Sensor Network 2020

Sampo Kellomäki (sampo@iki.fi)

9. marraskuuta 2020

Tiivistelmä

Investigations and concrete project for constructing a sensor network for monitoring temperature and humidity (TH) inside isolated walls, solar heat panels, or ventilation systems.

We investigate IEEE 802.15.4 based solution as well as 1-wire bus based approach. Sensors may be powered by wire, battery, or energy harvesting, e.g. RF field, small solar panel, or even temperature differential of a thermo-pair.

1 Introduction

- Use kicad for electronics design
- Use JLCPCB for manufacturing
    - Use "basic"set of components from JLCPCB catalog to minimize manufacturing cost

Preliminary investigation shows that there is no simple ready made 1-wire TH sensor available on market (T only is widely available). Therefore we need to shop for TH component and microcontroller that can interface it (I2C) to 1-wire and/or radio path and then make a PCB that contains both.

The master or controller node of the sensor network is expected to be Raspberry Pi equipped with suitable bridge adapter for 1-wire or radio. We may manufacture this adapter using the same microcontroller as used on the sensors themselves, indeed one of the sensor boards may be thus configured and attached with serial cable to Rapsberry Pi or with USB adapter to a laptop.

Initially we estimate that 200 sensors are needed. We plan to manufacture first a pilot of 10 (or 30) and once that is successful, including software development, run a second, bigger, batch.

In 1-wire case the master polls the nodes periodically. Due to wired transmission of electricity, this is not particularly power constrained.

In radio case, the nodes wake up according to some schedule and TX their measurement. They then remain in RX for a short while in case the master wants to send something to them.

2 Work estimate and BOM

Schematic design and reading datasheets: 3d

PCB design (and rereading datasheets): 4d

Ordering and manufacturing: 1-2 weeks real time, 1d engineering time,

```
1d admin time for customs, etc.
```

Firmware programming: 7d (?) for minimal level, 28d for usable product

2.1 sens-rf-1w-1 - MCU based solution (RF and 1-wire by software)

- CC2530F256RHAR 15.4 2.4 GHz TXRX SOC
  - 1.58 USD (LCSC C9120 basic)
  - `Texas-Instruments-TI-CC2530F256RHAR_C9120.pdf`
  - `swru191f.pdf` `TI-CC253x-User-Guide-swru191f.pdf`
  - `http://www.ti.com/lit/pdf/SWRU191` `swru191f.pdf` "CC253x/4x User's Guide"
    - p.81, table 7-1 "Peripheral `I/O` Pin Mapping"
  - IEEE 802.15.4, 4.5 dBm (ca. 3 mW, 10 m range)
    - O-QPSK, 16 symbols, 32 chips each (kind of DSSS), `CSMA/CA`.
    - `http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf` *** no longer available

- 21 GPIO, 8 ADC, 2x USART (SPI master / slave capable)
  - No I2C peripheral: we program manually bitbang style
- 256 KB Flash (programmed through debug interface), 8 KB RAM
- 2..3v6 internally regulated
  - 30 mA when -100 dBm RF_RX
  - 39 mA when 4.5 dBm RF_TX
  - 1 uA in sleep 2
  - 9 mA medium CPU activity, no TX, no RX
  - Zeners and capacitors leak far more
- Pinout and use
  1. GND
  2. GND
  3. GND
  4. GND
  5. P1_5 / MOSI0b / TX0b / *SPI-CLK1b* / RTS1b / OBS5
  6. P1_4 / Tim3Ch1 / MISO0b / RX0b / *SPI-SS1b* / CTS1b / OBS4
  7. P1_3 / *Tim3Ch0* / SPI-CLK0b / RTS0b / OBS3 – *Type2 CP PWM generation*
  8. **P1_2** / SPI-SS0b / CTS0b / *Tim1Ch0b* / OBS2 – **SoftI2C CLK**
  9. **P1_1** 20 mA (no PUPD) / Tim4Ch1 / Tim1Ch1b / OBS1 – **LED_G**
  10. DVDD2
  11. **P1_0** 20 mA (no PUPD) / Tim4Ch0 / Tim1Ch2b / OBS0 – **LED_R**
  12. P0_7 / **A7** / Tim1Ch2 – **Light intensity detector ADC**
  13. P0_6 / **A6** / Tim1Ch4 – **Switch read ADC** / Type2 CP ADC monitoring
  14. P0_5 / A5 / Analog CompP / SPI-CLK0 / RTS0 / MISO1 / Tim1Ch3 – Mains voltage ADC and zxing
  15. P0_4 / A4 / Analog CompN / SPI-SS0 / CTS0 / MOSI1 / Tim1Ch2 – Mains voltage ADC and zxing
  16. P0_3 / A3 / MOSI0 / **TX0** / SPI-CLK1 / RTS1 / Tim1Ch1
  17. P0_2 / A2 / MISO0 / **RX0** / SPI-SS1 / CTS1 / Tim1Ch0 / OpAmpO
  18. P0_1 / A1 / OpAmpN – Shunt ampere metering?
  19. P0_0 / A0 / OpAmpP – Shunt ampere metering?
  20. **RESET_N**

21. AVDD5
22. **XOSC_Q1** – XTAL1 32 MHz
23. **XOSC_Q2** – XTAL1 32 MHz
24. AVDD3
25. RF_P – To antenna or balun
26. RF_N – To antenna or balun
27. AVDD2
28. AVDD1
29. AVDD4
30. RBIAS (R301 56k to GND)
31. AVDD6
32. **P2_4 /** XOSC32K_Q1 – **SoftI2C SDA3 /** *Relay*
33. **P2_3 /** XOSC32K_Q2 / Tim4Ch1b – **SoftI2C SDA2 /** *Plug switch*
34. P2_2 **/ Debug clock**
35. P2_1 **/ Debug data**
36. **P2_0 /** Tim4Ch0b – **SoftI2C SDA1**
37. P1_7 / *MISO1b* / RX1b / Tim3Ch1b
38. P1_6 / *MOSI1b* / TX1b / Tim3Ch0b – 1w-data
39. DVDD1
40. DCOUPL (1v8, C401 1uF cap to GND)

- Conflict between RX0 and OpAmpO?
- DMA channels: ADC, AES x2, `USART/SPI` x2, RADIO, CRC16?, Debug (during debug mode only)
- `https://www.instructables.com/ARDUINO-AS-A-8051-PROGRAMMER/`
- L252, L261: 2 nH
- C251, C261: 18p 0603: Samsung CL10C180JB8NNNC LCSC C1647 0.007 USD
- C262, C252: 1p 0603: Samsung CL10C010CB8NNNC LCSC C23969 0.0103 USD
- C253: 2p2 0603: Samsung CL10C2R2CB8NNNC LCSC C84711 0.0078 USD ext (2p C21895 basic)
- C231, C221: 27p 0603: Samsung CL10C270JB8NNNC LCSC C1656 0.0147 USD
- C401: 1u 0603: Samsung CL10A105KB8NNNC LCSC C15849 0.0128 USD

- R301, RSW2: 56k 0603: Uniroyal 0603WAF5602T5E LCSC C23206 0.0021 USD
- C1,C2,C3,Cbyp: 100n 0603: Yageo CC0603KRX7R9BB104 LCSC C14663 0.0068 USD
- R1, R2, R3, R4, RSW1, RPH1: 10k 0603: Uniroyal 0603WAF1002T5E LCSC C25804 0.0018 USD
  - 10k pullup resistor at 3v3 means 0.3 mA which is well within range of 4 mA GPIO capability
- LED_G: 0603: Everlight `19-217/GHC-YR1S2/3T` LCSC C72043 0.039 USD
- LED_R: 0603: Hubei KENTO KT-0603R LCSC C2286 0.0079 USD
- LED_P: 0603: Everlight 19-213/Y2C-CQ2R2L/3T(CY) LCSC C72038 0.024 USD
- R_G, R_R, R_P: 220R 0603: Uniroyal 0603WAF2200T5E LCSC C22962 0.0021 USD
- SW1,SW2,SW3: XKB TS-1187A-B-A-B LCSC C318884 0.0224 USD ext
  - `1901111503_XKB-Enterprise-TS-1187A-B-A-B_C318884.pdf`
- D_LD1, DPH1 Light detector diode: 3528-PTSM D3528 LCSC C242256 0.1231 USD ext
  - `Shenzhen-Jing-Chuang-He-Li-Tech-3528-PTSM-D3528_C242256.pdf`
- Note for FETs: at reset GPIO is configured as input with pullup (except P1_0 and P1_1)

- Sensirion SHT21
  - DFN6 2.90 USD (ext) LCSC C84828
  - `Hum/temp/eob`, I2C, 2v1..3v6, 0.33 mA op, 0.15 uA sleep
  - `1811170704_Sensirion-SHT21_C84828.pdf`
  - Read several sensors by using separate SDA lines for each (or power lines?)

Clocking

`swru191f.pdf`, p.66, sec. 4.4 "Oscillators and Clocks"states "Note that operation of the RF transceiver requires that the 32-MHz crystal oscillator is used."

Thus we plan for 32 MHz crystal oscillator, but will test using internal 16 MHz RC oscillator as well. It may be possible to calibrate 16 MHz RC oscillator using 32 MHz crystal oscillator and hope to run RF from the RC source from there onwards, saving some energy if not components.

- XTAL1 7V32000035 32 MHz crystal
  - 0.13 USD (ext) LCSC C90934
  - SMD-3225_4P package
  - `TXC-Corp-7V3205LCSC-EEE4G_C90934.pdf`

2.1.1 Connectors

DBG/PROG – debugging and flash programming via debug port, serial console

1. USART0_TX (P0_3) – Serial console, TX from MCU (RX of host)
2. DEBUG CLOCK (P2_2)
3. 3v3
4. GND
5. RESET_N
6. 5v or NC
7. USART0_RX (P0_2) – Serial console. RX of MCU (TX of host)
8. DEBUG DATA (P2_1)

```
---2x4_HDR------
TX    12   DEBUG CLK
3v3   34   GND
RST_N 56   DEBUG DATA
RX    78   5v
```

As can be seen, the power and ground pins are between high frequency data and clock wires to provide some resilience against cross talk.

1-wire (1x3 hdr)

1. Data
2. 3v3
3. GND

Consider merging with I2C.

I2C (1x4 hdr)

1. SDA
2. GND
3. SCL
4. 3v3

SPI / GPIO

1. MISO (P1_7)
2. GND
3. 3v3
4. MOSI (P1_6), 1w-data
5. SS1 (P1_4)
6. CLK (P1_5)
7. 5v
8. SS2 (P1_3)
9. P0_5
10. P0_4
11. P0_1
12. P0_0

```
---2x4_HDR------
MISO  12   GND
3v3   34   MOSI/1w
SS1   56   CLK
5v    78   SS2
P0_5  9a   P0_4
P0_1  bc   P0_0
```

2.1.2 Power

- Battery option: max 3v6 direct
  - Avoid voltage drop in regulator
  - Zener crowbar with polyfuse (PTC) and reverse voltage protection always applied
  - Problematic with Li-ion cell 4v2 – the extra capacity is quickly wasted by dischargnig via zener crowbar
  - LiFePO4: 3v2 jännitteellä on 5% kapasiteetista jäljellä, kun taas 3v6 jännitteellä on 90-95%, eli näin saadaan 85-90% akun kapasiteetista hyödynnettyä.
- External regulator: 3v6 regulator (or 3v3 if cheap)
  - Various batteries, including Li-ions that go up to 4v2
  - Ultracap?
  - Externally supplied power from wires
  - Apparently voltage drop is up to 1v3, which means that for Li-ion battery, the minimum useful range is cut. We are better off adjusting the energy harvester to produce 3v6 directly
- Energy harvester, boost only: TI BQ25504RGTT
  - Load driven from same source as battery, which limits battery maximum charge to the maximum voltage that the load can tolerate, i.e. 3v6. Since Li-ion lower voltage tends to be around 3v2, this leaves limited range and does not allow charging to full battery 4v2 level.
  - Using regulator to drop from 4v2 to 3v6 is wasteful and ultimately leads roughly to same end result as only charging to 3v6 with the disadvantage that regulator looses up to 1v3 in conversion, thus limiting the useful lowest charge.

- Capacitor would allow use of full 3v6 to 2v1 range
  - E=0.5CV^2; E = 0.5C(3.6**3.6-2.1**2.1); E=4.275*C
    - 47 uF: 0.2 mWs
    - 100 uF: 0.4275 mWs
    - 120 mWs: 28 mF = 28070 uF: Clearly supercapacitor (or battery) is needed

- Energy harvester, boost and buck: TI bq25570
  - Buck converter has the advantage of efficiently dropping 4v2 to 3v6. When CPU is sleeping and only microamperes are consumed, the buck converter's own efficiency becomes paramount ...
- Reverse polarity protection diode and thermal fuse (series fuse has voltage drop)
- Budget for peak current of 40 mA during 1 second (3v * 0.04 = 120 mW -> 120 mWs) every 3600 seconds. During this time the voltage is allowed to drop from 3v6 to 2v1.
  - 1051 Ws (0.0003 kWh) in a year
- Budget for sleep leaks current: 1 uA for MCU sleep_2 + 3x 0.15 uA for sensor sleeps + 2 uA for protection zener + 6.5 uA for tantalum cap = 10 uA. Seems finding low leak capacitor is most important. Seems also that the CPU sleep is sufficient: no need for FET to cut CPU consumption

3v6 (3v3 since it is cheaper) Protection

- Zener 3v6 LL-34: Semtech ZMM3V6 LCSC C8057 0.0189 USD ext
  - DZ1,DZ2: Zener 3v3 LL-34: Semtech ZMM3V3 LCSC C8056 0.0178 USD basic
    - `1812121407_Semtech-ZMM3V3_C8056.pdf`
    - Leaks max 2 uA
    - LL-34 is called Mini-MELF in kicad Diode_SMD library
- Fuse 100 mA (PTC resettable poly-fuse or similar)
  - 0603 Littelfuse 0603L010YR LCSC C207010 0.0958 USD ext

    - 1808072345_Littelfuse-0603L010YR_C207010.pdf
    - 300 mA, R-0.9..6 ohm -> max voltage drop 0v25
  - 2410 TLC Electronic SCT500mA LCSC C312001 0.0515 USD ext
  - 1206 Shenzen-lanson 12B1150B LCSC C182449 0.0735 USD ext
    - Shenzhen-lanson-Elec-12B1150B_C182449.pdf

- Power Cap
  - CP1 100u, 6v3 CASE-B_3528 AVX TAJB107K006RNJ LCSC C16133 0.238 USD basic
    - AVX-TAJB107K006RNJ_C16133.pdf
    - 1210 case
    - DCL 6.3 uA
  - 47u, 10v 1206 Samsung CL31A476MPHNNNE LCSC C96123 0.107 USD basic
    - Samsung-Electro-Mechanics-CL31A476MPHNNNE_C96123.pdf
  - Cstor1 4u7 C19666 CL10A475KO8NNNC 0603 Samsung Electro-Mechanics Basic 4.7uF 16V 0603
    - https://datasheet.lcsc.com/szlcsc/Samsung-Electro-Mechanics-CL10A475
  - Cref1 10n: LCSC C57112 basic

- High voltage cap for solar panel side Chvr1 Cvh1 C_hv1
  - C29823 1206B475K500NT 1206 Guangdong Fenghua Advanced Tech 4.7uF 50V 1206 Basic
    - https://datasheet.lcsc.com/szlcsc/Guangdong-Fenghua-Advanced-Tech-12

- Super Cap
  - 100 mF, 5v5 KEMET FG0H104ZF Mouser 80-FG0H104ZF 1.39 EUR
    - KEM_S6013_FG-1104289.pdf
    - Solder through, terminal spacing 5.08 mm
    - Series resistance (internal resistance): 100 ohm * 0.04 A = 4 V drop
    - Recommended charging: via 1k resistor?!?, time 30 min

  - 100 mF 5v5 Elna DX-5R5H104U Mouser 555-DX-5R5H104U 1.74 EUR
    - Elna-DXe-5945.pdf
    - Series resistance (internal resistance): 75 ohm * 0.04 A = 3 V drop
  - 50 mF 5v5 AVX BZ015A503ZSB Mouser 581-BZ015A503ZSB 12.62 EUR
    - bestcap-775665.pdf
    - Series resistance (internal resistance): 0.08 ohm * 0.04 A = 0.0032 V drop
      - Easily usable, but expensive component

- Power switch
  - VBAT_OK (positive level) should switch on load (load should be off during cold start)
    - Will this be actively driven low (N_FET?) when power is not OK? Can this be used as RESET_N signal for the RF_MCU
  - MCU in reset consumes? Can reset line connected to VBAT_OK work as "enable"pin?
    - MCU's own reset / brownout logic? Will this be enough so we do not need _VBAT_OK?
    - How strongly can power harvester drive the MCU reset?
  - Silergy Corp SY6280AAC LCSC C55136 0.12 USD ext
    - Silergy-Corp-SY6280AAC_C55136.pdf
    - 2v4 .. 5v5
    - Active high ena

Battry Charging

- PwrHarv1: Energy Harvester Boost only: TI BQ25504RGTT LCSC C157471 2.70 USD ext
  - Texas-Instruments-TI-BQ25504RGTT_C157471.pdf
  - VQFN16
  - Needs 80 mV, 330 mV cold start)
  - Quiescent 0.33 uA

- Vin 0.13 .. 3 V recommended, max 5.5 V
- Load should be connected to VSTOR (datasheet sec 8.3.2, p.11)
  - Internal bidirectional PFET (2 ohm) will connect / disconnect battery to protect it
  - If load exceeds VSTOR output, it needs to be switched off while bettery is charging. VBATOK can be used for that.
- MPPT solar 80% 4M42 + 15M62 (15 uA)
  - Basic: 4M7 (LCSC C23163) + 5M1 (C13320) + 10M (C7250) = 19M8 (76%)
- Program for 3v6 max output to battery and load Rov1 = 1.5**10_000_000**1.25/3.6 = 5M208
  - Where was it said that VBIAS is 1v25? Datasheet sec 7.5 "Electrical Characteristics", p.6
  - Closest are 5M1 (LCSC C13330 basic) and 4M7 (LCSC C23163 basic) and 3M (C23156 basic)
  - To make RSUM about 10M, we would use Rov2=5M1 as well
    - VBAT_OV=1.5**1.25**(1+5.1/5.1) = 3v75 V, which is too high.
  - Try 4M7+5M1: 1.5**1.25**(1+4.7/5.1) = 3.603 V, which is good
- Program for 3v3 output to battery Rov1 = 1.5**10_000_000**1.25/3.3 = 5M682
  - Try 3M+5M1: 1.5**1.25**(1+3/5.1) = 2.978 V, which is perhaps too low
  - Try 3M+4M7: 1.5**1.25**(1+3/4.7) = 3.07 V, which is better, but still low
  - Try 2M+1M5+5M1: 1.5**1.25**(1+3.5/5.1) = 3.16 V
    - 1M5: LCSC C4172 basic
  - Try 2M+1M5+5M1: 1.5**1.25**(1+(1.5+2.2)/5.1) = 3.23 V Good
    - 2M2: LCSC C22938 basic
- Undervoltage protection
  - Try 3M+5M1: 1.25*(1+3/5.1) = 1.985 V which is too low
  - Try 4M7+5M1: 1.25*(1+4.7/5.1) = 2.401 V which is to the high side but good enough
  - Try 4M7+5M1: 1.25*(1+5.1/4.7) = 2.606 V which is to the high side but good enough

- VBAT_OK_PROG
  - Try 5M1 + 5M1: 1.25*(1+5.1/5.1) = 2.5 V which is good
- VBAT_OK_HYST
  - Try 5M1 + 5M1 + 1M: 1.25*(1+6.1/5.1) = 2.745 V which is good
    - 1M LCSC C22935 basic
- Cref1 10n: LCSC C57112 basic

- Energy Harvester Boost and Buck: TI BQ25570
  - ti-bq25570-energy-harvester-ds.pdf
  - VQFN20
  - Needs 100 mV, 600 mV cold start
  - Quiescent 0.488 uA
  - 0.005 uA from battery when sleeping
  - Buck can deliver up to 110 mA

- 3v6 regulator 100 mA
  - U1: SOT-223 Adv. Monolithic AMS1117-3.3 LCSC C6186 0.1207 USD basic
    - 1811201117_Advanced-Monolithic-Systems-AMS-AMS1117-3-3_C6186.pdf
    - 1A, drops from max 15v to 3v3
    - Dropout voltage 1v3 max, less at smaller currents

- Battery Management
  - 1906261508_Nanjing-Extension-Microelectronics-TP4056-42-ESOP8_C16581.pdf
    - all chinese, but with interesting diagrams
  - SOT-23-6L Nanjing TP4057-42-SOT26-R LCSC C12044 0.107 USD basic
    - Nanjing-Extension-Microelectronics-TP4057_C12044.pdf – all chinese
  - Renesas ICL7660AIBAZA-T LCSC C7881 0.9276 USD basic
    - Renesas-Electronics-Intersil-ICL7660AIBAZA-T_C7881.pdf
    - Super voltage converter: from supply to positive and negative multiples

- DS1 Schottky diode, desire lowest possible voltage drop
  - https://datasheet.lcsc.com/szlcsc/1903051003_MDD-Jiangsu-Yutai-Elec-SS14
  - LCSC C2480 basic
  - JEDEC DO-214AC/SMA molded plastic body
  - 0.55 V drop
- CR2032 battery clip: Q&J CR2032-BS-6-1 LCSC C70377 0.1305 USD ext
  - 1811061923_Q-J-CR2032-BS-6-1_C70377.pdf

2.1.3 Antenna

- 2440 center frequency means 0.123 m wave length
- Crystal
- Wire monopole
  - L252, L261: 2nH
    - 2nH is not available as basic part. Closest seems to be 2.7nH. 2nH is available as ext.
    - C43365 HF Inductors SDCL1005C2N0STDF 0402 Sunlord Extended HF Inductors 2nH ŏ0b10.3nH 300mA 0402
      - https://datasheet.lcsc.com/szlcsc/Sunlord-SDCL1005C2N0STDF_C43365
  - Lbst1: 22uH
    - C169396 SWRB1204S-220MT SMD-PIND-12.5x12.5x5_2.9x5.4 Sunlord Extended 22uH ŏ0b120% 2.9A SMD,12.5x12.5x5mm(max) RoHS
      - https://datasheet.lcsc.com/szlcsc/Sunlord-SWRB1204S-220MT_C169396
- Dipole with balun
- Looped dipole
- Energy harvesting loop
- RF switch or analog mux

https://www.66pacific.com/calculators/small-transmitting-loop-antenna-calculator.
- use 1/8 wavelength conductor length (circumference) for highest efficincy

  - less than 1/4 wavelwngth to avoid self resonance

- 0.12295 / 8 = 0.015369 m = 15.4 mm circumference, 2mm wide trace

  - 58% efficient, Q=-1270
  - 495 volts RMS, 2.56 A
  - Tuning cap 0 pF

- 0.12295 m = 123 mm circumference, 2mm wide trace

  - non ideal conductor length, potentially self resonant
  - 100% efficient, Q=-2.1
  - 40 volts RMS, 0.053 A
  - Tuning cap 0 pF

- 0.246 m = 246 mm circumference, 2mm wide trace

  - non ideal conductor length, potentially self resonant
  - 100% efficient, Q=-0.173
  - 13.2 volts RMS, 0.013 A
  - Tuning cap 0 pF

- 0.492 m = 492 mm circumference, 2mm wide trace

  - non ideal conductor length, potentially self resonant
  - 100% efficient, Q=-0.0104
  - 3.17 volts RMS, 0.003 A
  - Tuning cap 0 pF

https://www.66pacific.com/calculators/full-wave-loop-antenna-calculator.aspx
- 306/2440 = 0.1254 m (full wave loop antenna, ca. Z=100 ohm)

harvest-rf-eucap2014.pdf

- RF switch DP4T (balanced SPDT) 3GHz Murata XM0860SR-DL0801 Mouser 81-XM0860SR-DL0801 1.30 EUR

  - XM0860SR-125132.pdf
  - 1v7 .. 4v

- RF switch SP3T 6 GHz MMIC RFIC SW373C LCSC C391972 0.2123 USD ext

  - 1912111437_RFIC-SW373C_C391972.pdf
  - 2v5 .. 4v

- RF switch SPDT 6GHz NJR NJG1806K75-TE1 Mouser 513-NJG1806K75-TE1 0.241 EUR

  - NJG1806K75_E-779934.pdf
  - 3v3

- RF switch SPDT Infineon BGSA11GN10E6327XTSA1 Mouser 726-BGSA11GN10E6327X 0.449 EUR

  - Infineon_BGSA11GN10_DataSheet_v03_02_EN-1226152.pdf

- RF switch DPDT cross Infineon BGSX22G5A10E6327XTSA1 Mouser 726-BGSX22G5A10E6327 0.703 EUR

  - Infineon-BGSX_22G5A10_E6327-DS-v01_00-EN-1493686.pdf

- RF switch DPDT cross 3.8 GHz Maxscend MXD8646A LCSC C285565 0.1634 USD ext

  - 1912121108_Maxscend-MXD8646A_C285565.pdf
  - 1v8 .. 3v3

- RF switch DPDT cross 3.8 GHz Maxscend MXD8546F LCSC C285564 0.161 USD ext

  - 1912121108_Maxscend-MXD8546F_C285564.pdf
  - 1v8 .. 3v3

- Diode NXP BAP50-03,115 LCSC Part C130411 0.0633 USD ext

  - NXP-Semicon-BAP50-03-115_C130411.pdf

Hirose UF-L connector

`KiCAD_PCB_Antenna_Lib.zip` `http://www.ti.com/lit/an/swra416/swra416.pdf`

2.4 GHz PCB antennas provided in `RF_Antenna.pretty` (`Texas_SWRA117D_2.4GHz_Left.kicad_mod` and `Texas_SWRA117D_2.4GHz_Right.kicad_mod`)

`ds/swra117d.pdf`

`https://e2e.ti.com/support/wireless-connectivity/other-wireless/f/667/t/114288` - Board thickness is 0.8mm - Thicker board requires shorter trace (shorten the trace with dremel to tune for 1mm board) - Ground strip should be quarterwave (300/2450/4=0.030m), not smaller

```
and not much bigger either
```

`http://referencedesigner.com/tutorials/si/si_06.php` - 1 oz copper is 0.035mm thick - relative permittivity of FR-4 material 4.5 - PCB 1.0mm thick, requires trace 1.8mm wide for 50 ohm - PCB 0.8mm thick, requires trace 1.44mm wide for 50 ohm - Wider lines or smaller separation, higher capacitance, lower impedance - PCB 1mm thick, with trace 0.50mm wide gives 93 ohm - PCB 0.8mm thick, with trace 0.50mm wide gives 86 ohm - PCB 1mm thick, with trace 0.25mm wide gives 116 ohm - PCB 0.8mm thick, with trace 0.25mm wide gives 108 ohm

`http://www.mattikaki.fi/` `http://www.mattikaki.fi/antennit/HB9CV.htm`

2.2 sens-rf-1w-2 - 1-wire to I2C bridge based solution

*** TBW

3 kicad and JLCPCB tutorial

> N.B. We choose not to use eagle because we are concerned about the future of the free to use version.

`https://kicad-pcb.org/`

As of 20201005 current version is **5.1.7** which was used in this document.

```
sudo add-apt-repository --yes ppa:kicad/kicad-5.1-releases
sudo apt update
sudo apt install --install-recommends kicad
# If you want demo projects
sudo apt install kicad-demos
```

https://kicad-pcb.org/help/getting-started/

/home/sampo/bldc-hardware/design

https://app.ultralibrarian.com/search?queryText=cc2530 (requires login)

Download zip, unzip the zip. In *KiCAD Preferences* -> *Manage Symbol libraries* import the library. I tried first as project library with folder button "Add existing library to table"near bottom.

The files from ultralibrarian unzip to KiCAD/YYYY-MM-DD_hh-mm-ss subdirectories.

In schema view use right hand "Place"button. Dialog to find the library appears. Browse to find the component and place the component.

- Use large surface mount pads R0603 for resistors and C0603 for capacitors. 0805 is also hand solearable, but 0402 is not (easily).
- Large caps and nearly all diodes will require something larger

https://support.jlcpcb.com/article/84-how-to-generate-the-bom-and-centroid-file-f

Their FAQ (https://support.jlcpcb.com/article/83-smt-assembly-faqs) states that there is 3 USD reel loading fee for each ext part.

3.1 kicad shortcuts and hotkeys

eeschema

```
C-F1    Help (this window)
Esc     Stop command, return to pointer tool
F1/F2   Zoom In/Out
F3/F4   Zoom Redraw/Center
Home    Fit on Screen
@       Zoom to Selection
Space   Reset Local Coordinates
A-BkSp  Leave Sheet
BkSp    Delete Node
Del     Delete Item
E       Edit Item
U/V     Edit Symbol Reference / Value / Footprint
```

```
C-e     Edit with Symbol Editor
R       Rotate Item
X/Y     Mirror X/Y
N       Orient Normal Symbol
G       Drag (grab) Item. Moves connected wires.
M       Move Schematic Item. Leaves wires behind.
Tab     Move Block -> Drag Block
Return  Mouse Left Click
End     Mouse Left Double Click
C-s/C-o Save/Load Schematic
C-f     Find Item
F5      Find Next Item
S-F5    Find Next DRC Marker
C-A-f   Find and Replace
Ins     Repeat Last Item
C-z/C-y Undo/Redo
C-c/C-v/C-x  Copy/Paste/Cut Block
C-x     Highlight Connection?
C       Duplicate Symbol or Label
A/P     Add Symbol/Power
W/B     Begin Wire/Bus
K       End Line Wire Bus
L/H/C-L Add Label Loca/Hierarchical/Global
J/Q     Add Junction/No connect flag
S       Add Sheet
Z       Add Wire Entry / Add Bus Entry
I/T     Add Graphic PolyLine/Text
F8      Update PCB from Schematic
O       Autoplace Fields
```

pcbnew

```
+/-    Switch to next/previous layer
PgUp/PgDn  top/bot copper layer
m/g    Move/drag wire or footprint
x      Route new track
v      Add through via
b      Update ground polygon pours
Delete Remove a trace or component. Hower deletes whole trace, click selects\
 one segment to del
```

```
n       Next grid size. Use with caution. There will be tears
        if you use a grid outside of 50mils or 25 mils.
/       Flip track posture
```

3.2 kicad tools and tips

kicad hierarchical sheets

- Use create hierarchical label tool inside hierarchical to create connection point
- Use that tool to populate anything found inside hierarchical to populate the pin on top level, then wire it
- Using same sch but different *sheetname* works for submodules and will not create a conflict
- Component names containing ? or ending in nonnumeral are liable to get automatically assigned
- In same-sch-different-sheetname scenario, the component names are apparently stored with the sheet level, i.e. it is possible to profitably share the sch and even edit it, yet the component names do not get confused, at least not in trivial cases.

cvpcb - assign footprints

- In general, it seems that with `5.x` kicad it is preferable to add the footprints already during eeschema phase using "F"footprint shortcut while hovering over the component
- Spend time to add modules from kicad default library in `/usr/share/kicad/modules/`
  - for some reason these are not already added by default
- Missing footprints: SHT21_SEN, RHA40_4P5X4P5
  - found as `.kicad_mod` files from ultralibrarian

- Seems you have to do in eeschema *Right-click-on-component->Properties->Edit footprint* (shortcut key "F") which shows "SHT21_SEN". Click Select. New window for navigating libraries comes up. Select first "Libraries"(a directory) from left and then "SHT21". Now footprint Text in the first dialog (the one with Select button) has changed to "Libraries:SHT21". Click Ok and move to pcbnew. Now when you do Update PCB from schematic, the footprint is found ok.

  - Footprints thus created, however, did not come out in CPL (Choose Pick Place) phase of generation of production files. Be sure to double check.

- Missing footprint Package_DFN_QFN:QFN-16-1EP_3x3mm_P0.5mm_EP1.8x1.8mm

  - Found: Package_DFN_QFN:QFN-16-1EP_3x3mm_P0.5mm_EP1.75x1.75mm_ThermalVia

pcbnew - Draw the PCB traces

- Check that clearance is 0.2 (0.25 prevents routing near dense components) and minimum track width is 0.25

  - Board Settings button on left -> Design Rules -> Net Classes.
  - Adjust according to manufacturer specs

- Clicking on one of the copper pads of a component allows Moving it. The other parts of the footprint move even if not selected

- Remove GND from netlist (`.net` file, use emacs and save removed part as `gndd.net` so you can add it back later) to avoid clutter in ratsnest or make ground pour first and connect the GNDs to it

3.3 Check List

1. Design rules wrt clearance and trace width
2. Copper pours rational
3. Orientation of components. Add dot, plus (caps) or minus (diodes) to silk screen.

   When in production, be sure to double check these as the reels may have different orientation (90° or 270° error) for the components or they may be upside

down (180° error). Red dot from JLCPCB should align with PCM silk screen white dot (pin 1). Also check minus signs of diodes and plus signs of caps with polarity.

4. Check placement (CPL file), esp. wrt. orientation. Make sure the column headers match manufacturer's idiosynchracies (i.e. different names than kicad generates). Edit manually as needed.

5. Check BOM, Any components to omit? If not in BOM, also eliminate from CPL.

6. Generate BOM

7. Generate Gerbers and check them using the gerber viewing tool. Extensions

```
.gbl  = Gerber bottom layer copper
.gbs  = Gerber bottom solder mask
.gm1  = Gerber milling (edge cuts)
.gta  = Gerber top adhesive
.gts  = Gerber top solder mask
.gtp  = Gerber top paste
.gto  = Gerber top silk screen
.drl  = Excelion drill file
```

   Besides the "gerbers"and drill file (excelion G-codes) that go to the `.zip`, you need to generate the commercial BOM file which is a spreadsheet typically in `.csv` or `.xls` format. BOM associates each component reference ("designation"in JLCPCB terminology) to a component orderable from a specific warehouse (LCSC for JLCPCB), with warehouse specific product number.

   Besides the BOM file, also CPL (Component Placement) file needs to be generated. CPL associates each component reference ("designation"in JLCPCB terminology) to center point coodinates, rotation, and PCB layer (typically "top"). kicad can generate CPL file in `.csv` format. This file can be easily handedited to fit the reqirements of JLCPCB (or probably any other manufacturer). Be sure to check for orientation errors, see (3).

8. In generating drill file, use settings "inches", "Decimal format", "Postscript"(JLCPCB requirement, may vary with other manufacturers)

9. If you have missing components or wrong placements, handedit designations, e.g. RF_MCU1, in CPL (`-cpl.csv`), looking up coordinates from `.rpt` (somethimes components, with their coordinates, appear there even if they do not

appear in the automatically generated placement). Often the Y axis is inverted, i.e. what is positive in `.rpt`, needs to appear as negative in CPL. Look at the near by automatically generated components to sanity check the missing coordinates you are supplying.

`.pos`

```
### Module positions - created on Mon Nov  2 15:08:01 2020 ###
### Printed by Pcbnew version kicad 5.1.7-a382d34a8~87~ubuntu18.04.1
## Unit = mm, Angle = deg.
## Side : All
# Ref     Val             Package                                          \
      PosX       PosY       Rot  Side
TH3       SHT21           DFN6                                             \
   26.6700  -147.3200  180.0000  top
TH2       SHT21           DFN6                                             \
  147.3200  -147.3200    0.0000  top
TH1       SHT21           DFN6                                             \
  271.7800  -147.3200    0.0000  top
RF_MCU1   CC2530F256RHAR  VQFN40                                           \
  223.5200  -135.8900  270.0000  top
## End
```

10. Hand delete from BOM undesired components (such as not available from LCSC or in extended catalog)

11. Hand delete from CPL components that are not in BOM

3.4 JLCPCB.com

- 3 USD mounting fee (2020 price) per extended component
- Max 10 extended components per project
- https://support.jlcpcb.com/article/68-instructions-for-ordering
    - Silk screen: 0.8mm high text, 0.15mm thick glyph lines and features
    - 200 x 250 mm max size for regular order
    - 0.15mm board outline cut, 0.2mm in other instructions

- https://jlcpcb.com/capabilities/Capabilities
  - Material: FR-4, dielectric contant 4.5 for double sided PCB
  - Max dimension: 400 x 500 mm
  - Board thickness: 0.4/0.6/0.8/1.0/1.2/1.6/2.0mm (finished board)
  - Finished Outer Layer Copper 1 oz/2 oz (0.035mm/0.075mm)
  - Drill Hole Size (Mechanical) 0.20mm- 6.30mm
  - Min via hole: 0.2mm
  - Min via diam. 0.45mm
  - Min. Non-plated holes 0.50mm
  - Min. Plated Slots 0.65mm
  - Min. Non-Plated Slots 1.0mm
  - Minimum annular ring: 1oz 0.13mm, 2oz 0.2mm
  - Minimum plater thruhole (PTH): 0.3mm (1 or 2 oz)
  - Hole to hole *clearance(Different nets)* 0.54mm
  - Via to Via *clearance(Same nets)* 0.254mm
  - Pad to Pad *clearance(Pad without hole, Different nets)* 0.127mm
  - Pad to Pad *clearance(Pad with hole, Different nets)* 0.54mm
  - Via to Track 0.254mm
  - PTH to Track 0.33mm
  - NPTH to Track 0.254mm
  - Pad to Track 0.2mm
  - H/HOZ (Inner layer) Trace width 5mil (0.127mm) Spacing 5mil (0.127mm)
  - 1oz (Outer layer)
    - Trace width 1/2 layers: 5mil (0.127mm) 4/6 layers: 3.5mil(0.09mm)
    - Spacing 1/2 layers: 5mil (0.127mm) 4/6 layers: 3.5mil(0.09mm)
  - 2oz (Outer layer) 8mil (0.2mm) 8mil (0.2mm)
  - Solder mask opening/ expansion 0.05mm The solder mask should have a minimum of a 0.05 mm "growth/mask opening"around the pad to allow for any mis-registration.
  - Solder mask color green, red, yellow, blue, white, and black. Green by default.

  - Solder mask dielectric constant 3.8
  - Solder mask thickness 0.010-0.015mm
  - Pad To Silkscreen 0.15mm The Minimum Distance Between Pad and Silkscreen is 0.15mm.

- On silksceen, be sure to mark polarity of diodes (-) and capacitors (+) as well as ICs (.)
  - `https://support.jlcpcb.com/article/99-does-the-red-dot-means-pin-1-in-th`
  - "8.About the +/- symbol in the DFM"

- `https://forum.kicad.info/t/jlcpcb-design-rules/22591/4`

3.5 JLCPCB Order 1 (20201102)

- Board size 40x250mm
- Qty5, 1 different design
- 0.8 mm thickness, HASL `w/lead`, 10z Cu
- gnumeric crashed when trying to save BOM (in `Win95/97/2000 .xls` format)

mkdir order-jlcpcb-20201103-sens-rf-1w-1-gerber cd `gbr-20201030/` cp `sens-rf-1w-1-kicad-B_Cu sens-rf-1w-1-kicad-B_Mask.gbs sens-rf-1w-1-kicad-Edge_Cuts.gm1 sens-rf-1w-1-kicad- sens-rf-1w-1-kicad-F_Cu.gtl sens-rf-1w-1-kicad-F_Mask.gts sens-rf-1w-1-kicad-F_Pas sens-rf-1w-1-kicad-F_SilkS.gto sens-rf-1w-1-kicad.drl ../order-jlcpcb-20201103-sen` cd .. zip -r `order-jlcpcb-20201103-sens-rf-1w-1-gerber.zip` order-jlcpcb-20201103-sens-rf-1w-1-gerber unzip -l `order-jlcpcb-20201103-sens-rf-1w-1-gerber.zip`

- gerber viewer feature of the web site did not work

PCB prototype:Y1-3299470A Gerber file - order-jlcpcb-20201103-sens-rf-1w-1-gerber_Y1 Build Time: 1-2 days PCB Qty:5 Price USD 6.80 + shipping

Order No:W202011030625472 AB mc Declined

CPL must be `.xls` or `.csv`, plain ascii not accepted

Header names must be Designator,Mid X,Mid Y,Layer,Rotation

Gerber Viewer PCB prototype:Y2-3299470A

Gerber file - order-jlcpcb-20201103-sens-rf-1w-1-gerber_Y2

Build Time: 2-3 days

PCB Qty:5

Layers:

Dimension:mm*mm

Different Design:

Delivery Fomat:Single PCB

PCB Thickness:

Impedance:

PCB Color:

Surface Finish:

Copper Weight: oz

Gold Fingers:Yes 45°

Flying Probe Test:

Castellated Holes:

Remove Order Number: 6.80$SMTAssembly$: $SMT020110395742Gerberfile-order-jlcpcb-20201103-sens-rf-1w-1-gerber_Y2BuildTime$: $3daysAssemblySide$: $TopsideBillofMaterial$: $ViewDetail$138.61

USD 156.21 incl. shipping

2020-11-04W202011040056560

3.6 Work Estimates

Estimate for sensor network board with 3-4 active components (easily a 2 week project)

- Schematic and reading datasheets: 1-3d
- PCB layout 3-4d
- BOM and order preparation 1d
- Shipping and customs: 2d, 1 week of real time)

3.7 Competition

- https://www.wellpcb.com/special/choosing-standard-pcb-thickness.html
  - PCB and assembly
- PCBWay - assembly palvelu jossa he ostavat omalla alennuksellaan Digikeystä halutut komponentit

4 Monitorointi

Koska kosteuden tiivistyminen ja jäätyminen ovat olellisia kuoren terveyteen vaikuttavia tekijöitä, uteliaat haluavat instrumentoida seinät, katon, ja lattian TH-antureilla. Ideaalitapauksessa nämä anturit saavat energiansa, ja kommunikoivat, radiolla ja ne rakennettaisiin pysyvästi seinän sisään. Toiseksi paras olisi ohuella kaapelilla kiinnitetty anturi joka rakennetaan seinän sisään.

8 vuotta ladattavalla kondensaattorilla toimiva anturi tai vaihdettavalla patterilla (8 vuotta kestävä) anturi voi tulla kysymykseen, mutta ne pitäisi voida vaihtaa seinän sisältä - tai sitten projekti julistetaan valmiiksi ja sähköttömät anturit vain hylätään seinän sisään.

Kuvassa TH-anturit ovat punaisia pisteitä. Niitä tarvitaan 27-36 kpl/seinä, katto ja lattia (kaikkiin nurkkiin ja kaikkien sivujen keskelle sekä ihan keskelle, sisäpintaan, ulkopintaan, ja keskelle), eli yhteensä n. 200 kpl. Voidaan myös säästää ja instrumentoida vain jokin seinä tai vähentää instrumentoinnin tiheyttä. Eristeiden väliset instrumentit saattavat olla interpoloitavissa ja niitä ei ehkä tarvita kaikkialle. Yksi täysin instrumentoitu seinä voi antaa osviittaa kuinka vähemmän instrumentoidut seinät kannattaisi interpoloida. Toisaalta eri seinät ovat erilaisia: toisissa on ulkoseinä vuorauksena, toisissa oma kipsilevynsä. Joissain seinissä on ovia ym. hankalia rakenteita jotka ansaitsevat omat mittapisteensä.

- 1-wire bus?
- RF powered PTH sensor – reults RF measurement devices, not ones powered by RF
- NFC powered PTH sensor
- Radio powered PTH sensor
- ZigBee Green Power, BLE, 6LowPAN

- Zero Power Wireless Sensor
- Wireless Sensor Network
- Energy harvesting
  - Energy Harvesting Transducer EMF Electro Magnetic Field
    - `https://eu.mouser.com/applications/storing_harvested_energy/`
  - Energy Processor
    - TI BQ25570 7.71 USD
      - `/t/ti-bq25570-energy-harvester-ds.pdf`
      - `/t/ti-sluuaa7a-energy-harvester-user-guide.pdf`
    - TI BQ25504RGTT LCSC C157471 2.70 USD ext
      - `Texas-Instruments-TI-BQ25504RGTT_C157471.pdf`
  - `https://www.digikey.com/en/articles/rf-energy-harvesting-batteries-not-i`
  - Wireless charging coil
    - `/t/wlc_rx_wr483245-15f5-g_en.pdf` 5.37 USD @ digikey
- Wearable sensors
- RFID Gen II
- AVR with ISM radio (one way is enough)
- PTH
  - evince `/t/ENG_DS_MS8607-02BA01_B3.pdf` &
    - 1v5 - 3v6
    - I_peak 1.25 mA
    - I_stdby 0.00003 mA (0.03 uA)
    - I_convert 0.78 uA
    - Conversion time: 0.56 + 0.56 + 2.12 = 3.24 ms
    - I2C 400 kHz
    - 4-wires: VDD, GND, SDA, SCL
    - calc 1ma**3v**30ms = 900 `uJ/read`. Some modules promise 4600 uJ, so this seems feasible
- HR202L Humidity Sensor (olevinaan .5 euro/kpl)

DS18B20

4.1 Protokolla

- Master polls each slave (requires two way radio)
- Slaves randomly wake up and transmit, ignoring collisions (slaves are TX only)
- Slaves randomly wake up and perform CDMA (requires two way radio)
    - After sending message, slave listens for a while so that master can send a message if required

4.2 Device Outline

Integrated factory made device would be best. Otherwise

- Sensor
- MCU
    - Radio interface
    - I2C (or SPI or ADC) for sensor
    - GPIO for one button (or three or four)
    - GPIO for one led (or three)
    - SWDIO, JTAG, or similar programming and debugging
    - RESET button
- Antenna for radio (crystal or PCB trace or `U.xx` connector)
- Energy harvesting chip
- RF harvesting transducer (may be combined with antenna?)
- Energy storage capacitor or similar
    - Option to instantly charge the cap
- 3 sensor multiprobe with just one radio and energy harvest?

`sparkfun.com`

4.3 One-Wire Bus (1-wire)

Nimestä huolimatta tarvitaan 2 tai 3 johtoa: GND, Power, ja Data (data ja power voivat olla sama, jos sensorissa on riittävä kondensaattori ja virran kulutus on riittävän vähäistä).

- Dallas Semiconductor
- Maxim
- Rochester Electronics

https://www.digikey.com/product-detail/en/maxim-integrated/DS18B20/DS18B20-ND/420
- 5.14 USD

https://eu.mouser.com/Search/Refine?N=4291596096&Keyword=1-wire -
MAX31889ALT+ temp sens. 2.61 USD

- /t/MAX31889-1840152.pdf

- MAX22520GWP+ digital sensor interface 3.53 USD

- /t/MAX22520-1710309.pdf

https://www.digikey.com/en/products/filter/humidity-moisture-sensors/529?s=N4IgTC
SHT30A-DIS-B10KS Sensirion_Temperature_Sensors_STS3xA_Datasheet.pdf
- 2.10 USD - I2C Hum/temp

Adafruit 4535 - HTS221 - 3.50 USD

Adafruit 4566 - AHT20 - 3.80 USD

SI7020-A20-GMR - 2.50 USD - I2C Hum/temp

https://www.sparkfun.com/products/16618 - AHT20 (ASAIR)

DS28E17 (1).pdf - 1-wire slave to I2C master bridge - 1 USD?, not avail in JLCPCB

DS2408S+ - /t/DS2408S_C143303.pdf

https://jlcpcb.com/parts/componentSearch?secondSortUuid=e9f8a4232722490d81048667f
Moisture Sensors&firstSortUuid=caefd6d6f2bc4507bdc2837246857fba

- Sensirion SHT21 2.90 USD (ext)

  - Hum/temp, I2C, 2v1..3v6, 0.33 mA op, 0.15 uA sleep
  - 1811170704_Sensirion-SHT21_C84828.pdf

- Goertek SPL06-007 0.96 USD (ext)
  - Barometric pressure sensor
  - /t/1907081118_Goertek-SPL06-007_C233787.pdf

- NRF24L01P-R proprietary 2.4 GHz TXRX com 1.23 USD (basic)
  - /t/Nordic-Semicon-NRF24L01P-R_C8791.pdf

- CC2530F256RHAR 15.4 2.4 GHz TXRX SOC 1.58 USD (basic)
  - /t/Texas-Instruments-TI-CC2530F256RHAR_C9120.pdf

- CMT2210LS-ESR 433 MHz TXRX com 0.21 USD (ext)
  - /t/HopeRF-Micro-electronics-CMT2210LS-ESR_C77204.pdf

- TDA7786M am/fm radio 2.56 USD (ext)
  - /t/1905131609_STMicroelectronics-TDA7786M_C391149.pdf

https://openlabs.co/OSHW/Raspberry-Pi-802.15.4-radio - Atmel AT86RF233, which is a 2.4GHz SPI-to-antenna transceiver IC with 802.15.4 hardware assist - Atmel-8351-MCU_Wireless-AT86RF233_Datasheet.pdf

https://jlcpcb.com/parts/componentSearch?secondSortUuid=3fcfe56c935a49c6adb8d73c5
Sensors - Accelerometers&firstSortUuid=caefd6d6f2bc4507bdc2837246857fba *
MMA8452QR1 1.17 USD NXP

  - /t/Freescale-Semicon-MMA8452QR1_C11360.pdf
  - 3-axis accelerometer 8g (gmax 5,000?!?, DS p.9)

* KXTJ3-1057 0.41 USD Kionix

  - /t/1911051531_ROHM-Semicon-KXTJ3-1057_C442603.pdf
  - 3 axis 16g

kicad tinycad

5 Software (firmware) for RF_MCU CC2530

Despite intel 8051 having harvard architecture (separate busses), the CC2530 implementation MCS51 core actually has large overlaps between address spaces of different busses and banks, see [CC2530UG] sec. 2.2 "Memory".

The executable code appears in its own memory space, of which lowest 32 KB is always flash bank 0. After reset, code starts executing from address 0x0000 in this bank.

Upper 32 Kbytes can be any selectable bank (any one of 1-7 that has 32 KB) of flash memory. However, if code execution from SRAM is desired, then SRAM will overlap the first 8 KB of the flash bank mapped there. Thus storing code in this area is less attracive, unless we wish to switch between the two models.

At any rate, since the 256 KB flash is partitioned to such inconvenient banks and especially since bank 0 can not be moved, we need to think very carefully what should be flashed where.

- Bank 0: boot loader, library, trampoline for bank switching
  - boot loader that looks at other banks to see which one should execute. It may check an integrity checksum and will choose the bank with highest ordinal number.
  - library has common library functions like *min_sprintf()* that rarely change
  - when application does not fit in single 32 KB (or just 24 KB if SRAM is mapped), a trampoline to jump from one bank to another may need to be implemented. Ideally this should integrate with SDCC mechanism for this case
  - interrupt handlers need to be in bank 0 lest things get too hairy

sdcc –codeseg option and banks, –model-small (default), –model-medium, etc., see [SDCCMAN] sec. "Memory Models".

- Use __addressmod to define custom address spaces for banks (p.36, p.41), see also sec. 4.1.3 "Bankswitching", pp.63-65.
- __data / __near: variable is in directly addressable RAM (first 128 bytes, default in small memory model)

- __idata: variable in indirectly addressable RAM (last 128 bytes), shared with stack (grows up)
- __xdata / __far: variable is in external RAM
- __sfr __at (0x92) _XPAGE; /* Texas Instruments (Chipcon) a.k.a. MPAGE */ (see p.62)
- Interrupt service routine: void timer_isr (void) __interrupt (1) __using (1) (see pp.44-45)
- `8051/MCS51` is little endian (but some CC2530 peripherals are big endian)

5.1 Flashing and Entering Debug Mode

Flashing is started by holding RESET_N pin low, sending 2 falling edge transitions on DEBUG_CLK pin, and then bringing RESET_N high, see [CC2530UG] p.51, sec. 3 "Debug Interface". After entering debug mode, DEBUG_DATA is used for synchronous serial communication: Data is set before causing raising edge on DEBUG_CLK and data is read on falling edge of DEBUG_CLK. Data is byte oriented with most significant bit first. t_dirchange_ = 83 ns, but programmer needs to check whether DEBUG_DATA is being pulled low before starting DEBUG_CLK to pump out the bits from the MCU. Fastest SPI clock is 125 ns high and 125 ns low, for total 250 ns clock period, i.e. 4 MHz. It may be safe to assume DEBUG_CLK to use similar frequencies, though the programmer serial interface may be slower. [SWRA124] p.4 states fastest clock period (for CC2511) as 42 ns, i.e. 23.8 MHz.

For all this, a hardware programmer is needed, but one can easily be built from Arduino or Black Magic Probe (BMP) using 3 GPIO pins and bitbanging (adapting SPI interface for purpose may also be possible). The PCB design already contains debug connector to facilitate such connection.

Besides the documented debug commands, the debug interface can feed, using DEBUG_INSTR command, an arbitrary instruction directly from wire to CPU and have it executed. This is quite powerful as it allows inspection and modification of any memory location or register.

```
0x10 0001 0XXX CHIP_ERASE
0x18 0001 1XXX WR_CONFIG
0x19 0001 1X01 WR_CONFIG [SWRA124], p.5, Table 2 "Debug Commands"
0x20 0010 0XXX RD_CONFIG
```

```
0x28 0010 1XXX GET_PC (2 output)
0x30 0011 0XXX READ_STATUS
0x38 0011 1XXX SET_HW_BRKPNT
0x3B 0011 1X11 SET_HW_BRKPNT [SWRA124]
0x40 0100 0XXX HALT
0x44 0100 01XX HALT [SWRA124]
0x48 0100 1XXX RESUME
0x4C 0100 11XX RESUME [SWRA124]
0x51 0101 0X01 DEBUG_INSTR
0x52 0101 0X10 DEBUG_INSTR
0x53 0101 0X11 DEBUG_INSTR
0x55 0101 0101 DEBUG_INSTR [SWRA124]
0x56 0101 0110 DEBUG_INSTR [SWRA124]
0x57 0101 0111 DEBUG_INSTR [SWRA124]
0x58 0101 1XXX STEP_INSTR
0x5C 0101 11XX STEP_INSTR [SWRA124]
0x60 0110 0XXX GET_BM
0x64 0110 01XX STEP_REPLACE [SWRA124]
0x68 0110 1XXX GET_CHIP_ID (2 output)
0x68 0110 1000 GET_CHIP_ID (2 output) [SWRA124]
0x80 1000 0kkk BURST_WRITE
```

Flash has 128 x 2048 byte pages for total of 256 KB, the smallest unit of erasure being 2048 bytes. Smallest unit of write is a 32 bit word. Reads from CPU go to flash directly at byte level, whereas erases happen at 2048 byte page level and writes must go through flash controller at 32 bit word level. Erase sets all bits of the page to 1, write can set some of them to 0. A 0 bit can not be set back to 1 except through erase. In all processing, flash is assumed to be write-once memory and no attempt, unless explicitly documented, will be made to use the second-write-can-flip-more-bits-to-zero behaviour.

Flashing is done with special BURST_WRITE debug command that can carry up to 2048 bytes of data. Each byte is written on its turn to DEBUG_DATA register. DMA must be set up, typically using DEBUG_INSTR commands, to transfere data from this register to FWDATA register from which flash controller writes to actual flash memory.

There is no similar burst_read -command, so validating content of memory may be a bit slower. The DMA from Debug Interface to FWDATA is triggered by DBG_BW event rather than by FLASH event. Care must be taken to not exceed flash speed (the FLASH based event would take care of the timing correctly, but would not handle the underflow at the DEBUG_DATA register.

An alternative, a more sophisticated, strategy would be to first transfer using DMA from DEBUG_DATA register to XDATA RAM memory region at speed triggered by DBG_BW. This could happen while the page erase is happening. Then second DMA would be triggered to move from XDATA RAM to FWDATA register triggered using FLASH events, ensuring correct flash timing.

After flashing, it is interesting to know if the operation was successful. Reading and copying, to the serial port or to the debug interface, of the entire block would be inefficeint. Instead, the flashing routine should do the reading and compute and send the CRC16 checksum, which should be sufficient for the sender to ascertain that the write was successful.

DMA descriptor for DEBUG_DATA direct to FLASH (with timing caveats)

```
SRCADDR = &DEBUG_DATA;  // read register, updated through debug burst write
DESTADDR = &FWDATA;     // write register
VLEN=0;      // Use LEN
LEN=2048;
WORDSIZE=0;  // 8 bit
TMODE=2;     // Repeated Single
TRIG=31;     // Debug interface burst write (DBG_BW), see [CC2530UG], p.99, tab
SRCINC=0;
DESTINC=0;
IRQMASK=0;
PRIORITY=2;  // High: DMA trumps CPU
```

Step 1: DMA descriptor for DEBUG_DATA to XDATA RAM (no timing caveat, during page erase)

```
SRCADDR = &DEBUG_DATA;  // read register, updated through debug burst write
DESTADDR = flash_buf;   // write memory in XDATA
VLEN=0;      // Use LEN
LEN=2048;
WORDSIZE=0;  // 8 bit
TMODE=2;     // Repeated Single
TRIG=31;     // Debug interface burst write (DBG_BW), see [CC2530UG], p.99, tab
SRCINC=0;
DESTINC=1;
IRQMASK=0;
PRIORITY=2;  // High: DMA trumps CPU
```

Step 2: DMA descriptor for XDATA RAM to FWDATA (no timing caveat, at flash speed)

```
SRCADDR = flash_buf;    // read memory buffer in XDATA
DESTADDR = &FWDATA;     // write register
VLEN=0;      // Use LEN
LEN=2048;
WORDSIZE=0;  // 8 bit
TMODE=2;     // Repeated Single
TRIG=18;     // FLASH, see [CC2530UG], p.99, table-8-1 "DMA Trigger Sources"
SRCINC=1;
DESTINC=0;
IRQMASK=0;
PRIORITY=2;  // High: DMA trumps CPU
```

5.1.1 DEBUG_INSTR bootstrap for turning on LED

This is simplest test to show that debug interface works and chip is alive. In particular, this avoids the DMA programming required for BURST_WRITE, thus ensuring success in the early part of the learning curve.

This can be extended to byte-by-byte DEBUG_INST based programming.

*** TBW

5.1.2 BURST_WRITE bootstrap for flashing

This is the "production"mode where BURST_WRITE with DMA is used. DMA source must be DEBUG_DATA register or in XDATA memory space. Destination is FWDATA register. Flash controller triggers DMA when it is ready to write.

*** TBW

5.1.3 Arduino Hardware Example

`https://www.instructables.com/ARDUINO-AS-A-8051-PROGRAMMER/`

`FXYKI38I4PHHHKP.ino` - Seems to be bit banging (despite calling the pins MOSI and MISO) - Most of the smarts seem to be in a windows program which is not available - Not necessarity directly applicable to CC2530

0xac 0x53 prog enable 0xac 0x9f erase chip

5.1.4 BMP Hardware Example

*** TBD

5.1.5 GDB interface

*** TBD

5.1.6 Links about flashing CC2530

`https://ptvo.info/how-to-select-and-flash-cc2530-144/` - CC-debugger or SmartRF04EB programming hardware

5.2 C development environment (SDCC + CCLib)

IAR EW8051 is bundled with all the required files for CC2530 to start development: - Seems windows based IDE

Open source alternative is SDCC (which includes SDCDB which can use ucSim s51 to simulate 8051) and CCLib + Arduino based flash programming solution. newcdb at `http://ec2drv.sourceforge.net/` is an effort to connect directly to the hardware.

CCLib - Arduino implementation of CC debug protocol * `https://github.com/wavesoft/CCLib` * git clone `https://github.com/wavesoft/CCLib.git`

cd `CCLib/Python` `./cc_info.py` -p `/dev/ttyS0` `./cc_read_flash.py` -p `/dev/ttyS0` –out=output.hex `./cc_write_flash.py` -p `/dev/ttyS0` –in=output.hex –erase `./cc_resume.py` -p `/dev/ttyS0` # Exit debug mode and resume normal execution

Analyzing source...

- `Arduino/CCDebugger.cpp` contains *CCDebugger::chipErase()* which issues the debug command 0x10
  - Enter debug mode step dance must already have been performed, see *CCDebugger::enter()*
  - Called by `Python/cc_write_flash.py`
- `Arduino/CCDebugger.cpp` *CCDebugger::enter()* manipulates RESET_N and DC to get to debug mode

- Called by `Python/cclib/ccproxy.py` *enter()* called by *chipErase()* and by *CCLibProxy::__init__()*
  - Called by `Python/cclib/ccdebugger.py` *openCCDebugger()*
    - Called by `Python/cc_write_flash.py` et al.

- `Arduino/CCDebugger.cpp` search for DEBUG_INSTR to see how machine code is fed to debug command DEBUG_INSTR (0x51 or 0x52 or 0x53). *CCDebugger::exec()* family of functions
  - Called by `Python/cclib/ccproxy.py` *instr()* family
    - `Python/cclib/chip/cc2510.py` – actual 8051 assy for performing the operations using DEBUG_INSTR. In particular: *writeFlashPage()* is interesting

```
routine8_2 = [

#; Initialize the data pointer
0x90, 0xF0, 0x00,  #MOV DPTR, #0F000H;
#; Outer loops
0x7F, (((words_per_flash_page)>>8)&0xFF), #MOV R7, #imm;
0x7E, ((words_per_flash_page)&0xFF),  #MOV R6, #imm;
0x75, 0xAE, 0x02,  #MOV FLC, #02H; // WRITE
#; Inner loops
0x7D, self.flashWordSize,  #writeLoop:     MOV R5, #imm;
0xE0,  #writeWordLoop: MOVX A, @DPTR;
0xA3,  #INC DPTR;
0xF5, 0xAF,  #MOV FWDATA, A;
0xDD, 0xFA,  #DJNZ R5, writeWordLoop;
#; Wait for completion
0xE5, 0xAE,  #writeWaitLoop: MOV A, FLC;
0x20, 0xE6, 0xFB,  #JB ACC_SWBSY, writeWaitLoop;
0xDE, 0xF1,  #DJNZ R6, writeLoop;
0xDF, 0xEF,  #DJNZ R7, writeLoop;
#set green led for debugging info (DO NOT USE THIS!)
#LED_GREEN_DIR |= (1<<LED_GREEN_PIN);
#0x43, 0xFF, 0x18,#      [24]  935         orl     _P2DIR,#0x10
#LED_GREEN_PORT = (1<<LED_GREEN_PIN);
#0x75, 0xA0, 0x18,#      [24]  937         mov     _P2,#0x10
#; Done with writing, fake a breakpoint in order to HALT the cpu
0xA5   #DB 0xA5;
```

```
]
```

```
    - This bootstrap does not seem to use DMA controller, but rather
      direct write to flash controller FWDATA register. DPTR intially
      points to 0xf000 which might be the input of burst write
```

```
   - See [SWRA124]
```

SDCC - Small Device C Compiler * `http://sdcc.sourceforge.net/` * Open source and actively maintained, cc2530 support at least since 2012 * sudo apt install sdcc – gives 3.5.0 from 2018. Latest as of 2020 is 4.0.0 * `/t/sdcc-src-4.0.0.tar.bz2`

```
  - sdcc-4.0.0/device/include/mcs51/cc2530.h
```

sudo apt install flex bison # versions in `/apps/bin` seem too old # bison: cannot open file '/apps/share/bison/m4sugar/m4sugar.m4': No such file or directory

cd sdcc-4.0.0 `./configure` –prefix=/apps/ # old gputils would need upgrading make # sdcc-4.0.0/device/lib/pic14/missing: line 81: aclocal-1.16: command not found autoreconf -f -i # did not work, still gives error as follows #configure.ac:33: error: Please use exactly Autoconf 2.64 instead of 2.69. cd `device/lib/pic14/` touch `aclocal.m4` configure touch `Makefile.am` `Makefile.in` # if they exist cd - make # touching the troublesome files did work

`https://sourceforge.net/projects/cmon51/` `/t/cmon51-1.2.zip`

5.3 Software Architecture

Following use cases and device roles can be identified

1. Measurement node that performs periodic RF-TX and listens RF-RX to receive occasional command. Sleeps otherwise. Bootloading via debug port. Bonus features:

    - Also read commands from serial port RX (wake on command support?)
    - Also print everything to serial port TX (in addition to RF-TX)
    - Also read 1-wire inputs, as a 1-wire slave, from GPIO (wake on command support?)
        - In response, write measurements to 1-wire GPIO pin

- Serial port based `bootloader/flaher`
- RF based `bootloader/flasher`
- 1-wire based `bootloader/flasher`
- New firmware and backup firmware both in flash (256KB divided by 2)

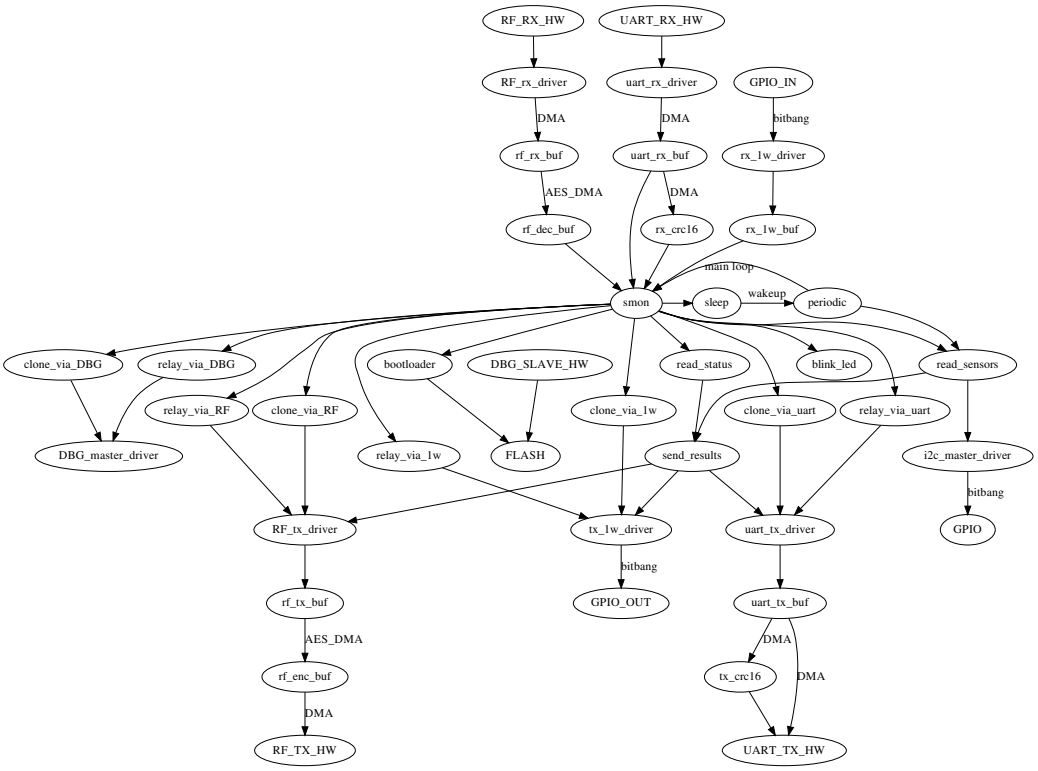
2. Master / control relay node

   - Serial port attachment (via USB adapter) to main host PC or Laptop
   - RF and/or 1-wire master communication with measurement nodes
     - relay results to serial port
     - send commands received from serial port to other RF and 1-wire connected nodes
     - Clown own firmware over RF or 1-wire
     - Accept CCdebug style commands from RF or 1-wire and relay them to DEBUG_DATA and DEBUG_CLK
   - Debug port master mode
     - Clown own firmware
     - Accept CCdebug style commands from serial port and relay them to DEBUG_DATA and DEBUG_CLK
   - `Stand-alone/detached` serial clone of own firmware

3. Host PC / Laptop control program for talking to (2) over serial port. User inteface could be similar to CCLib

4. CCLib enabled Arduino (or BMP?) and corresponding CCLib host tools like `cc_write_flash.py`

N.B. When writing flash, the code MUST run from SRAM, see [**?**] secs 2.2.1 and 6.2.1, p.73, or through DEBUG_INSTR. The latter method only only works through debug port and is no suitable for writing new image via serial port or RF. OTOH, [**?**] sec. 6.3.1 "Performing Flash Erase From Flash Memory"seems to imply that it is possible to perfom write operations from FLASH stored program: CPU will just block until flash operation has been performed (obviously changing the contents of flash from under executing program can lead to catastrophe).



Kuva 1: Sensor net monitor (smon)

5.3.1 Boot and memory banking

CC2530 boots from address 0x0000 which is in common memory (bank 0, always in addresses 0-0x7fff). FMAP is reset to 1 meaning that bank 1 appears in addresses 0x8000-0xffff. There are 7 banks of 32 KB, in addition to common bank 0.

MPAGE is reset to 0, i.e. page 0 of XDATA RAM area.

We would like a system where "application"code is flashed to one of the 7 banks such that it is possible to have new and old version of the code. Bootloader at 0x0000 will scan all banks, locating the bank with highest *prio* (u8 at 0x8003) and switch to it and then jumps to 0x8000. Avanced version may also check CRC16.

```
struct smon_image {
  U8 jmp;     /* 0x8000 op code for jump */
  U8 where_L; /* 0x8001 address of jump L */
  U8 where_H; /* 0x8002 address of jump H */
  U8 rel;     /* 0x8003 image release number indicating priority over previous
  U16 len;    /* 0x8004 length of the image (and header including crc16) */
```

```
  U16 crc16;  /* 0x8006 checksum for validating the image */
  U8 image[]; /* 0x8008 The image data. */
}
```

Interrupt vectors starting in common flash memory at 0x0000 (bank 0)

```
R  0x00 jmp RESET -- 3 bytes of space for jmp
 0 0x03 RFERR     -- 8 bytes of space for jmp or other processing
 1 0x0b ADC
 2 0x13 URX0      -- USART0 RX complete
 3 0x1b URX1
 4 0x23 jmp ENC   -- AES complete
 5 0x2B SLEEP TIMER
 6 0x33 P2INT     -- Port 2 interrupts
 7 0x3b UTX0      -- USART1 TX complete
 8 0x43 DMA
 9 0x4b T1        -- Timer 1 capture, compare, overflow
10 0x53 T2
11 0x5b T3
12 0x63 T4
13 0x6b P0INT     -- Port 0 interrupts
14 0x73 UTX1
15 0x7b P1INT
16 0x83 RF        -- RF general interrupts
17 0x8b WDT       -- Watchdog overflow in timer mode
   0x93           -- First free location
```

5.3.2 I/O, interrupts, and DMA

There are 5 DMA channels available, which is not very many considering the different I/O devices.

1. RF_RX or RF_TX clear text buffers and AES (via ENCDI or ENCDO).
2. AES to/from RF_TX (via ENCDI, RFD, and ENCDO registers)
   - RF_TX or RF_RX (can not simultaneously RX and TX, thus collision detection is not possible and RX is not needed during TX; instead Clear-Channel-Assesment (CCA) bit and backoff timeouts are used for Collision Avoidance)

- In effect the DMA (1) is chained with DMA (2).

3. UART RX
4. UART TX
5. ADC, FLASH write

5.3.3 Radio

RX and TX FIFOs are 128 bytes (8 x AES128 blocks). IEEE 802.15.4 frame length field maximum 127, from which we must deduce control and address fields somewhere between 5 and 25 bytes, thus pessimally the frame payload length can not exceed 102 bytes. 96 bytes would be 6 AES128 blocks.

192 us TX to RX and RX to TX turnaround time.

IEEE 802.15.4-2006 Frame Format

```
Preamble SFD Len [ FCF Seq AddrInfo Payload FCS ]
```

- SFD (Start-of-Frame Delimiter): 0x7a
- FCF (Frame Control Field)
  - Frame type? 0=beacon, 1=data, 2=ack, 3=mac_cmd
  - Sec ena
  - Frame pending
  - Ack req
  - Intra PAN / PAN ID compression: source and destination PAN_ID is same and only appears on source
  - Reserved-7-8-9
  - Destination addressing mode (DAM)
  - Frame version 12-13: 00=2003, 01=2006
  - Source addressing mode (SAM)
- Seq is 8 bit number
- AddrInfo contains source and/or destination address per addressing modes in FCF

Typical FCS is 001 1 0 0 1 000 10 01 10 meaning data + sec + no_pending + no_ackreq + Intra + res0 + 16bit-dst + 2006 + 16bit-src, seen other way 0b01001100 0b01100100 = 0x4a 0x64 = 0x4c64 (bigendian, "Ld").

Alternative for zero length dest addr (control node) would be 0x4c60 (bigendian, "L‘"). Without security, the values would be 0x4464 ("Dd") and 0x4460 ("D‘") respectively.

SAM and DAM can be 00 = zero length address (no PAM_ID nor address; elsewhere it is said that 00 indicates the PAN Controller), 01 = reserved, 10 (2) = address is 16 bit short address, 11 (3) = address is 64 bit long address. If Intra PAN bit is set, PAN_ID does not appear. Otherwise, 16 bit PAN_ID is prepended to addresses. We plan to use PAN_ID + 16 bit addresses, i.e. 4 bytes per address.

Use SRC match against PAN_ID and address 0xffff (any) to receive all frames.

Use SRC_MATCH_DONE interrupt to set up DMA from RFD to AES and another DMA from AES to clear text rf_dec_buf.

It seems there is no destination address matching logic in hardware. Hence, it seems advisable that the control node sets the source address of the packets it sends to the addressed node's address (i.e. use the destination node's address as source address), or just to broadcast 0xffff. Thus the sense of the source and destination fields seems inverted. The slave node sets source address to itself and destination to zero length, i.e. addressing the controller. However, if slave wants to address peers, then it sets source to broadcast and destination to itself.

Use DST match PAN_ID 0xffff (broadcast) and PAN_ID self to match only relevant packets.

Check AES DMA complete bit in main command loop to determine whether packet is ready

`https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation` * AES128-OFB: Output feedback mode is friendly to error correcting codes applied before encryption

```
- Stream cipher, i.e. no padding needed (but how to implement no padding in cc2
```

* AES128-CTR: Counter mode: Nonce-key pair must never be used twice due to XOR of streams * AES128-CCM: `https://en.wikipedia.org/wiki/CCM_mode`, defived from CTR mode, see caveat

5.3.4 1-wire

*** TBW

1-wire can be implemented with GPIO and bitbanging, but the allocated GPIO happens to be USART2 so it is possible to investigate some automation.

5.4 Commands

The communications on serial port, radio, etc. consist of messages with the following structure

```
L FCF SEQ DST SRC I O data FCS NL
```

where

- L is the 7 bit length including FCF and FCS and excluding itself. This is same as 15.4 Frame Length. Thus maximum length overall is 127 of which payload data level is ***
- FCF is the 16 bit frame control field per 15.4. For serial communications this is passed as 0x4464 (bigendian, "Dd").

  For radio this is typically 0x4c64 (bigendian, "Ld") or for zero length dest addr (control node) 0x4c60 (bigendian, "L‘").
- SEQ is one byte sequence number. By convention 0x20 is used if no sequence number is required.
- DST is the "destination"address in PPNN where PP 16 bit PAN_ID and NN is 16 bit node ID. In small networks the PAN_ID or even node ID can be chosen to be readable characters. It is also possible to avoid null bytes and newline characters (0x0a), but none of this is required - they are merely conventions to make reading dumps easier.
- SRC is the "source"address, i.e. slave node address in NN format, i.e. 2 bytes consisting of NN 16 bit node id. PAN_ID is not included here as PAN_ID compression is always used and thus PAN_ID always appears in the destination address (except the controller case?!?).
- I is request id byte. If message is a reply, then I matches that of the request. Using different I fields, it is possible to have several similar requests open simultaneously. By convention events that do not expect reply use id of "0".

- O is operation code byte which can indicate also that message is a reply. The convention is that requests use capital letter and their responses lowercase letter. Spontaneous messages also use lowercase letter.
- data is the binary payload data
- FCS (Frame Check Sequence) is the CRC16 checksum in big endian order for verifying the integrity of the data. For serial communication FCS is optional if NL or CRNL appears, see NL bullet, below. The eventuality that CRC16 might really have started by 0x0a or contain 0x0d 0x0a, is recognized and it is accepted that no integrity check is possible for messages with such CRC16. In 802.15.4 radio path, the FCS mandatory.
- NL is an optional New Line (NL, 0x0a)) character or CRNL sequence (0x0d 0x0a). The NL field does not appear in the radiopath.

Despite the optional NL and some debug messages that may be in plain text ascii making this appear line oriented protocol, all fields including O, I, and data are fully binary.

Example message

```
0e 44 64 20  6e 31 ff ff  65 32 30 58  59 cc cc 0a    .Dd n1..a20XY...
```

Here

- length is 0x0e, 14 decimal excluding length itself and the newline pad
- PAN_ID 0x6e31 "n1"
- dst node 0xffff (broadcast)
- src node 0x6532 "a2"(`slave/peer` node being addressed)
- requrest id "0", meaning spontaneous message (neither request nor answer)
- operation 0x58 "X"
- request payload 0x59 "Y"
- FCS (marked as `cc cc`, but not realistically calculated)
- Optional newline (0x0a) termination (common on serial port, not used on radio)

5.4.1 Summary of operations

N.B. By convention, operation verbs are represented with 0 (spontaneous) or 1 (command request) prefix where the prefix expresses the request id. The actual request id may vary according to the situation, but in this documentation we follow this convention so that it is easier to search the documentation as searching for single letters would be tedious.

- 0v = Verbose debug message. The payload is typically a readable ascii or latin1 string and terminating newline is highly likely to be used, possibly in place of FCS. Such debug message may be forwarded from radio domain to serial port connecting to laptop. The format of debug messages is

```
L FCF SEQ PAN DST SRC 0vFNLNdebugmsgFCS/NL
```

  where

  - *L+, +FCF*, *SEQ*, *PAN*, *DST*, and *SRC* are as usual. In forwarding debug messages from a field node to controlling node *SRC* will be of interest.
  - *FN* is the function name hashed into 16 bit binary number (bigendian). Computed at compile time by macro *AKBOX_FN(__FUNCTION__)*. This allows determination of source file as well, using tables prepared at build time. See command `log-pretty.pl -fn FN:LN`
  - *LN* is the 16 bit line number within the source file, as given by __LINE__ macro at compile time.
  - *debugmsg* is an arbitrarily formatted message, typically *sprintf(3)* formatted ascii, but could just as well be binary. By convention the first few letters will tell which kind of debug message is in question. This allows the kind to be indicated without having to resort to *FNLN* info, which is liable to change as the source code is edited.
  - *FCS/NL* is the frame check sequence on radio path and typically just NL on serial port

  When preparing formats for *debugmsg*, one should be frugal in order to not exceed packet length restriction, especially considering the possibility of forwarding and relaying overhead. The recommendation is to not to exceed

64 bytes (leaves space for two layers of forwarding or relaying). Excessively long message will be truncated. It will not be divided in two messages, but also not discarded entirely. Truncated message may be missing inner layer *FCS* fields.

- 0d = Debug message. Same syntax and considerations as for 0v.
- 0i = Info message. Same syntax and considerations as for 0v.
- 0w = Warning message. Same syntax and considerations as for 0v.
- 0e = Error message. Same syntax and considerations as for 0v.
- 0a = application layer ack message. Request id will correlate (instead of generic 0).
- 0n = application layer nak message. Request id will correlate (instead of generic 0).
- 1F = Forward to all other channels. Payload contains message to be sent. Forwarding is used typically when laptop sends over serial a message that should be forwarded to all nodes connected to radio and/or 1-wire. The forwarded message contains the L thru FCS fields as usual within the payload envelope. Multihop source routed forwarding is possible by multiple layers of encapsulation.
- 0f = Forward messages from field towards control node. Each forwarding node wraps the original message in 0f envelope. As a required setting, at least q, a, n, e, and w messages are forwarded. Depending on settings i messages and d messages are forwarded and in some cases v messages.
- 1Q = Sensor payload query. Response will be 1q. If sensor payload is spontaneously reported, 0q message is used. The response is an array of one byte tag and one byte value pairs where tags are "T"and "H"and values are signed 8 bit ints (two's complement) that represent Celcisus degrees in temperature (-120..120°C) and relative humidity (0-100%).

  A more rate tag "P"represents the air pressure in 10 x Pa, i.e. 0 = 0 hPa (total vacuum), 1 = 0.1 hPa = 100 milliBar, 10 = 1 hPa = 1000 mBar, 100 = 10 hPa = 10 bar, 250 = 25 bar = 25 atm.
- 1S = Status query, response is 1s. See sec. 5.4.2.
- 1Usn = Upload a new image at flash page *s+ (XCODE address s*1024, maximum 256 KB). The length of the image is +n* pages of 64 bytes. See section 5.4.3.

- 1F1Usn = Upload and forward image to all channels. Uses the Forward mechanism to propagate 1Isn.
- 1Cs = Clone image at flash page +s+ (XCODE address s*1024) to debug interface. The length of the image is in the smon_image header.
- 1Lrg = Set leds. r controls red (1=on,0=off,-=do not change), g controls the green. Only as many positions are supplied as needed. In future there could be more leds.

5.4.2 Status (1S)

- 1S = Status query. Response 1s.

  struct stat_rp

```
char req_id;    /* From request, typically '1' */
char op;        /* 's' */
char whyboot;   /* Reason for most recent boot: P=Power-on, R=Reset line
i8   T_core;    /* CPU core temperature -128..127 degrees Celcius */
u32  us;        /* microseconds since boot */
i32  ux_ts;     /* Wall Clock Time in seconds since Unix epoch */
u8   cur_flash; /* Flash page number of the currently running image */
u8   cur_rel;   /* Release number of the currently running image */
u16  cur_crc16; /* CRC16 of the currently running image */
u8   nxt_flash; /* Flash page number of the image that would be selected
u8   nxt_rel;   /* Release number of the image that would be selected in
u16  nxt_crc16; /* CRC16 of the image that would be selected in boot */
```

Reports time since boot, real time, reason for last boot (power on, reset line, watchdog, software reboot), which flash image is running (rel and page), which flash image has highest release number (rel and page), CPU core temperature, and other staus information. Can be composed to 1F1S to make the same query to all connected nodes.

5.4.3 Upload flash image (1Usn)

- 1Isn = Upload a new image at flash page +s+ (XCODE address s*1024, maximum 256 KB).

The length of the image is *n+ pages of 64 bytes. Since maximum packet length is much shorter than the required data length, the data will be sent as multiple 64 byte packets, with sequence number indicating the address:* `s1024n64`, for maximum transfer of 16 KB, thus it takes two such transfers for a single bank.

The new image should not be loaded over an existing working image, but rather in a free area or over an old undesired image. It is entirely possible that the upload fails in some way, thus there should be backup image that still works.

The new image contains *rel* the release number and crc16 checksum. If the checksum verifies and the rel is the highest number around, then in the next boot the image will be used. It is possible to jump to the new image even before the next boot.

If image is not completely transmitted, ... *** retransmit mechanism, query completeness mechanism

Viitteet

[CC2530DS] "CC2530F256 Datasheet (Rev. B)", Texas-Instruments-TI-CC2530F256RHAR_C9120.pdf, Texas Instruments, 2011.

[CC2530UG] "CC253x/4x User's Guide (Rev. F)", swru191f.pdf, Texas Instruments, 2014.

[SWRA124] "Debug and programming Interface Specifications", http://www.ti.com/lit/ug/swra124/swra124.pdf

[SDCCMAN] "SDCC Compiler User Guide"

[I15.4-2006] IEEE Std. 802.15.4-2006: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low Rate Wireless Personal Area Networks (LR-WPANs) http://standards.ieee.org/getieee802/download/802.15.4-2006.pdf (link removed by IEEE)