<KIV> KATEDRA INFORMATIKY A VÝPOČETNÍ TECHNIKY

**Petr Kocián**

**Master’s thesis**

Computer Science and Engineering
**Computer Engineering and Networks**
2023/2024

Thesis supervisor:
**doc. Ing. Tomáš Koutný, Ph.D.**

FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

# Real-Time Concept for SmartCGMS

## Abstract

*SmartCGMS is a framework for continuous glucose monitoring (CGM) systems. It has been used extensively for testing and simulations, but it needs to improve its support for execution on portable devices (e.g., Raspberry Pi Zero or ESP32). This paper proposes and implements a real-time SmartCGMS concept based on FreeRTOS. It has been deployed as a native application on an ESP32 and a Raspberry Pi Zero. For experimentation with loadable modules, the concept was compiled to WebAssembly (WASM), which can be executed on ESP32, Raspberry Pi, x86-64/AMD64, and on the web.*

## Introduction

The purpose of executing SmartCGMS on a System on a Chip is to create a closed-loop CGM system. A closed-loop CGM system is a set of devices that can autonomously control diabetic patients’ blood glucose levels based on measurements from a sensor by injecting insulin using an insulin pump. The SmartCGMS framework is capable of implementing a closed-loop system with SmartCGMS components. The goal of this thesis was to propose, implement, and evaluate a SmartCGMS concept for Raspberry Pi Zero (ARMv6) and ESP32 (Xtensa). The concept must maintain compatibility with desktop devices (x86-64/AMD64), where computationally intensive pre-clinical simulations take place.

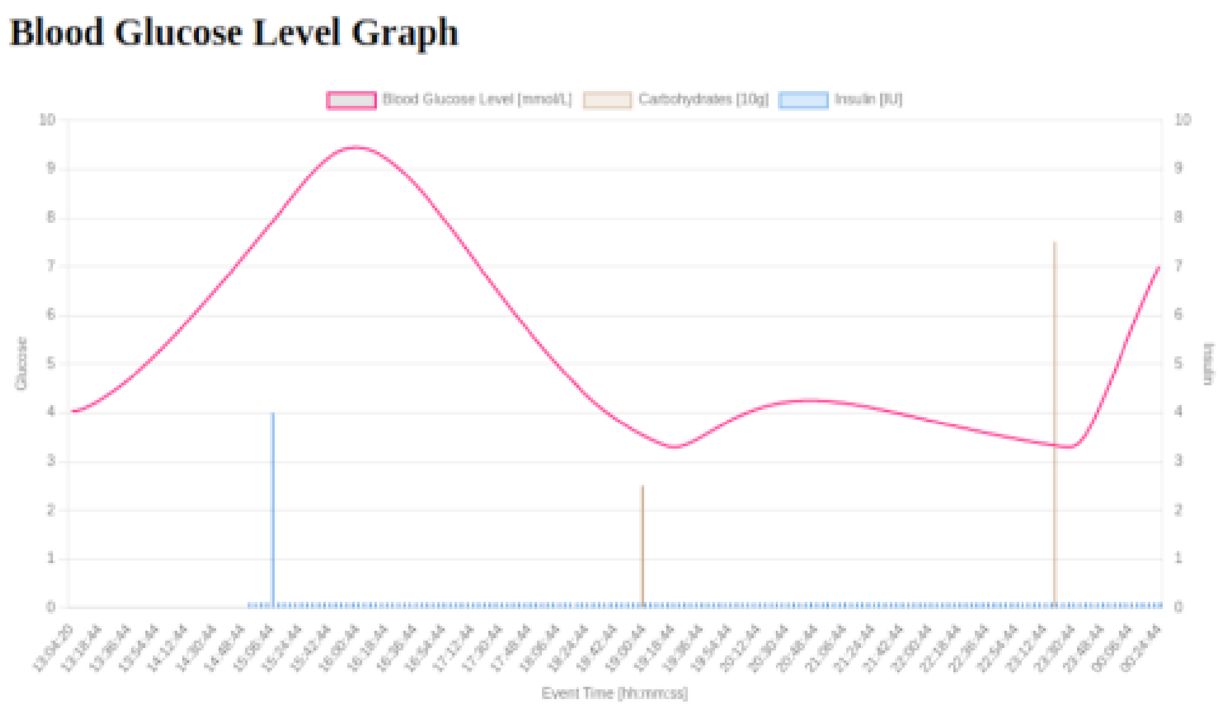
## WebAssembly

In addition to targeting ESP32 and Raspberry Pi Zero W running FreeRTOS, the concept was compiled to WASM. This allows us to execute the SmartCGMS concept on the web and on any device running a WASM runtime. WASM modules can be updated during runtime without restarting the device. Additionally, the modules are platform-agnostic. We can visualize and evaluate the behavior of a SmartCGMS WASM module using *in-silico* models in a web application and then deploy the same module directly to a portable device. This workflow can be seen in the figure.

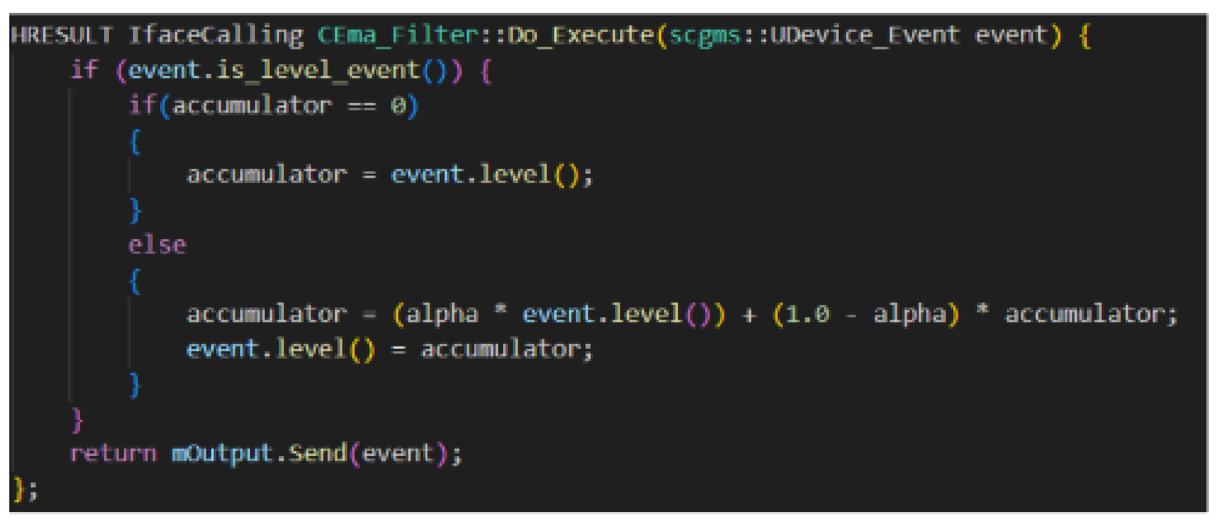
## Evaluation and Results

The first part of the evaluation focused on verifying the correct functionality of the SmartCGMS concept on the newly introduced platforms. The results confirmed that the SmartCGMS concept works as expected on all supported platforms.

The second part of the evaluation focused on comparing the SmartCGMS concept performance when compiled as a native application versus a WASM module. The evaluation was conducted on GNU/Linux using WASM micro runtime (WAMR). The results were in line with findings published from other, independent benchmarks. The execution time has prolonged, but it still meets practical deadlines.

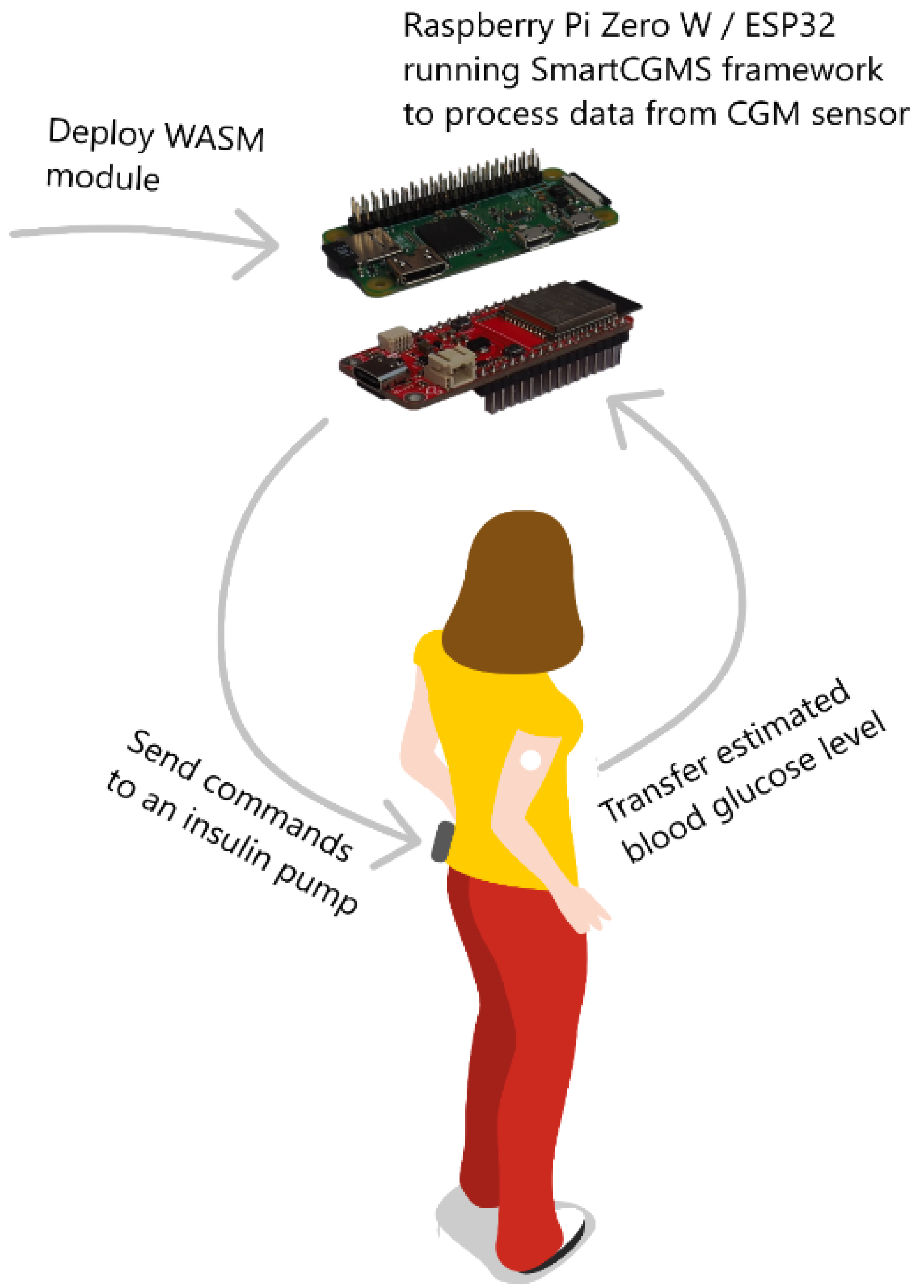


Visualize behaviour of the SmartCGMS WASM module on the web



```
HRESULT IfaceCalling CEma_Filter::Do_Execute(scgms::UDevice_Event event) {
    if (event.is_level_event()) {
        if(accumulator == 0)
        {
            accumulator = event.level();
        }
        else
        {
            accumulator = (alpha * event.level()) + (1.0 - alpha) * accumulator;
            event.level() = accumulator;
        }
    }
    return mOutput.Send(event);
};
```

Develop the SmartCGMS concept application


Deploy WASM module
Raspberry Pi Zero W / ESP32 running SmartCGMS framework to process data from CGM sensor
Send commands to an insulin pump
Transfer estimated blood glucose level

## The SmartCGMS Concept

Only necessary parts of SmartCGMS were modified to maintain maximum compatibility across high-performance and portable platforms. In addition to modifying the SmartCGMS codebase, it was necessary to provide a way how to compile SmartCGMS components into the concept. A preprocessor tool that modifies source code files of SmartCGMS components was implemented. It allows the compilation of SmartCGMS components across all supported platforms.

WAMR is also available on ESP32, but the SmartCGMS WASM module was too large for its limited DRAM. This issue could be addressed either by a SW upgrade (e.g., using a different WASM interpreter) or by a HW upgrade.

## Conclusion

The proposed concept was successfully implemented and tested. It offers a way to execute SmartCGMS on a portable device, with a real-time operating system. Future work should focus on practical experimentation with such devices and making any empirically justified modifications.