Pervasive Computing Implementation——smart-pi-on-edge

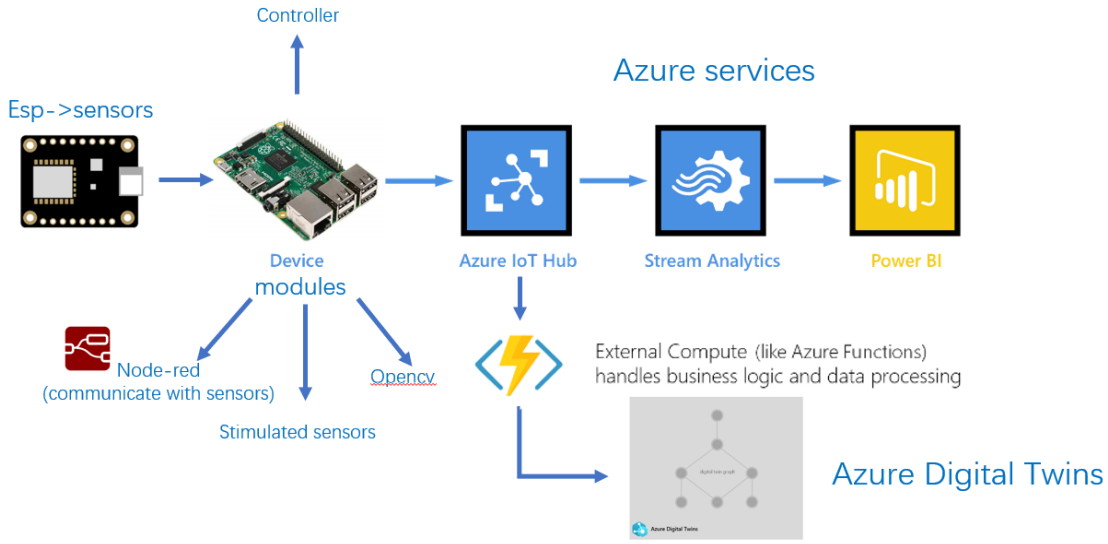
Introduction

This project aims to explore the implementation of Pervasive Computing in the context of Internet of Things (IoT). Pervasive Computing refers to the integration of smart devices and technologies into our everyday surroundings, enabling seamless connectivity and intelligent interactions. By combining the power of ESP8266 with sensors, Raspberry Pi as edge device with multiple modules, and Azure Cloud services, this project creates a scalable IoT system.

Project Overview

The project revolves around building a IoT system that utilizes various components to enable pervasive connectivity and intelligent control. The key components of the project include ESP8266, Raspberry Pi with 4 modules and Azure Cloud services, including Azure Digital Twins.

The project aligns with course in follow aspects:
Edge computing discussed in 1.1
Cloud computing discussed in 1.4 (SaaS, PaaS, FaaS)
Digital Twins discussed in 1.5 (DT, SDK, DTDL)
Flow based programming mentioned in Lab 02



Project structure

Project Problem Analysis

The project aims to provide smart home services as a service provider for customers. In this scenario, we face several challenges:

- IoT control latency: Due to the user's expectation of real-time response, we need to address the issue of IoT control latency, ensuring that user commands are promptly communicated and executed.

- Scalability and flexibility of the system: As the number of users and devices increases, we require a scalable system architecture that can accommodate more users and devices while being flexible enough to adapt to different home devices and vendors.

- Data processing and decision-making: To deliver an intelligent home control experience, we need to process and analyze a large volume of sensor data and make decisions and actions based on user settings and preferences.

For solving challenges, several solution were implemented:

Edge computing: By deploying processing and decision-making capabilities on edge devices, we can reduce communication latency with the cloud platform, improving response speed.

IoT Gateway: Node-RED and Raspberry Pi together act as an IoT gateway, enabling communication and integration between the cloud platform and the IoT devices.

Azure Digital Twins: By using Azure Digital Twins, we can model and represent the physical environment, devices, and their relationships in a digital form. This allows us to gain a holistic view of the smart home ecosystem and easily manage and control the devices. (Virtural LED for example)

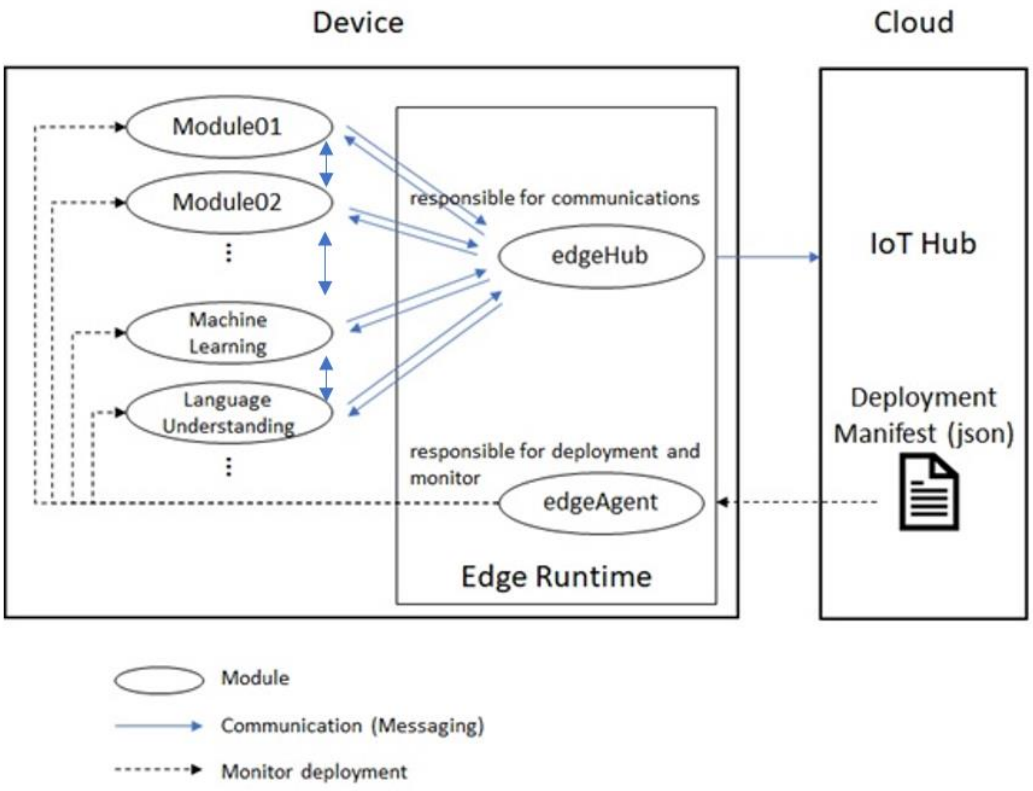
PaaS and SaaS: Leveraging cloud platform-provided managed services and software components, we can rapidly build and deploy smart home services, simplifying the development and maintenance process. Also, decisions and data processing are implemented on those platform and services.

Project Design

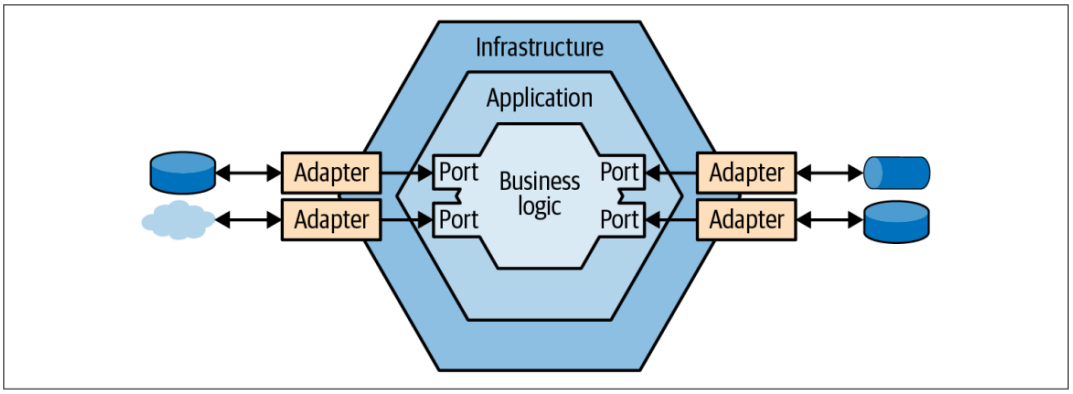
In Azure IoT Edge, each module can be thought of as an independent node that receives input data, processes it, and passes it on to the next module. This flow of data between modules reflects the flow of information processing.

Flow-based programming is a programming paradigm that defines applications as networks of black box processes. FBP is thus naturally component-oriented.

Flow-based programming emphasizes composability and reusability between modules. In Azure IoT Edge, it enable us to build a complete processing flow by combining different modules, each of which is responsible for a different function or processing task. This modular design makes application development and maintenance more flexible and scalable. In my case, the data flow between modules is in JSON format. Each module receives the JSON data, performs its designated processing or function, and passes the data to the next module in the flow. Finally, in the control module, the data from various modules is integrated and processed before being sent to the IoT Hub endpoint.



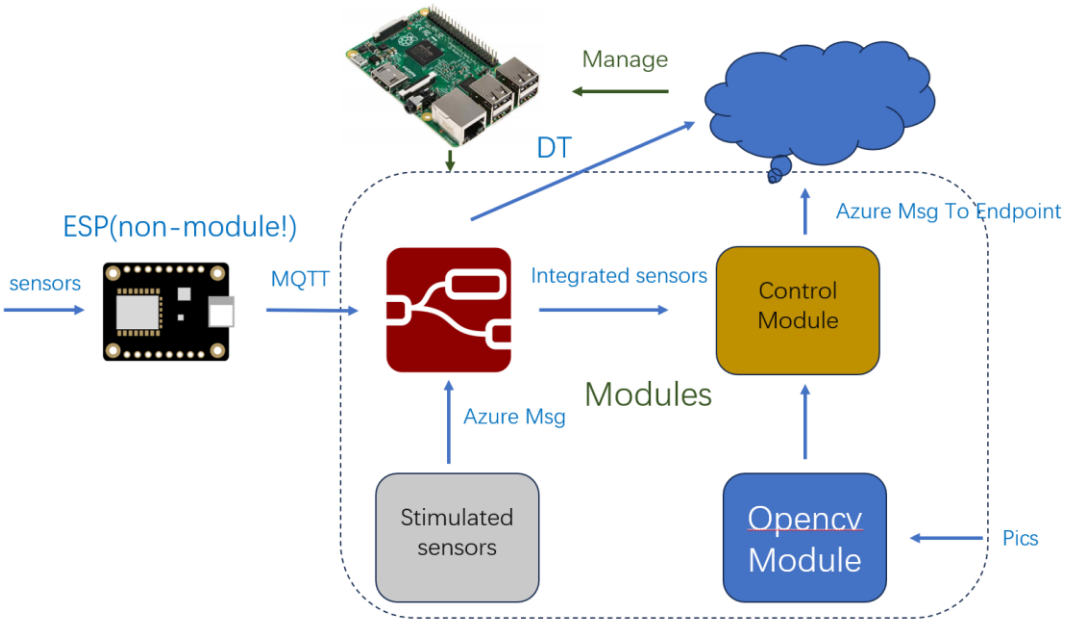
Structure of an Azure IoT edge device



Port & adapter structure

- Modularity: The application is divided into independent modules, where each module is responsible for a specific task. This modular approach enables better understanding and management of the entire system, as well as the ability to replace or extend modules when needed.

- Scalability: By connecting modules together, the system's functionality can be easily expanded. For example, new modules can be added to process different types of sensor data or incorporate additional control logic.



Modules on edge

Project Structure

The key components used in the project include:

Raspberry Pi:

The Raspberry Pi serves as the edge device, hosting the Node-RED environment and running the edge modules. It acts as a central hub for data processing and coordination at the edge of the cloud.

```
    },
    "$edgeHub": {
      "properties.desired": {
        "schemaVersion": "1.1",
        "routes": {
          "SimulatedTemperatureSensorToNodered": {
            "route": "FROM /messages/modules/SimulatedTemperatureSensor/* INTO BrokeredEndpoint(\"/modules/NodeREDonIoTEdge/inputs/input1\")"
          },
          "FromNodeREDonIoTEdge": {
            "route": "FROM /messages/modules/NodeREDonIoTEdge/outputs/* INTO BrokeredEndpoint(\"/modules/EdgeContorlModule/inputs/inputFromNodered\")"
          },
          "FromCv": {
            "route": "FROM /messages/modules/OpencvModule/outputs/* INTO BrokeredEndpoint(\"/modules/EdgeContorlModule/inputs/inputFromCv\")"
          },
          "FromEdgeContorlModuleToEndpoint": {
            "route": "FROM /messages/modules/EdgeContorlModule/outputs/* INTO INTO $upstream"
          }
        },
        "storeAndForwardConfiguration": {
          "timeToLiveSecs": 7200
```

Message routing in the endpoint of device

ESP8266:

It has 2 sensors: light detector and human detector. It is connected to local Wi-Fi network and facilitating MQTT communication with the Node-RED gateway. The ESP8266 module represents a lower-level device that **cannot directly connect to the cloud**.

Stimulated sensors(**module**):

Installed from edge marketplace. This module represents the capability to **directly connect** sensor devices to the cloud (such as Azure SDK).

Node-RED(**module**):

It acts as the edge module and gateway, responsible for receiving data from the ESP8266 modules over MQTT, and messages from Stimulated sensors, routed in the endpoint of the edge device. It has the ability to aggregate input from multiple different sources and output digital twins, and send messages. In addition, the node-red dashboard is also deployed. Node-RED plays a crucial role as a gateway. It serves as a bridge in the IoT system.

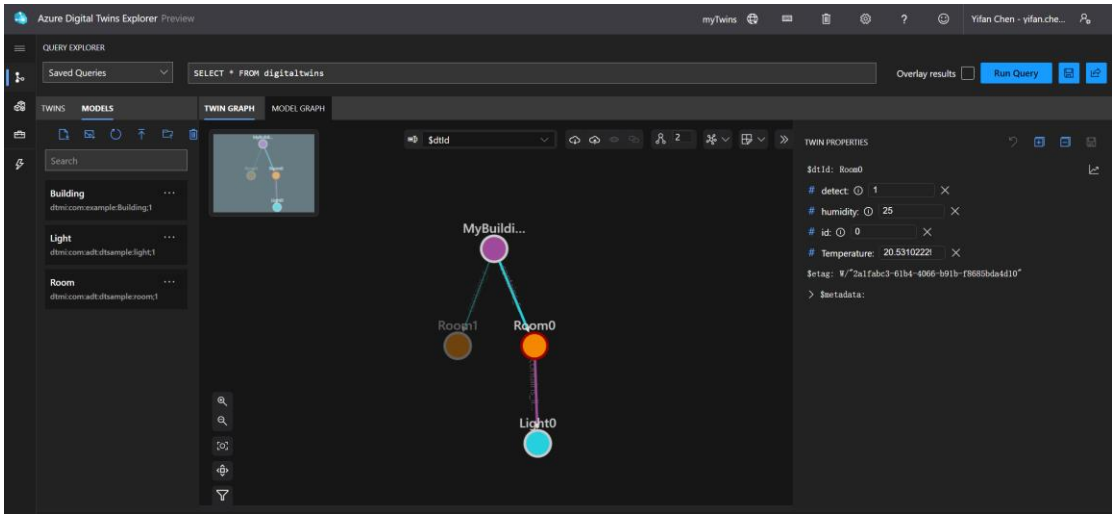
OpenCV(**module**):

The OpenCV library is used to read and analyze images captured by the connected devices, allowing the system to determine if there are any changes in the room environment.

Control module(**module**):

Performing various tasks, including image processing using OpenCV and controlling the virtual LED lights based on sensor information. It also integrates messages into the final output from device, to the endpoint of Azure iot hub. Then messages are routed to Azure Functions-> Digital twins, and Azure stream analysis-> Power BI.

The Control module is able to get properties from Digital twins, and then control assets locally.



Azure Cloud services:

The project integrates with Azure IoT Hub, Azure Stream Analytics, and Power BI to enable data visualization and analysis.

The project also implemented Azure functions and Azure Digital twins.

Azure functions receive messages routed from iot hub endpoint, and read the json data. Then using SDK updating twin models properties on the graph.

Azure Digital twins:

- Visualization and monitoring: Azure Digital Twins provides visualization tools and dashboards for real-time monitoring and visualization of device and system states. You can create custom dashboards to display device properties, metrics, and statuses, and perform real-time monitoring and alerting. This helps in better understanding and managing the operational status of devices.

- Real-time data processing and analytics: Digital Twins offers capabilities for real-time data processing and analytics, enabling you to monitor and analyze the data generated by devices. You can set up rules and conditions to trigger actions or alerts when the data meets specific criteria.
- Cloud service coordination: Digital Twins allows devices to collaborate and coordinate with each other. It enable us to define message passing, event triggering, and response rules between Azure services to facilitate their collaborative work.

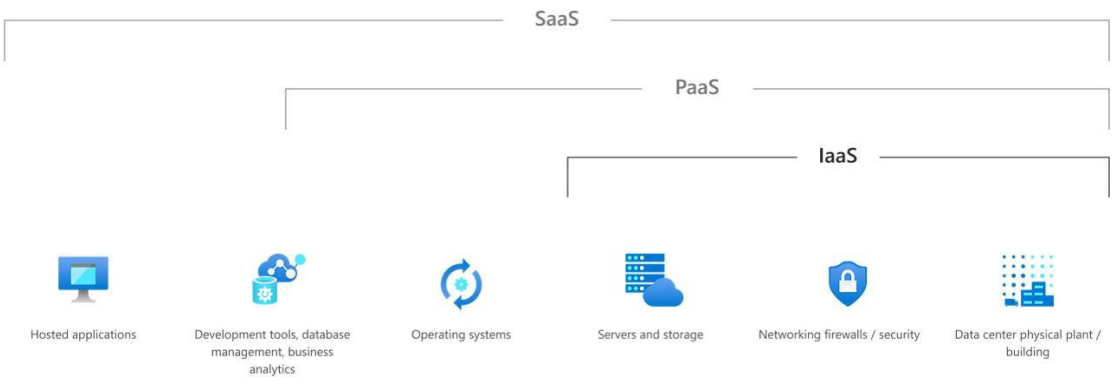
Interesting aspects:

- Edge computing with direct integration of sensors, enabling real-time data processing and decision-making at the edge.

- Development and integration of multiple modules, leveraging Azure IoT Edge Modules for enhanced portability and scalability.

-docker
Docker simplifies the packaging and deployment process, ensuring consistency and portability across different edge devices. By containerizing the modules, they can be easily managed and deployed from the cloud.

-Module **Twins** for Remote Monitoring and Management.
through module twins, we can easily monitor the state of modules, update configurations, and perform actions on specific modules from the cloud.



PaaS in the Project Context
In the project, the use of Azure IoT Edge modules and their deployment with Docker aligns with the Platform as a Service (PaaS) model. This approach provides a managed platform where developers can focus on building and deploying modules without the need to manage the underlying infrastructure. The platform takes care of managing the edge devices and provides services for module management, communication, and monitoring, enabling faster development and deployment of the IoT system.

Challenges:
-Development of modules using Azure IoT SDK and deployment using Docker.

-Effective utilization of the Azure platform, including **Azure Central, Power BI**, Azure Functions, and other services to achieve desired functionality. Also, SaaS could be costly…

-System integration, leveraging modular design principles and containerization techniques to simplify the overall system architecture and improve interoperability.

Possible extensions:

- Further development on Microsesrvices and architeture. Service-oriented design.
- The use of Node-RED as a **gateway** and the scalability for **expanding downstream devices** (Reference: Scaling Node-RED Horizontally for High Availability).
https://www.narendranaidu.com/2016/07/scaling-node-red-horizontally-for-high.html

Different topic/ Shared Subscription/ Other modules and extensions

- Further development on SaaS （iot central）. Dashbroad on the cloud.

References

https://jpaulm.github.io/fbp/index.html (flow based programming)
https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/iot
https://learn.microsoft.com/en-us/azure/architecture/guide/
https://www.narendranaidu.com/2016/07/scaling-node-red-horizontally-for-high.html