LUXPAK: Control

file: luxpac_control.doc
date: 15 June 08
cb

Index:

1	The LEGO® RCX brick
2	The SNOOP_LOG device
3	The bootloading process
4	The datalogging process
5	The RCX firmware
6	The "GASPERI" sensor multiplexer
7	The temperature control system
Annexe A	SNOOP_LOG schematics
Annexe B	"GASPERI" schematics
Annexe C	SNOOP_BURN
Annexe D	Mootz_board / Ozone sensor voltage stabilization

1. The LEGO® RCX brick

Since it is one of the purposes of the H.A.L.E. project to promote the LEGO® Mindstorms™ products by celebrating the 10th anniversary of the RCX brick, the LUXPAK team decided to use the RCX as the main control device for the payload. The NXT would have been the better choice for many practical reasons, but this also would have reduced the challenge and the thrill.

The RCX's missions may be described as follows:

- assure the logging of the scientific data
- control the payload heater system in order to guarantee that the devices, especially the sensors do work correctly
- survey the battery voltage and gradually switch off non-vital devices at low battery state
- allow easy human intervention on ground (battery install, system start, begin of the datalogging process, reset of the datalogger, upload data from the LUXPAK memory) ➔ Refer to the **LUXPAK User Guide** for more information

RCX connections: (cf. Fig.1 for connection colour code and directions of the leads)

- Sensor 1 (white) : ozone tube NTC thermistor (glued to the ceramic resistors)
- Sensor 2 (grey) : "GASPERI" input multiplexer
    - G_Port1 : Ozone box NTC thermistor
    - G_Port2 : Battery NTC thermistor
    - G_Port3 : RCX NTC thermistor

- Sensor 3 (green) : “Remove before launch”-button and green LED
- Output A (red): 3W-Heater A
- Output B (black) : 3W-Heater B
- Output C (yellow) : 800mW heater (beneath SNOOP_LOG), yellow LED and SNOOP_LOG “firmware presence” test pin

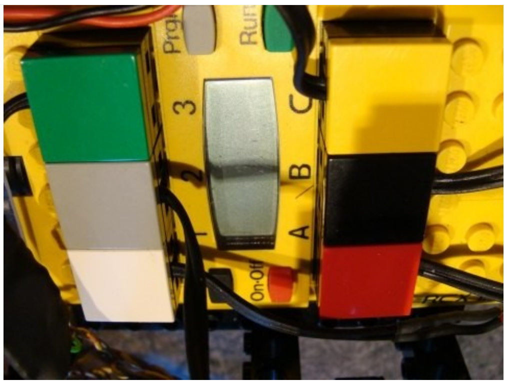


Fig. 1 : RCX connections (respect leads directions)

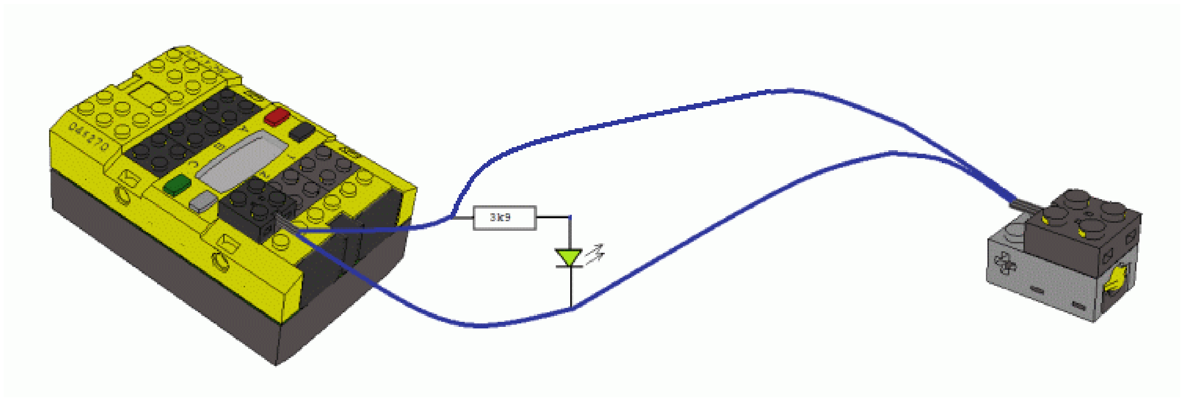


Fig. 2 : How the "Remove before launch" button and the green LED are connected to one RCX input port

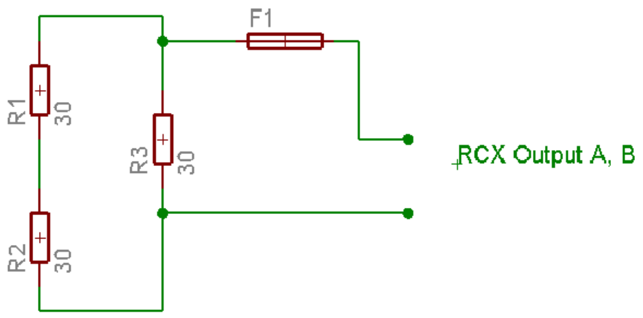


Fig. 3 : 3W-Heater configuration (a 91°C thermo-sensitive fuse is placed in series)

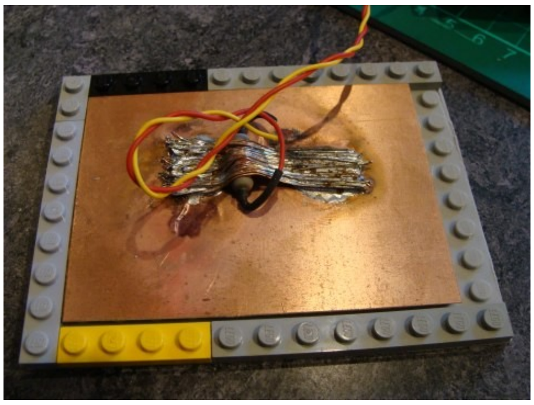


Fig. 4 : 800mW-Heater set-up. A 100R resistor is fixed to a copper plate.

---

2. The SNOOP_LOG device

The major argument against the RCX being used as the control board for the LUXPAK scientific measurements, is the lack of permanent memory. Without any flash memory extension, the RCX choice would inevitably lead to two major issues:

- complicated manipulation on ground before launch (the RCX firmware and control program need to be downloaded and manually run.)
- even if the RCX has excellent datalogging facilities, the expected hard parachute impact at the end of the flight could cause instant absence of battery power that would result in the loss of the RAM data

The LUXPAK team therefore decided to realize an additional EEPROM device responsible for the permanent storage of the data and for bootloading the RCX. The various aspects of the technology for such a device have been developed in a few preceding projects of the Konvikt LEGO® robotics group. Bringing all together:

- Infrared communication with the RCX. Preceding projects:
    - http://www.convict.lu/Jeunes/Mars_Mission_B/Mars_Mission_B_Main.htm (infrared/radio repeater)
    - http://www.convict.lu/Jeunes/Gaston/User_guide.htm (3D sound localization device)
- I2C communication between a Microchip PIC16F88 microcontroller and serial I2C EEPROM:
    - http://www.convict.lu/htm/rob/ir_us.htm (PIC/NXT I2C communication)
- RCX bootloader :
    - http://www.convict.lu/Jeunes/tiny_mini_worm/tiny_mini_worm.htm (the world’s unique RCX “virus”, a firmware that transmits itself to other RCXs.)
- Analog digital conversion on a PIC16F88:
    - http://www.elektor.com/magazines/2007/july-august/stereo-robot-ears.197459.lynkx (article by Claude Baumann and Laurent Kneip)
- Switching on the RCX programically:
    - http://www.convict.lu/htm/rob/mars_IV.htm

SNOOP_LOG combines all these functions. The schematics may be found in the annexe A of this document. SNOOP_LOG's functionality is described with the flow-chart in Fig. 6. The connections between the RCX and SNOOP_LOG are multiple in order to assure the main tasks:

- detect the presence of the RCX (measure the main logical voltage on the RCX via a wire that is connected to the RCX's infrared receiver power supply pins-marked "RCX 5V TSOP")
- physically switch on the RCX (via opto-coupler)
- download the firmware to the RCX (via infrared LED)
- operate the ADCs of 4 sensors referenced to 2V
- receive important data such as sensor readings and system time from the RCX (via infrared receiver)

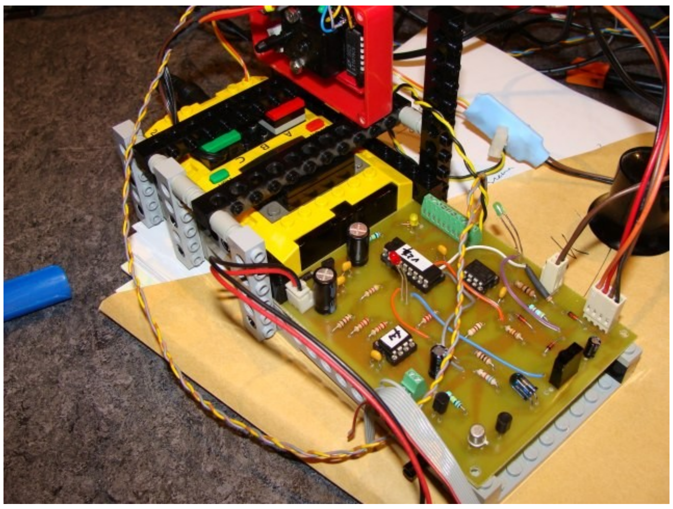


Fig. 5: How SNOOP_LOG is placed next to the RCX brick

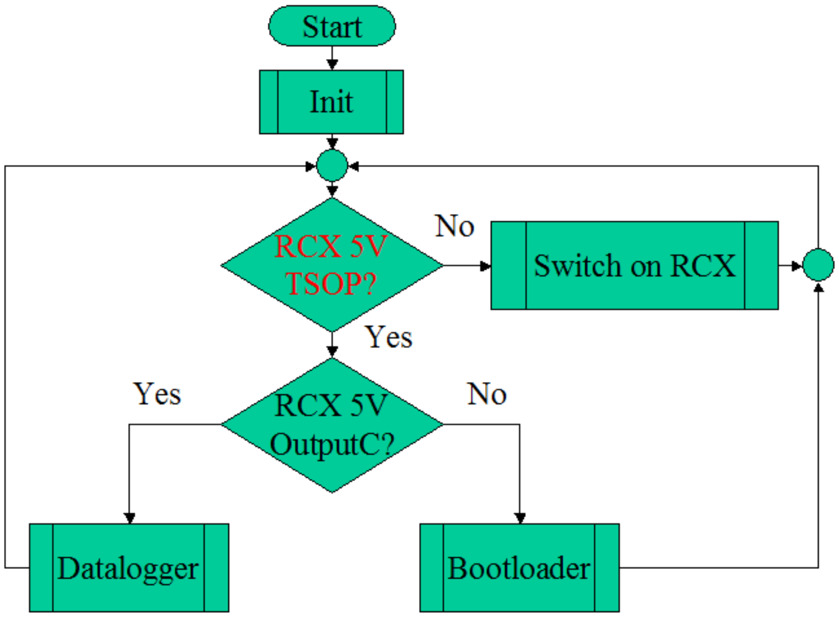


Fig. 6 : How SNOOP_LOG works

The SNOOP_LOG firmware has been realized in ULTIMATE ROBOLAB® for PICS, an experimental software, developed by Claude Baumann and Laurent Kneip in collaboration with the Centre of Engineering Educational Outreach

(CEEO) at Tufts University, MA. The SNOOP_LOG firmware (**version 25**) and all the related files can be found on the web-site.

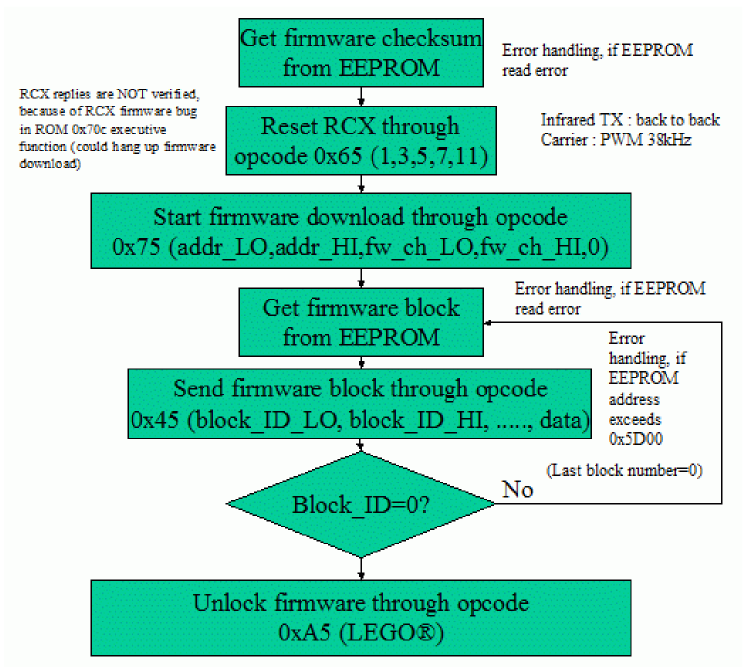
3. The bootloading process

Once the RCX is switched on, SNOOP_LOG checks, if it detects a TTL HIGH state on RCX PortC (denoted RCX 5V OutputC on Fig. 6), which is not given with the RCX ROM-executive, because the RCX outputs are switched off, while the executive is controlling the RCX. If the LUXPAK RCX firmware is running, PortC is immediately switched on and SNOOP_LOG executes the datalogging function, else it starts downloading the RCX firmware.

The bootloading procedure follows the flow-chart on Fig. 7. Note that similarly to the RCX-“virus” project, the LUXPAK bootloading process does not wait for any reply of the RCX, because of a resident RCX bug. At each sent package LUXPAK’s red LED blinks once. Bootloading lasts for about 5 minutes.

Error messages are communicated as follows:

- red LED rapidly flashes during a short moment, then blinks the number of times that corresponds to the SNOOP_LOG error:
    1. DEVICE ERROR 1 : I2C error (ACK missing); only happens, if there is no EEPROM mounted on the socket
    2. DEVICE ERROR 2 : UART error; only happens in datalogging mode, if an IR-message from the RCX was not well received; happens only extremely rarely
    3. DEVICE ERROR 3 : There was an attempt to read a firmware block beyond EEPROM address 24000; only happens, if there is no correct firmware in the fw-EEPROM
    4. DEVICE ERROR 4 : Datalogger full; only happens, if the datalogging process runs longer than 3 hours



Get firmware checksum from EEPROM
Error handling, if EEPROM read error
RCX replies are NOT verified, because of RCX firmware bug in ROM 0x70c executive function (could hang up firmware download)
Reset RCX through opcode 0x65 (1,3,5,7,11)
Infrared TX : back to back Carrier : PWM 38kHz
Start firmware download through opcode 0x75 (addr_LO,addr_HI,fw_ch_LO,fw_ch_HI,0)
Get firmware block from EEPROM
Error handling, if EEPROM read error
Error handling, if EEPROM address exceeds 0x5D00
Send firmware block through opcode 0x45 (block_ID_LO, block_ID_HI, ..., data)
(Last block number=0)
Block_ID=0?
No
Unlock firmware through opcode 0xA5 (LEGO®)

Fig. 7 : Bootloading sequence

4. The datalogging process

Once SNOOP_LOG detects the TTL HIGH state on RCX PortC, then the device switches to datalogging mode. It is a passive mode, since the control of the datalogging process has passed to the RCX.

The RCX sends the command to write a data-point to SNOOP_LOG's RAM as an ordinary RCX SETVAR instruction (opcode 0x14) with the 4 byte-parameters:

1. source (unused)
2. variable ID (0..11)
3. data_LO (dummy for variables 0..4)
4. data_HI (dummy for variables 0..4)

- If SNOOP_LOG receives variable IDs 0..4, then SNOOP_LOG samples its own ADC ports and stores the data in RAM (Note that variable 3 always reads 0x3FF, because the SNOOP_LOG analog port 3 represents the 2.078V reference voltage.
    - i. Variable 0 : Air pressure
    - ii. Var. 1 : Ozone concentration
    - iii. Var. 2 : Air temperature
    - iv. Var. 3 : Reference voltage (value always 0x3FF)
    - v. Var. 4 : Back scattered light intensity
- Variable 5 : system-time (seconds since start)
- Var 6 : tube heater NTC thermistor values (raw)
- Var 7 : "GASPERI" multiplexed NTC thermistor values (raw)
    - i. If an reading error occurred, the RCX adds 0x80 to the LO byte (LO byte first = little endian)
    - ii. G_Port0 value also is sent in order to identify eventual reading errors
    - iii. G_Port1..3 are identifiable through high nibble 2, 4 or 6 in LO byte
        1. G_Port1 : Ozone sensor temperature
        2. G_Port2 : Battery temperature
        3. G_Port3 : RCX temperature
- Var 8 : Sensor 3 value (might help find supply-voltage variations)
- Var 9 : OutputA state (LO : 1=fwd, 2=rev, 3=brake, 4=float; HI=power 0..255)
- Var 10 : Supply voltage
- Var 11 : 0x0D0A string (end of line constant :LF/CR); useful for restoring data later

The RCX sends the SETVAR command almost every 0.5 second. After 6 seconds, a data-record is complete and it sends a normal message with opcode 0xF7 and parameter 1, then waits 2 seconds. This message tells SNOOP_LOG to store the data-record to EEPROM and increment the record pointer, which is stored in the PIC on-chip EEPROM. If the data is well stored in EEPROM SNOOP_LOG shortly blinks the red LED once.

Typical SNOOP_LOG data:

ADC0	ADC1	ADC2	Ref.	ADC4	TIME	NTC1	GMUX	SEN3	OUTA	VOLT	LFCR
3403	1C00	8103	FF03	0400	0300	A401	3201	6400	01FF	9029	0D0A
3403	1C00	8103	FF03	0500	0B00	A601	9C22	6400	01FF	6F29	0D0A
3403	1D00	8103	FF03	0500	1300	A501	9C42	6400	01FF	6029	0D0A
3403	1C00	8103	FF03	0400	1B00	A601	5862	6400	01FF	5029	0D0A
3403	1C00	8203	FF03	0400	2300	A501	3201	6400	01FF	4229	0D0A
3403	1C00	8103	FF03	0500	2C00	A501	9B22	6400	01FF	3B29	0D0A
3403	1C00	8003	FF03	0500	3400	A101	9E42	6400	01FF	CC28	0D0A
…											

Table 1 : How SNOOP_LOG stores data in EEPROM

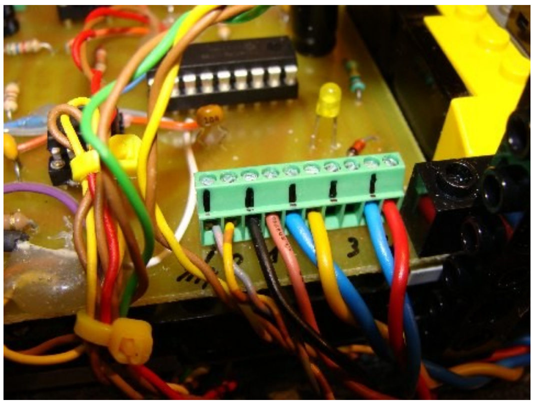


Fig. 8 : Sensor connections to SNOOP_LOG : 0=Pressure sensor; 1=Ozone sensor; 2=PT100; 4=Light sensor; black marks: always GND

5. The RCX firmware

The RCX firmware has been realized with ULTIMATE ROBOLAB®. By difference to most RCX programming environments, the outcome of this software is a firmware to the H8/3292 microcontroller that is at the heart of the RCX. (The current firmware **version 27** and all the related files can be found on the web-site.)

Once loaded, the firmware immediately starts running. The first thing to do is to power PortC in order to tell SNOOP_LOG that the firmware is correctly running. Then the RCX proceeds to a few self-tests. Errors that may appear are visibly and acoustically notified. (Refer to the User Guide for troubleshooting.)

With PortC powering, the RCX also supplies the 800mW heater beneath SNOOP_LOG. This heating will be active during most of the LUXPAK operation time. Now the firmware waits for the “remove before launch” protection panel to be removed from the touch sensor (cf. Fig. 9).

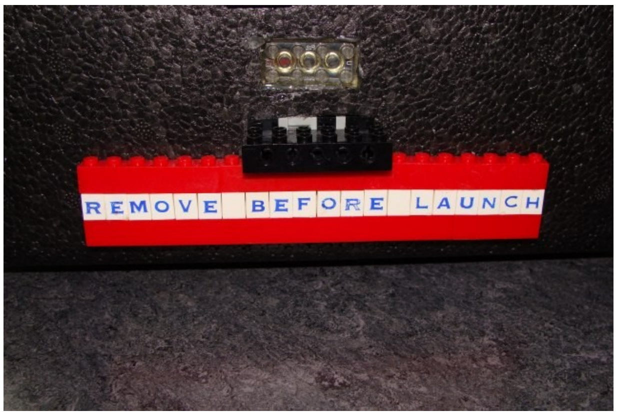


Fig. 9 : As long as this panel is in place, the firmware does not start the datalogging process.

LUXPAK's main switch needs to be activated half an hour before launch, because the ozone-sensor must be pre-charged at least for that duration.

➔ TIP: Once the panel is removed, the firmware starts the datalogging process. The system time is reset at this moment. If, by accident, the firmware was lost during flight, SNOOP_LOG would try to reinstall and run it. Then the datalogging would start immediately, since the panel would be missing. Note however, that in this case the system time would restart from zero. It is wise to reinstall the panel after flight, when trying to upload the data to the PC. (Refer to the **LUXPAK User Guide** for data upload.)

During the datalogging, the firmware executes the following tasks:

- Main task :
  - surveys the RCX button states and react accordingly
  - the RUN-button does not trigger any interrupt
  - the PRGM-button triggers the sending of an 0xF7 message with parameter 2, telling SNOOP_LOG to upload the data on the serial channel, while keeping the infrared carrier asleep
  - pressing both buttons at the same time (RUN-button a bit earlier) ➔ RCX sends out a few messages 0xF7 with parameter 0x39 that causes SNOOP_LOG to reset the internal EEPROM data-pointer.
    - ➔TIP: if this happened by accident, NO data has been wiped. Only the data-pointer is cleared. To recover the data, the data EEPROM must be taken off SNOOP_LOG and the data can be read by electronics means.
- Sensor2_NTCs task :
  - controls the GASPERI input multiplexer
  - reads the various interior temperatures and sets the heater power accordingly
  - detects reading errors on the GASPERI board (extremely rare)
  - convert NTC raw values to temperature
- Heater task :
  - Controls the heater
  - If all is well, sets the heater-power given asynchronously by the previous task.

    - If the battery voltage drops below 8.3V, then the 3W-heaters are not powered.
    - This is also the case, if the datalogging process had been kept running during more than 2 hours.
    - Anyway, during the first 10 minutes both 3W heaters are switched on at half the maximum power, in order to preheat the tube and the isolation material. The batteries are supposed to be quite fresh -just half an hour of moderate current drain happened before flight- and they should not be hot.
    - If the temperature in the tube exceeds 80°C –controlled by Sensor1 and a special conversion function-, the heaters are switched off.
- Sample task :
    - Controls the sampling process
    - Also surveys the batteries
        - If the voltage drops below 8V, the RCX starts saving battery power by switching off the green LED (RCX sensor3 to “unpowered”.
        - If the voltage continues dropping –below 6.7V- then the RCX stops the GASPERI sensor multiplexer, which also is a “powered” sensor.
        - Below 6.5V the RCX switches off PortC –and with it the firmware looses contact with SNOOP_LOG-. Now any interrupt handling is stopped, except that the H8/3292 watchdog-timer is started with the non-maskable interrupt (NMI) that should shut down the RCX and generate a reset, once it is triggered. The main program ends in an eternal loop, from which it can only be recovered by a system restart with fresh batteries. All these measures are necessary to prevent, that due to brown-out conditions, the microcontroller could end in unpredictable reactions.
- System task :
    - Runs in the background at 500Hz during 1ms
    - Serial opcode handler
    - Reads the battery state
    - Reads the RCX sensors
    - Reads the RCX buttons (ON/OFF=switch off the RCX; PRGM&ON/OFF= reset the RCX; VIEW=show various utile data values)
    - Controls the sound driver
    - Controls system and user timers

---

6. The “GASPERI” sensor multiplexer

This excellent device has already been successfully built into robot GASTON, where it has run for many years now without any fail. Since we changed two resistors from the original design, we will reproduce with permission the schematics in Annexe B.

The device changes the input port at each falling pulse of more than 0.5ms. The pulse is realized by software means, while reconfiguring the sensor port from “powered” to “unpowered”. The Gasperi-board multiplexes 4 ports (called G_Ports here) to one RCX input port. G_Port0 is used as the reference port to let the RCX software know, when a cycle starts.

The ports on Mike Gasperi’s mux are used as follows:

- G_Port1 : Ozone box NTC thermistor
- G_Port2 : Battery NTC thermistor
- G_Port3 : RCX NTC thermistor

The RCX recognizes the reference G_Port0, if it reads raw values below 340. The response of 10k NTC Digikey thermistors differ from the datasheets because of the RCX-internal 10k pull-up resistor and the mux circuitry.

The adjusted response curve is given in Fig. 10.

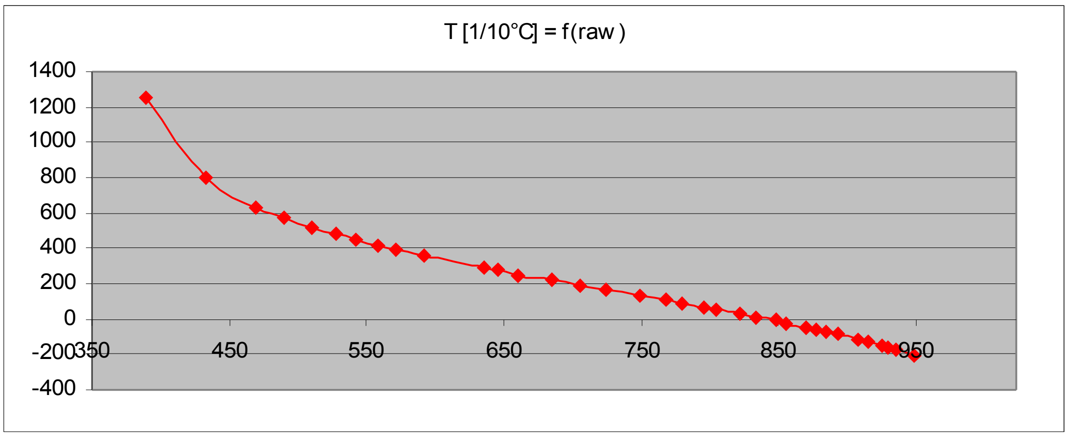


Fig. 10 : Non-linear response of the 10k NTC thermistors used in LUXPAK

Although ULTIMATE ROBOLAB has a built-in single precision floating-point mathematical kernel, it was decided to approximate the calibration function with fixed-point math for speed and memory reasons:

$$raw \leq 470 \Rightarrow T(raw)[1/10^{\circ} C] = -30 * raw/4 + 4121$$

$$470 < raw < 660 \Rightarrow T(raw) = -20 * raw/11 + 1450$$

$$\leq 660 \Rightarrow T(raw) = -31 * raw/21 + 1225$$

7. The temperature control system

During flight, LUXPAK will exposed to extreme cold. The excellent isolation characteristics of the polystyrene box have been studied in the document **luxpak_equip.doc**. LUXPAK has the following heat sources, thermometers and regulations:

- 800mW heater (beneath SNOOP_LOG)

    - No sensor monitors the temperature of this heater, because there is no risk of overheating due to the generous radiator size
    - The 800mW has no regulation. However at low battery state, the power is reduced.
- Two 3W heaters (wrapped around the ozone-sensor tubing)
    - An NTC thermistor is glued to the heating resistances; note this sensor is independent from the GASPERI input multiplexer.
    - The 3W heaters are powered through two electrically independent ports that however are driven together by software.
        - A hierarchical control regulates the 3W-heaters:
            - Lowest priority : ozone-sensor temperature; 2-point regulation with hysteresis (thresholds: 10.5°C and 11.5°C)
            - Medium priority : the output power is a linear function of the battery temperature [1/10°C]:
                - p=-17*battery_temperature/10+850 (ozone sensor needs heating)
                - p=-5*bat_temp/3+412 (ozone sensor doesn't need any heating) cf. Fig. 11
            - High priority : if the RCX temperature grows above 60°C, then switch off the 3W-heaters until the temperature is safe again (could lead to oscillations)
            - Highest priority : if the battery voltage is less than 8.3V switch off the 3W-heaters; (could lead to oscillations)
            - Priority plus : if the temperature in the tube reaches 80°C switch off the 3W-heaters
            - Super priority : if the system runs for more than 2 hours, then completely switch off the 3W-heaters and the heater tasks; during the first 10 minutes the 3W-heaters are powered at 50% duty cycle. We assume that there is no risk of overheating.
            - Fail-safe mode: the thermo fuses that are glued to the battery pack will melt at 91°C (+/-3°), if the previous protections fail. The 3W-heaters will not be recoverable.
- RCX :
    - the indicated heaters are powered through three RCX output ports ➔ the high currents heat the output driver circuitry
    - The maximum current of 1A is still far from the 1.5A fuse that is used in the RCX. The RCX output drivers have current limiting devices (max. 0.5A per port)
    - The "Mootz-board" (refer to **luxpak_equip.doc**) is situated in the RCX battery case to profit from this heat
    - An NTC thermistor is glued to the Mootz-board in order to monitor the temperature and protect the RCX from overheating
    - The Mootz-board has no hardware temperature compensation. Thus the PT100 data has to be adjusted by software means.
- Batteries :

- LUXPAK uses ENERGIZER L91 batteries (8x1.5=12V)
- These batteries tend to reduce the inner resistance at high current. They heat quite a lot and there is a serious risk of overheating.
- An NTC thermistor is glued to the battery pack in order to survey the temperature.
- Thermo-fuses (cut-off temperature 91°C+/-3°) are added to each of the 3W heater circuits. They are glued to the batteries in order to protect them from overheating (although the Energizer L91 batteries have a built-in PTC thermistor that acts as a current limiter, and with it the risk of overheating is limited.)

- Ozone sensor nose :
  - The air that already has been preheated through the tube heaters is also heated by the sensor nose.
  - One NTC thermistor is placed on the bottom of the ozone sensor case, in order to receive information, if the tube heater has to heat or not.
- The pressure sensor has a built-in temperature compensation

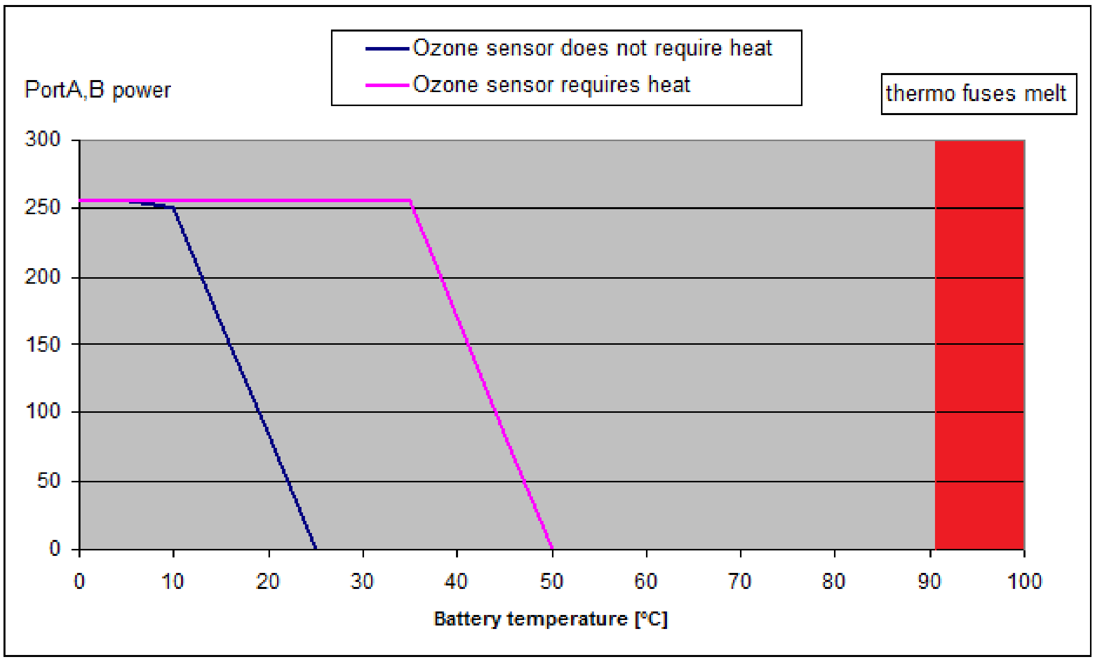


Fig. 11 : 3W-heater power in function of the battery temperature

The mux-independent NTC thermistor's response on the RCX Sensor1 port is given by the following calibration function:

$$raw \leq 90 \Rightarrow T(raw) = -31 * raw / 5 + 1444$$

$$90 < raw \leq 200 \Rightarrow T(raw) = -8 * raw / 3 + 1125$$

$$200 < raw \leq 850 \Rightarrow T(raw) = -raw + 790$$

$$850 < raw \Rightarrow T(raw) = -20 * raw / 9 + 1831$$

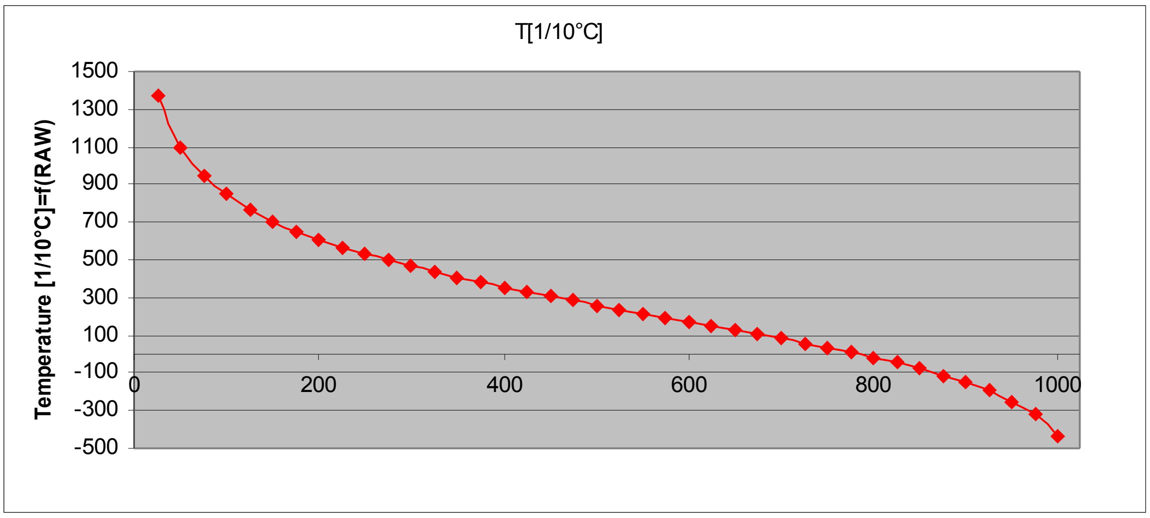


Fig. 12 : The non-linear10k NTC thermistor calibration function. The x-axis represent the RCX raw values

---

Refer to http://www.convict.lu/htm/rob/hale.htm in order to follow the development journal.

Annexe A : SNOOP_LOG schematics

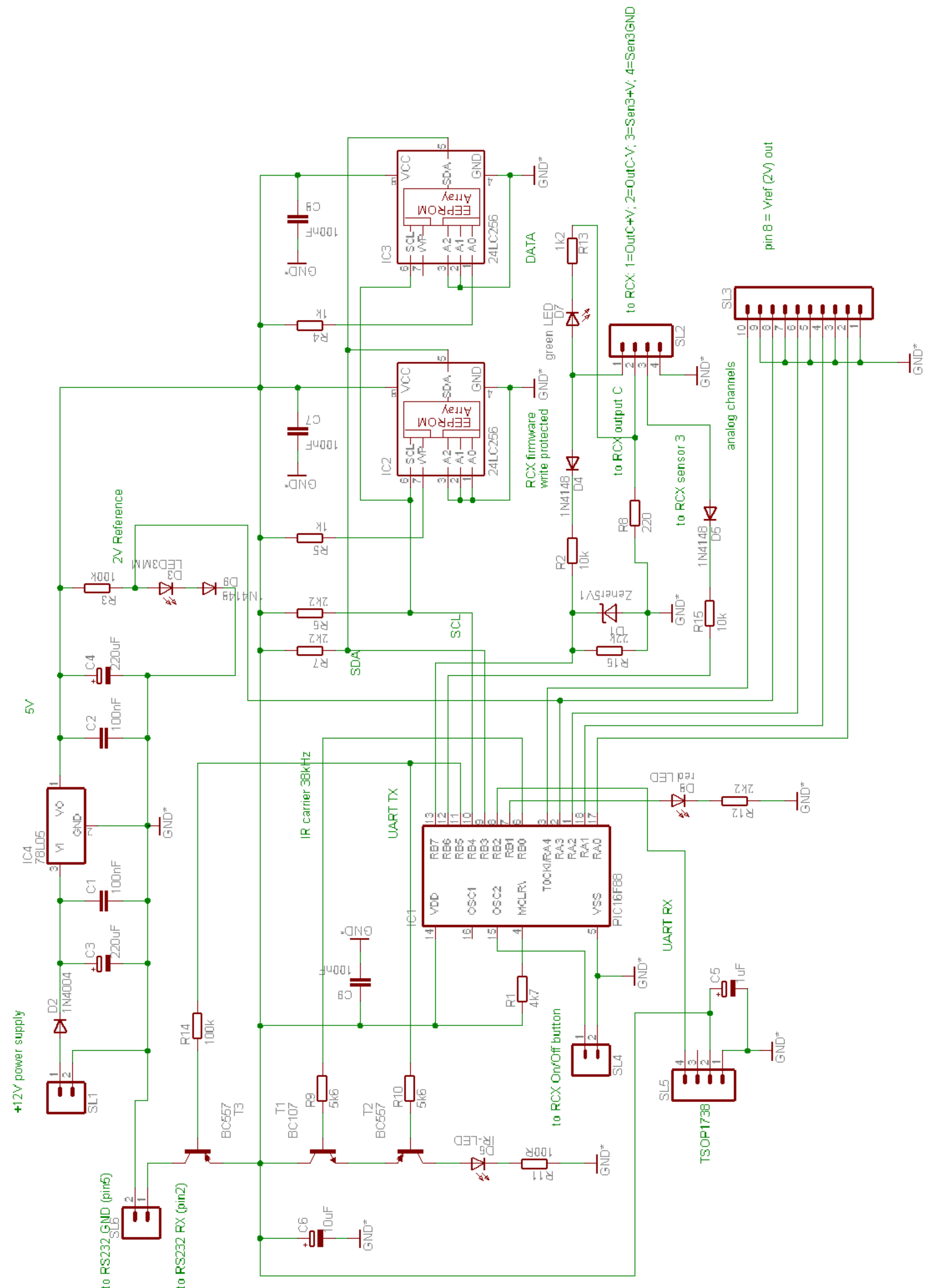


Fig. 1 : SNOOP_LOG schematics

Circuit functionality:

PIC16F88 is configured with internal oscillator at 8MHz. Thus no crystal is needed and Osc1 and Osc2 can be used as digital outputs. (Osc1 is not connected though.)

MCLR is configured as input by default in ULTIMATE ROBOLAB FOR PICS. To avoid floating input, the port is pulled up to Vcc.

T1 and T2 form a logical AND-gate and they are driven in saturation. The infrared LED D6 will only be switched on, if both transistors are conducting. If a serial signal is transmitted over the infrared channel, the 38kHz carrier (PIC PWM) must be active. Since the TX line is HIGH by in the case of a SPACE, and LOW in the case of a MARK bit, T2 must be PNP.

The collector of T3 can be connected directly to the RS232 RX line of a PC. The TX signals can always be monitored by a PC. This is particularly useful -besides the final data-upload- if debugging messages are being sent from the PIC via TX with carrier shut down. The normal infrared transmission is not affected by the additional debugging messages. (The actual PIC firmware v. 25 has no debugging messages anymore.)

D2, D4 and D5 are polarity protecting diodes only

In order to save energy, the red LED D8 is only switched on rarely, while the green LED D7 is constantly on (see below). The yellow LED D3 does not emit light, because of the 100k resistor. In series with D9 this LED is used to produce the constant reference voltage of 2V. (Note that the voltage drop through a LED does not depend on the current) All visible light LEDs are low current devices.

The resistor/diode bridge D1/D4/R2/R8/R16 produces a valid TTL voltage for the PIC input RB7 from one of the RCX output 9V voltages. R8 is necessary to decouple the device ground from the floating ground of the RCX output bridge.

The serial EEPROM IC2 contains the RCX firmware. The chip's Write Protection (WP) is pulled up to avoid accidental overwriting. Device address A2A1A0 = 0.

The serial EEPROM IC3 is going to receive the logger data. WP is kept unconnected, internally tied LOW. Thus the EEPROM can be written to. The device address A2A1A0=1

All sub-circuits are decoupled through capacitors.

Note that initially the volt-detection (detection of presence) of the RCX was wired to RCX Sensor3. Now pins 3 and 4 of SL2 are directly connected to the power supply of the RCX infrared receiver as shown on the next picture, where pin3 connects +V:

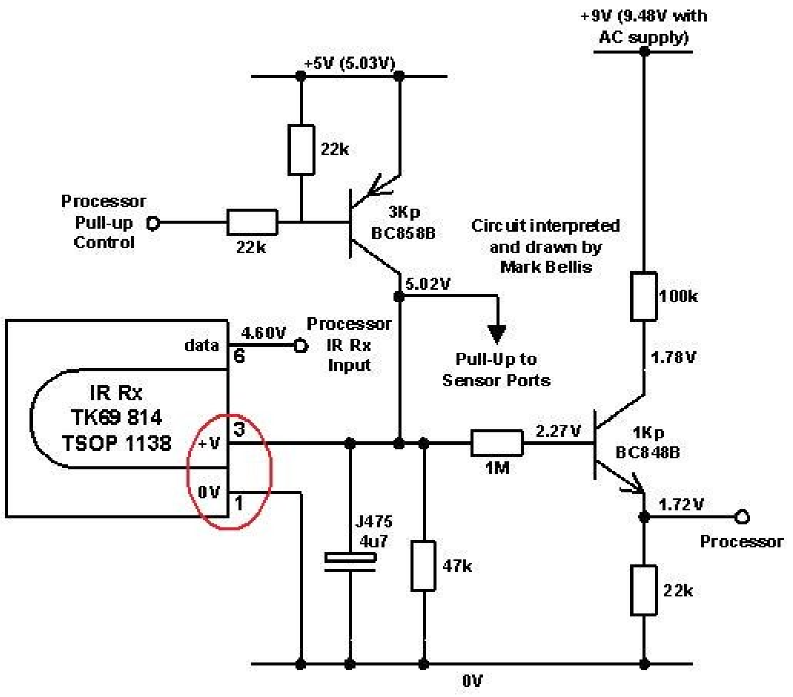


Fig. 2 : Where the RCX logical TTL voltage (HIGH=5V) is found (credits to Mark Bellis)

The RCX is switched on via opto-coupler as shown on Fig. 3. The presence of the firmware is detected by the voltage measured on RCX OutputC.

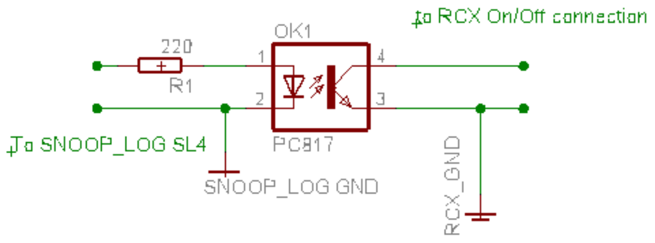


Fig. 3 : Opto-coupler between SNOOP_LOG and RCX On/Off circuitry

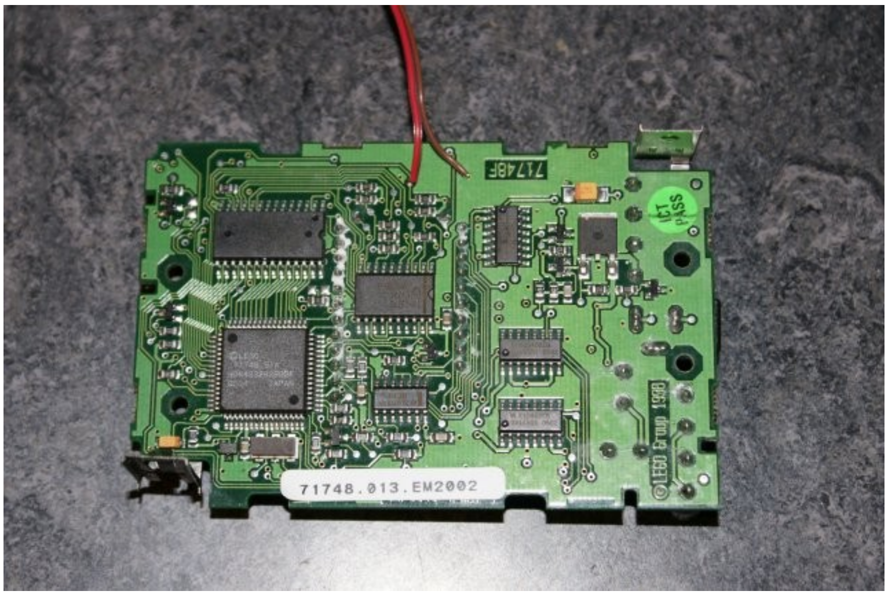


Fig. 4 : Red → Optocoupler pin4; Brown → pin3 (cf. Fig. 3)

Annexe B : Mike Gasperi's RCX input multiplexer

(reproduced with permission)

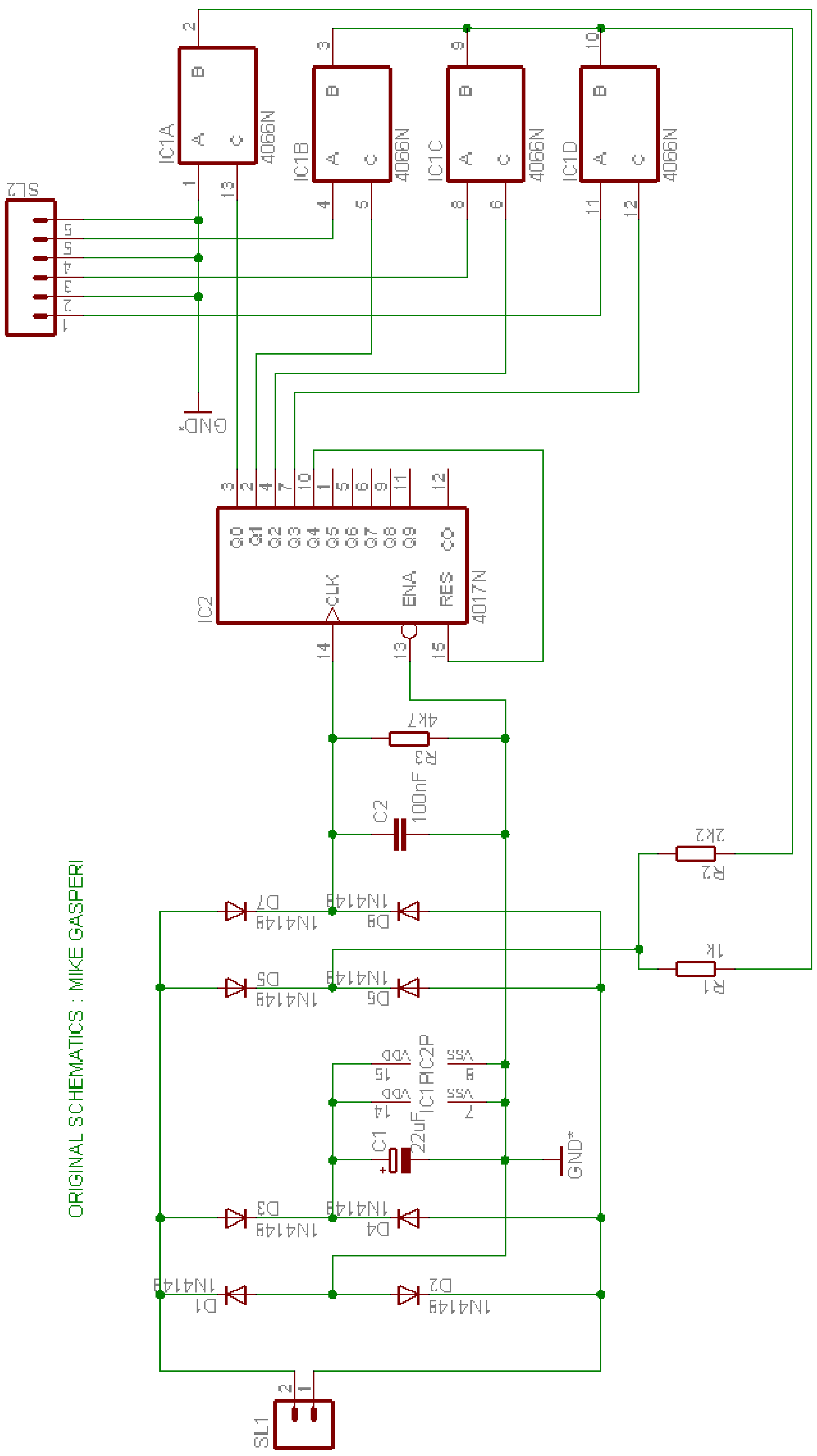


Fig. 1 : A very astute solution to multiplex a single RCX sensor port

Annexe C : SNOOP_BURN schematics

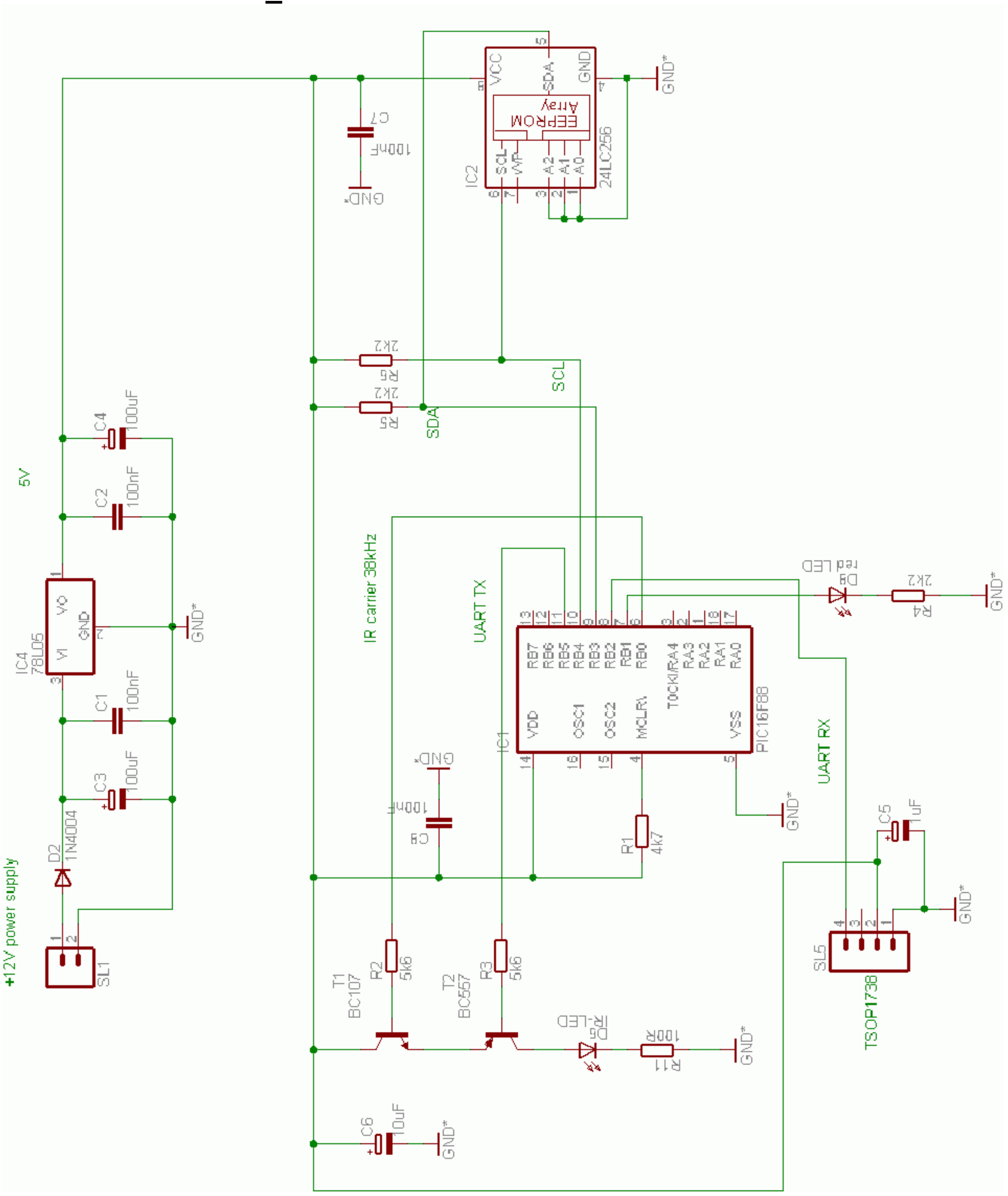


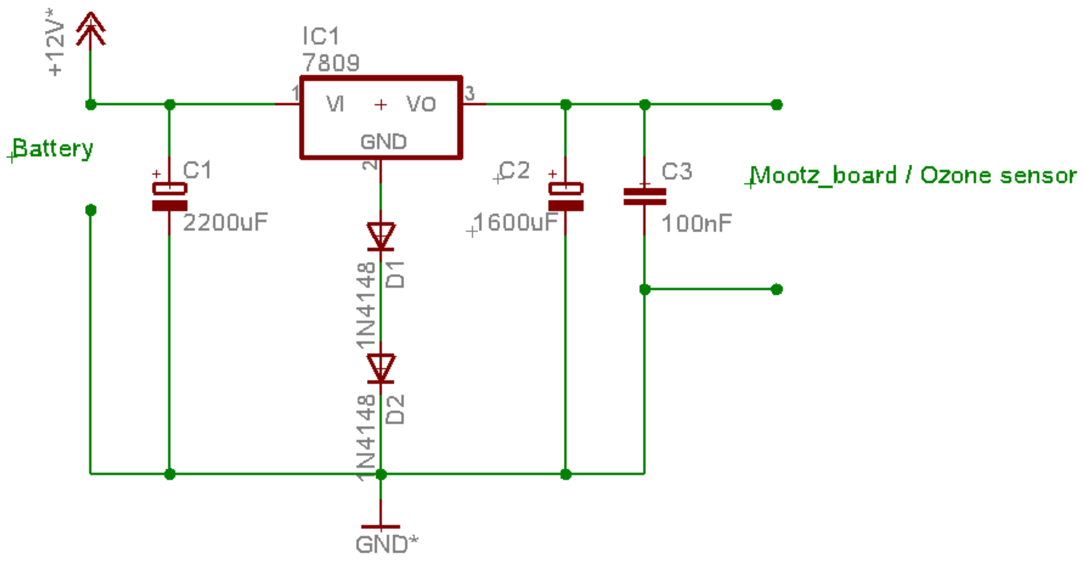
Fig. 1 : The SNOOP_BURN circuitry in fact represents a simplified SNOOP_LOG device. The major differences are in the firmware.

Firmware functionality:

- SNOOP_BURN is able to obey to the LEGO® RCX firmware download via IR-tower
- It reacts on the following opcodes (refer to the LEGO Mindstorms SDK documentation for further detail)
  - 0x10 (Ping)
  - 0x65 (Erase firmware)
  - 0x75 (Start firmware download)
  - 0x45 (Transmit firmware block)
  - 0xA5 (unlock firmware)

- Some differences to the normal download procedure must be observed:
    - The PC downloading program must divide the firmware into blocks of 89 data bytes, instead of the normal 200. The reason for this limitation is that the SNOOP_BURN stores each received byte, except the 0x55 FF 00 –header and the complement bytes, in its RX buffer of 96 bytes. 0x45 packets have the form:
        - 0x45 (0x4D?) / block_ID (LO) / block_ID (HI) / byte_count (LO) / byte_count (HI, always=0) / Data byte [0] / .... / Data byte [byte_count-1] / block_checksum
        - Since SNOOP_BURN also stores the 6 red coloured bytes as block information data, SNOOP_LOG later must not reproduce these values.
        - As usual, the downloader program has also to add the packet checksum (7), which is stored by SNOOP_BURN too. Thus the total packet size is 89+7=96 bytes.
    - The IR-tower must be configured with 200ms write and read timenout.
    - The downloading program must wait a longer time after the sending of each packet, because SNOOP_BURN needs a longer time to burn the data from RAM into EEPROM.
    - The data payload of the last packet must be filled with dummy zeros in order to assure the same packet size. (Note that the last block is marked with ID_number 0)
- SNOOP_BURN stores the firmware checksum in EEPROM at address 0x7FFE/FF (little endian).
- The current SNOOP_BURN firmware version 1.0 does not produce any specific error messages in the case of I2C missing ACK, since this error never appeared in any of the tests.
- UART-errors would lead to a bad or missing reply that the download software should be capable of detecting. In that case the lost packet should be resent. Note that it is essential that the download program respects the toggle bit. If by error the downloader sends out two successive 0x45 or 0x4D messages with different block ID numbers, SNOOP_BURN will not consider this as a bad thing, but later the RCX will, when an attempt of firmware download will be made from SNOOP_LOG with the EEPROM containing such badly marked packet opcodes. So, remember that successive block packets must alternatively use 0x45 0x4D, toggling the 4th bit!

Annexe D : Mootz_board / Ozone sensor voltage stabilization



This represents a typical voltage stabilization with one important change from normal: the two diodes in series that are added to the 2nd pin of the 7809 regulator increases the output voltage from 9V to 10.40V. The purpose of this additional stabilization is to reduce the influence of the voltage drop that is the result of the heater activity due to the high currents that are drained form the batteries.