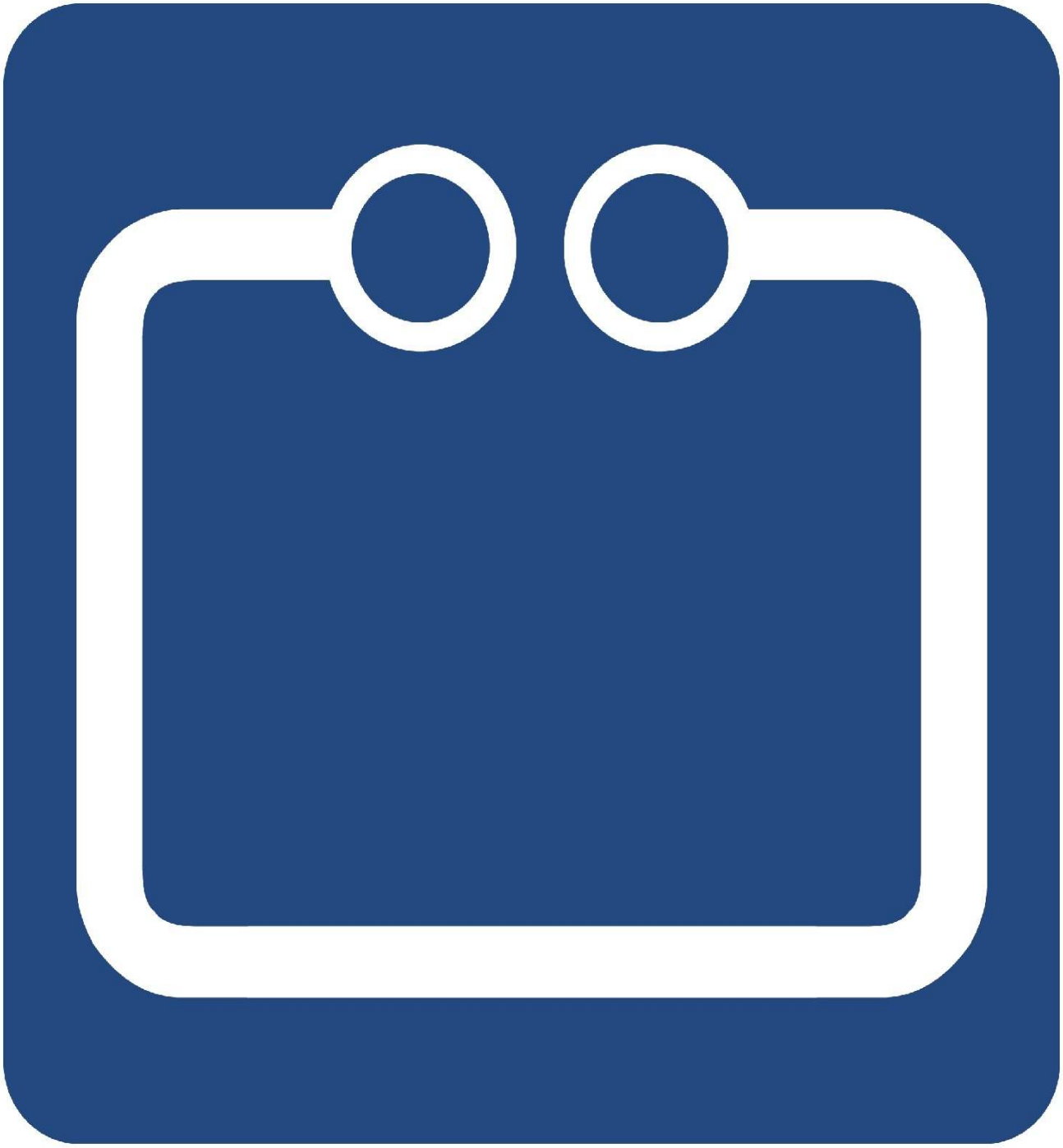

Practical Guide to Implementing Communication Protocols in C++ (for Embedded Systems)

Table of Contents

Guide to Implementing Communication Protocols in C++ (for Embedded Systems)

Almost every electronic device/component nowadays has to be able to communicate to other devices, components, or outside world over some I/O link. Such communication is implemented using various communication protocols.

At first glance the implementation of communication protocols seems to be quite an easy and straightforward process. Every message has predefined fields, that need to be serialised and deserialised according to the protocol specification. Every serialised message is wrapped in a transport data to ensure a safe delivery to the other end over some I/O link. However, there are multiple pitfalls and wrong design choices that can lead to a cumbersome, bloated, and difficult to maintain source code. It becomes especially noticable when the development of the product progresses, and initially developed small communication protocol grows to contain many more messages than initially planned. Adding a new message in such state can become a tedious, time consuming and error-prone process.

This book suggests flexible, generic and easily extendable design architecture, which allows creation of a generic C++(11) library. This library may be used later on to implement many binary communication protocols using simple declarative statements of class and type definitions.

As stated in the book's title, the main focus of this book is a development for embedded systems (including bare-metal ones). There is no use of RTTI and/or exceptions. I also make a significant effort to minimise usage of dynamic memory allocation and provide means to exclude it altogether if needed. All the presented techniques and design choices are also applicable to non-embedded systems which don't have limitations of the latter.