Assignment #1 for Operating systems for embedded systems

WEATHER STATION

Castro Daniele *S253244*

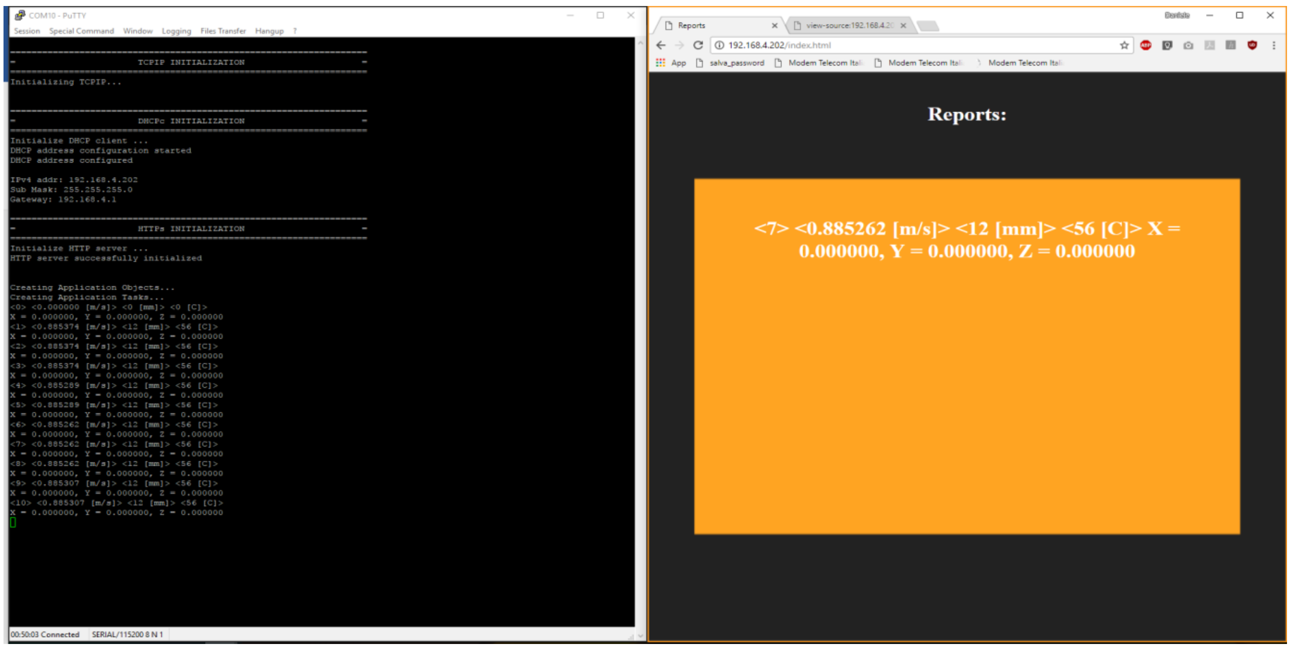
Date : 20-12-2017

BEHAVIOUR DESCRIPTION

Pins are PTC3 for the wind sensor, PTB2 for the rain sensor and PTB3 for the thermometer.
On turning on the board some initialization messages will be displayed through the virtual USB serial port terminal including the board IP address acquired through the DHCP client implemented in the on board operating system. After that, webpage will be ready and weather station messages will be displayed both on the page and on the serial terminal according to this format:

<1853> <0.885258 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 1.000977

XYZ axis will be zero for about 16 measurements. This is caused by the delay functions necessary for making the accelerometer start working.



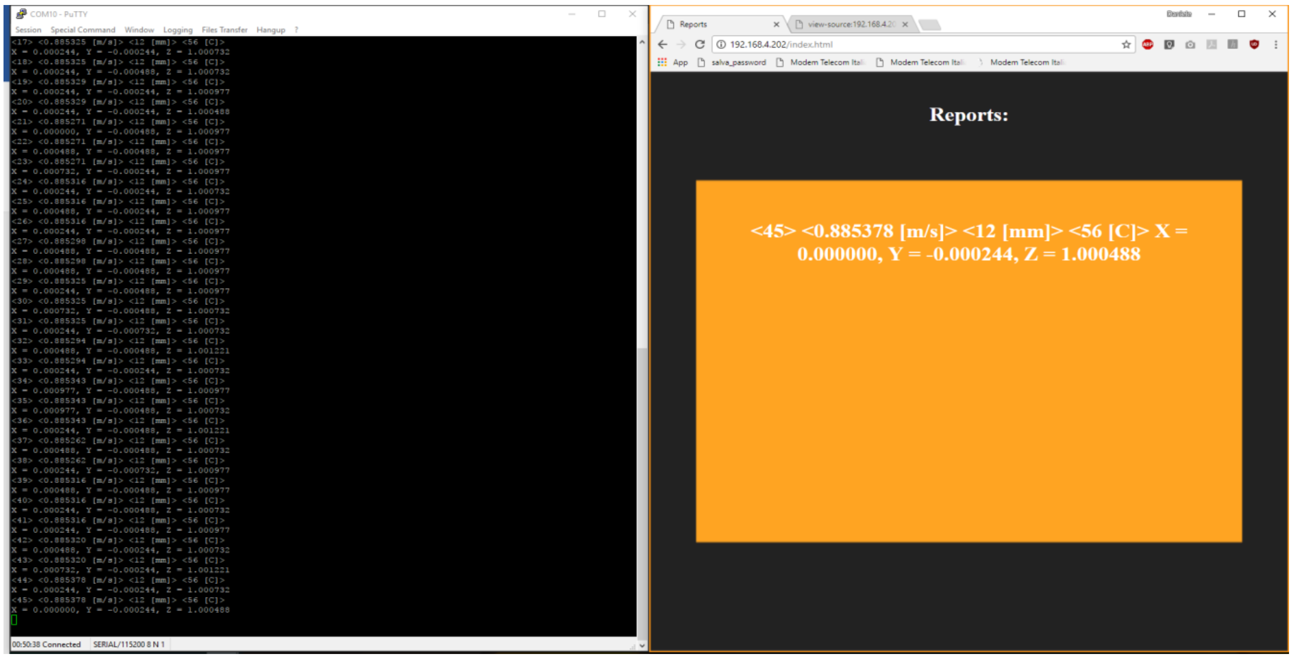
```
------------------------------------------------------------------
-                     TCPIP INITIALIZATION                       -
------------------------------------------------------------------
Initializing TCPIP...


------------------------------------------------------------------
-                     DHCPc INITIALIZATION                       -
------------------------------------------------------------------
Initialize DHCP client ...
DHCP address configuration started
DHCP address configured

IPv4 addr: 192.168.4.202
Sub Mask: 255.255.255.0
Gateway: 192.168.4.1

------------------------------------------------------------------
-                     HTTPs INITIALIZATION                       -
------------------------------------------------------------------
Initialize HTTP server ...
HTTP server successfully initialized


Creating Application Objects...
Creating Application Tasks...
<0> <0.000000 [m/s]> <0 [mm]> <0 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<1> <0.885374 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<2> <0.885374 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<3> <0.885374 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<4> <0.885289 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<5> <0.885289 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<6> <0.885262 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<7> <0.885262 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<8> <0.885262 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<9> <0.885307 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
<10> <0.885307 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = 0.000000, Z = 0.000000
```

Reports:

<7> <0.885262 [m/s]> <12 [mm]> <56 [C]> X = 0.000000, Y = 0.000000, Z = 0.000000

COM10 - PuTTY
Session Special Command Window Logging Files Transfer Hangup ?
00:50:03 Connected SERIAL/115200 8 N 1
Reports
192.168.4.202/index.html

This figure shows initialization messages and first zero accelerometer values



```
<17> <0.885325 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000732
<18> <0.885325 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000488, Z = 1.000732
<19> <0.885329 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000977
<20> <0.885329 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000488
<21> <0.885271 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = -0.000488, Z = 1.000977
<22> <0.885271 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000488, Z = 1.000977
<23> <0.885271 [m/s]> <12 [mm]> <56 [C]>
X = 0.000732, Y = -0.000244, Z = 1.000977
<24> <0.885316 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000732
<25> <0.885316 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000244, Z = 1.000977
<26> <0.885316 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000977
<27> <0.885298 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000488, Z = 1.000977
<28> <0.885298 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000488, Z = 1.000977
<29> <0.885325 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000488, Z = 1.000977
<30> <0.885325 [m/s]> <12 [mm]> <56 [C]>
X = 0.000732, Y = -0.000488, Z = 1.000732
<31> <0.885325 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000732, Z = 1.000732
<32> <0.885294 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000488, Z = 1.001221
<33> <0.885294 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000732
<34> <0.885343 [m/s]> <12 [mm]> <56 [C]>
X = 0.000977, Y = -0.000488, Z = 1.000977
<35> <0.885343 [m/s]> <12 [mm]> <56 [C]>
X = 0.000977, Y = -0.000488, Z = 1.000732
<36> <0.885343 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000488, Z = 1.001221
<37> <0.885262 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000488, Z = 1.000732
<38> <0.885262 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000732, Z = 1.000977
<39> <0.885316 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000488, Z = 1.000977
<40> <0.885316 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000488, Z = 1.000732
<41> <0.885316 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000488, Z = 1.000977
<42> <0.885320 [m/s]> <12 [mm]> <56 [C]>
X = 0.000488, Y = -0.000244, Z = 1.000732
<43> <0.885320 [m/s]> <12 [mm]> <56 [C]>
X = 0.000732, Y = -0.000244, Z = 1.001221
<44> <0.885378 [m/s]> <12 [mm]> <56 [C]>
X = 0.000244, Y = -0.000244, Z = 1.000732
<45> <0.885378 [m/s]> <12 [mm]> <56 [C]>
X = 0.000000, Y = -0.000244, Z = 1.000488
```

Reports:

<45> <0.885378 [m/s]> <12 [mm]> <56 [C]> X = 0.000000, Y = -0.000244, Z = 1.000488

COM10 - PuTTY
Session Special Command Window Logging Files Transfer Hangup ?
00:50:38 Connected SERIAL/115200 8 N 1
Reports
192.168.4.202/index.html

This other figure shows the correct initialized accelerometer values

BRIEF CODE DESCRIPTION

The present document will be a brief description of what I implemented and how I made this first OSES assignment using µC/OS in the FRDM-K64F development board based on MK64FN1M0VLL12 micro controller unit. It will only explain how task, interrupt handlers and initializations functions interact together and how they work from a high level point of view. For more information about configuration registers, settings and secondary called functions you must refer to the code comments.

I implemented all the 9 requests (including the optional ones).

All the requests are performed by these four tasks:

```
static      void TaskFtm0( void *p_arg );

static      void TaskIrqAdc0( void *p_arg );

static      void AccelerometerTask( void *p_arg);

static      void Print_task( void *p_arg );
```

Starting from the first assignment request I will explain the `TaskFtm0();` task: it senses the period of a square waveform coming from a wind sensor on PTC3 pin. It setups the FTM0 timer and enables the interrupts for this peripheral calling the `ftm0_setup();` function. I didn't modify this function much from the original one provided by the labs. Except for the `FTM_SC_PS(0)` flag that was modified in `FTM_SC_PS(7)`. This enables the FTM0 prescaler to divide the peripheral clock frequency by 128 instead of one. Then the task enters a loop that launches the dual capture process, it waits for a semaphore to be unlocked, it modifies the value to print and waits 1 second for the next update.

Values are captured by the `void    ftm0_int_hdlr( void );` interrupt handler function that responds to any rising or falling edge and timer overflow interrupt event. Basically, it counts in hardware the amount of time between the two kind of edges. It takes care of possible overflows (in software) and if they are more than 100, without any kind of edge detected, the variable to be printed will be cleaned to 0. In this way I can handle low frequencies and 0Hz case. Each time calculations are done a semaphore is unlocked and the `TaskFtm0();` proceeds.

The `TaskIrqAdc0()` task for the second and third assignment request is responsible of taking the analog values on PTB2 and PTB3 pins. Respectively for rain and temperature sensors. It setups the ADC0 peripheral calling `setup_adc0();` and enters a loop where we choose which pin to sample by using

```
ADC0_SC1A = (12 & ADC_SC1_ADCH_MASK) | ADC_SC1_AIEN_MASK;
```

(in this case 12 stands for PTB2 and 13 for PTB3). Then it waits for the semaphore to be unlocked and modifies the respective values to be printed. It does this sequence twice, for PTB2 and PTB3 to be sampled, and then it waits 60 seconds.
`adc0_int_hdlr( void );` is the interrupt handler function for ADC0. When a sample is ready an interrupt is sent and this ISR gets the value and unlocks the semaphore for the respective above described task.

The `AccelerometerTask();` task for the eighth assignment request initializes both the I2C peripheral of the microcontroller and the FXOS8700CQ sensor mounted on the board for being properly calibrated at startup time. Clearly, sensor is initialized and calibrated using I2C peripheral that is also used for reading values. Initializing and calibrating functions are in calling sequence:
`I2C_Init();`, `FXOS8700CQ_Init();` and `FXOS8700CQ_Accel_Calibration();`.
After that a while cycle reads the output values of the sensor when a semaphore is unlocked and computes the XYZ axis in g's format. This process is cycled every second.

The `static void PTC6_int_hdlr( void )` interrupt handler function unlocks the semaphore every time the FXOS8700CQ holds low the PTC6 GPIO pin. In this way the `AccelerometerTask();` task can be interrupt driven because the sensor can assert a signal every time new data are available.

Finally, the `Print_task();` for the fifth and ninth assignment request prints sensed data both through the USB virtual serial port and in to the webpage calling the `static void PrintInWebPage(char* string)` function. The `Print_task();` creates the string with all the values to be printed and passes this string both to `BSP_Ser_Printf();` for the virtual USB serial port and to `PrintInWebPage();` that will concatenate the webpage located in `webpages_static.h` as:

```
#define PAGEP1_LENGTH       0x000002AE
#define PAGEP2_LENGTH       0x00000031
#define PAGE_LENGTH      PAGEP1_LENGTH + PAGEP2_LENGTH + 150
#define BACKGROUND_LENGTH       0x0001589A

...

const unsigned char PageP1[] = ...

const unsigned char PageP2[] = ...

unsigned char Page[PAGE_LENGTH];

const unsigned char Background[] = ...
```

the string will be concatenated between `PageP1[]` and `PageP2[]`.

The web page is a simple array of hexadecimal values obtained translating the html file using srec_cat.exe. Storing the webpage as hexadecimal values instead of characters allows me to maintain the formatting of the index.html file and to store the background image directly in the microcontroller memory instead of loading it from the web. In This way images will be loaded also without an internet connection.

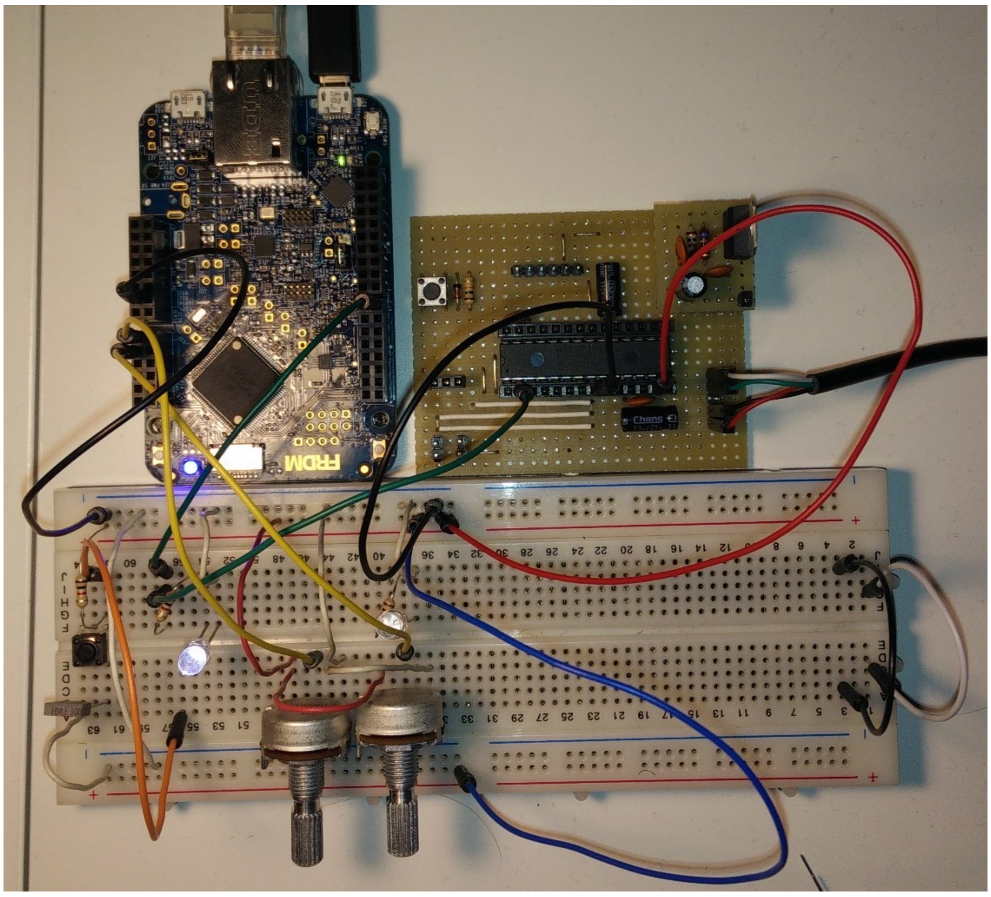
The web page is constantly updated (reconcatenated) every second, but initialized one time in the `static void AppTaskStart (void *p_arg)` with the function `AppHTTPs_Init((unsigned char*)Page, (CPU_INT32U) PAGE_LENGTH, STATIC_INDEX_HTML_NAME, (unsigned char*)Background, (CPU_INT32U) BACKGROUND_LENGTH, STATIC_BACKGROUND_NAME);`
I modified it for supporting all these parameters because I had to modify the page at runtime. Originally the web page was static. Now I include `webpages_static.h` in the `app.c` file instead of in `app_http-s.c` so I had to pass the web page and the background image pointers to this function through parameters.
In this way, I was able to modify the contents of these pointers at runtime while being initialized only at startup.

The webpage is projected for resizing the background image the same size as the browser window and, automatically, to refresh the content of the weather report after every 5 seconds.
Text will always be placed inside the orange rectangle of the background image.

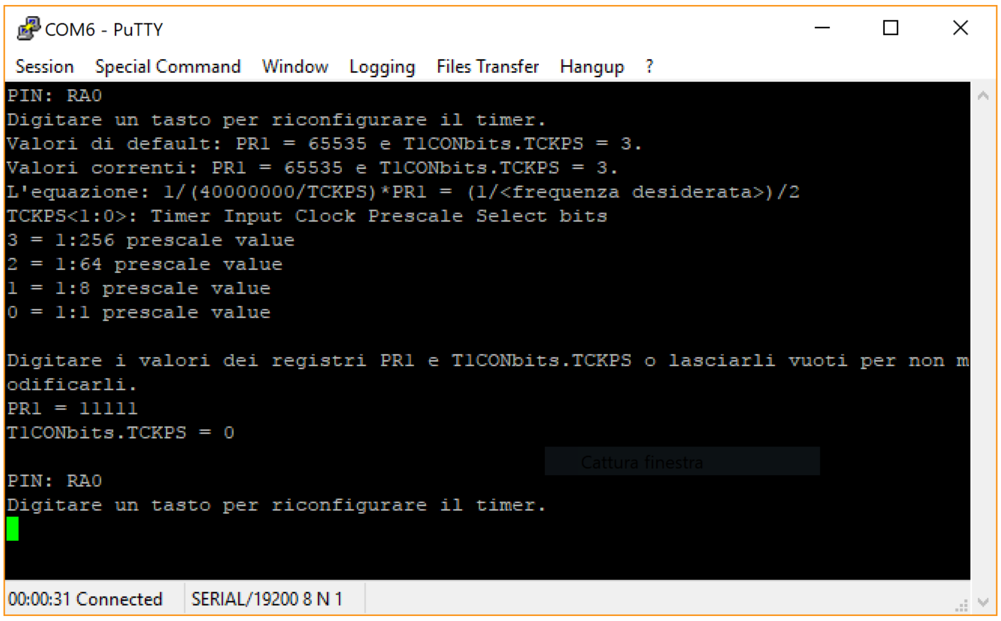
Html code is provided in the assignment folder.

TESTING TOOLS

For testing the board I used two 10k potentiometers in a breadboard, a push button and an handmade development board based on dsPIC33FJ128GP802 digital signal controller. I provided the dsPIC source code for testing inside the assignment directory.



The board under test



```
PIN: RA0
Digitare un tasto per riconfigurare il timer.
Valori di default: PR1 = 65535 e T1CONbits.TCKPS = 3.
Valori correnti: PR1 = 65535 e T1CONbits.TCKPS = 3.
L'equazione: 1/(40000000/TCKPS)*PR1 = (1/<frequenza desiderata>)/2
TCKPS<1:0>: Timer Input Clock Prescale Select bits
3 = 1:256 prescale value
2 = 1:64 prescale value
1 = 1:8 prescale value
0 = 1:1 prescale value

Digitare i valori dei registri PR1 e T1CONbits.TCKPS o lasciarli vuoti per non m
odificarli.
PR1 = 11111
T1CONbits.TCKPS = 0

PIN: RA0
Digitare un tasto per riconfigurare il timer.
```

The serial terminal of the dsPIC with the testing program