Microsoft Azure Internet of Things

Microsoft

IOT & Azure Scenarios

Predict equipment failures before they happen

Head off potential problems, while promoting equipment efficiency, with predictive maintenance. Collect and analyze data from your connected assets to proactively plan maintenance, decrease downtime, and improve retention of the asset value.

Rolls-Royce

"Our TotalCare maintenance program was revolutionary in the '90s, so we're pioneers ourselves, and by collaborating with a fellow pioneer like Microsoft, we can absolutely bring innovative digital solutions to airlines now."

Alex Dulewicz, Head of Marketing for Services

Increase asset reliability with predictive maintenance

Focus on what matters most to your customers: reliability. Vastly improve operations and asset availability with predictive, and even preemptive, maintenance by gathering and transforming data from sensors and systems.

thyssenkrupp

"I call it the 'virtual troubleshooter'. When the elevator reports that it has a problem, it sends out an error code and the three or four most probable causes of that error code. In effect, our field technician is being coached by this expert citizen."

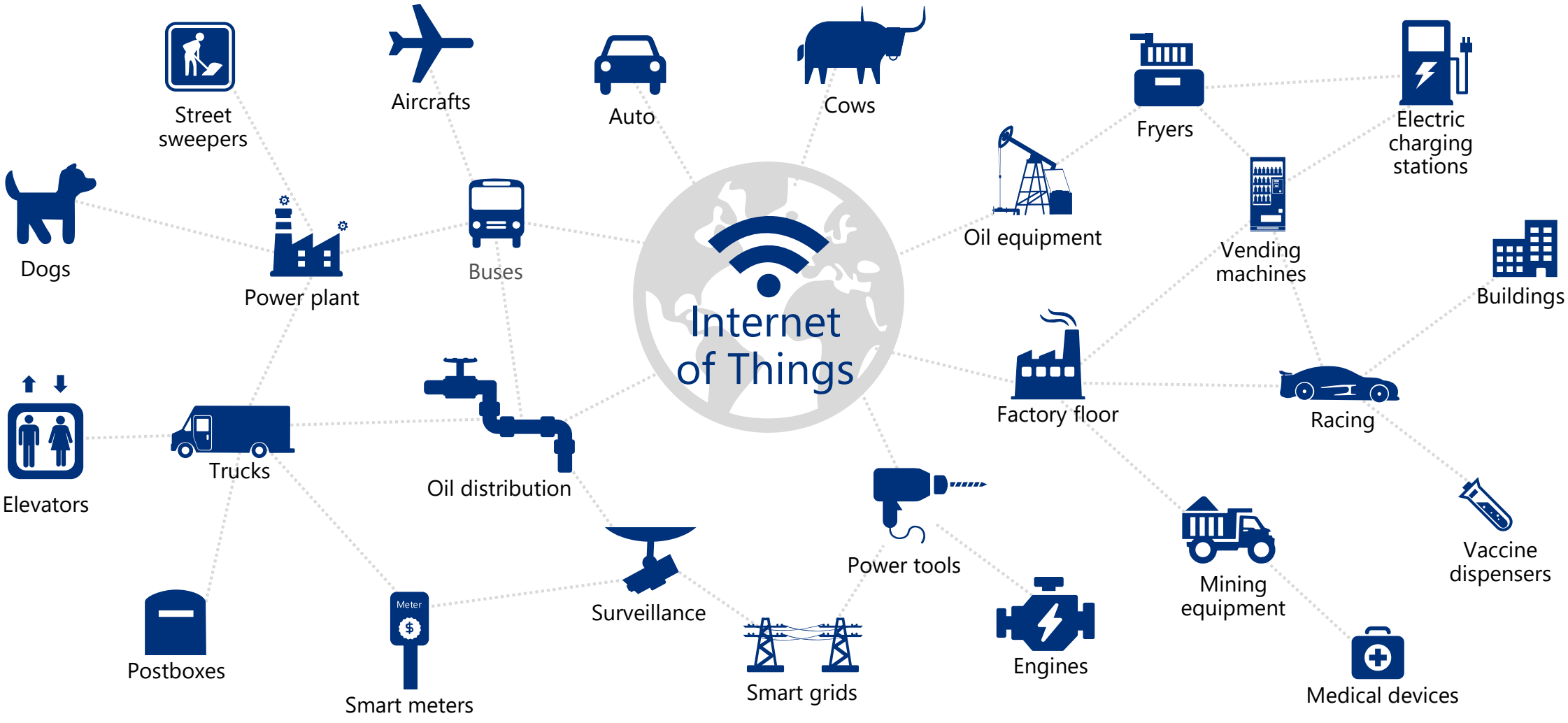
Rory Smith, Director of Strategic Development for the Americas

Benefits

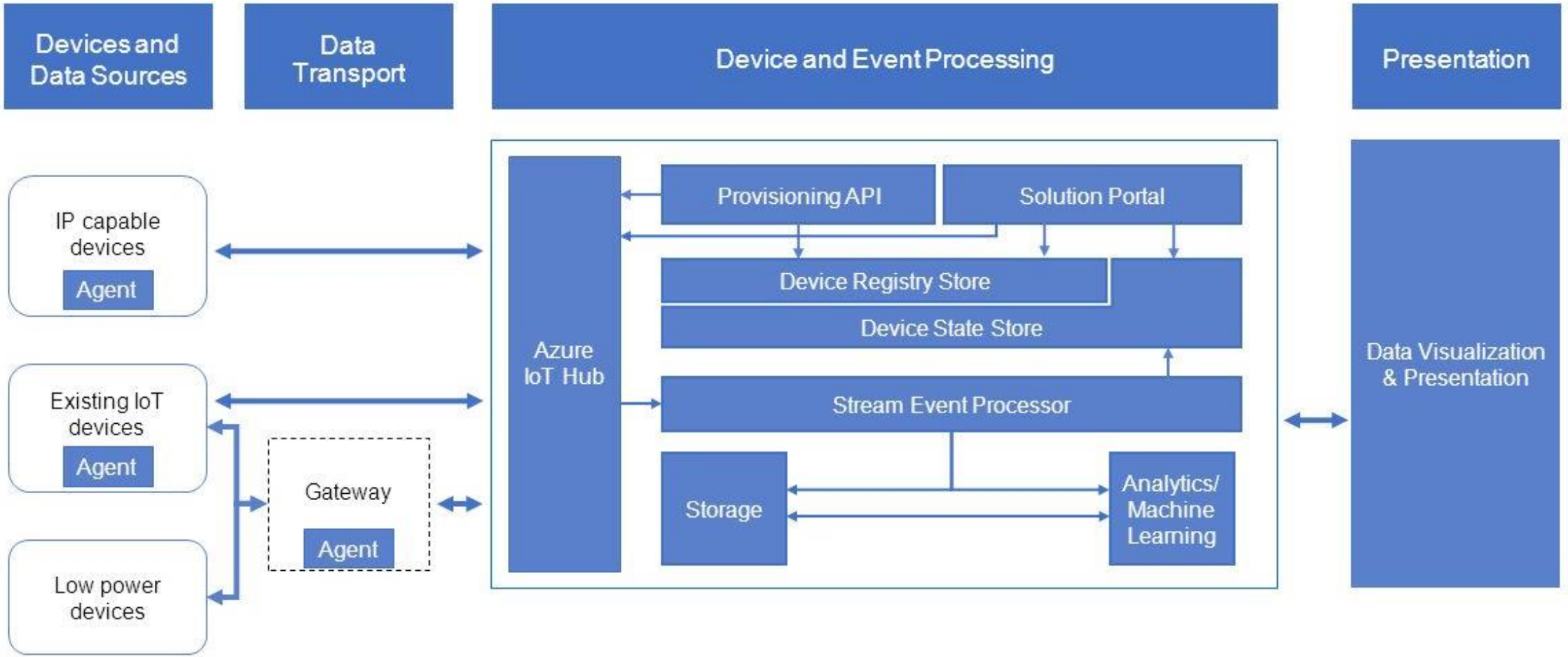
- Improves access to production and supply chain data worldwide, reducing costly downtime and maintenance, and increasing productivity
- Accelerates and supports business growth with a highly scalable cloud platform
- Spurs innovation and builds a competitive advantage with easier development and faster time-to-market with new features

IoT is already delivering tangible results

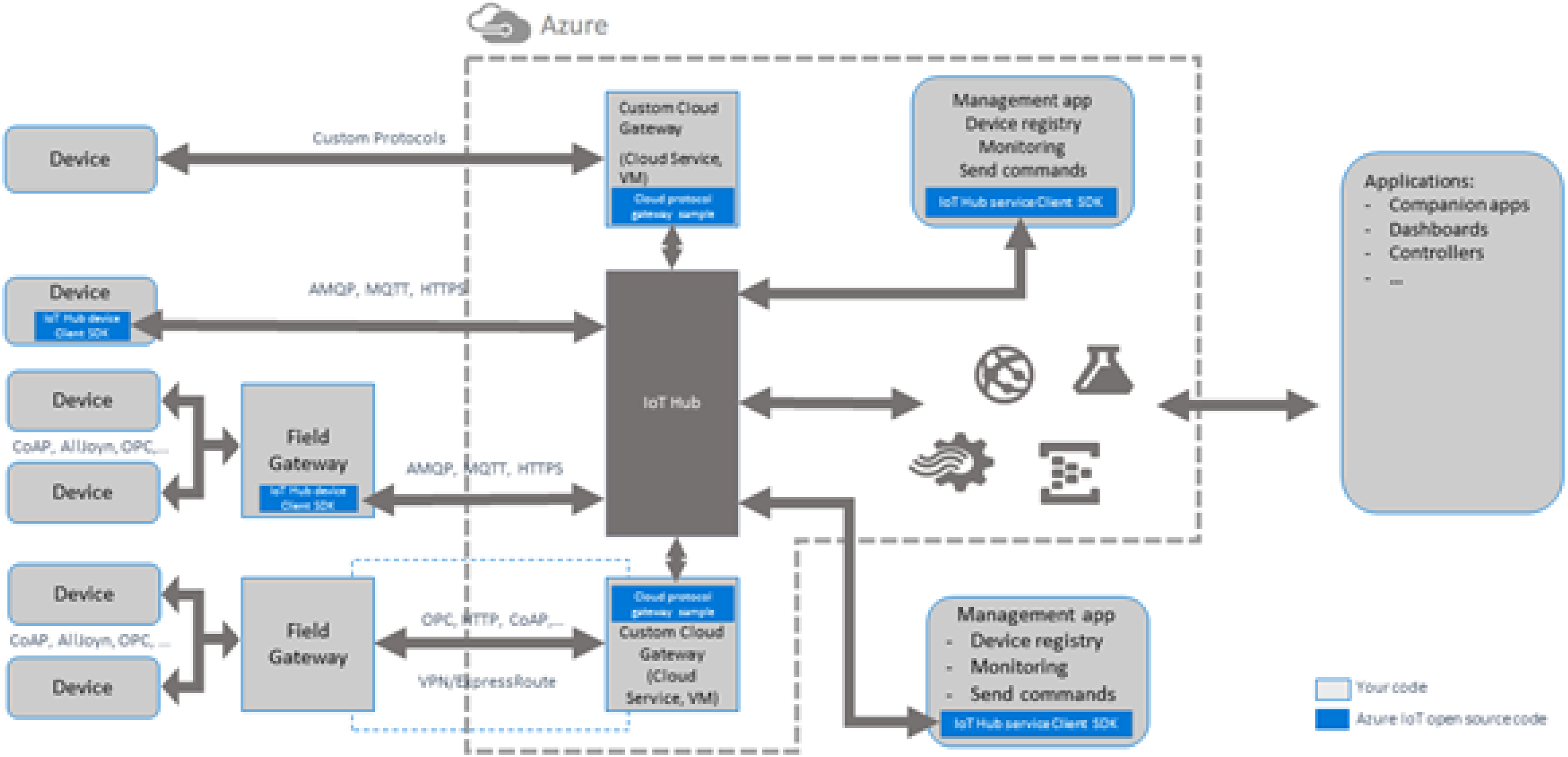
Innovation at work – real world IoT use cases



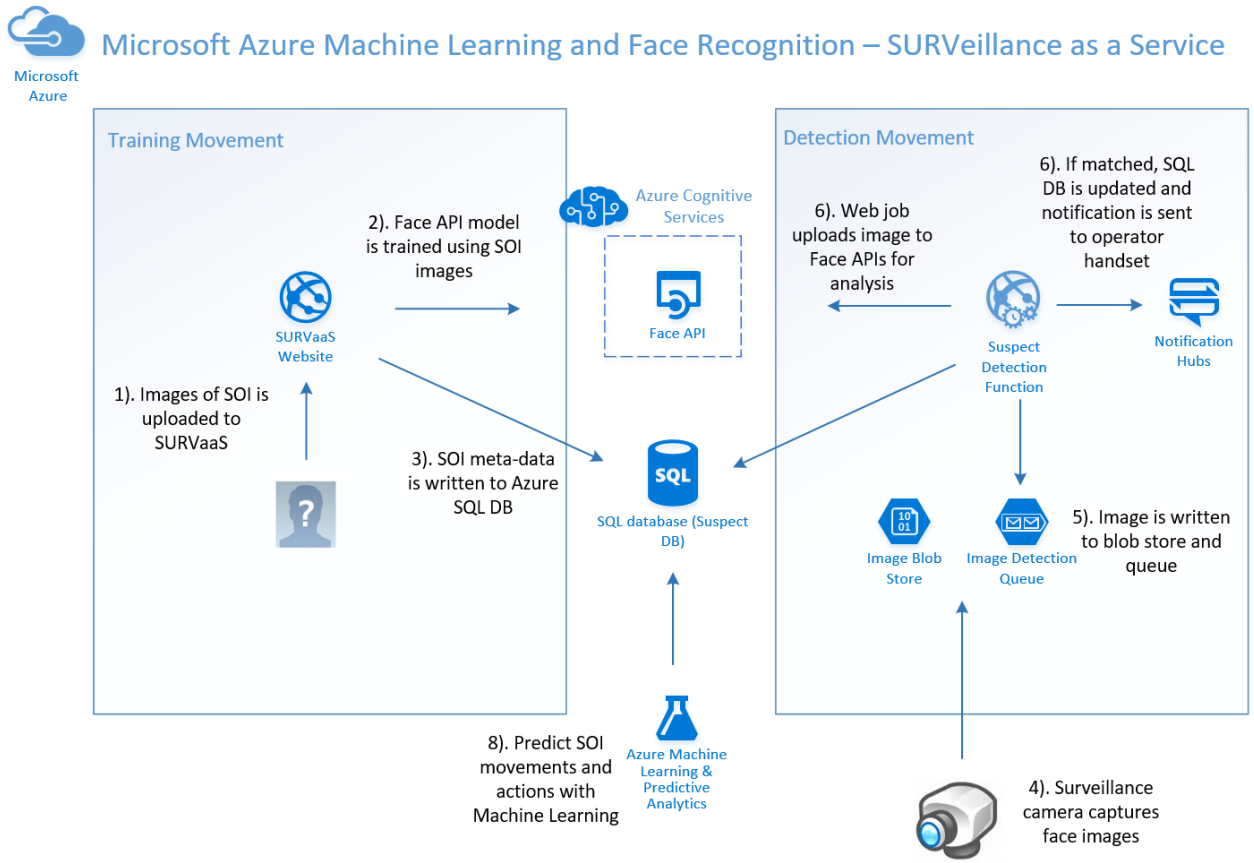
Azure IOT Solution Architecture

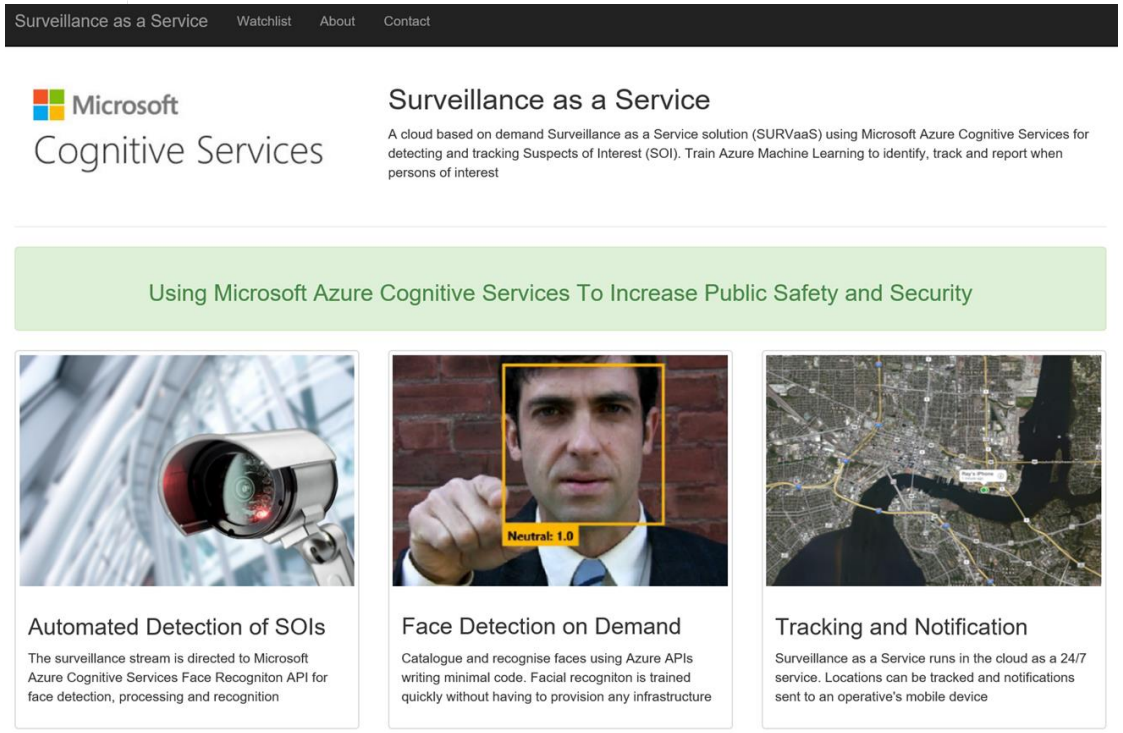


Azure IOT Solution Architecture



Combine IOT and Azure AI





Getting Started – A first test app

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-node-node-getstarted

Connect your simulated device to your IoT hub using Node

2017-8-31 • 10 minutes to read • Contributors all

Node.js

Introduction

Azure IoT Hub is a fully managed service that enables reliable and secure bi-directional communications between millions of Internet of Things (IoT) devices and a solution back end. One of the biggest challenges that IoT projects face is how to reliably and securely connect devices to the solution back end. To address this challenge, IoT Hub:

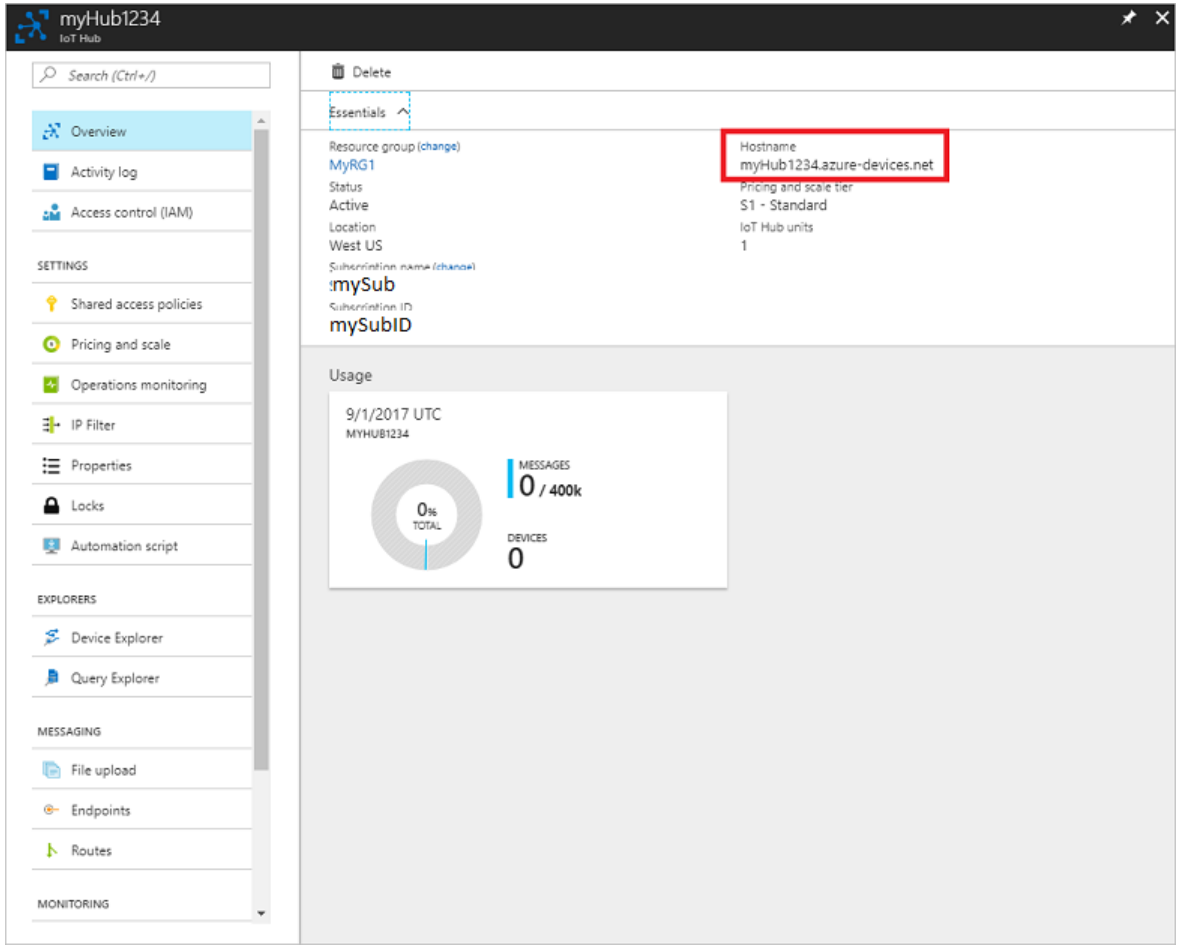
- Offers reliable device-to-cloud and cloud-to-device hyper-scale messaging.
- Enables secure communications using per-device security credentials and access control.
- Includes device libraries for the most popular languages and platforms.

This tutorial shows you how to:

- Use the Azure portal to create an IoT hub.
- Create a device identity in your IoT hub.
- Create a simulated device app that sends telemetry to your solution back end.

At the end of this tutorial, you have three Node.js console apps:

- **CreateDeviceIdentity.js**, which creates a device identity and associated security key to connect your simulated device app.
- **ReadDeviceToCloudMessages.js**, which displays the telemetry sent by your simulated device app.



Helpful Links

For participants who are more advanced and ready to start connecting their devices to the Azure Internet of Things services we have the Azure IOT Developer Center https://azure.microsoft.com/en-gb/develop/iot/

Instructions for getting started with Azure IOT - https://azure.microsoft.com/en-gb/blog/developer-s-introduction-to-azure-iot/

Azure IOT patterns -

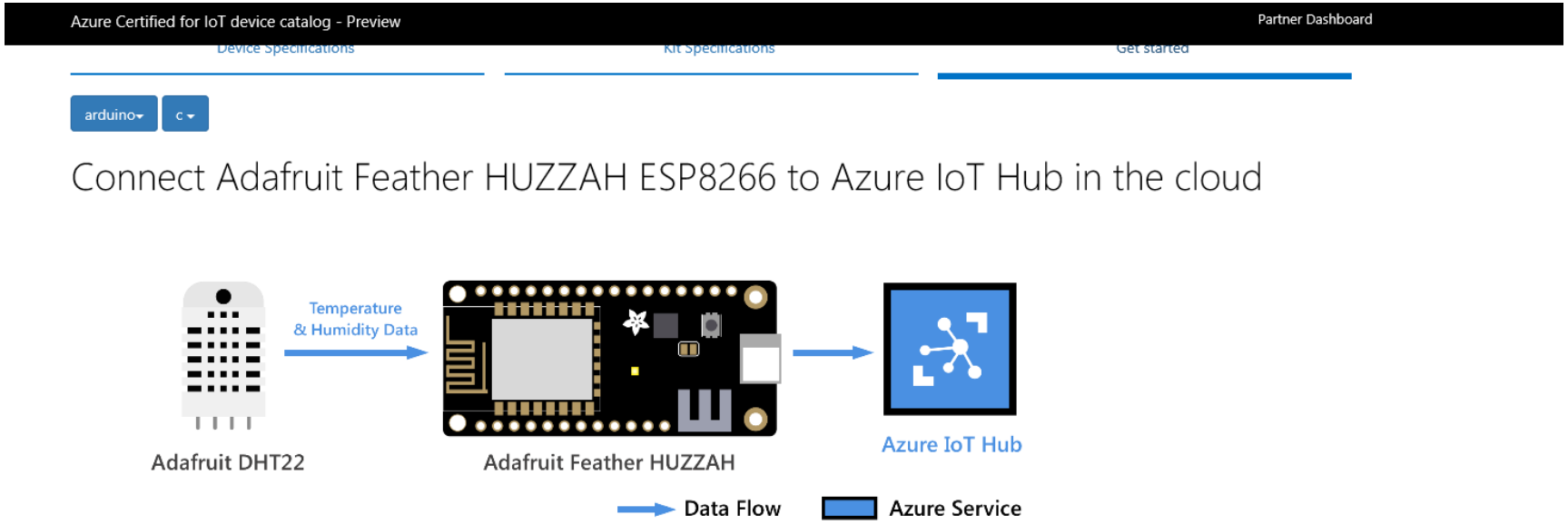
For learning how to code microbits and other circuit based hardware kits, Microsoft have created "MakeCode" https://makecode.com/ which offers a MicroBit simulator and development environment. The site also has programming guides and browser based development environments to learn how to write JavaScript.

How to stream data from the Microbit to Azure here - https://blogs.msdn.microsoft.com/uk_faculty_connection/2016/08/01/getting-started-microbit-microsofts-new-www-codethemicrobit-com-environment/

Instructions and libraries for Arduino and Azure IOT here - https://github.com/Azure/azure-iot-arduino

Adafruit Feather HUZZAH & Azure IoT Hub

https://catalog.azureiotsuite.com/details?title=Adafruit-Feather-Huzzah&source=home-page



What you do

Connect Adafruit Feather HUZZAH ESP8266 to an IoT hub that you create. Then you run a sample application on ESP8266 to collect the temperature and humidity data from a DHT22 sensor. Finally, you send the sensor data to your IoT hub.

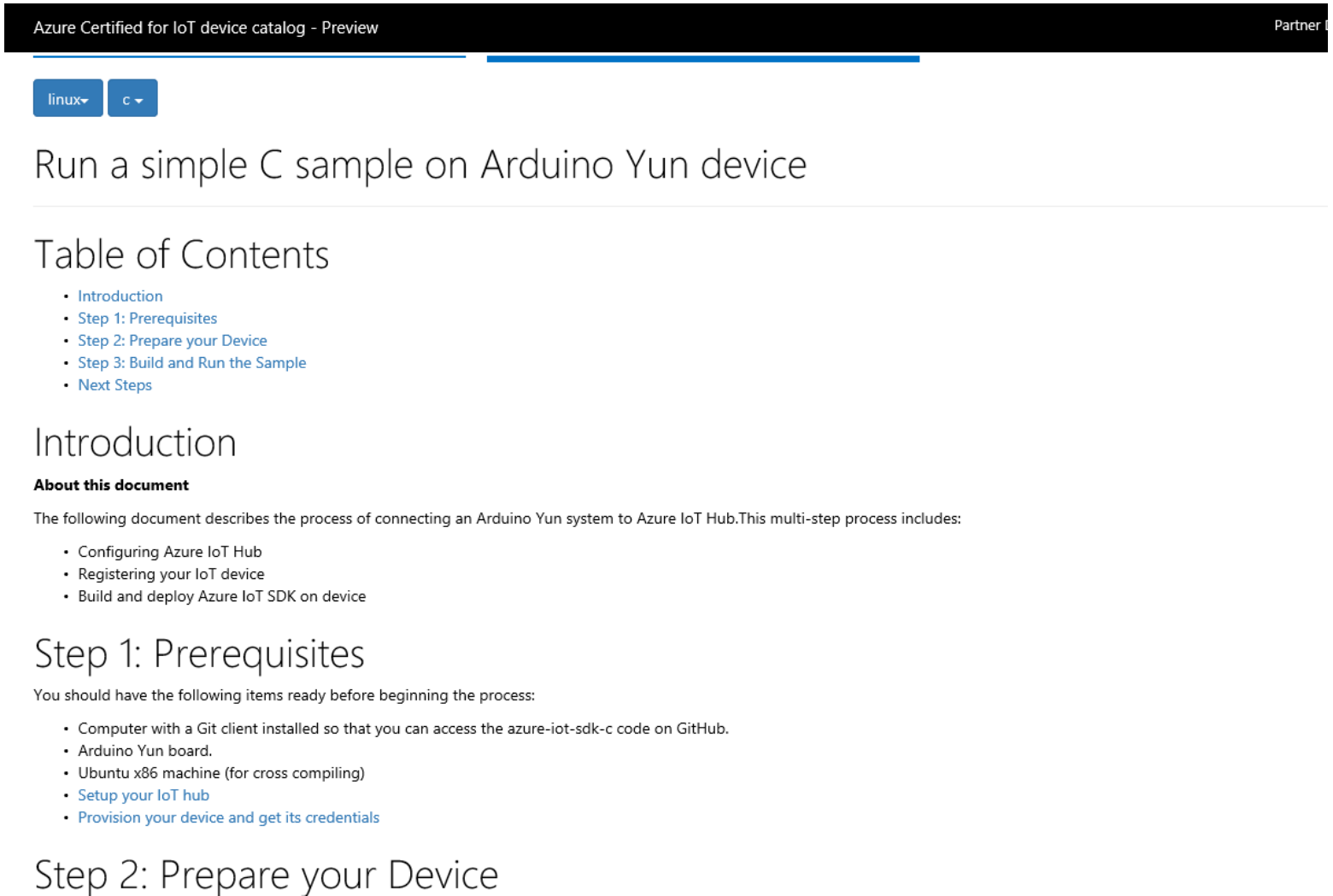
> **NOTE:** If you're using other ESP8266 boards, you can still follow these steps to connect it to your IoT hub. Depending on the ESP8266 board you're using, you might need to reconfigure the `LED_PIN`. For example, if you're using ESP8266 from AI-Thinker, you might change it from `0` to `2`. Don't have a kit yet? Get it from the Azure website.

What you learn

- How to create an IoT hub and register a device for Feather HUZZAH ESP8266
- How to connect Feather HUZZAH ESP8266 with the sensor and your computer

Arduino & Azure IoT Hub

https://catalog.azureiotsuite.com/details?title=Arduino_Yun&source=home-page



Azure IoT Hub library for Arduino

https://github.com/Azure/azure-iot-arduino

AzureIoTHub - Azure IoT Hub library for Arduino

This library is a port of the Microsoft Azure IoT device SDK for C to Arduino. It allows you to use several Arduino compatible boards with Azure IoT Hub. Please submit any contribution directly to azure-iot-sdks.

Currently supported hardware:

- Atmel SAMD Based boards
  - Arduino/Genuino MKR1000
  - Arduino/Genuino Zero and WiFi Shield 101
  - Adafruit Feather M0
- ESP8266 based boards with esp8266/arduino
  - SparkFun Thing
  - Adafruit Feather Huzzah

Prerequisites

You should have the following ready before beginning with any board:

- Setup your IoT hub
- Provision your device and get its credentials
- Arduino IDE 1.6.12
- Install the `AzureIoTHub` library via the Arduino IDE Library Manager
- Install the `AzureIoTUtility` library via the Arduino IDE Library Manager
- Install the `AzureIoTProtocol_MQTT` library via the Arduino IDE Library Manager

Simple Sample Instructions

Raspberry Pi Simulator & Azure

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-raspberry-pi-web-simulator-get-started

Microsoft Azure

SALES 1-800-867-1389 | CONTACT SALES | MY ACCOUN

Why Azure | Solutions | Products | Documentation | Pricing | Training | Marketplace | Partners | Blog | Resources | Support

Azure / IoT Hub

Filter

- Overview
- Get Started
  - Set up your device
    - Simulate a device on your PC
    - Use an online simulator
    - Use a physical device
  - Extended IoT scenarios
- How To
- Reference
- Related
- Resources

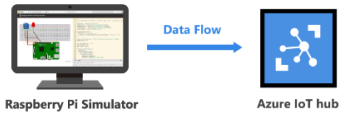
Connect Raspberry Pi online simulator to Azure IoT Hub (Node.js)

2017-7-28 • 5 min to read • Contributors

Raspberry Pi web simulator

In this tutorial, you begin by learning the basics of working with Raspberry Pi online simulator. You then learn how to seamlessly connect the Pi simulator to the cloud by using Azure IoT Hub.

If you have physical devices, visit Connect Raspberry Pi to Azure IoT Hub to get started.

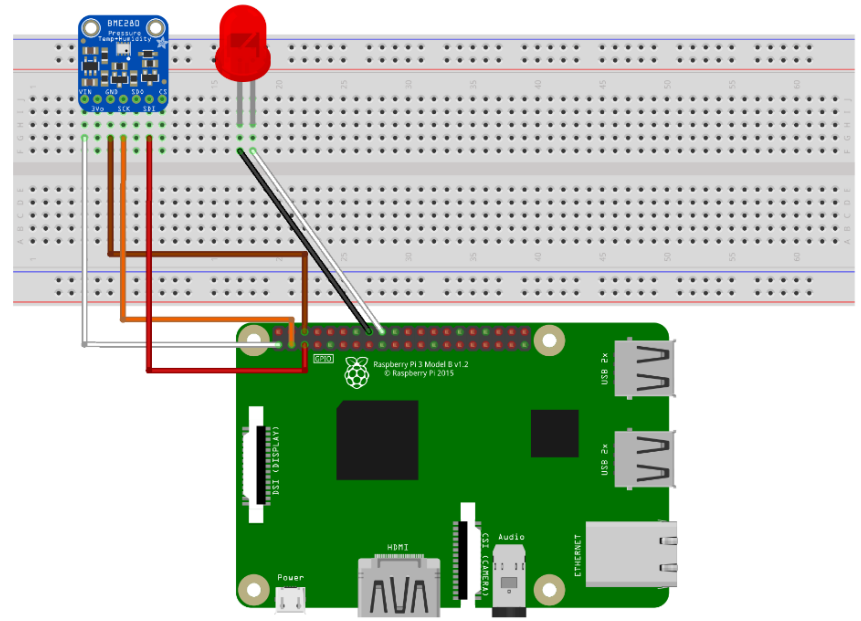


START RASPBERRY PI SIMULATOR

What you do

- Learn the basics of Raspberry Pi online simulator.
- Create an IoT hub.
- Register a device for Pi in your IoT hub.
- Run a sample application on Pi to send simulated sensor data to your IoT hub.

Raspberry Pi Azure IoT Online Simulator



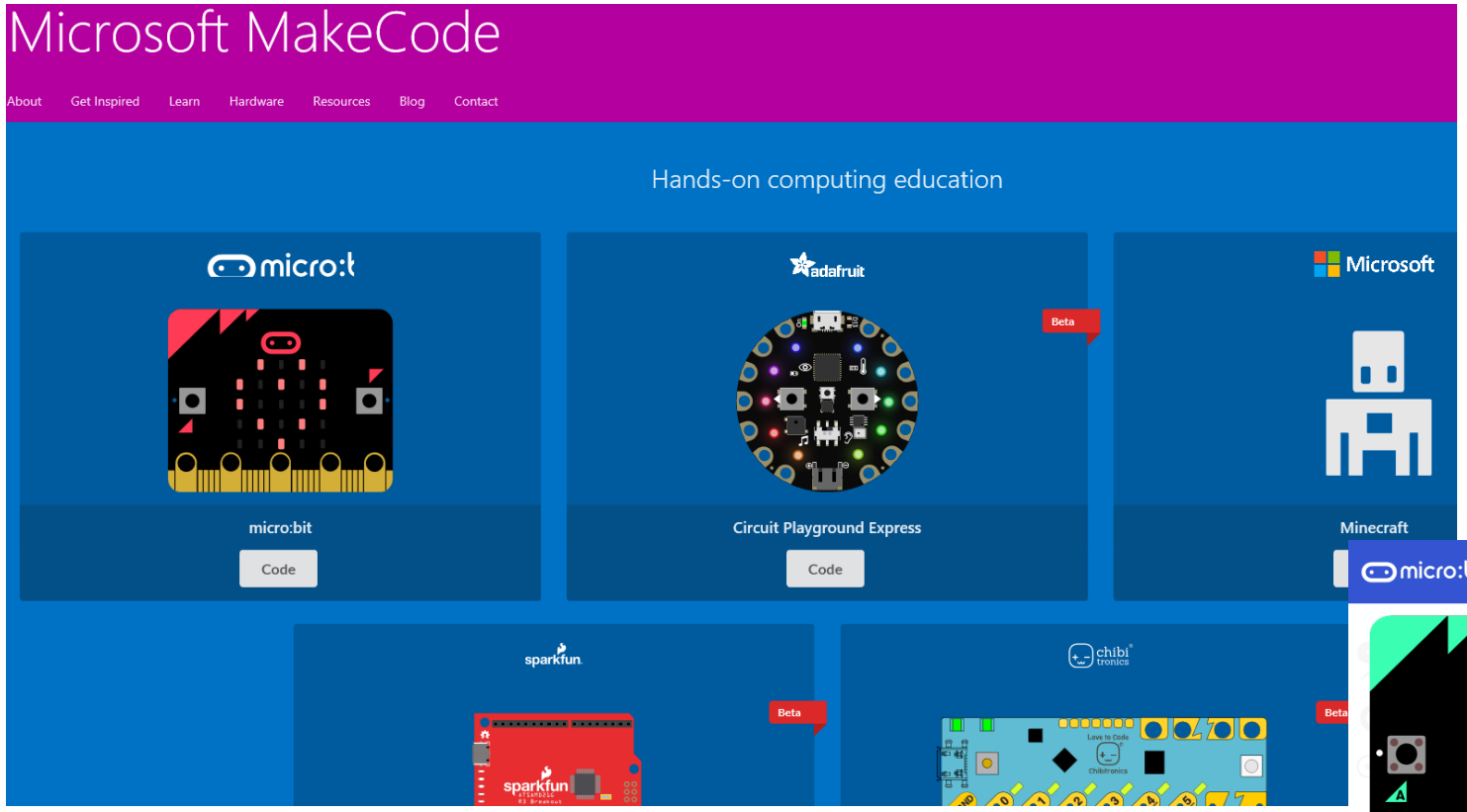

```
/*
* IoT Hub Raspberry Pi NodeJS - Microsoft Sample Code - Copyright (c) 2017 - Licensed MIT
*/
const wpi = require('wiring-pi');
const Client = require('azure-iot-device').Client;
const Message = require('azure-iot-device').Message;
const Protocol = require('azure-iot-device-mqtt').Mqtt;
const BME280 = require('bme280-sensor');

const BME280_OPTION = {
  i2cBusNo: 1, // defaults to 1
  i2cAddress: BME280.BME280_DEFAULT_I2C_ADDRESS() // defaults to 0x77
};

const connectionString = '[Your IoT hub device connection string]';
const LEDPin = 4;

var sendingMessage = false;
var messageId = 0;
var client, sensor;
var blinkLEDTimeout = null;

function getMessage(cb) {
  messageId++;
  sensor.readSensorData()
    .then(function (data) {
      cb(JSON.stringify({
        messageId: messageId,
        deviceId: 'Raspberry Pi Web Client',
        temperature: data.temperature_C,
        humidity: data.humidity
      }), data.temperature_C > 30);
    })
    .catch(function (err) {
      console.error('Failed to read out sensor data: ' + err);
    });
}

function sendMessage() {
  if (!sendingMessage) { return; }

  getMessage(function (content, temperatureAlert) {
    var message = new Message(content);
    message.properties.add('temperatureAlert', temperatureAlert.toString());
    console.log('Sending message: ' + content);
```

Run Reset

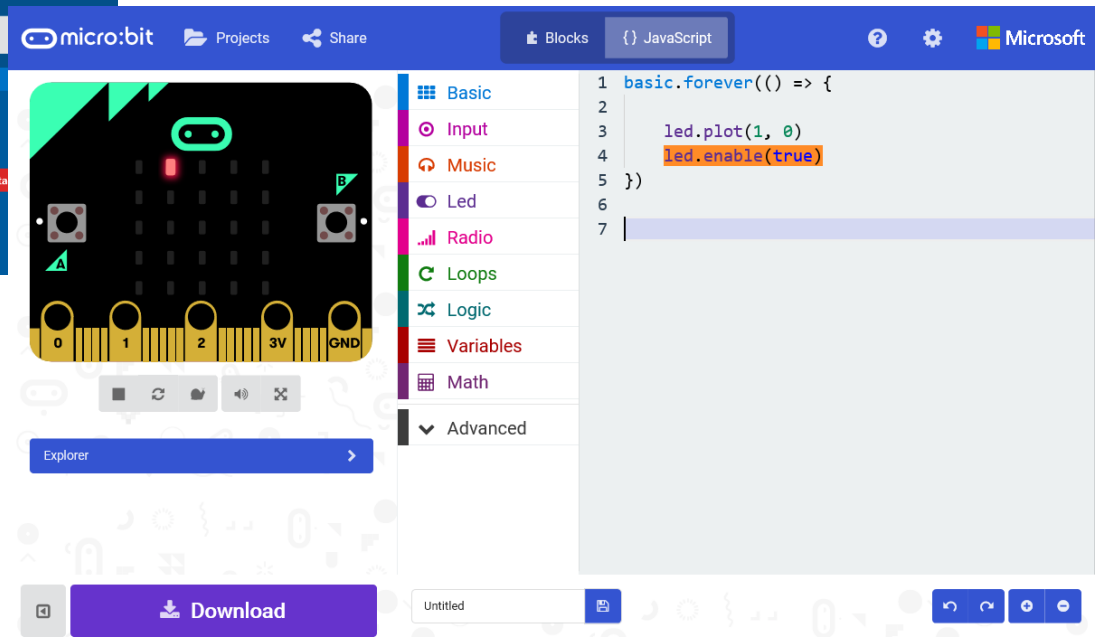
Click `Run` button to run the sample code(When sample is running, code is read-only).
Click `Stop` button to stop the sample code running.
Click `Reset` to reset the code.We keep your changes to the editor even you refresh the page.

>

Learn to Code

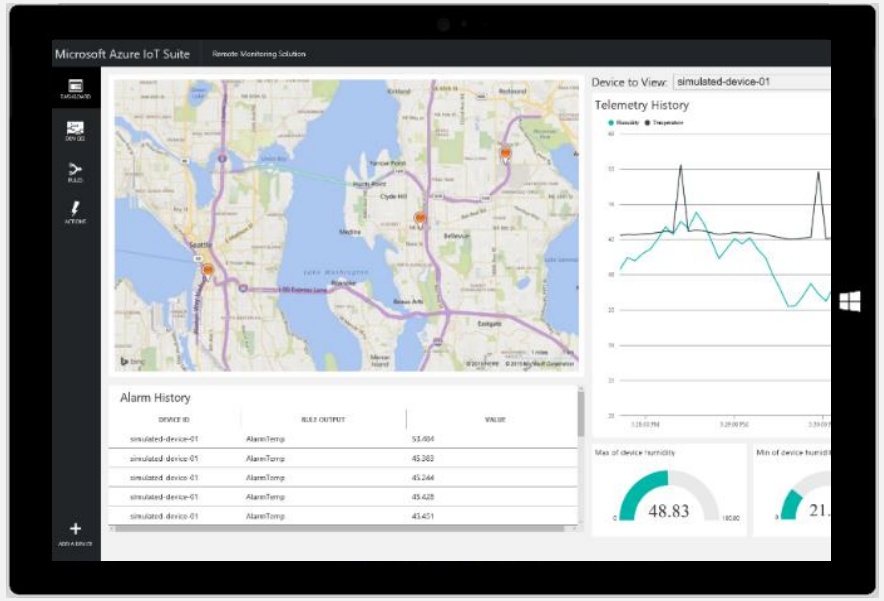


https://makecode.com/



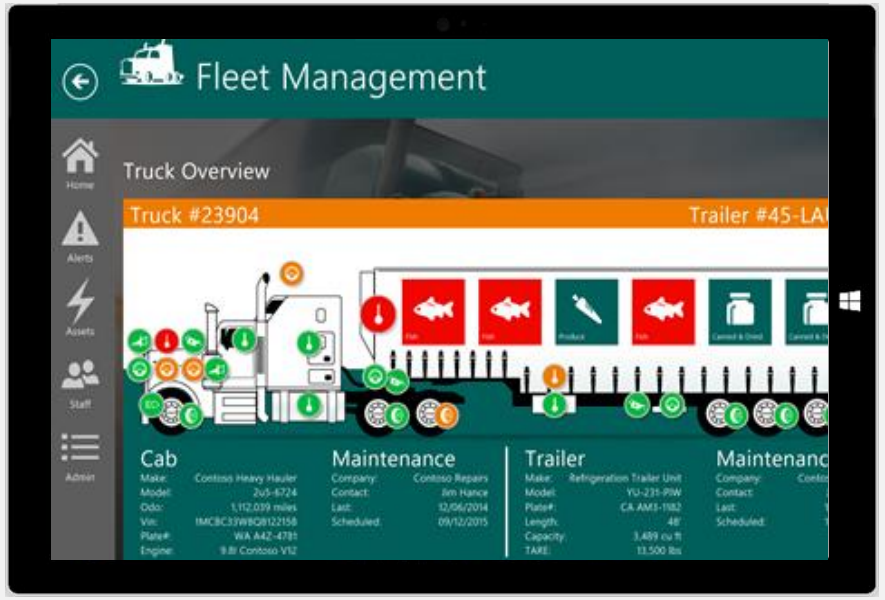
Accelerate time to value with preconfigured solutions

Start quickly for common IoT scenarios



- Get started in minutes
- Modify existing rules and alerts
- Add your devices and begin tailor to your needs

Finish with your Internet of Things application



- Fine-tuned to specific assets and processes
- Highly visual for your real-time operational data
- Integrate with back-end systems

Microsoft

DEMO

https://rolls-royce.azurewebsites.net