Witekio

Over-the-Air Update System for Microcontrollers

Full Metal Update

Team: Lerzan Cengiz, Connor Kafka, Anusha Kamat
Industry Mentors: Cedric Vincent, Adrien Leravat

ELECTRICAL & COMPUTER ENGINEERING

PROBLEM STATEMENT

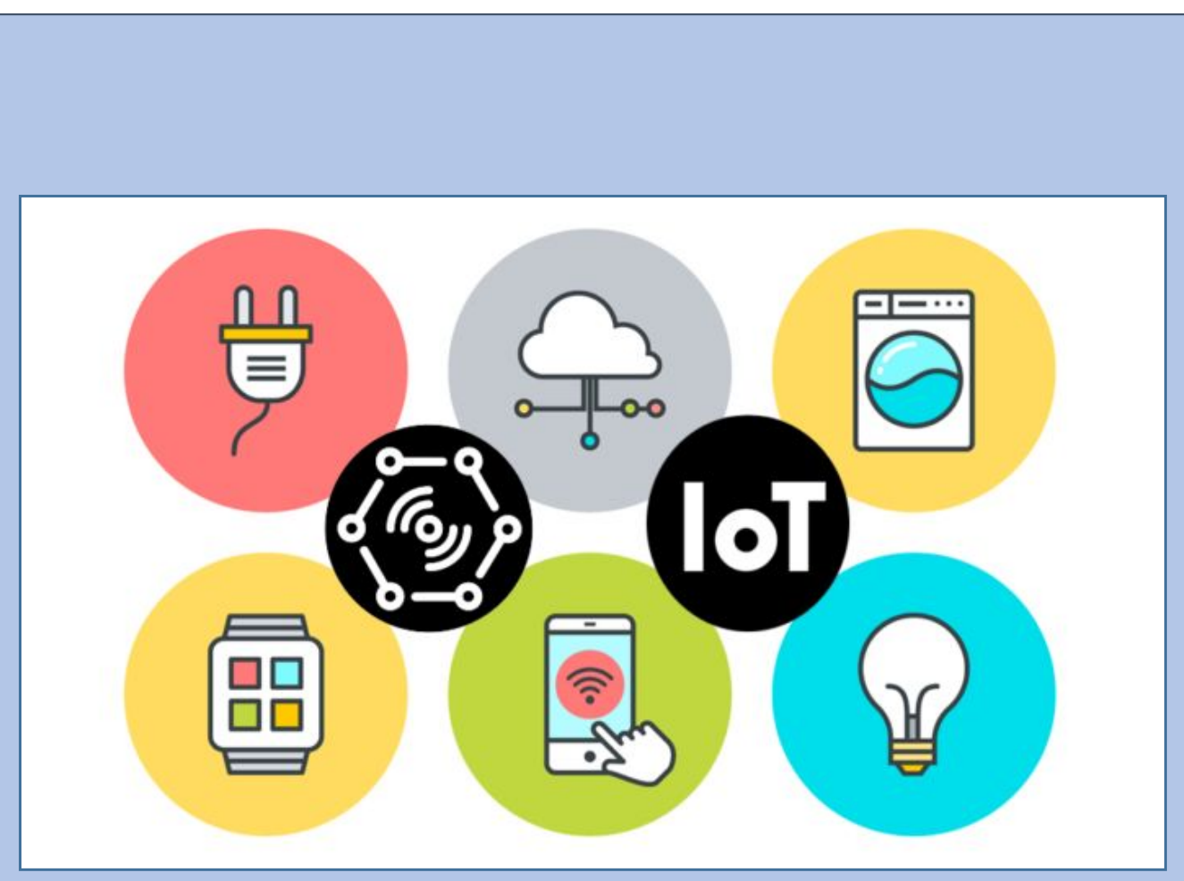


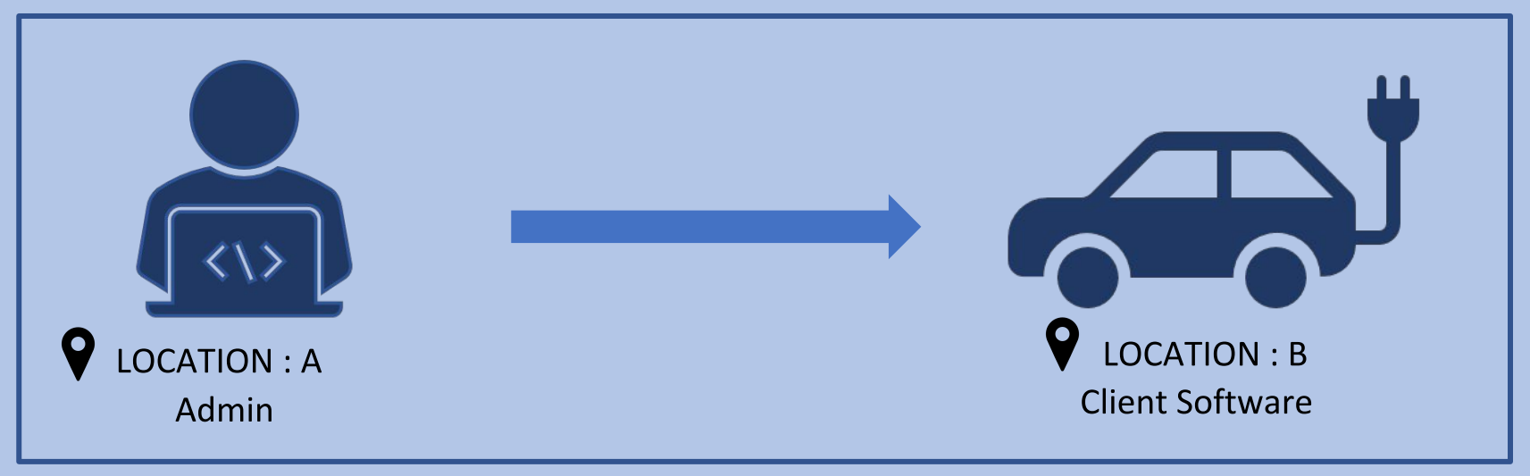
Fig1: IOT devices and connectivity[1]

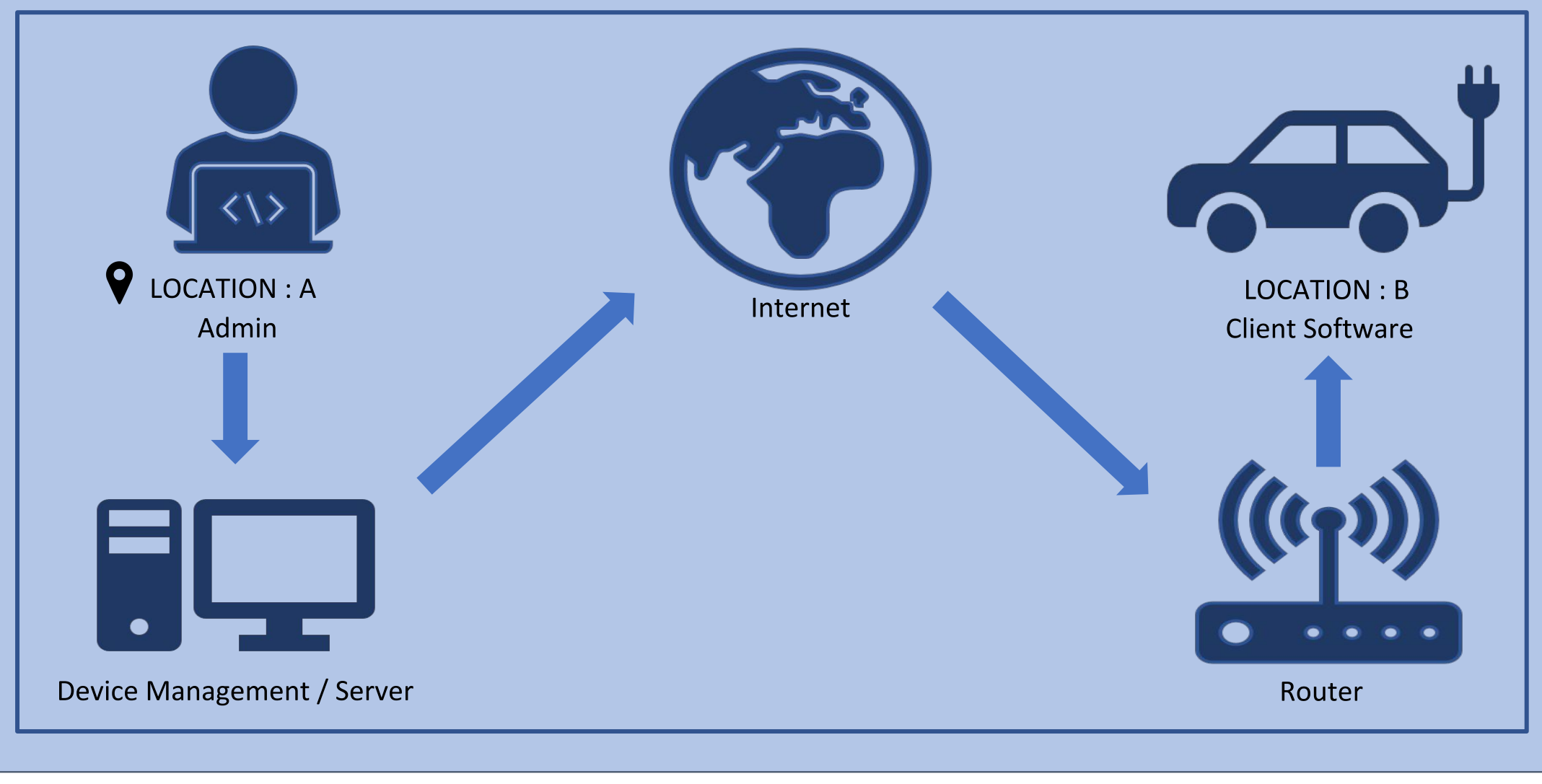
In today's world, there are so many active devices within the Internet of Things (IoT) field. We need a mechanism to constantly update, upgrade, and maintain these devices.

We need an automated process that can be initiated from a single location to simplify device management and provisioning.

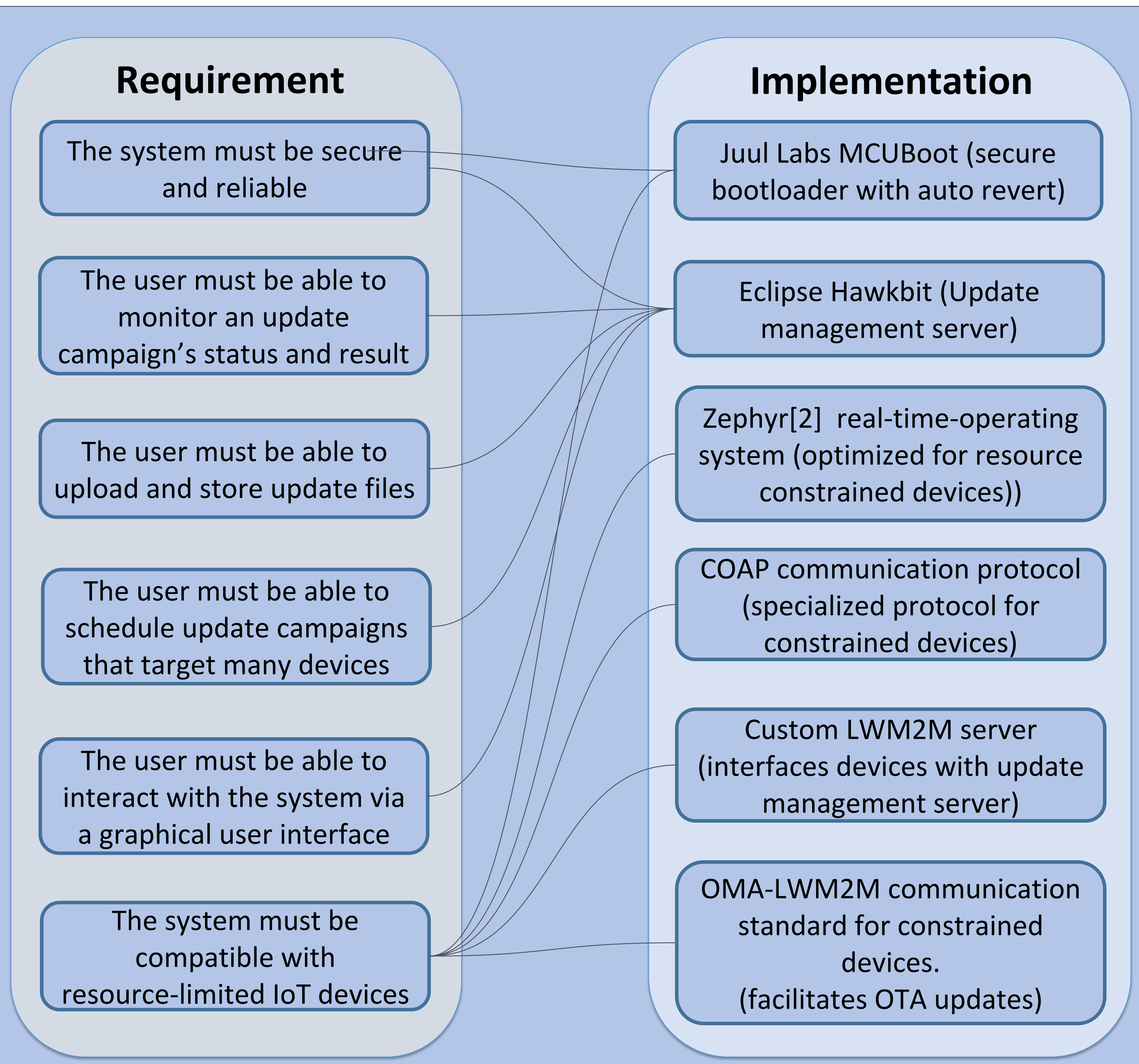
IOT devices need to managed and updated remotely to:

- Provide security updates.
- Add new features.
- Monitor device health.
- Reach devices located in remote areas.
- Reduce logistic delay for updates.





REQUIREMENTS



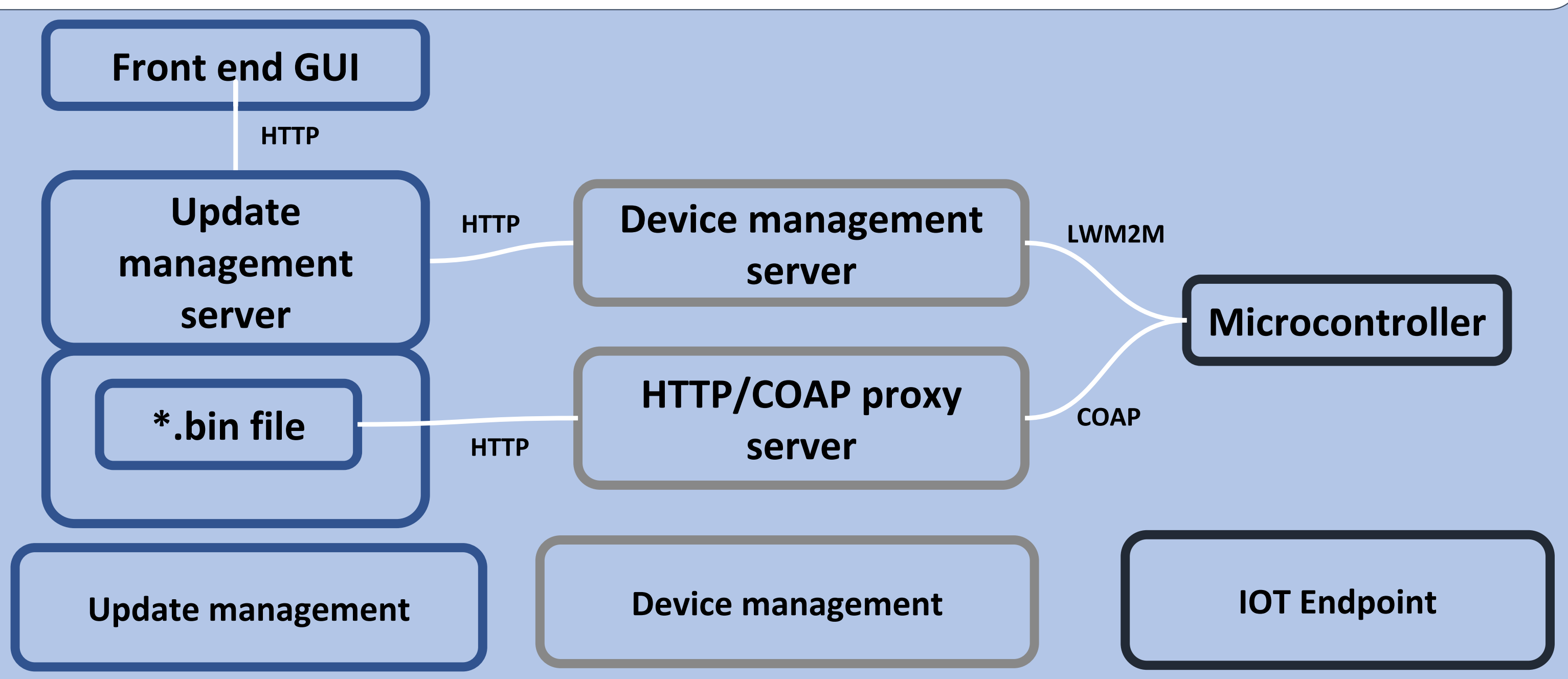
Requirement

- The system must be secure and reliable
- The user must be able to monitor an update campaign's status and result
- The user must be able to upload and store update files
- The user must be able to schedule update campaigns that target many devices
- The user must be able to interact with the system via a graphical user interface
- The system must be compatible with resource-limited IoT devices

Implementation

- Juul Labs MCUBoot (secure bootloader with auto revert)
- Eclipse Hawkbit (Update management server)
- Zephyr[2] real-time-operating system (optimized for resource constrained devices))
- COAP communication protocol (specialized protocol for constrained devices)
- Custom LWM2M server (interfaces devices with update management server)
- OMA-LWM2M communication standard for constrained devices. (facilitates OTA updates)

IMPLEMENTATION



Front end GUI
HTTP
Update management server
*.bin file
HTTP
Device management server
HTTP/COAP proxy server
HTTP
LWM2M
COAP
Microcontroller
Update management
Device management
IOT Endpoint

GOALS

- Upload multiple binary files.
- Schedule update rollouts to many devices.

- Monitor:
  - Device status.
  - Device usage and uptime.
- Command updates.

- Share status with server.
- Apply an update.
- Run customer application code.

TECHNOLOGIES

- Witekio Full Metal Update front end
- Eclipse Hawkbit update management server

- Eclipse Californium COAP/HTTP Proxy
- Eclipse Leshan LWM2M server library

- Zephyr RTOS LWM2M library
- Foundries IO LWM2M client implementation

FUNCTIONAL DESCRIPTION

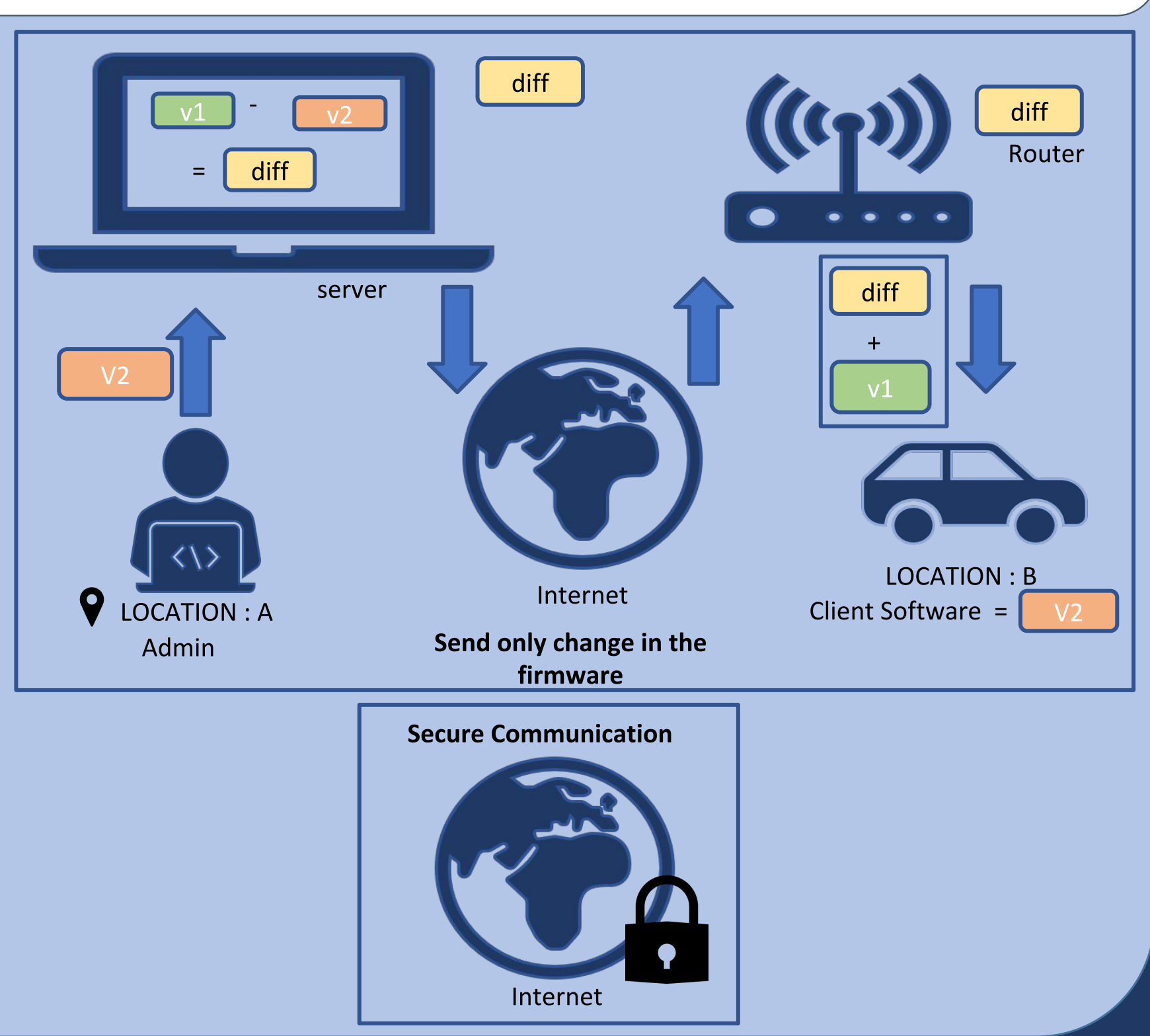
- Hawkbit interacts with the front-end GUI.
- Hawkbit stores update files.
- Hawkbit provides a REST API for Leshan to poll.

- The device management server polls the update maangement server for available updates and forwards the download link to the microcontroller.
- The proxy translates COAP requests to HTTP requests.

- Ethernet connectivity
- Booloader configured for image swap mechanism.

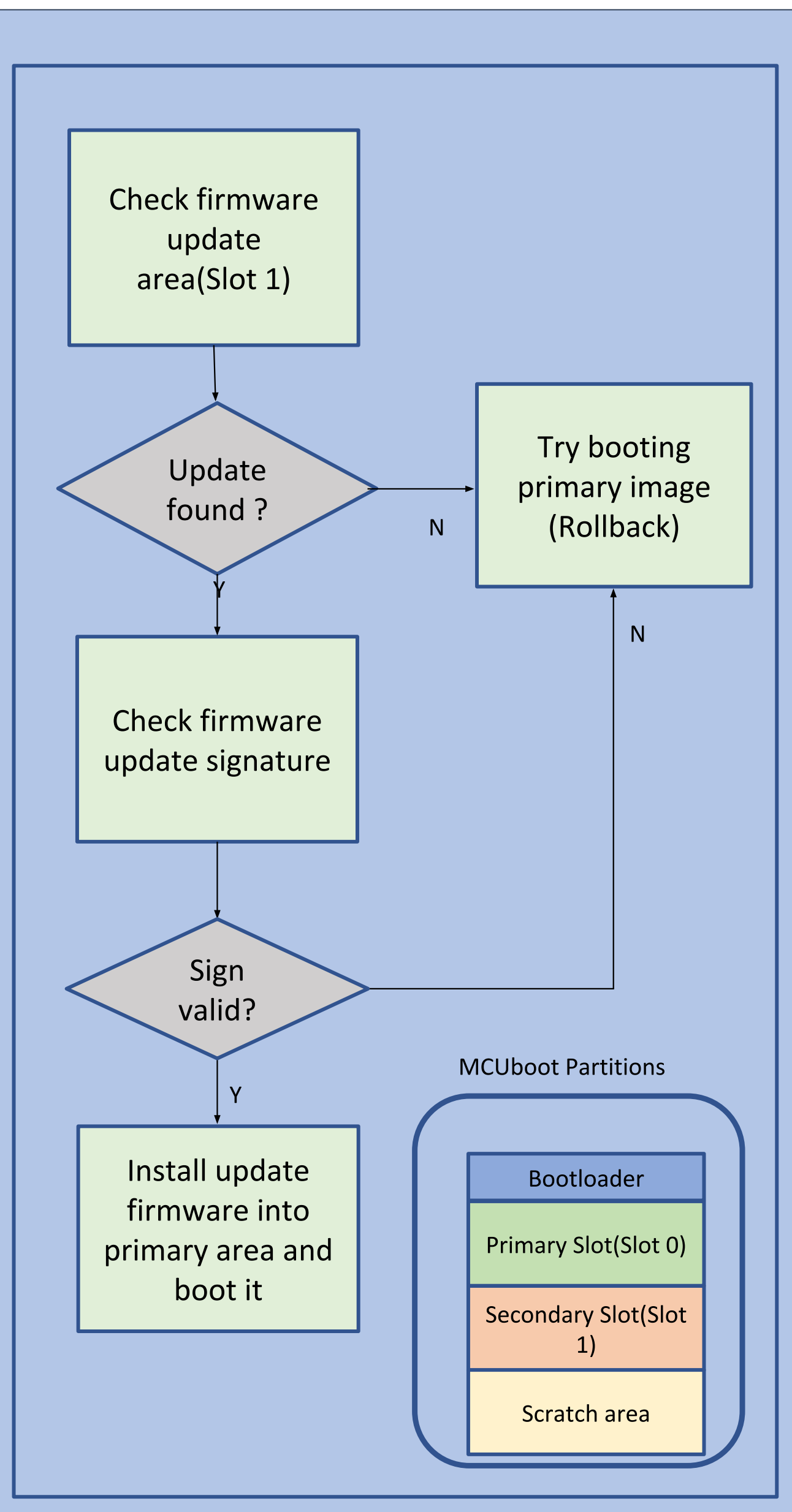
FUTURE WORK

- Send only a difference file in case of small changes. Patch the difference file to already present firmware in the microcontroller and create updated version.
  - Reduces transmission bandwidth.
  - Reduces power consumption.
- Send the firmware using the Transfer Layer Security (TLS) protocol for secure binary file transfers.



RESULTS

- ZephyrOS runs on the microcontroller, which has a secure bootloader in place.
- The client software runs on the microcontroller and checks if a valid firmware update is received from a known source.
- Once the client can determine that there is a firmware update, the client sends a COAP request, which is translated into an HTTP request to the server to request and downloads the update image.
- The server runs and manages the deployment of the firmware image to the appropriate client device.
- The update management application (Hawkbit) provides a GUI to queue the firmware updates for various client devices.
- Lightweight Machine-to-Machine (LWM2M) protocol is used to communicate.



Check firmware update area(Slot 1)
Update found ?
N
Try booting primary image (Rollback)
Y
Check firmware update signature
Sign valid?
N
Y
Install update firmware into primary area and boot it
MCUboot Partitions
Bootloader
Primary Slot(Slot 0)
Secondary Slot(Slot 1)
Scratch area

CONCLUSION

- We were able to send firmware images successfully over the air to microcontrollers.
- The GUI displays which devices are attached to the system.
- The server monitors resources on the devices and enables the user to execute a firmware update remotely.
- The client can send a request to retrieve the update image
- Client GET requests are translated via a proxy to be compatible with an HTTP server
- The client verifies the source of the incoming firmware image and its validity.
- The secure bootloader enables rollback mechanism in case of a faulty firmware image.

REFERENCES

1. Mimoso, M. (2017). *Legislation Proposed to Secure Connected IoT Devices*. [online] Threatpost.com. Available at: https://threatpost.com/legislation-proposed-to-secure-connected-iot-devices/127152/ [Accessed 23 May 2019].
2. Docs.zephyrproject.org. (2019). *Introduction — Zephyr Project Documentation*. [online] Available at: https://docs.zephyrproject.org/latest/introduction/index.html [Accessed 28 May 2019].