LoRaBridge 2

Endbericht | Call 18 | Projekt ID 6693

Inhalt

1 Einleitung

Heute, im Jahr 2025, gibt es mehrere Smarthome-Gadgets für die Einrichtung einfacher Automatisierungen, wie z. B. die Regulierung von Luftfeuchtigkeit und Temperatur oder die Steuerung der Beleuchtung. Ein traditionelles Einrichtungsverfahren für solche Automatisierungen beinhaltet den Kauf einiger Geräte und eines Gateways von einem ausgewählten Hersteller. Die Geräte verbinden sich mit dem Gateway, das die Daten über eine Internetverbindung auf ein Endgerät des Nutzers, wie z.B. ein Smartphone, überträgt. Die Abhängigkeit von einer permanenten Internetverbindung macht eine solche Einstellung ungeeignet für Anwendungsfälle, in denen die Automatisierung z.B. an einem Ort außerhalb der Reichweite des Heim-WLAN betrieben werden soll. Für solche Szenarien stehen den Benutzer*innen Offline-Lösungen zur Verfügung, z.B. Steckdosen mit Zeitschaltfunktion oder Open-Source-Tools, die nur manuell aktualisiert/konfiguriert werden können.

Um das oben genannte Problem zu lösen, haben wir im Rahmen des Projekts LoRaBridge 2 eine neue Konfigurationslösung für die Hausautomation demonstriert. In unserer Lösung wurde eine drahtlose Langstreckenverbindung über LoRaWAN aufgebaut, um Automation-Setups und Parameteraktualisierungen an die Automation Bridge-Einheit (die an einem entfernten Standort platziert ist) zu senden. Zweitens werden Sensormesswerte und Laufzeitinformationen über die Automatisierungen über dieselbe Verbindung an einen Benutzer*in übertragen. Dieses Projekt basiert auf dem (von Netidee finanzierten) Basisprojekt LoRaBridge, das eine Reichweitenerweiterung für Zigbee-basierte Smarthome-Sensoren ermöglichte. Daher kann die LoRaBridge 2 als Erweiterung des ursprünglichen Konzepts für die Hausautomation interpretiert werden.

Der innovative Aspekt des Projekts besteht aus zwei Hauptkomponenten der Software:

1. **Komprimierung von Konfigurationsdaten für die Automatisierung**
2. **Intuitive Benutzeroberfläche**

Obwohl die direkte Übertragung von Konfigurationen für die Heimautomatisierung über LoRaWAN theoretisch möglich ist, leidet die praktische Umsetzung unter einer sehr langen Programmierungslatenz, da die Größen der Konfigurationsdateien in der Regel im Bereich von 5-50 kB liegen. Dies würde zu einer Setup-Latenz zwischen Tagen und Wochen führen. Mit unseren Kompressionsmethoden kann die Latenz auf wenige Minuten reduziert werden. Für die Automatisierungskonfiguration selbst haben wir ein Konzept im visuellen Programmierstil gewählt, das dem beliebten NodeRED-Framework ähnelt. Im Gegensatz zu NodeRED haben wir uns darauf konzentriert, eine Benutzeroberfläche zu entwerfen, die nur die notwendigen Komponenten enthält, um einfache Automatisierungen zu erstellen, die man an entfernten Standorten einsetzen kann. Daher ist es nicht erforderlich, Konfigurationsdateien manuell zu berühren oder zu programmieren, um die gewünschten Automatisierungsfunktionen zu implementieren. In unserer Hoffnung soll dies auch Anwender mit wenig bis gar keinen technischen Kenntnissen dazu einladen, die Ergebnisse von LoRaBridge 2 zu nutzen.

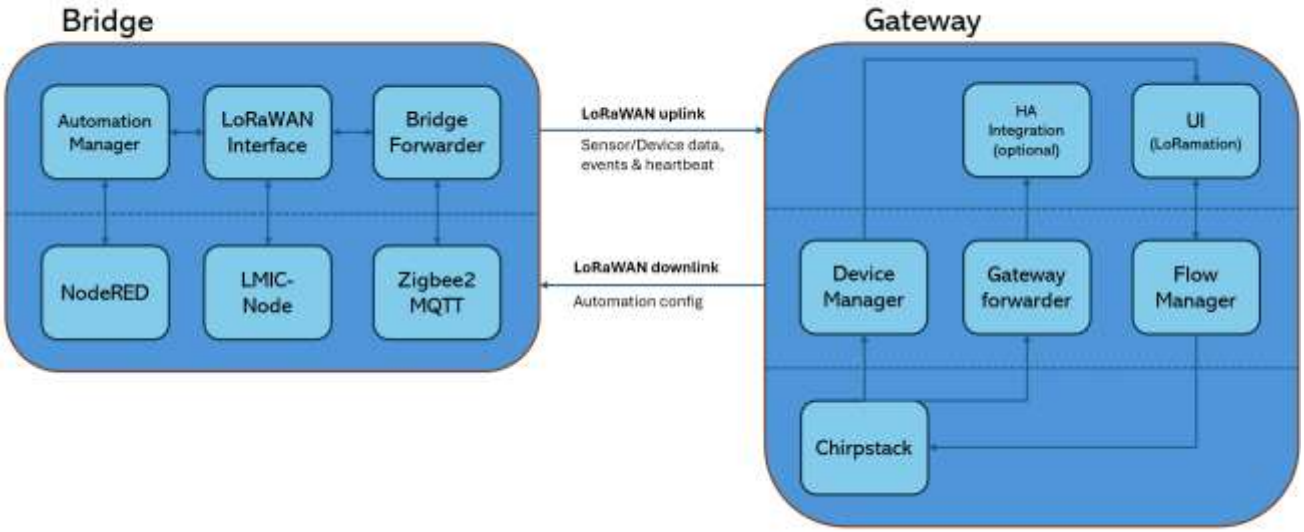
2 Projektbeschreibung

In diesem Kapitel gehen wir auf die erreichten Projektziele und die damit verbundenen Arbeiten ein.

Ziel 1: Systemupdate auf die ursprüngliche LoRaBridge-Architektur

Zu Beginn des Projekts haben wir einige der ursprünglichen LoRaBridge Software- und Hardwarekomponenten modernisiert. Auf der Hardware-Seite haben wir die Adafruit LoRa Bonnet durch einen ESP32 LoRaWAN USB-Dongle ersetzt. Dies war notwendig, da der LoRaWAN-Software-Stack für die Bonnet bereits veraltet war und nur begrenzte Frequenzbänder unterstützte. Daher standen für ESP32-basierte LoRaWAN-Module aktuellere LoRaWAN-Stacks zur Verfügung, wie z. B. das von uns ausgewählte LMIC-Node Repository. Dies eröffnete die Unterstützung für weltweite Frequenzbänder und auch für die Zeitsynchronisation, die eine Voraussetzung für Heimautomatisierungen ist, z. B. Lichter, die durch einen zeitbasierten Schalter gesteuert werden. Das Ausführen von LoRaWAN-Datentransaktionen auf einer separaten Mikrocontroller-Einheit brachte auch einige Vorteile mit sich: Erstens wurden die potenziellen Empfangsprobleme im Zusammenhang mit der Betriebssystemlatenz auf einem Raspberry PI gelöst. Zweitens kann die Antenne der Bridge-Einheit flexibler montiert werden, z.B. mit einem USB-Verlängerungskabel, um die LoRaWAN-Link Qualität zu verbessern.

Ziel 2: Systemarchitektur



Vor den eigentlichen Entwicklungsaktivitäten haben wir einen Plan für die Systemarchitektur erstellt. Das obige Systemdiagramm veranschaulicht den ursprünglichen Plan der wichtigsten Softwarekomponenten, die auf der Bridge Raspberry Pi und auf dem lokalen LoRaWAN-Gateway bereitgestellt werden sollen. Zu den neuen Ergänzungen der LoRaBridge-Architektur gehören der Automationmanager, NodeRED (externes Framework), der Gateway-seitige Gerätemanager, der Gateway-seitige Automation-Flow-Manager und die Benutzeroberfläche. Unten findet man die Kurzbeschreibungen der Komponenten:

Automatisierungsmanager (Bridge): Python-Skript, das Konfigurationsbefehle dekomprimiert, Konfigurationen im LoRaBridge-Flo -Format speichert und Automatisierungen nach NodeRED exportiert und hochlädt.

Gerätemanager (Gateway): Python-Skript, das auf „Device Join“ -Nachrichten lauscht und jedes bei Zigbee2MQTT registrierte Zigbee-Gerät aufzeichnet. Informationen über verfügbare Geräte in der Benutzeroberfläche werden von dieser Komponente bereitgestellt.

Flow-Manager (Gateway): Python-Skript, das die LoRaBridge-Automatisierung aus benutzerdefinierten Konfigurationen über die Benutzeroberfläche zusammensetzt.

Benutzeroberfläche/LoRaMation (Gateway): Webbasierte Benutzeroberfläche, die in Svelte/Javascript geschrieben ist. Implementiert eine auf visueller Programmierung basierende Programmierumgebung zum Definieren von Automatisierungskonfigurationen.

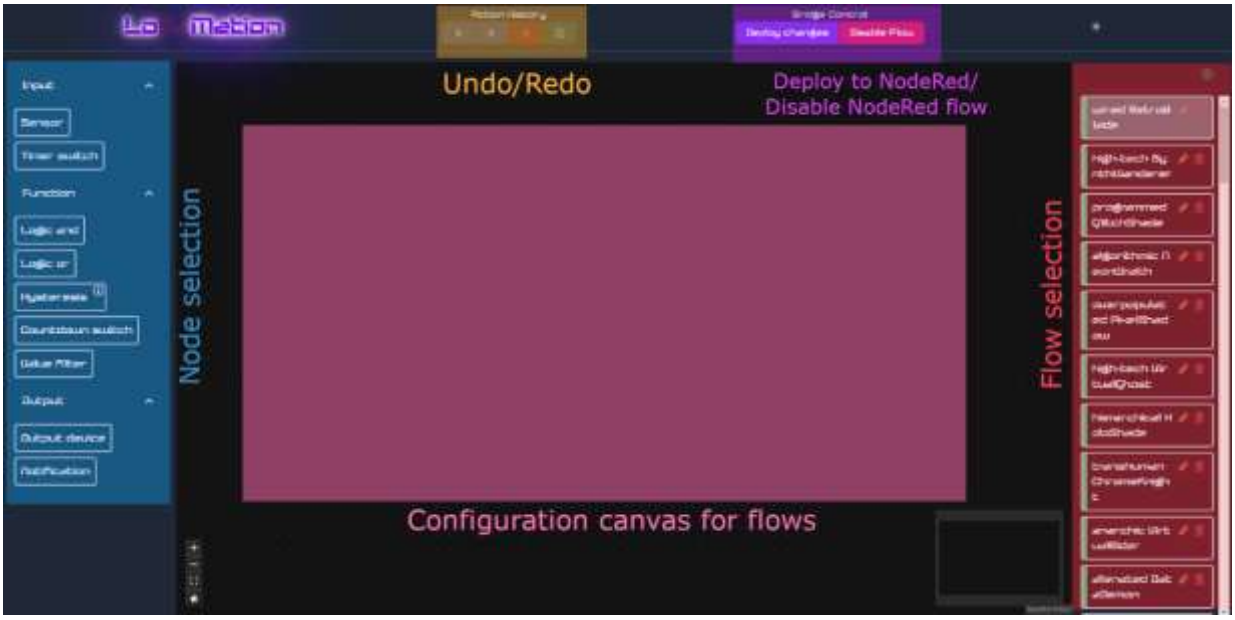
Es wurden einige neue Funktionen zu bestehenden Funktionsblöcken der ursprünglichen LoRaBridge hinzugefügt. Dazu gehören z.B. Timesync-Downlink-Nachrichten, benutzerdefinierte Ereignisse (z.B. Warnmeldungen), Systemereignisse und ein optimierter Uplink-LoRaWAN-Scheduler.

Ziel 3: Komprimierung von Automatisierungsregeln

Der zentrale Teil des LoRaBridge 2 Automatisierungskonzepts liegt in der starken Komprimierung der Konfigurationsdaten der Automatisierung. In der Regel verlassen sich die Frameworks für die Heimautomatisierung auf Standarddatenstrukturen wie JSON, um Automatisierungen zu speichern, die eine gute Flexibilität, Modularität und Lesbarkeit ermöglichen. Die daraus resultierenden Dateigrößen sind jedoch umständlich über LoRaWAN-Downlink-Nachrichten zu übertragen, da die vorgesehene Nachrichtenkapazität von wenigen Bytes deutlich kleiner ist als die Dateigröße selbst (wenige kB). Daher haben wir ein benutzerdefiniertes Komprimierungsschema geplant, das es uns ermöglicht, individuelle Automatisierungskonfigurationsbefehle mit einzelnen LoRaWAN-Downlink-Paketen zu übertragen.

Die Funktionsweise unseres Schemas ist kurz wie folgt. Auf der Bridge-Seite haben wir eine Gruppe grundlegender Automatisierungsbausteine in NodeRED definiert, z. B. Hysterese, logische Operationen, Datenquellen usw. Diese Operationen, die wir als *LoRaBride-Knoten* bezeichnen, werden als Vorlagen gespeichert, die zu tatsächlichen Automatisierungen verbunden werden können. Auf diese Weise sind die Mindestinformationen, die für die Konfiguration von Automatisierungen erforderlich sind, das Hinzufügen von Knoten, das Verbinden von Knoten und das Aktualisieren von Parametern. Um den Aufwand für die Downlink-Kommunikation auf ein absolutes Minimum zu reduzieren, haben wir die maximale Anzahl von Flüssen und Knoten auf 256 begrenzt, was der Kapazität eines einzelnen Bytes entspricht. Auf diese Weise führt z.B. der Befehl "add node" zu einer Downlink-Nachricht mit vier Bytes.

Ziel 4: Benutzeroberfläche für Automatisierungskonfiguration



Das zweite größere Ziel, das für das 2. Projekthalbjahr geplant war, war die Implementierung einer einfach zu bedienenden Benutzeroberfläche für die Programmierung der Hausautomation. Der obige Screenshot zeigt die fertige Benutzeroberfläche, die einen Benutzer*in ein visuelles Programmierkonzept bietet. Die Programmierung selbst erfordert nur wenige Schritte, die unten ausführlich erklärt sind:

Oben befindet sich eine Menüleiste, in welcher Einstellungen getroffen und Aktionen ausgeführt werden können. Der restliche Bereich gliedert sich in 3 Spalten. Die linke Spalte enthält die verfügbaren Automatisierungsknoten (grafische Element zum Aufbau des Flow-Diagramms), die per Drag & Drop in den Visualisierungsbereich gezogen werden können. Die mittlere Spalte dient der Visualisierung des Flows-Diagramms. Hier können die zuvor hinzugefügten Knoten verbunden werden, um den Ablauf der Automatisierung festzulegen. Weiters können innerhalb der Knoten die Eigenschaften des jeweiligen Knotens beziehungsweise des Automatisierungselement über verschiedene Eingabefelder konfiguriert werden. Die gerätebezogenen Knoten “Sensor” und “Output Device” stellen zudem eine Liste der verfügbaren Geräte sowie die per Gerät verfügbaren Sensorattribute zur Verfügung.

Die rechte Spalte dient der Verwaltung der einzelnen Flows. Flows können in diesem Kontext vereinfacht, wie Dateien betrachtet werden. Dementsprechend können neue Flows angelegt und bestehende Flows umbenannt und gelöscht werden. Ein Klick auf einen bestimmten Flow in der rechten Spalte lädt diesen und stellt dessen Darstellung in der mittleren Spalte dar.

Änderungen des Flow-Diagramms werden automatisch gespeichert und können über die Menüleiste rückgängig gemacht oder wiederhergestellt werden. Weiters kann der Flow ebenfalls per Menüleiste auf die Bridge übertragen und aktiviert beziehungsweise deaktiviert werden.

Um das Rückgängigmachen von Änderungen zu ermöglichen, werden die einzelnen Versionen eines Flows in einer CouchDB Datenbank gespeichert, welche sich lokal im Browser befindet. Zusätzlich werden die Änderungen mit einer CouchDB Datenbank auf dem Gateway synchronisiert. Dadurch können Flows auch offline geändert und später hochgeladen werden, beziehungsweise gehen keine Daten bei dem Verlust der Internetverbindung verloren.

Die Schaltfläche “Status” öffnet einen eigenen Status-Bereich, der Nachrichten von der Bridge wie beispielsweise kritische Fehlermeldungen enthält.

Die aktuelle Auswahl an Knoten unterstützt die Konfiguration größerer Automatisierungen (d.h. Setups, die aus mehreren Sensoren und Aktoren bestehen). Beispiele für solche Konfigurationen sind die Einbrucherkennung mit mehreren Bewegungsmeldern und Tür-/Fenstersensoren, die Regelung von Temperatur/Luftfeuchtigkeit einzelner Räume eines Gebäudes oder die Steuerung von Lichtgruppen mit Bewegungsmeldern.

Ziel 5: Testen mit realen Automatisierungskonfigurationen

Nachdem wir die Funktionalität der Softwarekomponenten erfolgreich validiert hatten, haben wir die Programmierung von Automatisierungen in zwei Szenarien getestet, die nahe an realistischen Anwendungsfällen liegen. Als Test Use-Cases wurden sowohl Indoor- als auch Outdoor-Szenarien

ausgewählt. Ersteres stellt einen Use-Case in einem Mehrfamilienhaus dar, in dem Mieter Zigbee-Geräte installiert haben, z. B. in einem Keller, und letzteres stellt einen Schutzanwendungsfall dar, bei dem die Entfernung zwischen dem Haus des Gartenbesitzers und dem Garten selbst für eine Zigbee-Verbindung unerreichbar ist. In jedem Szenario konfigurierten wir eine einfache, auf Bewegungserkennung basierende Lichtsteuerung für die Bridge Einheit. Der Hauptzweck dieser Tests bestand darin, die Leistung der Übertragung der Automatisierungskonfiguration über einen LoRaWAN-Downlink zu messen. Denn nach unseren Ergebnissen können solche einfache Automatisierungen innerhalb weniger Minuten eingesetzt werden. Dies lag über unseren Erwartungen, und daher gehen wir optimistisch davon aus, dass auch komplexere Automatisierungen in angemessener Zeit bereitgestellt werden können.

3 Verlauf der Arbeitspakete

3.1 Arbeitspaket 1 – Detailplanung und Formales am Projektstart

Das Projekt wurde mit der Erstellung eines detaillierten Zeit- und Arbeitsplans, der pünktlich an Netidee geliefert wurde, angefangen. Der Plan besteht kurz gefasst aus drei Hauptteilen: 1) Planung und Forschung (AP2, AP3), 2) Prototyp-Implementierung (AP4, AP6) und 3) Validierung (AP5). In der ersten Phase haben wir uns damit beschäftigt, die beste, verfügbare Software und Hardware auszuwählen. Darüber hinaus werden wir vor dem Prototyping die Konfiguration der Heimautomatisierung über LoRaWAN festlegen, die Definitionen für unterstützte Automatisierungsgeräte und -algorithmen, Komprimierungsregeln, das LoRaWAN-Downlink-Protokoll für die Datenübertragung sowie die verwendeten Datenstrukturen enthält. Die eigentliche Implementierungsarbeit wurde in der zweiten Phase stattgefunden, die hauptsächlich aus der Software-/Firmware-Entwicklung bestand. Die dritte Phase umfasste die Validierung der Konfigurationsprogrammierung für die Hausautomation über LoRaWAN. Der Fokus lag hier darauf, sicherzustellen, dass keine fehlerhaften Konfigurationen z.B. aufgrund fehlender LoRaWAN-Pakete im Home Automation Gateway landen.

Gleich zu Beginn des Projekts haben wir eine GitHub-Organisation für das Projekt zur Versionskontrolle und für das Issue-Tracking vorbereitet. Für die Aufgabenverfolgung/das Brainstorming verwendeten wir weiterhin das Trello-Board, was sich bereits während des ursprünglichen LoRaBridge-Projekts als vorteilhaft erwiesen hat. Zu Beginn des Projekts wurde auch der erste Netidee-Blogbeitrag erstellt.

3.2 Arbeitspaket 2 – Recherche Software/Hardware/Schnittstellen

Da unser Long-Range Automatisierungskonzept 3rd-Party-Frameworks für z.B. die ZigBee Schnittstelle und LoRaWAN-Anbindung erfordert, war es wichtig, die neuesten verfügbaren open source Optionen zu nutzen. Bei der Auswahl der geeigneten Hardware und Software waren unsere Hauptkriterien zum einen die Verfügbarkeit (z. B. Unterstützung für mehrere Funkmodule) und zum anderen die aktive Wartung/Community-Support.

Hardwareseitig haben wir weiterhin die Raspberry PI 4B+ Plattform verwendet, da sie ein gutes Gleichgewicht zwischen verfügbaren Rechenressourcen, Energieverbrauch und Anschaffungspreis bietet. Daher haben wir das Raspberry PI LoRa-Modul von LoRaBridge 1 durch ein ESP32 LoRaWAN Entwicklungsboard ersetzt. Wie es sich herausstellte, ist die LoRaWAN-Stack-Unterstützung für das Erstgenannte etwas veraltet, was unseren Fokus auf alternative LoRaWAN Bibliotheken lenkte. Diese Designentscheidung bringt einige optionale Vorteile mit sich, wie z. B. die Unabhängigkeit des Raspberry PI als Automatisierungsgateway. Zweitens könnte man die WLAN-Schnittstelle des ESP32 benutzen, um Daten vom Raspberry PI zum LoRaWAN-Modul zu leiten, um die LoRaWAN-Antenne flexibler zu platzieren. Dieses optionale Designziel kann die LoRaWAN-Reichweite in Anwendungsfällen verbessern, in denen die LoRaWAN-Verbindung an ihre Grenzen stößt. Der Rest der Hardware ähnelt jener, die im ursprünglichen LoRaBridge-Projekt verwendet wurde. Das Raspberry PI LoRaWAN-Gateway-Modul und der TI CC2652-basierte ZigBee-Wireless-Adapter werden in LoRaBridge 2 weiter zum Einsatz kommen.

Auf der Softwareseite wurde die Unterstützung von LoRaWAN-Modulen auf das LMIC-Node-Repository aktualisiert, bei dem es sich um einen LMIC-basierten LoRaWAN-Stack handelt, der auf die ESP32-Mikrocontrollerfamilie ausgerichtet ist. Somit kann die bereits implementierte Automatisierungskonfiguration über LoRaWAN mit über 10 verschiedenen LoRaWAN-Entwicklungsmodulen realisiert werden. Das LMIC-Node-Repository bietet auch Unterstützung für andere Frequenzbänder, z. B. in den USA oder Indien. Da das Repository den LoRaWAN-Standard 1.0.3 implementiert, können die Netzwerk-Timesync Anfragen verwendet werden, um die Zeit des Heimautomatisierungs-Gateways über LoRaWAN zu aktualisieren. Als Automatisierungsframework haben wir das beliebte Node-RED-Framework gewählt, das auf einer visuellen Flow-basierten Programmierung basiert. Als eines der beliebtesten open source Tools zur Implementierung von Automatisierungen mit Unterstützung für das Self-Hosting auf einem Raspberry PI, ist es eine gute Basis für das LoRaBridge Automation Gateway.

3.3 Arbeitspaket 3 - Automatisierungskonzept

Das Hauptziel des AP3 war es, ein Konzept für die Heimautomatisierung zu entwickeln, das die Einrichtung und Konfiguration über einen LoRaWAN-Downlink ermöglicht. Schließlich haben wir ein vereinfachtes Abstraktionsmodell entwickelt, das aus verschiedenen Aktionen, Knoten (Nodes) und Verbindungen zwischen Knoten besteht. Da es in unserem Modell nur eine begrenzte Anzahl an möglichen Geräten und Automatisierungen gibt, konnten wir die Informationen, die zur Darstellung

von z.B. Knoten/Aktionen benötigt werden, auf ein absolutes Minimum reduzieren. Um eine Vorstellung davon zu bekommen, wie diese Komprimierung funktioniert, benötigt in unserem Modell beispielsweise eine Aktion zum Hinzufügen einer Automatisierungsregel nur vier Bytes. Im Vergleich dazu würde eine äquivalente Node-RED-Aktion mit JSON-Objekten einige hundert Bytes benötigen. Daraus ergibt sich ein Kompressionsverhältnis von ca. 300, was eine Automatisierungskonfiguration über LoRaWAN ermöglicht.

Die folgenden Aktionen wurden umgesetzt:

- *Knoten hinzufügen*
- *Knoten entfernen*
- *Gerät hinzufügen*
- *Aktualisierung der Parameter*
- *Knoten verbinden*
- *Knoten trennen*
- *Hinzufügen eines Flows**
- *Flow aktivieren*
- *Flow deaktivieren*
- *Flow entfernen*
- ***Flow in Nodered hochladen***
- ***Zeit-Synchronisierung Anfrage***
- ***Geräte Anfrage***
- Ablauf abgeschlossen

*(*Ein Flow ist eine Sammlung von mehreren zusammenverbundenen Knoten)*

Die folgenden Knoten sind unterstützt:

- Binäres Gerät (z. B. ein Smart Plug)
- Binärer Sensor (z.B. ein Bewegungssensor)
- Numerischer Sensor (z. B. ein Temperatursensor)
- Logisches UND
- Logisches ODER
- Zeitschaltuhr
- Hysterese
- Countdown-Schalter
- Filterung der Werte
- Benutzernachricht
- (Benutzerdefinierter Knoten)

Die Automatisierungsknoten, wie z.B. die Zeitschaltuhr, wurden in JavaScript implementiert, wie es in Node-RED unterstützt wird. Der Workflow zur Implementierung von Automatisierungsknoten beginnt mit der Definition einer Node-RED JSON-Vorlage. Anschließend sollen die Knoteneingänge, -ausgänge sowie konfigurierbare Parameter in das Template eingefügt werden. Mit einem solchen Workflow können Benutzer*innen benutzerdefinierte Automatisierungsknoten implementieren. Die oben genannten Knoten und Aktionen sollen die Anforderungen einfacher Use-Cases erfüllen, auf die LoRaBridge 2 abzielt.

3.4 Arbeitspaket 4 - Proof-of-Concept Prototyp

Der erste Prototyp des in AP3 konzipierten Hausautomationskonzepts wurde erfolgreich umgesetzt. Die zentralen Komponenten, die wir verwendet haben, waren Node-RED, ChirpStack (von LoRaBridge 1), die LoRa-Node-basierte Firmware für das LoRaWAN-Modul und die in Python geschriebene Automatisierungsmanager-Software. Die intensivste Entwicklungsphase umfasste die Programmierung des Automatisierungsmanagers und der Firmware für die LoRaWAN-Downlink-Kommunikation. Im Folgenden finden Sie einen kurzen Überblick über die implementierten Softwarekomponenten.

Automatisierungsmanager: Die komprimierten Automatisierungsbefehle werden aus Redis Datenbank bezogen und anschließend geparst. Der Manager prüft dann auf fehlerhafte/nicht unterstützte Befehle/Aktionen und führt die gewünschte Funktion aus. Sobald eine Automatisierungseinrichtung abgeschlossen ist, führt der Manager die Dekomprimierung der definierten Aktionen/Knoten in einen Node-RED-Flow durch. Schnittstellen zu Zigbee2MQTT und zu Node-RED werden über MQTT bzw. REST realisiert. Die MQTT-Schnittstelle wird verwendet, um ZigBee-Geräte von Benutzer*innen den Geräteindizes des Automatisierungsmanagers zuzuordnen. Die Node-RED REST-Schnittstelle wird verwendet, um den generierten Node-RED-Flow auf Node-RED hochzuladen.

LoRaWAN-Modul: Die Firmware für das ESP32-basierte LoRaWAN-Modul übernimmt zwei Aufgaben. Zunächst werden die Sensor-/Gerätedaten vom Bridge-Gerätemanager über eine serielle USB-Schnittstelle abgerufen und über eine LoRaWAN-Verbindung zu ChirpStack geschickt. Zweitens werden komprimierte Automatisierungskonfigurationspakete über LoRaWAN-Downlink (als Teil von MAC-Befehlen und ACK-Paketen) an das Modul übertragen. Falls keine Gerätedaten für die Uplink-Übertragung vorgesehen sind, sendet die Firmware ein "Heartbeat"-Paket, damit eine periodische Downlink-Kommunikation stattfinden kann.

Gerätemanager: Diese Komponente fordert von der Bridge die verfügbaren Geräte an. Die Antworten werden von der Konverter-Komponente geparst und per MQTT weitergeleitet. Der Gerätemanager empfängt diese MQTT-Nachrichten sowie Sensorwerte und speichert Gerätename und Attribute (verfügbare Sensoreigenschaften wie etwa Temperatur) in der Redis Datenbank. Zusätzlich werden die Gerätedaten per MQTT zur Verwendung durch Plug-ins (Homeassistant Integration) weitergeleitet.

3.5 Arbeitspaket 5 - Validierung und Tests

Die erste Validierungstests wurden für den Automation Manager vorbereitet, der viele zu verifizierende Funktionen in sich trägt. Zunächst wurden Testskripte geschrieben, die Automatisierungskonfigurationsbefehle direkt in die Redis-Datenbank der Bridge Einheit hochlädt. Auf diese Weise wurde die Fehlersuche/-behebung beschleunigt, da die Latenz durch das LoRaWAN vermieden werden konnte. Sobald wir die minimal funktionsfähige Version des Automatisierungsmanagers erreicht hatten, wurde ein einfaches LoRaWAN-Downlink-Scheduler-Skript geschrieben, um die Automatisierungskonfiguration über LoRaWAN zu testen.

Mit der Planung von Automatisierungsbefehlen haben wir die Konfiguration von zwei einfachen Automatisierungen getestet: Lichtsteuerung per Zeitschaltuhr und bewegungssensorgesteuerte Leuchten. Den Validierungsergebnissen zufolge können einfache Automatisierungen über LoRaWAN in etwa 1-60 Minuten konfiguriert werden, abhängig von der Qualität der LoRaWAN-Verbindung. Alles in allem bestätigten die positiven Ergebnisse die Funktionalität unseres Systems, was definitiv ein Highlight des ersten Projekthalbjahres war.

In der zweiten Hälfte des Projektjahres haben wir weitere Validierungsarbeiten durchgeführt. Erstens, als weitere Automatisierungsfunktionen hinzugefügt wurden, haben wir die Funktionalität kontinuierlich auf mögliche Fehler und falsches Code-Verhalten getestet. Zweitens konnten wir mit den neuen Automatisierungsfunktionen weitere Automatisierungen validieren. Beispiele hierfür sind die Temperaturregelung mit Hysteresefunktion (z.B. Beheizung eines Raumes im Winter) und Benutzerbenachrichtigungen (Kellerfeuchte nach/während der Hochwasserperiode in Niederösterreich). Darüber hinaus wurde ein mobiler Prüfstand vorbereitet, der es uns ermöglichte, die Brückeneinheit an Orten einzusetzen, die realitätsnahe Anwendungsfälle darstellen, wie z.B. im Keller oder im Vorgarten unserer Universität.

3.6 Arbeitspaket 6 - Benutzeroberfläche

Im Zuge dieses Arbeitspakets wurde eine Weboberfläche zur Erstellung von Automatisierungsflows auf dem Gateway sowie die notwendigen Integrationen mit den übrigen Komponenten entwickelt. Die Weboberfläche basiert auf dem Web Application Framework SvelteKit in Kombination mit dem UI Component Framework Svelte. Die Oberfläche nutzt für die einzelnen UI-Komponenten (wie beispielsweise gestylte Buttons) Flowbite Svelte, während zur Visualisierung der Flows Svelte Flow eingesetzt wird. Einen besonderen Fokus haben wir auf die Usability der Benutzeroberfläche gelegt, um die Erstellung der Automatisierungen möglichst intuitiv zu gestalten. Hierfür wurde bereits bei der Aufteilung der Oberfläche auf bekannte, häufig verwendete Strukturen aufgebaut.

Um die Weboberfläche in das System zu integrieren, wurde der Flowmanager als eigenständige Komponente geschaffen. Der Flowmanager lädt Flows aus der Redis Datenbank, welche von der Weboberfläche beim Deployen der Flows dort gespeichert werden, vergleicht neue Versionen gegebenenfalls mit der vorherigen Version und erstellt die notwendigen komprimierten Aktionsbefehle. Diese Befehle werden an die Bridge übertragen und dort von dem Automatisierungsmanager in den eigentlichen Node-RED Flow umgewandelt. Nach der Übertragung wird die Korrektheit des übermittelten Flows auf beiden Seiten via Prüfsummenvergleichs geprüft.

3.7 Arbeitspaket 7 - Projektmanagement und Kommunikation

Da sich unser Entwicklungsteam gegenüber dem Basisprojekt LoRaBridge nicht verändert hat, mussten wir unseren Arbeitsablauf nicht anpassen, so dass die Arbeitszeit für das Management wie ursprünglich geplant genutzt werden konnte. Während des gesamten Projekts hielten wir ein bis zwei wöchentliche Meetings ab, in denen wir die technischen Probleme diskutierten und den Fortschritt einzelner Arbeitspakete bewerteten. Die Kommunikation mit Netidee wurde per E-Mail hergestellt, was sehr gut funktionierte.

Laut unserem ursprünglichen Projektplan streben wir eine Open-Access Publikation an, um insbesondere unsere Kompressionsmethoden zu verbreiten. Dieses Ziel ist nach wie vor gültig, und wir gehen davon aus, dass die Ergebnisse im Jahr 2025 veröffentlicht werden. Um die Verbreitung zu beschleunigen, planen wir die Veröffentlichung eines Preprint-Manuskripts im Open Access, z.B. in ArXiv. Der Grund, warum dies im Jahr 2024 nicht stattgefunden hat, liegt darin, dass wir mehr Stunden in AP4-AP6 verbraucht haben, als wir erwartet hatten. Auch der Veröffentlichungsprozess dauert in der Regel bis zu mehreren Monaten. Sobald die Ergebnisse veröffentlicht sind, werden wir sie auf die Netidee-Projektseite sowie auf die Dokumentation verlinken.

3.8 Arbeitspaket 8 – Dokumentation und formales Projektende

In diesem Arbeitspaket haben wir Benutzer*in- und Entwicklerdokumentation, Endbericht, Zusammenfassung und den Letzter Blog-Beitrag erfasst.

4 Umsetzung Förderauflagen

Für das Projekt LoRaBridge 2 wurden keine besonderen Anforderungen oder Vereinbarungen zur Finanzierung getroffen.

5 Liste Projektendergebnisse

Kurzbeschreibung der erreichten Projektendergebnisse jeweils mit Open Source Lizenz und Webadresse (netidee Vorgaben beachten!)

1	*Projektzwischenbericht*	*CC BY 4.0*	*https://www.netidee.at/lorabridge-2*
2	*Projektendbericht*	*CC BY 4.0*	*https://www.netidee.at/lorabridge-2*
3	*Entwickler_innen-DOKUMENTATION*	*CC BY 4.0*	*https://lorabridge2.github.io/*
4	*Anwender_innen-DOKUMENTATION*	*CC BY 4.0*	*https://lorabridge2.github.io/*
5	*Veröffentlichungsfähiger Einseiter / Zusammenfassung*	*CC BY 4.0*	*https://www.netidee.at/lorabridge-2*
6	*Dokumentation Externkommunikation zur Erreichung Sichtbarkeit /Nachhaltigkeit (als Teil des Endberichtes)*	*CC BY 4.0*	*https://www.netidee.at/lorabridge-2*
7	*Software client (LoRaMation Benutzeroberfläche)*	*GPL-3.0*	*https://github.com/lorabridge2*
8	*Software Modul (Home Automation Manager & LoRaWAN Schnittstelle)*	*GPL-3.0*	*https://github.com/lorabridge2*

6 Verwertung der Projektergebnisse in der Praxis

Da unser Fokus während des Projekts darauf lag, die Machbarkeit der Konfiguration der Homeautomation über LoRaWAN zu demonstrieren (was nach unserem besten Wissen noch nie zuvor geschehen ist), wird die praktische Verwertung der Ergebnisse nach dem Projektzeitraum schrittweise erfolgen. Es ist auch wichtig zu beachten, dass das LoRaBridge-Framework auf dem technologischen Reifegrad ca. auf den Stufen 6-7 (Demonstration von Systemprototypen in Betriebsumgebungen) liegt. Daher freuen wir uns darauf, als nächsten Schritt den Quellcode und unsere Ergebnisse an Open-Source-Heimautomatisierungs-Communities weiterzugeben, in der Hoffnung, nützliches Feedback über das Systemverhalten und mögliche Fehler zu sammeln. Detaillierte Pläne zur Verbreitung finden Sie im nächsten Abschnitt.

7 Öffentlichkeitsarbeit/ Vernetzung

Während des Projekts wurden folgende Verbreitungsaktivitäten etabliert.

- Sieben netidee Blogbeiträge (https://www.netidee.at/lorabridge2)
- Social-Media-Beiträge (Hackrnews und Home Assistant Forum)
- European Researchers Night 2024 in Graz (Live-Demo)
- WWW-Projektseite wurde eingerichtet (https://research.fhstp.ac.at/projekte/lorabridge-2)

In Zukunft wollen wir unsere Zielgruppen (Nutzer und Entwickler von Heimwerken) erreichen durch:

- Weitere Social-Media-Präsenzen in Hackrnews, passenden Subreddits und Foren (Home Assistant und Zigbee2MQTT)
- Konferenzpublikation/Workshop in geeigneter Location
- Pressemitteilung über die PR-Abteilung der FH St. Pölten

8 Eigene Projektwebsite

Zu diesem Zeitpunkt gibt es keine zusätzliche Projekt-WWW-Site.

9 Geplante Aktivitäten nach netidee-Projektende

Da unsere Kompressionstechniken weiterentwickelt werden können, um sie an die Anforderungen anderer Anwendungen anzupassen, wie z.B. die Aktualisierung/Steuerung von Edge-Computing-Einheiten, werden wir wahrscheinlich Forschungsförderung zu diesem Thema beantragen.

Die kleinen Folgeaktivitäten zur Verbesserung der Projektqualität (z. B. Fehlerbehebung, Code-Reviews, Reaktion auf Benutzerfeedback und Aktualisierung von Softwarekomponenten) werden auf freiwilliger Basis durchgeführt. Für größere Entwicklungsaktivitäten, wie z. B. eine neue benutzerdefinierte Integration oder neue Automatisierungsknoten, könnten Open-Source-Fonds wie Netidee/NLNet finanziert werden.

Das LoRaBridge-Framework soll auch als Teil der studentischen Projekte integriert werden, die das praktische Erlernen der Programmierung von Hausautomationsprogrammen beinhalten.

10 Anregungen für Weiterentwicklungen durch Dritte

Da es sich bei der Software, die während LoRaBridge 2 entwickelt wurde, um einen Proof-of-Concept-Prototyp handelt, sind weitere Tests unter realen Bedingungen erforderlich, um mögliche Fehler/Fehlverhalten zu finden. Darüber hinaus ist die Erweiterung der Automatisierungsfunktionen auch ein möglicher Weg für die Weiterentwicklung. Beispiele für solche Aktivitäten sind:

- Konfiguration komplexer Automatisierungen mit LoRaMation
- Testen der LoRaBridge-Automatisierungsknoten mit verschiedenen Anwendungsfällen
- Entwicklung weiterer Automatisierungsknoten
- Integration weiterer Geräteschnittstellen über MQTT, z.B. ESPHome