GPS-Disciplined Shortwave Beacon for High Rate Of Measurement Ionospheric Propagation Analysis

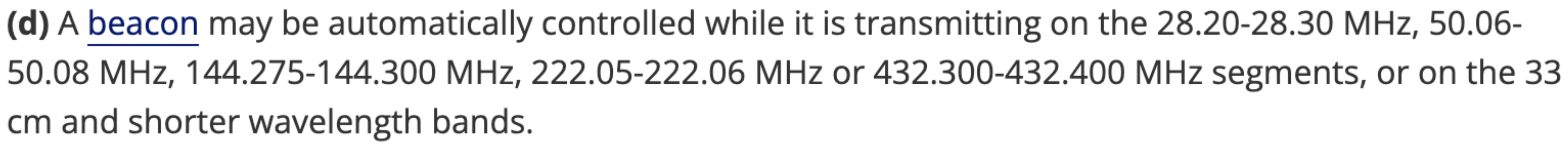
Ruslan Gindullin W2HAT/R9WFW[1]

[1]Case Amateur Radio Club W8EDU

So, beacons

Design requirements

- For ionospheric sounding, it is important to gather data 24 hours a day.
  - Automated unattended operation is required
  - FCC part 97.203 (d) on unattended beacon operation:



(d) A beacon may be automatically controlled while it is transmitting on the 28.20-28.30 MHz, 50.06-50.08 MHz, 144.275-144.300 MHz, 222.05-222.06 MHz or 432.300-432.400 MHz segments, or on the 33 cm and shorter wavelength bands.

- Since we are interested in HF propagation, we are limited to the 10m band only.

Screenshot: https://law.cornell.edu/cfr/text/47/97.203

Design requirements

- Time precision
  - The height of the F2 region of the ionosphere (which is responsible for most of the HF propagation) varies between approximately 200 and 400 kilometers.
  - The goal is to be able to discern between working a station 100 kilometers away directly (line-of-sight) or through skywave propagation.
  - The difference in length of the path in such scenario would be 23.6km
  - 23.6 km/300000 km/s=77.6*10^-6 seconds = 77.6 microseconds
  - For confident measurement, the timing precision of the signals should be half of that.
  - Thus, a timing precision of **30 microseconds** would be the requirement.
  - A signal of such precision that would sync the transmissions to UTC time could be provided by the 1PPS signal of a GPSDO unit.

Design requirements

- Frequency/phase precision
  - The goal is to be able to detect a one cycle per second doppler shift.
  - That puts us at a frequency resolution of at least **0.5 Hz**.
  - Therefore, a required frequency precision for confident measurement is at **0.1 Hz.**
  - A DDS clocked by a good GPSDO is capable of achieving such (and even greater!) precision with little phase noise.

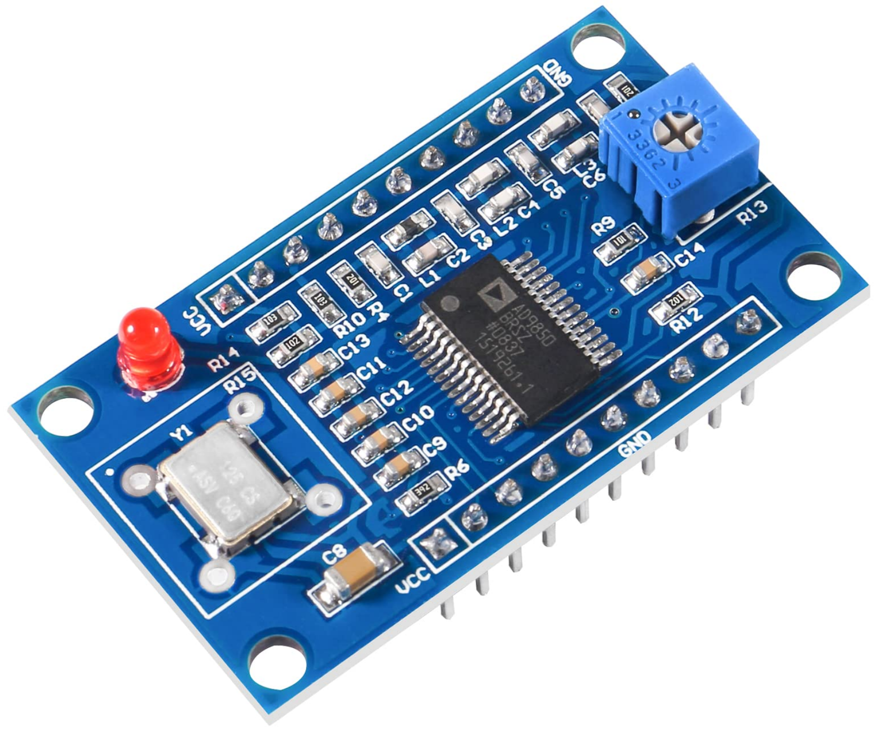
Design decisions

- DDS
  - **AD9850** is sufficient for this purpose.
  - An assembled module on PCB is available for $20
- DDS Driver
  - MCU is not deterministic.
  - FPGAs are cheap nowadays.
  - An FPGA dev board capable of such timings is available for $30
    - **Sipeed Tang Nano 20K**
- However, I'm not that good at verilog :)
  - MCU loading the word into the DDS, FPGA clocking and triggering it.
  - Any MCU capable of loading the DDS fast enough is fine.
  - The choice fell on an **Adafruit Metro ESP32-S2** dev board
    - ESP32 is fast
    - Arduino Uno format - could be ported to other Uno-compatible MCU devboards

Components

AD9850 DDS chip

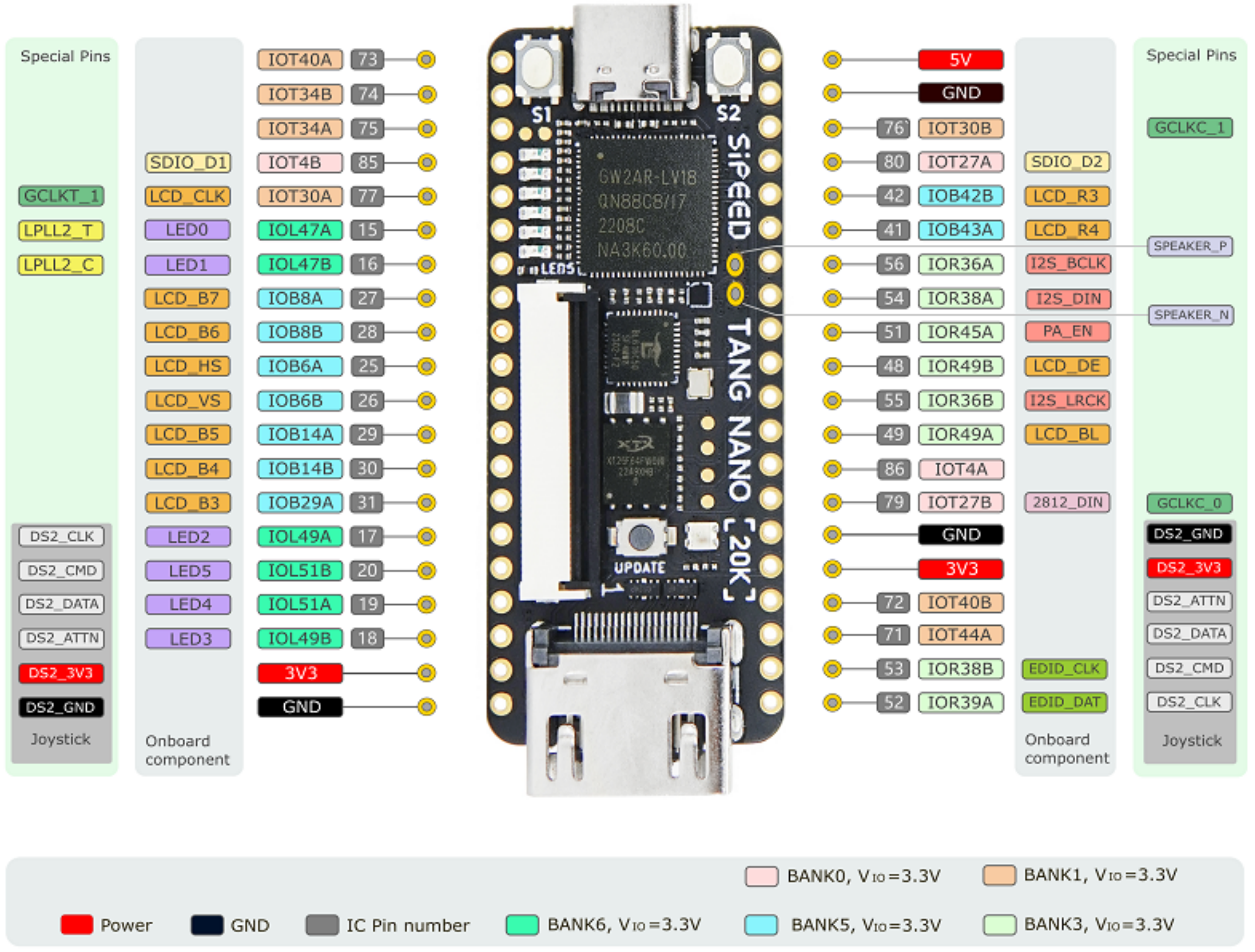
- Takes a **40-bit** frequency & phase word
- Two modes of operation: Serial loading and Parallel loading
- So far, serial is fast enough but parallel could be implemented as a further improvement.
- On the PCB module, I had to cut the trace from the onboard oscillator to solder up the SMA pigtail for the GPS reference.
- Frequency output is updated on the positive edge of the **FQ_UD** pin.



Components

Sipeed Tang Nano 20K

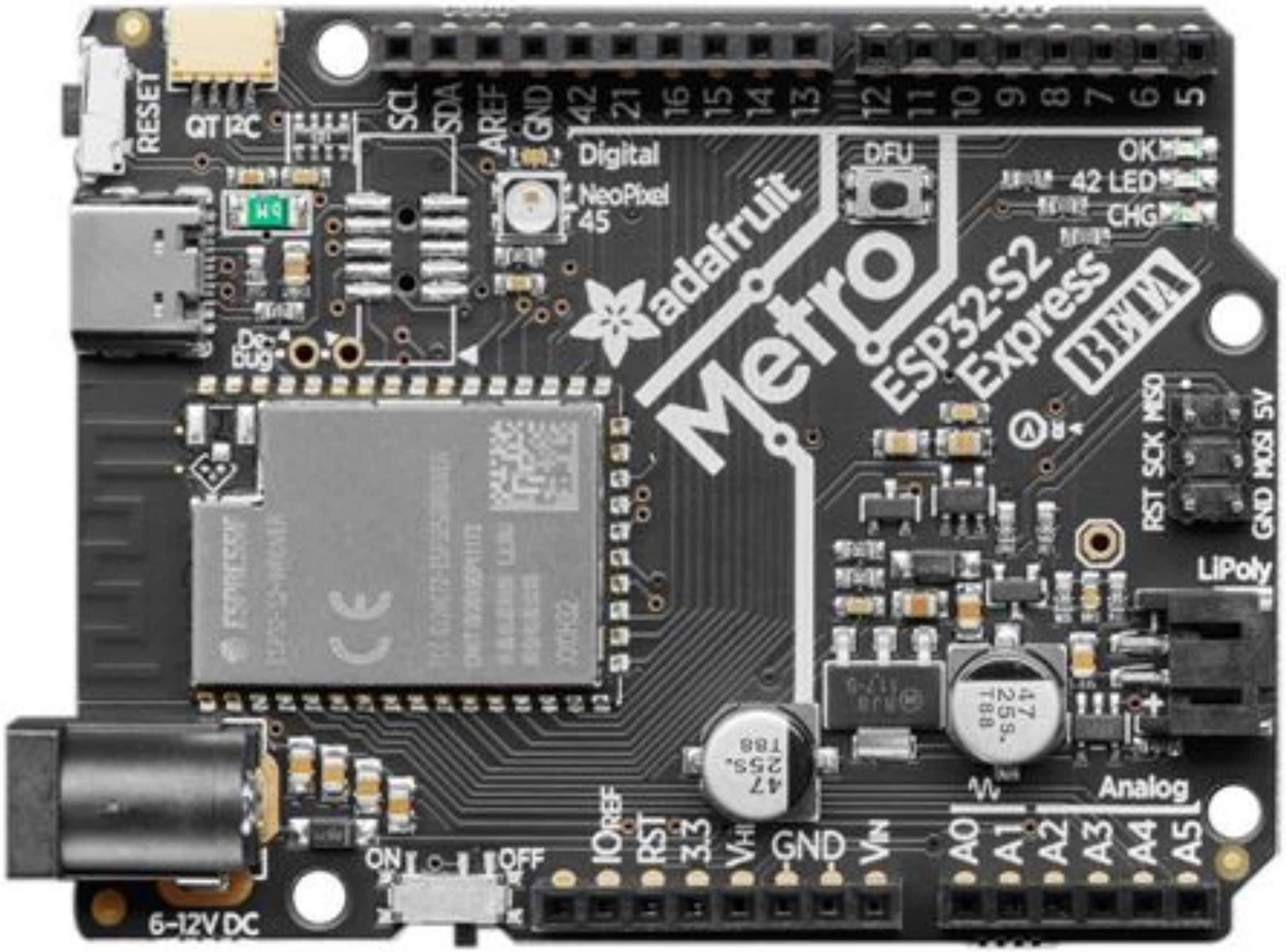
- Cheap and small!
- Readily available
- Not the most versatile SDK
- A small but active and growing community of makers using it.
- Packs quite a powerful FPGA chip by GoWin
- Ideal for critical timing applications and complex logic applications



Components

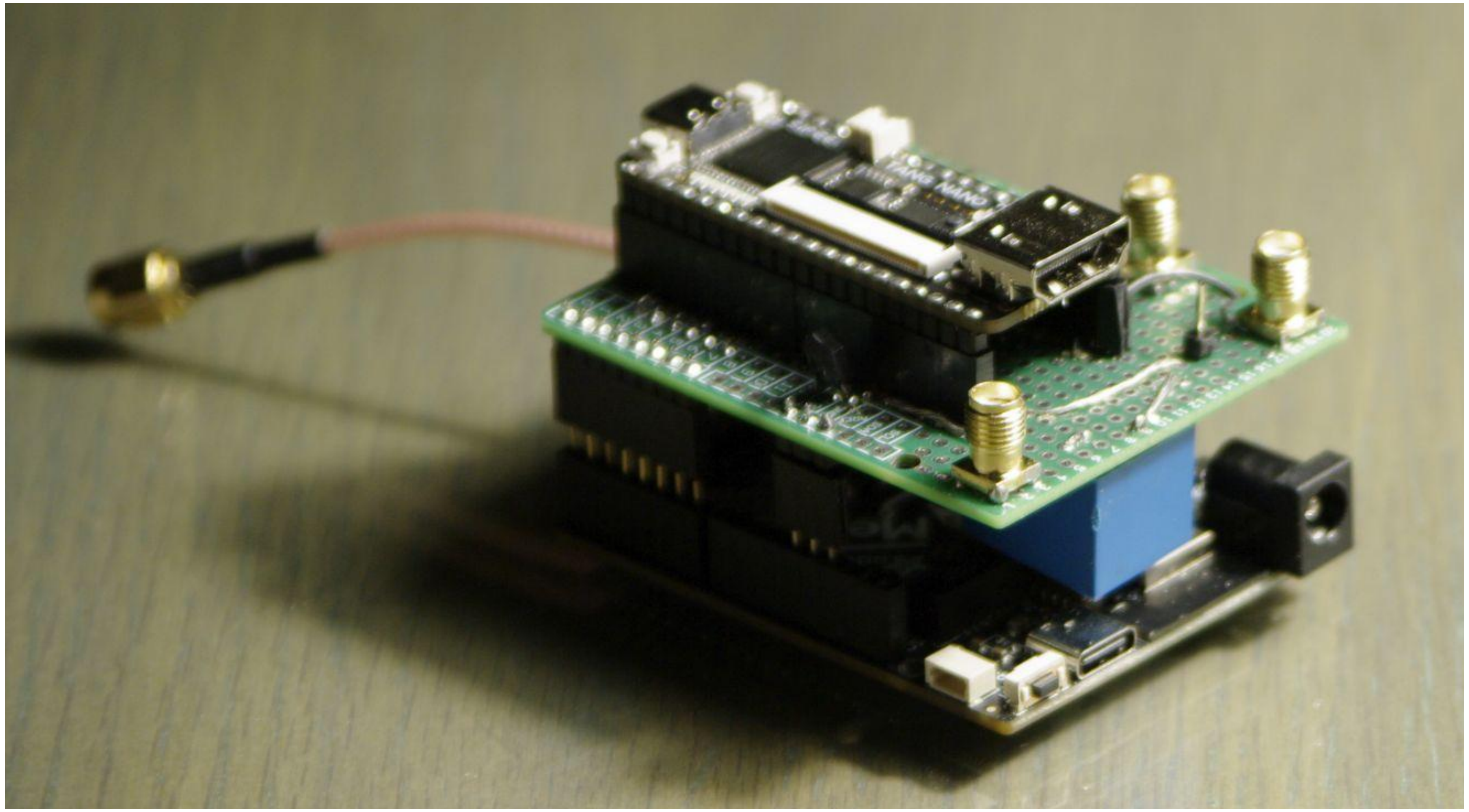
Adafruit Metro ESP32-S2

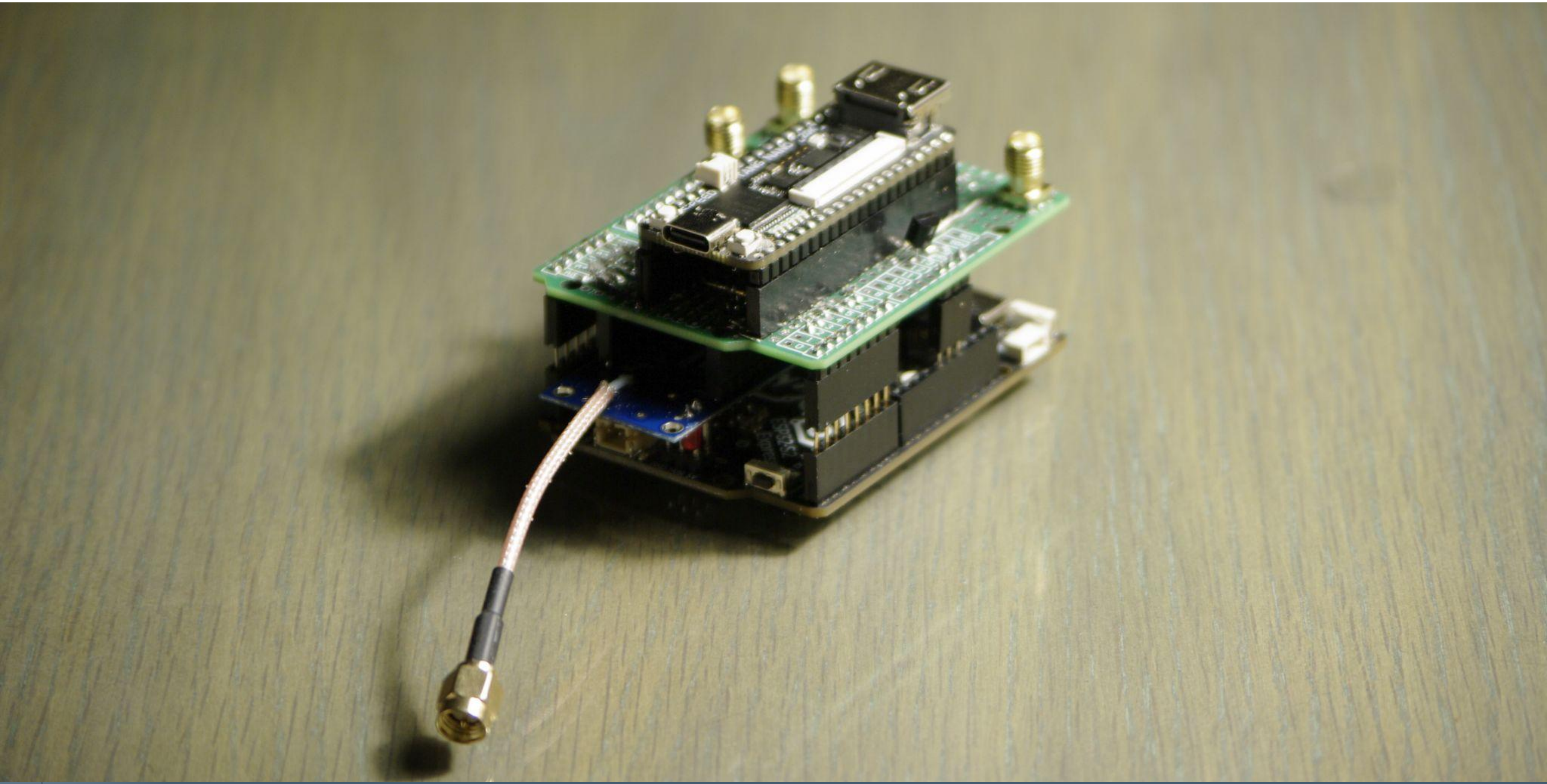
- ESP32 runs FreeRTOS

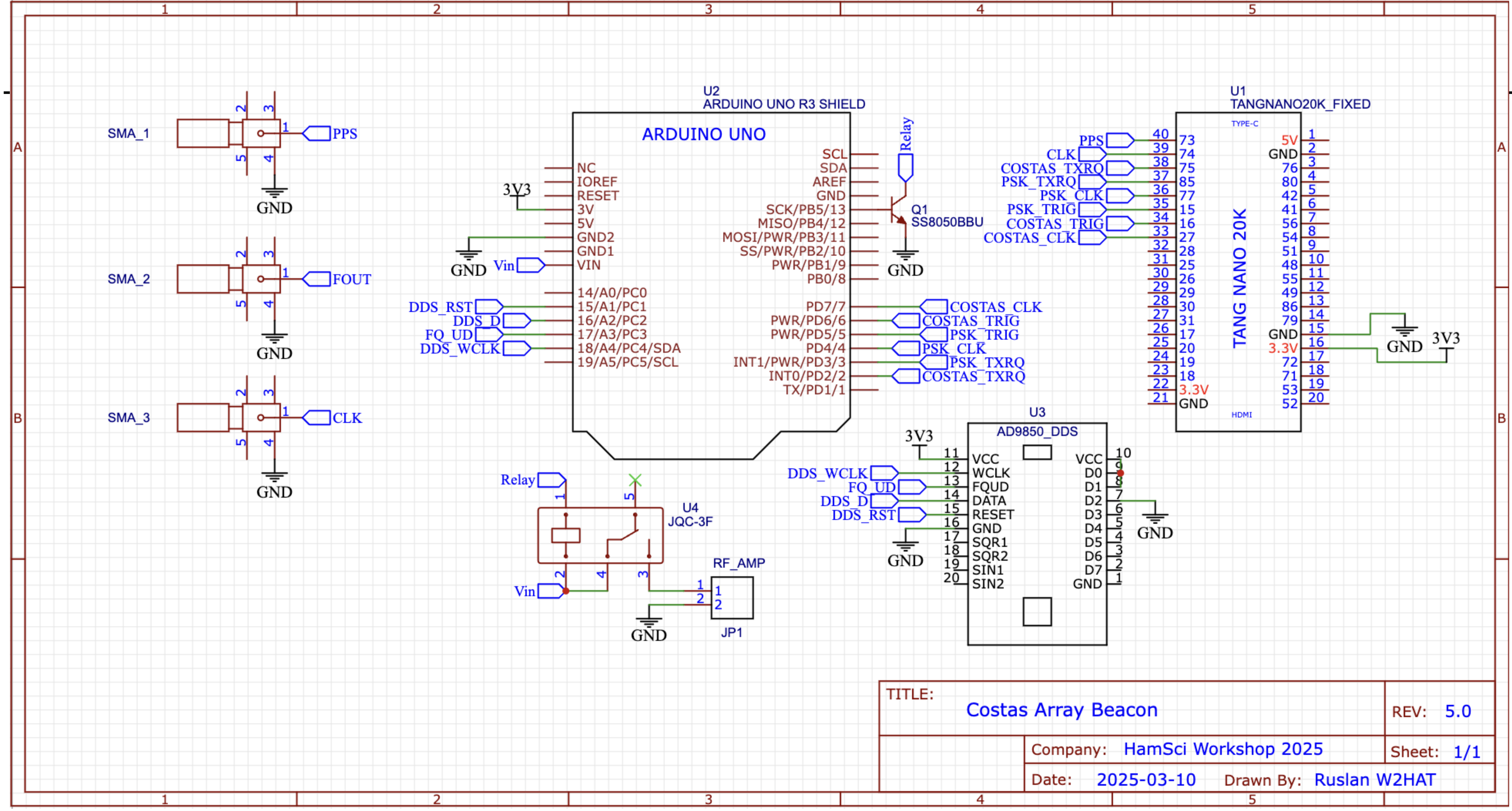


Hardware design

- Very modular!
- Requires an external GPSDO (built-in GPSDO todo?)
- Requires an external PA (1 mW output power)







TITLE: Costas Array Beacon

REV: 5.0

Company: HamSci Workshop 2025

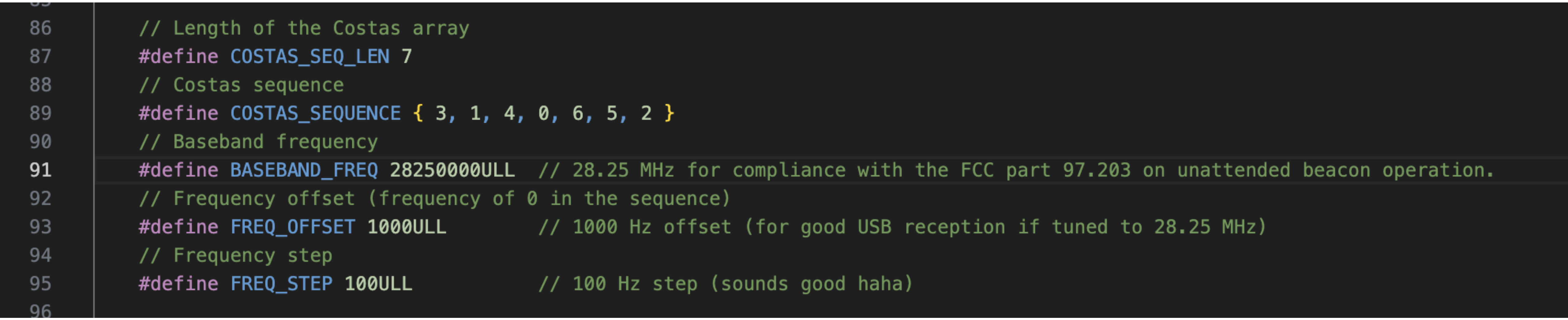
Sheet: 1/1

Date: 2025-03-10

Drawn By: Ruslan W2HAT

Software design

- Used Rob Tillaart’s AD985X library. Slightly modified it to fit the use case.
- Writing bare metal code quickly goes out of hand.
- FreeRTOS is the easiest solution for this task.
- Written using PlatformIO IDE using Arduino framework.
- Parameters are configurable in the header file.
- Code is open source and is publicly posted to Github.

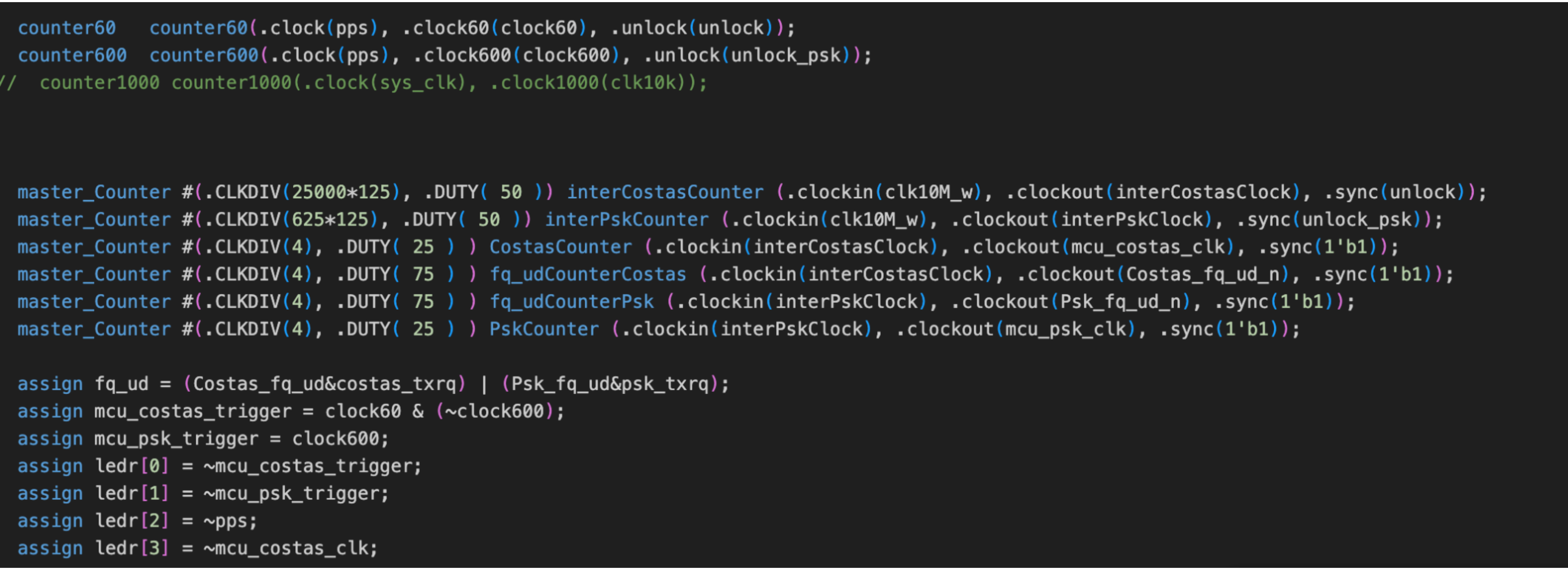


```
    // Length of the Costas array
    #define COSTAS_SEQ_LEN 7
    // Costas sequence
    #define COSTAS_SEQUENCE { 3, 1, 4, 0, 6, 5, 2 }
    // Baseband frequency
    #define BASEBAND_FREQ 28250000ULL  // 28.25 MHz for compliance with the FCC part 97.203 on unattended beacon operation.
    // Frequency offset (frequency of 0 in the sequence)
    #define FREQ_OFFSET 1000ULL        // 1000 Hz offset (for good USB reception if tuned to 28.25 MHz)
    // Frequency step
    #define FREQ_STEP 100ULL           // 100 Hz step (sounds good haha)
```

Gateware design

Not much here, just a few counter-based clock dividers

Could definitely be done without an external MCU (softcore RISC-V?), just not by me. (yet?)

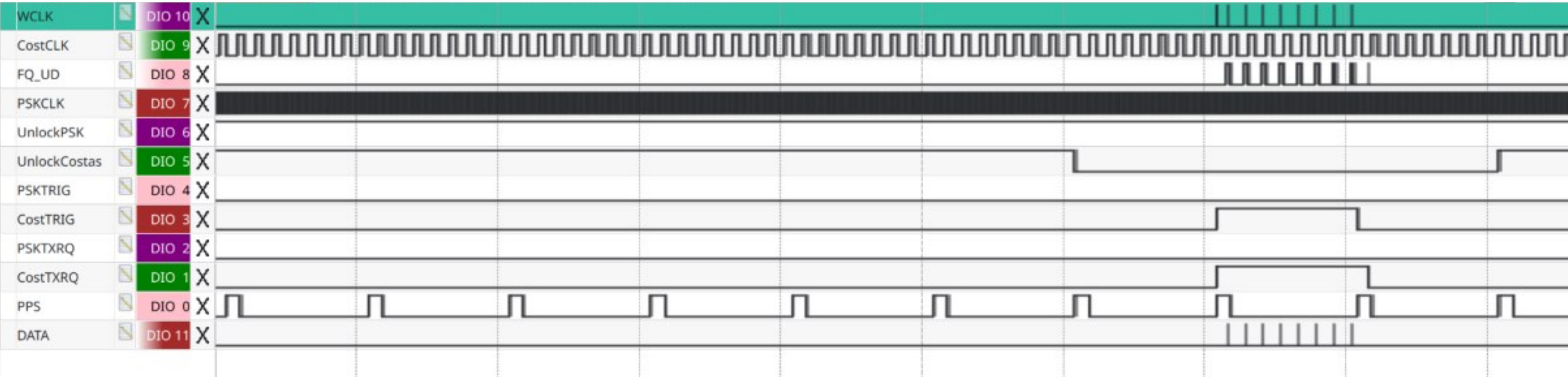


```
  counter60   counter60(.clock(pps), .clock60(clock60), .unlock(unlock));
  counter600  counter600(.clock(pps), .clock600(clock600), .unlock(unlock_psk));
//  counter1000 counter1000(.clock(sys_clk), .clock1000(clk10k));



  master_Counter #(.CLKDIV(25000*125), .DUTY( 50 )) interCostasCounter (.clockin(clk10M_w), .clockout(interCostasClock), .sync(unlock));
  master_Counter #(.CLKDIV(625*125), .DUTY( 50 )) interPskCounter (.clockin(clk10M_w), .clockout(interPskClock), .sync(unlock_psk));
  master_Counter #(.CLKDIV(4), .DUTY( 25 ) ) CostasCounter (.clockin(interCostasClock), .clockout(mcu_costas_clk), .sync(1'b1));
  master_Counter #(.CLKDIV(4), .DUTY( 75 ) ) fq_udCounterCostas (.clockin(interCostasClock), .clockout(Costas_fq_ud_n), .sync(1'b1));
  master_Counter #(.CLKDIV(4), .DUTY( 75 ) ) fq_udCounterPsk (.clockin(interPskClock), .clockout(Psk_fq_ud_n), .sync(1'b1));
  master_Counter #(.CLKDIV(4), .DUTY( 25 ) ) PskCounter (.clockin(interPskClock), .clockout(mcu_psk_clk), .sync(1'b1));

  assign fq_ud = (Costas_fq_ud&costas_txrq) | (Psk_fq_ud&psk_txrq);
  assign mcu_costas_trigger = clock60 & (~clock600);
  assign mcu_psk_trigger = clock600;
  assign ledr[0] = ~mcu_costas_trigger;
  assign ledr[1] = ~mcu_psk_trigger;
  assign ledr[2] = ~pps;
  assign ledr[3] = ~mcu_costas_clk;
```

Operation and data

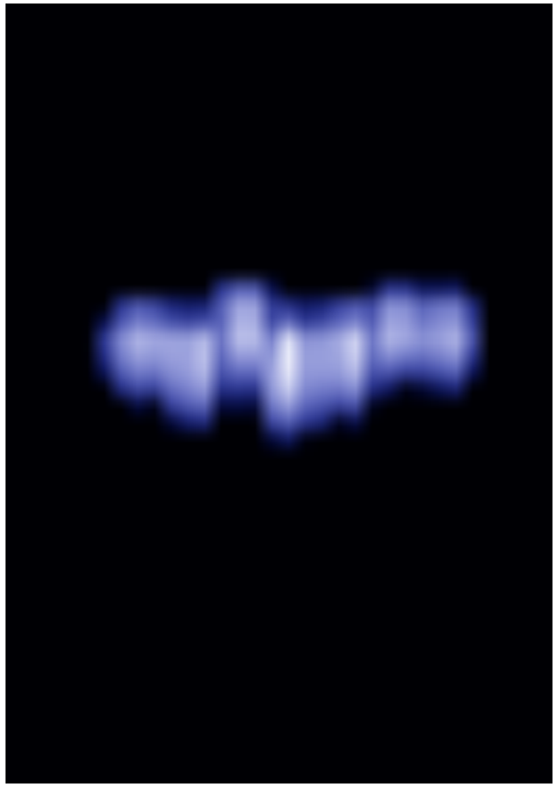
So far I have tested the signal timings with a logic analyser only with the initial 100ms/frequency costas array transmissions

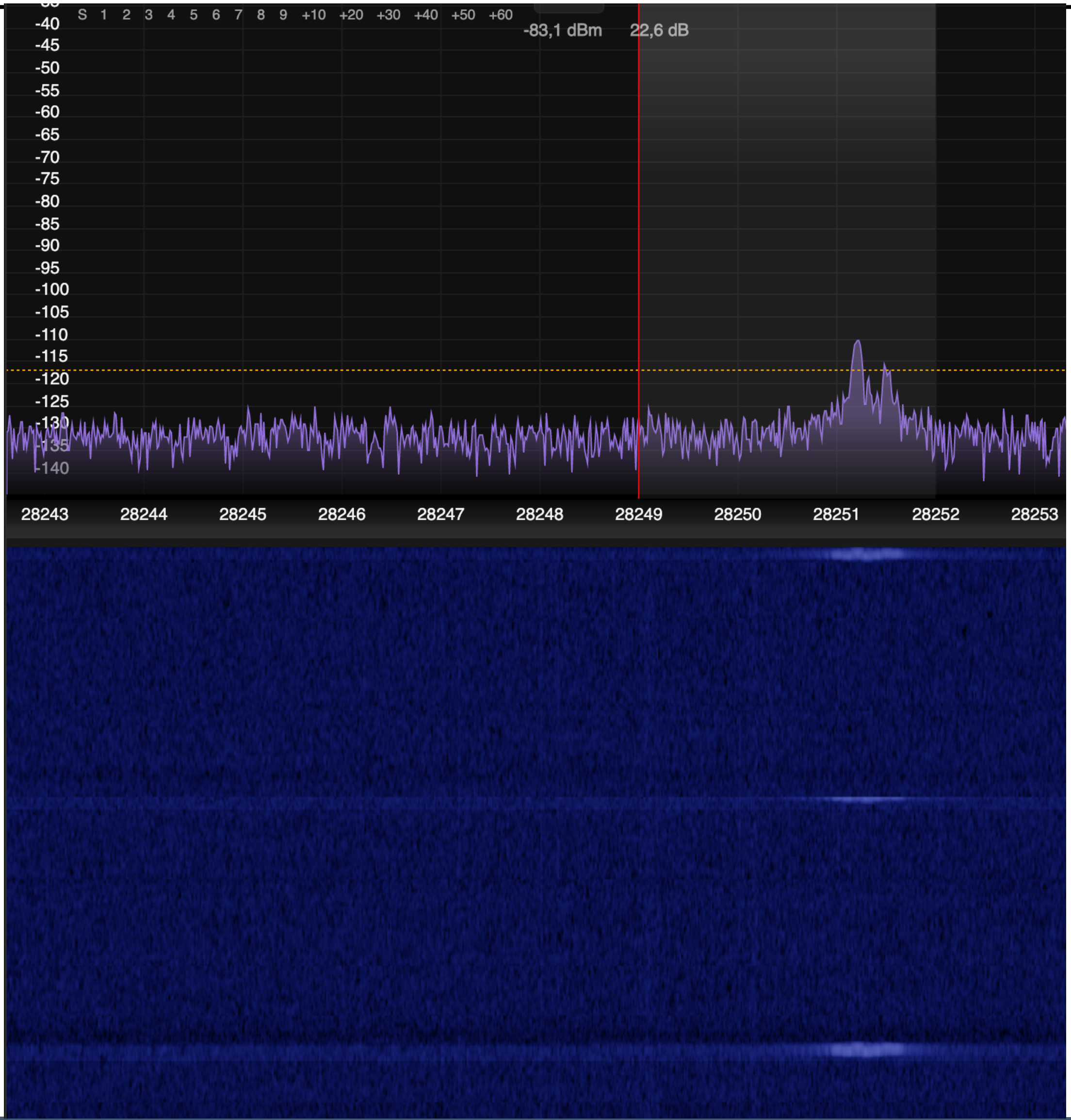
The timings were within the 30 microsecond constraint.



Operation and data

I have also tested the transmission of 10ms/frequency costas arrays, but I did not measure the timings due to the midterm season.





So, what do we have?

A beacon capable of transmitting configurable Costas arrays with precise timings and PSK-encoded Station ID and telemetry (QTH, timestamps in the works) that is very easy to build with off-the-shelf parts and costs under $100 in parts.

Would be very useful to conduct a measurement experiment with the beacons located far away, which brings us to…

Call to Action

- I would be happy if there were 4-5 stations located around the east and west coast willing to run this beacon for a few days to see if I get the results I am hoping for.
- If you are good at Verilog and RTL design, I would really appreciate contribution to this project - getting rid of an additional hardware MCU would make the beacon even cheaper and easier to build, as well as make the timings much tighter. Also feel free to contribute to the MCU codebase! The Github links are available in the last slide.
- FPGAs to masses! They're getting cheap enough for an average maker to be able to use them in their projects. Verilog is fun!

Questions

I would like to thank **Dr. David Kazdan AD8Y**, **John Gibbons N8OBJ**, and **Dr. Kristina Collins KD8OXT**, for the idea of the project and lots of useful insight into the signal processing and the black magic that RF engineering seems to be.

I would also like to thank the **HamSCI community** and the **organisers** for the attention to detail with scheduling and the opportunity to present here today.

Thank you!

Ruslan Gindullin W2HAT/R9WFW rxg695@case.edu
https://github.com/ruslanbd/Costas_v5_ESP - MCU design repository
https://github.com/ruslanbd/Costas_v3 - FPGA design repository