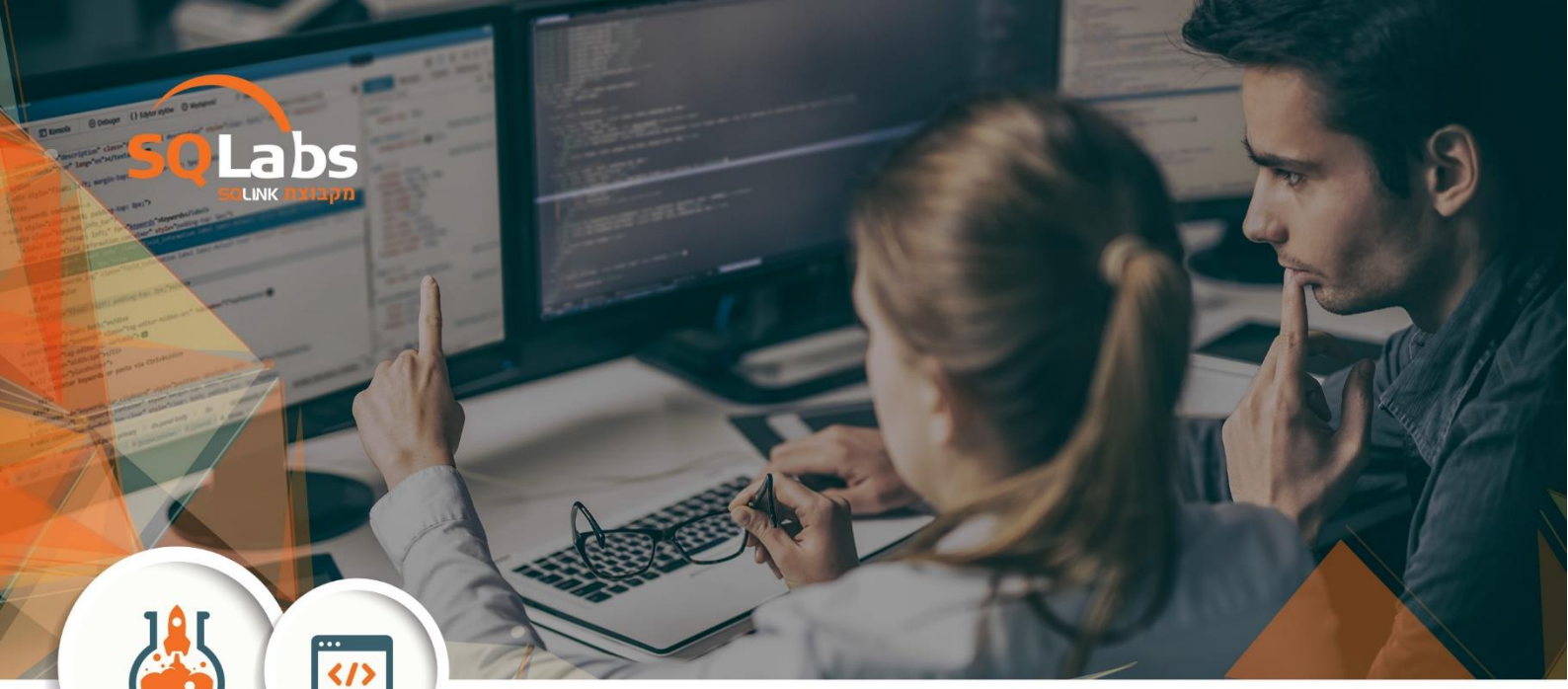

RT Embedded Programming Course – 500 Hours

Course summary

The objective of this course is teach and exercise the students in all aspects of low level programming. This means C, it's preprocessor, GCC Compiler and it's flags, C++, standard libraries, debugging, writing quality code, testing and of course operating system and device drivers. Since Linux is gaining ground in the embedded market we will teach Linux device driver authoring as well as VxWorks programming and explain the profound differences between the environments. We will also teach RT issues, where latency comes from, tools to help us find latency, and how to design RT systems from scratch.

The course will involve exercises of any topic taught and will challenge the students to the peak of their ability since we are trying to teach an issue which usually requires many years of experience in the industry.

The course will be taught by a known industry veteran who was involved in RT projects throughout the Industry and who has already taught these topics successfully in various companies and institutions.

We don't expect to produce industry veterans from the course. We do expect that a student completing the course will be able to step in to a junior position in any RT or Embedded project and be able to fill his position without needing instruction on any of the basics of RT or Embedded. Thus he/she not be a burden on the team in which he/she is in but rather a productive member from the start. This is a high bar to set since the RT and Embedded field is one of the most challenging in the IT industry.

Audience

- Bachelors of computer science, math, computer engineering
- Bachelors of any exact science
- People with real experience in programming (in any language) who wish to enter the RT and Embedded space.

Audience will be tested for compatibility to the educational program.

Hardware on which the course will be taught

Linux virtual machines on Intel machines with at least 8 GB RAM.

Topics

Introduction to low level programming

Assembly:

- The CPU and multi CPU model (theory)
- X86 assembly on Linux

Linux:

- Secure operating system fundamentals (theory)
- Shell and regular work:
  - Bash
  - Writing Bash scripts
  - Linux file system

C and C++ programming (practice is over Linux):

- C programming
- Large project build systems (theory)
- GNU Make
- The standard C library
- Linux systems programming (including multi threaded and multi process programming)
- Object oriented programming fundamentals (theory)
- C++ programming
- The standard C++ library
- Design patterns for C++ developers
- Templates introduction
- STL introduction
- Object oriented programming using the C language
- GCC in more depth

Hardware:

- PCI, USB, I2C/SPI

Linux kernel:

- Linux kernel including boot process and writing device drivers

Real time:

- Principles of RT systems (theory)
- RT on Linux

Non secure (embedded) operating systems:

- Principles of non secure operating systems (theory)
- Programming on VxWorks

Concluding project.