Tiny Ground Station for a LoRa PocketQube

Goals:

Develop a system capable of receiving telemetry from a picosatellite in orbit. This Ground Station (GS) can be used as a main GS but it's mainly intended for a field operation sometimes used as a platform for IoT sink systems (e.g. meteo stations, multi sensor/actuator network gateway).

The GS shall receive a series of strings from a serial port connected to a PQube-ISTsatONE which is receiving telemetry data from a similar system supposedly in orbit. Each string corresponds to packets received by the receiving PQube and comprising several fields: I) packet number, ii) RSSI value, iii) SNR value, and a number of fields corresponding to values of sensors aboard like Temperature, Humidity and Acceleration; an extra field carrying values of pressure might be also present in these strings.

Such packets shall be stored in a circular buffer that stores the last 20 received packets, which may be accessed later through specific commands. Every packet shall be stored with a timestamp specifying the data/time when it was received.

Additionally, the GS shall concurrently collect values of room temperature provided by a sensor connected through I2C. Upon a command given on the systems' keyboard a PWM (Pulse Width Modulation) signal will be generated to activate a buzzer connected to an output port through an optocoupler.

The system shall be programmed through a console consisting in a 4x4 keypad and an LCD.

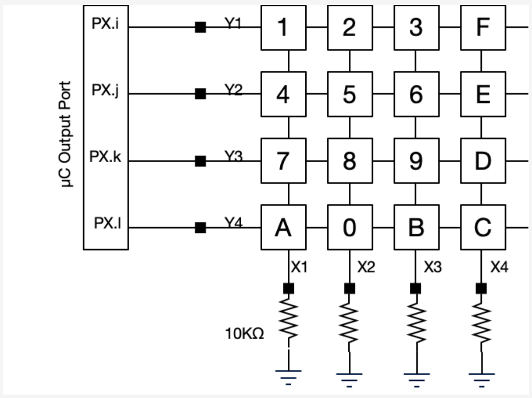
The console's LCD shall display continuously the instantaneous temperature measured by the room temperature sensor. Upon commands given on the keypad, it shall display: i) the number of received messages (strings) from the satellite; ii) an entire string which packet number has been provided; iii) the RSSI and SNR values relative to the packet whose number has been provided.

Functions:	After reset the systems waits for the setup of date and time in the format: MM-DD-YYYY hh:mm:ss. Then, upon reception of a command, it starts measuring temperatures and receiving data from the PQube. Commands: Programming the system ▪ Date Setup - key to be defined by the group ▪ Time Setup - key to be defined by the group Operating the system ▪ Start operation - key to be defined by the group ▪ # of received packets - key to be defined by the group ▪ Display packet # - key + packet number ▪ Display RSSI of packet # - key + packet number
I/O:	The system should comprise 2 microcontrollers, one receiving LoRa messages (PQube) and the other controlling the temperature sensor and the buzzer. The communication between them should be guaranteed by an asynchronous serial port (9600N8). The GS microcontroller board (TIVA) has to be connected to the following devices (interfaces in brackets): - Temp. Sensor (I2C) - LCD (11 GPIO max.)* - 4x4 Keypad (8 GPIO)* - Buzzer (1 GPIO with an optocoupler) * - see below the details of such connections
Performance	All activities must occur concurrently. FreeRTOS programming is required for assuring concurrency and the correctness of time accounting (Real Time Clock) in the TIVA board. It's important not to forget to add two 8,2 KΩ pull-up resistors to the I2C lines. The I2C bus speed is highly dependent on these resistors and some devices need to use lower values for these pull-ups.

Obs.	Groups shall start by developing simple functions to: - read a key from the keypad (careful w/ debouncing); - write ASCII characters to the LCD; - get a temperature reading from temperature sensing; - issue simple commands to activate the buzzer (optocoupler interface is needed); - receive strings from the PQube.ISTsatONE. Then, these individual functions can be merged together to form a simple working system, yet without concurrency (if one keeps a key pressed the system will stop). At this stage, a function for counting time should be created. The final step consists in transforming functions (or some groups of functions) into tasks that can run concurrently (the system continues to work even though one keeps a key pressed) and build a Main function that creates the tasks, the IPC mechanisms and launches the system.

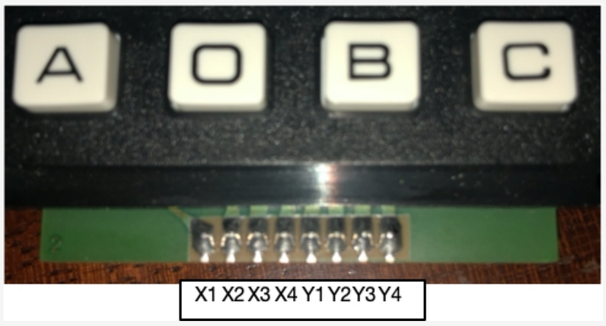
Hardware details - Keyboard

The keyboard is a simple contact matrix with 4 rows and 4 columns. Each key (a press button), when pressed, connects a row with a column, as depicted in next figure.



The microcontroller should output a combination of bits (e.g. 1000 for row Y1, 0100 for row Y2, etc.) to check whether any key in that row is pressed. The key is given by the position of a '1' in the 4 bits read from the input port connected to the terminals X1-X4 of the keypad. Supposing the µC output port has [Y1,Y4]=0010 and the reading of the input port gives [X1,X4]=0001 then the key being pressed is 'D'. Note that an output of [Y1,Y4]=1111 can be used to check if any key is pressed (if [X1,X4]=0000 there are none) but the exact key cannot be determined this way. For that we need a continuous scanning through all the rows, one bit at a time.

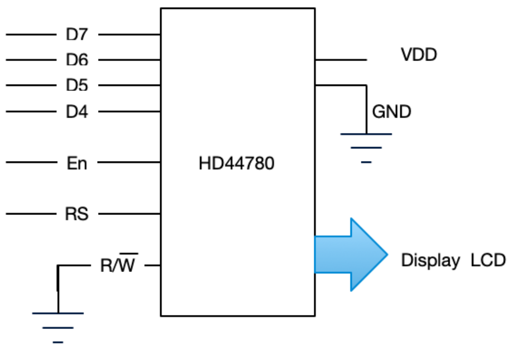
The keyboard pinout is shown below.



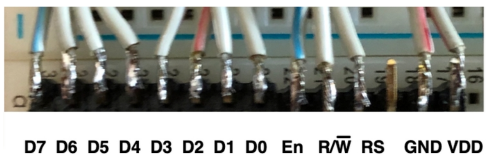
To facilitate the interface between a microcontroller and the LCD display, Hitachi developed the HD44780 module containing a controller capable of handling control instructions or data to be writing on the display.

The module allows the writing of a line with 20 characters on an LCD display, each character being written with 40 points 5(columns)x8(rows) according to a standard available in ROM containing 240 symbols.

To write on the display it is necessary to send a configuration instruction followed by an instruction containing the ASCII code of the symbol to be written. The 4-bit interface mode is used here.



The 4-bit interface mode is used here. The microcontroller-HD44780 connection diagram is shown in the figure and is implemented as shown below.



Since R/nW=0, the interface is write-only. If RS (Register Shift)=0, the transmitted data are for the display control instruction register (IR: 8-bit Instruction Register). If RS=1, the data is to be written to the data register (DR: 8-bit Data Register) in order to be transferred either to the display RAM (DDRAM - Display Data RAM) or to the character generation RAM (CGRAM). The 8-bit writing to any of the registers is done in 2 steps, with 4 bits at a time, starting with the most significant nibble. The writing of each nibble is performed on the positive edge of En.

The HD44780 has a 7-bit counter that stores the address of the next DDRAM or CGRAM position. The DDRAM has 80 positions of which only 20 are used in the display. In the supporting website one can find more detail about this device as well as code (LCD.c) that can be adopted (from a PIC) for coding the LCD driver.

Packets transporting telemetry data from the orbiting satellite have a defined format. They are received in the GS which displays the following information in the transceiver (receiving PQube) console:

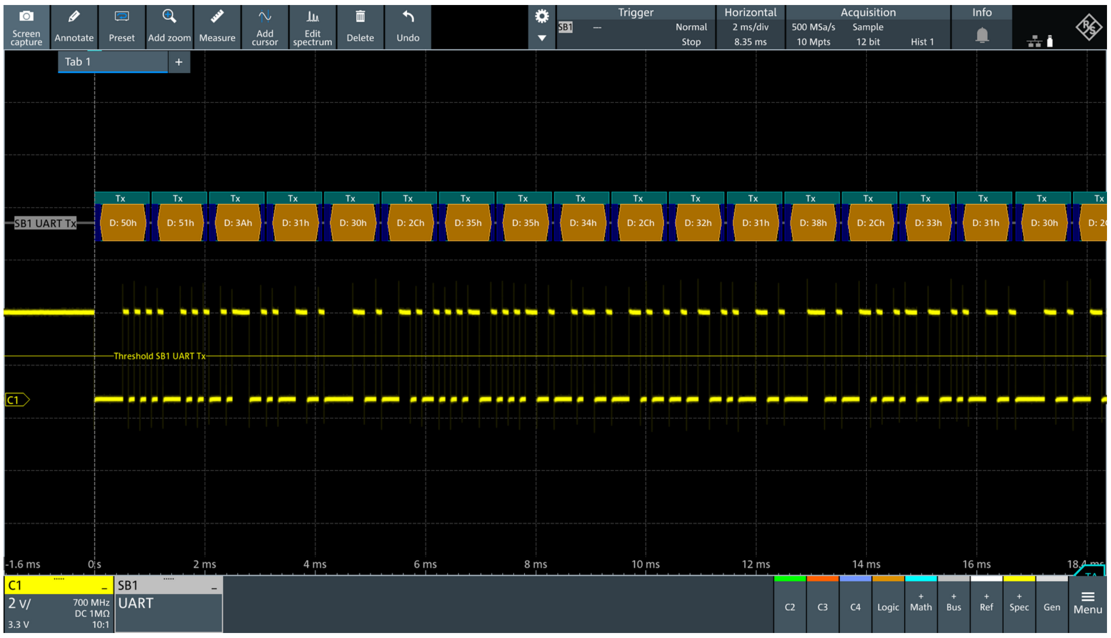
```
RX Length: 32
RX RSSI: -57
RX SNR: 6
50 51 3A 31 30 2C 35 36 35 2C 32 33 34 2C 33 31 39 2C 36 34 33 2C 31 30 33 2C 33 30 39 2C 30 0
PQ:10,565,234,319,643,103,309,0 ; 32,-57,6
```

There are a number of different fields corresponding to the readings of different sensors aboard plus additional information that can be used for several purposes. The packet starts with the sub-string “PQ” followed by:

1) packet number,
2) light sensor,
3) CPU temp,
4) body average temp,
5) average humidity,
6) average acceleration,
7) pressure.

The string may be terminated by a null although this terminator might not be present in the final implementation.

This string is also sent to the serial port #2 (P2 expansion UART in the PQube-IST-sat.ONE). However, in this case, there is more information appended, separated by the sub-string “ ; “. These information consist of: the length and the RSSI value of the received packet plus the S/N ratio with which the same packet has been received. An example is shown in the last line of the transcript of the transceiver console above. Below, a snapshot of the signal available at P2 – Tx line showing the same packet, is depicted.


Screen capture
Annotate
Preset
Add zoom
Measure
Add cursor
Edit spectrum
Delete
Undo
Tab 1
Trigger
SB1
Normal
Stop
Horizontal
2 ms/div
8.35 ms
Acquisition
500 MSa/s
Sample
10 Mpts
12 bit
Hist 1
Info
SB1 UART Tx
Tx
D: 50h
D: 51h
D: 3Ah
D: 31h
D: 30h
D: 2Ch
D: 35h
D: 35h
D: 34h
D: 2Ch
D: 32h
D: 31h
D: 38h
D: 2Ch
D: 33h
D: 31h
D: 30h
Threshold SB1 UART Tx
C1
-1.6 ms
0 s
2 ms
4 ms
6 ms
8 ms
10 ms
12 ms
14 ms
16 ms
C1
2 V/
700 MHz
DC 1MΩ
3.3 V
10:1
SB1
UART
C2
C3
C4
Logic
Math
Bus
Ref
Spec
Gen
Menu