i.e. is it on or off? But to be really sure of the condition, a physical feedback of the output is no luxury thing, also because the position of a bistable relay is not certain when the project is reset. Remember: the circuit around optocoupler IC1 only indicates that RE1 is closed and that the mains voltage is present on terminal block K2. Whether the load actually works (i.e. whether it consumes power) is not revealed by the state feedback and should therefore be checked for real and on the spot.

Manual operation. The idea behind this project is to be able to switch a load from a (large) distance by pressing a button. But if you are around anyway at the receiver end it's great to be able to operate it manually without needing the LoRa Button and the LoRa Node. Moreover, if the LoRa connection is lost for whatever reason, it's comfy if you can still switch gears.

Mounting and installation. The soldering of the board has few challenges. Apart from the common-mode filter and (possibly) connector K3, it's all plain sailing with through-hole components only. K3's connector pads are quite large and even L1 is unlikely to throw a spanner in the solder works.

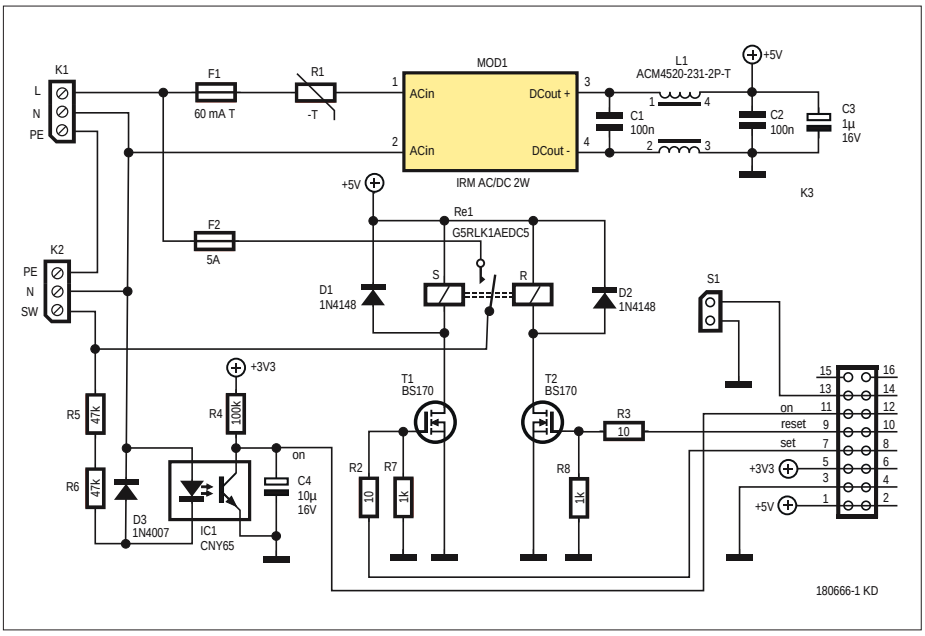


Figure 11. Schematic of the LoRa AC Switch. This is not a 'dumb' device but it feeds the actual relay on/off state back to the LoRa Node.

COMPONENT LIST

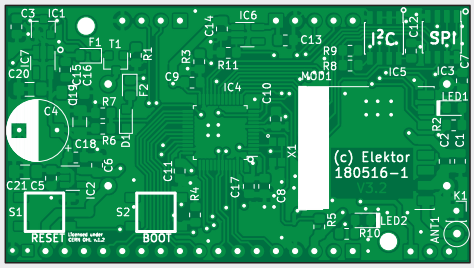
For LoRAWAN Experimental Platform board
2 required, 1 for Switch, 1 for Button

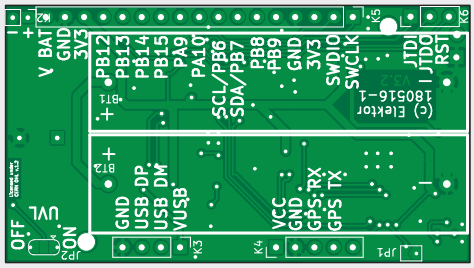
Resistors
R1 = 100kΩ, thick film, 5%, 0.1W, 150V
R2,R10 = 680Ω, thick film, 5%, 0.1W, 150V
R3,R8,R9 = 4.7kΩ, thick film, 5%, 0.1W, 150V
R4,R5,R11 = 10kΩ, thick film, 5%, 0.1W, 150V
R6,R7 = 150kΩ, thick film, 5%, 0.1W, 150V

Capacitors
C1,C5,C8,C14,C15 = 4.7µF, 16V, X7R, SMD 0805
C2,C3,C7,C9,C10,C11,C12,C17,C18 = 100nF, 50V, X7R, SMD 0805
C4 = see text
C6 = 470nF, 25V, X7R, SMD 0805
C16 = not fitted
C20 = not fitted
C19,C21 = 100µF, 10V, SMD 1206 [3216 Metric], ± 20%, X5R

Semiconductors
D1 = PMEG3010EJ, 115, diode, 30V, 1A
LED1 = low current, red, SMD 0805
LED2 = low current, green, SMD 0805
T1 = SI2347DS, MOSFET, p-channel, 5A, 30V, 0.033Ω
IC1 = TLV809K33DBVR, voltage supervisor
IC2 = TLV75533PDBVR, 3V3 LDO, 500mA, low-I_Q, SOT-23-5
IC3 = optional. Space for: AT25SF081-SSHD-T 8Mbit flash IC
IC4 = STM32F072C8T6TR ARM Cortex-M0 microcontroller
IC5 = optional. Space for: ATECC608A-SSHDA-B, CryptoAuthentification
IC6,IC7 = MAX40200AUK+T 'Ideal Diode' controller, 1-channel, 1A
MOD1 = RFM95W-868S2 LoRa transceiver, Elektor Store SKU 18715

Miscellaneous
X1 = 32.768 kHz oscillator module, SMD, 3.2mm x 1.5mm (Abracon ASH7KW-32.768KHZ-L-T)
S1,S2 = switch, tactile feedback, 12V, 50mA (Multicomp TM-553I-Q-T/R)
Bt1, Bt2 = optional. Space for: AAA battery holder with PCB pins (Multicomp MP000341)
F1,F2 = 500mA PPTC resettable fuse (Bel Fuse 0ZCK0050FF2E)
K1 = PCB pin





K2 = 2-pin pinheader
K3 = 4-pin SIL pinheader
K4 = 5-pin pinheader
K5 = 17-way SIL pinheader
K6 = 3-pin pinheader
K5,K6,K3,K4 = 40-pin SIL pinheader
ANT1 = Wire antenna, 1mm enamelled copper wire, length 8.2cm
PCB 180516 V3.1 from Elektor Store

COMPONENT LIST

For Lora AC Switch Board

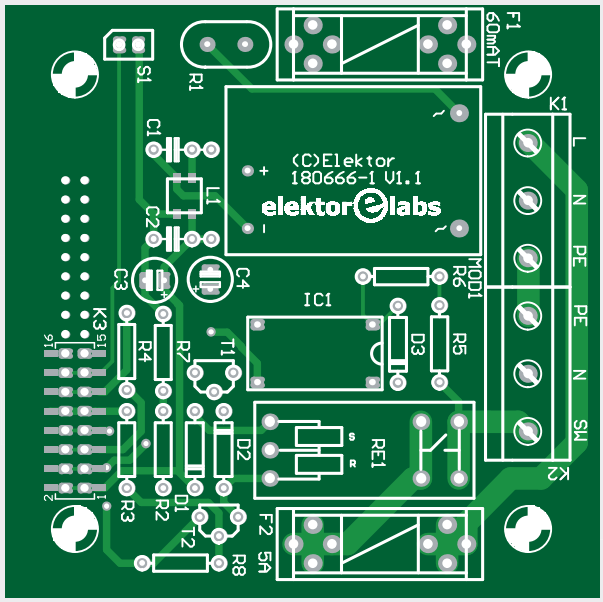
Resistors
R1 = NTC, 120Ω, Epcos type B57236S0121M000
R2,R3 = 10Ω, carbon film, 5%, 0.25W, 250V
R4 = 100kΩ, carbon film, 5%, 0.25W, 250V
R5,R6 = 47kΩ, carbon film, 5%, 0.25W, 250V
R7,R8 = 1kΩ, carbon film, 5%, 0.25W, 250V

Inductor
L1 = ACM4520V-231-2P-T common-mode filter (Farnell # 2455201)

Capacitors
C1,C2 = 100nF, 50V, X7R, 0.2" (5.08mm) pitch
C3 = 1µF, 16V, radial, 5mm
C4 = 10µF, 16V, radial, 5mm

Semiconductors
D1,D2 = 1N4148
D3 = 1N4007
T1,T2 = BS170
IC1 = CNY65 optocoupler

Miscellaneous
RE1 = G5RLK1AEDC5 power relay, latching dual coil, 5V, 16A, SPST (Omron)
K1,K2 = 3-way PCB screw terminal block, 0.3" (7.62 mm), 500V
K3 = 2-row board-to-board connector, 0.1" (2.54mm) pitch, 16 contacts, receptacle, WR-PHD series, surface mount (Würth # 610316243021)
S1 = Pushbutton switch, IP68 class (Alcoswitch PB6B2FM3M1CAL07)
MOD1 = AC/DC PCB mount power supply 5V, 200mA (Mean Well IRM-02-5)
F1,F2 = Fuseholder, PCB mount, 5x20mm
F1 = 60mAT fuse, 20mm
F2 = 5AT fuse, 20mm



Case: Spelsberg type TK PS 94 x 94 x 57 mm, IP66
PCB 180666-1 V1.1 from Elektor Store

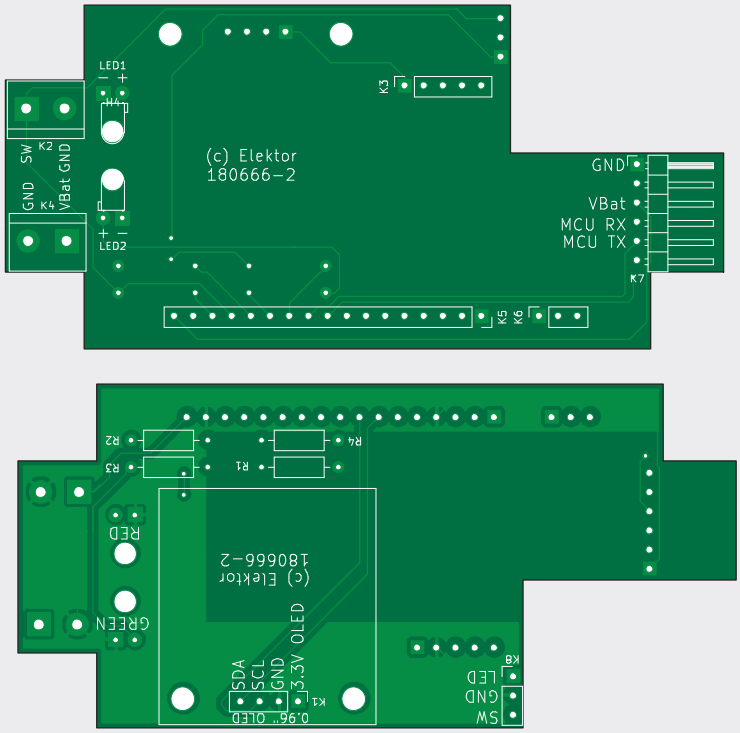
COMPONENT LIST

For LoRa Button board

Resistors
R1 = 270Ω, carbon film, 5%, 0.25W, 250V
R2 = 10kΩ, carbon film, 5%, 0.25W, 250V
R3,R4 = 680Ω, carbon film, 5%, 0.25W, 250V

Semiconductors
LED2 = red, 3mm
LED1 = green, 3mm

Miscellaneous
K2,K4 = 2-way PCB screw terminal block, 0.2" pitch (5.08mm), 630V
K3 = pinheader socket, breakable, 1 row, 5-way, vertical
K5 = pinheader socket, breakable, 1 row, 17-way, vertical (see text)
K6 = Pinheader socket, breakable, 1 row, 3-way, vertical
1 pc. Switch with blue integral LED
Blue 0.96" OLED display, I²C, 4-pin (optional)
Case: HAMMOND 1591-ATBU (Ice Blue)
PCB 180666-2 V1.2 from Elektor Store



K3 employs only seven of the sixteen pins on this PCB. It has two options as discussed below but both assume the LoRa Node board has a standard 0.1-inch pitched, angled connector in position K5. The seven pins in positions 11 through 17 (see board 180516-1) are enough, but K3 can also be a 17-pin pinheader (cut to length) over the full length of K5.
The Würth Elektronik connector identified as K3 in the parts list may be more difficult to obtain compared to a custom cut pinheader. Still, we went for it because of its low height allowing the two boards, mounted at right angles, to be fitted in the recommended housing. We had to take the second row of contacts for granted — we simply couldn't find a single-row socket strip with a suitable height.

However, the footprint on the PCB for connector K3 was adjusted in such a way that normal 0.1-inch through-hole socket strips can also be used. They are useful if you choose a different, taller housing. You can also omit K3 altogether and solder the LoRa Node's right-angled connector the to both PCBs, but then you can't easily separate the PCB anymore.
Pushbutton S1 is mounted on the housing and connected to the PCB with two wires.

Software installation

The complete software package for the Elektor LoRa Node is available at the expected place [6]. The archive file also includes a detailed installation procedure which cannot be printed here due to lack of space.

When it comes to firmware development for the LoRa Node board you can choose the hard-core "C" way using either the STMCubeIDE, or the easy-peasy way with the Arduino IDE. For the latter, you need to download the Arduino IDE and the STM32 Arduino Core. In the Arduino IDE, add this url to the Additional Boards Manager:

Web Links

[1] KiCAD Like a Pro book: www.elektor.com/kicad-like-a-pro
[2] GPS module from Elektor Store: www.elektor.com/open-smart-gps-serial-gps-module-for-arduino-apm2-5-flight-control
[3] STM32duino project: https://github.com/stm32duino/Arduino_Core_STM32
[4] Velocity factor: https://en.wikipedia.org/wiki/Velocity_factor
[5] STM32CubeProgrammer: www.st.com/en/development-tools/stm32cubeprog.html
[6] Project software: www.elektormagazine.com/180666-01
[7] STM32 boards adding: https://github.com/stm32duino/wiki/wiki/Getting-Started

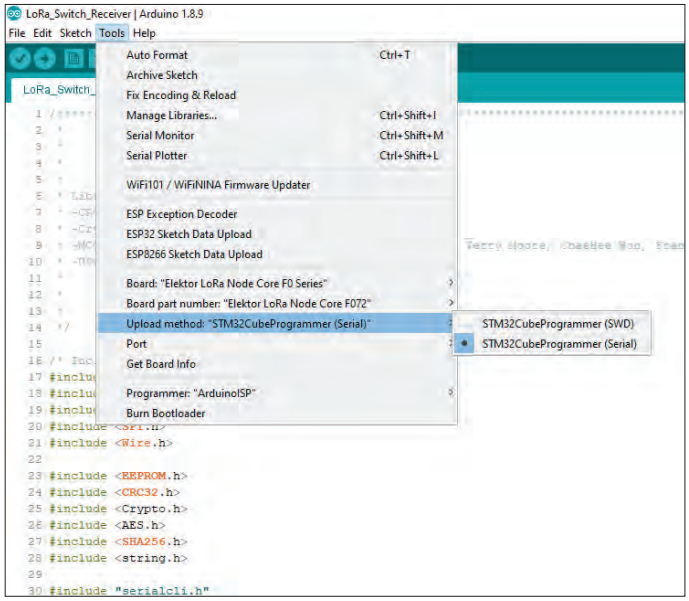


Figure 12. Espy and copy these board settings.

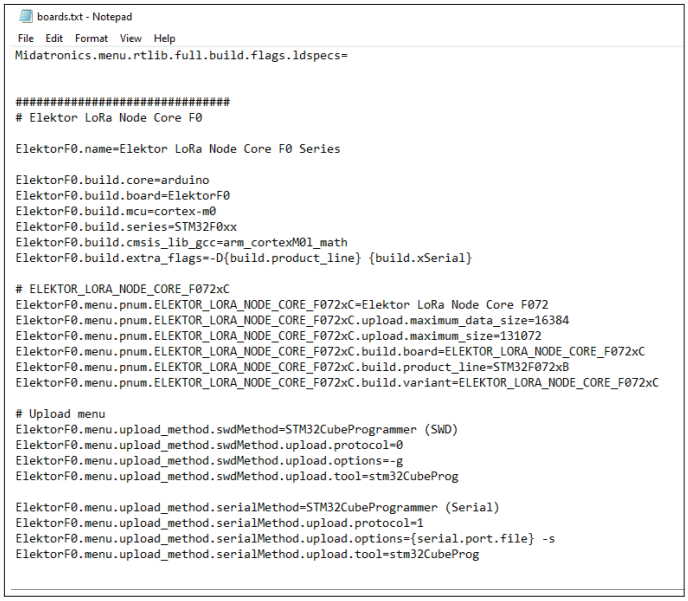


Figure 13. This was just one of the the steps taken by Elektor Labs to get the LoRa Node (an incarnation of the LoRaWAN Node Experimental Platform) approved and registered as an STM32Arduino device. Elektor readers do not have to "enjoy" this administrative process; the project software is ready to roll.

`https://github.com/stm32duino/BoardManagerFiles/raw/master/STM32/package_stm_index.json`

and install the STM32 Boards following the instructions posted at [7].
Now you have the STM32 core but for the Elektor LoRa Node you need to add a few things to your local filesystem. For Windows go to:

`%localappdata%\Arduino15\packages\STM32\hardware\stm32\1.7.0`

and at the end of 'Boards.txt' add the contents of the file called *Add_To_Board.txt* we wrote for you. Also copy the folder:
`ELEKTOR_LORA_NODE_CORE_F072xC`
to the 'Versions' folder. Now you should be good to go and ready to compile the software for the board. At the end you need to define the Boardsettings shown in **Figure 12**.

At this point you are good to go to compile software for the board. To do an upload you need to install the STM32CubeProgrammer [5]. If you now wish to upload code, put the board into bootloader mode by pressing Boot and toggling the Reset button. Now you can upload new firmware to the board.

For your benefit

At the start of the article we said that the story was going to be partly chronological. Since mid-December 2019, the Elektor LoRaWAN Node Experimental Platform board definitions have been merged into the official STM32Arduino Git repository, after complying with the rather strict conditions set by the *STM32duino* 'gremium' **(Figure 13 —** thanks guys). As a happy consequence, you no longer need to patch anything as described above, and you can simply select the Elektor LoRa Node board from the list of approved items. ⏮

(180666-01)