Emulated Hardware for Embedded Control System Testing

Josefine Möllerström & Max Nyberg Carlsson

Have you used any electronic appliance today? If so, chances are you used an embedded system. With the increasing number of embedded microcontrollers, emulating hardware could be used to aid development and testing [1].

We created an emulation mimicking the main microcontroller of the Crazyflie 2.1 quadcopter, Figure 1, which passes the startup tests. We did this using the open source framework Renode [2] to implement peripherals and build the virtual platform. The emulated platform should be used for further research and by Bitcraze.

Figure 1: The Bitcraze Crazyflie 2.1 [3].

Several peripherals pre-existed in Renode but some had to be either modified or implemented from scratch. An example of some peripherals that had to be implemented in order to build the virtual platform, is the Crazyflie sensors. All the peripherals were written in C# and connected to each other in Renode to emulate how the real hardware is connected.

Embedded systems can be tested on different levels, e.g. running on the real hardware or having parts emulated. Since the research area surrounding different testing levels is relatively unexplored there is a need to have a case study that can be used to compare different levels. The Crazyflie is a good case study with regards to complexity and open source firmware. The goal is for the emulation to be used in further studies to compare the strength and weaknesses of the different levels. The aim with this research is to find ways to make development and testing more efficient. As control failures usually occur due to software problems, improving the development cycle is important and can help guarantee the safety of control systems.

The emulation can also be used by Bitcraze AB, the Crazyflie creators. Since embedded system testing is hard, with opaque parts and limited debugging possibilities, tools that aid are valuable. With emulated hardware extreme cases may be tested much easier, such as pausing the system mid-flight to investigate. It would also be possible to include the emulation in their CI pipeline as an extra automatic testing check whenever changes to the firmware are proposed.

During testing of the emulation different test cases were created by introducing bugs in the firmware. The purpose of each test was for the emulation to result in a fail if the self tests did not pass or the system crashed.

The Crazyflie has two microcontrollers, one executing the main firmware and one mainly for radio communication. A limitation to the emulation was that the second microcontroller was not included and the responses from the second controller to the main one had to be simulated. Expanding this functionality would allow more interesting tests to be performed and should be prioritised in future development.

There are still some improvements that can be made to our emulation to increase the accuracy. It is difficult to imitate every property of a system for creating a perfect emulator. The main goal for the thesis to be successful was for the Crazyflie firmware to be run on the emulator and pass the self test successfully. This was achieved and the resulting emulator can be used by Bitcraze and in further research work.

References

[1] Möllerström, J and Nyberg Carlsson, M. (2021). "Emulation of the Crazyflie 2.1 Hardware for Embedded Control System Testing"

[2] Renode, "Renode", 2020, URL: `https://renode.io/`

[3] Bitcraze AB, "Crazyflie 2.1 | Bitcraze", 2020, URL: `https://www.bitcraze.io/`.