Sustainable Communication Networks
Prof. Dr. Anna Förster

An Autonomous Service Management For IoT Applications

Prerequisites:	- Good knowledge of Python and embedded programming - Have experience working with database
Level:	- This topic is appropriate for Master Students
Language:	- English

Introduction

In the context of Internet-of-Things (IoT), efficient and flexible service management techniques are essential to improve performance and cost-effectiveness. In this regard, it is crucial to equip the IoT devices with tools that allow a flexible, well-performing, and automated way of efficient service provisioning.

The Toit platform completely restructures the IoT development process and aims to fix the operational flexibility issues in IoT devices. This new platform is an end-to-end IoT software platform and a new programming language. Using a Toit API allows you to continuously update the code on your ESP32 microcontrollers, even over cellular connections and monitor IoT devices remotely.

The presented project focuses on the idea of service isolation and modularisation at the level of edge devices to observe IoT services and manage them under real-time requirements for an IoT smart farming scenario.

Project description

The following tasks need to be executed:

- Step 1: Developing ToiT programs to read measurements from Humidity+Temperature and accelerometer sensors:
  - In this step, write a code to read data from sensors with Toit language. The Humidity+Temperature sensors collect data every 10 minutes and accelerometer sensors measure constantly. Collected data is transferred via Wifi or Cellular and stored in the database
- Step 2: Implementing a MQTT to share data between your IoT edge devices and server:
  - In this step, we would like to have a message queueing system that allows us to share data between devices. To do this, use adafruit MQTT API to have to publish/subscriber and carry out messaging. This step is a requirement for the next step.

- Step 4: Define a model that is able to detect abnormal situations with accelerometer sensors:
  - In this step, analyse the data with some statistical evaluation or machine learning techniques to find a threshold to recognise a malicious event
- Step 3: Manage IoT services:
  - In this step, based on specific conditions that come from analysing the collected data, stop/start corresponding IoT services. You need to use MQTT and publish the analysing result for this step. Then a system administrator can reach this message and stop services.
- Step 5: Measuring the total energy consumption of ESP32 with and without TOIT+MQTT:
  - It is important how much energy the sensor takes to sense, process and transmit the data over the Wifi or cellular. In each phase, a different amount of current(mA) is taken, resulting in the consumed energy (mAh). So, consider two scenarios, one for raw implementation and the other for ToiT+MQTT implementation, where an accelerometer sensor constantly collects data, process and send data, and a Humidity+Temperature sensor every 10 mins does this one day. After one hour, a malicious event will happen, and the accelerometer not reliable anymore. With the help of ToiT+MQTT and the developed model, stop services related to the accelerometer. Note the energy consumption for that duration for both scenarios.
- Step 6: Documentation, publishing in Github, and presentation of the work.

Contact

If you are interested in this work, please contact us via mail: projects@comnets.uni-bremen.de

References

1. https://toit.io/
2. https://docs.toit.io/tutorials/mqtt
3. https://learn.adafruit.com/adafruit-io/mqtt-api