WEB ASSEMBLY OPERATING SYSTEM FOR IOT DEVICES.



**UNIVERSITY OF NAIROBI**

**WEB ASSEMBLY BASED OPERATING SYSTEM FOR IOT DEVICES**

By James Kiarie

Registration Number: P15/ 1667/2019

Supervised by Dr. Wanjiku Ng’ang’a

November 2022

This project proposal has been submitted in partial fulfillment of the requirements of the Bachelor of Science in Computer Science as offered by the University of Nairobi.

# Acknowledgements

Dr. Wanjiku, for kind guidance.

Phillip Opperman, writer of “Writing an OS is Rust” blog. The blog singlehandedly made OS development to stop looking like a craft reserved for wizards.

Stephen Marz, writer of the “The Adventures of OS” blog. Took the OS development tutorials beyond just writing “hello world”.

Lin Clark, for simplifying complex webassembly jargon using mortal language... and simple cartoon illustrations.

The writers of “Wasmachine: Bring IOT up to speed with Web Assembly OS” research paper. This was the bedrock of this project.

All the Rust, wasm and riscv teams for creating an awesome well\_thought\_out tech.

# Introduction

It is a dream of many a computer scientist to control matter. To make all matter all around us programmable.

One step towards this direction is through embedded programming. There was a time when embedded programming was simple, all you had to was to read the data sheet of a piece of hardware, abstract that data sheet using data structures and finally manipulate the exposed registers using MMIO programming. Deployment was simple too, sealing off the programming jack pin was enough. Maintaining the embedded software was not a common occurence.

But now, things have changed. Things have become complex. Both the hardware and software are more complicated. For example, reading the hardware data sheet is not enough, you have to understand the ISA, the corresponding hardwired security implementations and additional compatible circuit extensions. Embedded software now deals with network connectivity... Bringing in a whole set of cybersecurity modules and cloud integration drama.

Now the IoT architecture roughly looks like this [1] :

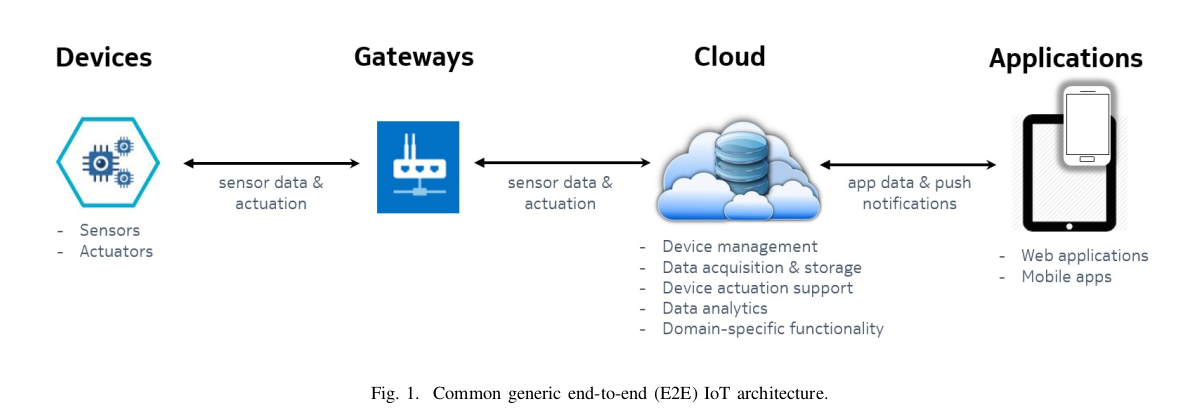
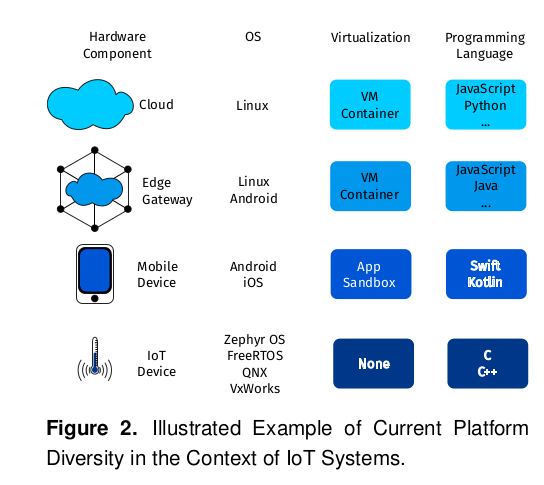


Figure 1 Image borrowed from Taivalsaari, On the development of IoT systems [1]

As a result of the above IOT architecture, building an average IOT system requires the development team to use a lot of unassociated technologies. For example; Assembly for the hardware, C/Rust for embedded programming, distributed programming for building the immediate network infrastructure, docker and kurbenetes for implementing microservices over the cloud, web languages to build a website that acts as a remote interface to the embedded devices... And probably kotlin for a mobile app that interacts with the embedded devices.

Here is a rough illustration of the different technologies used [2] :



This complexity can be summarized to three causes ; heterogeneity of the devices used, inconsistency of the communication protocols used between the heterogenous devices and the necessity to use specialized tools and programming languages.

Below are the solutions as proposed in the paper by Mikkonen [2].

To solve the problem of heterogeneity of devices, Mikkonen proposes a universal API for that describes the abstraction and interaction with known generic devices. For example, the API should specify how to abstract and interact with cameras or heat sensors.

To solve the inconsistency of communication protocols used between heterogenous devices, Mikkonen proposes a standard set of communication protocols to be specified for each known heterogeneous interaction. The paper was in support for the use of existing web protocols for simplicity and easy adoption.

The problem of having too many tools and programming languages involved in development is solved by making the resultant programs portable and wrapped up using APIs. The portability is achieved by packaging the programs as containers or setting up compatible virtual machines on top of the incompatible execution environments.

The first two solutions have been partially solved by initiatives such as Web of Things Standards (WoT) and Open Communication Foundation (OCF).

The last solution is majorly solved through the use of container technology like Docker. There have been propositions and early implementations of using web assembly containers