Synopsis: The objective is to develop a prototype for Simple (IoT-enabled) Remote Temperature Management System that will activate a fan or air conditioning unit etc., (and optionally take action using IFTTT) when the temperature and/or humidity exceeds a user specified threshold. The concept architecture is illustrated on page 4.

To accomplish this project, you will need to develop a custom hardware and software solution. Some software will run locally on the device hardware to read and control sensors, and some software will be developed for the Azure Cloud and implemented as function as service (FaaS), and used activate sensors on your device, from the Cloud. The current version of this document is not intended as tutorial, but rather a checklist (with enough detail of the process) of the hardware/software parts required along with a basic outline of the steps required to produce a functional prototype.

**** Before you begin:** Putting the pieces together to build a complete working IoT device (from scratch) that communicates with the Cloud is great exercise, but it requires some foundational background/programming skills to be successful. I recommend viewing this tutorial first, to help you get started:

1. View the first hour: https://www.youtube.com/watch?v=sKaSBh1M4M4
2. Related tutorial for the video: https://docs.microsoft.com/en-us/learn/modules/create-iot-device-dotnet/1-introduction

1.0 Hardware prerequisites:
You will need…

1 x

- Raspberry Pi (version 2b/3b/4b), running the latest version of Raspberry PI OS
    - (note: the RPi 2b requires a WIFI dongle for internet connectivity)
- T-Cobbler plus GPIO breakout and a full-sized breadboard for circuit prototyping
- BME280 humidity/barometric pressure/temperature sensor
- 5 mm LED (example)
- 220-ohm resistor (example)
- Jumper wires (example)
- 5v Relay Board Relay Module 1 Channel (example)

1.1 Construct the circuit

- connecting the BME 280 (Temp & Humi) sensor using i2c, as shown in the picture (below) is a complete solution for the circuit (only use one LED and resistor)
- Note: **Use this detailed tutorial** to help you construct your circuit hardware device:
    - https://docs.microsoft.com/en-us/learn/modules/create-iot-device-dotnet/2-construct-iot-hardware

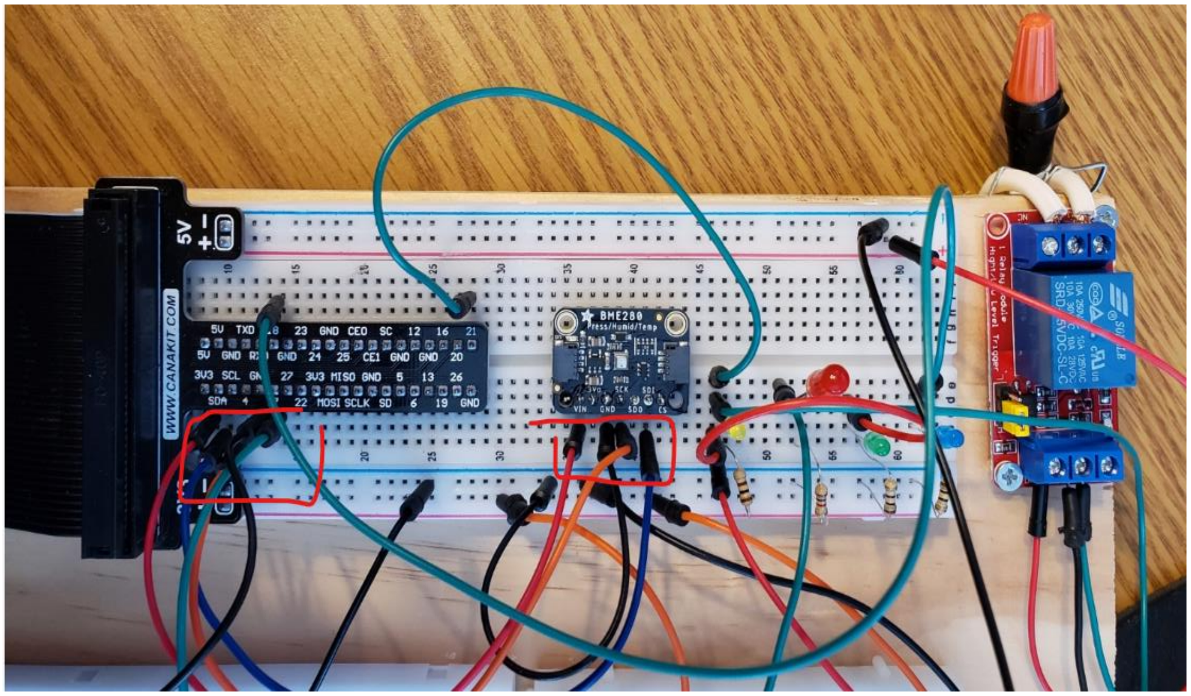

Figure 1: Example device solution

2. Software requirements: Develop a software application to read from the BME sensor over I2c

Prerequisites: (note can be accomplished using several programming languages, we discuss C# in doc, which very much like Java). Regardless of the language you choose, the code logic is pretty much the same (Python etc.)

- **You will need:**

- Visual Studio Code with Microsoft C# extensions.
    - .Net 7.0 and the .NET IoT Libraries GPIO and Bindings
    - Setup SSH to deploy code to your RPi

Note: building the code to control devices is a bit to bite off the first around, and a project in and of itself. Please use this (related) tutorial to get started. It will basically give you the C# code example you need to control your device:

- https://docs.microsoft.com/en-us/learn/modules/create-iot-device-dotnet/4-use-dotnet-iot-libraries

2.1 Write the code to read the BME temperature and humidity telemetry and send to the Azure Cloud IoT Hub service using the Azure MQTT device SDKs

Prerequisites: You will need …

- **Azure Cloud subscription (Anthony has a trial Azure subscription)**, with the roles to create resources
    - Create an IoT hub (free tier) and manually provision (register) your device)
        - Note: you get a device connection string, enabling your device to connect securely using MQTT/TLS 1.2 to the Azure IoT Hub cloud service
    - Add the Azure device SDK device library packages to your project and develop a class with member functions that will read the BME 280 sensor data and package the data as JSON formatted telemetry that sends to IoT Hub over MQTT. Your *Send* function should specify and interval defining the frequency for sending to the Cloud.
    - Please use this example to help you create an IoT hub instance in the Azure portal:
        - https://docs.microsoft.com/en-us/azure/iot-develop/quickstart-send-telemetry-iot-hub?toc=%2Fazure%2Fiot-hub%2Ftoc.json&bc=%2Fazure%2Fiot-hub%2Fbreadcrumb%2Ftoc.json&pivots=programming-language-csharp

3.0 Build a function runs in Cloud, and triggers whenever telemetry arrives at the IoT hub.

To accomplish this, you will need to develop code that is deployed as an Azure managed Function-as-a-service (FaaS). Essentially, this code reads and decode the JSON formatted telemetry as it arrives at the IoT Hub and monitors the temperature/humidity. If the temperature and/or humidity exceed some specified threshold, then the function code should send a cloud to device message back to your device (running on premises, and activate the relay), which in-turn activates an external device such as a fan.

Please use this example to begin understanding of Azure Functions: https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-event-iot-trigger?tabs=in-process%2Cfunctionsv2%2Cextensionv5&pivots=programming-language-csharp

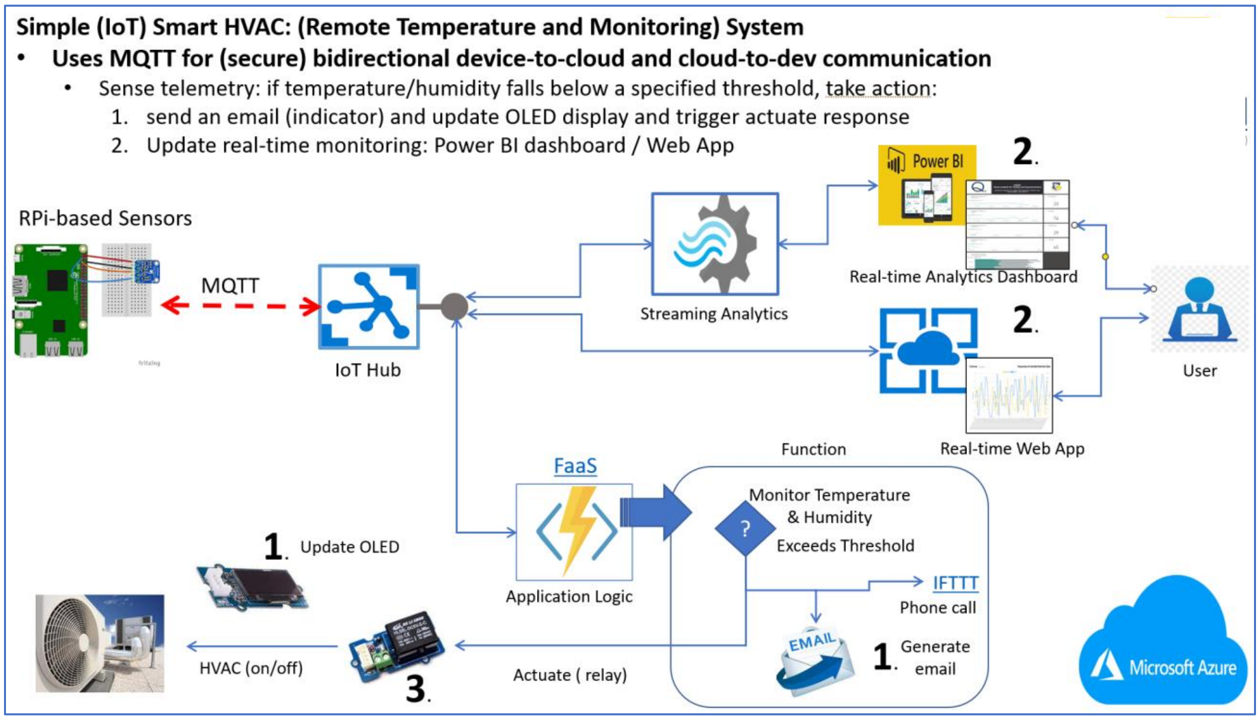
Simple (IoT) Smart HVAC: (Remote Temperature and Monitoring) System
• Uses MQTT for (secure) bidirectional device-to-cloud and cloud-to-dev communication
• Sense telemetry: if temperature/humidity falls below a specified threshold, take action:
1. send an email (indicator) and update OLED display and trigger actuate response
2. Update real-time monitoring: Power BI dashboard / Web App
RPi-based Sensors
MQTT
IoT Hub
Streaming Analytics
Power BI
2.
Real-time Analytics Dashboard
2.
Real-time Web App
User
FaaS
Application Logic
Function
Monitor Temperature & Humidity
Exceeds Threshold
IFTTT
Phone call
EMAIL
1. Generate email
1. Update OLED
HVAC (on/off)
3.
Actuate (relay)
Microsoft Azure

Figure 2: Conceptual Architecture