BUILDING IOT

Open Remote is a 100% free open source IoT platform. The user-friendly interface makes it possible for companies and individuals to integrate and manage their assets within one central location, design applications and workflows, and visualize data in dashboards. The platform also supports an array of data processing tools.

Some of the key capabilities of Open Remote are:

Integration of different asset types,

support of standard protocols such as HTTP or MQTT as well as specific protocols like KNX for connecting IoT devices,

a customizable manager interface for the automation, monitoring, and control of processes, apps, and devices,

a mobile app so that you can connect to your phone services and push notifications,

Multi-tenancy with user roles and a full account management service.

ThingsBoard is an open-source IoT platform for the collection processing, analysis, and visualization of data. The platform also comes with a device management service and users can connect their devices using any standard protocol such as HTTP, MQTT, or CoAP. With ThingsBoard, users can build and manage their own workflows.

ThingsBoard features include:

IoT device management with monitoring and control mechanisms,

scalability with the capability to orchestrate multiple devices at the same time, across the entire IoT ecosystem

enables users to create and manage alerts for connected devices (e.g. in the event of a disconnect or inactivity), other assets, and customers with real-time alarm monitoring.

extending default functionality with customizable rule chains, widgets, and transport implementations,

multi-tenancy

Thinger.io

Thinger.io is a scalable IoT cloud platform for connecting devices. You have an easy-to-navigate ready-to-use cloud infrastructure that allows users to integrate, monitor, and control millions of IoT devices. The platform creators believe in the importance of a great developer experience. In their own words, the platform is “easy to use, and easy to understand. No more complex code for doing simple things.”

The platform is hardware-agnostic. You can connect any device, the most typical being Arduino, ESP8266, Raspberry Pi, and Intel Edison. You simply install the server in your own cloud and use the open-source libraries for integrating the IoT devices.

Mainflux

This open-source IoT platform focuses on edge computing. Mainflux is patent-free and end-to-end, under an Apache 2.0 license, and covering most of the things needed for developing IoT solutions, applications, and products.
Users can benefit from the platform's full transparency, full user control over their own assets, as well as the possibility to do community testing, support, and bug fixes. As the platform was built as a set of containerized microservices orchestrated by Kubernetes, there is no vendor lock-in.

Mainflux provides the complete infrastructure and middleware to execute:

- Device management
- Data aggregation & data management
- Connectivity & message routing
- IoT application enablement
- Analytics

Arduino

Arduino is one of the best-known names in open-source IoT projects, encompassing both hardware and software. The Arduino Cloud IoT is a cloud solution to configure, program, and connect IoT devices using the Arduino IoT Cloud service. The Arduino software includes two types of integrated development environments (IDE1 and IDE2). A variety of boards, shields, and carriers make up the hardware palette.

The blend of IoT hardware and software makes Arduino an easy-to-implement, easy-to-use IoT platform that needs no further introduction.

Programming Languages Using IOT

JAVA

As far as IoT app development is concerned, JAVA has the most prominence in the market. In the year 2019, it was the most popular programming language with an overall rating of 16.61%. The programming language alone has powered close to three billion devices.

One of the major reasons why JAVA is prominent because of its Code Once Run Anywhere functionality. This means that developers can code their app once and run it on any device that is compatible with the programming language (from cell phones to the simplest of devices).

Besides, JAVA is object-oriented, which allows you to develop apps for both cloud and edge nodes. Its interoperable functionalities and availability of extensive libraries, make JAVA an ideal programming language for IoT development.

Python

Another in-demand programming language choice for IoT Product development, Python offers seamless code readability and simple syntax features. Being an interpreted language, it is highly compatible with object-oriented, structured, and functional programming.

Python can be integrated with other programming languages like Java and C++, and the language also works across diverse platforms including Linux and Windows. It's comprehensive library and solid community support make Python a developer's delight. For IoT apps that require extensive data analysis, Python is the ultimate choice.

LUA

We get it. LUA is not the programming language you thought would make it to the list, right? However, the features and functionalities it offers to an IoT ecosystem make it inevitable in the industry. For the uninitiated, LUA is a high-level and general-purpose programming language. It's major functionality caters to the niche of the embedded system. This means that the programming language can only function when embedded in host clients.

One of the standout features of LUA is its offering of frameworks like Node.lua. This framework allows developers to build IoT-specific apps and integrate IoT-centric functionalities that a system would need. This includes portability, battery efficiency, data management and more.

PHPoC

Developers who are familiar with PHP will experience a minimal learning curve with PHPoC. Abbreviated as PHP-on-Chip, this is an IoT hardware platform-based programming language. The syntax of this programming language is very much similar to that of PHP. Some of the IoT-specific functionalities offered by the programming language include UART, RTC, SPI, and more.

With core benefits like real-time debugging, prebuilt networking abilities, and airtight security features, PHPoC is ideal for the development of applications like cybersecurity, CMS, and image processing units.

C Language

Launched close to two decades back, the programming language is still as relevant as it gets. Amidst all the competition from the recent programming languages, C continues to be preferred by developers to build IoT applications.

The language offers flexibility to developers apart from offering features like interoperability, rich libraries, and portability. C is also super compatible with micro-controllers that are integral peripheral devices of IoT architectures.

However, one hurdle developers are most likely to face is the initial difficulty in learning the programming language. The syntax is quite complicated and its architecture is layered. Once this is overcome, there is no turning back for developers developing an IoT application.

Swift

Swift is very unique in this list. While other programming languages discussed earlier can be used to develop IoT applications for the web, smartphones, and devices across multiple platforms, Swift is a language specifically designed to develop IoT applications for iOS devices.

It falls in line with Apple’s design guidelines, allowing developers to build IoT apps that meet the company’s benchmarking standards. It’s fast, secure, and seamlessly handles errors and offers tons of programming patterns.

If your IoT app development goals only revolve around iOS devices, this is the most ideal programming language. As far as the learning curve is concerned, it’s simple syntax and ease of coding make it one of the easiest programming languages to master. With increasing prominence, aspiring IoT developers should get hands on with Swift as soon as possible.

JavaScript

Hailed in the world of web development, JavaScript shares its libraries with other programming languages. The use of JavaScript for IoT app development makes powering devices interoperable. One of the best advantages of JavaScript is that it works across diverse environments including gateways and the cloud.

The presence of an active developer community ensures that all your hiccups can be solved through direct responses, featured articles, tutorials, and more. With sensors being inevitable in IoT architecture, JavaScript's event-driven features make the environment it runs in more functional and efficient.

Go

Relatively a new programming language, it's quickly gaining prominence in the IoT app development space because of its range of features and benefits. To start with, Go offers optimized code that makes it perfect for projects that involve small computing devices with very limited power and memory resources.

Next, it is highly concurrent, meaning it can route thousands (or even millions) of data streams seamlessly and can run several asynchronous data streams simultaneously. Developers intending to get started with Go will face a minimal learning curve as the language is really intuitive and super-friendly.

Rust

Seasoned developers will know that the most preferred programming languages for IoT projects are C and C++. Thanks to their memory management and runtime capabilities, they are critically deployed across the world. However, at the other end of the spectrum exists a very similar programming language hailed as an alternative to the other two programming languages.

Called Rust, developers can experience all the benefits of C and C++ a little bit faster with the programming language. With airtight documentation, resourceful error messages, and solid compiler, Rust is an ideal programming language for both starters and veterans alike.

Wrapping Up

Apart from these, there are also programming languages like Ruby and ParaSail that offer unique benefits to IoT developers. To all the freshers and aspirants out there, get started with one of these languages and master it at your own pace. Understand how you can become a power developer with the language you master and become irreplaceable in the market.

API Management For IoT

API management for IoT

API is the core of IoT as it's what makes IoT devices use the internet for communicating and conducting allotted operations efficiently. Communication API in IoT plays an important role as it empowers IoT devices for seamless information exchange.

However, the API abundance can be a challenge for the organization as it takes a lot to enforce effective security policies for APIs. So, let's move about the Internet of Things API and key API security practices to adopt in this post.

securing apps on aws with wallarm

Sign up ->

Learning Objectives

What is Internet of Things?

What is an IoT API?

API Types in IoT

The important role of APIs in the IoT

How API management helps the IoT Developers?

API vendors for the Internet of Things

IoT API security

Subscribe for the latest news

Subscribe

What is Internet of Things?

Abbreviated as IoT, it is an inventive approach framed to let devices/applications connect or communicate using the internet for sharing precious data back and forth. It's a huge set-up of linked applications, resources, devices, and end-users that share information regarding their operations and surroundings.

The system functions on concepts of data analyses and automation while dealing effectively with sensors, cloud messaging, sensors, and AI. IoT devices are best known for their unmatched performance, upscale utility, and better control.

What is an IoT API?

In general, API is the term referring to standard framework collection, protocols, and resources dictating the generic web and mobile application. It defines the communication rules that every application component must follow while exchanging information with each other.

APIs that are used in the creation of IoT solutions are known as IoT APIs. They are the web services application programming interfaces. They work in a similar fashion and make seamless data flow, with HTTPbeing the medium. Using the IoT API lets developers design advanced applications that are easy to integrate with other web services.

IoT APIs make the end-user experience exceptional in multiple manners. For instance, they can log in to the websites easily and bring a good number of applications on board.

Internet of Things

API Types in IoT

APIs in IoT are highly diverse to support IoT development by all means. The key API types are as mentioned next.

SOAP

SOAP APIs are crucial for IoT devices development as they make building a communication bridge between the servers and the clients. The API supports only XML-based data transfer.

REST

IoT REST APIs useful for HTTP data transmission and for empowering IoT devices to stay associated with the rest of the world. These APIs are driven by architectural principles and boast features like interfaces simplicity, instant resources identification during the request, and manipulation of particular interfaces.

JSON and XML

A bit older than SOAP APIs, JSON and XML IoT APIs are based on a simple approach and consume limited bandwidth.

The important role of APIs in the IoT

What makes API crucial for IoT and any other device/software is the fact that they support effective utilization of pre-existing functions to ensure smooth software processing while keeping developers free from the need of reprogramming again and again.

The world of IoT is too complex demanding continual contact between multiple agents involved. API usage makes the task achievable as integrating assorted IoT components with each other is 100% possible.

API usage empowers the IoT world and makes innumerable as fresh development and integration opportunities are rendered.

IoT APIs serve as amazing technical development resources as unmatched flexibility is granted.

Speaking of cybersecurity, APIs are essential as developers can use them to gain control over access requests. This limited and controlled access keeps the DDoS attack possibilities less than expected.

Considering all this, it’s easy to conclude that the world of IoT will be on the brink of fall, with less productivity, and higher security risks if APIs are not involved.

How API management helps the IoT Developers?

IoT is a trending technology and is going to stay here for a little longer than one can expect. However, its effective utilization depends upon the degrees of API management as if that’s not happening, enterprises and end-users are going to have a tough time with IoT devices. Here is how:

In the case of IoT devices, efficient management and handling can empower developers to decide which API should be used and when should he or she revoke the access. Developers are allowed to set single or multiple connection criteria for apps and IoT devices.

The continual API monitoring is a crucial aspect in the process that further supports early detection of any unforeseen caveats and failure possibilities.

It presents a clear picture of API utility and usage for IoT.

The deeper insights into the developer experience with the API can be earned as one compares the standard experience and delivers an end-user experience to spot the gaps.

IoT development requiring emergency API utilization is fully supported as it grants details about API scaling, use quota, and throttling.

API version update is easier than ever with good management standards being maintained as new version updates are offered timely.

API benefits

API vendors for the Internet of Things

Seeing the surge in API usage in IoT, certain vendors are bearing the responsibility of offering tailor-made communication API that meets all the IoT API standards.

Withings API

With Withings APIs developers involved in the development of measurement devices can be benefitted hugely as the API can share the collected data over the internet. Most commonly, the collected information by this API vendor is ECG and EKG, body weight, and sleep cycles.

Garmin Health API

Perfect choice for developers involved in developing the IoT appliances operating in the health care and activity industry, Garmin Health APIs can monitor around 30 types of activities. Data related to total sleep hours, steps walked, stress level, heart rate, and many more.

Garmin Health API

Google Assistant API

This API is capable of being integrated into IoT devices easily and supports operations like voice control, natural language processing, hotword detection, and many other facilities. It guides developers through-and-through during the management and conversation stint. Using the API, developers can easily make IoT devices voice-controlled by phones, displays, watches, TV, laptops, and Google Home Devices.

Google Assistant API

Direct search about details like weather, traffic, news, light, task management, and many more other details can be made.

Apple HomeKit

The APIs coming from this vendor serves as a doable platform for connecting Siri and iPhone with the Apple-based home devices and appliances. Accessible with the help of Apple iOS8 SDK, the APIs can make devices like lights, garages, doors, TV, and many more to be controlled directly via voice.

IoT API security

The increased number of cyber-attacks and risks is a serious concern for anyone offering a service or product. IoT devices are also not out of the reach of threat actors. In fact, IoT devices of all applications bear the highest cyber risks as they operate using internet connectivity. If adequate and viable IoT API security practices are not practiced, multiple security risks will engulf the IoT devices.

Out of 80% of organizations that have used IoT devices in any form, 20% have already encountered cyber attacks, says Gartner’s recent report.

To stay safe from this danger, it’s crucial to back-up API with efficacious security practices.

Below mentioned methods can enhance security for the API.

API authentication must be activated from the beginning.

Robust API protocol with military-grade encryption and identity tokens should be deployed.

User and device authorization should be enforced.

Web API IoT should feature traffic throttling triggers to put a cap on DDoS attacks.

Web APIs should be regularly monitored and their use cases should be audited to spot any vulnerabilities in the infancy stage and develop a pro-active remedial action.

Web Services

Managing the development and administering of network-embedded systems is becoming complex, with the increasing number and diversity of connected components. In future, the size and the scope of Internet will increase as physical devices (for example, home appliances and sensors) are becoming smart, sharing and communicating data over Internet — creating an eco-system of Internet of Things (IoT).

With the adoption of IoT, the usage of Internet have surpassed beyond personal computers and smart phones. There are more than billions of embedded devices (sensors) in this connected world, and Gartner predicts 6.4 billion connected "Things" will be in use by 2016. Today, the key business priorities encompass improvement of customer experience, leveraging data and analytics for strategic decision making. Thereby, IoT helps in transforming by increasing the operational efficiencies and valued experience. However, the early adopters use IoT for revenue play, rather than infusing innovation.

This exponential growth opens windows for both possibilities and challenges; such as how to seamlessly to integrate constraint devices with the Web. Embedded environment opens scope for devices to not only become IP-enabled and interconnected on the Internet, but speak the same language. This will help communicate and interoperate over the Web.

Such developments expanded the scope of the Web Services domain. It provides many advantages for both application development and maintenance, such as a well-structured application consisting of different services implementing only a single aspect of the application. While developing software for the embedded domain, one had to deal with hardware interaction like reading a sensor.

Web developers have access to a plethora of technologies that increases the prowess to create best web-based applications in a short time span. Huge possibilities create a challenge for the developers, as they have to juggle while choosing specific approach for developing specific part of web applications, depending on the evolving standards or approaches.

SOAP and REST

To support my thoughts, I provide an example of two approaches that are mistakenly termed as "Web Services," Representational State Transfer (REST) and Simple Object Access Protocol (SOAP). They are interchangeably used, though being different.

SOAP

SOAP is defined as a standard communication protocol for exchanging data between two endpoints. It relies exclusively on XML for messaging services in three ways. The Envelop that defines the "what is in the message" and "how to process the message;" a set of coding rules for datatypes; the layout of the remote procedure calls and responses gathered.

It uses standard protocols like HTTP and SMTP. Originally, Microsoft developed SOAP to substitute older technologies like Distributed Component Object Model (DCOM) and Common Object Request Broker Architecture (CORBA). SOAP supports expansions such as WS-Security, WS-Addressing,

WS-Policy, and so forth. Although SOAP is highly extensible, the developers only use pieces of it for specific tasks.

SOAP is intolerant to errors. This is a restriction as in some programming languages the developers need to build requests manually, which is time consuming. However, SOAP has a significant feature "built in error handling;" so if there is a problem with the request, there will be a response about the error information that can be used to fix the problem. It can be automatized when used with certain language products. Table 1 summarizes the benefits and limitations of SOAP.

REST

REST is a comparatively new approach that sought to fix the limitations of SOAP. It provides a simple method of accessing Web Services. REST is a lightweight alternative that relies on URL in many cases. REST is an architectural style to build client-server applications. It has been applied to describe the desired web architecture. It doesn't contain additional messaging layer and focusses on design rules for creating stateless services.

REST is often used in the mobile applications, automated business processes, and social networking websites. Fundamental REST principles are as follows:

Client-Server Communication: All applications developed in this approach must also be client-server in principle.

Cacheable: Data must be marked as cacheable to be reused as the response to the same subsequent request.

Uniform Interface: All components must interact through a single uniform interface. The implementation of change can be made in isolation.

The REST approach is generally characterized by creating a network of resources that can be accessed by HTTP methods. It requires the developers to use HTTP methods explicitly and that should be consistent with the protocol definition. Table 2 of this document best describes the benefits and limitations of REST.

SOAP versus REST

In the software development community, the developers try to establish the fact that one approach is better than the other, but this statement is incorrect! Each approach has definite advantages and problematic grey areas. One needs to choose between SOAP and REST based on the programming language, the environment in which it is used, and the requirements of the application. To avoid a tight rope walk later one has to consider a comparative study of SOAP and REST as in Table 3.

Reality

The inference of this exercise states that both SOAP and REST are suitable for IoT. SOAP fits better for the requirements of business applications; while REST is suited for IoT applications, comprising mobile and embedded devices. In future, we will witness the growth of REST as IT vendors seek to provide open and well-defined interfaces for application and infrastructure services. The growth of enterprise hybrid cloud, besides public and private cloud is the driving factors for Web Services. The web services will act as a catalyst for extending the Internet to the unconnected environments; thereby making IoT a reality.

INTEGRATION OF SENSORS AND ACUTATORS WITH RASPERRY PI/ARUDINO/OTHER BOARDS

Raspberry Pi Humidity Sensor using the DHT22

In this Raspberry Pi humidity sensor tutorial, we will show you how to connect the DHT22 sensor to the Raspberry Pi and how you can use Python to read data from the sensor.

The DHT22 is a versatile and low-cost humidity sensor that can also calculate the temperature of an area. This sensor has a relatively long transmission distance, allowing the sensor to transmit data through wires up to 20m away from the Raspberry Pi.

As a bonus, the DHT22 is a digital sensor with an inbuilt analog to digital converter. The converter makes it a lot easier to connect the sensor to the Raspberry Pi as you do not need to deal with any additional chips.

The biggest downside to the DHT11 and DHT22 sensors is that they are quite slow sensors. They have a sampling rate of once every second for the DHT11 and once every 2 seconds for the DHT22.

EQUIPMENTS

Raspberry Pi

8GB SD Card or Micro SD Card if you're using a Raspberry Pi 2, 3 or B+

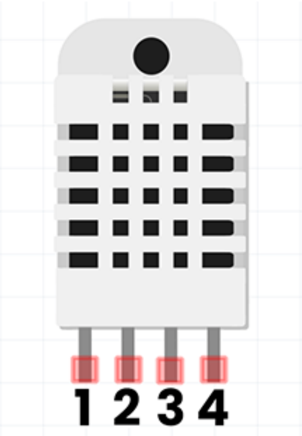
Breadboard wire

DHT22 Humidity Sensor

10k ohm resistor (Brown, Black, Orange, Gold)

In this section of the tutorial, we will show you the process on how to connect your DHT22 humidity sensor to the Raspberry Pi.
Thanks to the DHT22 being a digital sensor, it is incredibly straightforward to connect to the Raspberry Pi. The single data pin is able to connect directly to the Raspberry Pi's GPIO pins. You can follow our guide below to see how to connect the DHT22 to your Raspberry Pi.
To make things easier when assembling the humidity sensor circuit we have included the pinout of the DHT22 sensor.

This diagram should help you work out where each pin needs to go on the Raspberry Pi



Pin 1 is VCC (Power Supply)

Pin 2 is DATA (The data signal)

Pin 3 is NULL (Do not connect)

Pin 4 is GND (Ground)

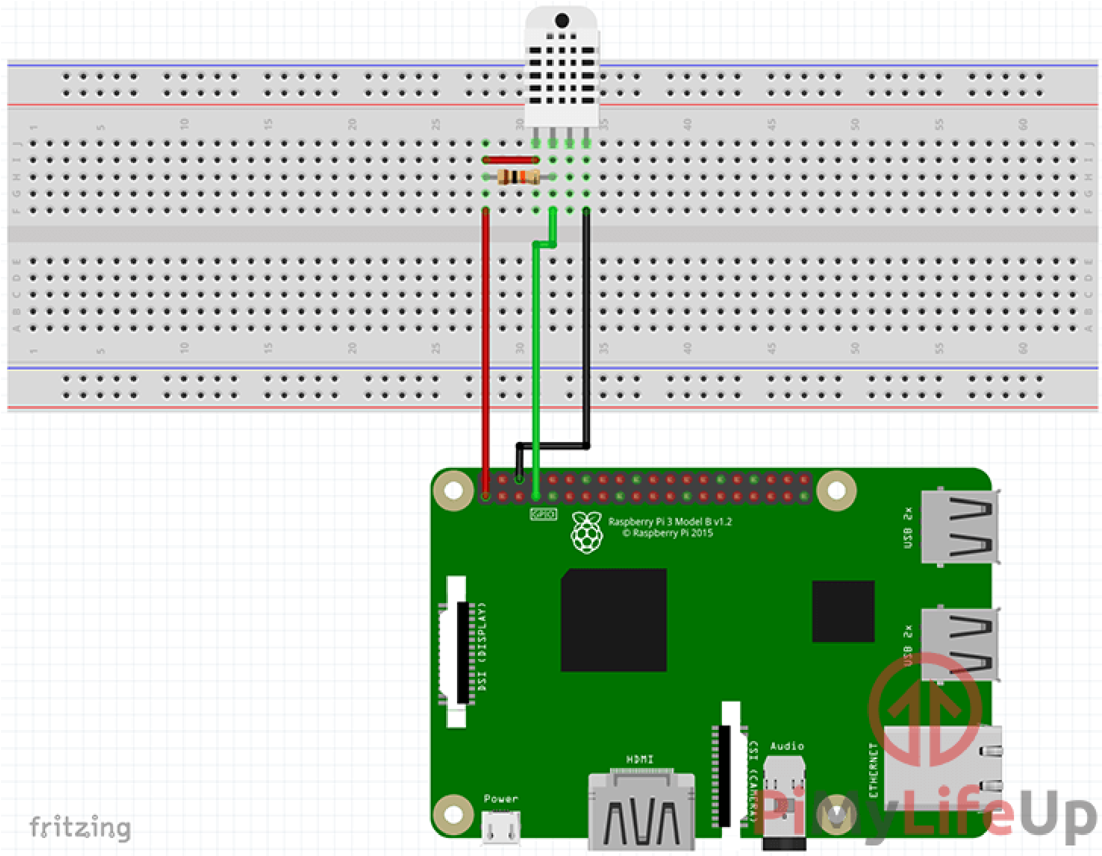
You can either rely on our two diagrams below to see what pins need to go to what or use our written steps below.

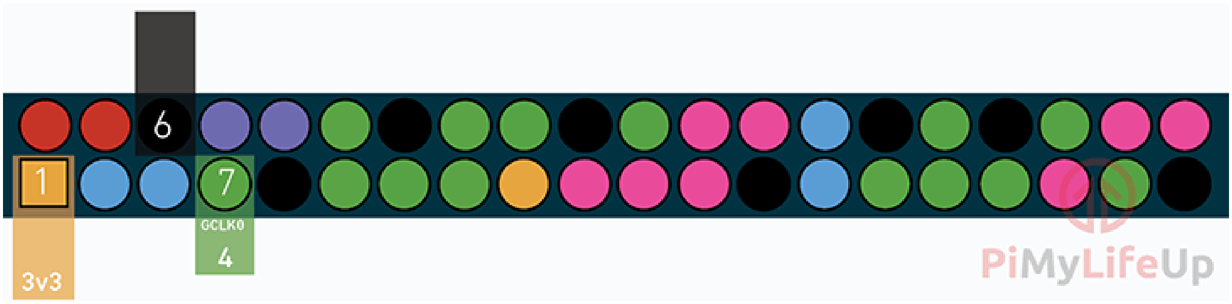
Place a 10k resistor between Pin 1 and Pin 2 of the DHT22

Wire Pin 1 of the DHT22 to Physical Pin 1 (3v3) on the Pi

Wire Pin 2 of the DHT22 to Physical Pin 7 (GPIO4) on the Pi

Wire Pin 4 of the DHT22 to Physical Pin 6 (GND) on the Pi





Programming for the Raspberry Pi humidity sensor

```
import Adafruit_DHT
DHT_SENSOR = Adafruit_DHT.DHT22
DHT_PIN = 4
while True:
   humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
   if humidity is not None and temperature is not None:
      print("Temp={0:0.1f}*C  Humidity={1:0.1f}%".format(temperature, humidity))
   else:
      print("Failed to retrieve data from humidity sensor")
```