CCCP - COSYLAB COMMON CONTROL PLATFORM

Miha Rescic, Cosylab, Ljubljana, Slovenia
Ziga Kroflic, University of Ljubljana, Ljubljana, Slovenia

Abstract

Cosylab common control platform (CCCP) is a lightweight hardware control platform designed to provide a simple interface to various types of hardware components and fast and simple integration of such hardware into control systems. The core of the platform is the scripting language lua. This lightweight and flexible scripting language provides software real-time control of hardware modules over all provided connections (RS232, Ethernet, USB, SPI, CAN, I2C, GPIO) as well as fast and simple ways of implementing modules for more complex structures (FPGA). The platform provides various levels of control with an embedded GUI or full remote control over an embedded web server, archiving capabilities with a database back-end and different device simulator modes. The platform's small footprint, high degree of flexibility and high level of hardware abstraction make the CCCP an ideal control platform for more complicated hardware instruments and at the same time a perfect main control board for devices that incorporate various complex hardware elements. The design and possible implementations of this platform will be discussed in this article.

INTRODUCTION

Development of a control system is never an easy nor a straightforward task. With the complexity of today’s technologies, if we’re speaking of technologies in general or of technologies applied in specific fields, the number of different components or building blocks of the control systems and the complexity overall grow rapidly.

Within this rapidly expanding field it is very difficult to find a common ground and usually much effort is spent on developing highly specific solutions capable of tackling only a limited array of problems. Thinking of common grounds in control systems field brings to mind a reusable, as generic as possible platform that would represent the base of the control system. This was the motivation behind CCCP: minimize the efforts needed for base platform development and allow emphasis on more specific and complex components development, integration, testing and QA.

ARCHITECTURE

The crucial element of the platform is the architecture. CCCP tries to keep logical entities separated from each other as much as possible. This way, reusability and efficient design are possible.

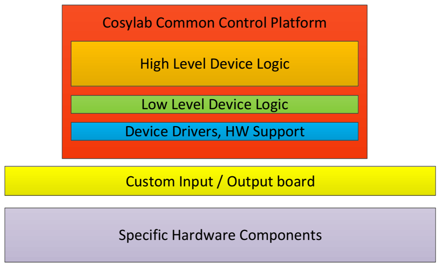
Cosylab Common Control Platform
High Level Device Logic
Low Level Device Logic
Device Drivers, HW Support
Custom Input / Output board
Specific Hardware Components

Figure 1: CCCP Architecture

Custom input / output board

On the lowest level of the CCCP architecture is the customized input / output board. Although the board itself is not a part of the CCCP platform it provides problem or component specific solutions regarding hardware connections, specific protocol implementations or more advanced logic (see Fig. 2). The custom board development is bundled together with the CCCP platform development in order to provide the optimum solution for the specific problem.

Some of the IO board’s main purposes are described below.

- Target hardware development away from the platform core and towards specific implementation needs.
- Provide advanced logic and (hard) real-time support with FPGA.
- Allow connectivity with existing CCCP IOs or implementation of any custom IO required.
- Minimize the complexity of custom HW development.
- Minimize the amount of redundant development efforts regarding non-reusable hardware.

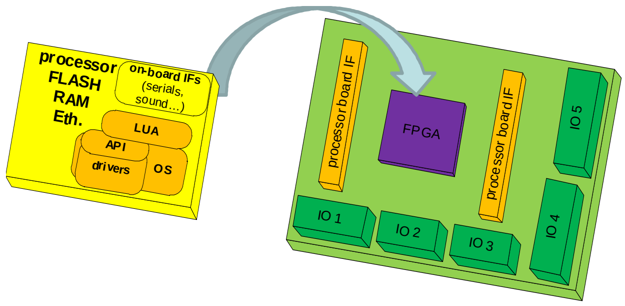

Figure 2: Custom IO board

Device drivers and hardware support

The layer residing directly over the custom IO board, the lowest layer of the CCCP core architecture, provides all the needed logic to communicate with hardware. It contains two crucial elements:

- OS specific drivers providing basic system input / output functionality.
- HW modules providing an interface layer between the underlying hardware components and higher-level device logic.

The HW modules not only allow high level logic to easily interact with the underlying hardware but create an abstraction layer between the two that can be replaced by a mock layer in absence of actual hardware. A mock layer or a simulated layer makes development possible without actual hardware and allows more flexible testing (without hardware limitations) and much faster integration.

Low level device logic

The low level device logic incorporates all the services and layer logic needed for the CCCP platform to function properly. They lay the foundation for the higher layer logic and provide the tools that allow developers and engineers faster development.

Some of the main components residing in the low level device logic:

- HTTP server for northbound communication and control.
- Priority task scheduler with support for interrupts from HW modules.
- Lightweight database for storing data, events and all-purpose logging.
- Generic FIFO queues for inter-process data exchange.

Most of the low level logic is written in C programming language but some segments also use components written in the scripting language lua.

High level device logic

The highest CCCP architectural layer is where the magic happens. This layer, also called the “instrument” logic layer, is developed entirely with the scripting language lua.

The choice of scripting language over a programming language has at least these advantages:

- All the complex implementations are done in lower layers thus abstracted away from the developer.
- Because of simpler syntax, robustness and user friendliness scripting languages make development available to other team members as well, e.g. engineers.

The use of a higher level of logic together with an application and UI framework (e.g. Nokia’s Qt) makes it possible to further upgrade the device with GUIs and other device interfaces (touch screens, ...).

COMMON CONTROL PLATFORM

In order to provide a truly common platform there are some aspects of the platform that need to taken into consideration.

Customizability

Common platform must provide enough flexibility to allow easy customisation for various implementations. Therefore, the core CCCP has no direct IO connectors or switches. It only provides a standard TX-DIMM connector with standard pinout. In order to connect the common control platform to corresponding control system components a separate IO board must be developed.

By mechanically separating the logical parts into two components (CCCP and the IO board) we achieve a high degree of flexibility and customizability. With the custom IO board approach the solution can be very problem specific but still at the same time very generic since all of the core logic is kept on the CCCP platform. The IO board merely serves as an interface to hardware components whereas the implementation of the logic resides on the generic CCCP board and can be further reused in other various control systems or subsystems.

Size and form factor

One of the first limitations a standard common platform encounters is its size and form factor (see Fig .3) but the size of CCCP (DIMM200-module standard size: 67.6 mm x 26 mm x 3.6 mm) makes is suitable for almost any application.

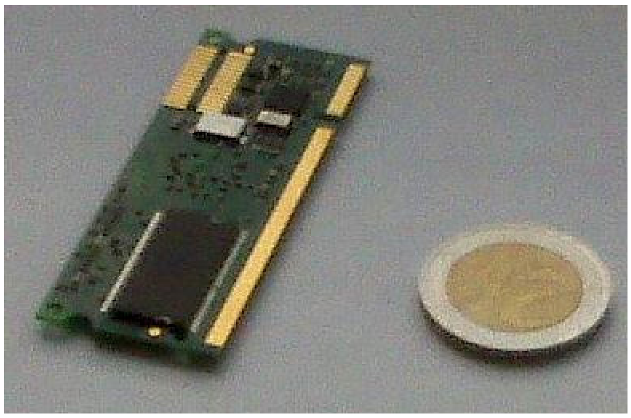

Figure 3: CCCP size

Processor and operating system

The other important aspect of the common platform is the choice of the processor and the operating system. This is why CCCP is powered by an ARM9 400 MHz processor with the operating system of choice being Linux running the 2.6 kernel.

The combination of ARM processor and Linux OS allow users and developers to use a wide range of existing tools, from cross-compilers to integrated development environments.

Connectivity

In order to connect various components to the common platform a number of standard IOs must be supported. CCCP provides the following possibilities (only the most common options are mentioned):

- 10 / 100 Ethernet
- 5 x UART
- 3 x SPI
- 3 x I^{2}C
- 4 x GPIO
- 3 x 12 bit ADC
- 2 x CAN

PRODUCT COMPONENT SIMULATION

The most important aspect of CCCP is the possibility to substitute real hardware components with mock or simulated components (see Fig. 4).

The architectural layering described above, especially its *Device driver and HW support* layer allows a smooth interchangeability between real hardware components and software-simulated components. Because the hardware modules are essentially exposed to higher level device logic it is, after all the interfaces have been defined and with the use of lua flexibility, quite straightforward to make the switch. The simulated device components are implemented at a higher level of logic (in the high-level, lua logic layer) therefore they are overriding any actual hardware components.

Agile development

The process of mocking or simulating absent hardware components makes it possible to introduce new approaches to otherwise rigid hardware development field. One of these approaches is agile development. Some of the benefits:

- Difficult and complex tasks can be dealt with earlier.
- Problems and complications are discovered earlier and therefore resolved earlier.
- Development process can be split into multiple tasks from the beginning and therefore modified based on completion of and feedback from such tasks.

Test driven development

Testing in hardware development is usually the last stage of development process. With the introduction of simulated components the testing can take place from a very early stage onwards.

- Every step of development can be backed up and controlled by matching tests.
- Tests provide feedback and allow the agile process mentioned above to function properly.

REAL-WORLD IMPLEMENTATIONS

Some of the possible use cases of CCCP control platform are described below.

Remote hardware control

One of the basic examples of CCCP usage would be remote monitoring and control of hardware devices, e.g. household appliances.

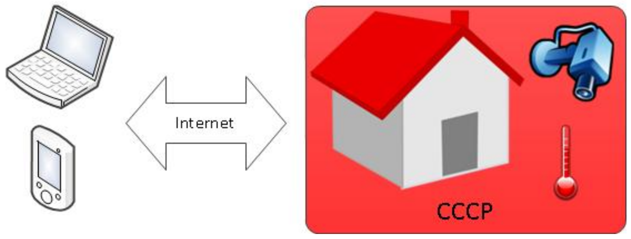

Figure 4: Household control and monitoring

Specific instrument interface

CCCP could also provide an interface to various complex instruments and simplify the integration of these components into the control system.

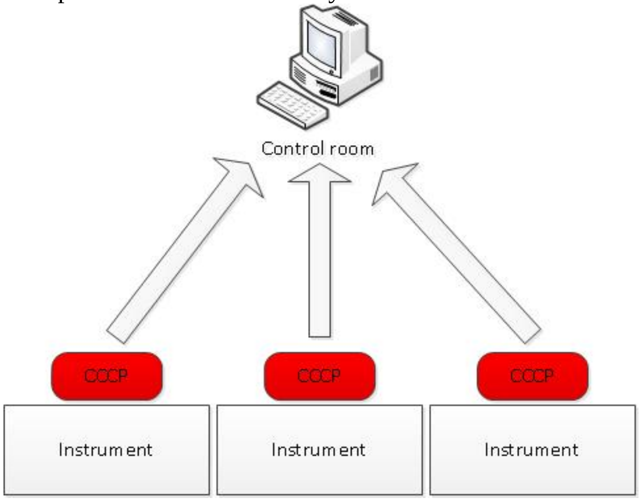

Figure 5: Specific instrument interface

CONCLUSION

Cosylab Common Control Platform presents a different approach to a somehow rigid field of hardware development. With the modular approach regarding hardware and software architecture, simple input and output interfaces, flexible scripting language core logic and device component simulation capabilities it gives our customers a number of benefits.

- Faster time to market with lower development costs.
- Better developer utilization and efficiency. Faster hardware integration, validation and verification.
- Minimized overdevelopment and complexity with maximized flexibility.
- Optimized development process by test driven development and task segmentation.

Small footprint, high degree of flexibility and high level of hardware abstraction make the CCCP an ideal control platform for complicated hardware instruments.