SEPTEMBER & OCTOBER 2024 | ELEKTORMAGAZINE.COM

elektor DIGITAL BONUS EDITION

design > share > earn

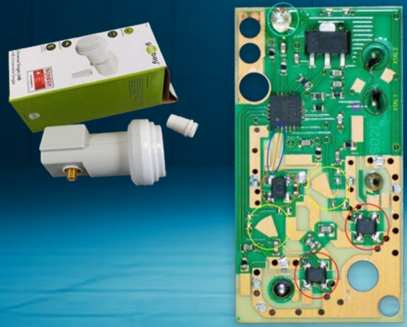
532B
SINCE 1961

FOCUS ON
Wireless & Communication

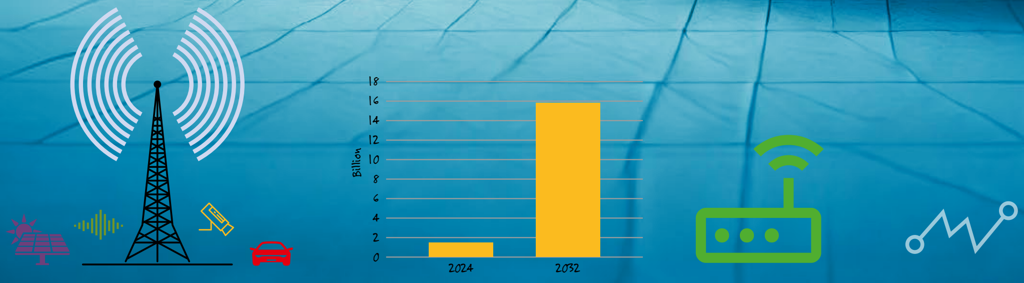
The Ultra-Portable ATS25 max-Decoder Receiver

ESP32-Based ArtNet to DMX Converter
Upgrade Your Legacy DMX Device

Modifying Satellite LNBs for 10 GHz Experiments
Unlocking the X-Band on a Budget



Infographics
Wireless and Communication



CONTENTS

The **September/ October 2024 edition** of *ElektorMag* is available at newsstands and in the Elektor Store.

EDITORIAL

C. J. Abate
Content Director, Elektor

Bonus Content for Wireless Innovators!

If you're looking for more wireless-related content, you've come to the right place. This bonus edition of *ElektorMag* features additional content to inspire you to design and develop new applications.

Want to learn about satellite low noise blocks (LNB)? Read "Modifying Satellite LNBs for 10 GHz Experiments" if you want to get started with the X-band.

Are you a ham or general DIY radio fan? Check out the ATS25 max-Decoder receiver, which long-time Elektor editor Jan Buiting thoroughly reviews. Jan explains that "the tuning and listening experience from the ATS25 max-Decoder is level with many 'big box' receivers."

Curious about ArtNet and DMX? While new ArtNet devices are backward-compatible with the DMX standard, older DMX devices are not inherently compatible with ArtNet. In this edition, we introduce an interface solution that allows you to upgrade your legacy DMX devices, enabling them to work seamlessly with the ArtNet standard.

We hope you enjoy this Bonus edition. We aim to inspire your DIY electronics journey by providing you with hands-on insights and innovative concepts. As you embark on your own projects, don't forget to document your progress on the Elektor Labs platform!

The Team

International Editor-in-Chief: Jens Nickel | **Content Director:** C. J. Abate | **International Editorial Staff:** Asma Adhimi, Roberto Armani, Eric Bogers, Jan Buiting, Stuart Cording, Rolf Gerstendorf (RG), Ton Giesberts, Ouafae Hassani, Hedwig Hennekens, Saad Imtiaz, Alina Neacsu, Dr. Thomas Scherer, Jean-Francois Simon, Clemens Valens, Brian Tristam Williams | **Regular Contributors:** David Ashton, Tam Hanna, Ilse Joostens, Prof. Dr. Martin Ossmann, Alfred Rosenkränzer | **Graphic Design & Prepress:** Harmen Heida, Sylvia Sopamena, Patrick Wielders | **Publisher:** Erik Jansen | **Technical questions:** editor@elektor.com

COLOPHON

Elektor September/October 2024 Bonus Edition

Elektor Magazine is published 8 times a year by
Elektor International Media b.v.
PO Box 11, 6114 ZG Susteren, The Netherlands

Phone: +31 46 4389444

elektor.com | elektormagazine.com

For all your questions
service@elektor.com

Become a Member
elektormagazine.com/membership

Advertising & Sponsoring
Büsra Kas
Tel. +49 (0)241 95509178
busra.kas@elektor.com
elektormagazine.com/advertising

Print
Senefelder Misset, Mercuriusstraat 35,
7006 RK Doetinchem, The Netherlands

Distribution
IPS Group, Carl-Zeiss-Straße 5
53340 Meckenheim, Germany
Phone: +49 2225 88010

PEFC Certified
This product is from sustainably managed forests and controlled sources
PEFC/30-31-151
www.pefc.org

Modifying Satellite LNBs for 10 GHz Experiments

Unlocking the X-Band on a Budget

By Sebastian Westerhold, AI5GW (Germany)

The IEEE X band spans from 8 to 12 GHz and houses many useful services, such as amateur radio, radar, and radio astronomy. With this low-cost project, entering the magical world of double-digit GHz becomes possible! Instead of designing a microwave RF circuit from scratch, which has many critical aspects, the solution is presented here is a modification of a commercial LNB.

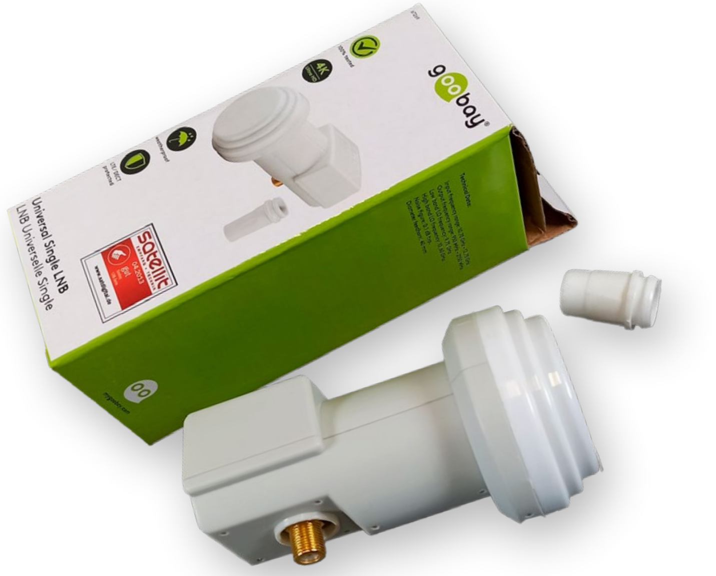


Figure 1: Goobay 67269 Universal Single-LNB for satellite TV reception.

With the launch of the geostationary TV satellite, Es'hail 2 (also known as Quartar-OSCAR 100 or QO-100), which also carries a linear transponder for amateur radio use with a downlink centered around 10.45 GHz, DIY X-band projects have seen a significant upturn. Due to the closeness to the Ku-band commonly used for TV satellite reception, inexpensive LNBs for television reception provide a great starting point for delving into the world of double-digit GHz DIY projects.

Whether the goal is to listen to radio amateurs from all of Europe or to upgrade an existing spectrum analyzer to cover 10...12 GHz for less than €10, this article provides what's needed, by showing how to turn an inexpensive broadcast TV reception LNB into a stable X-band downconverter with adjustable local oscillator frequency.

Modification Principle

In Europe, Ku-band satellite TV downlinks are primarily situated in the frequency range of 10.70...12.75 GHz, which is conveniently close to the 10.45...10.50 GHz range used by amateur radio satellites such as QO-100. As a matter of fact, it is so close that a commercial TV low-noise block downconverter (LNB) can theoretically be used directly without the need of any modification.

Unfortunately, the frequency stability requirements for broadband TV signals are a lot more relaxed than what is desirable for narrowband single-sideband (SSB) voice — or even CW — signals used by amateur radio operators. Therefore, the primary goal of the modification shown here is to increase the stability of the LNB's local oscillator. This is achieved by injecting an externally provided reference signal for the PLL. Being able to adjust the local oscillator (LO) frequency can be done freely after performing the modification, which turns the LNB into an inexpensive and powerful downconverter.

Theory of Operation

The chosen LNB for this article is shown in **Figure 1**. It is a universal single LNB (Goobay 67269) with a recommended retail price of around €6 [1], so it leaves plenty of room for errors without breaking the hobby bank. The reception plane is linearly polarized. The exact polarization is selected by varying the supply voltage. Vertical polarization is selected by supplying 11...14 V DC to the LNB. Increasing the supply voltage to 16...18 V changes to horizontal polarity. The LNB's block diagram is illustrated in **Figure 2**, including a blue-enhanced example of a possible selection.

A large gain — typically 57 dB — compensates sufficiently for any losses from the off-frequency bands. While circular waveguides and (corrugated) horn antennas are typically extremely broadband by nature, they do have a cut-off frequency below which the attenuation constant increases quickly.

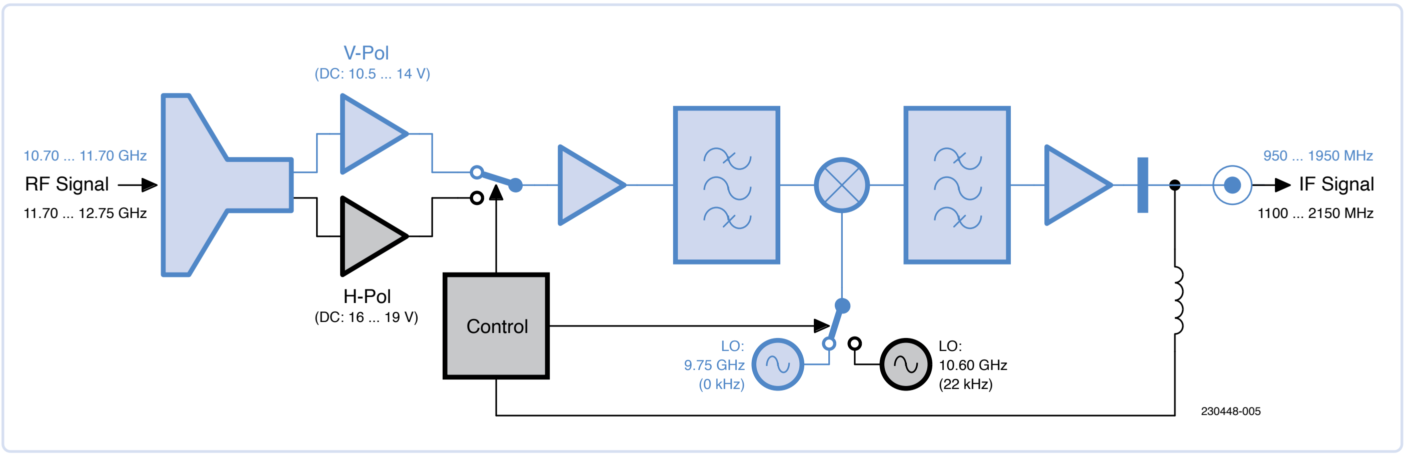


Figure 2: Block diagram of a Ku-band satellite TV LNB.

The LNB is intended for reception between 10.70 GHz and 11.70 GHz (low band) and 11.70...12.75 GHz (high band). The intermediate frequency (IF) range spans from 950 MHz to 1,950 MHz (low band) and from 1,100 MHz to 2,150 MHz (high band). This corresponds to local oscillator frequencies of 9.75 GHz (low band) and 10.60 GHz (high band). Switching from low band to high band is achieved by an external 22 kHz control signal. If no 22 kHz control signal is detected, low-band operation is selected by default. The frequency reference for the phase-locked loop (PLL) is a 25 MHz crystal. The crystal's frequency is multiplied by 390 in low-band operation, while, in high-band operation, the multiplication factor is 424.

Being able to inject an external reference signal therefore opens the possibility to change the LO frequency to enable the use of existing communications receivers designed for the 2 m (144 MHz) or 70 cm (432 MHz) amateur radio band (see **Table 1**). One example: Injecting 26.525 MHz instead of 25 MHz results, when multiplied by 390, in an LO frequency of 10.345 GHz and therefore an IF of around 144 MHz at 10.489 GHz.

Inside the LNB

After removing the plastic cover using a flat screwdriver, the metal cover for the PCB housing can be removed after loosening two screws (Torx T8). As visible in **Figure 3**, the LNB's circuit is centered around a fully integrated PLL-/LO-/cownconverter chip, marked 3566E. Very little information about this chip can be found. However, knowing the general working principle of such an IC, the relevant parts of the circuit can be reverse-engineered without specific device information at hand.

Table 1: Possible Reference Frequencies and the Resulting IF Frequency Range for QO-100 Narrow-Band Reception.

Reference	LO	IF (at 10,489.5 MHz)	IF (at 10,490.0 MHz)
25.000 MHz	9,750.00 MHz	739.50 MHz	740.00 MHz
25.780 MHz	10,054.20 MHz	435.30 MHz	435.80 MHz
25.788 MHz	10,057.32 MHz	432.18 MHz	432.68 MHz
26.525 MHz	10,345.53 MHz	144.36 MHz	144.86 MHz

Aside from two radial stubs in the supply lines of the two FET front-end amplifiers (one for each polarization plane — circled in yellow on Figure 3) there are almost no filters on the PCB, which is good for using the LNB outside of its design parameters. After the second amplifier stage (common for both polarizations, circled in orange), there is a single coupled line band-pass filter (within the blue ellipse), which will inadvertently attenuate signals outside the designed pass band. But again, there's plenty of gain to make up for any anticipated losses.

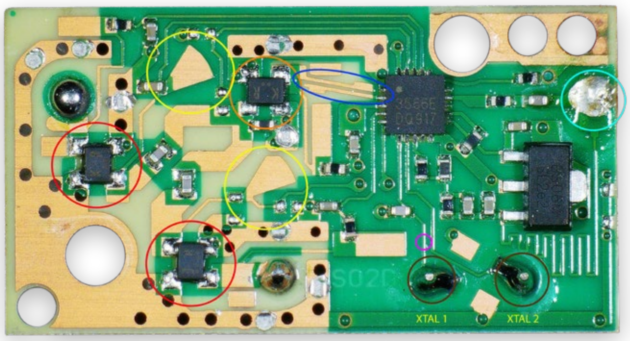


Figure 3: Top view of the LNB's PCB. Please refer to the text for identification of the various components.

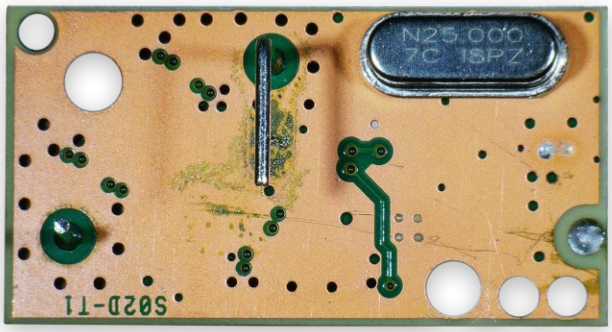


Figure 4: Bottom view of the PCB, with the 25 MHz reference crystal in the top-right corner.

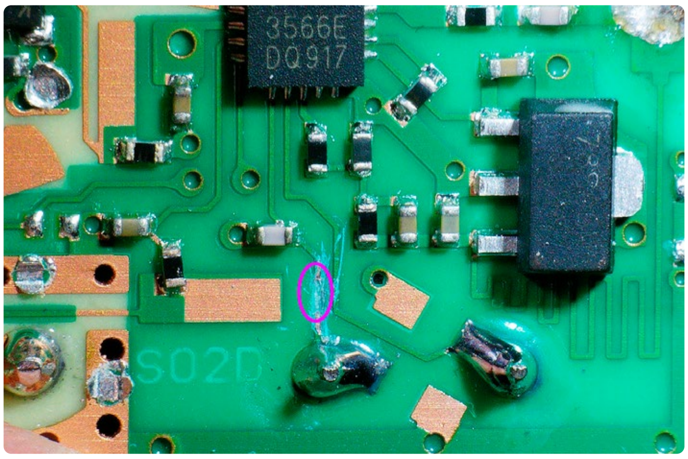


Figure 5: Cut the track leading to XTAL1 where indicated.

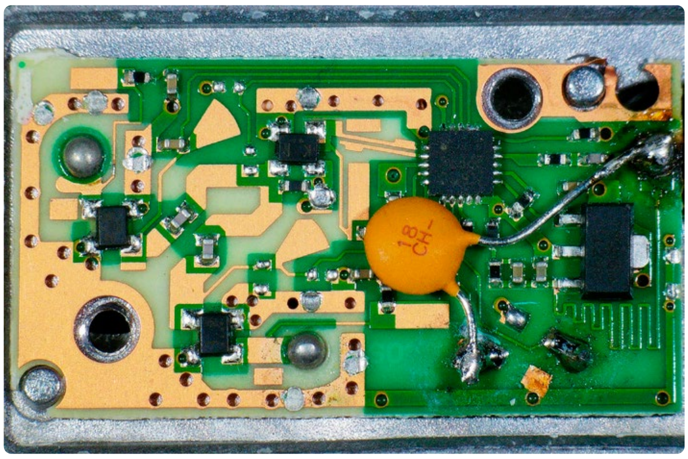


Figure 6: The 18-pF capacitor soldered between the XTAL1 and F connector input pins.

The DC supply voltage from the F-connector (whose solder-side is circled in light-blue) is passed to a 7806-type voltage regulator through a printed circuit board inductor, supplying the 3566E with stable 6 V DC. The brown-circled XTAL 1 and XTAL 2 pins on Figure 3 are connected to the 25 MHz reference crystal located on the back (silkscreen) side of the PCB, as shown in **Figure 4**.

After desoldering a single solder joint right above the F-connector, the PCB can be removed from the housing. It was found to be best to apply slight upward pressure to the PCB using a screwdriver while heating the solder joint.

Modification

The modification can be summarized as removing the 25 MHz crystal and replacing it with a series LC band-pass filter connected to the input connector, to allow an external reference clock signal to find its way into the PLL. The primary challenge is to fit the components into the tight aluminum case.

Selected from available standard values, an inductance value of 2.2 µH and a capacitance value of 18 pF are chosen. Resulting impedance Z_{LC} can be calculated using the equation

$$Z_{LC} = |\, 2\pi f L - \frac{1}{2\,\pi\, f C}|$$

where *f* is t/(he freq)uency in Hz, *L* the inductance in H and *C* the capacitance in F. At 25 MHz, this results in an impedance of ≈ 8.1 Ω.

The crystal was desoldered and the inductor soldered in its place. The trace between PIN XTAL 1 and the 3566E (enhanced in purple on **Figure 5**) is cut using a sharp object. Care needs to be taken to not break the thin and fragile PCB. A connection is then established between XTAL 1 and the center pin of the input connector using the 18 pF capacitor, as shown in **Figure 6**. Both the inductor and the capacitor are protected from accidental contact with the casing or other traces using polyimide tape. Electrical tape or even paper can be used instead.

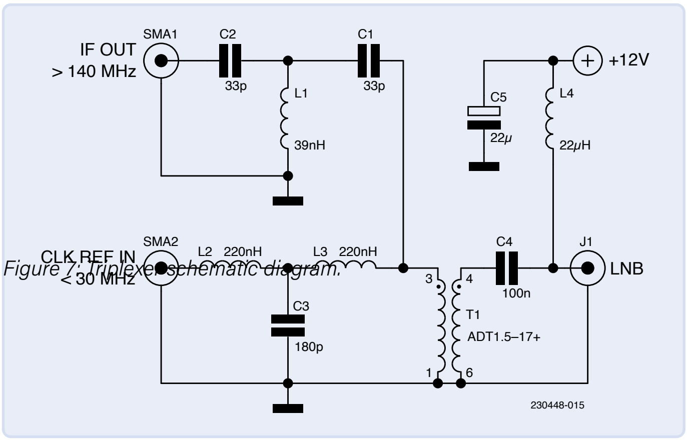


Figure 7: Triplexer schematic diagram.

Diplexer With Integrated Bias-T (Triplexer)

Operating the modified LNB requires some means of feeding different signals — through the same coax cable — to and from the LNB. It must be provided with the proper DC supply voltage, as well as the reference clock signal. Additionally, the returned IF signal present on the same coax cable must be separated as well. Supplying power is typically achieved through a bias-T. Separating the reference and IF signals can be achieved using a diplexer. Combining the two into a single circuit will make it a triplexer. Diplexers and triplexers are very often used by hams as a kind of RF combiner/splitter, for the purpose

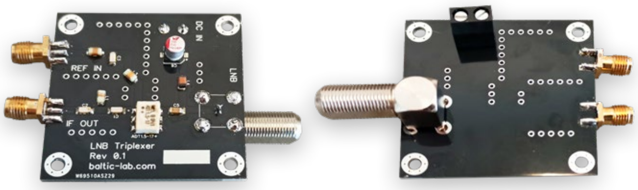


Figure 8: The triplexer for reference frequency and IF signal separation and supply voltage injection. On the back side (right), the F connector for the LNB and the input voltage screw-terminal block are located.

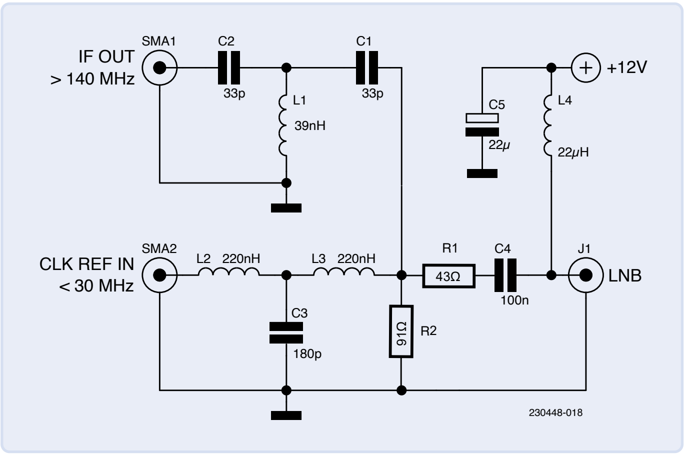


Figure 9: The triplexer circuit, without the HF transformer and with the add-on of R1 and R2 in a minimum-loss-pad configuration.

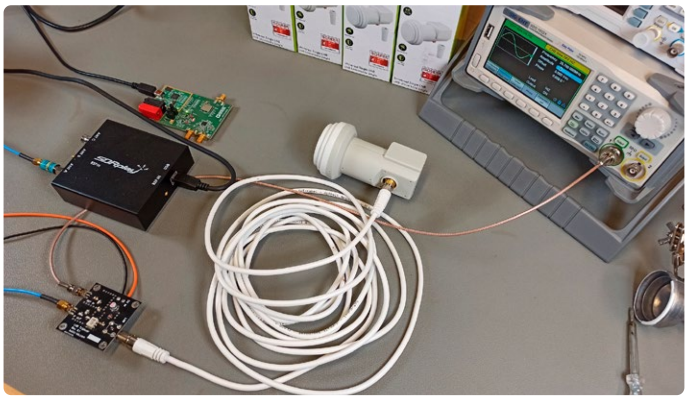


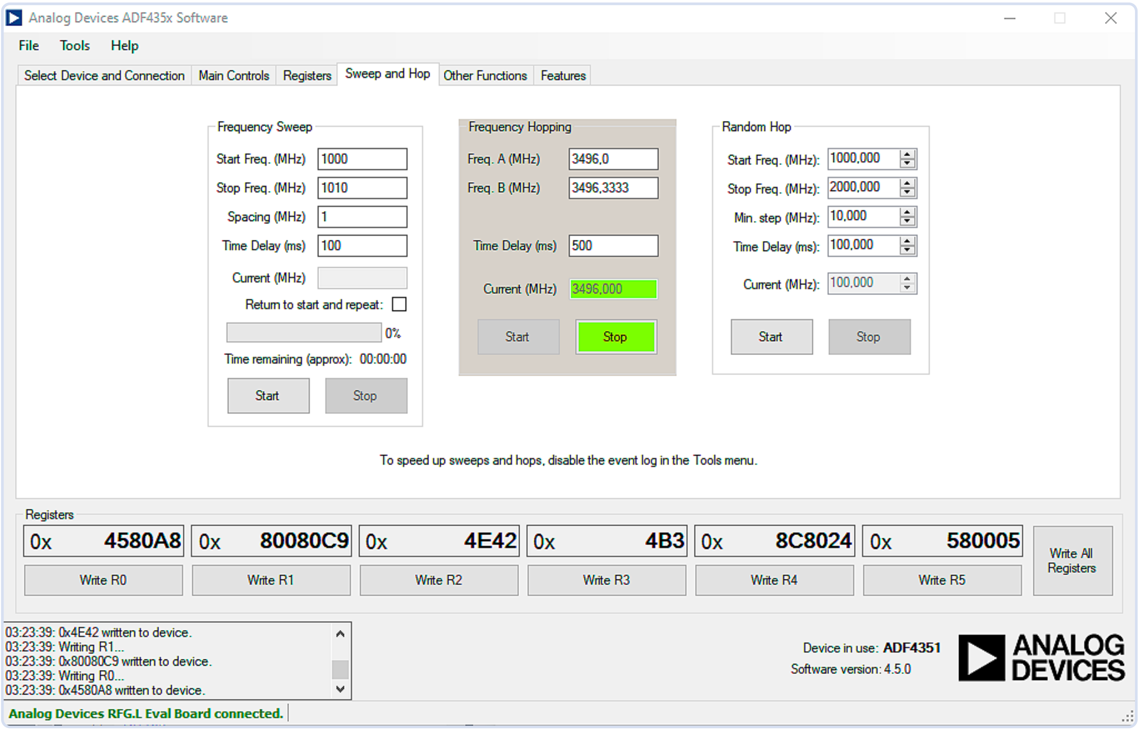
Figure 10: Test setup with a signal generator, triplexer, SDR receiver and ADF4351 used as third-harmonic X-band signal generator.

of using two/three RTX devices to reach two/three antennas with a single 50 Ω or 75 Ω coaxial cable where, at the opposite end, a similar device separates the different frequency bands.

In our case, the triplexer, whose schematic is shown in **Figure 7**, also needs to provide some means of impedance matching between the LNB's 75 Ω system impedance and the more-commonly-found 50 Ω for the IF receiver and reference clock generator. This circuit acts as an HF mixer, injecting the signal from the external generator into the LNB. This signal reaches T1 through the "T" low-pass filter consisting of L2, C3, and L3, while the IF signal coming out of the LNB through T1 reaches the "T" high-pass filter consisting of C1, L1, and C2.

It should be noted that the impedance mismatch can safely be ignored for first experiments. The resulting mismatch loss without impedance matching is 0.18 dB, corresponding to a forward power of 96% or a reflected power of 4%.

Figure 8a and **Figure 8b** show my implementation of a triplexer with surface mount components and a Mini-Circuits ADT1.5-17+ transformer. Of course, the entire circuit can also be built using discrete components, and the transformer can be replaced with a minimum-loss pad (MLP) using a 43 Ω resistor in series with the 75 Ω side and a 91 Ω shunt resistor across the 50 Ω side, as shown in the modified schematic in **Figure 9**. Note, however, that this mod will result in a


Analog Devices ADF435x Software
File Tools Help
Select Device and Connection | Main Controls | Registers | Sweep and Hop | Other Functions | Features
Start
Stop
Write R0
Write R1
Write R2
Write R3
Write R4
Write R5
Write All Registers
ANALOG DEVICES

Frequency Sweep
Start Freq. (MHz) 1000
Stop Freq. (MHz) 1010
Spacing (MHz) 1
Time Delay (ms) 100
Current (MHz)
Return to start and repeat: ☐
0%
Time remaining (approx): 00:00:00

Frequency Hopping
Freq. A (MHz) 3496,0
Freq. B (MHz) 3496,3333
Time Delay (ms) 500
Current (MHz) 3496,000

Random Hop
Start Freq. (MHz): 1000,000
Stop Freq. (MHz): 2000,000
Min. step (MHz): 10,000
Time Delay (ms): 100,000
Current (MHz): 100,000

To speed up sweeps and hops, disable the event log in the Tools menu.

Registers
0x 4580A8 | 0x 80080C9 | 0x 4E42 | 0x 4B3 | 0x 8C8024 | 0x 580005

```
03:23:39: 0x4E42 written to device.
03:23:39: Writing R1...
03:23:39: 0x80080C9 written to device.
03:23:39: Writing R0...
03:23:39: 0x4580A8 written to device.
```

Device in use: ADF4351
Software version: 4.5.0

Analog Devices RFG.L Eval Board connected.

Figure 11: Analog Devices ADF4351 software used to establish frequency hopping between 3,496 and 3,496.333 MHz for third harmonic X-band test signal generation.

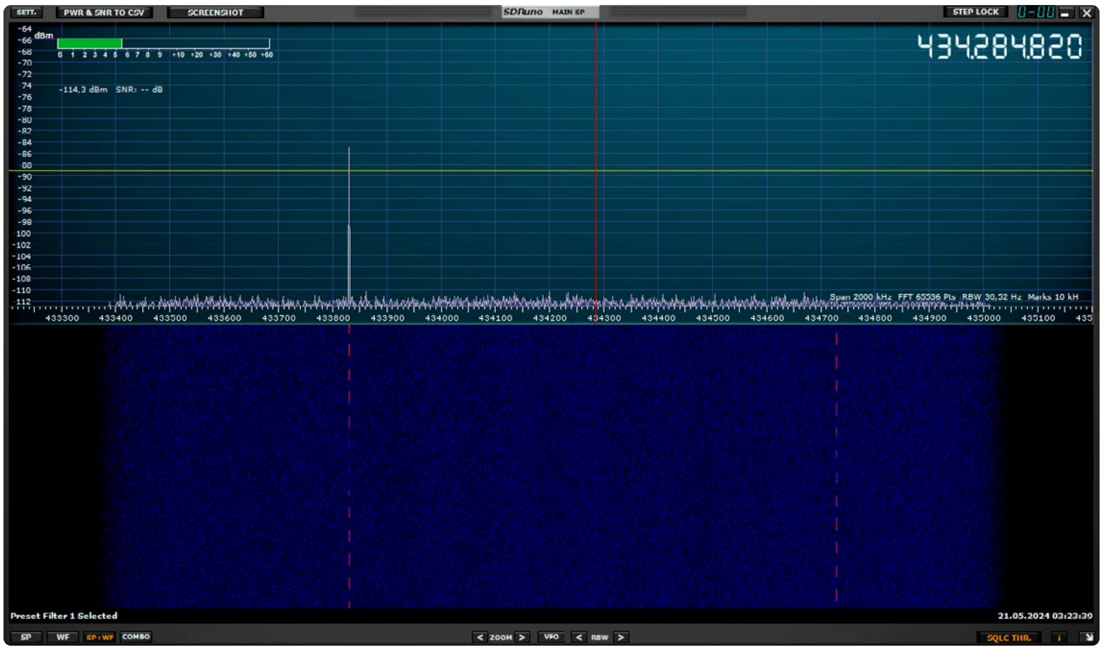


Figure 12: The received IF signal's SDRuno spectrum display, with clearly visible frequency hops spaced 1 MHz apart.

power loss of around 6 dB, which is larger than the mismatch loss if impedance matching is omitted. But, it will ensure that the waves traveling through the coax are well-behaved.

Testing and Results

Figure 10 shows a testing session, where the LNB is connected to the Triplexer using a few meters of coax with an impedance of 75 Ω. The reference clock input is connected to a signal generator and the IF output is connected to a SDRplay RSPdx SDR receiver (see [2] and the Related Product text box) through a 10 dB attenuator. With the signal generator's output still disabled, a voltage of 14 V DC is applied to the LNB through the triplexer. The current draw is observed to be around 60 mA. The signal generator is set to a frequency of 25.78 MHz (LO = 10054.20 MHz) and the signal amplitude is slowly increased. At around 2 V peak-to-peak, a jump in current consumption to roughly 90 mA can be observed. The PLL is now locked to the external clock reference signal.

For a test, two signals — 3,496 MHz and 3,496.333 MHz — are generated using an ADF4351 evaluation board. The third harmonics provide suitable test signals with frequencies of 10.488 GHz and 10.489 GHz, respectively. As can be seen in the screenshot in **Figure 11**, the ADF4351 evaluation board is set to hop between both frequencies at the rate of 500 ms.

The expected IF output centered around 434.3 MHz is observed using SDRuno, an advanced software-defined radio application platform [3], as shown in **Figure 12**. To make sure that the IF signal is indeed the desired, down-converted X-band harmonic, the frequency of the generated signal is changed by a small amount and the change in the IF frequency is noted. The IF signal's frequency change must be equal to the change in fundamental frequency times the harmonic number. In the case of the signals generated by the ADF4351, the expected and observed change between the two frequencies is 1 MHz, confirming the reception of the correct harmonic.

Due to the strong amplification of the LNB, even the 23rd harmonic of an amateur radio transceiver tuned to 456 MHz can be picked up at 10.448 GHz. This is extremely useful if higher frequency test sources are not available, and stands true to my claim that this modification and corresponding experiments can be done on a budget.

Room for Improvement

This article shows that it is possible to successfully modify an inexpensive TV LNB to accept an external reference signal, and the experiments demonstrate that the LNB is capable of down-converting X-band signals outside the manufacturer's intended frequency range.

Room for improvement is multifaceted and only limited by one's imagination. For instance, the PCB inductor can be replaced by a larger SMD inductor to limit loading of the external reference signal. Both the radial stubs and coupled-line bandpass can be improved by adding a small amount of conductive material. One can also completely remove the feedhorn and add an SMA connector by soldering some semi-rigid cable directly to the input of the FET amplifier, through a small capacitor to block the DC bias voltage present on the FET's gate.

The range of possible use cases is manifold, and only a few examples are mentioned in this article. Considering the low price of the LNB itself — as well as the components used — I would like to invite you to experiment with your own ideas and applications!

230448-01

About the Author

Sebastian Westerhold is a self-taught independent researcher and blogger with a passion for radio frequency and DSP. He began writing for Elektor in 2008 and has since published numerous articles on his blog [4], YouTube channel [5], and Elektor. Currently, he is studying Electrical Engineering at the University of Applied Sciences Kiel. Prior to his studies, he already earned higher semester academic credits from the Chair of High-Frequency Technology at Kiel University (Christian-Albrechts-Universität zu Kiel) through a junior study program for gifted students, while he was still attending a technical high school.

Questions or Comments?

Do you have technical questions or comments to submit about this article? You may write to the author at sebastian@baltic-lab.com or contact the editorial team of Elektor at editor@elektor.com.

Related Products

> **SDRplay RSPdx – Single-Tuner 14-bit SDR Receiver (1 kHz to 2 GHz)**
www.elektor.com/20422

WEB LINKS

[1] Goobay Universal Single LNB (Wentronic): https://wentronic.com/de/universal-single-lnb-67269
[2] SDRplay RSPdx: https://sdrplay.com/rspdx
[3] SDRuno: https://sdrplay.com/sdruno
[4] Baltic Lab website: https://baltic-lab.com
[5] Baltic Lab channel on YT: https://www.youtube.com/c/balticlab

Cutting through Complexity and **Scaling with a Unified Development Platform**

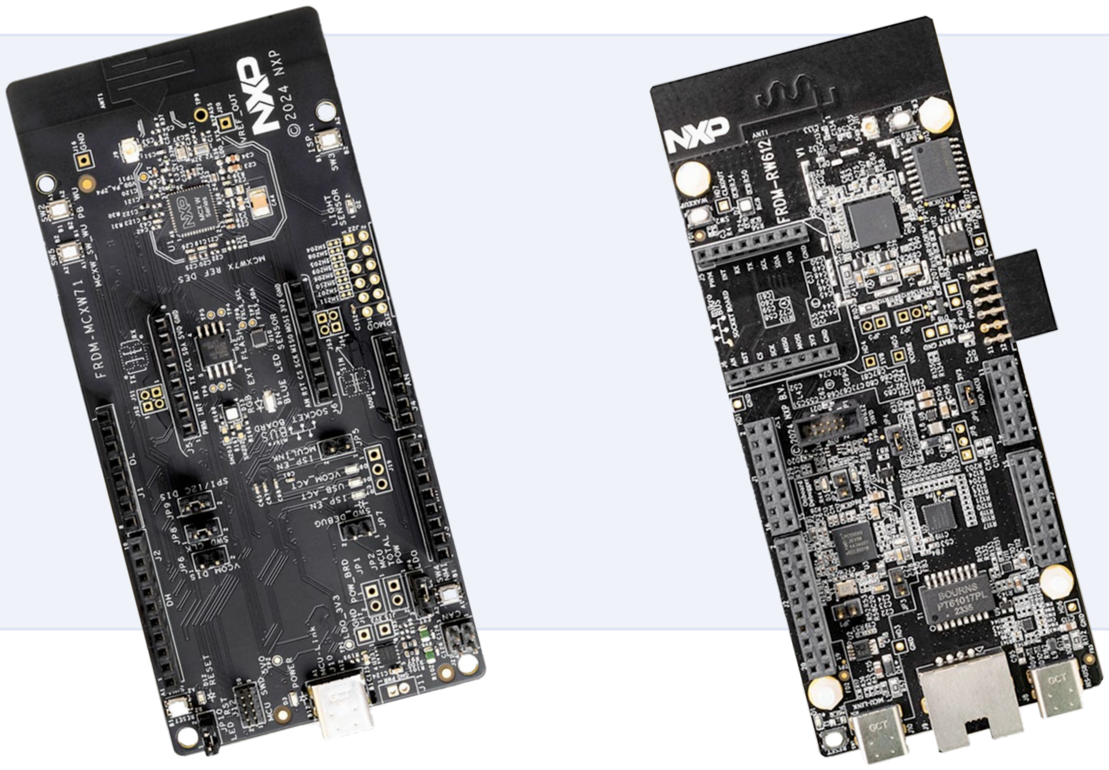


Figure 1: MCX W, and RW61x FRDM boards.

By Charlie Ice (NXP Semiconductors)

The IoT has brought incredible innovations, which have transformed factories, buildings, and homes. At the same time, the IoT has brought ever-increasing complexity to embedded developers. Customers, both consumers and businesses, expect an array of products with varying features, capabilities, and connectivity. IoT products must meet strict energy efficiency regulations, requiring sophisticated motor control and power management. Many incorporate AI/ML to bring intelligence to the end devices. Finally, security has become central to IoT, with new regulations such as the EU Cyber Resilience Act and the U.S. Cyber Trust Mark raising the bar for security on IoT devices.

Faced with this myriad of product variants and sophisticated systems, developers face unprecedented complexity. However, using a platform approach to the integrated development environment (IDE), software, and hardware allows developers to more effectively manage that complexity, scale their efforts, and meet the demands of the ever-changing IoT.

Integrated Development Environment

The interface that an embedded developer likely uses more than any other is the IDE, which provides the portal for code editing, programming, and debugging the embedded device. Many developers work with multiple processors from different suppliers, which often forces them to interact with multiple IDEs from different vendors. Configuring and maintaining the build scripts and debug settings across different IDEs quickly becomes a time-consuming chore. Standardizing on an IDE across embedded devices greatly simplifies a central part of the development process, which is why many embedded device suppliers have adopted Visual Studio Code (VS Code) as their IDE. Supported across Windows, Mac, and Linux, Microsoft offers VS Code as a free development environment with the latest features, such as easy access to GitHub, an advanced text editor, and workspace configuration. Embedded device suppliers provide VS Code plugins that allow their devices to be programmed and debugged directly in VS Code. For example, NXP's plugin offers support across the MCU portfolio, including MCX, wireless, and i.MX RT MCUs. By selecting VS Code as their IDE platform, developers can set up their IDE and use that configuration across products, suppliers, and embedded devices, saving time and effort.

Software Development Kit

The next key piece for an embedded developer is the software development kit (SDK) that supports the embedded device. Most developers designing IoT products not only use different devices in their products, but also need to incorporate different middleware, such as touch, voice, or graphics, into the product. Adopting embedded devices with a common SDK platform greatly accelerates software development. In addition, a common SDK platform allows middleware to easily be added or removed depending on the product variant. Many embedded device vendors now offer a common SDK for their devices. For example, as shown in **Figure 2**, NXP's MCUXpresso SDK supports all of NXP's Arm Cortex-M based devices, including MCX, RW61x, and i.MX RT. This allows developers to use different processors in their designs while still using the same SDK. In addition, the MCUXpresso SDK includes middleware, such as eIQ for AI/ML, that can easily be added or removed from a project. A common SDK platform not only accelerates development, but also allows designers to scale their work by reusing their code across various projects and embedded devices.

Common Hardware Platform

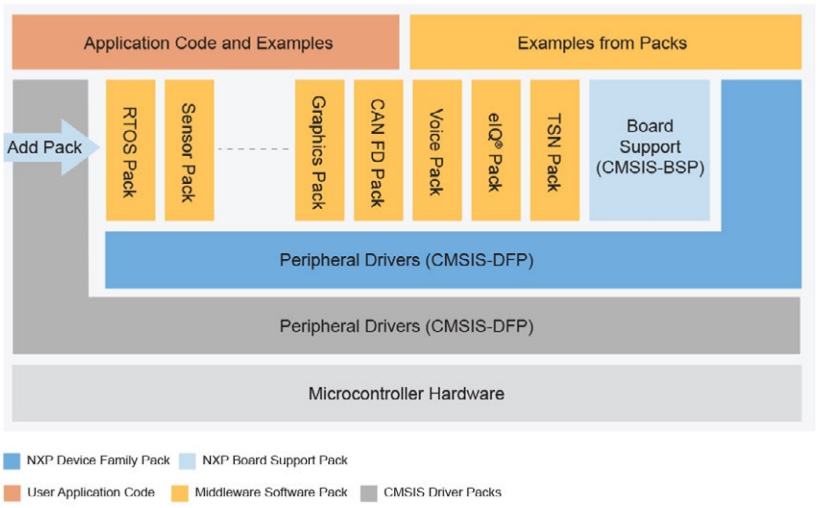
Beyond software development, adopting a common prototyping hardware platform reduces complexity and allows designers to quickly prove out new designs. Most IoT offerings include a range of products with various embedded processors inside them. Therefore, the hardware prototyping platform must be common across different processors, while also allowing developers to easily add new functionality. Re-spinning a board to try out each new feature or having to learn new hardware for each processor creates unnecessary complexity and inefficiency. Instead, developers must either create their own common hardware platform or select an embedded device supplier who offers such a platform. For example, as shown in **Figure 1**, NXP's FRDM board platform offers low cost, compact development boards across the entire MCX and RW61x portfolios. In addition, a hardware platform needs easy prototyping, which is often achieved through add-on boards. By adopting common headers, such as the Arduino and mikroBUS headers used on the NXP FRDM boards, designers can leverage the wide range of available add-on boards or quickly spin their own add-on board. A flexible and common hardware platform across embedded devices improves efficiency and allows designers to focus on innovation.

Adopting a platform for the IDE, SDK, and prototyping hardware greatly reduces the complexities of IoT design and frees designers to choose the right processors and features for their design. By selecting common software and hardware development platforms, designers can achieve the scale and design efficiency to meet the ever-increasing complexity and demands of the IoT. ◀

240461-01

Figure 2: MCUXpresso SDK structure.

▼



The Ultra-Portable ATS25 max-Decoder Receiver

By Jan Buiting (Elektor)

73, all hams and radio fans! Once again, Elektor presents a pronounced combination of embedded and radio technology in the form of the ATS25 max-Decoder. This device is a powerful, ultra-compact multimode LW/SW/MW/FM/Ham band DSP receiver, powered by an ESP32 core and with Wi-Fi connectivity for advanced decoding modes.

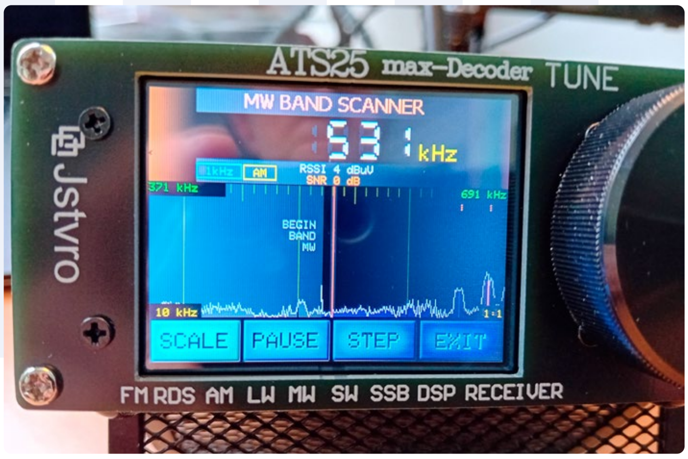


Figure 1: Scanning the MW band for activity using the built-in spectrum analyzer. Not a lot to hear at daytime but come again near dusk and the party starts.

Radio in general and ham radio in particular have been pioneering, challenging and rewarding fields of application for fans of microcontrollers, programming, and small embedded systems. This is mostly due to the arrival, not so long ago, of ICs and drop-in integrated modules that take care of all the high-frequency signal conditioning and then offer a set of pins responsive to digital levels for the control and even for outputting digital levels. What more can a programmer want?

Embedded — With a Radio Attached

One IC family that's been highly successful in encouraging even the staunchest supporters of all-analog RF technology to embrace (or at least accept) "digital", is the SkyWorks Si473x series. Of these, the Si4735 [1] is a "CMOS 100% AM/FM/SW/LW radio receiver," meaning it integrates the complete tuner function from antenna input to audio output. Although the chip went viral among hams and other radio lovers when Arduino software appeared controlling it, its superb performance wasn't fully exhausted until an international group of hams and programmers grabbed a slightly more powerful microcontroller called ESP32 and started to write ingenious software, not just for the LW/MW/SW/FM broadcast bands but also for the radio amateur bands from 160 m through 10 m and the associated "ham modes" like CW, RTTY, USB/LSB, FAX, and others.

The Si4735 chip and the ESP32 micro first showed up jointly in the "ATS25" multimode receiver, which has been around for a few years now, performing well as a general-purpose radio for listening to broadcasts and ham communications. Recently, however, the "Jstvro" group added a bunch of options, advanced digital decoding, Wi-Fi, and more intelligent DSP audio processing to the standard ATS25, which in fact got a complete makeover in terms of control software.

Can't Believe It's So Small

The first thing that struck me when unboxing my review copy of the ATS25 max-Decoder was its compactness, sturdiness, light weight, and the presence of just one control on the front panel: a 40 mm diameter knob with the immediate feel of a rotary encoder with push-button action. On the back of the radio are two antenna

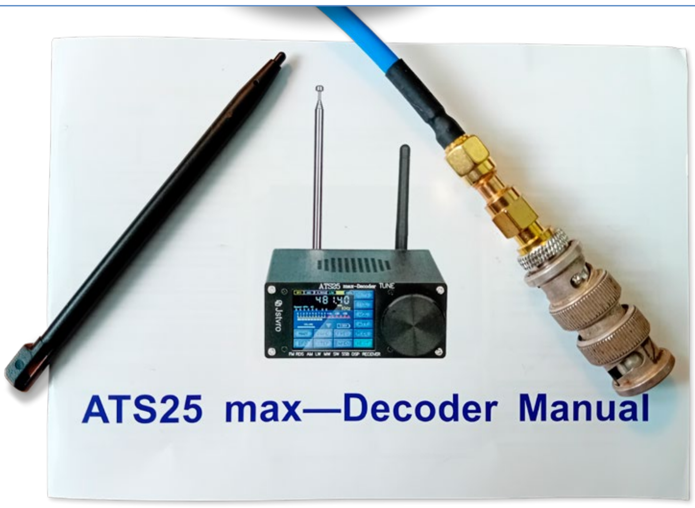


Figure 2: My makeshift SMA-to-BNC adapter (in two steps but 50 Ω throughout), the indispensable stylus (left) and the printed manual.

inputs — one SMA socket for the Wi-Fi (2.4 GHz) antenna and one BNC socket for the antenna. There's also a USB-C connector, which doubles for charging the internal Li-Ion battery and for communication with your PC. A headphones socket and two minute slide switches complete the "user interface" on the back panel.

I was pleasantly surprised by the inclusion in the package of a stylus pen, a foldable Wi-Fi antenna, a telescopic rod antenna for FM, a USB-C cable, and even a cloth which I guess is for dust protection or for cleaning the touch screen. Apart from the radio, of course, the best find in the box was the ATX25 max-Decoder Manual which, although not perfect, is a league above the usual grot you get with Chinese electronics. The 24-page manual is properly printed, has clear pictures with legends and captions to show the most important functions, and is really worth browsing with the radio off! I was informed that Elektor considers enhancing the manual and printing their own version.

Antenna First — Always

All microcontroller fans must remember this: it's not bits and bytes, but a proper antenna that's your radio's best friend. The included telescopic antenna is great for FM band reception when fully extended, but if you use it on LW/MW/SW or ham bands, you will be disappointed. Instead, roll out at least 10 meters of flexible wire, string it up outdoors and simply clip the free end to the telescopic antenna. Then the fun begins, unless.... As in my case, there's massive interference from (digital) sources like LED lights, computers, smartphones, and my 7.5 kWp solar panel installation, in particular, the SolarEdge 3-phase inverter. So I switched that off and got rid of 180-kHz spaced harmonics and spurious signals extending to well over 60 MHz. The ATS25's internal noise is present but manageable, as I found by silencing the antenna input with a 50-ohm terminator. The remaining noise is burst-like from operating the rotary encoder, and from the OLED screen.

Due to radio wave propagation, daytime reception in the MW and LW bands is poor even if you have a wire antenna. However, the ATX25 max-Decoder was able to find several stations around 675 kHz and 1000 kHz and turned out to work just as well as my Grundig Yacht-Boy and even a 1960s Philips "BX" tube radio, both using ferrite rod antennas. I also found good old Droitwich on 198 kHz LW with a BBC cricket match report. The MW band in particular comes alive in the evening hours and is great fun to tune across to find pirate and "low-power" stations for private use.

On my ATX25 max-Decoder, the FM band extends from 64 to 108 MHz, which I haven't seen before and allows me to at least monitor the 4-meter (70 MHz) band, which is not available under the "HAM" bands. It also picked up some 80 MHz police radio traffic from just across the border in Germany.

I eventually changed the antenna for a wire loop, which responds to magnetic fields instead of electric fields, as does the plain wire antenna. However, the small loop antenna available from Elektor is terminated in an SMA connector while the ATS25 receiver has BNC socket. So I slapped up an adapter by cascading an SMA-to-BNC adapter and a BNC male-to-male adapter. This worked well and reduced man-made noise massively, thanks to the loop being directive so it can be tuned to eliminate noise sources like LED lamps, LCD screens, etc.

It's best to hang up the loop antenna outdoors, as far away from noise sources, and feed the cable inside to your receiver.

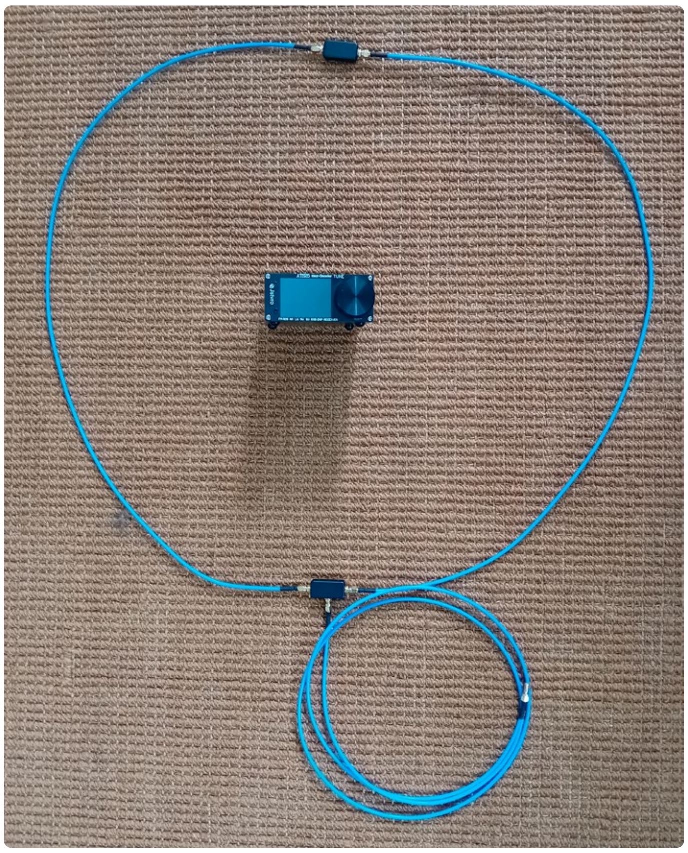


Figure 3: The "YouLoop" from the Elektor Store is a good starting point for a proper antenna. Remember though that it's not really suitable for permanent outdoor use. Also, there should be nothing inside the loop. :-)

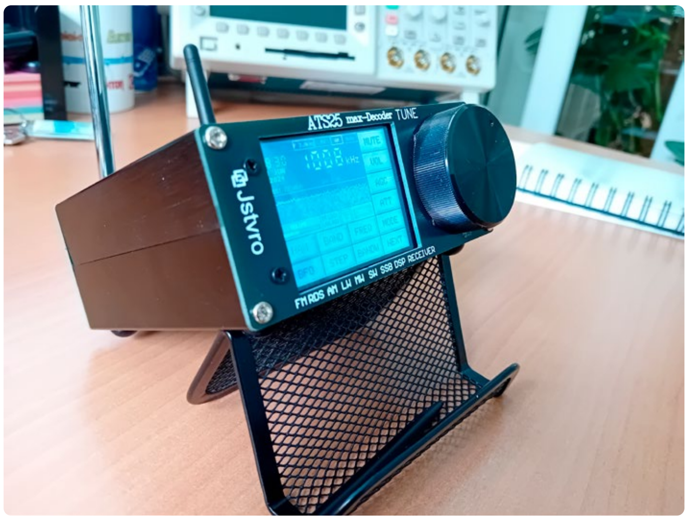
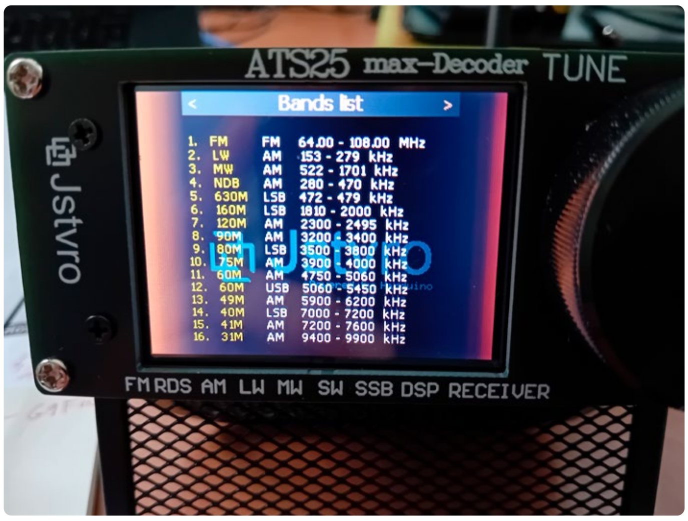


Figure 4: I used a cheap smartphone holder to secure the receiver at a comfortable angle for viewing and operating. The stylus also has its place.



Bands list

1.	FM	FM	64.00 - 108.00 MHz
2.	LW	AM	153 - 279 kHz
3.	MW	AM	522 - 1701 kHz
4.	NDB	AM	280 - 470 kHz
5.	630M	LSB	472 - 479 kHz
6.	160M	LSB	1810 - 2000 kHz
7.	120M	AM	2300 - 2495 kHz
8.	90M	AM	3200 - 3400 kHz
9.	80M	LSB	3500 - 3800 kHz
10.	75M	AM	3900 - 4000 kHz
11.	60M	AM	4750 - 5060 kHz
12.	60M	USB	5060 - 5450 kHz
13.	49M	AM	5900 - 6200 kHz
14.	40M	LSB	7000 - 7200 kHz
15.	41M	AM	7200 - 7600 kHz
16.	31M	AM	9400 - 9900 kHz

Figure 5: Jstvro team (and Elektor readers), can we have VLF, 4 meters (70 MHz) and 2 meters (144 MHz) as well, please?

Welcome to Radio

Once you know some of the quirks of radio wave propagation [2], the tuning and listening experience from the ATS25 max-Decoder is level with many "big box" receivers. I found that operating the ATS25 with the supplied stylus requires getting used to and initially, I mislaid my stylus and was forced to select some of the ultra-small fields and icons on the touch screen with my fingertip, but that too worked. Another slight problem was keeping the radio steady and positioned at a slight angle with the rather stiff (semi-rigid) coax cable from the loop antenna connected at the back. I solved this by placing the radio on a smartphone holder. Yes, I am used to SW receivers weighing 20 kg and up (Collins, Hallicrafters, you know).

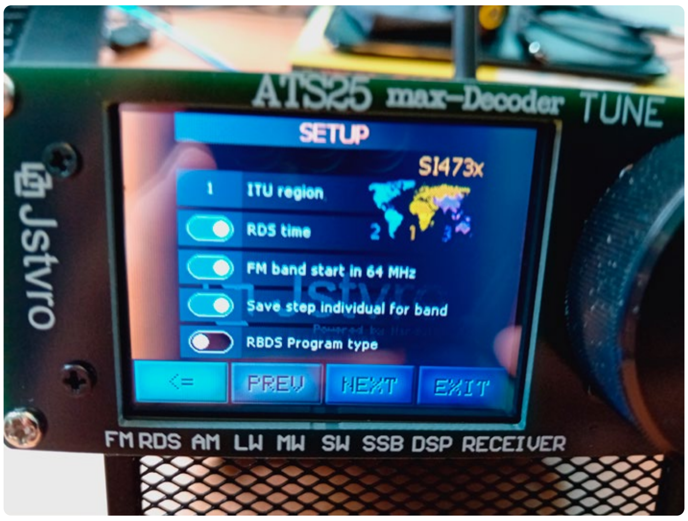
I was able to listen in on many local QSOs on 80-meters SSB (3.5 MHz), which is really a "chat" band with daytime distances of up to 300 kms between stations. On 20 meters (14 MHz) I was able to receive old school RTTY and even some SSTV which I did not attempt to decode on my PC, though. The 40-m band (7 MHz) is great to exercise the automatic CW decoding abilities of the ATS25 max-Decoder, provided the radio is linked to your local Wi-Fi network.

I found the automatic gain control (AGC) and RF attenuation functions to be slightly inconsistent and unable to ward off high-power stations in the bands, as with some Chinese broadcasts.

Getting Online with Wi-Fi

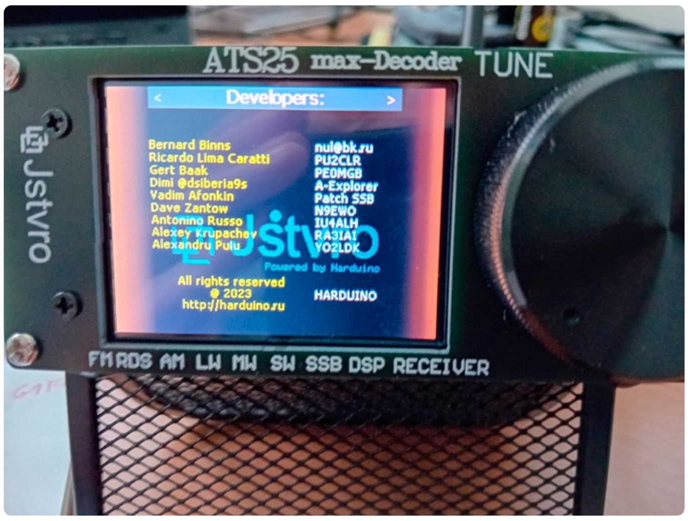
Many of the advanced digital decoding functions like FT8 and CW, but also NTP timekeeping, rely partly or wholly on connection to the Internet. The ATS25 max-Decoder is easy to put online, but the process is subject to improvement as far as the instructions are concerned. The Wi-Fi connectivity, although crucial to the operation of the radio, is explained rather vaguely on the last (!) page of the Manual. Several steps are not mentioned.

The most reassuring thing to know is that Elektor supplies the ATS25 max-Decoder complete with the registered product key, so you do not have to apply for one with the makers. The product key is shown prominently when you switch on the radio. I suggest you write it down and keep it in safe place. Next, the challenge is to find the menu where the radio can log on to your Wi-Fi network. Duh, that's on one of the last pages of the SETUP menu! After switching on the radio, press *NEXT*, then *SETUP*, then *NEXT* (about 15 times!)



1 ITU region
RDS time
FM band start in 64 MHz
Save step individual for band
RBDS Program type

Figure 6: Menus...the ATS25 max-Decoder has pages and pages of them.



Developers:

Bernard Binns	nul@bk.ru
Ricardo Lima Caratti	PU2CLR
Gert Baak	PE0MGB
Dimi @dsiberia9s	A-Explorer
Vadim Afonkin	Patch SSB
Dave Zantow	N9EWO
Antonino Russo	IU4ALH
Alexey Krupachev	RA3IAI
Alexandru Pulu	YO2LDK

All rights reserved
@ 2023
http://harduino.ru
HARDUINO

Figure 7: Credit where credit is due. 73 to the developers!

until you see the *BT/Wi-Fi* menu. There, switch on: *Wireless on* and then *Select AP*. The next step is to select the desired Wi-Fi network and enter the password. My Wi-Fi scan failed a number of times, the software hung up, and I was forced to switch the radio off and on again. Exit the menu and Save your settings. Back in radio mode, touch the Wi-Fi symbol on the screen (it's tiny!) and it will light up green. You are now connected and ready to benefit from the brilliant online utilities created by the Jstvro group specially for the ATS25 max-Decoder.

With your registration key, you are also eligible for software updates. My radio indicates it's running version "Air 4.17 Beta" and I wasn't able to select *Bluetooth* in the *BT/Wi-Fi* menu, so I guess this requires an update or an extension of some sort in the near future.

Likes and Likeables

The "Retro" tuning scale and S-Meter, as well as the FT8 mode and the CW decoder soon became my favorite features of this ESP-powered radio with its incredible number of options and features.

The sheer portability of the radio plus its rechargeable battery which lasts many hours make it a great choice for outdoor use well away from man-made noise. Tuning and operation of the ATS25 max-Decoder are a delight once you've got the hang of using the supplied stylus to pick the right menus. The sound quality from the internal speaker is good but I prefer to plug in my headphones — borrowed from my Samsung smartphone set. They work better and will not annoy your housemates.

For this review, I did not have time to set up an IDE on my PC, run PUTTY at 115,200 baud and read decoded messages and station information on my PC. Nor was I able to explore the WiFi Access Point functionality offered by the radio.

What's missing? Not much and certainly feasible things in the future as the radio is a software powerhouse really. I would have loved:

- VLF band reception, say 50 kHz to 200 kHz for DCF/MSF timekeeping stations, lowfers, etc.
- 2-m band reception (144-146/148 MHz)
- VHF Airband reception (not allowed in all countries)
- NBFM for the CB band (40 channels, European)
- A little less Italian in the (English) Manual
- Better guidance for the Wi-Fi setup

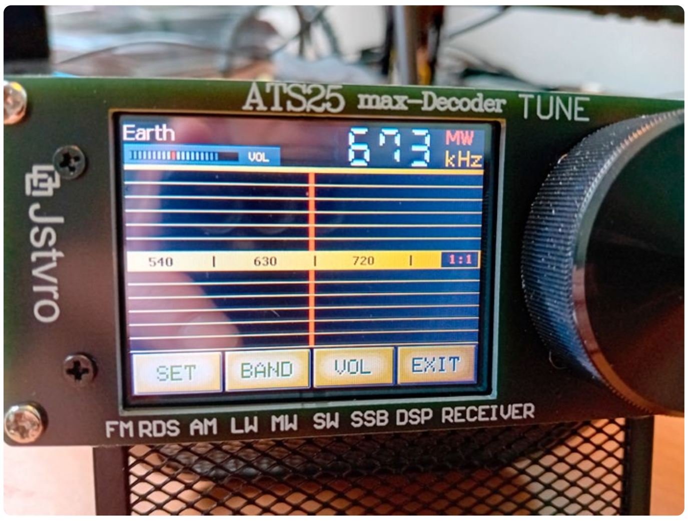


Figure 8: Totally charming: the "retro" mode for the frequency scale!

Now, since the ATS25 max-Decoder is ESP32 powered and Elektor readers are both numerous and knowledgeable as far as coding for this platform goes, I'm confident all of the above points can be fulfilled if not overcome "S9" (with great success).

The ATS25 max-Decoder with preregistered product key is available as an Elektor Labs Selected product from the Elektor Store.

240348-01

Related Products

- **ATS25 max-Decoder FM/HF Full-Band DSP Receiver**
  www.elektor.com/20869
- **YouLoop Portable Passive Magnetic Loop Antenna for HF and VHF**
  www.elektor.com/20681
- **The RF & Communications Collection (USB Stick)**
  www.elektor.com/20825

WEB LINKS

[1] Si4735: https://www.skyworksinc.com/en/products/audio-and-radio/si4734-35-am-fm-sw-lw-radio-receivers/si4735
[2] Radio wave propagation: https://en.wikipedia.org/wiki/Radio_propagation

elektor store

www.elektor.com

The Elektor Store

Never expensive, always surprising

The Elektor Store developed from the community store for Elektor's own products, such as books, magazines, kits and modules, into a mature web store that offers great value for surprising electronics. We offer the products that we ourselves are enthusiastic about or that we simply want to try out. If you have a nice suggestion, we are here: sale@elektor.com.

FNIRSI HRM-10 Battery Voltage Internal Resistance Tester

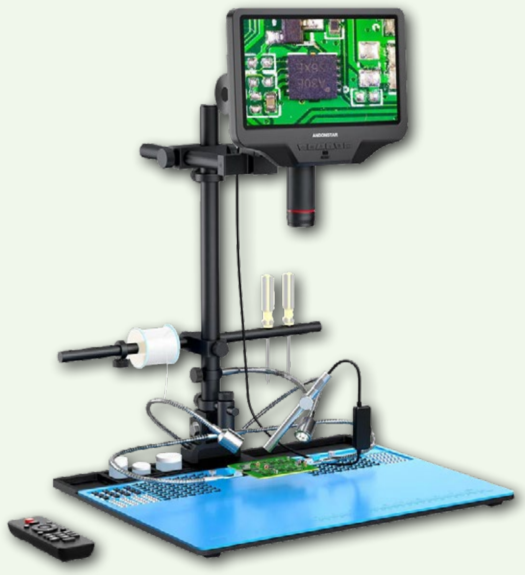
The FNIRSI HRM-10 is a portable, high-precision internal resistance meter. This device offers true four-wire measurement and is designed for both accuracy and ease of use. It automatically measures internal resistance and voltage values simultaneously, displaying the results clearly on its HD color screen. Users have the option to manually adjust voltage and resistance ranges to suit their needs. The device also includes a sorting mode that automatically filters the good and bad batteries based on user-set thresholds. Additionally, it supports the storage of historical data and allows for exporting measurement records in table format.

Price: ~~€59.95~~

Special Price: €49.95

www.elektor.com/20947

Andonstar AD409 Max-ES 10.1" Digital Microscope (incl. Endoscope)



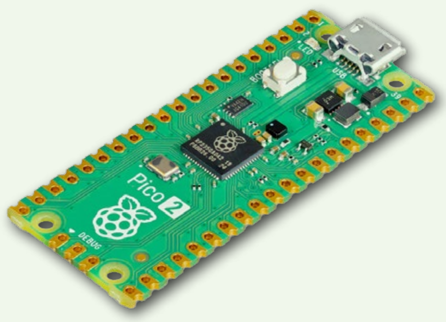
The Andonstar AD409 Max-ES boosts a high-quality metal lens and a unique UV filter design. Crafted from top-tier industrial-grade materials, it delivers unmatched precision and durability, ensuring a reliable product experience. The UV filter positioned in front of the metal lens blocks soldering heat, smoke, and dust, safeguarding the lens and making it perfect for soldering and maintenance professionals.

Price: ~~€499.00~~

Special Price: €349.00

www.elektor.com/20852

Raspberry Pi Pico 2



Price: €5.95

www.elektor.com/20950

Raspberry Pi 5 (2 GB RAM)

Price: €54.95

www.elektor.com/20951

KiCad Like A Pro (Bundle)

Price: ~~€94.95~~

Special Price: €74.95

www.elektor.com/20942

The Book of 555 Timer Projects

Price: €29.95

Member Price: €26.96

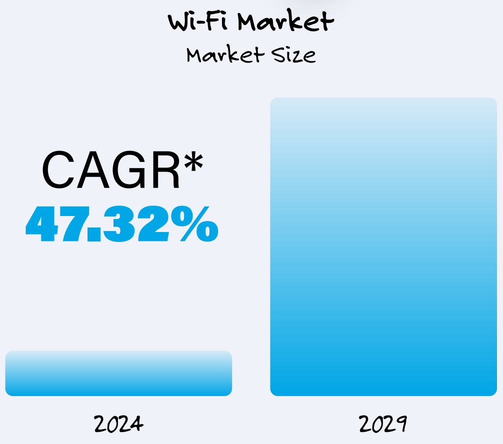
www.elektor.com/20948

Elektor infographic

Wireless and Communication

The Future of Wi-Fi 7

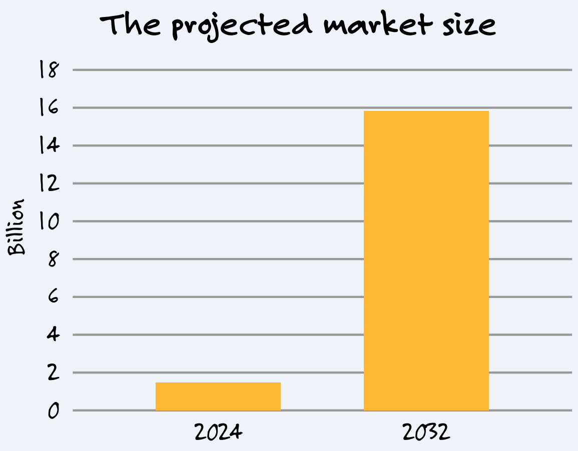
The introduction of Wi-Fi 7, the next-generation wireless technology, is expected to bring major improvements [1], including Multi-Link Operation (MLO), which enhances spectrum efficiency and tackles interference problems in crowded environments. However, the adoption of Wi-Fi 7 presents challenges, particularly its high cost compared to Wi-Fi 6, which might limit its usage in markets with a lower average selling price (ASP). Furthermore, the launch of Wi-Fi 7 will lead to a more varied Wi-Fi infrastructure ecosystem, resulting in a more diverse market between regions with and without unlicensed 6 GHz access. This diversification will lead to a wider range of products and increased customization to meet the specific needs of different markets.



(Source: Mordor Intelligence [2])

The Impact of AI on Wireless Networks and Telecom

Artificial intelligence (AI) is being incorporated into wireless networks and telecommunications, bringing significant changes to the industry [3]. This integration improves network efficiency through self-optimizing networks (SON), which can adapt in real-time to changing conditions. Customer engagement is also enhanced with advanced deep neural networks that enable human-like tasks and interactions. AI-driven software-defined networks (SDN) and network function virtualization (NFV) diversify network traffic, allowing for more sophisticated services and customer interactions. This synergy is shaping a new era in telecommunications, offering seamless connectivity and improved user experiences.



Key Drivers

- Increasing demand for data services
- Rollout of 5G networks
- Proliferation of IoT devices
- Need for network automation

Challenges

- **Data privacy and security**
  Compliance with GDPR, CCPA
  Need for robust encryption and authentication
- **Standardization and interoperability**
  Common protocols and interfaces required

33.68%

CAGR*

Anticipated growth rate from 2024 to 2032 [4].

*Compound annual growth rate

From Data Processing to Analytics

Edge computing has become a game-changer recently, reshaping data processing and analytics. It is increasingly influential across industries, with the market size expected to grow [5]. Security and privacy concerns are driving innovation in encryption and access control, while edge-to-cloud integration optimizes resource utilization.

$21.41 Billion

The expected growth of the market size in 2024.

$216.76 Billion

The expected growth of the market size in 2032 [5].

The Shift Toward Multi-Vendor Interoperability

Open RAN (radio access network) is an approach that promotes interoperability between cellular network equipment from different vendors. The O-RAN Alliance aims to transform the traditional monolithic, hardware-centric RAN design into an open, virtualized, and fully interoperable architecture [6]. This allows wireless network equipment providers to focus on specific software components rather than building entire RAN systems, enabling wireless service providers to mix and match components from multiple vendors.

The market conditions will continue to be challenging in the near term, but the long-term outlook is positive. According to Dell'Oro Group [7], Open RAN is expected to represent 20 to 30% of global RAN revenues by 2028, a substantial increase from the 7 to 10% projected for 2024.

240232-01

WEB LINKS

[1] ABI Research, "Wi-Fi 7 and Standard Power 6 GHz to Boost Wi-Fi Infrastructure Market Rebound in 2024 With 12.3% Year-on-Year Shipment Growth," Dec 2023: https://tinyurl.com/wi-fi-7-6-ghz

[2] Mordor Intelligence, "Wi-fi 7 Market Size & Share Analysis," 2023: https://mordorintelligence.com/industry-reports/wi-fi-7-market

[3] E. Jordan, "Part 1: The Synergy of AI and Wireless Technologies," 2024: https://tinyurl.com/the-synergy-of-ai

[4] Market Research Future, "AI in Telecommunication Market," June 2024: https://marketresearchfuture.com/reports/ai-in-telecommunication-market-6803

[5] Fortune Business Insights, "Edge Computing Market," May 2024: https://fortunebusinessinsights.com/edge-computing-market-103760

[6] MathWorks, "What Is Open RAN (O-RAN)?": https://mathworks.com/discovery/o-ran.html

[7] Dell'Oro Group, "Open RAN Now Projected to Comprise 20 Percent to 30 Percent of Global RAN by 2028," Feb 2024: https://tinyurl.com/challenging-open-ran

Elettronica In
www.elettronicain.it

ESP32-Based ArtNet to DMX Converter

Upgrade Your Legacy DMX Device

By Emanuele Signoretta (Italy)

The ArtNet protocol has brought the DMX standard to the network. The new ArtNet devices are, of course, backward-compatible with the DMX standard, but not vice versa. What to do then, with a legacy DMX device? The interface presented in this article allows you to upgrade it, granting compatibility with this new standard.

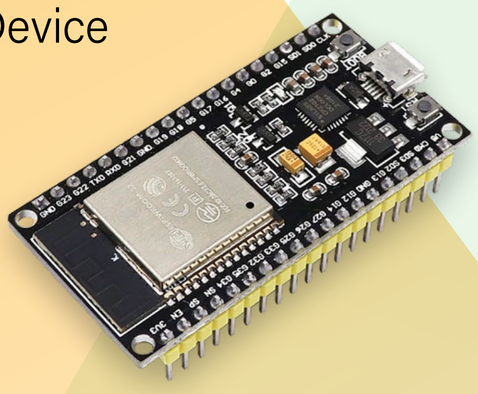


Figure 1: The ESP32 module at the heart of the project. (Source: Elettronica In)

The idea for this project came almost spontaneously, thinking about the backward compatibility of the ArtNet standard. A *universe* in both the ArtNet and DMX consists of 512 channels, and the values of each vary from 0 to 255. Most of the latest generation of remote-controlled devices (RDMs) used in the entertainment industry are compatible with both protocols, but how to deal with those that still do their job excellently but receive only the DMX signal? The simplest idea is to convert the signal externally from ArtNet to DMX through a wireless, compact, reconfigurable device.

Hardware

The whole project is built around a 38-pin version of the ESP32 module, shown in **Figure 1**, while the simple circuit diagram is illustrated in **Figure 2**. This board was chosen because of its high computing capacity and on-board memory, low cost, small package,

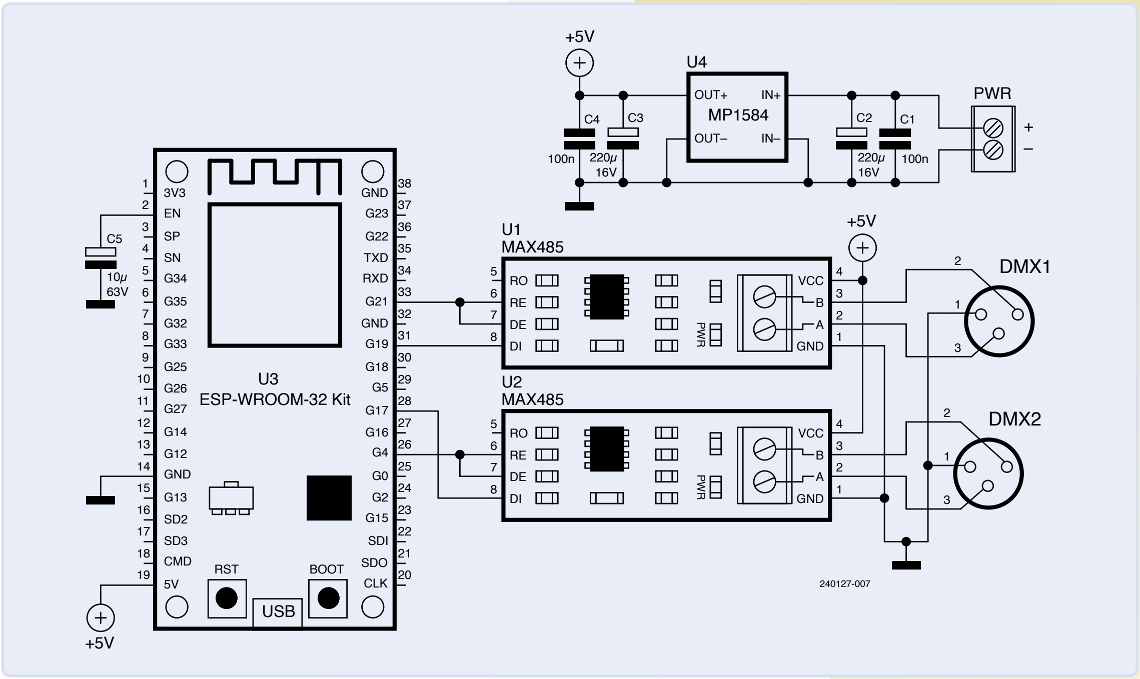


Figure 2: The simple schematic diagram of the converter.

and large community. I adopted a 5 V step-down DC-DC converter for the power supply.

For the DMX section, two TTL-DMX converters, complete with connectors, can be found online (**Figure 3**), but at a very high cost compared with that of the other components. We therefore decided to use two TTL-to-RS-485 converters (**Figure 4**) and two female 3-pin XLR panel connectors. Finally, we will need a DC, barrel-type panel connector and a plastic enclosure to close it all up.

Connection Diagram

I originally made a PCB for the project, with the 30-pin version of the ESP32 embedded (see **Figure 5**), but the extreme simplicity of the connections allows for implementation even without a printed circuit board, using the enhanced 38-pin module instead. In fact, the connection diagram is easy to interpret and is available in **Figure 6**. Let's start with the power section: the panel-mounted DC plug is connected to the input of the DC-DC converter. The 5 V output of the latter is connected to VIN on the ESP32 (Pin 19) as well as the VCC pins of both RS-485 modules. The same connection should be made for the GND pins of the three boards, which will be connected to the output ground of the DC-DC converter.

As for the signals, each module has three input pins: The RE and DE pins of module A should be connected to IO4 (Pin 26) of the ESP32 module, while the DI pin of the A transceiver should be connected to IO17 (Pin 28) of the ESP32. The output of each transceiver should be connected to an XLR connector according to the following diagram: Pins A and B of the MAX485 should be connected to pins 3 and 2 of the connector, respectively. Pin 1 of the connector can be connected to ground.

Sketch

The sketch for this project is based on the *ArtNetWiFiNeopixel* GitHub sketch by rstephan [1] and Mitch Weisbord's *DMX_Write* [2]. You can download this project's sketch directly from this project's GitHub repository [3].

As always, we will proceed to analyze it as a whole, divided into several listings, and dwell

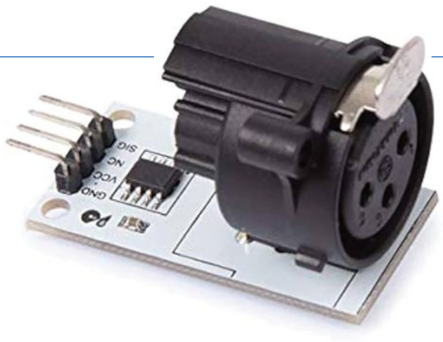


Figure 3: Ready-made TTL-DMX converters available on the market. (Source: Elettronica In)

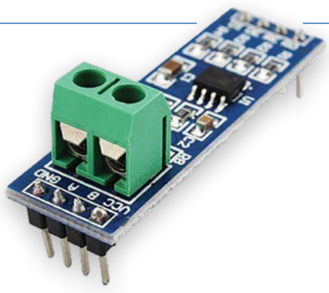


Figure 4: The TTL-to-RS-485 converter used in this project. (Source: Elettronica In)

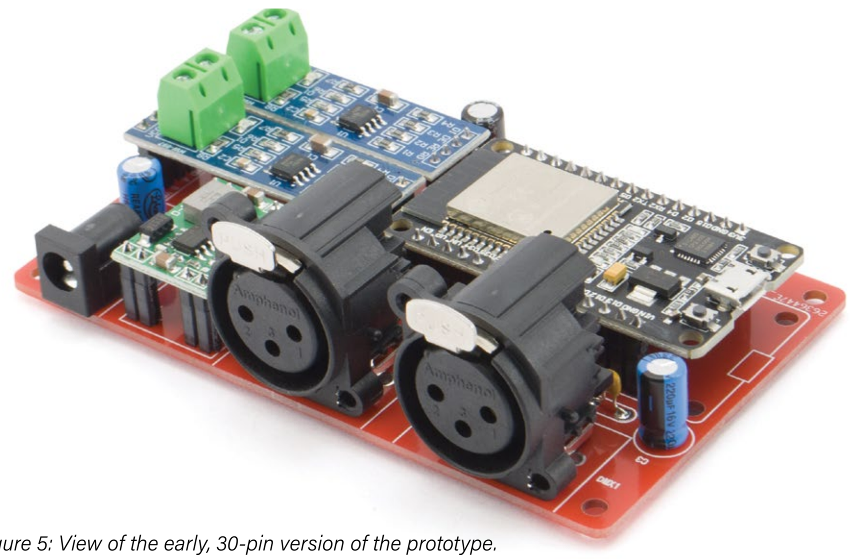


Figure 5: View of the early, 30-pin version of the prototype.

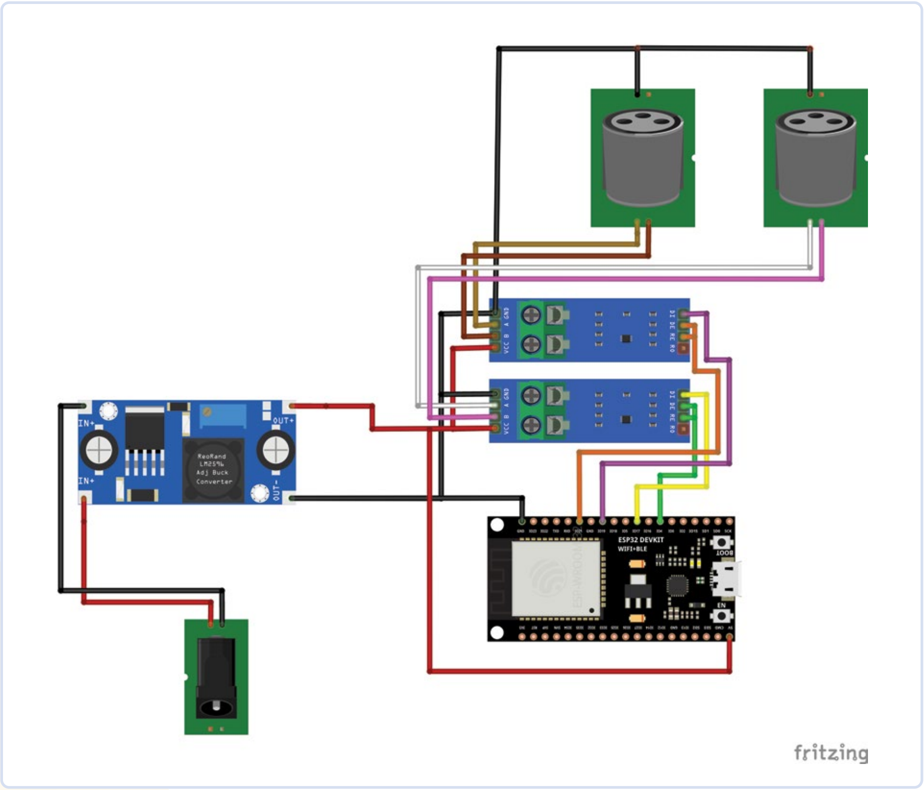


Figure 6: The project's wiring diagram.

Listing 1: Library inclusion and DHCP configuration.

```
#include <Arduino.h>
#include <esp_dmx.h>
#include "ArtnetWifi.h"
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
//#define DHCP_DISABLED
#ifdef DHCP_DISABLED
IPAddress local_IP(192, 168, 1, 154);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 255, 0);
IPAddress primaryDNS(192, 168, 1, 1); //optional
IPAddress secondaryDNS(1, 1, 1, 1); //optional
#endi
```

Table 1:
ESP32 to RS-485-A Module connections

ESP32 Pin	MAX485 Pin
4	DE
17	DI
4	RE

Table 2:
ESP32 to RS485-B Module connections

ESP32 Pin	MAX485 Pin
21	DE
19	DI
21	RE

on some highlights of the code. **Listing 1** attends to the inclusion of libraries and DHCP configuration. Let's have a more in-depth look. Several variables that are useful for compiling the sketch are included in the *Arduino.h* library, along with the inclusion of several other libraries containing basic functionality classes, such as `stdlib`, `stdio`, `string`, etc.

The *esp_dmx.h* library carries the whole set of instructions to handle sending signals to the TTL-to-RS-485 module.

Then we have the *ArtnetWifi.h* and *WiFi.h* libraries for receiving ArtNet packets and managing the connection to the access point, respectively. The *ESPmDNS.h*, *WiFiUdp.h*, and *ArduinoOTA.h* libraries, on the other hand, are functional for OTA firmware loading and setting the device name within the network, using the mDNS protocol.

Then follows a `#define DHCP_DISABLED`: through this directive we are going to choose whether to set a static IP address or to delegate everything to the DHCP server (commenting out the line). If we decide to use a static IP in the following lines of code, it is necessary to set all related constants to the desired IP, gateway and subnet to which it belongs. You can, furthermore, choose the DNS servers. If we choose to use a dynamic IP, it will still be useful (but not essential) to know the IP address of the ESP32.

We'll explain more about that later. **Listing 2** comprises the remaining part of the preprocessor. The objects related to the *WiFiUDP* and *ArtnetWifi* libraries are initially created. Next, in the string constants `ssid` and `password`, the name of the Wi-Fi network to which the ESP connects, as well as its password, are defined. The ESP pins to which both RS-485 transceivers will be connected are shown in **Table 1** and **Table 2**.

For each of the modules, we've set the receive pin to 16, although these are not actually connected and of no use in our design. Moving on, the hardware is set with the definition of the ESP TTL ports: We will use ports 1 and 2 for universe A and B, respectively. The data for the 2 DMX universes will be encapsulated in two `byte` arrays, `dataA` and `dataB`, whose sizes are defined in `DMX_PACKET_SIZE`. `DMX_PACKET_SIZE` is defined in the *esp_dmx.h* library and has a value of 513. Finally, all variables used for processing ArtNet packets are defined. We pay special attention to the `startUniverse` constant: this, in fact, indicates from which universe we want to start considering data.

It should be modified in case we intend to create an installation with multiple receivers from different universes. It may also need to be modified in cases where the program that sends packets has a numbering starting with 1 instead of 0. Constants are then defined regarding the maximum number of universes, channels, and other variables used to monitor states and process loops.

Listing 3 contains all the instructions included within `setup()`. The serial port is first initialized at 115,200 bits per second to allow debugging of the board, such as to verify the IP obtained. Moving forward, if set, all parameters are configured to obtain a static IP and, if an error occurs, a message is sent out on the serial port. A loop connection to the network is then attempted and, if successful, the IP of the board is printed to the serial port. The hostname *ESP32-ArtNetto-DMX-Converter* is then set, which will be displayed in the *Board Manager* section of the various IDEs. This function is very useful for being able to recognize one device among many in a "fleet." The OTA firmware loading management is then initialized, and next comes the ArtNet library.

Finally, the ESP output pins and serial port that will be allocated to each transceiver are set, and the priority of the interrupts that will be used is then defined. **Listing 4** contains the whole part related to the `onArtNetFrame()` function. First, it is checked that the value of each of the received universes belongs to the universes of our interest. If favorable, the flag of the corresponding cell of the array is set to 1. It proceeds to verify that all the required universes are received; otherwise, it exits the function. Next, we verify that each of the received channels belongs to universe *A* or universe *B* and will be included into the corresponding array.

Next, the data from each universe is inserted into the respective sent buffers and waits for the transmission to be complete. Finally, the flags related to the received universes are reset. **Listing 5** is the shortest and contains only the `loop()` function. Within the loop, it is verified that the ESP is correctly connected to the access point, and if so, ArtNet packets are received and processed.

Component List

Capacitors
C1, C4 = 100 nF, ceramic
C2, C3 = 220 µF, 16 V, electrolytic
C5 = 10 µF, 63 V, electrolytic

Modules
U1, U2 = TTL-to-RS-485 converter
U3 = ESP32
U4 = MP1584 DC-DC step-down converter, 5 V

Miscellaneous
DMX1, DMX2 = XLR connector, female, PCB type
PWR = barrel connector, female, PCB-type
4× 2-pin strip connector, female, pitch 2.54 mm
4× 4-pin strip connector, female, pitch 2.54 mm
2× 15-pin strip connector, female, pitch 2.54 mm
4× 2-pin strip connector, male, pitch 2.54 m
PCB S1716 (102×61 mm)

Listing 2: “WiFiUDP” and “ArtnetWifi” libraries object creation.

```
WiFiUDP UdpSend;
ArtnetWifi artnet;
const char* ssid = "MyArtNetNetwork";
const char* password = "MyArtNetNetwork";

/* First, lets define the hardware pins that we are using with our ESP32. We need to define which pin is
transmitting data and which pin is receiving data. DMX circuits also often need to be told when we are
transmitting and when we are receiving data. We can do this by defining an enable pin.*/

int transmitPinA = 17;
int receivePinA = 16; //Not connected
int enablePinA = 4;
int transmitPinB = 21;
int receivePinB = 16; //Not connected
int enablePinB = 19;

/* Make sure to double-check that these pins are compatible with your ESP32! Some ESP32s, such as the ESP32-
WROVER series, do not allow you to read or write data on pins 16 or 17, so it’s always good to read the
manuals.*/
/* Next, let’s decide which DMX port to use. The ESP32 has either 2 or 3 ports. Port 0 is typically used to
transmit serial data back to your Serial Monitor, so we shouldn’t use that port. Let’s use port 1! */

dmx_port_t dmxPortA = 1;
dmx_port_t dmxPortB = 2;

/* Now we want somewhere to store our DMX data. Since a single packet of DMX data can be up to 513 bytes
long, we want our array to be at least that long. This library knows that the max DMX packet size is 513, so
we can fill in the array size with DMX_PACKET_SIZE. */

byte dataA[DMX_PACKET_SIZE];
byte dataB[DMX_PACKET_SIZE];

//ArtNet settings const int startUniverse = 0;
// CHANGE FOR YOUR SETUP most software this is 1,
//some software send out artnet first universe as 0.

const int maxUniverses = 2;
const int numberOfChannels = 1024;
bool universesReceived[maxUniverses];
bool sendFrame = 1;
int previousDataLength = 0;
```

Listing 3: setup() function.

```
void setup() {

      /* Start the serial connection back to the computer so that we can log messages to the Serial Monitor.
      Let’s set the baud rate to 115200. */

      Serial.begin(115200);
      // Setup wifi
      WiFi.mode(WIFI_STA);
#ifdef DHCP_DISABLED
      //Comment to use DHCP instead of static IP
     if (!WiFi.config(local_IP, gateway, subnet, primaryDNS, secondaryDNS)) {
         Serial.println("STA Failed to configure");
     }
#endif
    WiFi.begin(ssid, password);
     delay(1000);
    Serial.println("\nConnecting");
    while (WiFi.status() != WL_CONNECTED) {
          Serial.print(".");
          delay(100);
     }
     Serial.println("\nConnected to the WiFi network");
     Serial.print("Local ESP32 IP: ");
     Serial.println(WiFi.localIP());
     // Port defaults to 3232
     // ArduinoOTA.setPort(3232);
     // Hostname defaults to esp3232-[MAC]
     ArduinoOTA.setHostname("ESP32-ArtNet-to-DMX-Converter");
     // No authentication by default
     // ArduinoOTA.setPassword("admin");
     // Password can be set with it’s md5 value as well
     // MD5(admin) = 21232f297a57a5a743894a0e4a801fc3
     // ArduinoOTA.setPasswordHash("21232f297a57a5a743894a0e4a801fc3");
     ArduinoOTA
          .onStart([]() {
            String type;
            if (ArduinoOTA.getCommand() == U_FLASH)
              type = "sketch";
            else // U_SPIFFS
              type = "filesystem";
            // NOTE: if updating SPIFFS this would be the
            // place to unmount SPIFFS using SPIFFS.end()
            Serial.println("Start updating " + type);
      })
      .onEnd([]() {
         Serial.println("\nEnd");
      })
      .onProgress([](unsigned int progress, unsigned int total) {
        Serial.printf("Progress: %u%%\r", (progress / (total / 100)));
      })
      .onError([](ota_error_t error) {
        Serial.printf("Error[%u]: ", error);
        if (error == OTA_AUTH_ERROR)
          Serial.println("Auth Failed");
        else if (error == OTA_BEGIN_ERROR)
          Serial.println("Begin Failed");
         else if (error == OTA_CONNECT_ERROR)
        Serial.println("Connect Failed");
      else if (error == OTA_RECEIVE_ERROR)
        Serial.println("Receive Failed");
      else if (error == OTA_END_ERROR)
        Serial.println("End Failed");
    });
  ArduinoOTA.begin();
  artnet.setArtDmxCallback(onArtNetFrame);
  artnet.begin("ESP32-ArtNet-to-DMX-Converter");
  /* Set the DMX hardware pins to the pins that we want to use. */
  dmx_set_pin(dmxPortA, transmitPinA, receivePinA, enablePinA);
  dmx_set_pin(dmxPortB, transmitPinB, receivePinB, enablePinB);

  /* Now we can install the DMX driver! We'll tell it which DMX port to use and which interrupt priority
  it should have. If you aren't sure which interrupt priority to use, you can use the macro DMX_DEFAULT_
  INTR_FLAG to set the interrupt to its default settings.*/

  dmx_driver_install(dmxPortA, DMX_DEFAULT_INTR_FLAGS);
  dmx_driver_install(dmxPortB, DMX_DEFAULT_INTR_FLAGS);
}
```

Listing 4: onArtNetFrame() function.

```
void onArtNetFrame(uint16_t universe, uint16_t numberOfChannels,
   uint8_t sequence, uint8_t* dmxData) {
   sendFrame = 1;
   // Store which universe has got in
   if ((universe - startUniverse) < maxUniverses)
     universesReceived[universe - startUniverse] = 1;
   for (int i = 0; i < maxUniverses; i++) {
     if (universesReceived[i] == 0) {
       //Serial.println("Broke");
       sendFrame = 0;
       break;
     }
   }
   // read universe and put into the right array of data
   for (int i = 0; i < numberOfChannels; i++) {
     if (universe == startUniverse)
       dataA[i + 1] = dmxData[i];
     else if (universe == startUniverse + 1)
       dataB[i + 1] = dmxData[i];
   }
   previousDataLength = numberOfChannels;
   dmx_write(dmxPortA, dataA, DMX_MAX_PACKET_SIZE);
   dmx_write(dmxPortB, dataB, DMX_MAX_PACKET_SIZE);
   dmx_send(dmxPortA, DMX_PACKET_SIZE);
   dmx_send(dmxPortB, DMX_PACKET_SIZE);
   dmx_wait_sent(dmxPortA, DMX_TIMEOUT_TICK);
   dmx_wait_sent(dmxPortB, DMX_TIMEOUT_TICK);
   // Reset universeReceived to 0
   memset(universesReceived, 0, maxUniverses);
}
```

Listing 5: loop() function.

```
void loop() {
   if ((WiFi.status() == WL_CONNECTED)) {
     artnet.read();
   }
 }
```

Assembly

It is time to think about mounting the converter. Just follow the connections previously shown in Figure 6. Remember that on the RS-485 transceivers, you will need to bridge the DE and RE pins. Next, prepare the XLR connectors by soldering the three wires (**Table 3**).

Drill the plastic box for each connector, drilling two holes for the screws and a larger one for its housing. Drill the last hole for the DC plug to which you will have previously soldered two wires. Now proceed to connect the wires from the DC plug to the input of the DC-DC converter. The output of the latter will be connected to the power supply of the ESP32 and the two transceivers, using jumpers and a pair of screw terminals. Load the firmware, close the box, and connect the power supply, then proceed with the configuration of the control software.

Software Configuration

Many software programs exist to manage ArtNet fixtures: open source, freemium, and paid. Among the free software (or free for non-commercial use) that we feel we should recommend is undoubtedly MagicQ, produced by Chamsys. In this case, we will just deal with the use of MagicQ. It is a cross-platform (Windows, Mac and Ubuntu) free downloadable program [4] that allows you to control up to 256 universes, manage pixel mapping and playback of HD video arranged on up to 8 layers.

We proceed by registering on the website and downloading the installer. Once the installation is done, we will have several applications of the suite: open *MagicQ PC* and wait for a screen to appear, as in **Figure 7**. Click on *Simple generic console*. In the new screen that comes up, we go to the menu and select *Setup   DMX I/O*. In the screen that will appear as displayed in **Figure 8**, we will be able to set the outputs of the different DMX universes. For the *Out Type* item we select *Art-Net* and choose *Art 0* as the *Out Uni* item. The critical part of the configuration comes when we have to choose the *Unicast* and *Unicast2* items.

These entries are used to actually indicate to which IP addresses we are going to transmit

Table 3:
RS485 Module to XLR connections

MAX485 Pin	DMX Signal	XLR Connector Pin
GND	Gnd/Shield	1
A	Data+	3
B	Data-	2

our packets. Suppose we are connected to a network with a 24-bit subnet (255.255.255.0) and therefore broadcast IP XXX.XXX.XXX.255, we can enter the latter address, and we will be sure that all devices in the network will receive packets. If, on the other hand, we want to avoid flooding all devices in the network by sending packets unnecessarily, we can directly set the IP of the affected device but, in this case, we recommend setting it to use a static IP.

We applied this configuration to the ArtNet 2 universe, which uses the same as Universe 1 as its data source, setting it in the *copy* column.

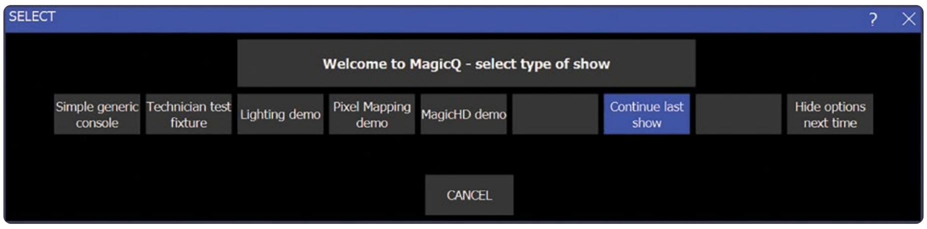
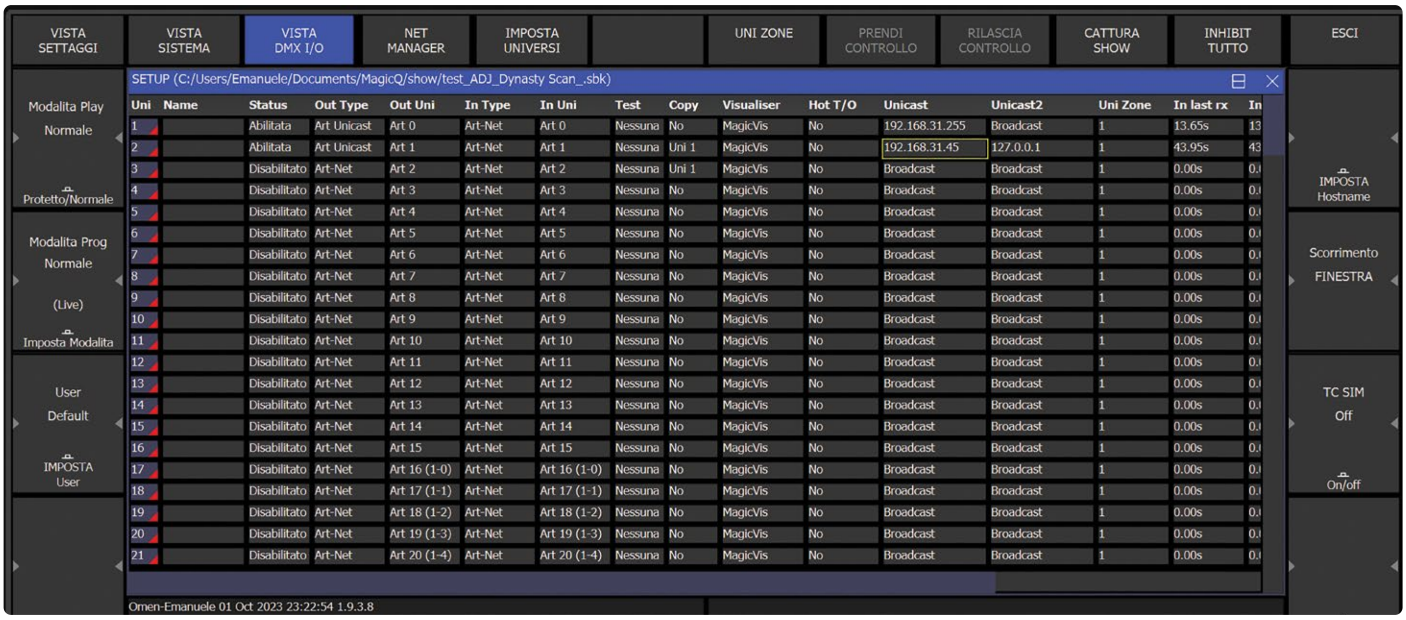


Figure 7: MagiQ introductory selection screenshot.

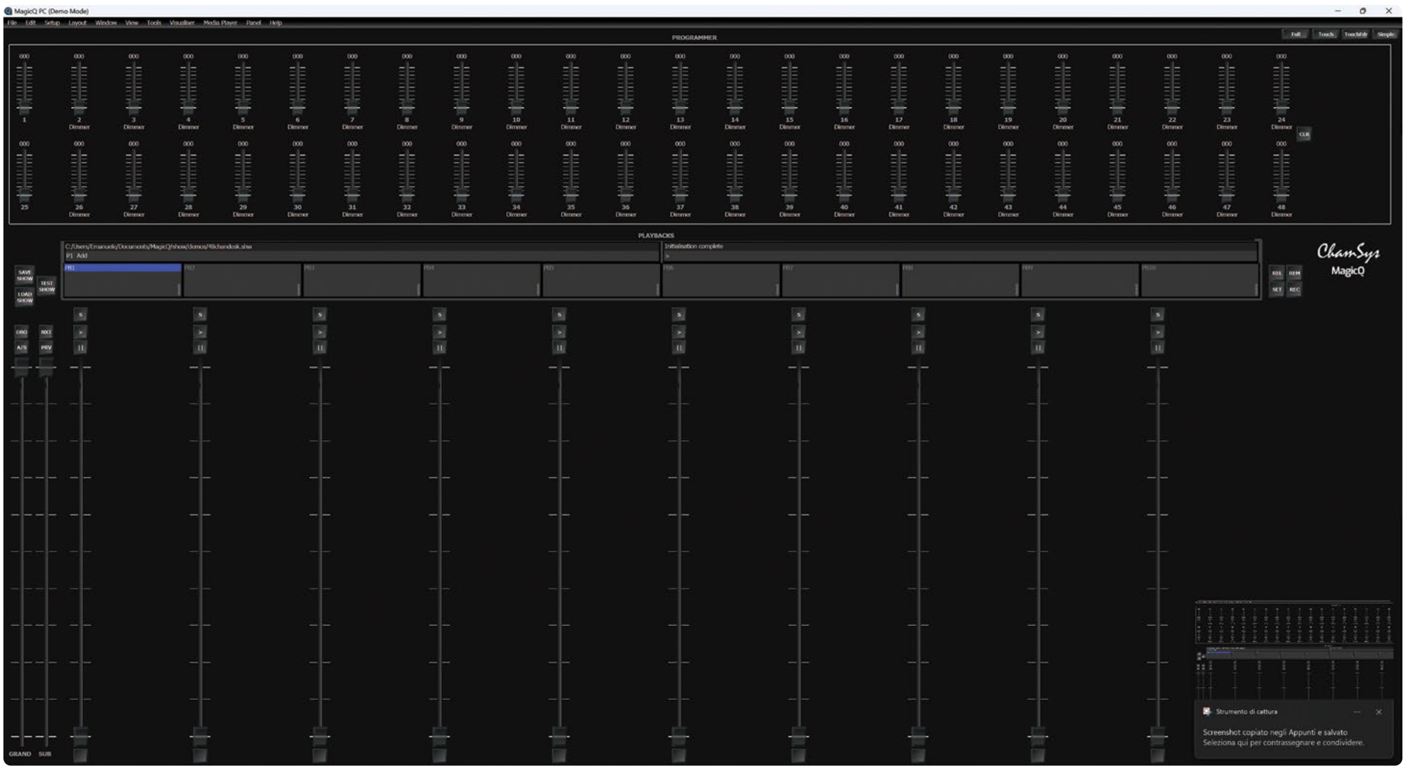


SETUP (C:/Users/Emanuele/Documents/MagicQ/show/test_ADJ_Dynasty Scan_.sbk)

Uni	Name	Status	Out Type	Out Uni	In Type	In Uni	Test	Copy	Visualiser	Hot T/O	Unicast	Unicast2	Uni Zone	In last rx	In
1		Abilitata	Art Unicast	Art 0	Art-Net	Art 0	Nessuna	No	MagicVis	No	192.168.31.255	Broadcast	1	13.65s	13
2		Abilitata	Art Unicast	Art 1	Art-Net	Art 1	Nessuna	Uni 1	MagicVis	No	192.168.31.45	127.0.0.1	1	43.95s	43
3		Disabilitato	Art-Net	Art 2	Art-Net	Art 2	Nessuna	Uni 1	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
4		Disabilitato	Art-Net	Art 3	Art-Net	Art 3	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
5		Disabilitato	Art-Net	Art 4	Art-Net	Art 4	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
6		Disabilitato	Art-Net	Art 5	Art-Net	Art 5	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
7		Disabilitato	Art-Net	Art 6	Art-Net	Art 6	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
8		Disabilitato	Art-Net	Art 7	Art-Net	Art 7	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
9		Disabilitato	Art-Net	Art 8	Art-Net	Art 8	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
10		Disabilitato	Art-Net	Art 9	Art-Net	Art 9	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
11		Disabilitato	Art-Net	Art 10	Art-Net	Art 10	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
12		Disabilitato	Art-Net	Art 11	Art-Net	Art 11	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
13		Disabilitato	Art-Net	Art 12	Art-Net	Art 12	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
14		Disabilitato	Art-Net	Art 13	Art-Net	Art 13	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
15		Disabilitato	Art-Net	Art 14	Art-Net	Art 14	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
16		Disabilitato	Art-Net	Art 15	Art-Net	Art 15	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
17		Disabilitato	Art-Net	Art 16 (1-0)	Art-Net	Art 16 (1-0)	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
18		Disabilitato	Art-Net	Art 17 (1-1)	Art-Net	Art 17 (1-1)	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
19		Disabilitato	Art-Net	Art 18 (1-2)	Art-Net	Art 18 (1-2)	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
20		Disabilitato	Art-Net	Art 19 (1-3)	Art-Net	Art 19 (1-3)	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.
21		Disabilitato	Art-Net	Art 20 (1-4)	Art-Net	Art 20 (1-4)	Nessuna	No	MagicVis	No	Broadcast	Broadcast	1	0.00s	0.

Omen-Emanuele 01 Oct 2023 23:22:54 1.9.3.8

Figure 8: ArtNet outputs configuration window.



C:/Users/Emanuele/Documents/MagicQ/show/demos/4lichandesk.shw

Initialisation complete

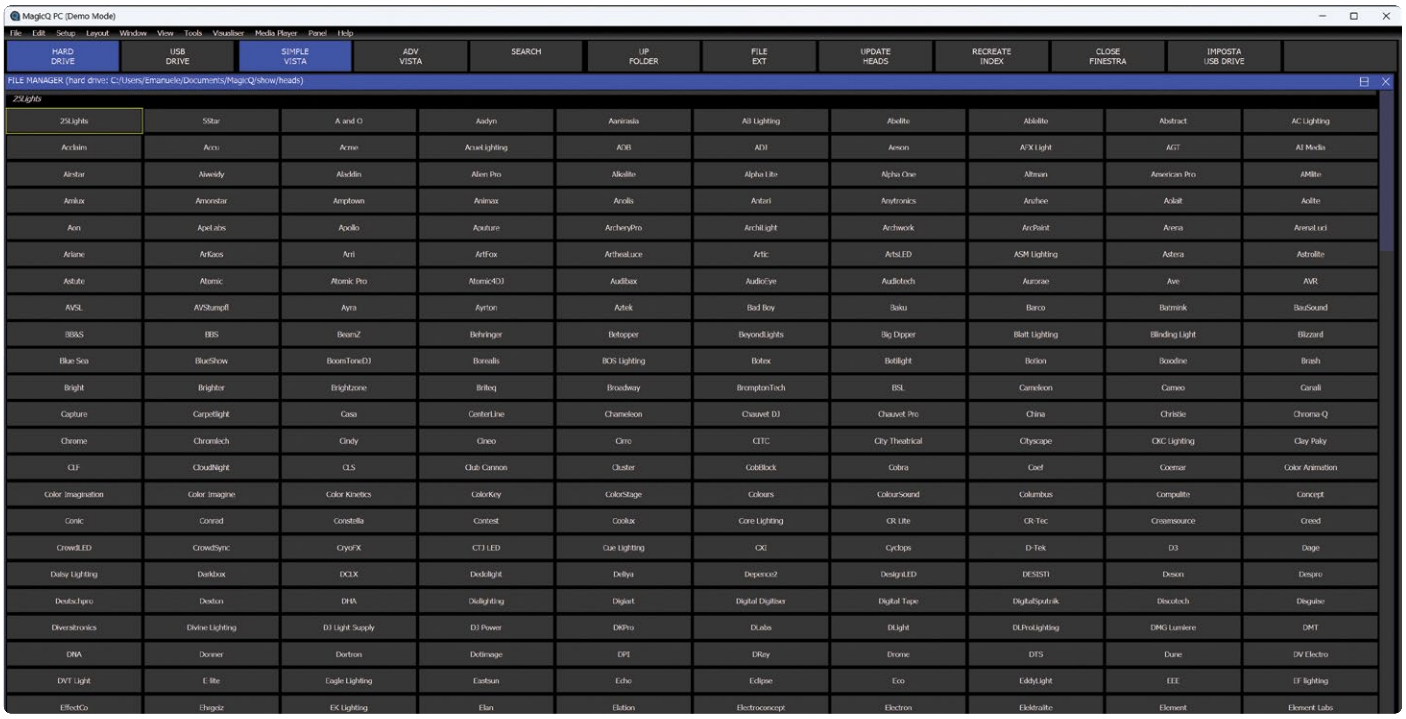
Strumento di cattura

Screenshot copiato negli Appunti e salvato
Seleziona qui per contrassegnare e condividere.

Figure 9: MagiQ sliders control panel.

You can see that we set the localhost address of the PC as in the *Unicast 2* field. We entered this configuration so that we could verify the actual packet transmission in debugging software such as ArtNetominator, downloadable at [5]. We did not enter this option for Universe 1 since, by using IP broadcast, packets automatically "go back" to the PC. Now, let's go with the cursor to the upper-right and click on *Single*, sliders will appear as shown in **Figure 9**.

To start packet transmission, we select *enable* and confirm. Let us now proceed to perform a simple fixture configuration, moving to the left about halfway up the screen and clicking on *Test show*. On the screen, we will see a long series of cells representing the manufacturers of the fixtures, as in **Figure 10**. We select the brand and all the pre-configured models available will appear.

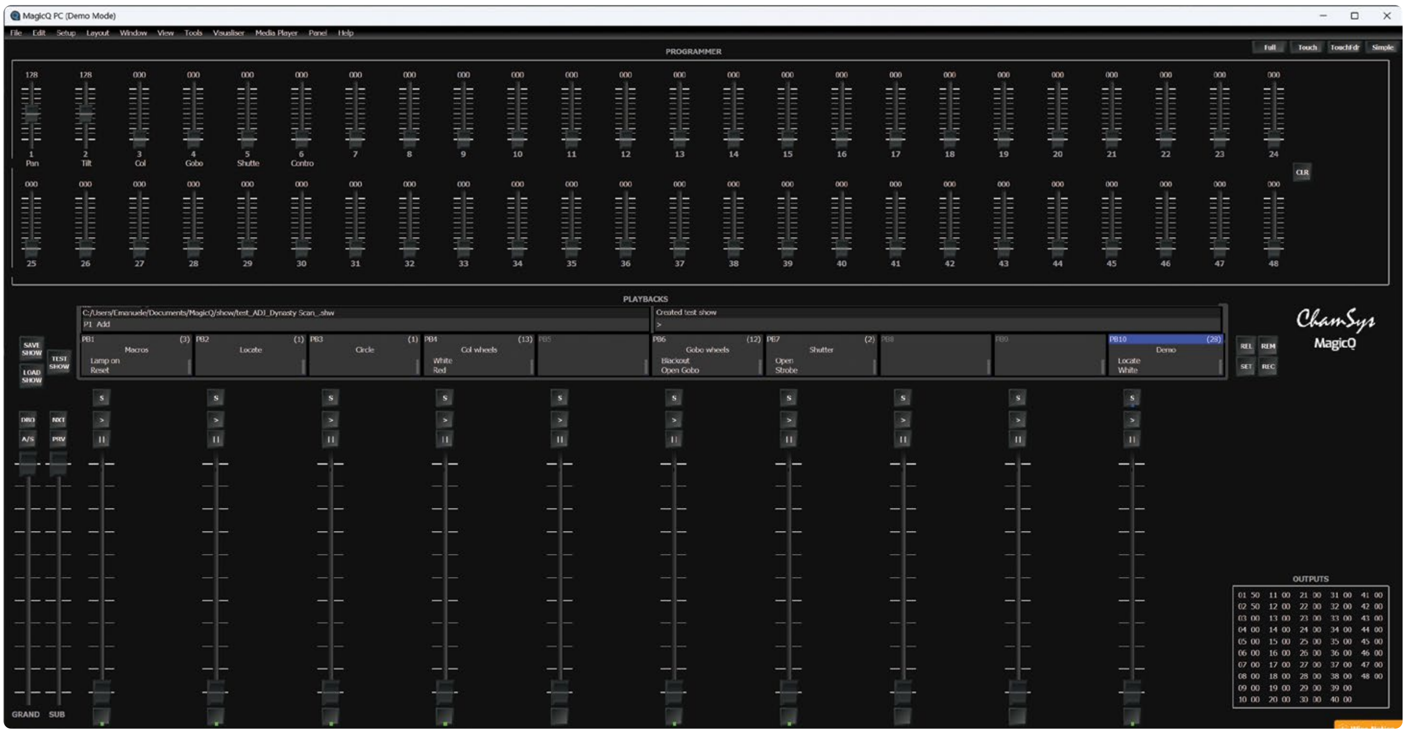


FILE MANAGER (hard drive: C:/Users/Emanuele/Documents/MagicQ/show/heads)

25Lights

25Lights	5Star	A and O	Aadyn	Aanirasia	AB Lighting	Abelite	Ablelite	Abstract	AC Lighting
Acclaim	Accu	Acme	AcueLighting	ADB	ADJ	Aeson	AFX Light	AGT	AI Media
Airstar	Aiweidy	Aladdin	Alien Pro	Alkalite	Alpha Lite	Alpha One	Altman	American Pro	AMlite
Amlux	Amonstar	Amptown	Animax	Anolis	Antari	Anytronics	Anzhee	Aolait	Aolite
Aon	ApeLabs	Apollo	Aputure	ArcheryPro	ArchiLight	Archwork	ArcPaint	Arena	ArenaLuci
Ariane	ArKaos	Arri	ArtFox	ArtheaLuce	Artic	ArtsLED	ASM Lighting	Astera	Astrolite
Astute	Atomic	Atomic Pro	Atomic4DJ	Audibax	AudioEye	Audiotech	Aurorae	Ave	AVR
AVSL	AVStumpfl	Ayra	Ayrton	Aztek	Bad Boy	Baku	Barco	Batmink	BauSound
BB&S	BBS	BeamZ	Behringer	Betopper	BeyondLights	Big Dipper	Blatt Lighting	Blinding Light	Blizzard
Blue Sea	BlueShow	BoomToneDJ	Borealis	BOS Lighting	Botex	Botilight	Botion	Boxdine	Brash
Bright	Brighter	Brightzone	Briteq	Broadway	BromptonTech	BSL	Cameleon	Cameo	Canali
Capture	Carpetlight	Casa	CenterLine	Chameleon	Chauvet DJ	Chauvet Pro	China	Christie	Chroma-Q
Chrome	Chromlech	Cindy	Cineo	Cirro	CITC	City Theatrical	Cityscape	CKC Lighting	Clay Paky
CLF	CloudNight	CLS	Club Cannon	Cluster	CobBlock	Cobra	Coef	Coemar	Color Animation
Color Imagination	Color Imagine	Color Kinetics	ColorKey	ColorStage	Colours	ColourSound	Columbus	Compulite	Concept
Conic	Conrad	Constella	Contest	Coolux	Core Lighting	CR Lite	CR-Tec	Creamsource	Creed
CrowdLED	CrowdSync	CryoFX	CTJ LED	Cue Lighting	CXI	Cyclops	D-Tek	D3	Dage
Daisy Lighting	Darkbox	DCLX	Dedolight	Dellya	Depence2	DesignLED	DESISTI	Deson	Despro
Deutschpro	Dexton	DHA	Dialighting	Digiart	Digital Digitiser	Digital Tape	DigitalSputnik	Discotech	Disguise
Diversitronics	Divine Lighting	DJ Light Supply	DJ Power	DKPro	DLabs	DLight	DLProLighting	DMG Lumiere	DMT
DNA	Donner	Dortron	Dotimage	DPI	DRey	Drome	DTS	Dune	DV Electro
DVT Light	E-lite	Eagle Lighting	Eastsun	Echo	Eclipse	Eco	EddyLight	EEE	EF lighting
EffectCo	Ehrgeiz	EK Lighting	Elan	Elation	Electroconcept	Electron	Elektralite	Element	Element Labs

Figure 10: Fixtures selection window, by brand and type.



```
C:/Users/Emanuele/Documents/MagicQ/show/test_ADJ_Dynasty Scan_.shw
P1 Add
Created test show
PB1 Macros (3)   Lamp on Reset
PB2 Locate (1)
PB3 Circle (1)
PB4 Col wheels (13)   White Red
PB6 Gobo wheels (12)   Blackout Open Gobo
PB7 Shutter (2)   Open Strobe
PB10 Demo (28)   Locate White
```

```
OUTPUTS
01 50   11 00   21 00   31 00   41 00
02 50   12 00   22 00   32 00   42 00
03 00   13 00   23 00   33 00   43 00
04 00   14 00   24 00   34 00   44 00
05 00   15 00   25 00   35 00   45 00
06 00   16 00   26 00   36 00   46 00
07 00   17 00   27 00   37 00   47 00
08 00   18 00   28 00   38 00   48 00
09 00   19 00   29 00   39 00
10 00   20 00   30 00   40 00
```

Figure 11: A configured fixture, with the available macros.

Having made our choice, we'll see a window like the one shown in **Figure 11**,in which we will either have individual sliders for basic commands or some macros that include running a mix of them over time. Let's test the operation, to make sure the devices and software are configured correctly. To take full advantage of the application's potential, we strongly recommend taking online training sessions that can be accessed for free at [6].

Some Final Considerations

We tested the operation of the system under several conditions and found different difficulties, depending on the access points used. In fact, it may happen that — depending on the access point — after a few hours the card goes into an error state and we have to reset it manually. It may happen, considering that this is a hobbyist solution, but by changing devices the problem is solved, allowing the card to work flawlessly for several days. Furthermore, we recommend using a dedicated network to avoid possible system crashes.

240127-01

Questions or Comments?

Do you have technical questions or comments about this article? Please feel free to contact the Elektor editorial team at editor@elektor.com.

59 Experiments with Arduino IDE
The Complete ESP32 Projects Guide
PDF
Dogan Ibrahim
elektor

WEB LINKS

[1] ArtNetWiFiNeopixel sketch : http://github.com/rstephan/ArtnetWifi
[2] Mitch Weisbord's "DMX_Write" sketch: http://github.com/someweisguy/esp_dmx
[3] Author's Github repository for this project: http://github.com/signorettae/ESP32-ArtNet-to-DMX
[4] MagiQ download website: https://chamsyslighting.com/pages/magicq-downloads
[5] ArtNetominator SW download page: https://lightjams.com/artnetominator
[6] Chamsys training webpage: https://chamsyslighting.com/pages/training-uk