Embedded Systems
Assignment 1

Enrico Simetti

ISME - Interuniversity Research Center on Integrated Systems for Marine Enviroment
DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering
University of Genova, Italy
enrico.simetti@unige.it

Assignment 1

1. Simulate an algorithm that needs 7 ms for its execution, and needs to work at 100 Hz. This is to emulate a real world scenario.
2. Acquire the three magnetometer axes at 25 Hz (set the data rate to 0b110 in the 0x4C register).
3. Do the average of the last 5 measurements.
4. Send it to the UART at 5 Hz using the protocol $MAG,x,y,z*, where x is the average value on the x-axis, y is the average value of the y-axis, and z is the average z-axis value.
5. Compute the angle to the magnetic North using atan2(y_avg,x_avg)
6. Send the computed angle at 5 Hz using the message $YAW,x*, where x is the angle in degrees.

Suggestions

1. The availability of only 3 ms to perform operations suggests the use of interrupts to handle the UART, both in reception and transmission.
2. The suggested way is to use separate circular buffers to store data coming and going to the UART.
3. Given that the IMU allows an SPI clock up to 7.5 MHz, the SPI interactions can be consider negligible (two bytes require around 2 microseconds at 7.5 MHz).
4. Make sure to handle properly shared data between interrupts and the main (i.e., disabling interrupts and creating a critical region in the main code).
5. Do you have any warning when compiling? Solve them! Sometimes, a warning hides a nasty bug (e.g., truncation due to overflow.)

Things that are evaluated

1. Are the tasks executed at the correct frequency? Are there missed deadlines because of some erroneous handling of the peripherals? Are busy-waiting performed when allowed?
2. Is the shared data handled correctly? Is there any possibility of wrong sequence of interrupts that might cause some erroneous behaviour?
3. Are circular buffers sized correctly? i.e., have you actually thought on how fast data is produced/received? Memory is a scarce resource in embedded systems.
4. Are the interrupts short in time? Are you performing unnecessary busy-waiting inside interrupts? Note: not all while loops are busy-waiting loops.
5. Is the code sufficiently commented, well written, formatted? Maintainability and quality of the code is key in embedded applications.

Final notes

1. In case of doubt, ask! Interactions is not forbidded, actually it is recommended.
2. Only a single group participant should submit the final project on Aulaweb. Please submit the whole project folder.
3. Use your group ID when creating the project in MPLAB X.