Robótica 2021-2022

1st José Luis García Salas

2nd Enrique Moreno Ávila

I. Introduction

Nowadays, the research field of robotics is a must-know discipline for all the engineers who are working in technologies like Internet of Things (IoT), automation of tasks, or home automation for people with reduced mobility. The use of robots for making the daily issues easier and natural to make life more simple is becoming so common, that in actual days is strange the house that does not have electronics integrated to work with assistant like Amazon's Alexxa or have the refrigerator or cleaner bot connected to the phones with the purpose of track its activity and route.

In this subject, following the line of work of previous years, we are going to learn how the fundamentals of robotics functions, as well as making our own simple programs to make the simulated robot obeys our commands and instructions with the objective of transfer our own software to a real robot facilitated by the department of RoboLab from the EPCC in the UNEX.

To make this software, we will use a library written in C maintained by RoboLab in EPCC, Aston University and many other collaborators called Robocomp, an open-source framework based in the idea of communicate components through public interfaces.

In this letter, we are going to explain step by step the procedure to make a good software for robot control, as well as the other technologies and tools used in the making.

II. First Class

Here we are going to introduce all the new software tools we use in the design of the robot software as well as we make for the first lesson of revise C/C++ knowledge.

A. Software Tools

We run all the software in a Ubuntu 20.04.3 Focal LTS. Because of the Robocomp library y heavily based on C, we are going to write our code on it, using the GCC/G++ compiler 10.3.0 version because it brings the last dependencies and libraries to work with. Generally, in the Ubuntu version we are using, the GCC/G++ compiler are installed in the 9.3.0 version. We had to update it installing the 10.3.0 version executing **sudo apt-get install gcc-10 g++-10** to install it, and then switching the version of the compiler which the S.O. is using with **sudo update-alternatives –install /usr/bin/gcc gcc /usr/bin/gcc-10 2 0 –slave /usr/bin/g++ g++ /usr/bin/g++-10**.

For running and coding in C/C++, we use CLion, an IDE developed and maintained by JetBrains, it is one of the best for this situations due to its capacity of build by itself all the dependencies between the libraries and files, as well as good maintainance with other associated compiling tools like CMake. One good alternative for this is the code editor Visual Studio Code (VSC).

Once we installed all this, we install the first important tool for the interfaces and libraries needed for the robot behaviour, it is called Qt5 software. This is the main toolkit to use the QT libraries related to C/C++, and despite the fact that we are using CLion in first place, QtToolkit have its own IDE called QtCreator and most important, QtDesigner, which is an interface designer for tools like timer counters and lcd displays and the main tools we are going to use for the visual part. As we mentioned, the main library we are going to use is Qt, specifically related to signals and slots, this library fits perfectly with the methodology of components, we can use objects declared as QTWidgets as the future components of the robots, in which each slot is a function triggered by a signal declared previously and connected by the **connect()** method.

For the version control, we use Git, a software of code-version-control which allows us to maintain different work-flows and previous status of the code in case we find an error or we make a critical mistake. With Git, we use the repository GitHub, in which we upload from our local repository of Git all the versions of the code for working everywhere in remote control without the need to do it only in classes.

The last tool used before we star to install all Robocomp library is Overleaf, Overleaf is a online text editor which allows us to have the documentation stored in the cloud, and modify the document simultaneously making the parallel documentation possible and we can review and improve our own mistakes or changing things with the security of being both of us at the same time.

Once all of these tools are installed, it is possible start installing all the Robocomp related software in the next classes and the new ZeroC middleware for the communication between interfaces.

B. QTimer and timerSimple

In this lesson we are given 2 different projects, both of them implementing the same thing, a simple timer with a counter of elapsed time and a slider to modify the period for the timer itself.

The first project, called ejemplo1, is the version of the timer implemented using QtTimer library. The main difficult here was to learn the functionality of signals and slots, and how they interconnect, despite this learning proccess, Qt libraries allow to make this very easy and simple to understand.

The second project, called timerSimple, is the same version of the timer, but using the std library of C/C++, the main goal of this project is to review and refresh the knowledge of C/C++ after so long time without use it. This timerSimple version has been heavier and deeper than its other version, we think that the most difficult part is understand the declaration and use of the thread and how is used and synchronized with other threads without hindering each other.

For the visual interface of the timer, there are two ways to modify and built it, we can go directly to the .ui file located in the project, it is a xml file and it can be modified adding manually the new objects like buttons or lcd displays and horizontal slayers. The other way, and it is the more simple way to do it, is open the .ui file with the tool of QtSoftware called QtDesigner, it is a graphic interface and you can drop the object in visual mode, making so much easier the interaction with the timer and interface.

As a conclusion to this practice, we think that the most hard part of it was the use and manage of the std library, probably because the long time we do not use C/C++, despite this, we adapt quickly and it has been an easy review practice.
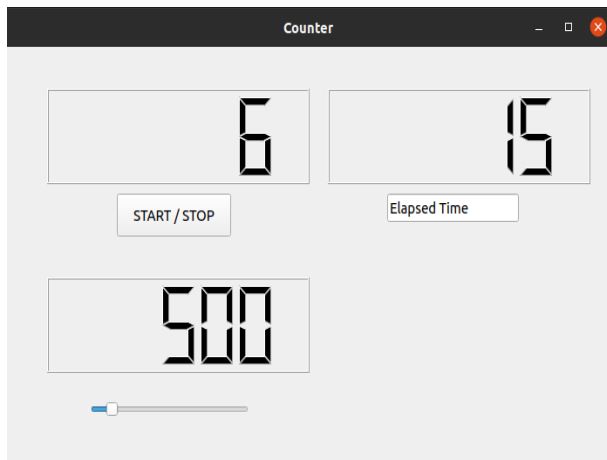

Fig. 1. Graphic representation from the interface of the timer.