Documentation
Sensool Developement kit

Objectives

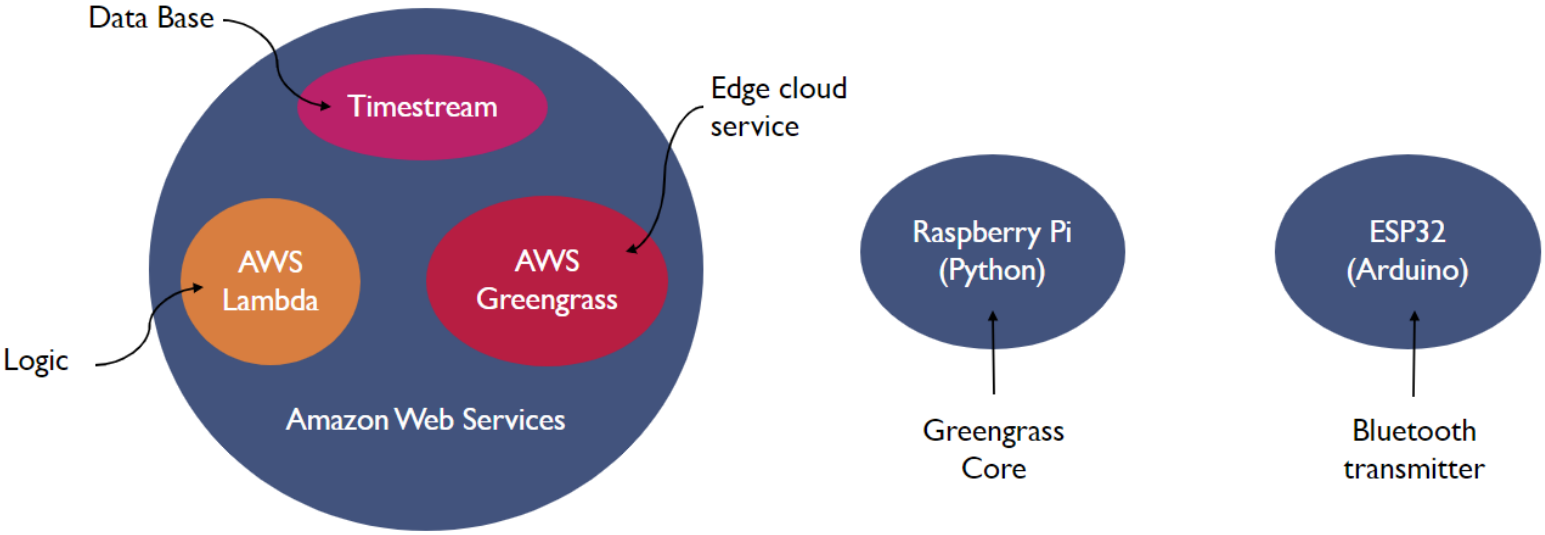
Sensool project aims to share a complete open source solution to interface sensors quickly and simply using AWS IoT cloud and AWS Greengrass feature. This repository is public and can be used, share and copied by anyone. The goal is to improve features over time by releasing to the public for free use. Do not hesitate to contact us at this e-mail address if you have any trouble : sensool22@gmail.com

Quick review

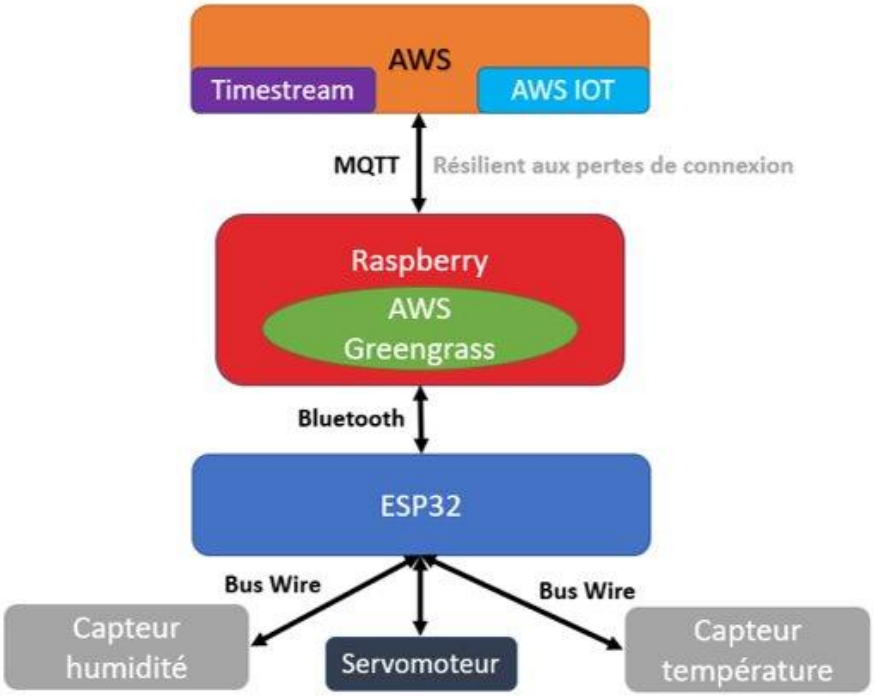
The solution is made to make your solution completely autonomous locally, even when there is no internet connection to the cloud. You can still save your data and keep your business working while the connection is lost, which make Sensool highly reliable and competitive on the IoT and sensor-centered market. Our solution is currently using AWS cloud services to get data to the cloud and AWS Greengrass to make an edge runtime service that can work locally at the edge. To program the logical part of our solution, we used AWS Lambda, that allows us to deploy lambda function that works locally.

As we want to make the solution applicable for everyone, highly flexible and improvable over time, we chose to use open source Hardware and Software. In terms of hardware, we are currently using Raspberry Pi to run the Greengrass core system that makes the system work at the edge of the network connectivity, we are also using ESP32 (Arduino) in order to deliver data over the air using Bluetooth technology for our local (autonomous) services. In terms of software, we are developing with high-level programming languages such as Python to get the data from Raspberry into the cloud, C/Arduino to get data from sensors and triggers and NodeJs to save our data on a database on the cloud.

Here are the technology we are currently using:



Here is how the actual system is working:



The communication technologies we are using in our project is Bluetooth connectivity locally and MQTT (publish/subscribe network protocol) in order to get the data into the cloud when the internet connection is back.

Basically, the project follows these steps:

Sensors → ESP32 (Bluetooth) → Raspberry Pi (MQTT) Topic → Lambda Function → Topic (MQTT) → Raspberry Pi (Bluetooth) → ESP32 → Triggers

Installation

Interfacing sensors with ESP32

You must first interface your sensors with ESP32 (if you need Bluetooth technology), you can use any Bluetooth/wifi module to get the data to the core, but the librairies that are in this repository are for the moment supported by ESP32 and plug & play.

Here is a list of sensors and librairies tested and supported by the ESP32 module:

- DHT11 temperature/humidity sensor
- Dallas Temperature sensor
- SensorTag
- Ultrasonic HC-SR04 sensor

Once your sensors are connected to the right GPIO pins on the ESP32, you can now download the installation package called “ESP32_local_scripts” that contains the code that get data for both Sensors and triggers.

1. Launch “ESP32_Sender_Sensors_Controller.ino” on the Arduino IDE, put the right GPIO Pin you used for data/signal in the variable called “DHTPIN”.

2. You can now release the code on the ESP and open the terminal to see if your data are received through the serial port.

Get the data to the ESP

The standard ESP code is sending data through Bluetooth technology. You should now setup the raspberry script that will receive this data.

1. Download the package called “Raspberry_local_scripts” which contains two scripts that will be used to make a bridge with the AWS cloud and the Greengrass core
2. To setup the Bluetooth connectivity, you should first connect you raspberry via Bluetooth to the ESP32 device via graphical interface.
3. Then, you can download the script called “ESP32_to_lambda” and have to setup the right MAC Address of the right ESP32 in order to receive the data. Get the MAC address and copy it in the variable called “addr”.

Setup Raspberry and MQTT connection

The next step is to setup the MQTT connectivity to the cloud.

You can follow these steps to configure your Greengrass core, install Greengrass and set up the Greengrass group which is made of a core (the raspberry) and some devices (the sensors/esp32) : https://docs.aws.amazon.com/fr_fr/greengrass/latest/developerguide/what-is-gg.html

Once the core is ready with Greengrass, the devices set up with right certificates and keys in them, we can start to code the bridge to the cloud to the local code. You should first add the right Certificate and Private Key in the code called “ESP32_to_lambda”.

Deploy the lambda function on the core

Download the package called “AWS_lambda_scripts” which contains a lambda function and all its libraries and SDK. Upload this function in AWS Lambda and link it to the right Greengrass group. Subscribe to the IoT Cloud Topics in this order: [IoT Cloud → Lambda function] with the right topic in filter and [lambda function → IoT Cloud] with another topic to get the data back after logical function.

You should now add to your local code every certificates and key path in order to get an identity to the devices you are using and allow them to be connected to the cloud. Everything needs to be set up in the “ESP32_to_lambda” script.

After the local code is linked to the AWS cloud, the data are published on the right topic and the lambda function getting this data and sending a response to them, we can set up the triggers.