Combining Bluetooth Mesh and KNX

The Best of Both Worlds

Mario Noseda, Manuel Böbel, Marcel Schreiner, Andreas Rüst
Zurich University of Applied Science (ZHAW)
Institute of Embedded Systems (InES)
Winterthur, Switzerland
mario.noseda@zhaw.ch
andreas.ruest@zhaw.ch

***Abstract*—Bluetooth Mesh (BT Mesh) is a promising wireless technology for building automation. At the same time, KNX is a well-established building automation system that has a vast installed base. Specifically, the strength of KNX lies in its proven semantic models. These models are the foundation for interoperability and the implementation of larger systems. The presented project demonstrates how a user can easily connect a new BT Mesh system to a well-established, wired KNX building automation system. Notably, the project achieves this through a self-developed stateless gateway, which allows controlling BT Mesh devices from the KNX network and vice versa. As a result, it is possible to leverage existing management systems from KNX building automation systems in BT Mesh networks. Furthermore, the project validates this concept using Home Assistant, a well-known open-source home automation platform and demonstrates, that heterogeneous KNX and BT Mesh systems are feasible.**

Keywords—Bluetooth Mesh, KNX, Home Assistant, gateway, topology scan, mesh networks,

I. Introduction

Many of today's buildings employ a cable-based building automation system. KNX [1] is among the most used and deployed building automation standards. Such systems not only encompass sensors and actuators with local control loops but also higher-layer automation stations and management software to control the building. Importantly, building automation, including its software, constitutes a significant investment for a building owner. Replacing an installed system comes at huge costs.

However, in the case of extension of existing installations, cable-based systems have their limits. E.g. the installation of additional lights and switches requires running the twisted-pair (TP) cable (most used physical communication medium for KNX) to all newly installed devices. Installation of such wires is often not feasible for cost reasons and constraints of the building structure. In contrast, a building owner can deploy wireless Bluetooth-Mesh-enabled devices with a minimal amount of installation effort. However, maintaining a new system in parallel to the already running building control system over KNX increases complexity and cost significantly.

So, what if you could easily combine the best of both worlds? The study covered in this paper aimed to investigate precisely this question by extending a preexisting KNX network with a new wireless Bluetooth Mesh network. An existing building that has already been fitted with lights and switches controlled over KNX using TP cables served as a fictional scenario for this use case.

Importantly, the two networks connect through a stateless gateway. This gateway is entirely agnostic to the devices and their associated models. Neither data nor addresses need to be reformatted or translated in the gateway. All the gateway does is to forward the KNX messages transparently in both directions. This approach has the benefit that updates to devices and topology have no impact on the gateway. This approach provides maximum flexibility as the gateway does not require an update if a new device type is introduced. In particular, our approach makes use of the option to create custom Bluetooth Mesh models. Models are objects used for describing the communicational abilities of a node. In our case, we have created custom Bluetooth models matching the KNX message format. Furthermore, "Home Assistant" has been used to showcase the connectivity potential if such a system is connected to an appropriate home automation platform. As a result, various other interfaces and protocols can be added to the current system with minimal effort.

This paper is structured accordingly. Section II discusses the protocols and tools used in this project. In the following section, we continue by describing the developed system itself. Section IV discusses the hardware used, and the key findings acquired during the implementation of the system and the final section draws appropriate conclusions.

II. Protocols & Tools

The project made use of various preexisting protocols and tools, which we will introduce here.

A. BT Mesh

BT Mesh has been published in July 2017 and introduces the capability for m:m (many-to-many) communications [2]. The message-oriented protocol makes use of a publish/subscribe architecture using various address types to connect logically associated devices efficiently. Messages are distributed using managed flooding, which means that nodes simply broadcast a packet and all receiving nodes compare the included address with their subscriptions and process the message if necessary. Nodes that implement the relay feature will then broadcast the message themselves and are responsible for its propagation through the whole network. Additionally, nodes can implement a low power and a friend feature to allow especially battery-constrained devices to be integrated into a BT Mesh network while still being able to go into a deep sleep state. Friend nodes receive and store messages meant for low power nodes while low power nodes then poll their friend node for missed messages after they wake up.

BT Mesh is a pure software stack and does not include its own physical layer but instead makes use of Bluetooth Low Energy (BLE). BT Mesh packets get sent as the payload of conventional BLE packets which theoretically enables all BLE ready devices to send and receive BT Mesh messages with just a driver update. Lastly, a provisioner (usually a cellphone) is needed to include new nodes into a network. During provisioning, the unprovisioned node is supplied with provisioning data (network key, application keys, etc.) which it needs to further act as a part of the network. Moreover, restarting or power cycling the node does not require the node to be provisioned again. Therefore, once provisioned, a node can start-up and immediately send and receive messages.

B. KNX

KNX is an open standard for building automation that may be used domestically as well as commercially. With its predecessor "EIB" dating back to 1990 [3], KNX has been well-established and grew to have a vast installed base. Devices may be connected using TP cable, powerline, radiofrequency or ethernet [4]. In a given KNX network, each appliance has a unique device address which is needed for the configuration of each unit. More importantly, group addresses can be assigned to the various devices which instructs them to process all messages sent to these group addresses. Furthermore, KNX makes use of a semantic library containing Datapoint Types (DPT). A DPT describes how the message is formatted (unsigned integer, float, etc.) and how the data is encoded (degrees Celsius, meters, etc.). However, DPTs are only used during configuration. They need to be stored in the devices as KNX messages do not carry any information about the DPT used to encode the payload.

C. Zephyr

We used Zephyr as the basis for the firmware running on the BT Mesh nodes. It is part of the Linux Foundation and is a real-time operating system (RTOS) designed for resource-constrained devices [5]. Thanks to Kconfig [6] and device tree [7], the Zephyr kernel, as well as the application, are highly configurable and platform-independent. Additionally, certified software stacks, including BLE, BT Mesh, OpenThread, etc. allow fast development.

D. Home Assistant

Home Assistant is an open-source platform for home automation usually run on a Raspberry Pi [8]. Instead of installing Raspbian and then running Home Assistant, hass.io [9] can be used, which is a full Linux distribution that includes everything necessary and is already setup correctly. So-called "Integrations" are used to connect Home Assistant to the various interfaces and protocols. At the time of this writing, users can choose from more than 1500 integrations (KNX, Google Home, Philips Hue, etc.) or create custom ones if necessary.

III. APPLICATION

Fig. 1 represents the architecture of the system. We have implemented various functionalities in order to demonstrate the different use cases possible within the scenario described in section I. First, a control loop for lighting (subsection A) represents two distinct forms of communication within the BT Mesh network itself. Second, connecting KNX and BT Mesh through a gateway shows the capability to communicate with another network (subsection B). Third, a network topology scan allows a system integrator to strategically place and configure the nodes in order to keep the BT Mesh from getting to sparse/dense (subsection C). Forth, adding Home Assistant shows how an existing control system for building automation

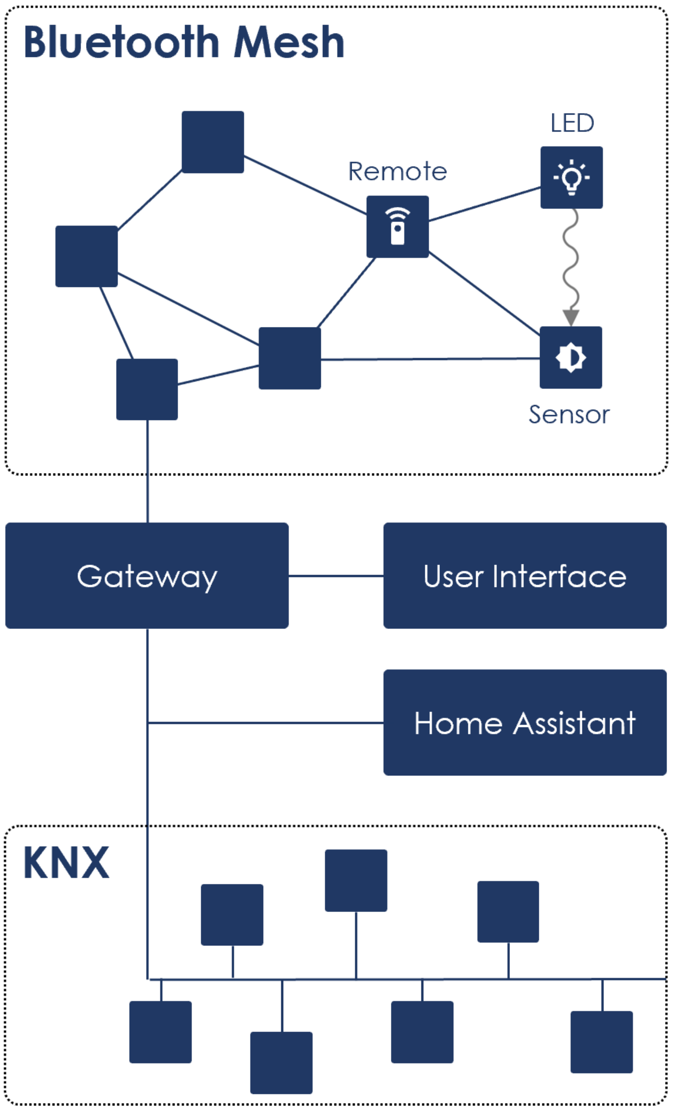


Fig. 1: Architecture of the system

can be leveraged and seamlessly extended to the BT Mesh part of the network. This adoption is made possible through the stateless gateway and the semantic KNX models implemented on the BT Mesh nodes.

A. *Control loop for lighting*

Implementing a control loop in the BT Mesh network demonstrates two distinct forms of communication: sporadic messages with the target value and periodical updates containing the actual value of the control loop.

A possible use case for such a loop could be as follows: The ambient brightness in a room may change drastically based on daytime and current weather conditions. However, constant brightness is usually desired. Moreover, lamps close to big windows could sometimes turn off entirely while parts of the room furthest away from the windows are still relatively dark. Therefore, we have implemented a control loop that does not let the user choose a power level output of the lamps (as a lot of conventional, dimmable lamps do) but rather the desired brightness. This control loop consists of three elements: (a) A remote node to send on/off and brightness signals triggered by the user, (b) the lamp node itself, and (c) a sensor node which periodically sends the ambient brightness to the lamp node. The latter allows for calculating the required amount of artificial light.

B. *KNX extension with BT Mesh*

Typically, BT Mesh nodes implement models predefined by the Bluetooth Special Interest Group (SIG). These models provide a semantic representation and therefore describe the functionality of a node and the structure of the data on a node. Particularly, they represent data points that can be read from or written to on devices like sensors, switches, and lamps. Additionally, BT Mesh allows the creation of custom models in order to handle situations not covered by the generic models.

Converting a KNX message to a BT Mesh message requires a gateway with complete knowledge of all present nodes and a table to translate KNX group addresses into BT Mesh addresses and vice versa. Therefore, instead of using the newly introduced BT SIG models, we implemented a BT Mesh model capable of sending any KNX message in a BT Mesh network. Furthermore, we also adopt the KNX addressing scheme inside our BT mesh network. Fully encapsulating a KNX message inside of a custom BT Mesh KNX model enables the use of a stateless gateway which does not need to know anything about the nodes or networks. This approach offers the benefit that an existing KNX network can be easily extended without extensive configuration of the gateway. As a result, a system integrator can communicate with nodes on the BT mesh with precisely the same tools as he has been using on KNX for years.

Fig. 2 illustrates the involved blocks and the basic compositions of the packets at distinct points. As an example, the user pushes the KNX switch (leftmost block), and a KNX message gets sent to the KNX IP gateway over the TP cable. The IP gateway packages the KNX message in an IP packet and forwards it to our gateway using ethernet. The Raspberry Pi in our gateway is not capable of sending BT Mesh messages itself. Thus, it extracts the KNX message and sends it to the BT interface over UART. The BT interface then packs the KNX

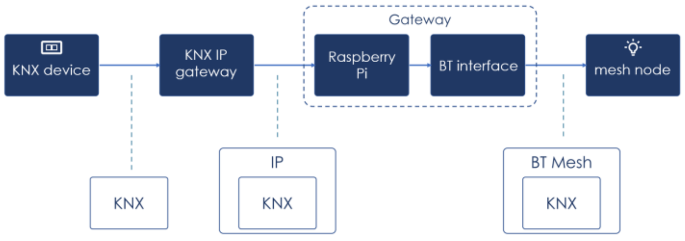


Fig. 2: Connecting KNX with BT Mesh through our gateway

message in our custom BT Mesh KNX model and sends it to the destination node over BT Mesh. The KNX message will then be extracted and parsed by the addressed node. Hence, KNX capable BT Mesh nodes not only need to be provisioned for the BT Mesh network but also need to be supplied with all their KNX configuration data. This procedure entails a somewhat higher configurational effort for these nodes. However, maintaining a gateway every time an address is changed, a node gets removed, etc. is probably more time-consuming and assuredly more error-prone.

C. *Network topology of BT Mesh*

As described in section II.A, relay nodes are used to forward messages for the managed flooding approach. They relay every packet only once, which prevents messages from traveling back and forth (messages also include a time to live (TTL) field, which is used to limit their lifetime). It is vital for these relay nodes to be in radio range of each other. Otherwise, messages may not be able to reach every addressed node in a given network. However, enabling the relay feature in all nodes may result in the network getting congested or completely overloaded. As a result, the planning of the individual features and placement of the nodes should be taken into careful consideration. A network topology scan has been developed during this study which commands the nodes to detect all other nodes in direct radio range and transmit this information to the gateway. First, the gateway instructs all nodes to start a topology scan. Second, all nodes send ping messages with a TTL value of 0 to a broadcast address. Messages will not be relayed, and therefore, only neighboring nodes will receive the message. Third, all nodes reply to the received pings. Forth, nodes save the sender addresses of the received replies in a neighbor list. Fifth, all nodes send their neighbor list to the gateway. After a successful measurement cycle, the gateway displays the current mesh topology using a webpage which allows the system integrator to distribute the relay nodes accordingly. Fig. 3 displays a possible outcome of such a topology scan.

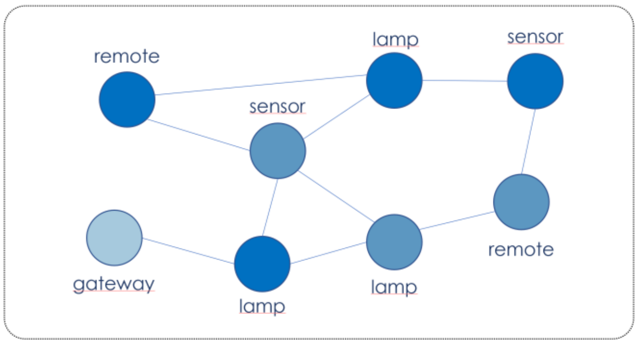
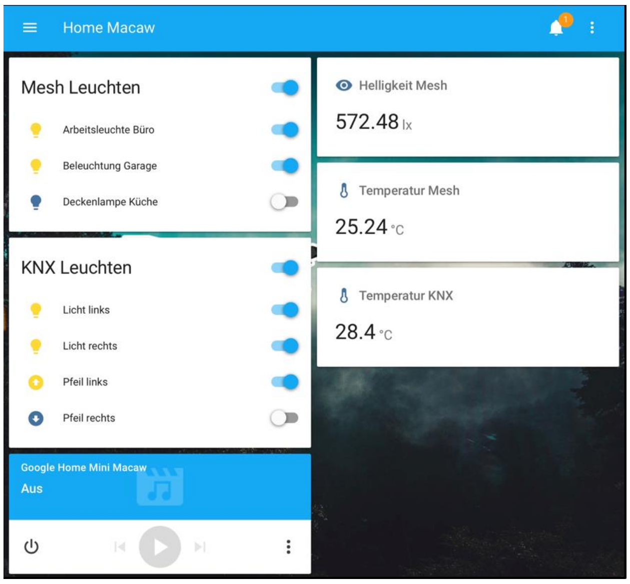


Fig. 3: Topology example for a BT Mesh network

D. Connectivity through Home Assistant

Home Assistant is one of a multitude of home automation platforms available today. It can be connected with KNX by using a KNX IP module that enables communication with a KNX twisted-pair network through TCP/IP or UDP/IP. In our case, Home Assistant serves as an example to show that an existing control system for KNX building automation systems can be used in a straightforward way to manage BT mesh nodes. Thanks to the transparent extension of the KNX network to BT mesh, there is no need to integrate newly defined BT mesh models on the control system. In the presented case, Home Assistant does not need to know any BT mesh models.

Moreover, Home Assistant provides an abundance of integrations that allow the already connected KNX and BT Mesh networks to be extended with even more interfaces and protocols. For example, the addition of Google Assistant, Siri, etc. is just a few mouse clicks away. Additionally, Home Assistant provides its own web interface for checking and controlling the system easily. Fig. 4 shows an exemplary Home Assistant dashboard.


Home Macaw

Mesh Leuchten
- Arbeitsleuchte Büro
- Beleuchtung Garage
- Deckenlampe Küche

Helligkeit Mesh 572.48 lx

Temperatur Mesh 25.24 °C

KNX Leuchten
- Licht links
- Licht rechts
- Pfeil links
- Pfeil rechts

Temperatur KNX 28.4 °C

Google Home Mini Macaw
Aus

Fig. 4: Home Assistant dashboard

IV. Results & Findings

The following subsections cover our system capable of handling all use cases described in section III.

A. BT Mesh

Fig. 5 displays the InES Sensornode hardware that we used for the sensor and remote nodes of the lighting control loop. Mainly, it consists of an nRF52840 MCU and various environmental sensors fitted on an evaluation board with buttons, LEDs, etc. in order to streamline development. Additionally, the board is designed with cutouts to enable the user to break out the core containing the MCU and the sensors. Consequently, the Sensornode can be deployed in a small form factor without any unnecessary pins and components usually only needed during development and debugging. Fig. 6 shows

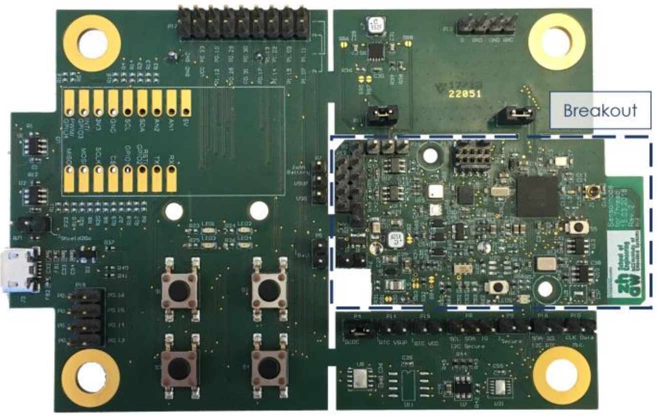


Fig. 5: InES Sensornode

the LEDTubes (LED lamp in the form factor of a conventional fluorescent tube) containing an nRF52832 MCU and radio for BLE, which we used for the lamp nodes of the control loop. With the use of sporadic and periodical messages, we were able to implement the lighting control loop, as described in section III.A. The user can set the brightness using the remote node, which immediately supplies the lamp node with the new setting. The lamp node updates the brightness periodically using the measured data supplied by the sensor node. Furthermore, the remote node is configured with the BT Mesh specific low power feature, whereas the other nodes are configured as relay nodes.

Fig. 6: LEDTube from LEDCity with BLE radio

Moreover, the Nordic SDK was used in a preliminary study to program the BT Mesh nodes instead of Zephyr RTOS. Switching to Zephyr increased our efficiency noticeably and accelerated the development progress. Threads and other RTOS specific advantages simplified the structure of the firmware immensely. Although the configuration of the kernel, race conditions and thread-safety introduced new issues, the benefits outweighed the disadvantages without any doubt.

BLE is well-established and can be found in a lot of today's devices. Using this as the physical layer is a very effective way to advertise BT Mesh as all these devices only need a driver update in order to be BT Mesh ready. Hence, there is no need to produce new radio hardware. However, BT Mesh nodes need to be provisioned once in order to include them in a given network. The BT SIG recommends a cellphone or other mobile device to be used as the provisioner [10]. Unfortunately, the configuration of each node and occasional connectivity issues took a significant amount of time.

B. BT Mesh & KNX gateway

Fig. 8 shows our gateway consisting of a Raspberry Pi and a BT interface which we implemented using an nRF52840 development kit (DK). The Raspberry Pi is responsible for managing the transmission of IP packets, whereas the DK handles the BT Mesh communication. They are connected over

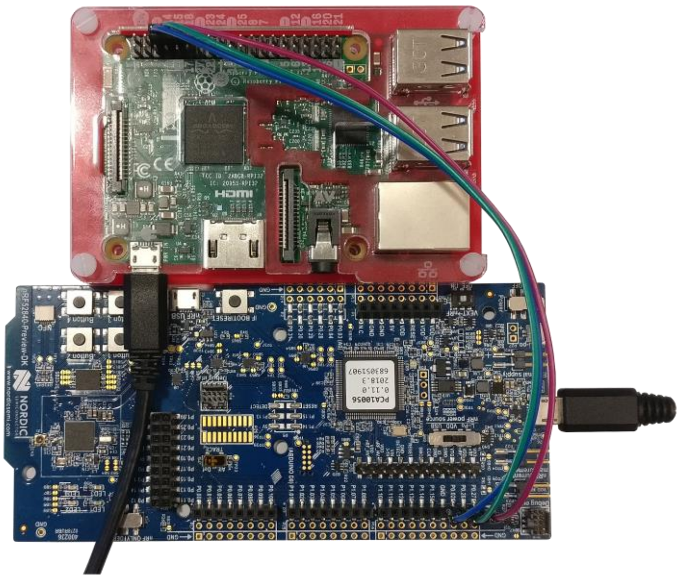


Fig. 8: Gateway hardware

UART using a proprietary protocol to check for transmission errors. As described in section III.B, the gateway does not have to be supplied with any address tables or model mappings as all the KNX messages can be transmitted using the same custom BT Mesh model. Thus, the gateway can run completely stateless and without the need for extensive configuration. Fig. 7 shows the KNX demonstrator which we used to mimic an already existing KNX network. It contains multiple KNX switches, lights, indicators, and a KNX IP gateway.

C. Topology scan

We have implemented a webpage running on the Raspberry Pi of the gateway displaying the last successful topology scan. As discussed in section III.C, care needs to be taken while planning the network structure and position of the relay nodes. It was possible during measurements to overload the network entirely by placing too many relay nodes in direct radio range of each other. Nodes store a user-definable number of messages in order to prevent relaying the same message multiple times. Nevertheless, with enough traffic and nodes, it was possible to let these buffers overflow, which resulted in messages getting relayed repeatedly. This behavior may not be a problem in a small network as the TTL field would be set suitably low. But BT Mesh allows the use of up to 32'767 nodes which provides the possibility of a very dense network using the highest TTL possible in order to reach every node. Such a network will be at risk of congestion if relay nodes are not distributed appropriately.

D. Controlling BT Mesh & KNX through Home Assistant

Setting up Home Assistant on a Raspberry Pi with hass.io was surprisingly simple, especially if compared to the benefits of this extensive platform. We then configured the Home Assistant to communicate directly with the KNX IP gateway

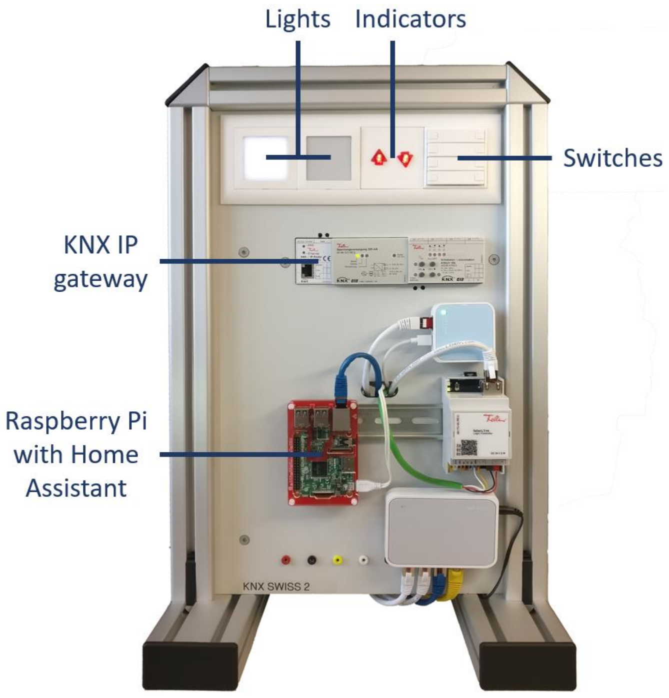


Fig. 7: KNX demonstrator

over ethernet. Nodes in the KNX, as well as the BT Mesh network, can then be controlled equally by Home Assistant thanks to our gateway.

V. Conclusion

In this study, we have shown that an existing KNX network can be easily extended with BT Mesh using a self-made stateless gateway and custom BT Mesh models. Moreover, messaging within BT Mesh itself is not affected by this, demonstrated by the always running control loop for lighting. However, relay nodes need to be placed strategically in order to strike a balance between path redundancy and susceptibility to overload. Using our topology scan, a system integrator can identify sparse/dense sectors and optimize coverage and performance of the network. Lastly, we were able to show that by using Home Assistant, a highly interconnectable network with a vast number of interfaces and protocols can be created with minimal effort.

Acknowledgment

KNX Swiss [11] supplied the project with a fully wired KNX demonstration system which acted as the preexisting KNX network described in section I. LEDCity [12] contributed programmable LEDTubes containing an nRF52832 MCU and BLE radio.

References

[1] KNX, [Online] Available: https://www.knx.org/knx-en/for-professionals/index.php [Accessed Jan. 2020]

[2] Introducing BT Mesh Networking, [Online] Available: https://www.bluetooth.com/blog/introducing-bluetooth-mesh-networking/ [Accessed Jan. 2020]

[3] A History of KNX, [Online] Available: http://www.knxuk.org/images/pdf/A_History_of_KNX.pdf [Accessed Jan. 2020]

[4] KNX Basics, [Online] Available: http://knx.fi/doc/esitteet/KNX-Basics_en.pdf [Accessed Jan. 2020]

[5] Zephyr Project, [Online] Available: https://www.zephyrproject.org/ [Accessed Jan. 2020]

[6] Kconfig, [Online] Available: https://www.kernel.org/doc/Documentation/kbuild/kconfig-language.txt [Accessed Jan. 2020]

[7] Device tree, [Online] Available: https://www.devicetree.org/ [Accessed Jan. 2020]

[8] Home Assistant, [Online] Available: https://www.home-assistant.io/ [Accessed Jan. 2020]

[9] hass.io, [Online] Available: https://www.home-assistant.io/hassio/ [Accessed Jan. 2020]

[10] Provisioning Bluetooth Mesh, [Online] Available: https://www.bluetooth.com/blog/provisioning-a-bluetooth-mesh-network-part-1/ [Accessed Jan. 2020]

[11] KNX Swiss, [Online] Available: https://www.knx.ch/knx-chde/ [Accessed Jan. 2020]

[12] LEDCity, [Online] Available: https://ledcity.ch/ [Accessed Jan. 2020]