UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Antonio Martí Campoy
amarti@disca.upv.es

Testing hardware

Methods and Tools for Software Quality

ESIGELEC

Introduction.

This document presents the steps you should follow to test the proper operation of hardware. Also, you will practice the use of the robot_library and its functions. Carry out the following exercises. Some of them present you a question or questions. Answers to these questions must be delivered to the teacher at the end of the course in a brief report, so it is a good idea to write them in a document.
Finally, if you have doubts or something behaves improperly, don't hesitate to ask to the teacher.

Exercise T1

The display included in the robot is a helpful component to get internal information from the system while it is running. This first exercise ask you to add code in function main to show in the display a counter from 0 to 255, and repeat again. Some advises:
Use MISRA-C compliant types, e.g. uint8_t for a variable in the range 0 to 255.
Don't forget to add the header file related to the display.
It is advisable to clear the display after initializing it.
In order to slow the execution of the loop, you can use intrinsic function __delay_cycles (cycles); taking into account that MCLK is 12MHz. How many cycles do you need to delay for 200 milliseconds?

Exercise T2

It is time to check the motors. Write a program to run both wheels forward at 20% during one second, then run at 50% during two seconds, then run at 100% during 3 second, then stop. Repeat the previous sequence with same percentages but running wheels in reverse. Then, repeat again.
You can add your new code to the previous project or create a new one. The code doesn't have to be efficient or elegant, just working. Don't forget to include the header file related to function for control of motors and enable global interruptions using __enable_interrupt(); intrinsic function. Did you use constants LEFT & RIGHT? Did you check the return value of function *Speed_motor*?

Exercise T3 (checkpoint)

In this exercise you will check the encoder for the left wheel. Write a program that initialise encoders, clear distance for left wheel and run the left wheel at 20%. Then show continuously in the display the value of steps. Don't forget to declare a variable of type *distance_type.*
If the encoder works properly modify your code to show turns instead of steps.
You may check with other speeds and change direction. Did you check the return value of function *Read_Distance*? What can you do if the function returns error?
WARNING: show your code to the teacher.

Exercise T4

Repeat the previous exercise with right side. Any doubt or problem, ask the teacher.

Exercise T5

Write a program to read from the compass every 100 milliseconds and show the bearing in the display. Uses low resolution function. Check the proper calibration of the compass.

Exercise T6 (checkpoint)

Repeat previous exercise but now use high resolution function. Check the proper calibration of the compass. Show your code to the teacher.

Exercise T7 (checkpoint)

Use LDRA tool to analyse your code. You can mix the code of all previous exercises in one file without regard to functional result since you are not going to run it. Show static analysis reports to the teacher.

Exercise T8 (checkpoint)

Did you notice that names of functions don't follow a common criterion? For example, compare *void **i**nit_display (void); void **I**nit_motors(void); and uint8_t compass_init(void);*
Did you see differences? What do you think? Discuss with the teacher and write your thoughts and proposals in your report.