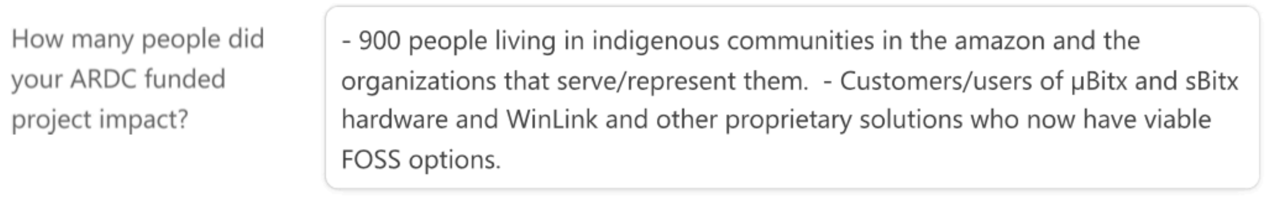
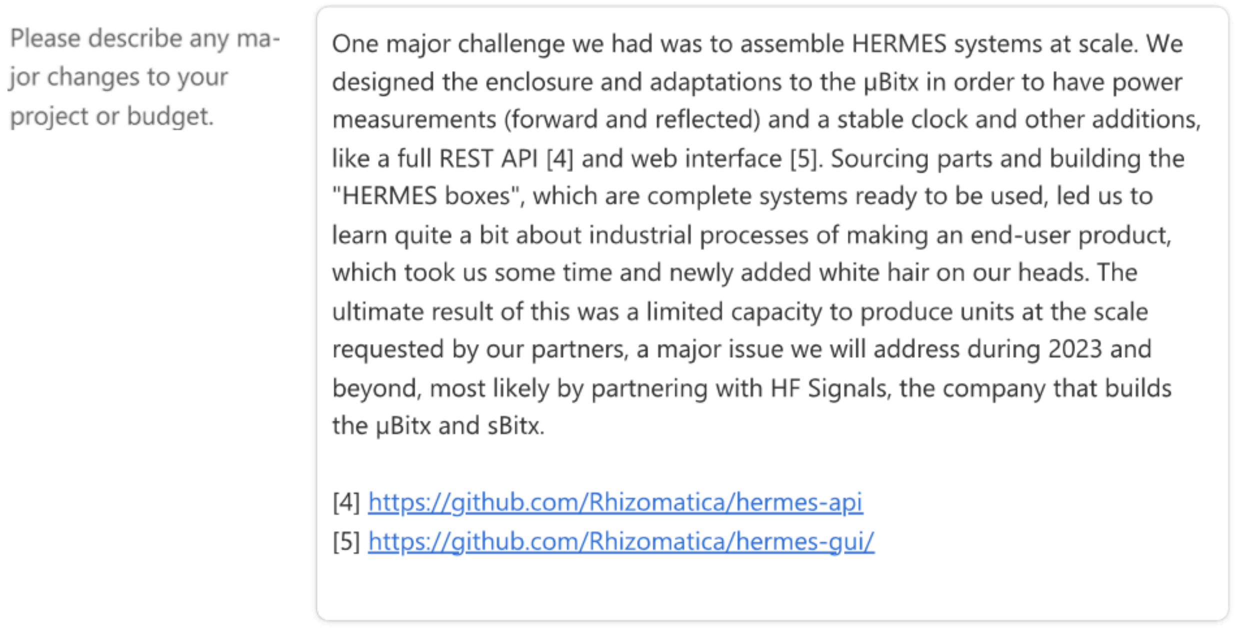
Rhizomatica Communications Final Report

Project Name: Digital HF Telecommunications for Civil and Amateur Uses



How many people did your ARDC funded project impact?	- 900 people living in indigenous communities in the amazon and the organizations that serve/represent them. - Customers/users of µBitx and sBitx hardware and WinLink and other proprietary solutions who now have viable FOSS options.



Please describe any major changes to your project or budget.

One major challenge we had was to assemble HERMES systems at scale. We designed the enclosure and adaptations to the µBitx in order to have power measurements (forward and reflected) and a stable clock and other additions, like a full REST API [4] and web interface [5]. Sourcing parts and building the "HERMES boxes", which are complete systems ready to be used, led us to learn quite a bit about industrial processes of making an end-user product, which took us some time and newly added white hair on our heads. The ultimate result of this was a limited capacity to produce units at the scale requested by our partners, a major issue we will address during 2023 and beyond, most likely by partnering with HF Signals, the company that builds the µBitx and sBitx.

[4] https://github.com/Rhizomatica/hermes-api
[5] https://github.com/Rhizomatica/hermes-gui/

Please share your results and anything else you would like us to know about your work:

Goals and Objectives:

Our project's goal was to push the development of our HERMES system forward in a few key ways:

- Develop a stand-alone, modular, open-source OFDM software modem
- Create a robust and easy to use hardware option for digital telecommunication over HF both narrowband and wideband.
- Incorporate state of the art audio and video compression to reduce payload size
- Evolve network topology options for deploying HERMES
- Explore both asynchronous UUCP data exchange, especially for email and file exchange, as well as synchronous data exchange.
- Get equipment into the hands of communities that most need it, specifically in the Amazon Rainforest and train them on how to install, use and maintain it.

Results: [note: a much more detailed report is attached]

The most relevant result of our project was to make improvements to modern digital HF telecommunication and ensure they are made available to all. We showed that HF telecom is affordable - a full HERMES HF setup costs under a thousand dollars, with no monthly fees - and stable, while able to reach places with no cabled or wireless infrastructure. The HERMES system works as a hub for telecommunication in each community, where users access services over WiFi, and their communication (e-mail and other messages) is exchanged with a city-located station over HF and routed over the Internet. Between the beginning of 2022 and early 2023, HERMES systems were deployed in 20 small communities in the Amazon rainforest region. We deployed the HERMES system in 5 communities in Brazil (in the State of Rondônia, with support from WWF and a local NGO named Kanindé). And HERMES were deployed in 15 places in Ecuador, in indigenous communities in the provinces of Pastaza, Sucumbios and Morona Santiago. We also set up a mobile station on a collective riverine transport boat, using a magnetic loop antenna as well as a prototype weather station. We estimate around 900 people directly benefited from the project.

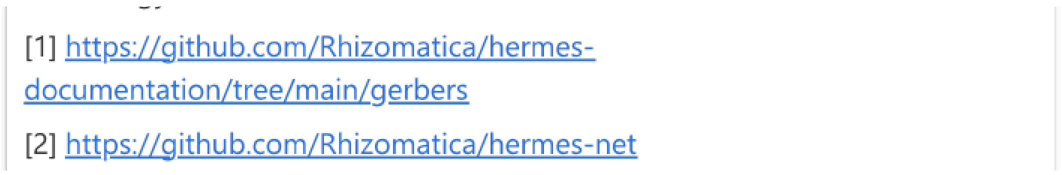
In terms of technical results, we started the development of and made substantial progress on a new free software SDR HF modem called Mercury

[3], which will eventually substitute VARA HF (the last remaining proprietary software in our HERMES system), and allow both "standard" HF channel-width (up to 3 kHz) communication and also wideband communication.

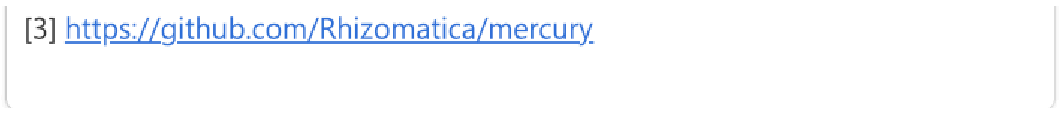
On the hardware front, we became directly involved in the refinement of Ashhar Farhan’s (VU2ESE) free hardware radio µBitx, and published all the improvements we contributed. Mr. Farhan accepted some of our improvements and included them in his next generation radio, the sBitx, which features a temperature compensated oscillator (which we fixed in the previous µBitx through the addition of a GPS PPS input comparator to dynamically adjust the LO drift [1]), forward and reflected power measurement circuitry (which we also independently developed for the µBitx) and a wideband pass-band filter which allows for up to 25 kHz of bandwidth, and will allow greater bitrate throughput and other benefits. The amateur radio community using home-brew and open source radio directly benefited from these developments. Additionally, the network stack we developed for e-mail and other types of messages [2] (including image and audio, through the use of the VVC codec for image and LPCNet for audio) exchange based on UUCP is now a viable alternative to WinLink and other proprietary solutions, which also benefits the ham-radio community looking for free software alternatives.

We consider our project a success, considering NGOs and local associations (WWF, OPAN, Kanindé, Instituto Socioambiental, Resex Chico Mendes) working with communities in hard to reach places got in touch with us and demonstrated interest in deploying our system in different contexts. People living in communities in the Amazon rainforest using the HERMES system are indeed the most inspiring and gratifying result.

At the same time, we know there is a gap in the development between other telecommunication systems and HF-based telecommunication systems, which became somewhat frozen in time when the first satellites were deployed in the 60's, so we are happy to contribute to the advance of telecommunication in the HF band and hopefully bring it up-to-date, using state-of-the-art technologies. This project helped re-position HF as a viable, modern technology for civilian use.


[1] https://github.com/Rhizomatica/hermes-documentation/tree/main/gerbers

[2] https://github.com/Rhizomatica/hermes-net


[3] https://github.com/Rhizomatica/mercury

Attachment:

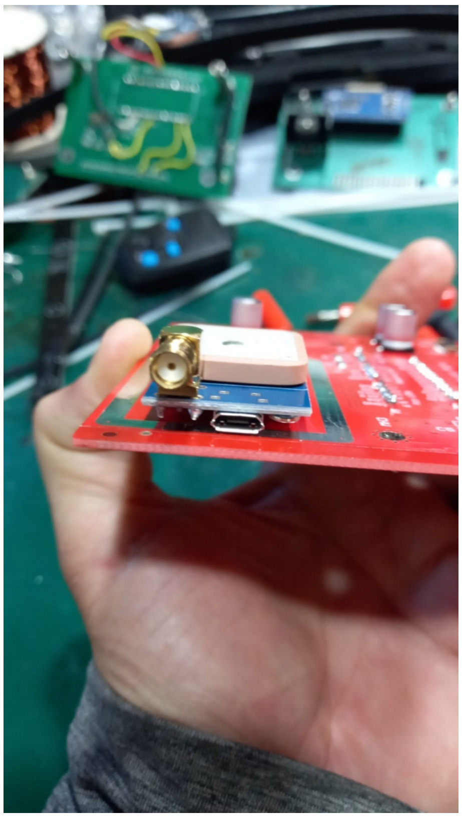
(ATTACHMENT) FINAL REPORT
RHIZOMATICA-ARDC

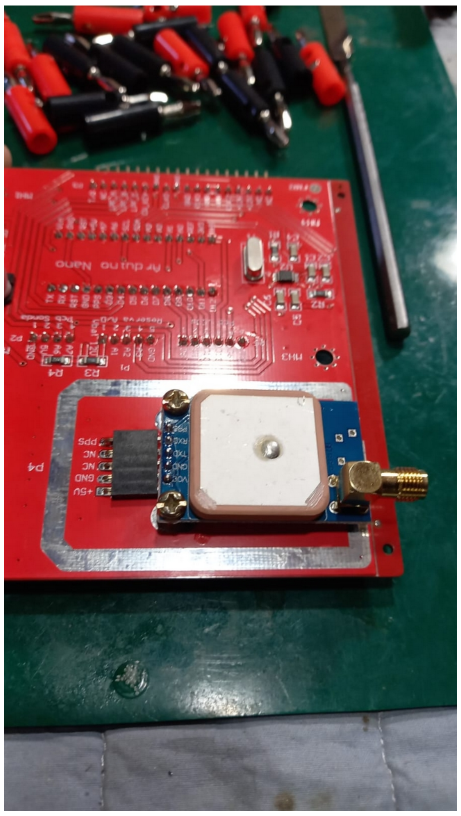
Grant Start Date: December 8, 2021
Grant End Date: February 1, 2023 (extended to March 31)

HF transceivers:

What we proposed: Improve our existing narrowband (3kHz) design. We have already built 10 v1 prototypes of the narrowband version, and will make further refinements towards a v2. This will include adding a GPSDO for precise frequency synthesis, power management for optimized battery use, built-in self-tests for better problem identification and embedded A/D and D/A converters to avoid use of computer sound cards for radio-computer connection.

What we did: We completed the GPS integration with the radio. The GPS (any model with PPS output can be used, we opted for the uBlox) provides a GPSDO capability to the radio LO (PPS used for this) and precise wall-clock to the Linux running in the computer connected to the radio (through USB connection). The board is a drop-in replacement to the original Raduino board of the original μBitx v6 (which our radio is based upon).
Arduino firmware code here:
https://github.com/Rhizomatica/hermes-net/tree/main/trx_v1-firmware
Userland code here:
https://github.com/Rhizomatica/hermes-net/tree/main/trx_v1-userland
Gerbers:
https://github.com/Rhizomatica/hermes-documentation/blob/main/gerbers/raduino_3903383A_Y7.rar





We improved our reflected and forward power measurement board (lambda bridge), which now has more stable and precise behavior versus previous versions.

We built 15 of these evolved versions of our radio. Five units were sent to the Brazilian Amazon, to be used by indigenous communities in the state of Rondônia, where they were successfully installed. Eight units were sent to Ecuador, to indigenous communities in the Pastaza, Sucumbios and Morona Santiago regions, and were also installed. The final two units stayed with us for testing. We carried out many tests during the grant period ranging from a few kilometers up to 650 km, typically using simple inverted V dipoles in the 5 MHz to 7 MHz frequency range.

What we proposed: Development of a wideband transceiver (up to 48kHz Tx and Rx bandwidth) prototype, which will allow higher transmission data rates and more concurrent users.

What we did: We researched two candidates for the base design of our wideband transceiver: the HermesLite2 and the sBitx. We did initial tests with both sBitx and HermesLite2 with success.. While the HermesLite2 has good signal synthesis and good reception, we had problems sourcing it, as it uses an FPGA which was hard to find due to supply chain issues. We managed to buy 4 units from Makerfabs (https://www.makerfabs.com/) using refurbished FPGAs. An additional consideration around the HermesLite2 is that the maximum tx power is 5W. The other candidate, the sBitx (https://www.hfsignals.com/index.php/sbitx/), uses much simpler, standard electronic components, and no FPGA. The stock sBitx can provide up to 50W (we managed to extract up to 60W with larger heat sinks just to test its limits). Our decision, therefore, is to base our wideband transceiver on the sBitx. While the decision about the hardware was made, our software-defined modem for wideband HF (Mercury) is not yet totally ready, so it will take a bit more time for us to have a full solution for wideband HF.

Modem:
What we proposed: Based on existing open-source OFDM modem implementations, we will add Media Access Control, channel aggregation, Automatic Repeat Request and Automatic Link Establishment logic in a way that allows the system to be used without any prior knowledge of HF operation. The system will allow multiple users to be concurrently connected, though the use of both time division and frequency division multiplexing.

What we did: We first carried out research on available OFDM modems, especially the one used by FreeDV and FreeDATA (David Rowe’s OFDM implementation available here: https://github.com/drowe67/codec2/blob/master/README_ofdm.md). We realized we need many different modes and higher performance OFDM/LDPC codes for our modem, in order to have the maximum possible throughput and efficient adaptive modulation (gear shifting), so we decided to write the modulation and LDPC from scratch. The modem developed during the year, called Mercury, uses state-of-the-art LDPC coding in 3 different code rates, supports many different OFDM modes (BPSK, QPSK, 16QAM and 64 QAM) and configurable bandwidth, guard interval, number of carriers and number of pilot carriers. Our new modem can adapt to very different channel conditions, for example, very low SNR up to high SNR, different doppler shift and spread and different multipath conditions. All the development put in the modem this year will allow the development of powerful adaptive links between stations, which will include modulation and coding adaptation given the available channel conditions. The modem also reports many parameters of the incoming signal, such as SNR and signal strength, which will allow for the development of a sophisticated MAC layer. We implemented a basic Automatic-Repeat-Request (ARQ) logic which makes the modem already useful for reliable data exchange. The ARQ code, still under active, heavy development, is available here:
https://github.com/Rhizomatica/mercury/tree/arq_alpha

We did not complete the development of the multiple users medium access (at this point, it is only point-to-point) nor the adaptive modulation selection (aka “gear shifting”), but we advanced on the understanding of the different modes of the modem. Next steps include the development of the logic for selection of the optimal modes to be used given a channel propagation condition, and the extension of the MAC layer to allow for multiple users medium access.

Relevant code of our modem is here:
https://github.com/Rhizomatica/mercury

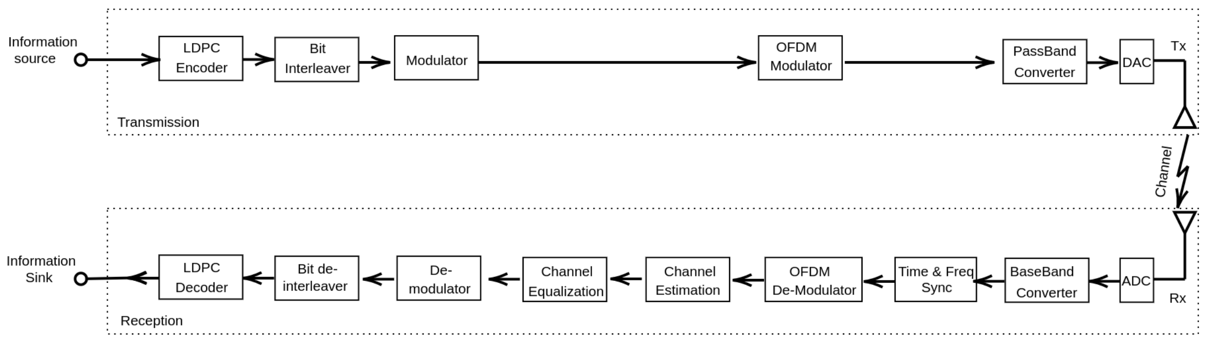


Figure 1: Mercury Soft-Modem diagram.

Overview:
Fig. 1 demonstrates the block diagram for Mercury software-defined modem that features an Orthogonal Frequency-Division Multiplexing (OFDM) modulator/demodulator with a Low-Density Parity-Check (LDPC) error correction code encoder/decoder with an embedded Additive White Gaussian Noise (AWGN) channel simulator.

Design:
The Mercury software modem was designed as high-performance modular C++ code to allow for further development and enhancement. High configurability was set as a priority avoiding hard-coded configuration to provide an adaptable modem for a wide range of necessities, environments, and hardware.

The modem was created without the use of external libraries (except for the ALSA Audio driver) permitting its porting to any operating system that supports a standard C++ compiler including embedded systems.

The modem was successfully tested on standard Intel-based computers with a 2 GHz processor and a couple of Gigabytes of RAM as well as a Raspberry Pi 3. The total memory requirement is under 10 MB.

OFDM:
The OFDM modulator and demodulator were designed with configurable bandwidth, configurable Fast Fourier Transform (FFT) size, configurable guard interval to remove the inter-symbol interference effect, and a configurable number of sub-channels.

OFDM Framing and Pilots - The implemented dynamic framing allows for a configurable number of symbols per OFDM frame, and configurable distribution of pilots via two parameters (Dx: frequency distance between pilots and Dy: time distance between pilots), Pilot boost is possible as well to define pilot carriers' power in relation to data carriers' power.

Bit Interleaving:
The bit interleaving and deinterleaving were implemented with the interleaving block size to combat pulse noise.

Modulation:
The modulation and demodulation blocks were implemented with six uniform constellation options: (BPSK, QPSK, 8QAM, 16QAM, 32QAM, 64QAM) and an approximate Log Likelihood soft demodulation to permit the deployment of powerful forward error correction codes.

Error Correction:
Forward error correction codes in the form of state-of-the-art LDPC codes were designed by a specially built Artificial intelligence agent (based on Artificial Immune Systems) and implemented with the size 1600 bits with three different code rates that define the protection level vs data rate (2/16, 8/16, 14/16).

The low-complexity Gradient Bit-Flipping (GBF) LDPC decoder with a configurable number of maximum decoding iterations was selected as our LDPC decoder.

Up/Down Frequency converters:
The conversion of the generated baseband signal to an intermediate passband signal at the transmission point was implemented with a configurable carrier frequency and output power. The produced passband signal can be interpolated to match the required output sampling frequency.

The conversion of the intermediate passband signal back to a baseband signal at the receiving end was implemented with a built-in Finite Impulse Response Filter designer with configurable transition bandwidth, cut frequency, and different window options (Rectangular, Hamming, Hanning, Blackman). The input passband signal can be decimated before passing it to the OFDM processing blocks.

The ALSA sound driver is used to interface the Mercury soft-modem blocks with the RF radio via the audio interface.

Synchronization:
Time synchronization is used to detect the start of each OFDM symbol and frame and was implemented with a configurable number of time synchronization detection peaks to be considered. A time synchronization lock was implemented to allow for phase symbol detection in case of successive OFDM frames without extra processing. Additionally, the number of symbols to be used in time synchronization is configurable.

To compensate for the carrier frequency offset (CFO) due to the Doppler shift and oscillators' frequency offset due to changes in temperature, a frequency synchronization was implemented with a configurable CFO sensitivity.

Channel Estimation and Equalization:
To reverse the channel effect and time synchronization mismatches, a channel estimator and equalizer are used. The channel estimator used the known transmitted pilots to calculate the channel effect in their sub-channels and the noise variance, then via one dimension and time and frequency interpolation predicts the channel effect in the data sub-carriers.

Once the channel is estimated, the channel equalizer performs the amplitude and phase correction for the data sub-carriers of the OFDM symbols.

Built-in Simulation:
Mercury operates in two simulation modes to provide the user with a prediction of the configuration performance, both simulation modes use an AWGN channel.

A baseband simulation permits the evaluation of the modulation and error correction capacity, while the passband simulation allows for the full transmission and reception stages to be simulated. The results can be printed on the terminal screen or (optionally) can be plotted using the Gnuplot library.

Test Modes:
Two test modes can be used, a random data transmission test mode or a reception test mode with/without plotting

TCP/IP interface:
An embedded TCP/IP interface was implemented to permit Mercury to act as a transparent data tunnel allowing for simpler and more efficient interface implementation.

Built-in Measurement:
In addition to the audio signal strength measurement, a Signal-to-Noise Ratio (SNR) is measured for the successfully decoded messages. Additionally, calculations such as Shannon limit, and bit rate available to the user were implemented to provide useful information to the user.

Tests:

During the development of the Mercury modem, several Black-Box, and White-Box tests were performed to ensure the functionality of each of the modem blocks. In addition, several simulations and performance tests were performed to evaluate the modem processing requirements and error-free data transmission capability.

A. Loop-Back test: This test was done via an audio loop-back connection and the deployment of the built-in AWGN channel simulator to evaluate the Mercury modem performance in comparison to the theoretical curves. Fig. 2 demonstrates the results of one of the passband simulation for BPSK modulation and an LDPC code with the code rate 2/16.

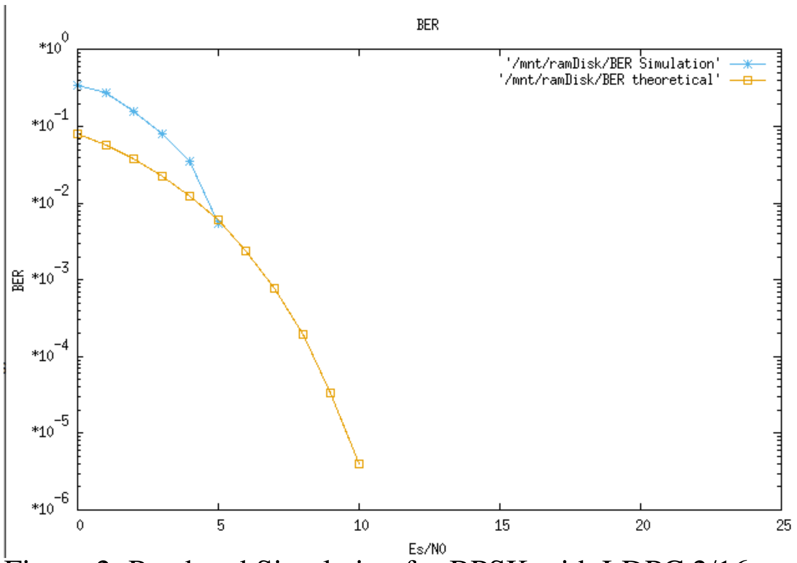


Figure 2: Passband Simulation for BPSK with LDPC 2/16.

B. Channel Simulator test: Once the Loop-Back test was successful, the IONOS Propagation Simulator shown in Fig. 3 was used as an additional externally-developed channel simulator to confirm the test results.

Figure 3: IONOS Propagation Simulator.

C. μBitx Air Gap test: The air gap test was performed by using two μBitx radios with dummy loads and transmission over a small air gap to ensure compatibility between Mercury and the μBitx hardware over different bandwidths and carrier frequencies.

D. Field test: The field test was performed over a 2.3 KHz bandwidth and 7115 Mhz carrier frequency over 90 km and 400 km distances. Fig. 4 shows the antenna setup of one of the test stations used in this phase. While the connection was established with error-free messages received when the channel conditions were relatively good (no interference from other radios and high SNR), poor channel conditions resulted in a loss of connection thus promoting the necessity for further development for poor channel conditions and low SNR values.

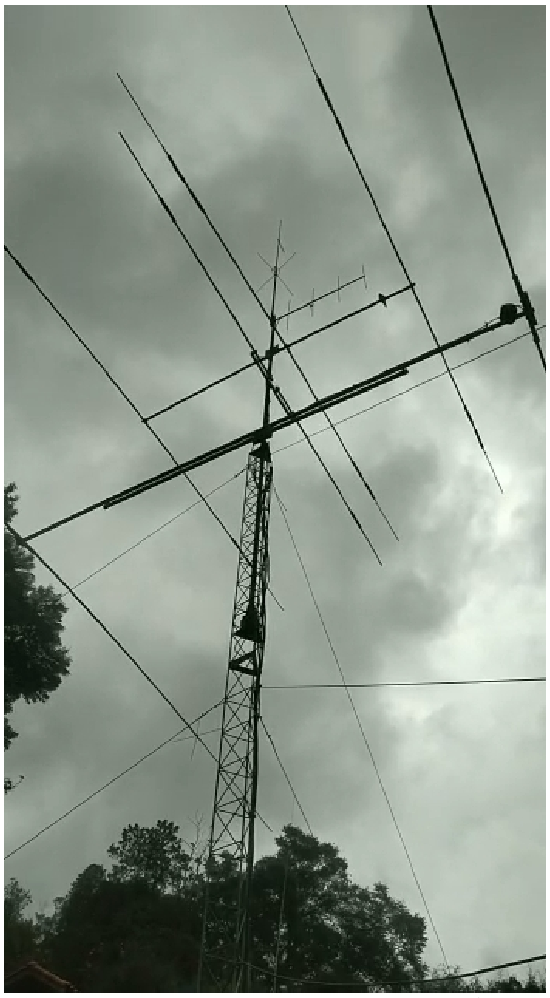


Figure 4: The antenna setup at one of the test stations.

Network topologies:

What we proposed: Point-to-point topology; Star topology; Mesh topology,

What we did: We implemented the Point-to-Point and Star topologies using VARA and Mercury, but other multi-point or mesh network topologies will only be implemented after we finish all the needed support in Mercury and our evolved MAC layer implementation.

Data exchange modes:

What we proposed: asynchronous UUCP data exchange, especially for email and file exchange.

What we did: We successfully implemented asynchronous data exchange with the use of the UUCP protocol, which supports VARA and ARDOP modems, and is ready to support our modem (Mercury). Relevant code is here: https://github.com/Rhizomatica/hermes-net/tree/main/uucpd

What we proposed: synchronous data exchange, including IP-based services like messaging and web.

What we did: While we can not yet do synchronous data exchange, because we don't have the needed support in the lower network layers yet (we are using just UUCP for now). We are

experimenting with bridging some services over email. We have tested bridging SMS, relevant code here:
https://github.com/Rhizomatica/hermes-messaging/
And web content using the Webxdc standard (https://webxdc.org/).

We also implemented sensor data (for now, charge controller and GPS data) gathering and transmission. Relevant code here:
https://github.com/Rhizomatica/hermes-sensors

Audio and Image encoding:
What we proposed: Evaluate the quality and computer complexity of state-of-the-art neural-network based audio and image encoders, like LPCNet, Lyra and SSMGAN for audio, and HIFIC for images. These codecs will be used for audio and image encoding (or transcoding) for both asynchronous and synchronous types to reduce payload size.

What we did: We tested three neural-network based audio encoders: LPCNet (https://github.com/xiph/LPCNet), Lyra (https://github.com/google/lyra) and NESC[1] (based on SSMGAN). The lowest rates these codecs can do are: LPCNet at 1.6 kbps, Lyra at 3.2 kbps and NESC at 1.6 kbps. NESC is not open source (yet), and we signed an NDA with Fraunhofer Institute in order to use it. NESC provides the best quality and is somewhat tolerant to background noise, while LPCNet is the best open-source audio codec available, providing good speech quality when background noise is not high. Lyra's bitrate is too high for the audio quality it provides when compared to LPCNet, which justifies our decision not to use it. We adopted LPCNet as our audio exchange codec, and we are using NESC for selected communities for evaluating the need for an improved open-source low-bitrate audio codec.

We tested many visual/video codecs, both machine-learning and non-machine-learning based. The machine-learning based ones, especially the HiFiC (https://github.com/tensorflow/compression/tree/master/models/hific) require too much resources (sometimes more than 16GB of RAM if no GPU is available), and most of them require a powerful GPU. So we opted to use the state-of-the-art H.266 (VVC) standard to compress still images.

The results of the compression of audio and image samples are available here:
https://hermes.radio/qa/

The code to compress images is available here:
https://github.com/Rhizomatica/hermes-net/tree/main/system_scripts/compression

Packages for the encoders are available here:
http://packages.hermes.radio/

Some scripts and the data-set used for the quality assessment experiment are here:
https://github.com/Rhizomatica/hermes-qa/

We integrated the transcoding of audio and image to LPCNet (or NESC) and VVC respectively in the email pipeline UUCP transport here:
https://github.com/Rhizomatica/hermes-net/tree/main/uuxcomp

Additional Resources: Photos, Media coverage, etc.

Our website with some photos and video:
https://www.rhizomatica.org/hermes/

- Media publications about the HERMES deployment in Brazil:
https://rondoniaovivo.com/noticia/geral/2022/06/04/protecao-aldeias-indigenas-de-ro-terao-sistema-inedito-de-monitoramento-no-brasil.html (Portuguese)

[1] Refer to: "Pia, N., Gupta, K., Korse, S., Multrus, M., Fuchs, G., 2022, NESC: Robust Neural End-2-End Speech Coding with GANs. Proc. Interspeech 2022, 4212-4216, doi: 10.21437/Interspeech.2022-430"

https://www.earthnewsterra.com.br/noticias/novo-sistema-de-internet-de-baixo-custo-chega-a-aldeias-em-rondonia/ (Portuguese)
https://www.earthnewsterra.com/news/new-low-cost-internet-system-reaches-indigenous-communities-in-the-amazon/ (English)

- Blog post about HERMES (photos of the deployment in Brazil, Rondonia state):
https://connecthumanity.fund/connecting-the-most-remote-communities/

- APC online publication (photos from both Brazil and Ecuador deployments):
https://www.apc.org/en/blog/seeding-change-rhizomaticas-high-frequency-radio-showcases-power-communication-remote-regions

- 48percent.org blog posts:
https://web.archive.org/web/20221215111202/https://www.48percent.org/using-high-frequency-digital-radio-systems-to-connect-amazonian-communities/
https://www.48percent.org/blog/community-radio-services-for-indigenous-communities-across-ecuadorian-amazon/

- Interview (video) in late 2022 about HERMES (in Portuguese):
https://www.youtube.com/watch?v=MAc2yjTRiPE

- Presentation in February 2023 at OsmoDevCall (
https://osmocom.org/projects/osmo-dev-con/wiki/OsmoDevCal):
Slides: https://github.com/Rhizomatica/hermes-documentation/blob/main/presentation/presentation_osmodevcall/2023-01-OSMO-presentation.pdf
Video: https://downloads.osmocom.org/videos/osmodevcall/osmodevcall-20230215-rafael2k-long-range-hf-comm_h264_420.mp4