modular mx

Embedded Upskilling

Advance Embedded C

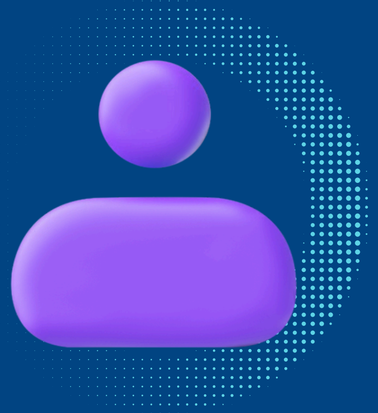
Pointers, Structures, Preprocess
GCC compiler, make builds and Unit Tests

modular mx

Editorial

C will be your main programming tool for microcontrollers. It is essential that we have a good command of it, and therefore, it's mandatory to start reinforcing this topic. More than 50 exercises will help with this task. They will reinforce basic topics such as variables, statements, loops, and functions, as well as more advanced concepts like pointers, structures, and the preprocessor. But not only that; you'll also learn where and why to use them, as you will create a Round-Robin scheduler with timers and waiting queues. Everything will be compiled using the terminal and a small makefile. You will need to perform unit testing on the programs you write.

modular mx



Who is it for?

This training is aimed at engineers with basic knowledge of the C language who wish to enhance their skills in syntax and usage of the language. Prior experience and knowledge in variable declaration, statements, and loops, as well as a basic understanding of functions, are required

modular mx

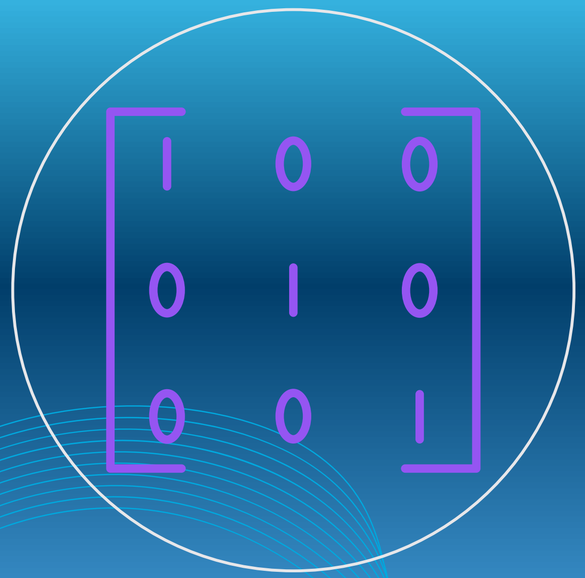
Basic, bytes and bits

We will briefly review how to declare variables in C, use if/else statements, for and while loops, as well as use functions to better structure our program. We will also likely compile using the command line for the first time.

We will create functions for bit manipulation in 8-bit and 32-bit variables by developing reusable functions using shift operators (<< and >>) and binary operators: and (&), or (|), xor (^), and not (~). All functions will need to undergo unit testing using the assert macro.

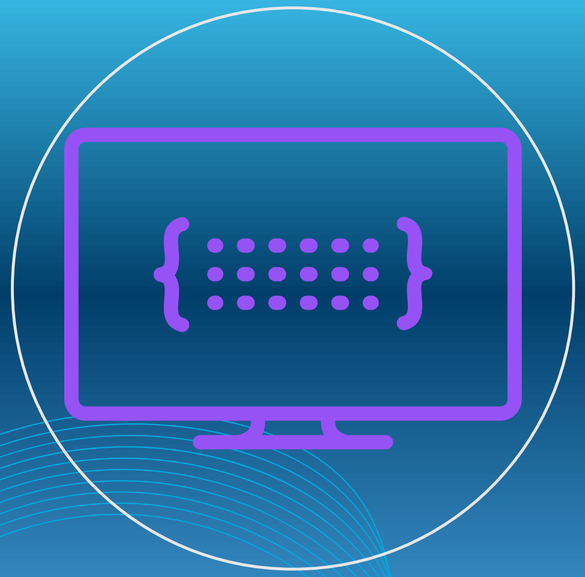
modular mx

Arrays and pointers



Data collections of the same type, called arrays, and pointers go hand in hand in C, so we will learn about them by defining functions with common sorting and searching algorithms. We will understand how pointers work with different data types in C, including pointer arithmetic, null pointers, function pointers, constant pointers, and pointers to constant content.

modular mx



Structures

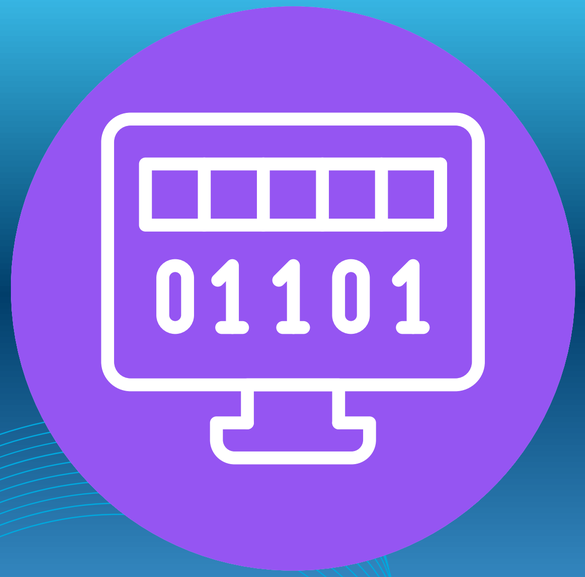
Definition and declaration of structures and unions for new data types. Passing structures to functions. Pointers to structures. Serialization and deserialization of structures. Differences between structures and unions. Handling of byte padding.

modular mx

Preprocess

The C preprocessor runs before the compilation stage and simply replaces text it doesn't handle variables, functions, or any kind of magic. In this part of the course, you'll master it completely, learning how to create macros, use conditional compilation, and define your own pragmas. That way, you'll finally stop thinking that a #define declares a constant.

modular mx



Circular buffers

It's time to create your first interesting algorithm: a circular buffer or waiting queue. In this project, you will apply all the knowledge you've acquired so far, including handling arrays with structures, null pointers, and generating your first reusable code in its own library or driver.

modular mx

Scheduler

You will write your own cooperative “Round-Robin run-to-completion” scheduler. This scheduler will be created using pointers to structures and function pointers. Software timers and the previously implemented waiting queues will be added. Of course, all code must be tested with unit testing, and a small makefile will be created to handle the build.

modular mx

Requirements and material

All sessions are virtual, so it doesn't matter where you are in the country or the world, as long as you have internet.

- [x] Computer with any OS
- [x] Internet connection
- [x] Basic programming in C

Material

2 remote sessions of 90 minutes each per week 24/7 access to material on Confluence, Slack group with participants and alumni

modular mx

Training details

ENROLL NOW

Starting date:
September, 1, 2025

Total training cost:
$2900.00 MXN

SPECIAL OFFER

Schedule meeting:
Monday and Thursday 6 to 7 pm UTC-6

Advance Embedded C

modular mx

Advance Embedded C