Emitting Hellschreiber from a Raspberry Pi GPIO: combining gr-hellschreiber with gr-rpitx

GNU Radio running on an embedded board as emitter

Jean-Michel Friedt\textsuperscript{1}, Thomas Lavarenne\textsuperscript{2}

\textsuperscript{1}FEMTO-ST Time & Frequency department, Besançon, France
\textsuperscript{1}Lycée Jean Rostand, Villepinte, France

jmfriedt@femto-st.fr

slides and references at jmfriedt.free.fr
gr-rpitx sink block

- gr-rpitx¹: connect any (10-250 kHz bandwidth limited by DMA rate) GNU Radio processing sequence to a transmitting sink
- be aware of the many harmonics and spurious: bandpass filtering is mandatory
- do not emit over the air unless you know what you are doing

¹https://github.com/jmfriedt/gr-rpitx
gr-hellschreiber² source and sink blocks

gr-hellschreiber: low bandwidth Fax-like graphical communication mode

- emitter block (to rpitx, running on the embedded board)
- receiver block (from RTL-SDR dongle, running on the embedded board)
- decoder block (ZMQ stream to binary picture): display ⇒ running on laptop

Options
ID: HellGen
Title: Hellschreiber Generator
Author: Barry D. Lavarenne
Description: Hellschreiber Generator
Output Language: Python
Generate Options: No GUI
Run Options: Prompt for Exit

Variable
ID: samp_rate
Value: 12.005k

Variable
ID: symbol_rate
Value: 122.5

Variable
ID: decim1
Value: 10

Variable
ID: decim2
Value: 1

Variable
ID: symbol_rate
Value: 122.5

Variable
ID: samp_rate
Value: 12.005k

Variable
ID: decim1
Value: 10

Variable
ID: decim2
Value: 1

Variable
ID: samp_rate
Value: 12.005k

Variable
ID: symbol_rate
Value: 122.5

Variable
ID: decim1
Value: 10

Variable
ID: decim2
Value: 1

Variable
ID: samp_rate
Value: 12.005k

Variable
ID: symbol_rate
Value: 122.5

Variable
ID: decim1
Value: 10

Variable
ID: decim2
Value: 1

Variable
ID: samp_rate
Value: 12.005k

Variable
ID: symbol_rate
Value: 122.5

Variable
ID: decim1
Value: 10

Variable
ID: decim2
Value: 1

98=7x14
=number of pixels/letter

https://github.com/tlavarenne/gr-hellschreiber
Hellschreiber fonts and communication protocol

a. Definition

Feld-Hell is an on-off keyed uncoded text transmission system. Individual text characters are defined by dots in a dot matrix, and the matrix is transmitted in a fixed raster-scanned sequence. The transmission of dots is nominally timed at 122.5 baud, but no synchronising system is used. The receiving system operates quasi-synchronously, i.e. is adjusted to run at about the same speed, and small errors in speed are compensated for by displaying the received image twice, so that phase errors have limited effect. The system is thus immune to the sync-related errors of many text transmission systems.

Feld-Hell is relatively immune to noise interference, does not require complex transmission equipment, and performs well with low power.

b. Coding

Feld-Hell is uncoded, but sent as a raster scan of text characters. At the receiving end the received data, with errors, is displayed for visual interpretation. Since no coding is used, errors introduce noise, and thus effect the readability of the text, but cannot change the characters from one to another, as happens in coded systems, such as radio teletype.

---

https://www.nonstopsystems.com/radio/hellschreiber-fonts.htm

https://www.qsl.net/zl1bpu/DOCS/Hellspec.pdf
gr-hellschreiber source and sink blocks

gr-hellschreiber: low bandwidth Fax-like graphical communication mode

► emitter block (to rpitx, running on the embedded board)
► receiver block (from RTL-SDR dongle, running on the embedded board)
► decoder block (ZMQ stream to binary picture): display ⇒ running on laptop

https://github.com/tlavarenne/gr-hellschreiber
GNU Radio 3.9 Buildroot cross compilation

- Connect blocks with GNU Radio Companion on the PC, export resulting Python script to RPi and execute on embedded board.

---


5. https://github.com/oscimp/oscimp_br2_external
ZeroMQ streaming and decoding on PC

- Low bandwidth by preliminary processing on RPi and only communicating resulting pixel color for display
- Recording with DVB-T and 0MQ streaming to PC for display (distributed embedded system)
Conclusion

- functional port of GNU Radio to Buildroot for developing on embedded boards
- transition from GNU Radio 3.8 to 3.9 eased by OOT Porting Guide
- Raspberry Pi (4)+RTL-SDR as affordable hardware to become familiar with narrow bandwidth (<200 kHz) communication protocols implemented as SDR