IoT Interoperability: Edge-based IoT Platform

Author: Thomas Mead

Aims and Objectives

Emedded Systems and IoT

The IoT has emerged as the next technological revolution through advancements in electronic and wireless communication technologies. By 2025, the International Data Corporation has forecast a total of 41.6 billion connected IoT devices generating 79.4 zettabytes of data. The connectivity of these embedded devices will provide diverse applications impacting all domains of modern society.

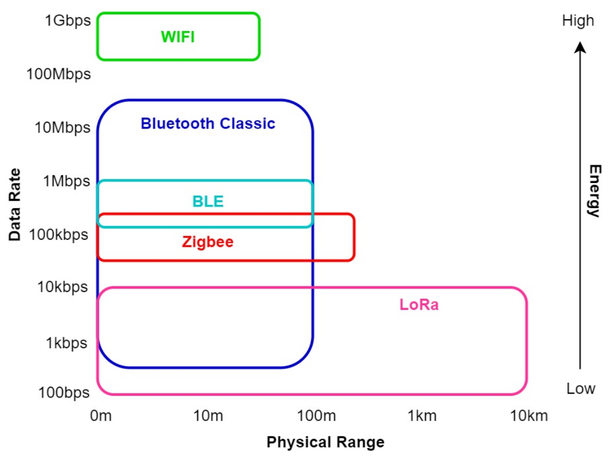
Many of these embedded IoT devices do not have native Internet capabilities and communicate over non-IP protocols. Therefore, these IoT devices require a communication gateway to bridge and enable Internet capabilities.

Interoperability

Since 1995, the International Telecommunication Union has recognised interoperability as the main issue limiting the success of the IoT. Within this technology, interoperability is defined as the ability of two systems to communicate and share services with each other. Many different areas of interoperability have emerged.

Technical interoperability is a dimension that focuses on both the physical and software compatibility of the devices; this includes the communicating protocols implemented. Wireless embedded devices are selected for a system based on their functionality. These devices are limited by factors including power energy consumption, communication bandwidth requirements, computation and security capabilities. The different requirements for each device demonstrate the importance of technical interoperability. For example, it is poor design to implement a high-power communication protocol for a resource-constained device as this would massively reduce the intended performance.

Most existing IoT vendors appear to neglect this issue by implementing a single protocol regardless of the embedded device requirements. These niche systems are isolated and unable to interact. This then forms a plethora of independent IoT systems which often require their own gateway. These systems result in the installation of a range of gateways for different applications.



Edge Computing

Another significant issue limiting the performance of the IoT is the current implementation of cloud computing integration. The high computing power of cloud services has been efficiently utilised for data management within this technology. However, with the increasing quantity of data generated from IoT devices, the bandwidth of a network is becoming strained. The speed of data transportation is becoming the bottleneck for the cloud-based computing paradigm. Often, the data transfer latency is too high and unable to fulfil the requirements of the IoT system. IoT-generated data must be stored, processed, analysed and acted upon at the edge of the network to reduce this issue.

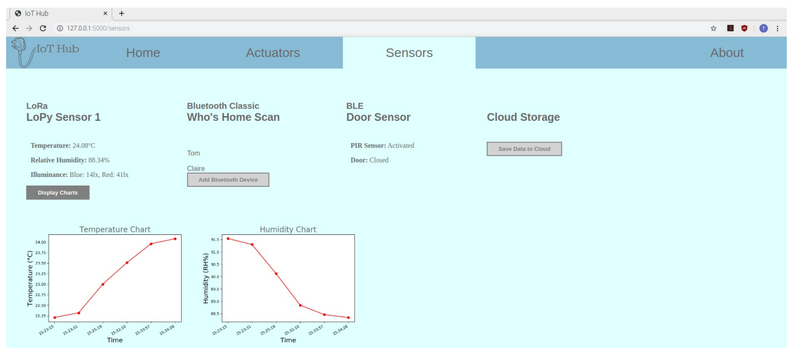
Project Aim

This project addresses these challenges and attempts to provide a solution by developing an edge-based IoT gateway hub. The design is a single gateway interface for a variety of heterogeneous IoT devices, which bridges between disparate communication protocols. All the connected devices implement home-automation functionality demonstrating a smart-home system. However, the primary aim of this project is to develop an edge-based solution to IoT technical interoperability which is easily adaptable to other useful applications. Hub-to-hub interoperability is also made possible through cloud integration within this project. As the number of connected IoT devices continue to rapidly expand, technical interoperability is critical for the scalability of this technology.

Design

IoT Hub

The edge-based IoT gateway is a centralised hub to a star network topology. This hub communicates over a range of protocols to a set of heterogeneous devices. All these devices implement their preferred protocol and provide home-automation functionality. The IoT hub design is developed on a Raspberry Pi 3B+ and written in Python and C. This performs communication interfacing, data management, cloud delivery and hosts a web server that runs a web interface. Through the web interface, the user can view the status of all the connected devices, control the actuators and read the sensor data.



Sensors Page of Graphical User Interface

Connected Embedded Devices

LoPy4 Multi-Sensor

The LoPy4 is programmed in MicroPython to communicate through LoRa. This device transmits the temperature, humidity and ambient light values from a interfaced Pysense multi-sensor board. When the device is not transmitting, it is configured to deep-sleep mode to heavily reduce power consumption.

NodeMCU Thermostat

The NodeMCU simulates a wireless thermostat through an output LED to indicate if the thermostat is on or off. It is programmed in Arduino code and contains an ESP8266 WiFi module to communicate to the IoT Hub. The hub uses the most recent temperature readings from the LoPy4 to determine the on-off state of the thermostat.

Tradfri Devices

Through CoAP over WiFi, commands are sent from the hub to the Tradfri Gateway to enable control of the connected Tradfri devices. These devices consisted of a smart, dimmable light bulb and a wall plug. The Zigbee protocol is implemented for Tradfri device-gateway connectivity.

Door Sensor

The door sensor sub-system is a Raspberry Pi 3B+ programmed in Python which communicates through Bluetooth Low-Energy (BLE). The Pi is integrated with a PIR sensor and a reed switch to provide a proximity sensor and a open-closed state identifier for the door.

Who's Home Scan

The system determines who is home through a Bluetooth classic scan. This scan identifies the resident's mobile phones when they are within the 30m Bluetooth proximity of the hub. If the device is found, the system assumes that specific resident is home. Mobile phones are suitable for this function as they use Bluetooth requires minimal power and most people do not leave their house without them.

Application

The application, written in Python, generates a graphical user interface for the system. This interface displays system information, provides actuator control, and enables access to cloud storage used for sensor data. The IoT Hub hosts the web server via a Flask application. This application combines HTML, CSS and JavaScript files, providing the page template form, style and additional functionality, respectively.

The Google Cloud Platform was implemented for external storage, which saves the LoPy4 sensor data to the user's personal Google Drive. This data is also stored locally on the hub to reduce data access speeds from the cloud.

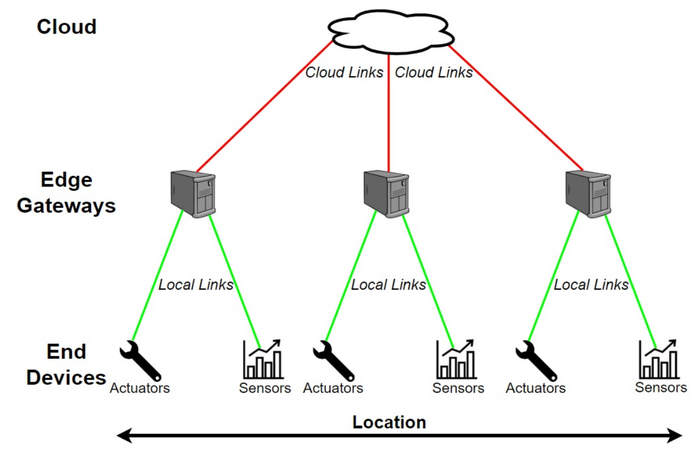
Results

Impact and Outcome

The edge-based IoT Hub is a solution to technical interoperability by providing a communication gateway for a range of standardised protocols. The system establishes seamless operation and cooperation of heterogeneous devices. It is Compatible with the communication protocols: LoRa, Bluetooth Classic, BLE, WiFi and Zigbee; and the application layer protocols: CoAP, HTTP and MQTT. The IoT Hub provides a bridge between the different device standards, data management and specifications. This is achieved through the protocol conversion interface for the connected devices, resulting in a platform through which any embedded device can communicate. Devices then implement their preferred protocol based on their resource availability. This then communicates through the hub to another device implementing a different protocol.

Introducing the edge computing paradigm massively reduces the computing load on the cloud. Tasks are completed at the edge of the network which reduces unnecessary bandwidth and computing resource usage. This then increases the system performance, speed and scalability. Through the cloud integration, hub-to-hub communication is also made possible and forms a potential infrastructure of IoT Hubs. Each hub has their own, independently defined local network with communication capabilities tailored to their connected devices and specification. The local hub network is tailored to their requirements, implementing functionalities for any given service. Although it has potential to be used for many different applications, the completed project successfully demonstrates the configuration of a smart home.

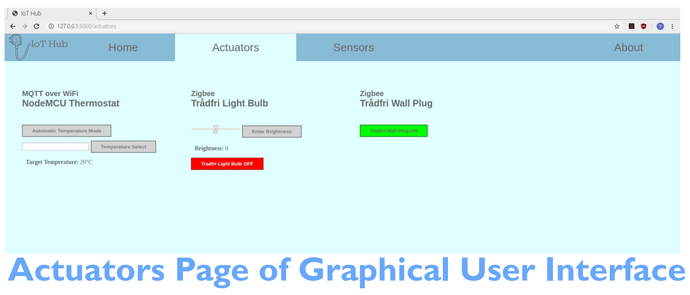
The functional structure of the Python application makes it simple for application developers and device integrators to build on the current system. By doing this, they can implement additional functionality to suit their system requirements. The addition of new modules can be achieved simply by correctly routing a new function to the Python application and assigning proper user interface functionality through the web development files.



Future Work

To further the development of technical interoperability, the introduction of new communication protocol capabilities is a potential area of future design. Also, another benefical feature to consider is the possible development of a second IoT Hub. This hub would be placed on a different local network to establish hub-to-hub communication through the cloud. This development would demonstrate the basis of the IoT Hub infrastructure.

Another factor limiting the scalability and deployment of this system is security and privacy. These issues did not display a primary concern with the development of this project. However, going forward, it will be a crucial area of design. The communication protocols integrated within the system support a secure, native, end-to-end encryption. Improvements to the overall system security are required to ensure there are no vulnerabilities that could be exploited.



Actuators Page of Graphical User Interface

Network Architecture