Digi-Key IoT Studio Mini Smart Home

Created by Brent Rubell



https://learn.adafruit.com/digikey-iot-studio-smart-home

Last updated on 2021-11-15 07:52:58 PM EST

Table of Contents

Overview 5

- Digi-Key IoT Studio IDE 6
- Digi-Key IoT Studio Mobile App 6
- Modular Smart Home Hub 6
- Parts 6
- Tools and Materials 9

Reviewing the Internet of Things 9

- Transports 10
- Protocols 11
- Services 12
- Adafruit IO 13
- Security 14

What is Digi-Key IoT Studio? 14

- Developing IoT Projects, Visually 14

Wiring 16

- BME680 Wiring 17
- Adding a Buzzer to the Quad FeatherWing 17
- Mini Relay FeatherWing Wiring 19
- Attaching Female JST Connectors to Quad FeatherWing 19

Assembly 21

- Assembling the Door 22
- Assembling the Fan 25
- Adding LEDs 32
- Assembling the Smart Home Hub 34
- Add a hole for the USB Cable 35

How Digi-Key IoT Studio Works 36

- Embedded Workspace 37
- Application Workspace 38
- Cloud Workspace 38

Getting Started 39

- Install DK IoT Studio Mobile App 39
- Install IoT Agent 39
- Creating a Digi-Key IoT Studio Project 40

Turning on the Lights 41

- Step-by-Step: Embedded Workspace 41
- Step-by-Step: Application Workspace 44
- Step-by-Step: Uploading Firmware 49
- Step-by-Step: DK IoT Studio Mobile App 52

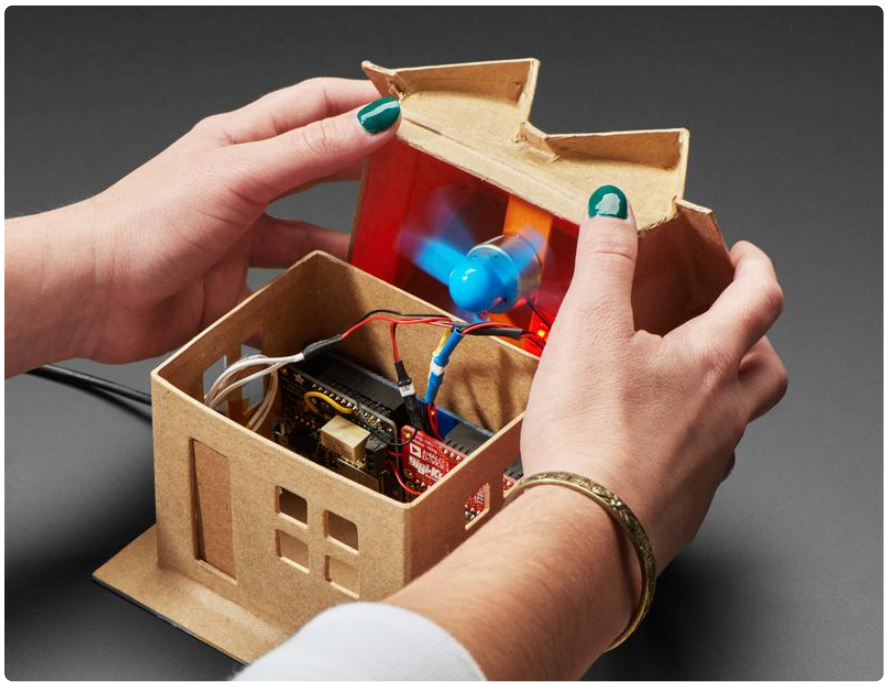
Monitoring your Home 54

- Step-by-Step: Embedded Workspace 55
- Step-by-Step: Application Workspace 66
- Step-by-Step: Cloud Workspace 75
- Code Usage 78

Keeping it Cool 80
• Step-by-Step: Embedded Workspace 80
• Code Usage 86
Door Alarm 88
• Step-by-Step: Embedded Workspace 89
• Step-by-Step: Application Workspace 94
• Code Usage 99
Dashboards 100
• Step-by-Step: Building a Dashboard 101
Conclusion 110

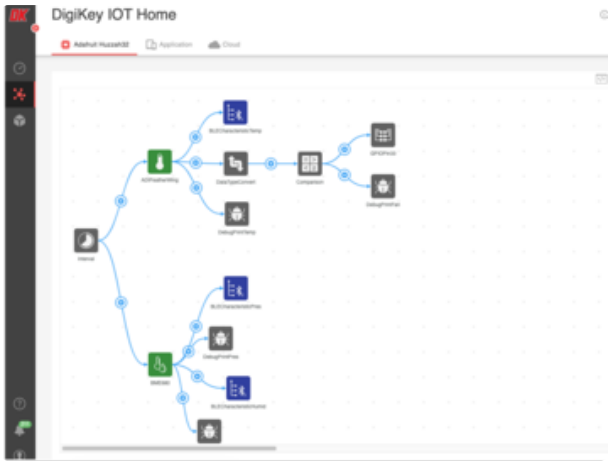
Overview

Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. This guide is no longer supported by Adafruit. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463



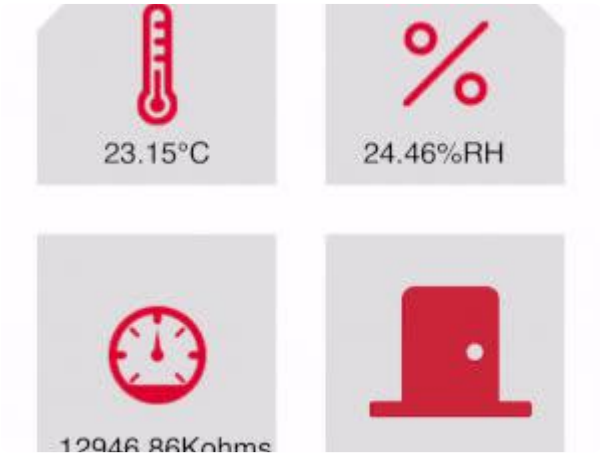
You'll be automating an adorable paper maché house. The small size of this build lets us explore wiring, user interaction, and firmware deployment without having to get a ladder out.

After building this project, you can re-purpose this project for your home or apartment. We've specifically selected components and sensors which are common in real-world IoT projects. You can also go further with this project, adding sensors to monitor different rooms in your house.



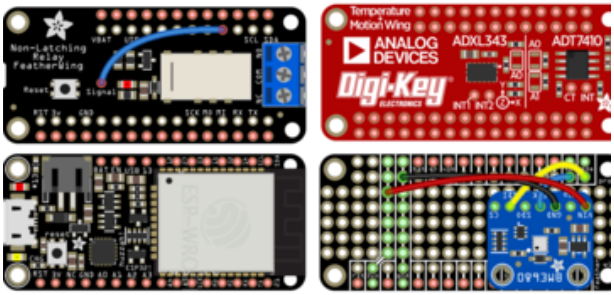
Digi-Key IoT Studio IDE

Unlike most other IoT platforms, Digi-Key IoT Studio (https://adafru.it/HlA) is code-less - all sensor interaction and storage is done automatically by the drag-and-drop IDE. You do not need to install toolchains, code editors or compile any code on your computer! Digi-Key Studio runs in your web browser, handles all aspects of developing an Internet-of-Things project, and even compiles your code remotely.



Digi-Key IoT Studio Mobile App

You'll build and program a mobile application to monitor and interact with your IoT Home over the Bluetooth Low Energy protocol. The Digi-Key IoT Studio Mobile App also acts as a bridge, sending your home's data to the free Digi-Key IoT Studio Cloud for long-term storage.

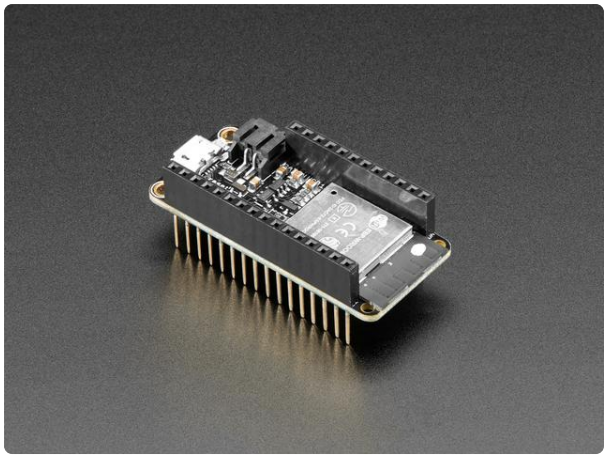


Modular Smart Home Hub

You'll start by building a hub for your smart home. We designed a modular smart-home hub which uses the Feather form factor. If you want to add more sensors or actuators onto your smart home hub, just snap on a new FeatherWing!

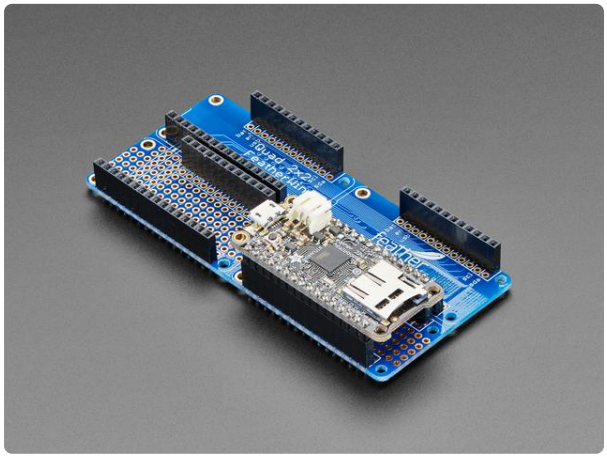
Parts

This tutorial uses the ESP32 Feather as the microcontroller brains, it can use both secure WiFi and Bluetooth LE as transports.



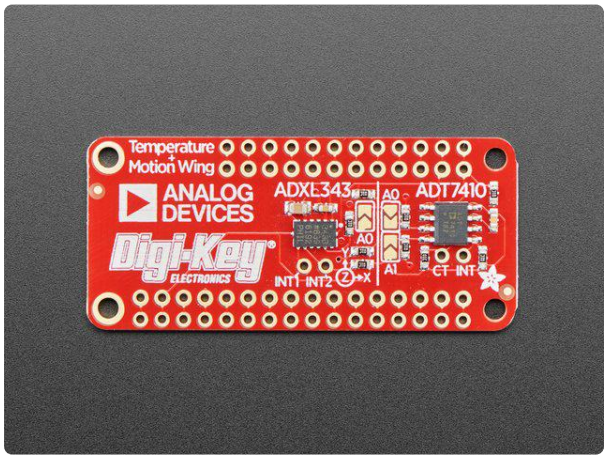
[Assembled Adafruit HUZZAH32 – ESP32 Feather Board]
Aww yeah, it's the Feather you have been waiting for, and now with stacking headers! The HUZZAH32 is our ESP32-based Feather, made with the...
https://www.adafruit.com/product/3619

To build the smart home's hub, you'll use a Quad 2x2 FeatherWing Kit. This kit contains four identical sets of headers for Feathers/FeatherWings and two prototyping areas.



[Adafruit Quad 2x2 FeatherWing Kit with Headers]
This is the FeatherWing Quad 2x2 - a prototyping add-on and more for all Feather boards. This is similar to our
https://www.adafruit.com/product/4253

For inputs, you'll use the Analog Devices Featherwing to obtain precision temperature readings and the Bosch BME680 to obtain humidity, barometric pressure, and air quality.



[Adafruit ADXL343 + ADT7410 Sensor FeatherWing]
Upgrade any Feather board with motion and precision temperature sensing, with this all-in-one sensing FeatherWing. It sports two fantastic sensors from Analog Devices: an ADXL343...
https://www.adafruit.com/product/4147

1 x Adafruit BME680 https://www.adafruit.com/product/3660
Adafruit BME680 - Temperature, Humidity, Pressure and Gas Sensor

---

We'll add a magnetic door sensor to detect when someone opens a door or window a nd add a buzzer to make a a loud 2KHz BEEEEEEEEEP when someone opens the door.

1 x Door Sensor https://www.adafruit.com/product/375
Magnetic contact switch (door sensor)

---

1 x Buzzer https://www.adafruit.com/product/1536
Buzzer 5V - Breadboard friendly

For outputs you'll use a Non-Latching Mini Relay FeatherWing to control a DC motor - that's our way of simulating an HVAC unit that turns on when it gets too hot or too cold.

You'll also connect some LED lights to a GPIO pin on the HUZZAH32, so you can turn on or off the lights from the DK IoT Mobile App, or when someone opens the door.

1 x Mini Relay FeatherWing https://www.adafruit.com/product/2895
Adafruit Non-Latching Mini Relay FeatherWing

---

1 x LED Sequin Pack https://www.adafruit.com/product/1755
Adafruit LED Sequins - Ruby Red - Pack of 5

---

1 x Spindle Motor (for fan) https://www.adafruit.com/product/3882
CD DVD Spindle Motor

---

1 x Trifoil Fan https://www.adafruit.com/product/3896
3-Bladed Trifoil Propeller Fan for DC Motor

You'll need a few JST-PH cables for this build. These cables will allow you to easily remove sensors connected to the IoT Home Hub from the enclosure with a nice click.

3 x JST-PH Extension Cable https://www.adafruit.com/product/1131
JST-PH Battery Extension Cable - 500mm

Tools and Materials

You also may want to pick up the following materials and tools to complete this project.

1 x USB Cable
USB cable - USB A to Micro-B - 3 foot long
https://www.adafruit.com/product/592

1 x Heat Shrink Pack, Assorted Diameters
Multi-Colored Heat Shrink Pack - 3/32" + 1/8" + 3/16" Diameters
https://www.adafruit.com/product/1649

1 x 22AWG Hook-up wire
Hook-up Wire Spool Set - 22AWG Stranded-Core - 6 x 25ft
https://www.adafruit.com/product/3111

1 x Wire Cutter/Strippers
Hakko Professional Quality 20-30 AWG Wire Strippers
https://www.adafruit.com/product/527

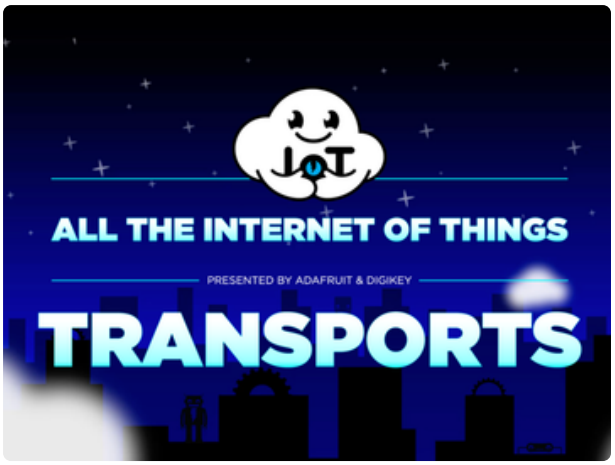
1 x Solder
Mini Solder spool - 60/40 lead rosin-core solder 0.031" diameter - 100g
https://www.adafruit.com/product/145

1 x Soldering Station
Hakko FX-888D
https://www.adafruit.com/product/1204

1 x Craft Knife
Slice Craft Knife with Ceramic Blade
https://www.adafruit.com/product/4306

Reviewing the Internet of Things

Over the videos, we have built up our knowledge, starting from the very lowest levels.



Transports

We started out with Transports where we analyzed the different ways to move bits of data around.

We talked about small-hop transports like Bluetooth Classic and Low Energy (https://adafru.it/Dik), ZigBee (https://adafru.it/Ive) and Z-Wave (https://adafru.it/Ive), and their use in small scale area networks, such as within a home.

We also covered longer range transports like WiFi (https://adafru.it/HIF), SigFox (https://adafru.it/Ivf) and LoRa (https://adafru.it/Ivf) that can cover a building.

Finally, long-distance protocols like Cellular or satellite (https://adafru.it/Ivf) can connect you anywhere in the world.

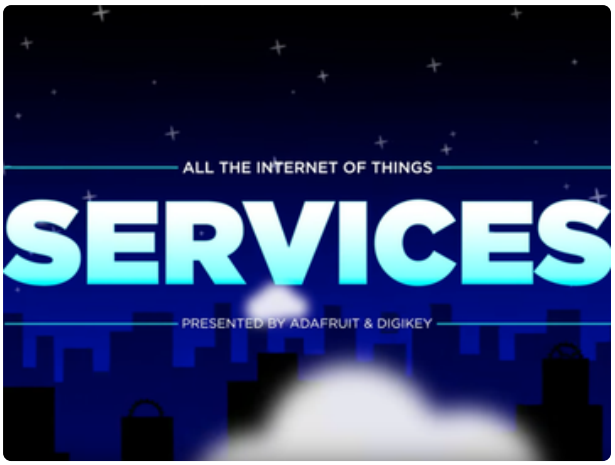
Each transport has different power needs, and bandwidth capabilities. Some are handy in that they connect directly to the Internet. Some need gateways to access the Internet, which can add cost and complexity. Transports directly affect your bill of materials and chipset selections, and it's very hard to adapt a product to use a new or different transport, and transports don't have a ton of overlap in their capabilities, so that's why it's the first thing to pick.



Protocols

Next, we dove into Protocols, which is how we program the Transport we've already decided on (https://adafru.it/lvA). This episode was a super-deep exploration of different transport protocols like HTTP (https://adafru.it/Evy), MQTT (https://adafru.it/Fmp), CoAP (https://adafru.it/lvB) as well as encoding methods such as XML (https://adafru.it/lvC), JSON (https://adafru.it/lvD) and base64 raw data.

Which one you use depends on your transport speed, bandwidth capabilities, and data responsiveness needs - HTTP gives you a direct connection to Internet services, but with a heavy price in data usage (https://adafru.it/Evy). MQTT is light and nimble, with little overhead data-wise, but requires a gateway to get you the data you want to the greater internet (https://adafru.it/Fmp). Usually there's a little more flexibility in what Protocols and encoding methods you get to use, and a good decision here can save you money and support pain later.



Services

Once we sorted out our hardware transport and our firmware protocol, we moved up the stack to the software Services. Just like protocols encode the data that moves over the lowest-level protocols, services package up and direct that encoded data. Services can also store data for later analysis (https://adafru.it/lvE), or provide meta-services (https://adafru.it/lvF) like notifying you by text-message (https://adafru.it/lvF) when no data has been reported for 24 hours. You could roll your own, but it's a huge effort and there's easily a hundred services available for any sort of IoT usage.

We covered some of the most popular ones, from Plotly (https://adafru.it/lwa) to Carriots (https://adafru.it/lwb), Particle (https://adafru.it/HIE) to AWS (https://adafru.it/lwc). Some services are prototyper-friendly, or are designed with scientists in mind, others are designed for enterprise-level deployments. We went through examples of using each service so you can find the right match for the complexity of your product (https://adafru.it/lwd).



Adafruit IO

After touring all the different services available, we took a deep dive into our own IoT service, (https://adafru.it/lwe)Adafruit IO (https://adafru.it/lwe). While we focused on just one service, there's a lot of similarities between Adafruit IO and other services, so even if you end up with a different IoT provider, it's beneficial to see how the pieces fit together.

This was also the first video where we provided code and schematics to build a complete project for temperature logging. We used WiFi as our transport (https://adafru.it/lwf), since its the most popular transport for Internet connectivity. The projects were shown with both Arduino/C microcontroller programming on a Feather ESP8266 (https://adafru.it/n6A) and in Python on a Raspberry Pi single board computer (https://adafru.it/lwA). The Arduino code utilized MQTT as a protocol, (https://adafru.it/lwB) and the Raspberry Pi used REST (https://adafru.it/lwB), showing you can get similar results with both low-power/low-level microcontrollers and higher-power/high-level microcomputers.

Together, we went through each step of signing up for an account (https://adafru.it/lwC), making a new feed (https://adafru.it/lwD), creating a dashboard (https://adafru.it/lwE), and plotting output data. Then, we explored extensions like webhooks (https://adafru.it/lwF), triggers (https://adafru.it/lxa), and integrations (https://adafru.it/lxb).



Security

After building our first project and connecting it to Adafruit IO, it was time to cover Security (https://adafru.it/lxc), a wide-ranging but important topic that has thankfully gotten more attention in recent years (https://adafru.it/lxc). Security practices are never-ending in complexity, and even as we write, new IoT security alerts and warnings are being published. Instead of looking at specific exploits or attacks, we categorized different ways that you can protect and harden your attack surface (https://adafru.it/lxd). Some of these (https://adafru.it/lxe)techniques are super simple, like having unique, un-guessable passwords. Some are more advanced, like requiring bi-directional TLS certificate checks and hardened crypto-storage.

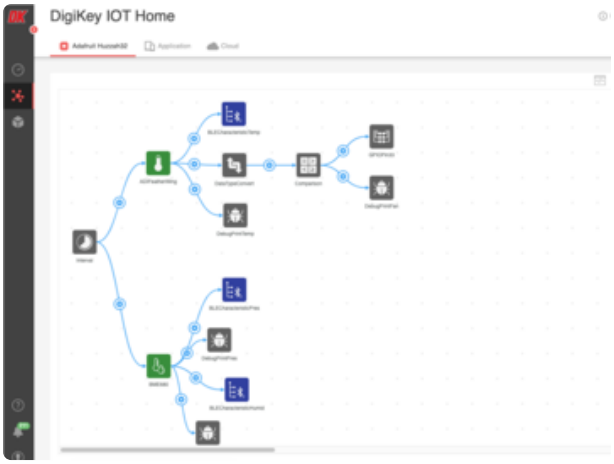
Either way, following the checklist we provided in the video will do a lot to protect your product, data and customers from prying eyes (https://adafru.it/lxe).

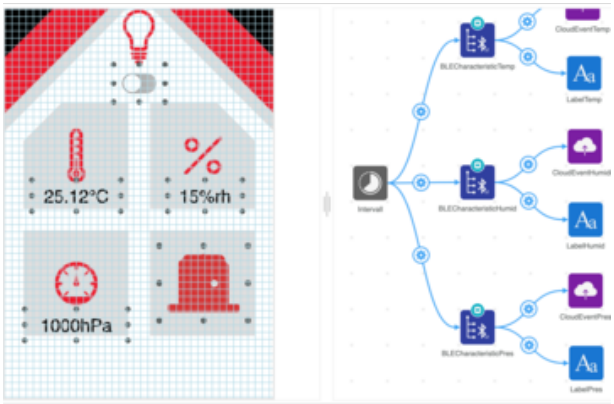
Finally, we've reached this final video - where we will explore using Digi-Key IoT Studio, an all-in-one solution for creating a complete IoT prototype or project.

What is Digi-Key IoT Studio?

Developing IoT Projects, Visually

Unlike most other IoT platforms, Digi-Key IoT Studio (https://adafru.it/HIA) is code-less - all sensor interaction and storage is done automatically by the drag-and-drop IDE. You do not need to install toolchains, code editors or compile any code on your computer! Digi-Key Studio runs in your web browser, handles all aspects of developing an Internet-of-Things project, and even compiles your code remotely.

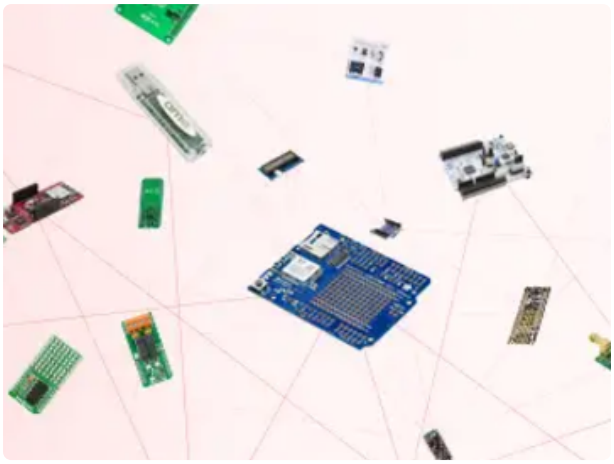






IoT projects involve programming firmware into a device, sending data from a device to "the cloud" and then interacting with the data and device using a website or application. Digi-Key IoT Studio comprises of three workspaces which simplify development:

- Embedded Workspace to build logic which runs on your device.
- Application Workspace to design, build, and program an application for your device.
- Cloud Workspace to construct flows for sending data to the free DK IoT Studio Cloud or Amazon Web Service.

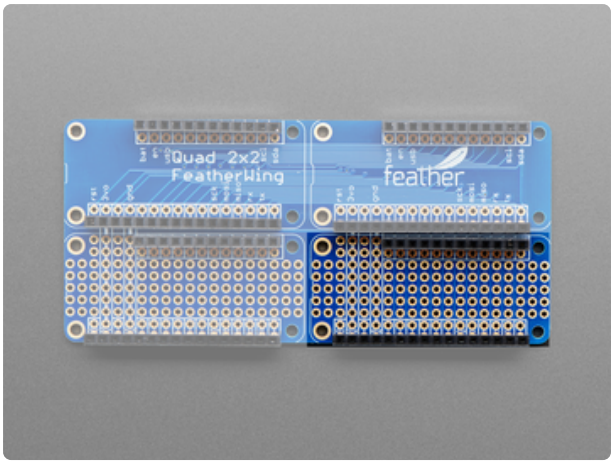


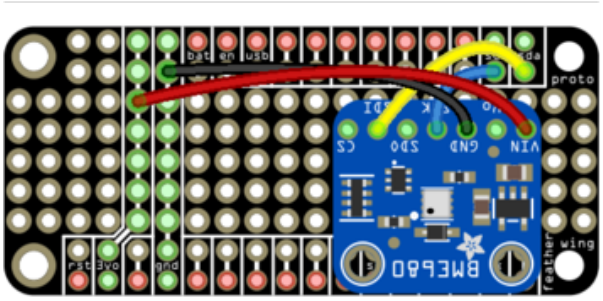
This high-level integration is possible because IoT Studio has built in support for a range of transport boards, and sensors (https://adafru.it/HlB). Note that not all boards and devices are supported - lucky for us, there's plenty of Adafruit products supported, including our series favorites: the ESP32 Feather (https://adafru.it/wcN) and the Analog Devices FeatherWing (https://adafru.it/HlC).

However, instead of just a simple temperature sensor data logger, we'll make a complete home automation system that you can follow along by picking up our IoT Studio Kit.

Wiring

The Smart Home Kit for Digi-Key IoT Studio - Feather ESP32 kit has been discontinued by Adafruit. If you make your own similar project, you will have to buy the parts separately.





BME680 Wiring

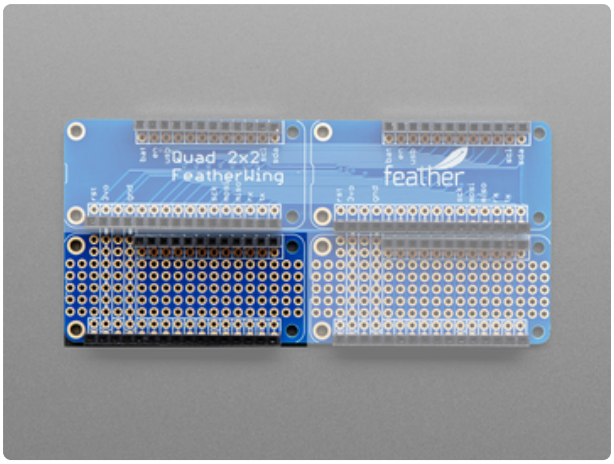
You'll be soldering and wiring the BME680 directly on the prototyping grid on the bottom right side of the Quad 2x2 FeatherWing.

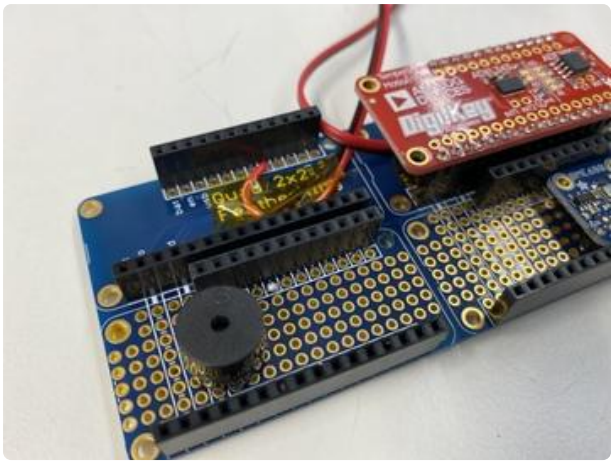
Make the following connections between the FeatherWing's prototyping grid and the BME680:

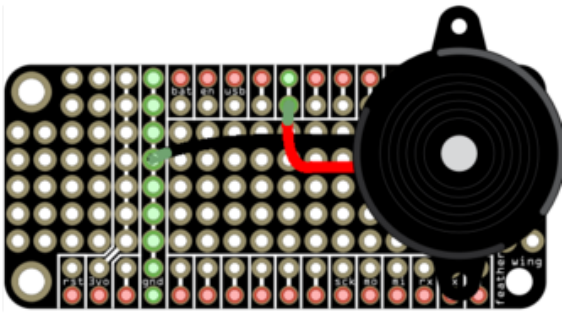
- FeatherWing 3V to sensor VIN
- FeatherWing GND to sensor GND
- FeatherWing SCL to sensor SCK
- FeatherWing SDA to sensor SDI

Adding a Buzzer to the Quad FeatherWing

There's another prototyping grid on the bottom left of the Quad 2x2 FeatherWing. You will be adding the buzzer to this grid.



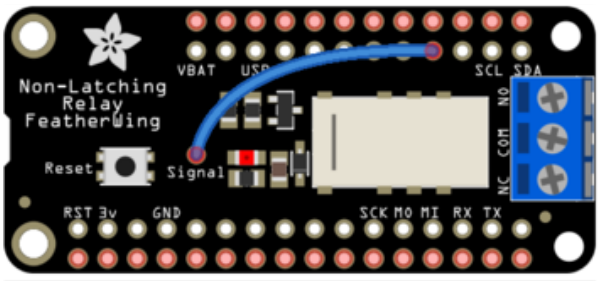




Place the buzzer on the prototyping grid. Make sure the buzzer's negative (short) lead is placed on the ground rail. Also, ensure the stacking headers will be able to fit besides the buzzer.

Solder the buzzer's negative (short) lead to the ground rail.

Solder a wire between GPIO Pin 12 and the positive (long) lead from the buzzer.



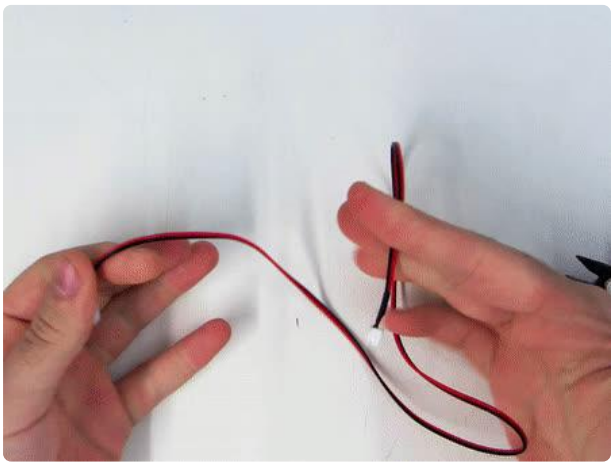
Mini Relay FeatherWing Wiring

Solder the 3-Pin terminal block to the the Non-Latching Mini Relay FeatherWing.

Solder a wire between the Signal pin and header pin 33.

Then, solder a red wire to the 3V header on the Non-Latching Mini Relay FeatherWing. Connect this wire to the NC pin on the 3-pin terminal.

Attaching Female JST Connectors to Quad FeatherWing



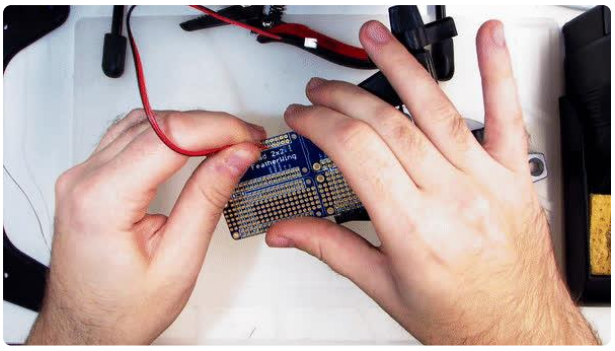


You'll use female JST connectors to connect the IoT Hub's base to the fan, LEDs, and door sensor.

Start by cutting a JST-PH cable in two. Make sure there's an even length of cable between both connectors.

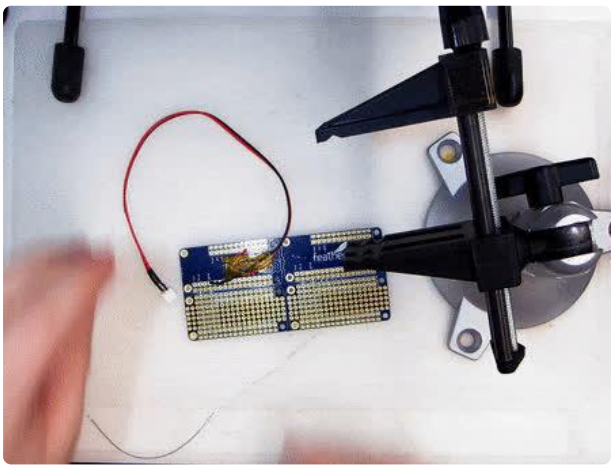
Then, strip off 5mm from the red and black cables.

You may want to tin ends of the cables with a bit of solder at this point.



On the top left side of the Quad FeatherWing, make the following connections between the JST-PH connector and the header pins on the FeatherWing:

- JST-PH Red to FeatherWing GPIO 15
- JST-PH Black to FeatherWing GND

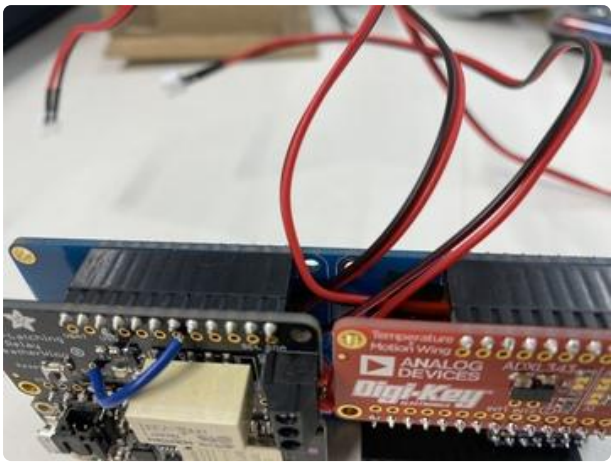


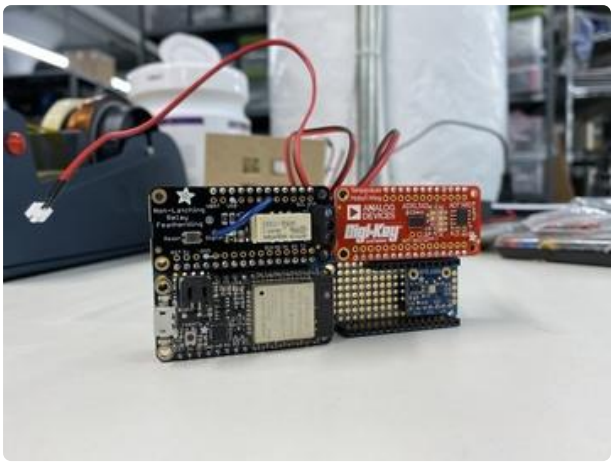
Just like before, cut and strip another JST-PH connector.

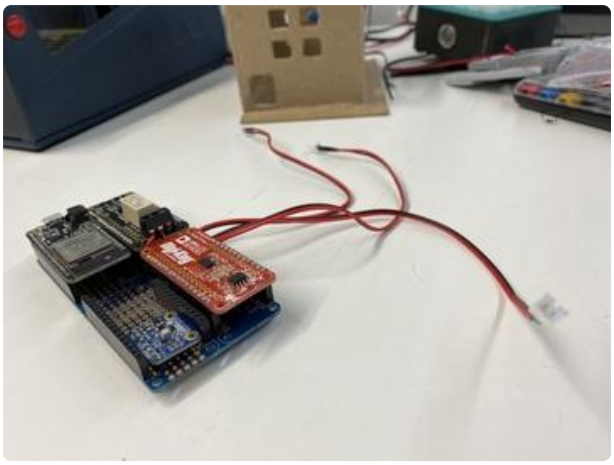
On the top right side of the Quad FeatherWing, make the following connections between the JST-PH connector and the header pins on the FeatherWing:

- JST-PH Red to FeatherWing GPIO 12
- JST-PH Black to FeatherWing GND

Solder all four sets of Feather headers to the Quad FeatherWing.







Before placing the Feather Huzzah32 and FeatherWings on the Quad FeatherWing, make sure that all the female JST connectors are routed through the middle and out the top of the Quad FeatherWing.

You may want to use tape to secure the cables to the FeatherWing's PCB.

Then, connect both FeatherWings and the Feather Huzzah32 to the Quad FeatherWing.

You've successfully built the IoT Home's Hub! Let's move on to building the rest of the IoT Home.

Assembly

The Smart Home Kit for Digi-Key IoT Studio - Feather ESP32 kit has been discontinued by Adafruit. If you make your own similar project, you will have to buy the parts separately.

Assembling the Door

Is anybody home? To detect when the door is either open or closed, you'll be using a reed switch (https://adafru.it/uF3). These super simple sensors operate on the principle that when a magnet is less than 0.5" away, the reed switch internally closes. We'll detect this change using a HUZZAH32 and send the door's status to the Digi-Key IoT Studio Project.

These switches aren't just for a cardboard home, they've been used in real-world projects such as the Adafruit IO Door Detector (https://adafru.it/jdq) project.



Using a hot glue gun, glue the reed switch to the left side of the door-frame.

If you don't have a hot glue gun handy, you can substitute a piece of double-sided tape instead.

You will want to ensure the switch is firmly in place and does not move around any.

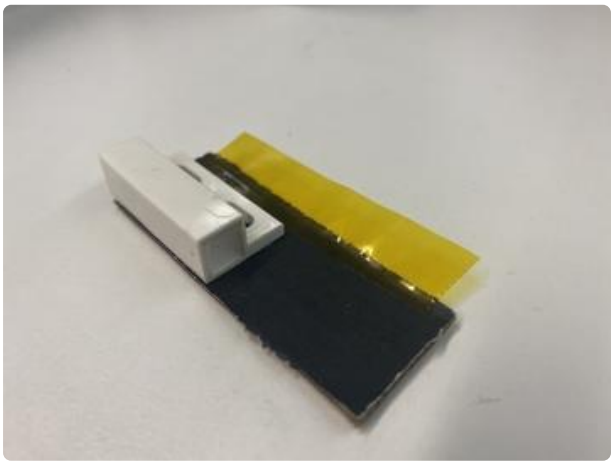


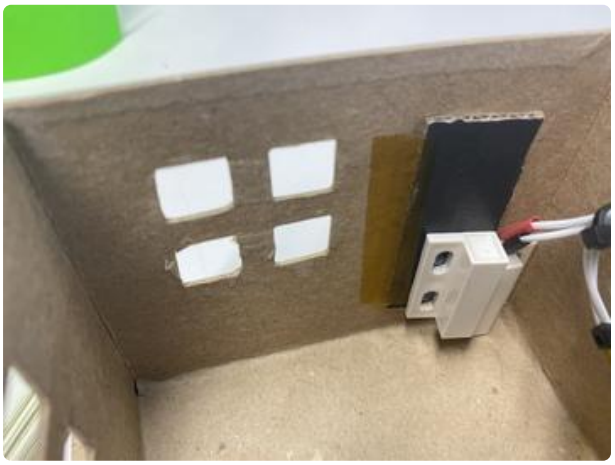
Next, grab the closest empty Adafruit/ Amazon/Digi-Key box (We know you have some around, we do!).

Using a ruler, sketch a door on a this piece of cardboard. Then, cut it out using a pair of scissors or a knife.



Using double sided tape or hot glue, affix the magnet to the door.





There's many ways to assemble a door hinge, we'll use tape to act as a hinge. Place a strip of tape on the door's edge and line it up with the doorway in the house. You should be able to open and close the door without it falling off.

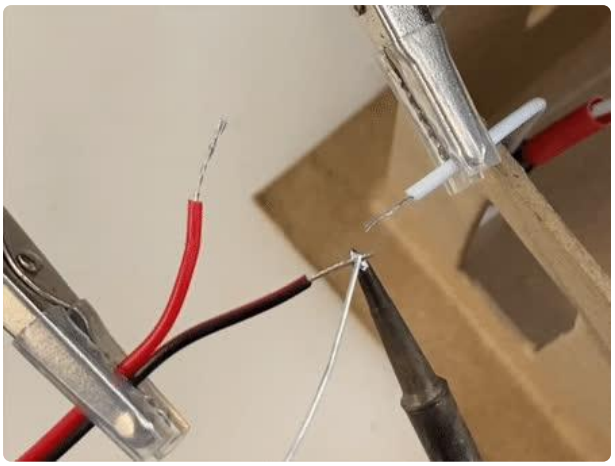
For a more permanent option, you may want to use a bamboo skewer like we did in the IO Home: Security Learning System Guide (https://adafru.it/HID)





Strip 3mm of the rubber coating from the door sensor cable. Then, strip 3mm from the male JST header cable.

Slide two narrow lengths of heat shrink down each cable for the door sensor. Then, slide a wider length of heat shrink over both cables.







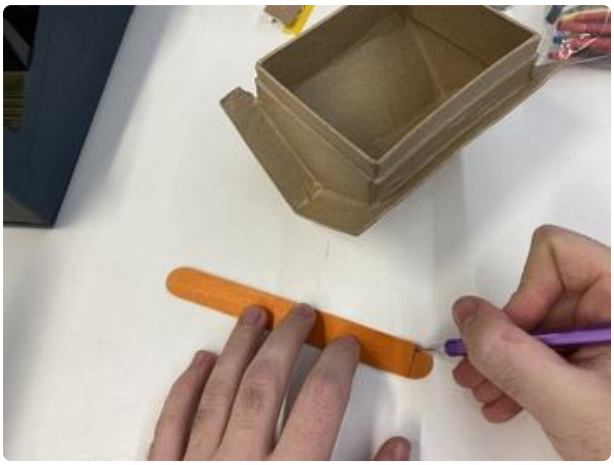
Using a soldering iron, tin the ends of both cables. Then apply heat with the soldering iron to connect them together.

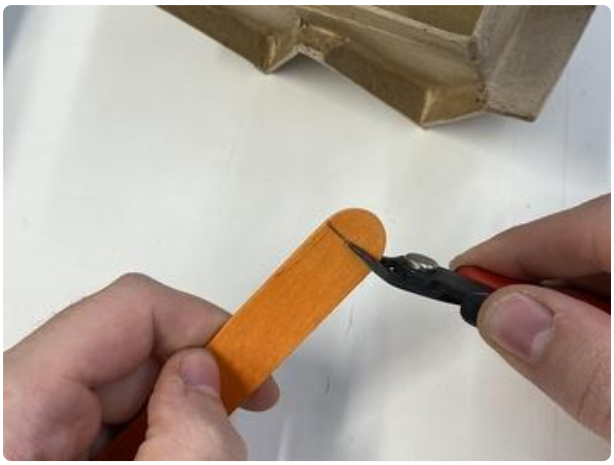
Once connected, slide the heat shrink over the soldering joint and apply heat.

Repeat this for both cables (red and black). Once you're done, slide the larger piece of heat shrink over both of the cables and apply heat.

Assembling the Fan

In order to attach the DC spindle motor to the roof of the home, you'll build a rafter-like-support using a popsicle stick.







With a pencil, mark a line 0.5" from the curved portion of the popsicle stick.

Using wire snips or scissors, cut along the line you marked. To avoid splitting the wood, take small "snips" instead of a large cut.

After cutting, you may want to use a file or some coarse sandpaper to sand down uneven edges.







- Insert one end of the popsicle stick in the inside corner of the roof.
- With a pencil, mark the popsicle stick at where the inside roof ends.
- Using wire snips or scissors, cut along the line you marked. To avoid splitting the wood, take small "snips" instead of a large cut.
- After cutting, you may want to use a file or some coarse sandpaper to sand down uneven edges.
- Test-fit the popsicle stick into the roof.



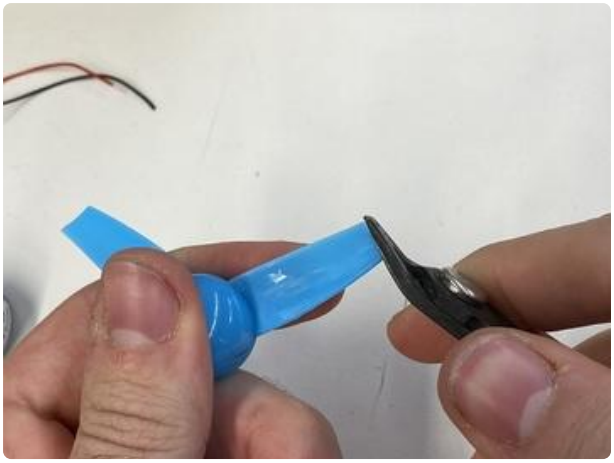


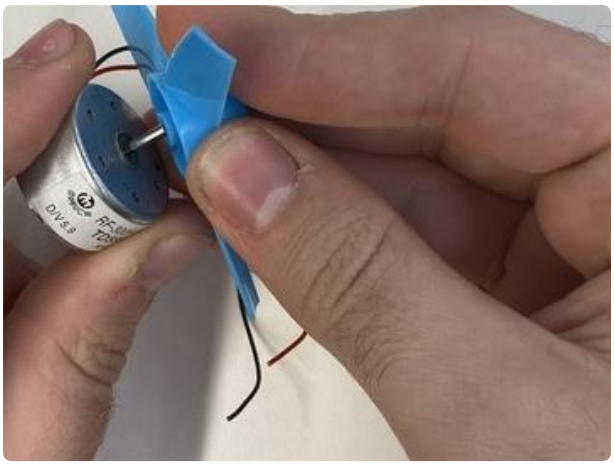


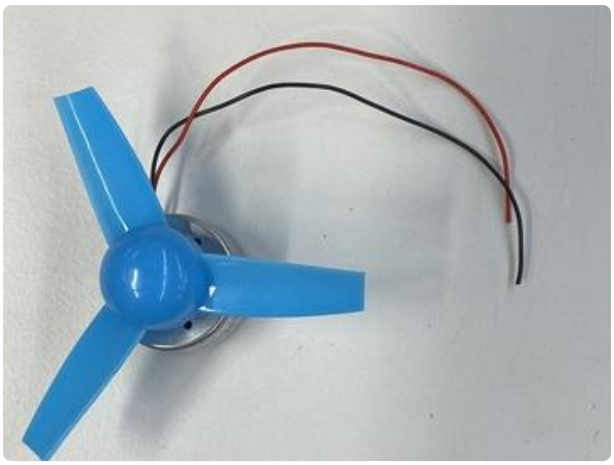


- Add a dab of glue (or hot glue) to either side of the popsicle stick. If the stick in your kit is bowed a little, add the glue to the bowed side. The motor will be mounted the flat side of the roof.
- Insert the popsicle stick into the roof and press down on it until it sets.

Let's leave this piece alone to dry.

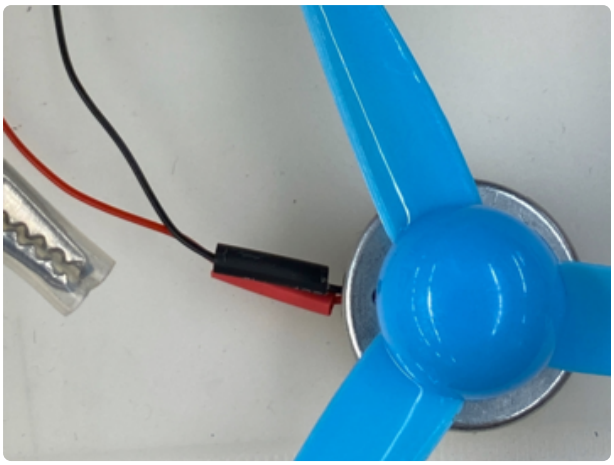


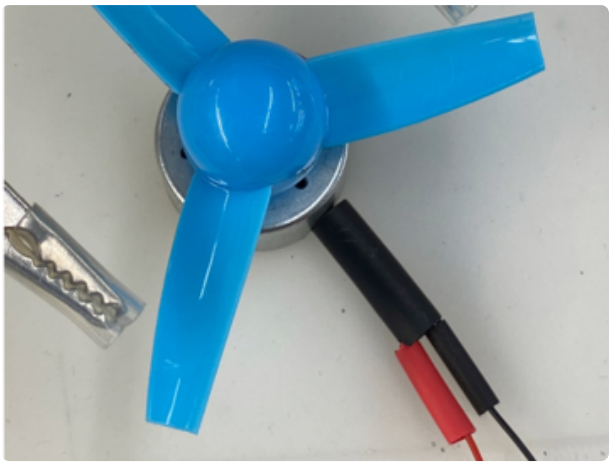




Modify the propeller by cutting ~0.4mm off each end of the propeller with a pair of scissors or wire cutters.

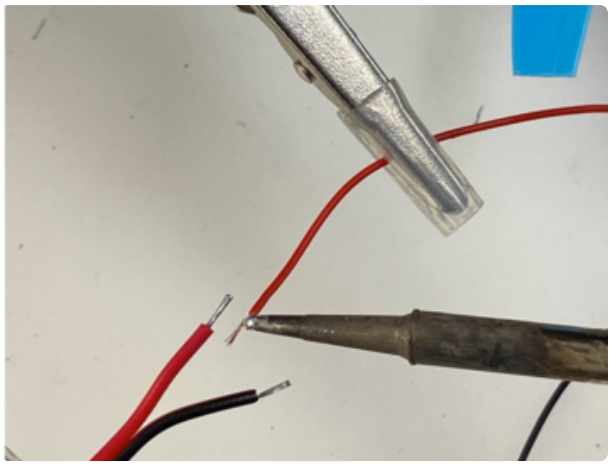
Press-fit the propeller onto the DC Spindle Motor. You may need to press hard, it's a tight fit.

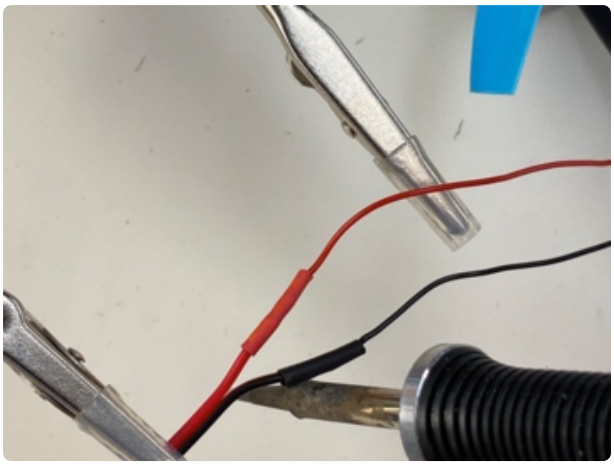


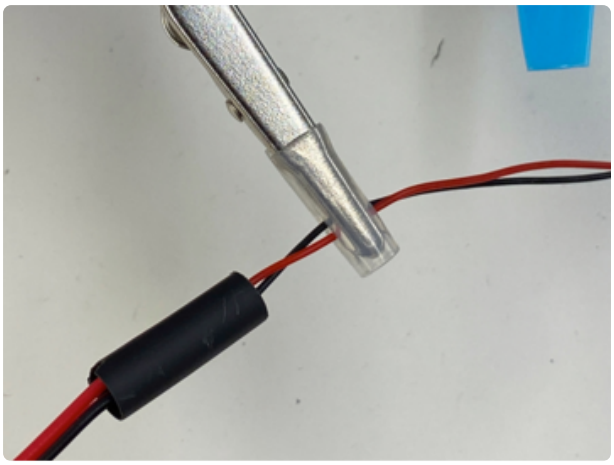


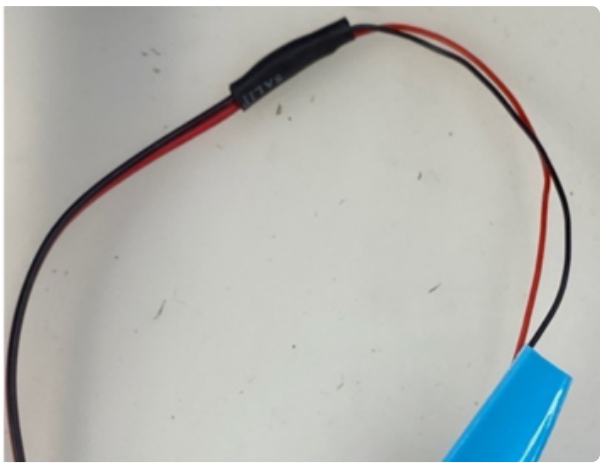
Strip 3mm of the rubber coating from the door sensor. Then, strip 3mm from the male JST header cable.

Slide two narrow lengths of heat shrink down each cable for the door sensor. Then, slide a wider length of heat shrink over both cables.





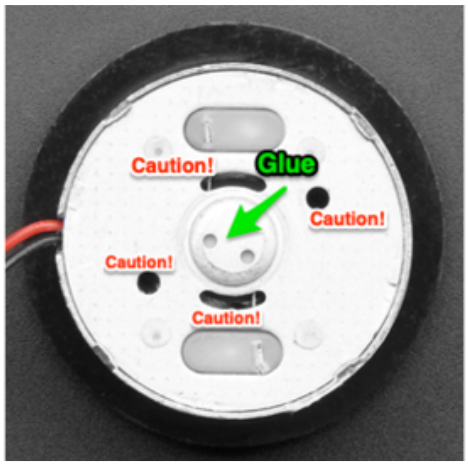




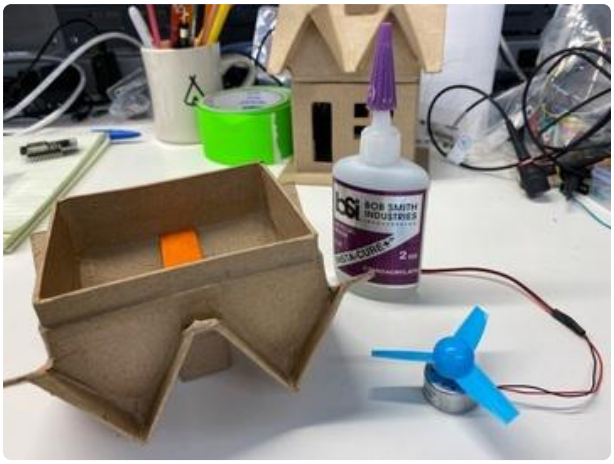
Using a soldering iron, tin the ends of both cables. Then apply heat with the soldering iron to connect them together.

Once connected, slide the heat shrink over the soldering joint and apply heat.

Repeat this for both cables (red and black). Once you're done, slide the larger piece of heat shrink over both of the cables and apply heat.



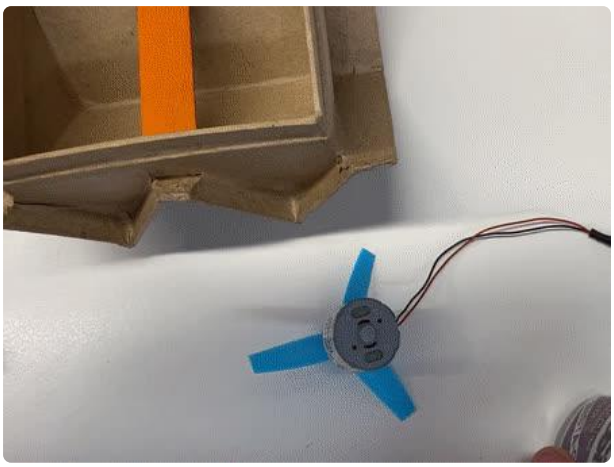
Apply a small dab of glue (or hot glue) to the back of the motor. We've marked on the image where the glue could leak into the motor.



Center the motor above the popsicle stick.

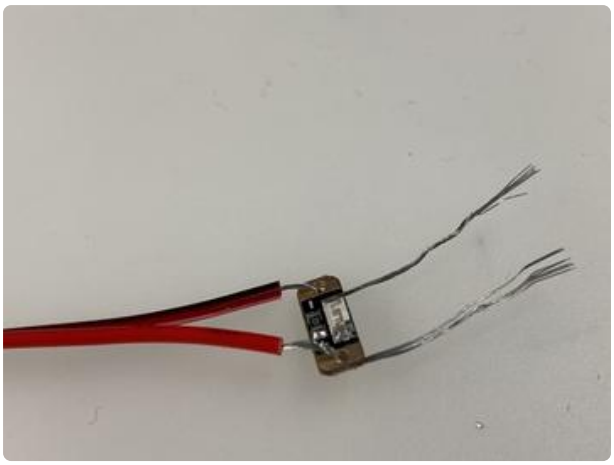
Before pressing down to set the glue, give the fan a spin to ensure it has clearance on all inside sides of the roof.

Press down on the motor to set the glue. Let's leave the roof to dry while we work on the LEDs



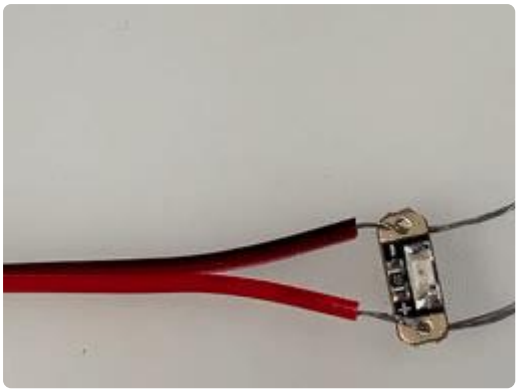
Adding LEDs

You're going to use five Adafruit LED Sequins (in Digi-Key red) to add some lights to your home. These small LEDs only show a single color and don't have digital control, making them simple to add to your home.

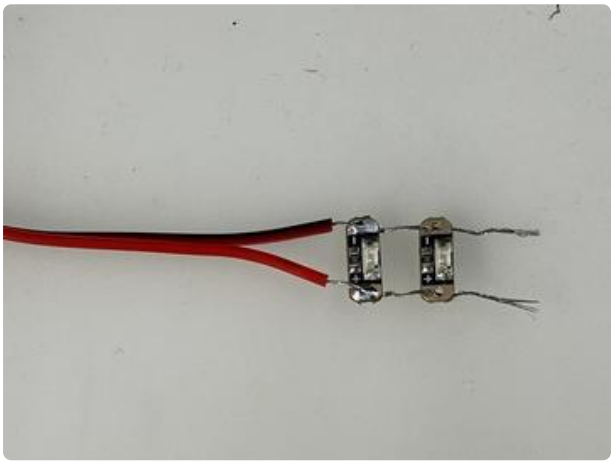


Using a JST cable with a male header, strip off a length of the rubber coating. Twist the stranded wire together and tin the bare wire using a soldering iron.

Slide a sequin LED over each end of the cable. Be sure to line up the red JST cable with the positive end (+) of the sequin.

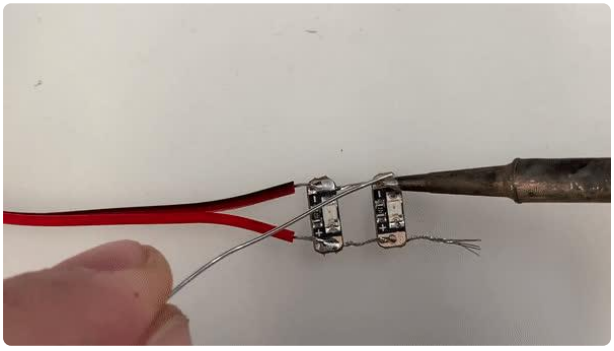


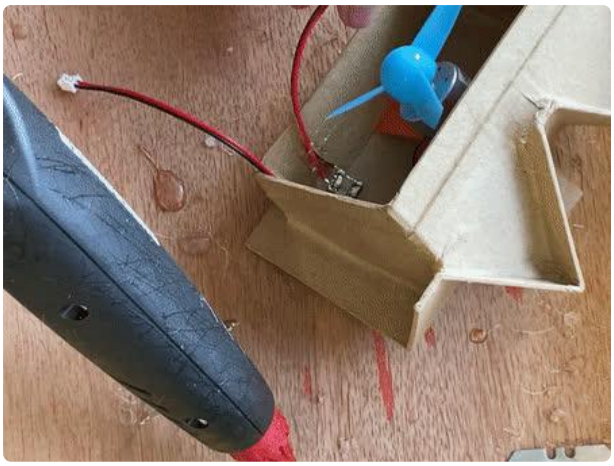
Solder the front and back of the sequin's copper pads to secure the sequin in-place.

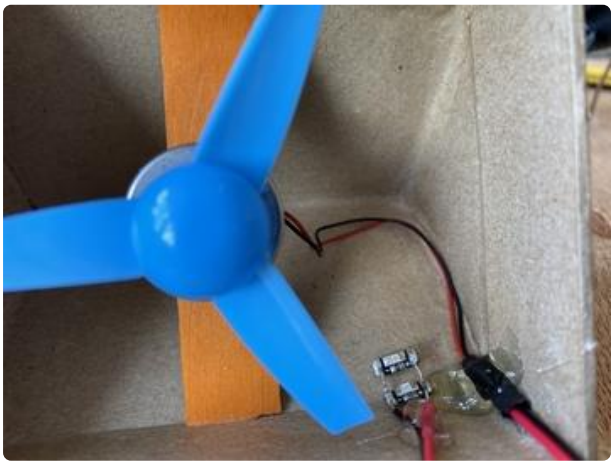


Add a second sequin to the chain, and solder it to the cable.

Five sequins are included with the Digi-Key IoT Home Kit. You may use all five or less (we used two here). Connect and solder the rest of the sequins.







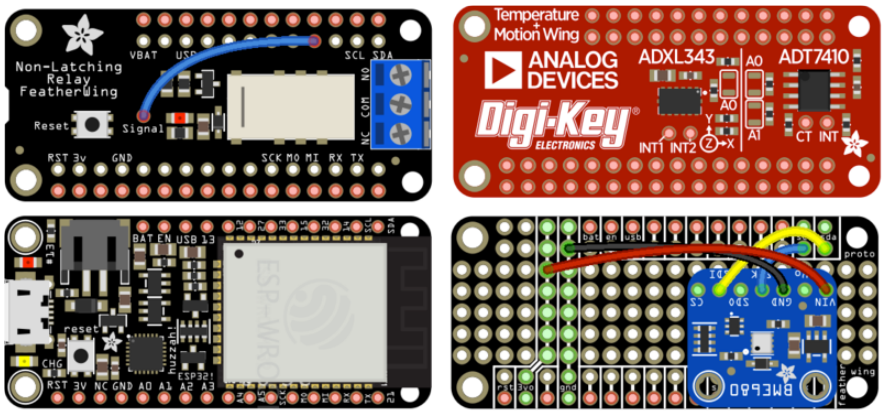
Using hot glue, secure the fan's cabling to the roof.

Then, apply hot glue to the JST cable connected to the LED sequins. You may glue the LED portion down, but, these LEDs are not too bright and adding glue may diffuse them too much.

Assembling the Smart Home Hub

Connect the Feather HUZZAH32 to the FeatherWing Quad Kit along with the ADI FeatherWing and the Non-Latching Relay FeatherWing.

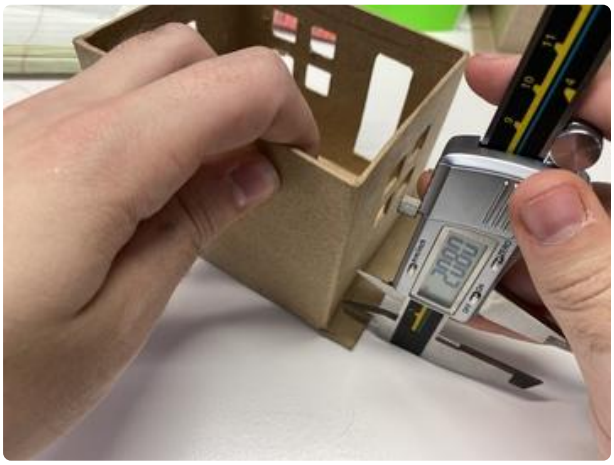
Make sure the orientation reflects the image below.

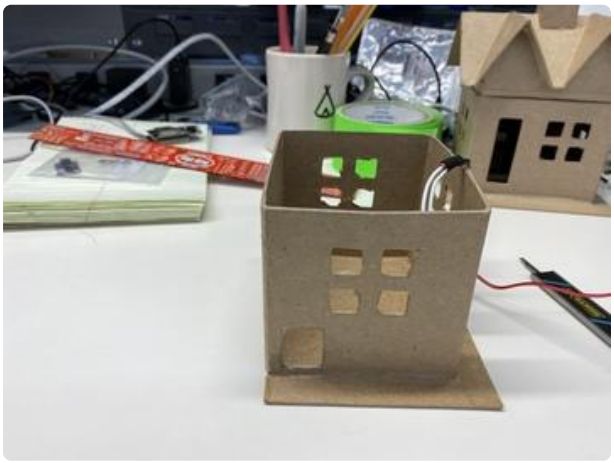


Add a hole for the USB Cable

We'll need to cut a hole on the bottom left side of the home for a USB cable.

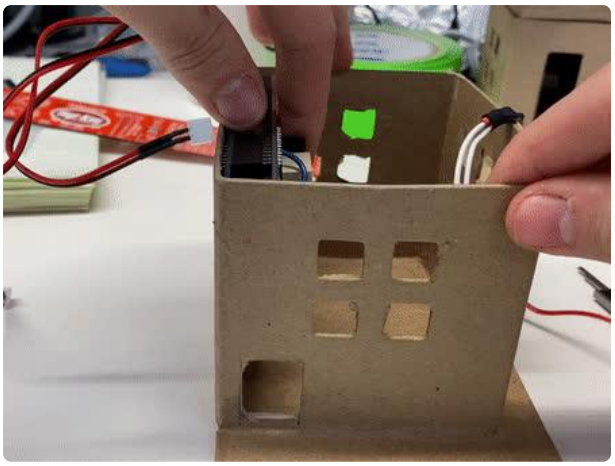




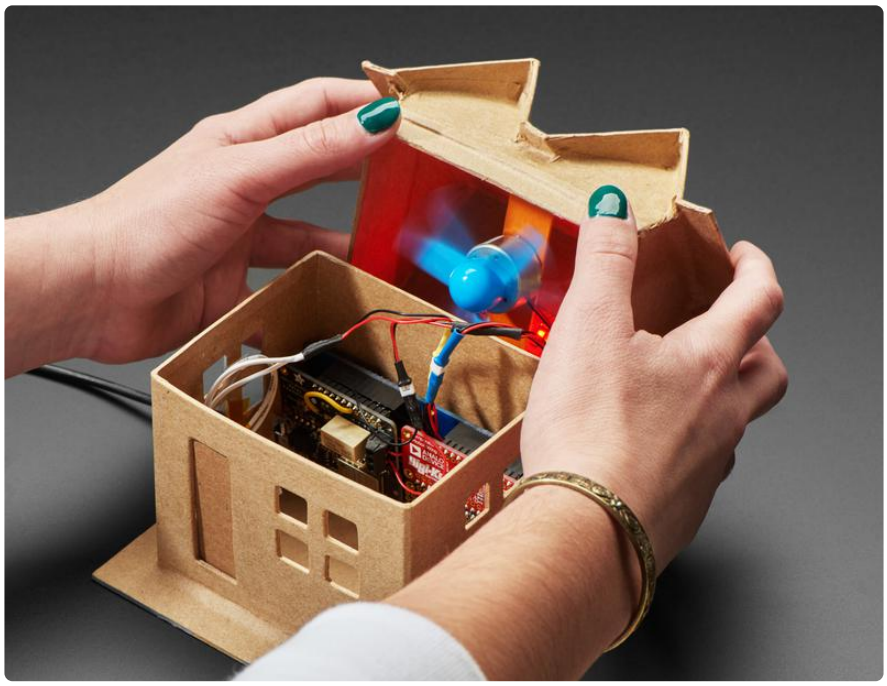


This portion is more tricky than it appears. The IOT House is made out of paper maché, not cardboard. If you cut this material slowly with a dull knife or scissors, it will de-laminate. You will want to use a very sharp X-Acto knife or something like the Slice Craft Knife which has a ceramic blade (https://adafru.it/FJU).

- On the bottom left side of the home, measure and mark a hole 20.8mm x 17.4mm
- Using a sharp blade, cut a hole for the USB cable from the bottom left side of the home.
- Slide the Smart Home Hub into the smart home



Connect up all the JST connectors and close the roof of the home.



With the Home fully assembled, let's move on to programming it!

How Digi-Key IoT Studio Works

Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

We've previously written about end-to-end services (https://adafru.it/HIE) which take care things like of data storage, event messaging and APIs like Particle and Electric

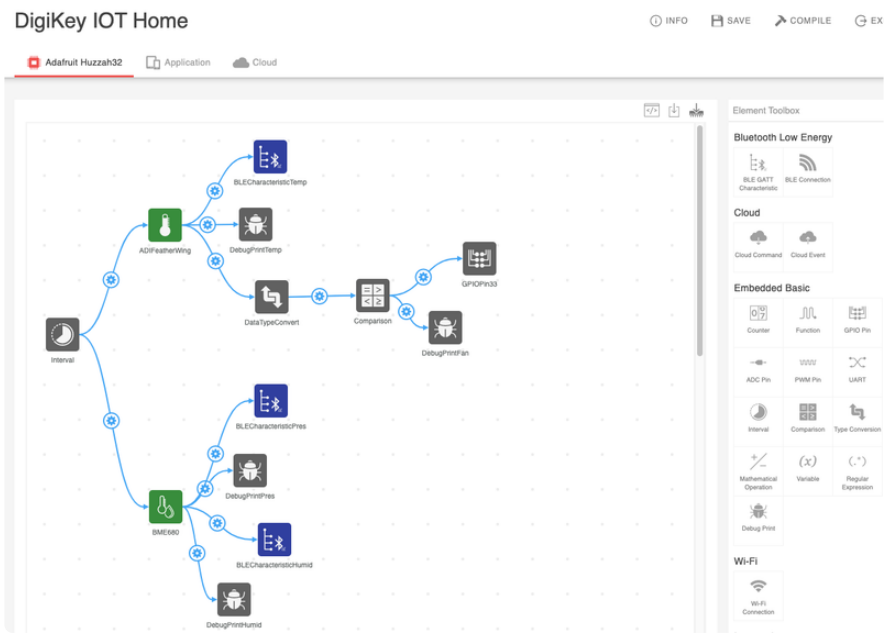
Imp. Digi-Key has entered the field of end-to-end Internet-of-Things services with their own service, Digi-Key IoT Studio.

However, unlike Particle (https://adafru.it/HIE) and Electric Imp (https://adafru.it/HIE), Digi-Key IoT Studio includes built-in board support for dozens of development boards which Digi-Key sells and drag and drop elements for breakouts, sensors, radio modules and more!

Digi-Key IoT Studio is a web-based integrated development environment (abbreviated IDE) that allows people of all skill levels to easily construct IoT Projects. You do not need to install software toolchains, code editors or compile any code on your computer! Digi-Key Studio runs in your web browser, handles all aspects of developing an Internet-of-Things project, and even compiles your code remotely.

Internet-of-Things projects involve programming firmware into a device, interacting with the device (possibly using a website or application), and sending data from a device to "the cloud". Digi-Key IoT Studio comprises of three workspaces which simplify development: an embedded workspace to build device logic, an Application workspace to build an application for your device, and a cloud workspace to construct a method to send data to the cloud.

Embedded Workspace

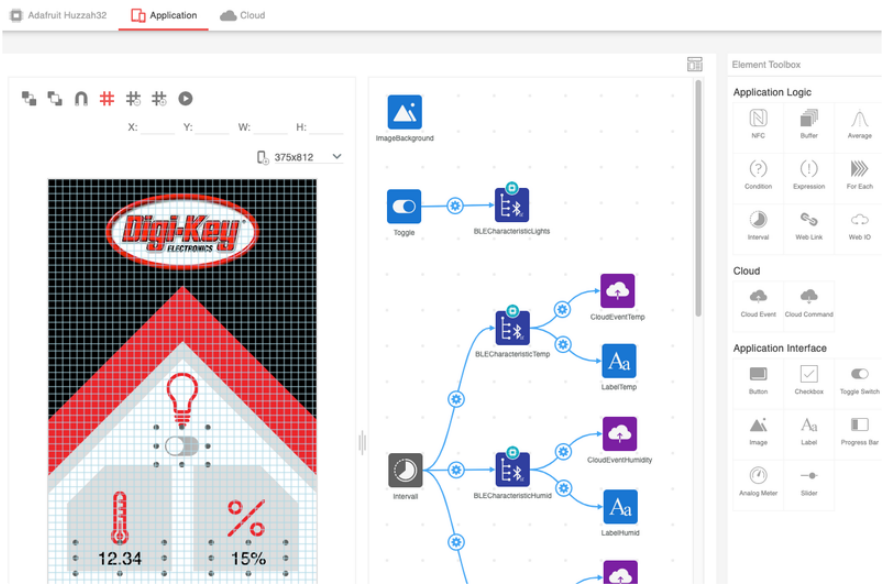


The Embedded Workspace in Digi-Key IoT Studio allows you to build embedded firmware which runs on your microcontroller. You'll be using this tab to program your HUZZAH32. Digi-Key IoT Studio supports two types of transports - Bluetooth Low Energy (BLE) (https://adafru.it/Dik) and WiFi (https://adafru.it/HIF).

You may configure data to be sent directly to the Digi-Key IoT Cloud over WiFi, or send data to an Application on a device (like a phone or tablet) over BLE.

Once you've finished building up your project's firmware, you can move on to building an application to interact with it.

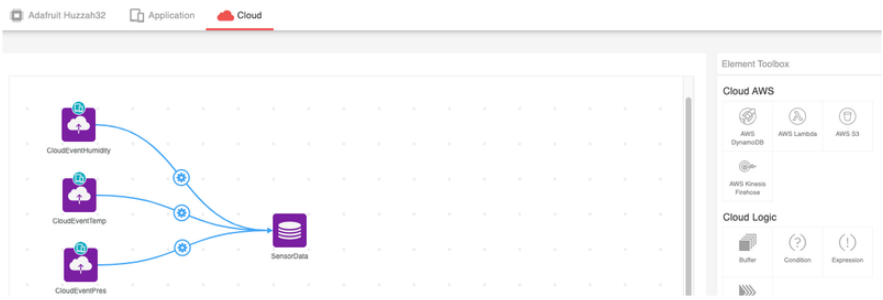
Application Workspace



The Application Workspace allows you to design an interface to interact with your microcontroller from a tablet or smartphone. This workspace is split into two sides - a visual "App Builder" which allows you to drag and drop interface elements onto an application and a workspace for building the application's logic.

Once you've built up an application, it's time to send data to the Digi-Key IoT Cloud Service!

Cloud Workspace



The Cloud Workspace uses WiFi (https://adafru.it/HIF) and the MQTT protocol (https://adafru.it/Fmp) to send data from your IoT Project to the internet. As with most end-to-end solutions (https://adafru.it/HIE), Digi-Key provides their own Cloud Storage which can store up to 10MB of data per-device. If your IoT project requires a large amount of cloud storage - you can still use Digi-Key IoT Studio. Digi-Key IoT Studio supports sending data from your project to larger cloud storage providers like Amazon AWS.

Getting Started

Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

Before you can use Digi-Key IoT Studio, you'll need to install two applications: DK IoT Studio Mobile and IoT Agent.

Install DK IoT Studio Mobile App

The Digi-Key IoT Studio app is an app which works with microcontrollers such as the Adafruit Feather ESP32 Huzzah and the Digi-Key IoT Platform. It lets you monitor your connected projects from your phone or tablet.

Download Digi-Key IoT for iOS

https://adafru.it/Hma

Download Digi-Key IoT for Android

https://adafru.it/Hmb

Install IoT Agent

You'll need the IoT Agent to program your HUZZAH32 from Digi-Key IoT Studio. The IoT Agent is a local application that's used to program hardware from your web browser. It runs on every major desktop operating system:

Download Digi-Key IoT Agent for Windows

https://adafru.it/Hmc

Download Digi-Key IoT Agent for macOS

https://adafru.it/Hmd

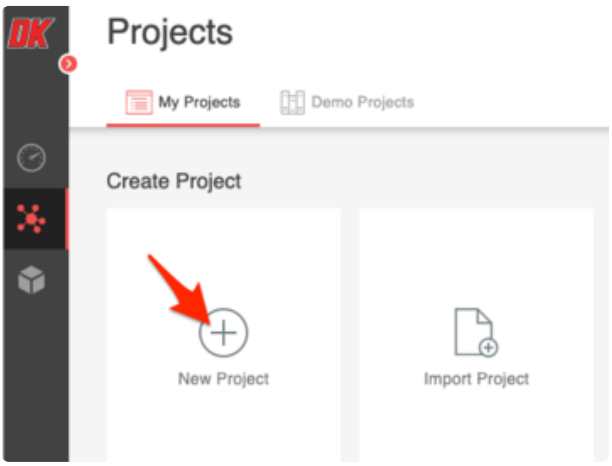
Download Digi-Key IoT Agent for Linux

https://adafru.it/Hme

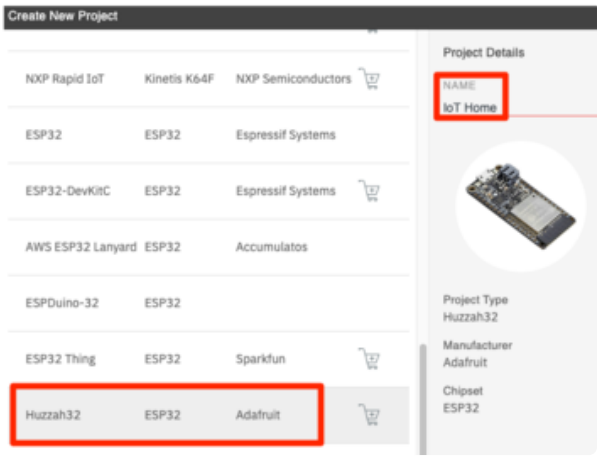
If you encounter an issue with running the IoT Agent, post up on the Digi-Key Tech Forum (https://adafru.it/Hmf) for support.

Creating a Digi-Key IoT Studio Project

Once you installed both the Digi-Key IoT Mobile App and IoT Agent, navigate to the Digi-Key IoT Studio website (https://adafru.it/HmA). You'll be prompted to log in with your Digi-Key account, or create one if you haven't yet.



You'll be brought to the Studio Projects screen. From the Studio Projects screen, create a new project by clicking the + button.



You'll be creating a project using the Adafruit HUZZAH32 board. Select the Huzzah32 and name the project IoT Home.



Once the project is created successfully, you'll be brought to the Embedded Workspace for your project.

The embedded workspace in Digi-Key IoT Studio Project allows you to build embedded firmware. You'll be using this workspace to program your HUZZAH32.

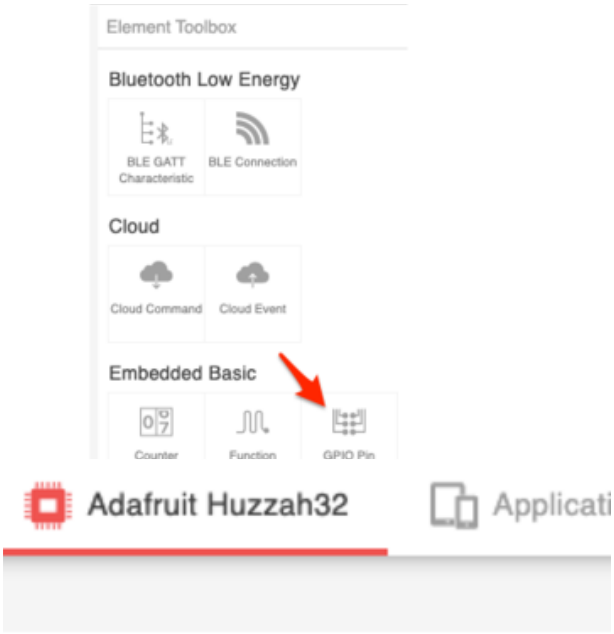
Let's get started by building a project which can turn on or off the smart home's LEDs from the mobile app.

Turning on the Lights

Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

Step-by-Step: Embedded Workspace

You're going to create a mobile application to turn on the IoT Home's lights from an application on your phone.



You'll start by adding embedded elements to your workspace. Earlier, you connected the LED lights to a pin on the Feather Huzzah32. You'll use a GPIO Pin Element to read or write the state of a GPIO pin.

- From the element toolbox within IoT Studio's Embedded workspace, click the GPIO Pin element.
- You should see a new GPIO Pin element appear on your workspace.





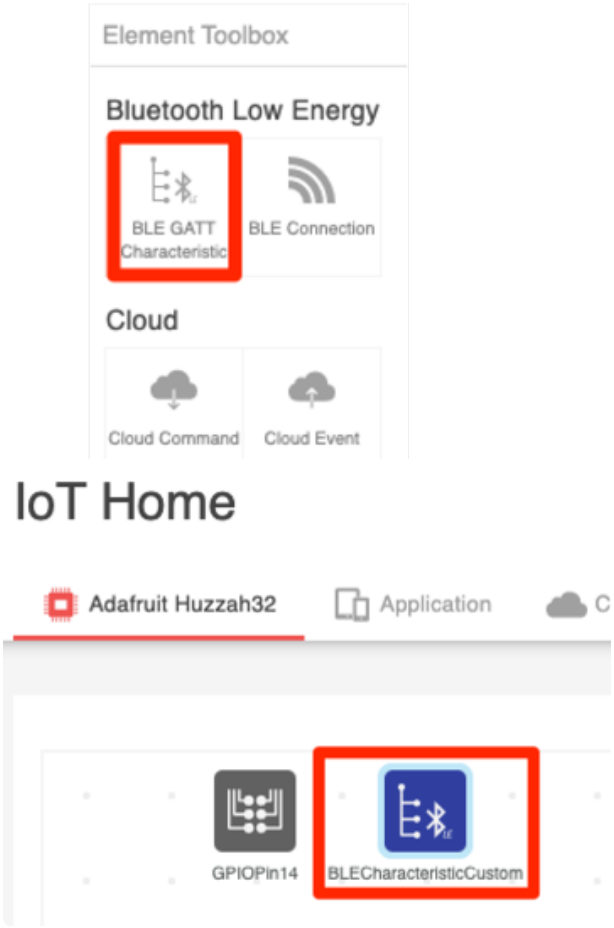
Click the GPIOPin element in the workspace to bring up the element's properties.

- Later in this project, you'll be adding another GPIO element. To avoid confusion, rename the element to GPIOPin14.

Your IoT Home uses the Feather Huzzah32's GPIO Pin #14 for the LEDs.

- Change the GPIO Pin to IO14.

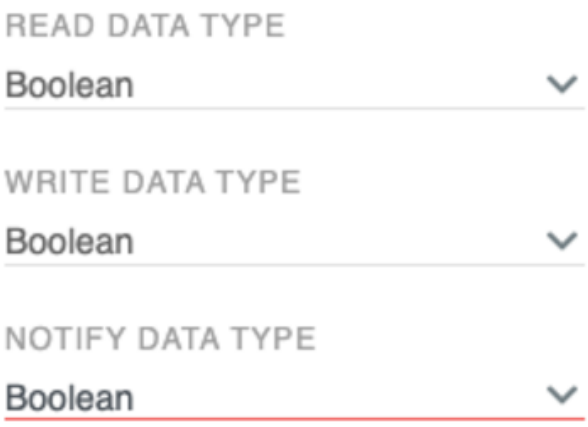
Congrats - you've set up your first IoT Studio Element. Now, you need a way to interact with the LEDs.



Digi-Key IoT Studio provides two different types of transports for your Feather Huzzah32 - Bluetooth Low Energy (BLE) and WiFi ("Cloud").

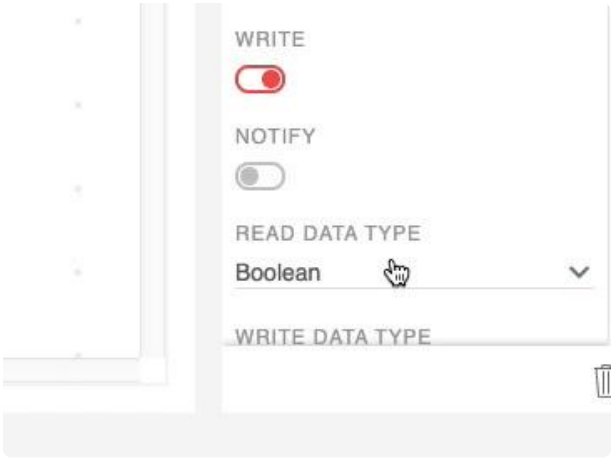
Since you'll be sending a small amount of data (a binary 1 or 0) to the IoT Home from the mobile application, and want it to respond quickly, you're going to use Bluetooth Low Energy as a transport.

- From the element toolbox within IoT Studio's embedded workspace, click the BLE Gatt Characteristic element.
- You should see a new element in your embedded workspace named BLECharacteristicCustom.



Click the BLE GATT Characteristic Element to bring up its properties. Later, you'll be connecting this element to a toggle switch on the mobile application. The toggle switch element can only send a 1 or a 0, you'll need to change the Data Type to a boolean.

- Change Read Data Type to Boolean
- Change Write Data Type to Boolean
- Change Notify Data Type to Boolean



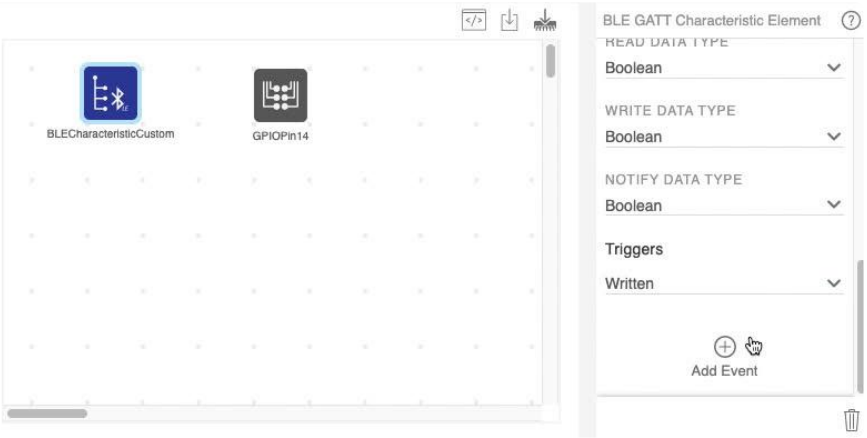
With the BLE GATT element and GPIO element configured, let's connect them together!

- Scroll down in the BLE GATT Characteristic Element Properties until you see Add Event.
- Click the + Symbol to create a new connector.

You're going to create a connector from the BLE Characteristic Element to the GPIO Pin.

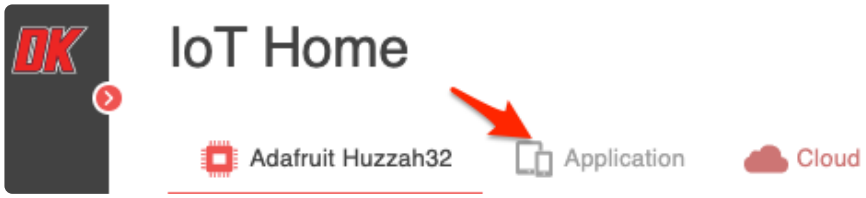
Click the + Symbol above Add Event in the BLE Charactertistic Element's properties. T hen, click the GPIO element.

Change the connector's ability to Set State.

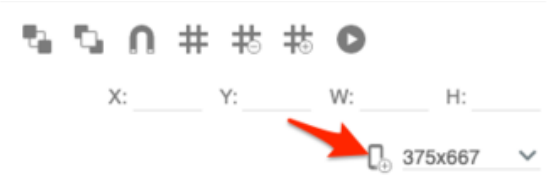


Step-by-Step: Application Workspace

With the firmware portion of the project completed, it's time to build a mobile application to interact with the embedded project. Click the Application Tab underneath your project's title to be brought to the Application workspace.

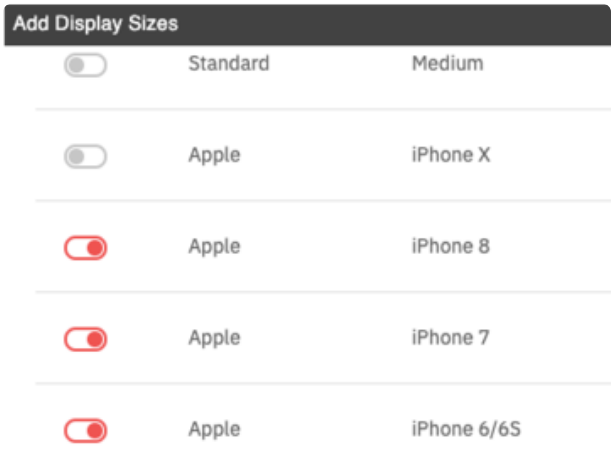


The Application Tab is different from the cloud or embedded tabs - it contains an App lication Builder to design the project's interface and an Application Workspace to construct the application's interface and logic.



Let's begin by changing the display size to match the device you're using.

- On the Application Builder, click the Add Layout button.
- Click the toggle switch to select the display size that best matches your device's display.
- Click Save

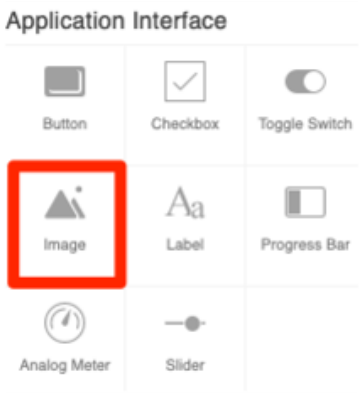


Let's add a background image for the application. We created one for the iPhone X display size. You may need to crop/resize the image to fit your device's display.

Download the background image by clicking the button below and save it to your Desktop.

Download iot-home-background.png

https://adafru.it/HmB



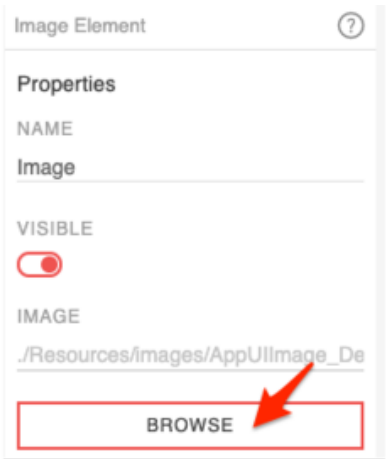
From the element toolbox under Application Interface, select the Image element.

An image element should appear on the workspace and the application builder.

- From the workspace, click the Image Element to bring up its element properties.
- From the Image Element's properties, click Browse and select the iot-home-background.png image you downloaded earlier.



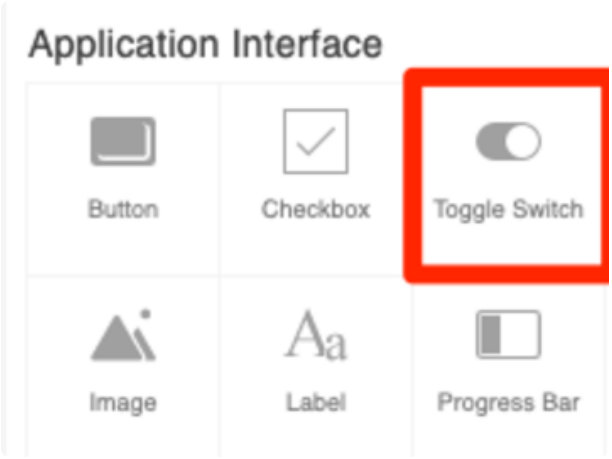
You can move the background image around and resize it to fit the size of the display you're using.





You'll notice that there is a BLE Characteristic Element on the Application Workspace - this isn't a mistake. In fact, this is the same element you added to the embedded workspace. Bluetooth and WiFi elements in IoT Studio often create mirror elements in other workspaces. One purpose of this is to send data bi-directionally between a mobile application and the embedded firmware.

You'll be be writing the state of the toggle switch from the app to the BLE characteristic element. The firmware will read this element's value and set the pin to the state of the toggle switch.

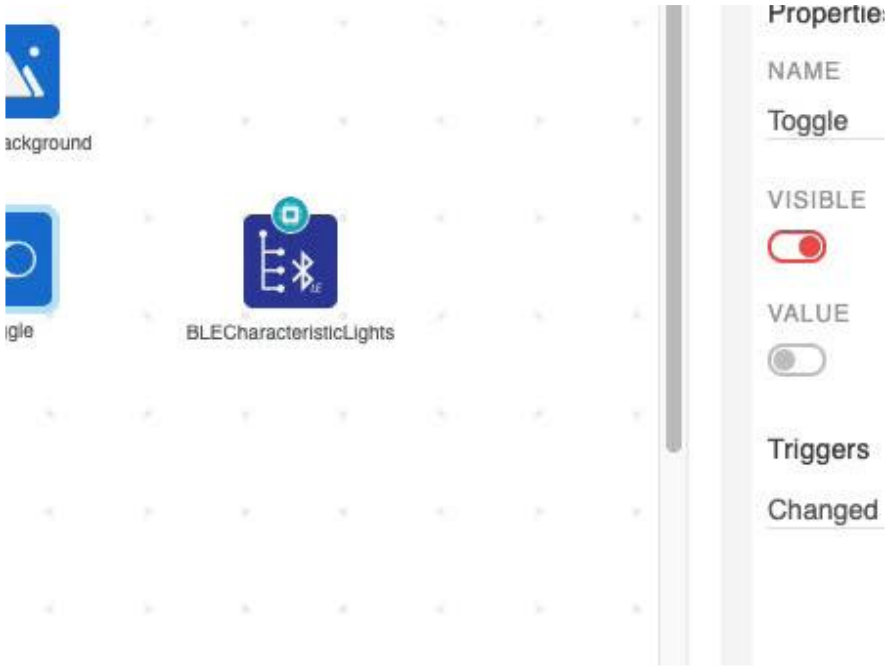


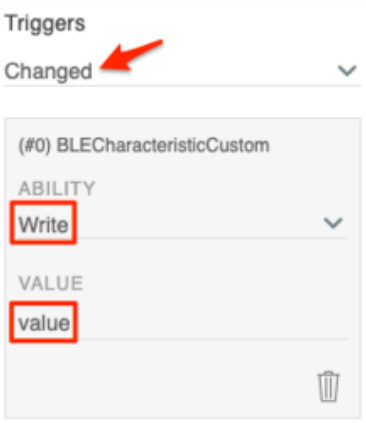
Digi-Key IoT Studio has an element which is perfect for controlling the lights in your home. The toggle Switch element is bi-directional and can be used to send either 1 or 0 to the lights to turn them on or off.

From Application Interface, select the Toggle Switch element.

Since you'll be sending data from the Toggle Switch element to the BLE Characteristic Element, change the order of the elements by dragging and dropping them such that the toggle switch is placed before the BLE Characteristic.

- Click the Toggle Element on the Workspace to bring up its element properties and click Add Event.
- Drag a connector from the toggle element to the BLE Characteristic element on your workspace.



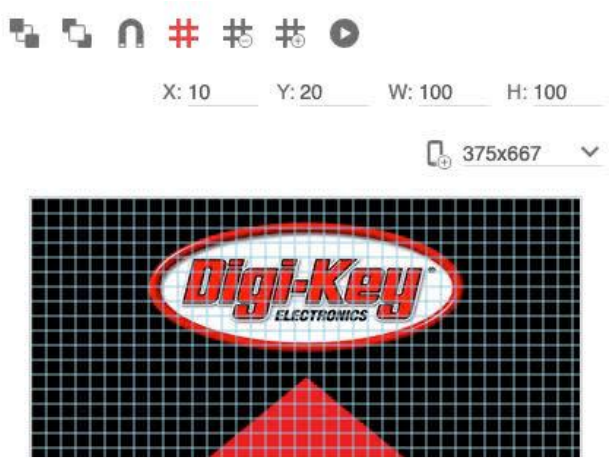


Each connector can execute different events based on a trigger. We'll want to write the state of the toggle switch (on or off) to the BLE Characteristic Element, so we'll use the Toggle Element's Changed trigger.

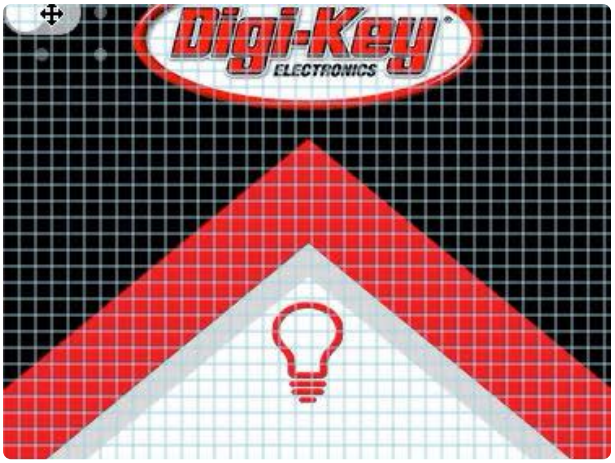
- From the Application Workspace, click the Toggle Element to bring up its properties.
- Verify that the Toggle Element trigger you are modifying is the Changed trigger.
- Set the Ability to Write
- Set the Value to value

You might be wondering why the Toggle element is not appearing on the application builder. The element is on the builder, but it's located in the background layer. The Application Builder supports two layers for images. The background is in front of the toggle element.

Let's bring the toggle element to the top layer so we can interact with it.



- From the Application Workspace, click the Toggle Element
- From the Application Builder Toolbar, click the Send to Front button.
    - The Toggle Button should appear on the interface.



To make sure you'll align this element correctly, you'll use Application Builder's Snap-to-Grid feature.

- From the Application Builder Toolbar, click the # icon.
- Click the toggle element and drag it underneath the lightbulb icon on the background image.

Step-by-Step: Uploading Firmware

You just created embedded firmware and a mobile application. Now, let's program the firmware onto the Feather Huzzah32 and control the IoT Home from your device.



Click the Compile button at the top of your project to compile the project. This will take anywhere from a few seconds to a few minutes, depending on how complicated your project is and how busy the build servers are.

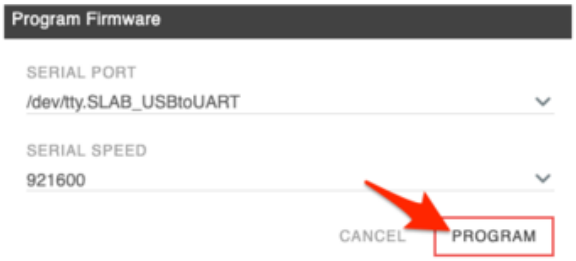
Once compiled successfully, a green status bar across the bottom of the screen will appear telling you that your project is compiled.

If there is a compilation error, a window with a descriptive compiler error will appear over the workspace.

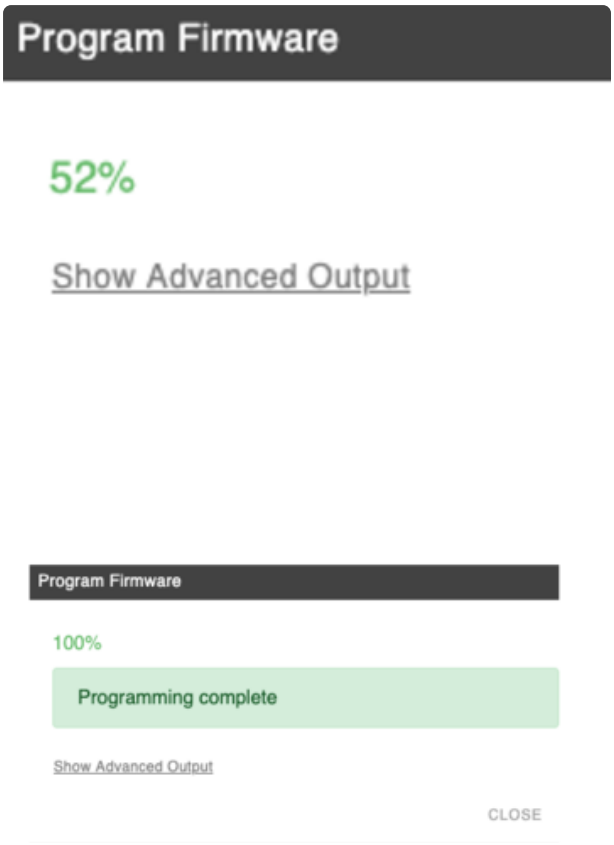


Next, plug your Huzzah32 into a port on your computer and make sure the IoT Agent application is running in the background (it should be visible in your system tray).

- From the Embedded Tab, click the Program Firmware Icon.
- Select the Serial Port your Huzzah32 is connected to and select a serial speed.
- Click Program



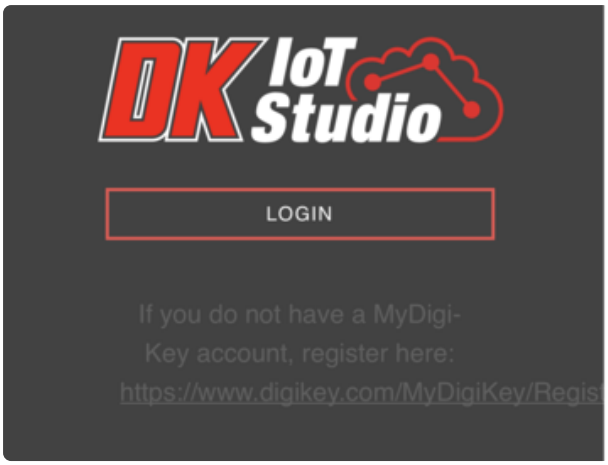
The Program Firmware window will update when the programming is complete.

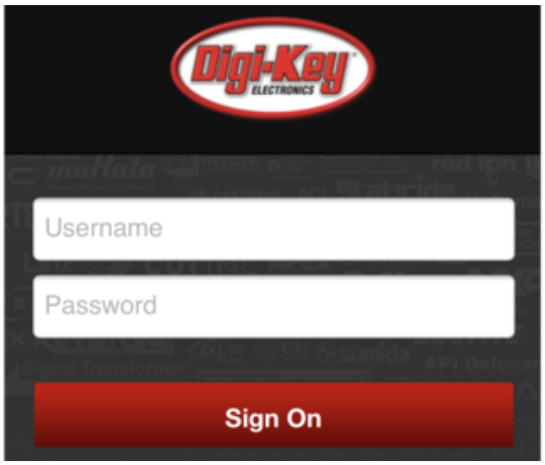


If your program uploaded successfully, move on to using the Digi-Key IoT Mobile Application to interact with your IoT Home's lights.

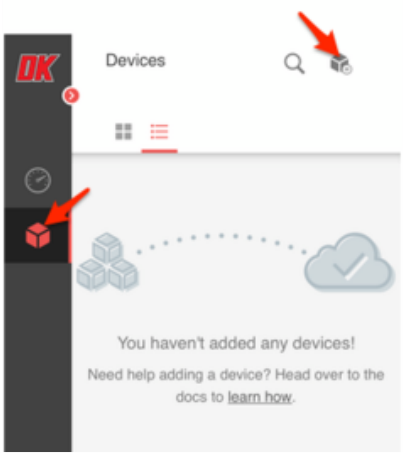
Step-by-Step: DK IoT Studio Mobile App

The DK Studio Mobile app "works in conjunction with a wireless device (your cell Phone) and the Digi-Key IoT Platform (cloud service) to monitor your custom-built connected things".





Open the application from your device. Click the Login button and log into your Digi-Key account.



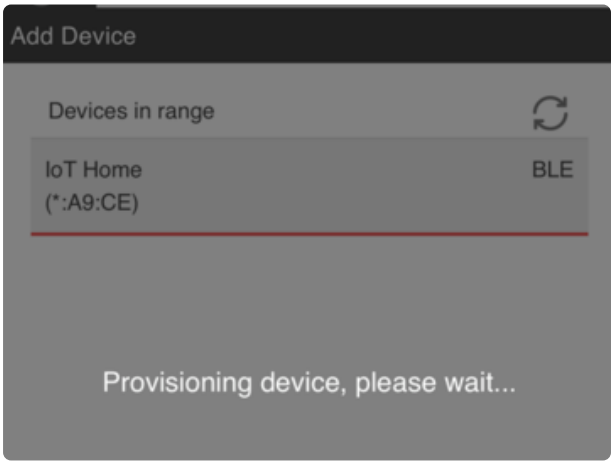
The DK IoT Studio Mobile App is a pared down version of the web-based Digi-Key IoT Studio. You can't edit or modify code but you can connect to your devices and view their status.

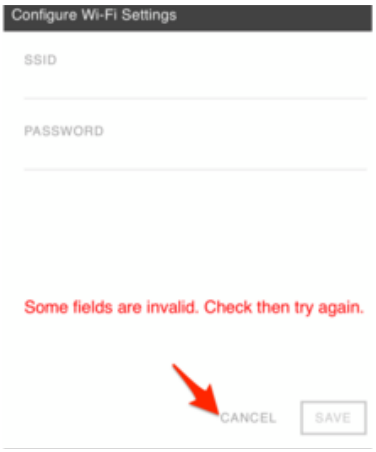
- From the sidebar, tap the Cube icon. This will bring up the Devices screen.
- From the top of the Devices screen, tap the cube with a plus icon (+) to bring up the Add Device screen.



Your IoT Home should appear as a device in range, along with its MAC address and transport (BLE).

- Tap the device name
  - Make sure not to unplug the Feather while the app is in the middle of the provisioning process.





The WiFi Provisioning screen will appear. Since you are using BLE instead of WiFi, tap the cancel button.

The app should load its Application View. Tap the toggle button - the lights should turn on and off!



Monitoring your Home

Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

Now that you can turn on the lights inside your home - you may be asking what else can you do? One popular feature people add to their smart-homes is environmental monitoring. Tracking your home's environmental data is useful for both short-term alerts (if the temperature in a room is too low, or the concentration of gas is too high) and long-term logging (logging data over a long period of time can help us detect abnormalities within the data).

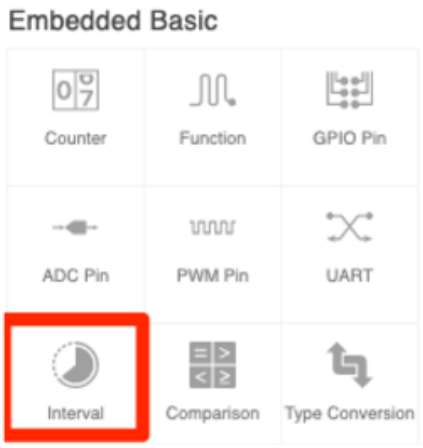
You're going to use the Embedded Workspace to build firmware for obtaining precision temperature readings from inside your home using the Analog Devices FeatherWing. Then, you'll add the Bosch BME680 to obtain readings for the relative humidity and the gas level.

Then, you'll use the Application Workspace to enhance your mobile application by adding labels to display readings from the sensors inside your home.

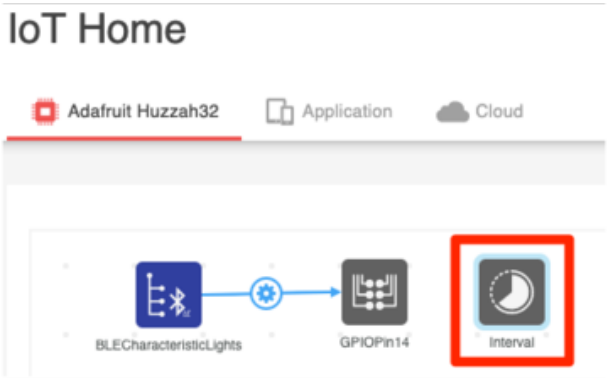
Finally, you'll switch to the Cloud Workspace to send data from your DK IoT App to DK IoT Studio's Cloud Storage.

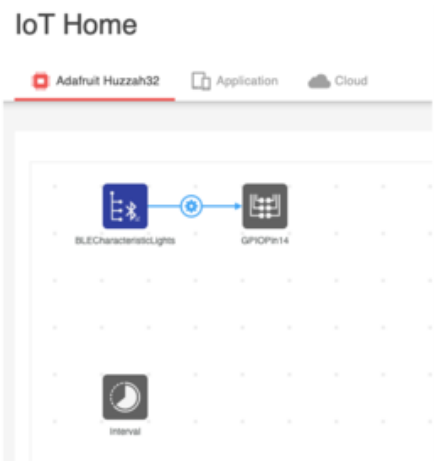
Step-by-Step: Embedded Workspace

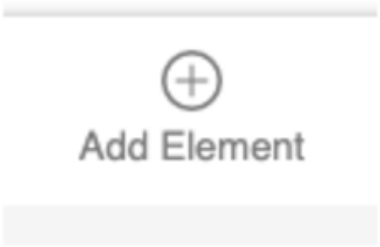
You'll begin by adding an interval element to your project. Digi-Key IoT uses interval elements to trigger connected elements. You can specify an interval in milliseconds. When the timer expires, it will trigger all elements connected to it.



- From the element toolbox within IoT Studio's Embedded workspace, click the Interval element.
- Move the Interval Element underneath the BLECharacteristic Element.







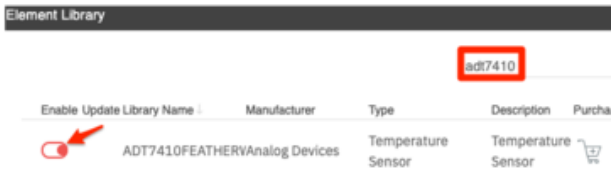
Elements for the BME680 or the Sensor FeatherWing are not within the default element toolbox - you'll have to manually add them from within the Element Library.

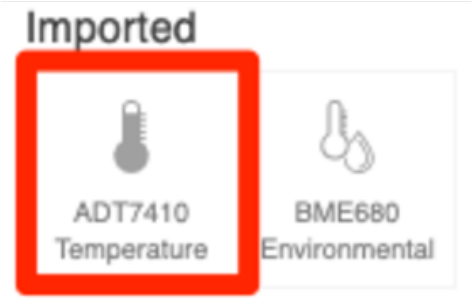
- From the element toolbox within IoT Studio's Embedded workspace, click Add Element
- From the Element Library, search for the BME680.
- Enable the BME680 library elements by clicking the toggle button.
- From the bottom of the Element Library, click Save



You will also need to import the ADT7410 sensor library.

- From the element library, search for the ADT7410FEATHERWING.
- Enable the ADT7410FEATHERWING library.
- From the bottom of the Element Library, click Save

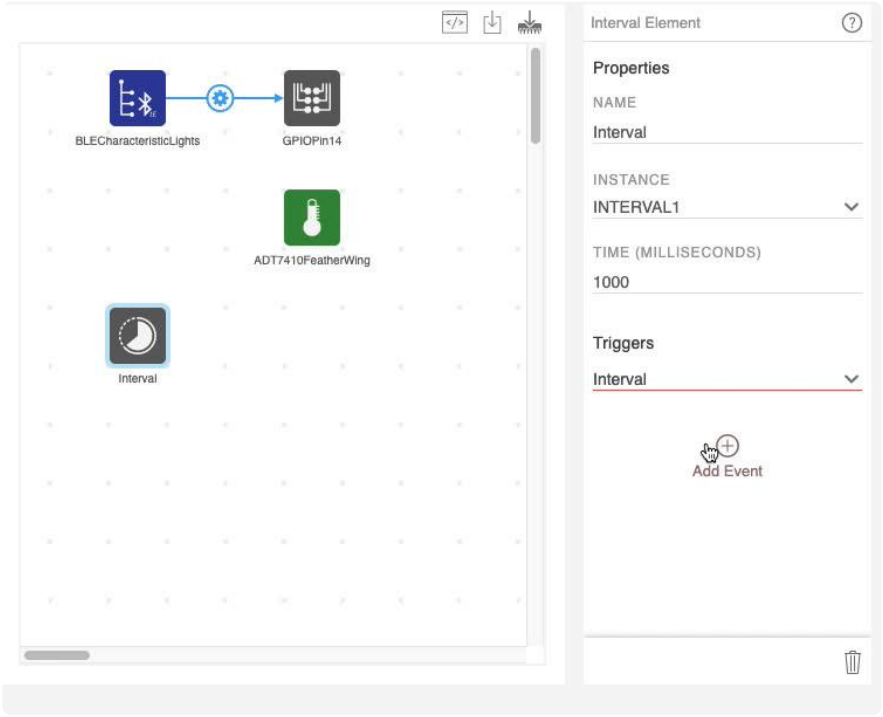




The ADT7410 and BME680 sensors should now appear in your Element Toolbox under Imported. Let's add the ADT7410 temperature sensor to the workspace.

- From the element toolbox within IoT Studio's Embedded workspace, click the ADT7410 Temperature Element to add it to the Workspace.
- Move the ADT7410 element underneath the GPIO element.

Make a new connection between the Interval Element and the ADT7410 Element.

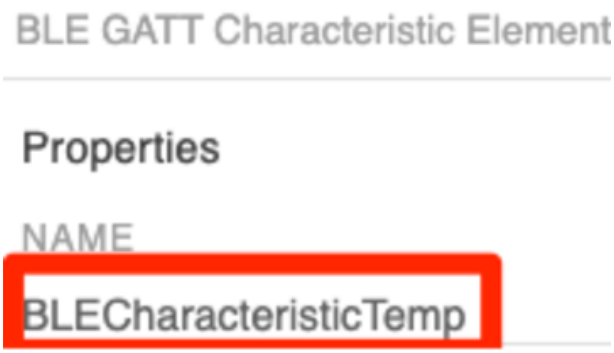


Every time the interval executes, the interval will trigger the ADT7410FeatherWing element to read the sensor's temperature in degrees Celsius.



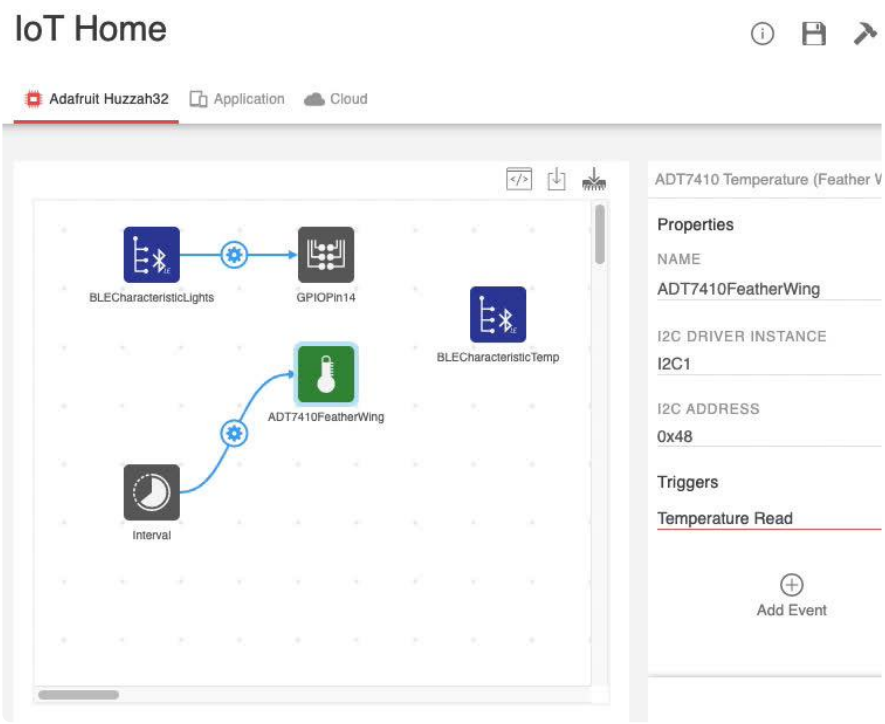
Let's create a new BLE GATT Characteristic to read the temperature sensor's value.

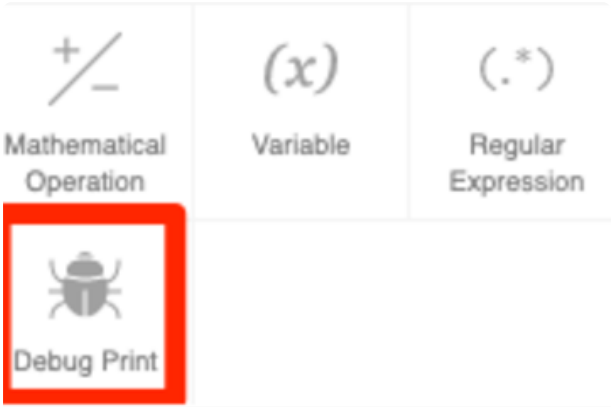
- From the element toolbox within IoT Studio's Embedded workspace, click the BLE GATT Characteristic Element to add it to the Workspace.
- Click the BLECharacteristicCustom Element on the Embedded workspace to bring up its properties.
- Change the element's name to BLECharacteristicTemp

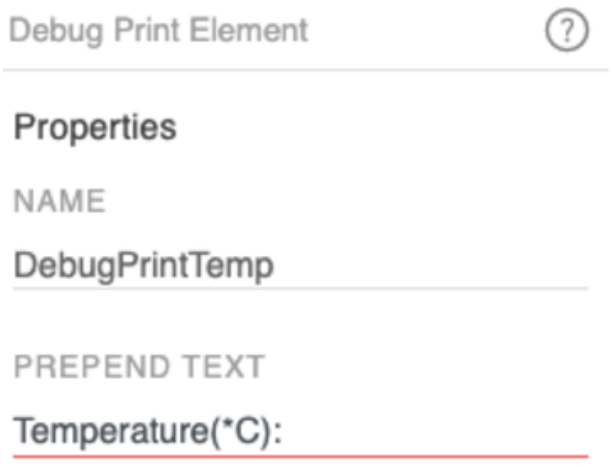


Let's add a new event to write data to the BLE Characteristic each time the temperature is is read.

- Click the ADT7410FeatherWing Element
- Click Add Event
- Add a connector between the ADT7410FeatherWing Element and the BLECharacteristicTemp Element.







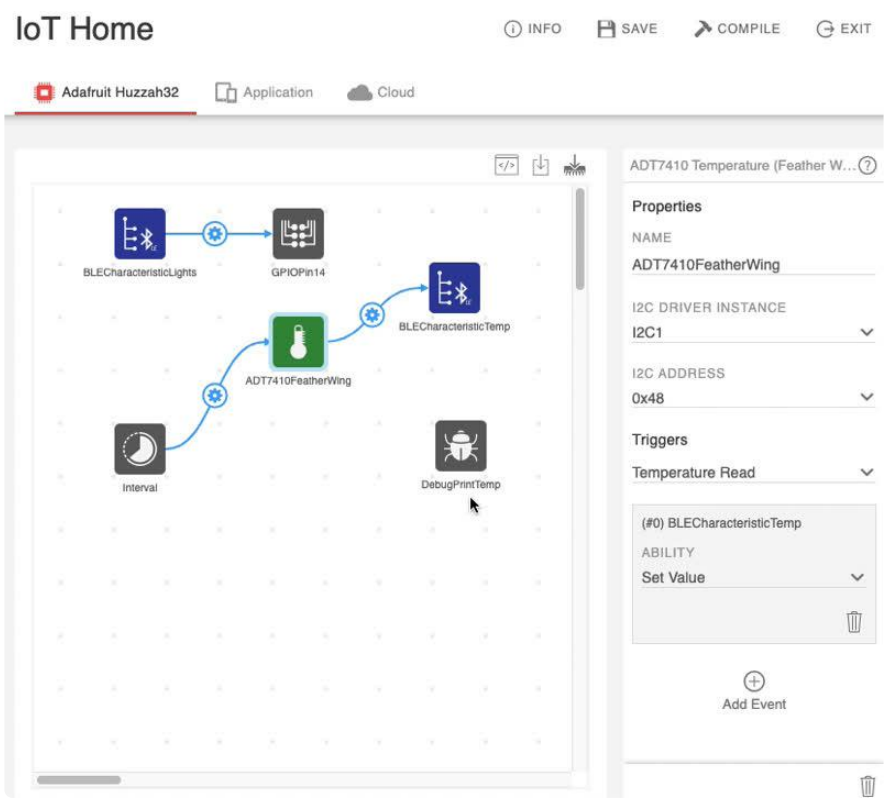
The debug print element. This element will print the sensor value to the Huzzah debug UART console (on your computer). This is useful for checking if the sensors are operating correctly, independently of the BLE connection.

- From the element toolbox within IoT Studio's Embedded workspace, click the Debug Print Element to add it to the Workspace.
- From the Debug Print Element's properties, name the element DebugPrintTemp.

You can add a label to differentiate which value is being printed to the console:

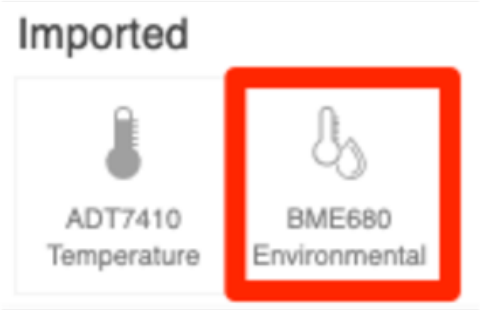
- Under Prepend Text, add the text Temperature (*C):
  - The temperature reading in the UART terminal will look something like: `Temperature (*C): 25.23`

Make a new connection between the ADT7410FeatherWing element and the DebugPrintTemp Element.

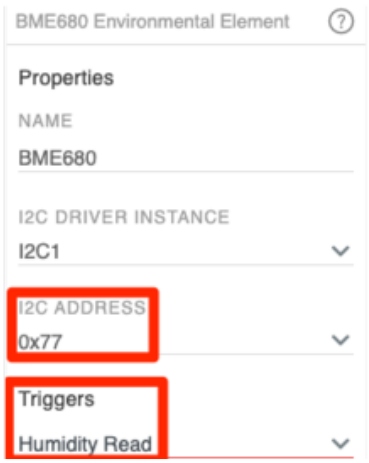


To summarize - every time the interval's timer expires, the ADT7410 will read the temperature. Once the temperature is read from the sensor, it'll be sent to the BLECharacteristicTemp element and printed to the UART console.

Next, re-do these steps, but for the BME680 sensor.

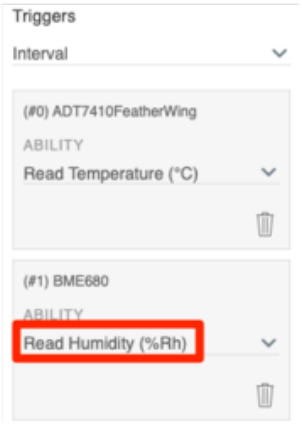


- From the element toolbox within IoT Studio's Embedded workspace, click the BME680 Element to add it to the Workspace.
- Click the BME680 element to bring up the properties.
- Change the I2C Address to 0x77
- Change the trigger to Humidity Read

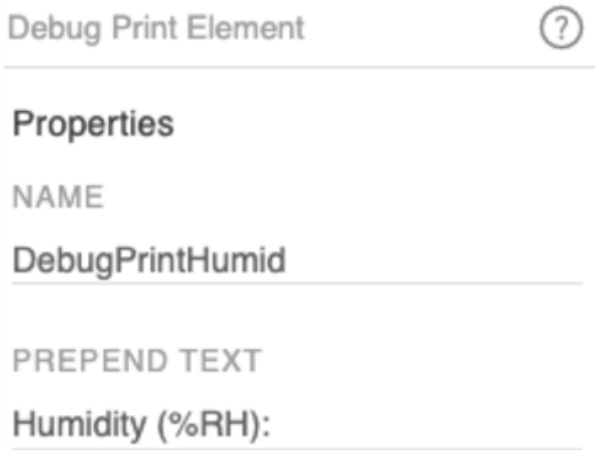


- Add a connector from the Interval Element to the BME680 Element.
- In the Interval Element's Properties, change the BME680's ability to Read Humidity (%RH)



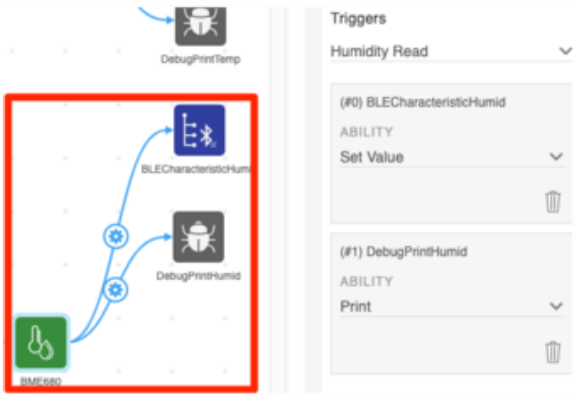




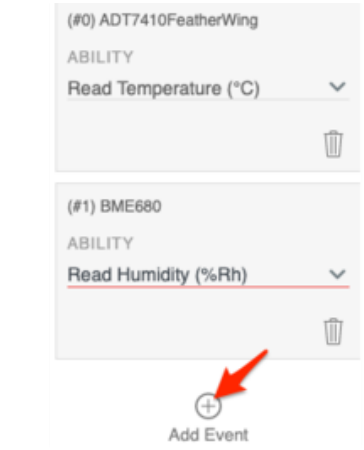


Let's add some elements to receive and print the data from the sensor.

- From the element toolbox within the IoT Studio Embedded workspace, click the BLE GATT Characteristic Element to add it to the Workspace.
- Change the name of this element to BLECharacteristicHumid
- From the element toolbox within IoT Studio's Embedded workspace, click the Debug Print Element to add it to the Workspace.
- Change the name of this element to DebugPrintHumid
- Modify the Prepend Text to Humidity (%RH)

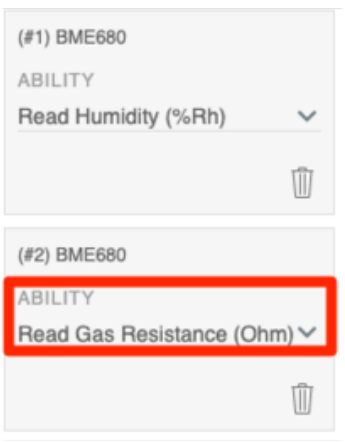


- Click the BME680 element to bring up its properties.
- Under the Humidity Read trigger, click Add Event
- Add a connector between the BME680 element and the BLE Characteristic Element
- Add a connector between the BME680 Element and the Debug Print Element.



Next, let's read data from the BME680's gas sensor. This sensor will read its resistance in ohms.

- From the element toolbox within IoT Studio's Embedded workspace, click the Interval Element to bring up its properties.
- From the Interval Properties, click Add Event
- Add a connector between the Interval Element and the BME680 Element.
- Change the ability of the connector to Read Gas Resistance (Ohm)

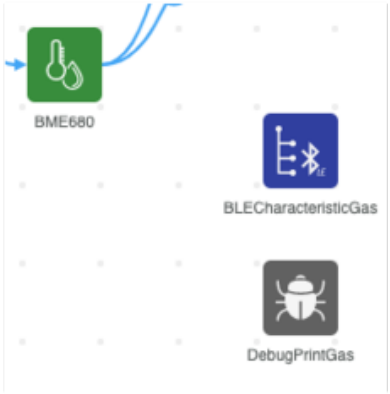




Click the BME680 Element in the Embedded Workspace to bring up its properties.

Change the Trigger from Humidity Read to Gas Resistance Read.

You'll notice that there are no events defined for this trigger, but there are connectors from the BME to other elements. Those connectors are specific to the Read Humidity trigger. You'll be adding similar events for the Gas Resistance Read trigger.



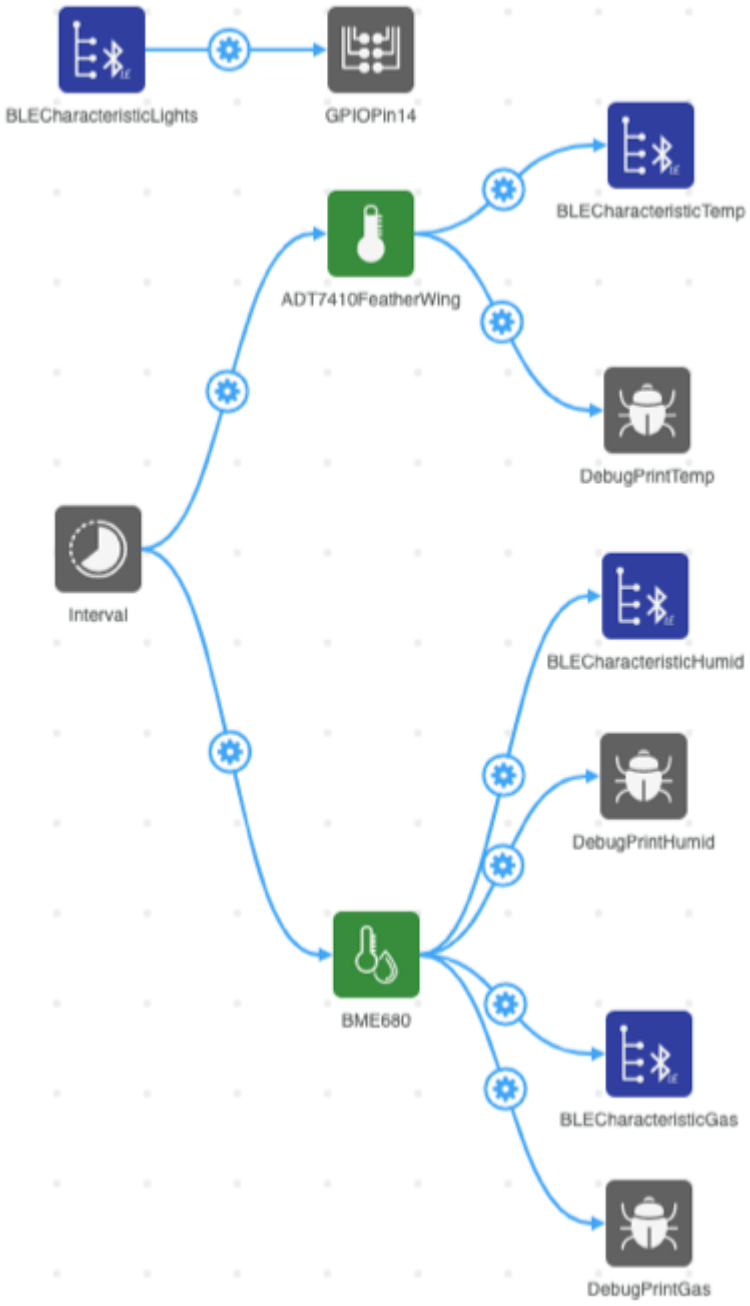


Let's add some elements to receive and print the data from the sensor.

- From the element toolbox within IoT Studio's Embedded workspace, click the BLE GATT Characteristic Element to add it to the Workspace.
- Change the name of this element to BLECharacteristicGas
- From the element toolbox within IoT Studio's Embedded workspace, click the Debug Print Element to add it to the Workspace.
- Change the name of this element to DebugPrintGas
- Modify the Prepend Text to Gas (ohms):

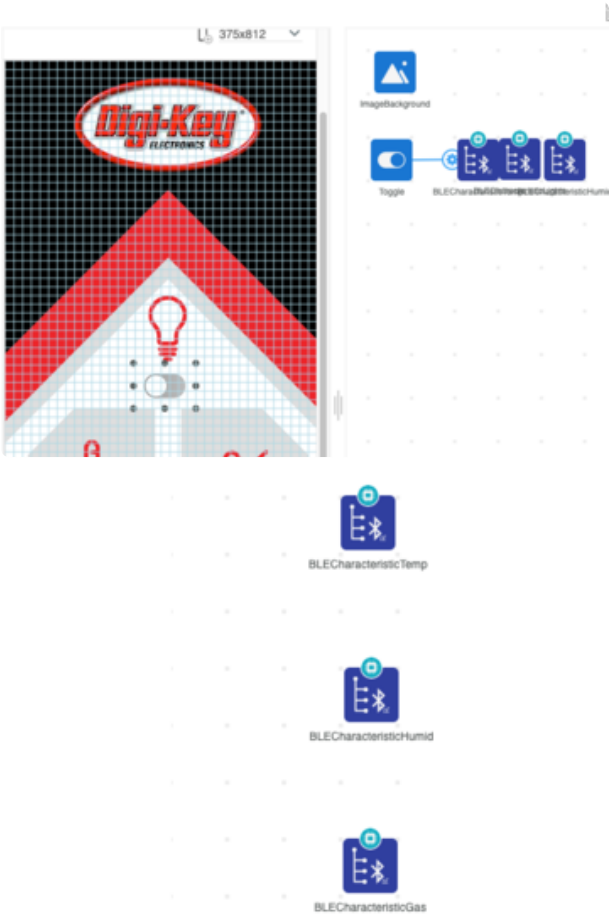
Once the Elements are configured, make a connection between the BME680 Element and both elements.

Make sure your embedded workspace looks like the following before moving on.

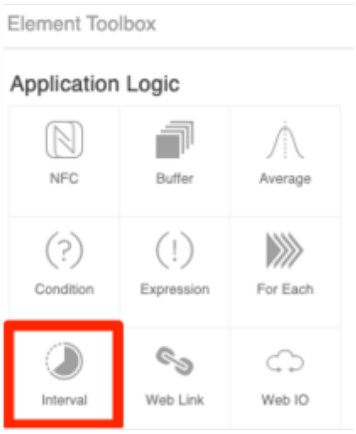


Step-by-Step: Application Workspace

The Mobile Application allows you visually display the sensor values. You'll add label elements to the interface to display the values read from the sensors.

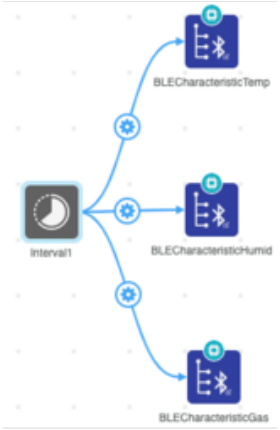


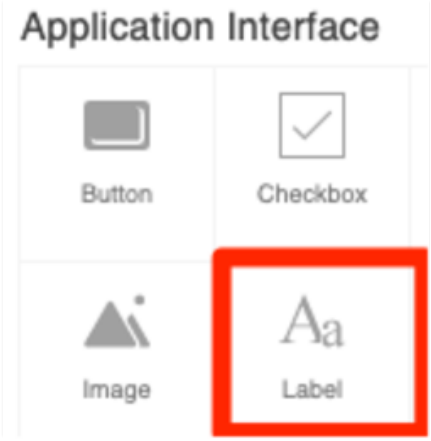
Navigate to the Application Tab. The BLE Characteristics you created in the embedded tab may be cluttered. Move them around to organize your workspace.

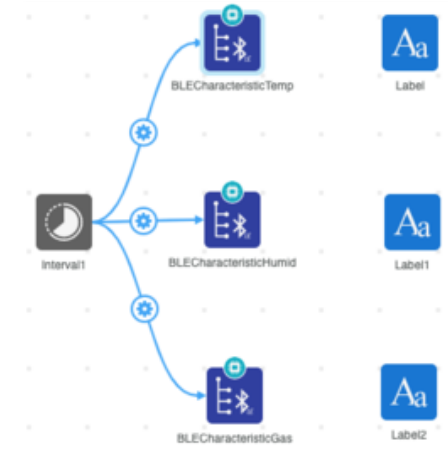


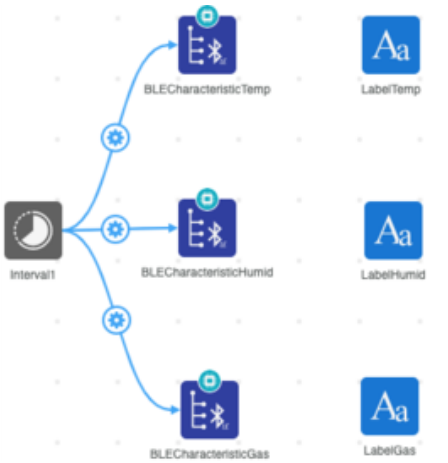
With an organized workspace, you're ready to start programming the application's logic.

- From the Element Toolbox, add an Interval Element.
- Click the Interval Element in the Application Workspace.
- From the Interval Element's Properties, click Add Event.
- Connect the Interval Element to the BLECharacteristicTemp Element
- Connect the Interval Element to the BLECharacteristicHumid and BLECharacteristicGas elements.









You'll want to add labels to display the values read by the BLE Characteristic Elements.

- From the Element Toolbox, click the Label button three times to add three Label elements.
- Rename the first label element to LabelTemp
- Rename the second label element to LabelHumid
- Rename the third label element to LabelGas



Let's add a connector between the BLE Characteristic Element and the Label Element. This will set the label's text to the value of data read by the BLE Characteristic.

- Click the BLECharacteristicTemp element to bring up its properties.
- Click Add Event
- Add a connector between BLECharacteristicTemp and LabelTemp.

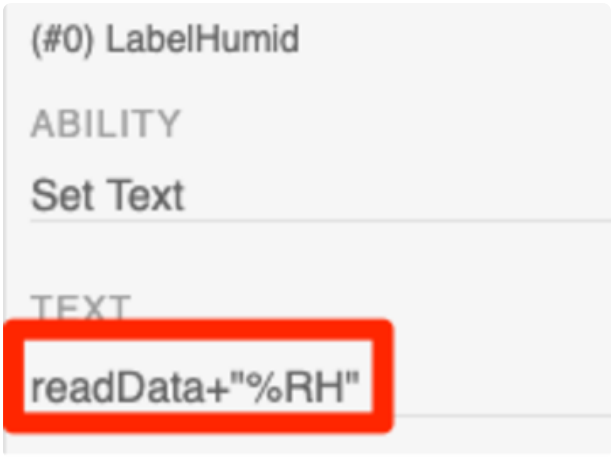


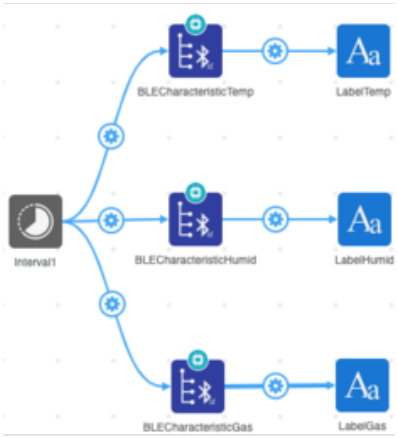
Next, you'll format the temperature to display on the label with its value and respective unit. The label's text can be formatted using Javascript.

- Click the connector between the BLECharacteristic Element and the Label element to bring up its properties.
- Under abilities, change the text to readData+"°C"



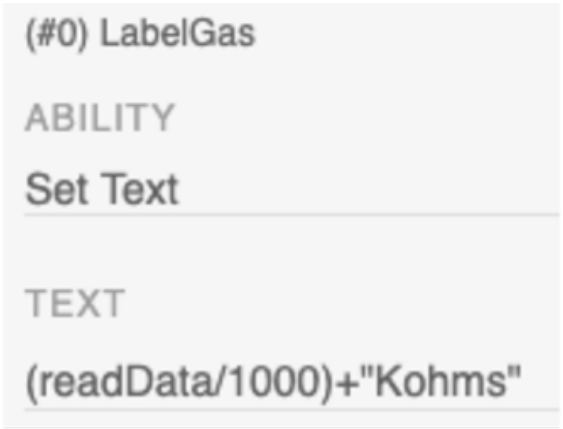
- Add a connector between BLECharacteristicHumid and LabelHumid
- Click the connector between BLECharacteristicHumid and LabelHumid to bring up the connector's properties.
- Under abilities, change the text to readData+"%RH"





Finally, you'll set up the connection between the gas sensor reading and its label on the mobile application.

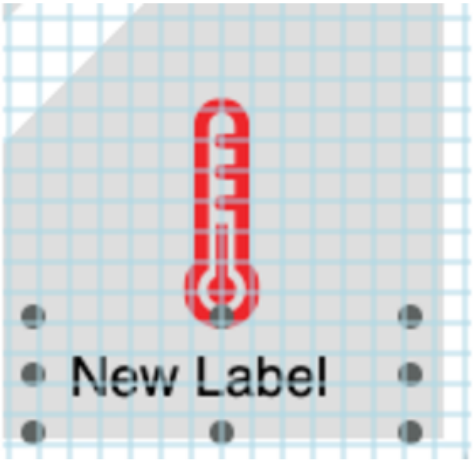
- Make a connection between BLECharacteristicGas and LabelGas
- Click the connector between BLECharacteristicGas and LabelGas to bring up the connector's properties.
- Under abilities, change the text to readData/100+"Kohms"

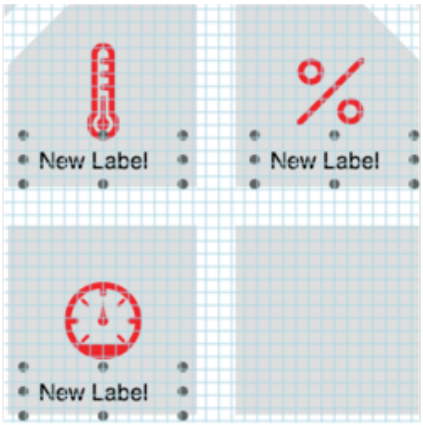


Let's take a short break from configuring our application's logic and configure the application's design.

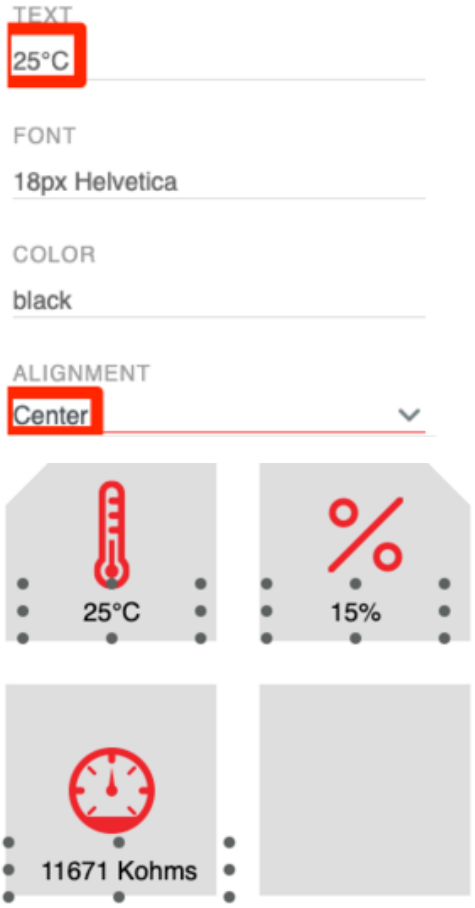


Click and drag LabelTemp underneath the thermometer icon.





- Bring the humidity label to the front and move it underneath the % symbol icon
- Bring the gas label to the front and move it underneath the gauge icon

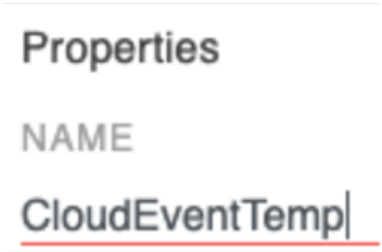


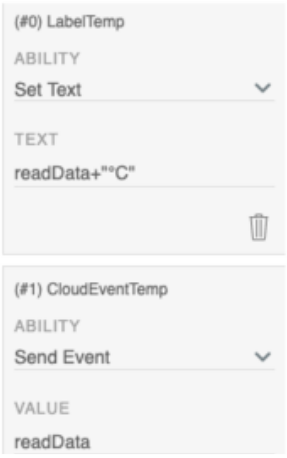
Let's add some placeholder text to our interface and center the labels to format our text

- Click LabelTemp to bring up the element's properties
- Change the label's text to 25°C
- Change the label's alignment to Center

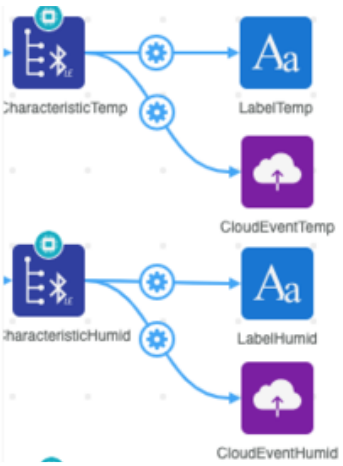
Repeat this for the humidity and gas labels.

While data is will display on the application, this data is not stored. To store this data, you'll need to add Cloud elements to our Application Workspace. These elements will create mirrored elements in the Cloud Workspace.

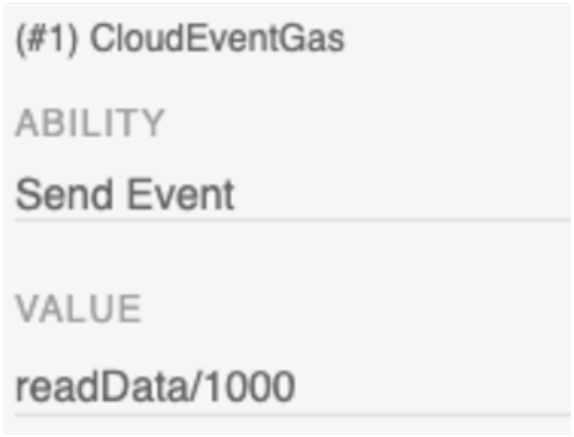






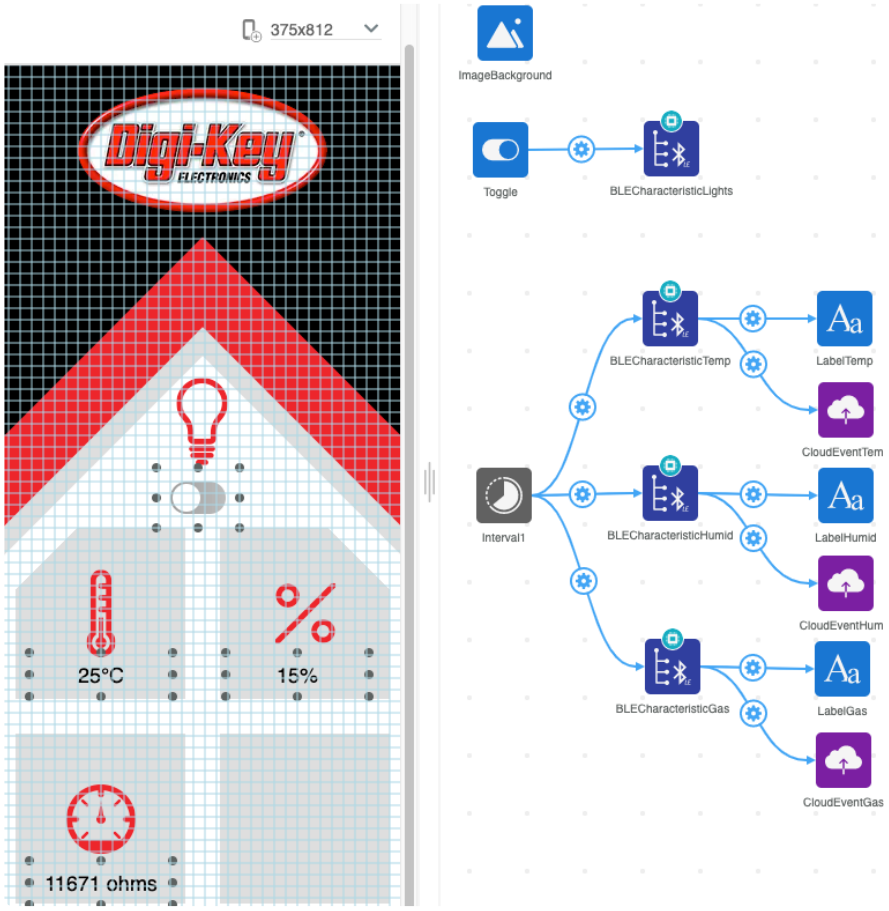


- From the Element Toolbox, click Cloud Event
- Rename the Cloud Event to CloudEventTemp
- Click BLECharacteristicTemp to bring up its properties.
- Click Add Event.
- Add a connector between BLECharacteristicTemp and CloudEvent
- Create a new element named CloudEventHumid and connect it to BLECharacteristicHumidity



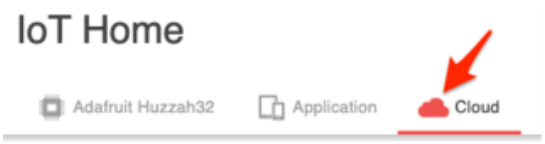
- Create a new Cloud Event element
- Name the element CloudEventGas
- Add a connector between BLECharacteristicGas and CloudEventGas
- Click the gear element on the connector.
- Change the value to readData/1000

You've programmed your mobile application to read values from BLE Characteristics and designed and interface to display these formatted values. Make sure your interface looks similar to the following screenshot before moving on.



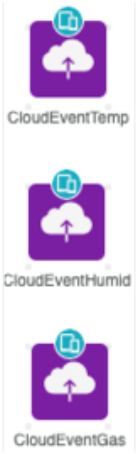
Step-by-Step: Cloud Workspace

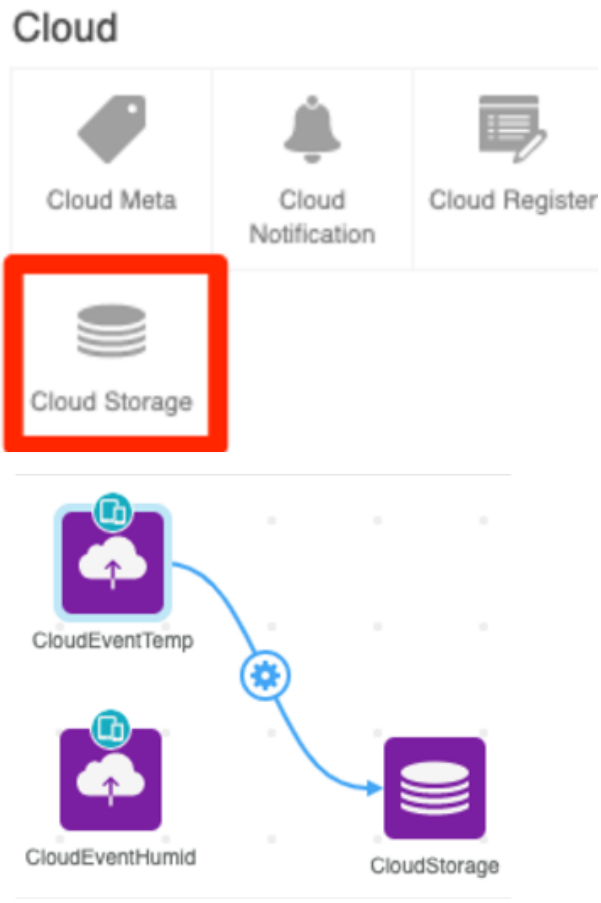
The Cloud tab is a workspace in DK IoT Studio for enabling an IoT project's connectivity over WiFi with cloud services (Digi-Key IoT Cloud or Amazon Web Services).



- Click the Cloud Tab underneath your project's name to be brought to the Cloud Workspace.
- Your cloud workspace should contain the three cloud events you created in the application workspace: CloudEventTemp, CloudEventHumid, CloudEventGas.

You may want to organize the elements in the Cloud Workspace.

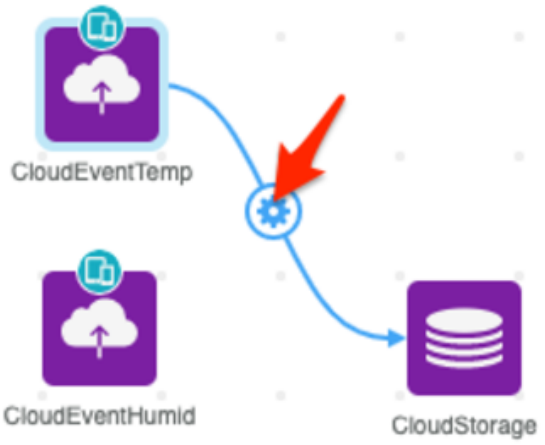




You'll need somewhere to send this data. Luckily, Digi-Key IoT offers up to 10 megabytes of cloud storage per device. To add data to this cloud storage, use the Cloud Storage Element.

- From the Cloud Element Toolbox, click the cloud storage element to add it to your Cloud Workspace.
- On the Cloud Workspace, click CloudEventTemp. Then, click Add Event.
- Make a connector between the CloudEventTemp and CloudStorage elements

While you could send raw data to the cloud storage element, it would be hard to discern which data is the temperature data and which is the gas data. To help identify which value is being sent to the cloud storage, you'll associate each value with some metadata (information about the data).



Click the connector between the cloud event and the cloud storage.

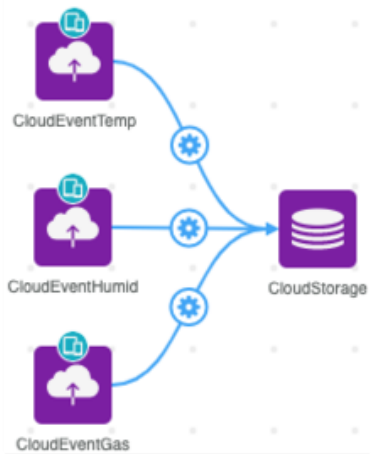
Under the Add Data ability, change the value to {temperature:value}





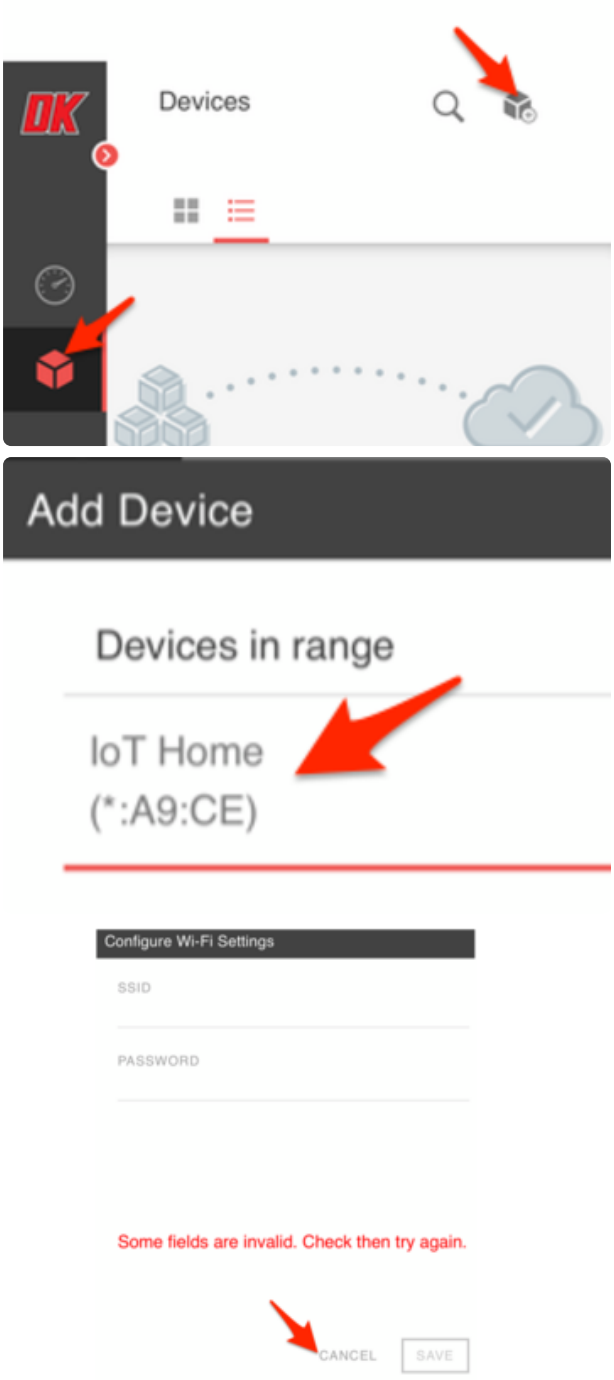
Repeat the steps above to add connectors and labels for the humidity and gas values.



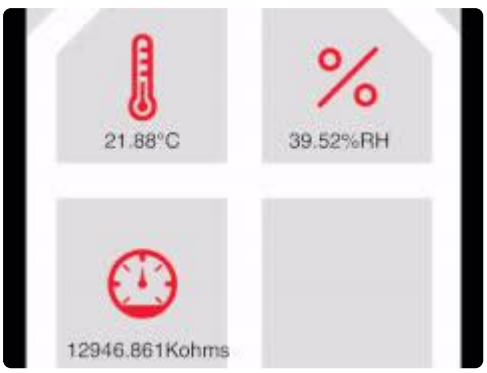


Code Usage

Compile the project. Then, upload the project to the Feather HUZZAH32.



- Open the DK IoT Studio Mobile App on your device.
- Tap the Devices tab on the sidebar
- Add a new device to the devices page by tapping the + icon
- Your IoT Home should appear as a device in range. Tap the IoT Home Project.
- On the Configure WiFi Settings screen, tap CANCEL



The Application's interface should load and populate with values from the sensors.

Try breathing directly on the Sensor FeatherWing's ADT7410 sensor to increase its temperature or humidity. You should observe the values on the application view change.

Keeping it Cool

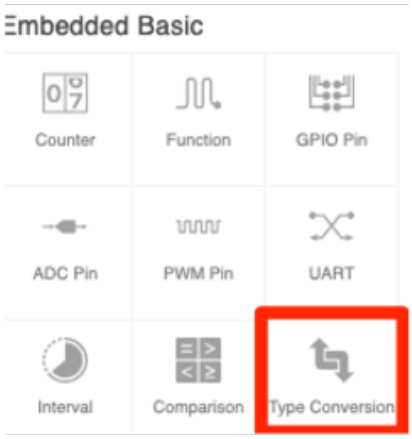
Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

While monitoring your home's temperature is useful, your home will not be able to respond to changes in temperature. While you could open a window (or remove the roof) to air it out, it'd be useful to control the fan inside your home to regulate temperature.

You'll be building code to self-regulate the temperature inside the home to your preferred temperature.

Step-by-Step: Embedded Workspace

This section builds on the previous section, Monitoring your Home. You'll be adding a comparison element to evaluate the temperature and a GPIO element to control the relay's state.



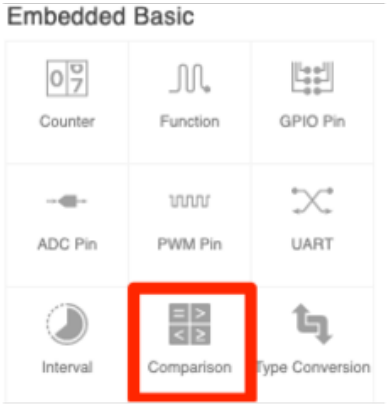


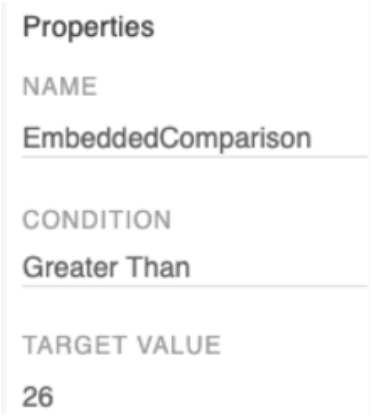
The ADI FeatherWing reads the temperature as a floating point value. At the time of writing, Digi-Key IoT Studio is unable to use floating point values with comparison elements. To resolve this, convert the floating point value into an integer value.

- From the Embedded Workspace's Element Toolbox, click the Type Conversion element.
- Click the DataTypeConvert Element to open its properties.
- In the Type Conversion Element's properties, change the output type to Integer



Add a new connector between the ADT7410FeatherWing Element and the DataTypeConvert element.





With the temperature reading converted to an integer, it's time to add some logic to our embedded firmware.

- From the Embedded Workspace's Element Toolbox, click the Comparison element.
- Change the Comparison Element's condition to Greater Than
- Change the Target Value to the maximum temperature value.
    - If the temperature sensor reads a value greater than the Target Value, the relay will switch and turn on the motor.

Add a new connector between the DataTypeConvert Element and the EmbeddedComparison Element.

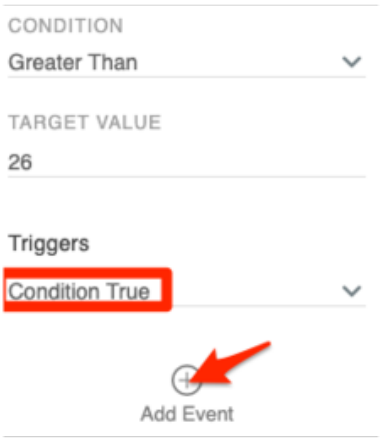




Next, you need to add a GPIO element to control the relay. The relay is connected to GPIO Pin 33 on the ESP32 Huzzah.

- From the Embedded Workspace's Element Toolbox, click the GPIO Pin element.
- From the GPIOPin Element's properties, change the Name to GPIOPin33
- Change the GPIO Pin to IO33





You'll turn on the relay when the Comparison Element evaluates that the expression is True.

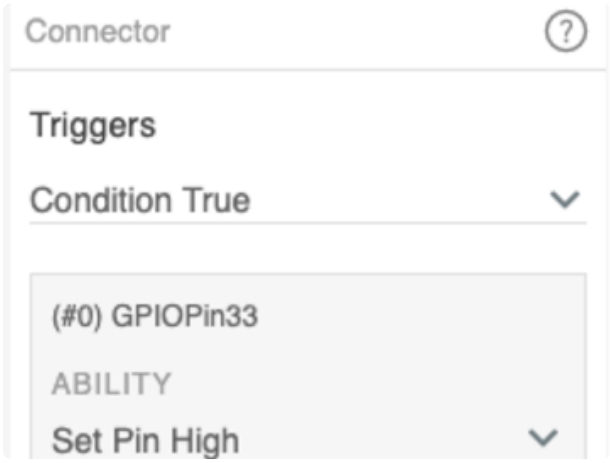
- Click the EmbeddedComparsion Element to open its properties
- Under Triggers, ensure the trigger is set to Condition True.
- Add a connector between the EmbeddedComparison Element and GPIOPin33

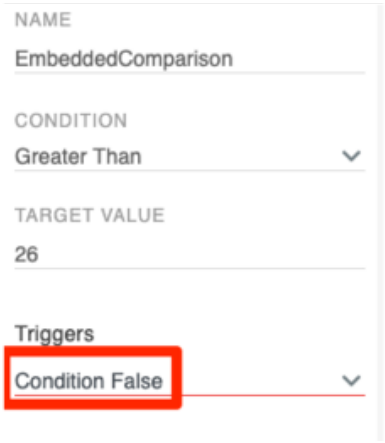




To turn on the relay if the Target Value is reached,

- Click the connector between the comparison element and the GPIO pin.
- Change the GPIO Pin's Ability to Set Pin High

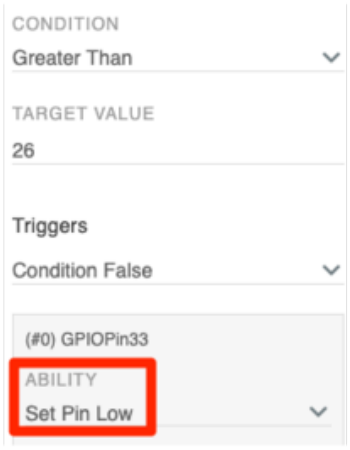




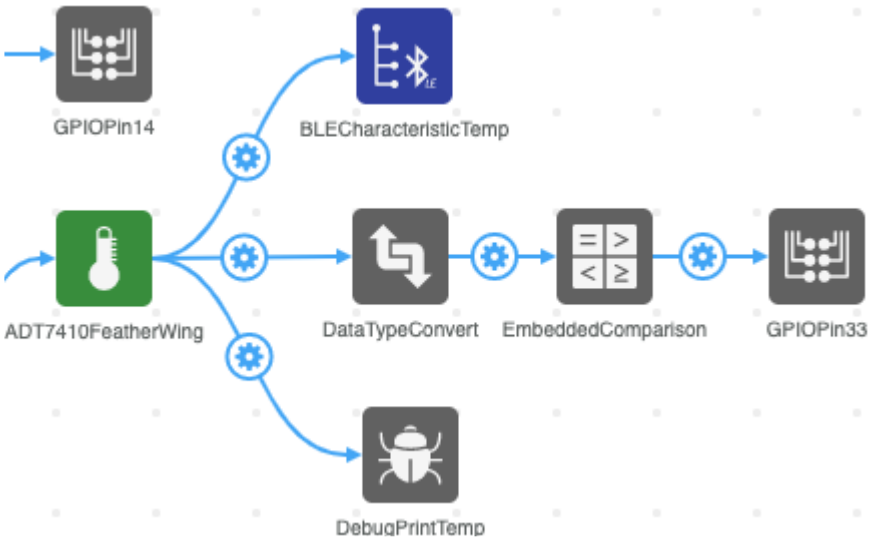
Our IoT Home turns on the fan when the temperature increases past a target value, but what if it the home cooled off? You'll need to turn off the relay.

- Click the EmbeddedComparsion element to open its properties.
- Switch Condition True to Condition False
- Add a connector between the EmbeddedComparison Element and the GPIO Pin Element.
- Change the GPIO Pin's ability to Set Pin Low





Your "code" for the ADT7410 temperature sensor should look like the following...

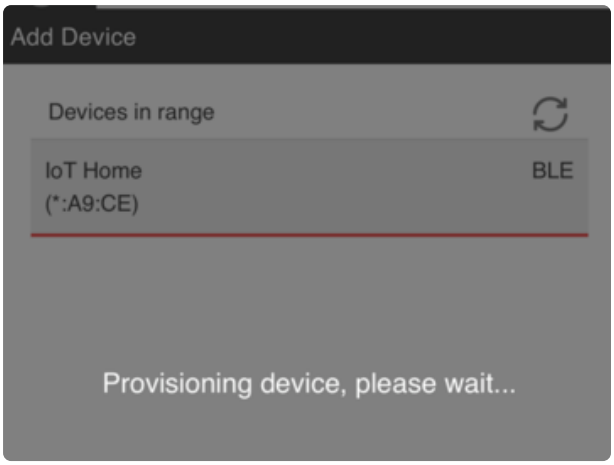


Code Usage

Compile the project. Then, upload the project to the Feather HUZZAH32.



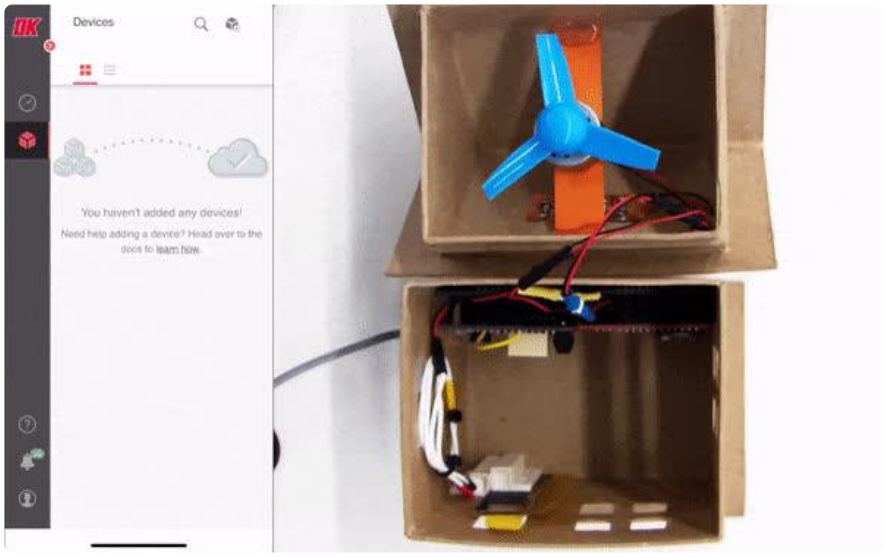
- Open the DK IoT Studio Mobile App on your device.
- Tap the Devices tab on the sidebar
- Add a new device to the devices page by tapping the + icon
- Your IoT Home should appear as a device in range. Tap the IoT Home Project.
- On the Configure WiFi Settings screen, tap CANCEL





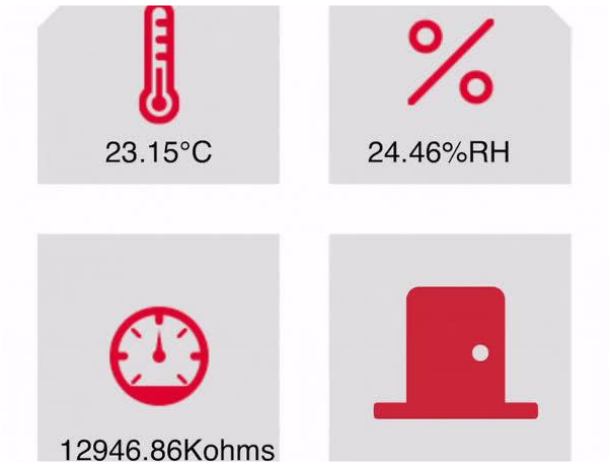
Once connected, the application view will display the temperature sensor's value. Try holding your finger over the ADT7410 (or use a hair-dryer outside to the IoT Home) turn on the fan.

The fan will turn off after the temperature decreases below its target value.



Door Alarm

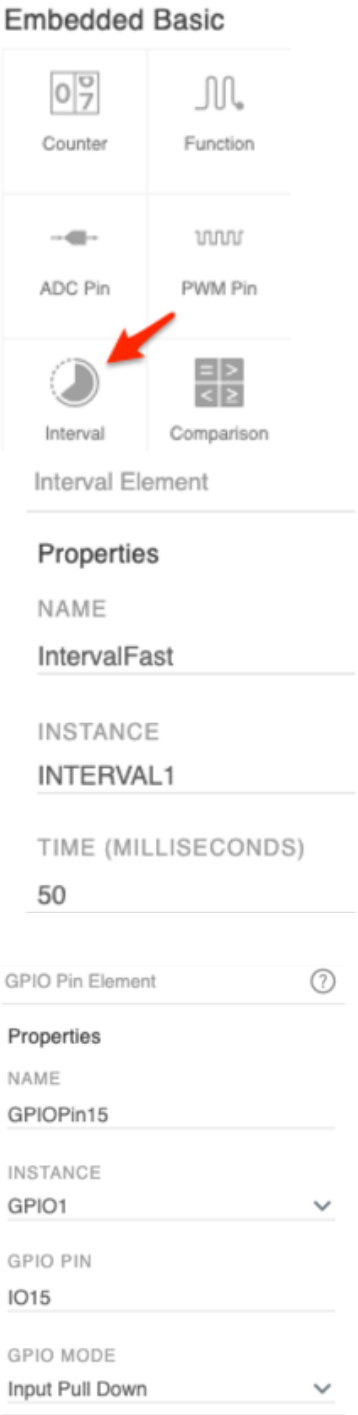
Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463



Intruder alert! You'll be using the magnetic door sensor to detect when someone opens or closes the front door. You'll program your IoT Home to sound an alarm when someone opens the door.

But what if someone enters our home while you're away? You'll add the ability to monitor your home's status remotely to your mobile app.

Step-by-Step: Embedded Workspace



If someone enters your house, you would want the home security system to notify you as quickly as possible. You'll be adding an interval with a shorter timer to the embedded workspace to read the status of the door sensor.

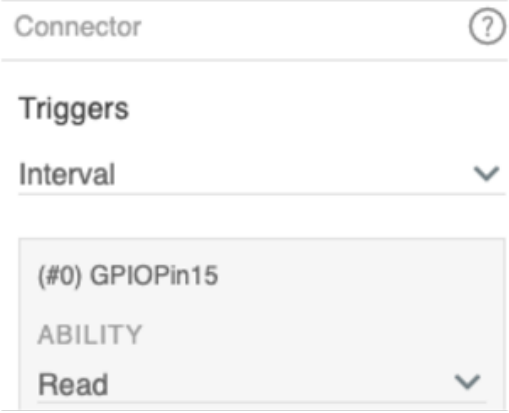
- From the Embedded Workspace's Element Toolbox, click the Interval element.
- Name the interval IntervalFast
- In the Interval Element's properties, set the time to 50 milliseconds

You'll need to read the state of the GPIO Pin connected to the door sensor, GPIO Pin 15.

- Add a new GPIO Element to the Workspace
- Change the name of the GPIO Pin Element to GPIOPin15
- Change the GPIO Pin to IO15
- Change the GPIO Mode to Input Pull Down



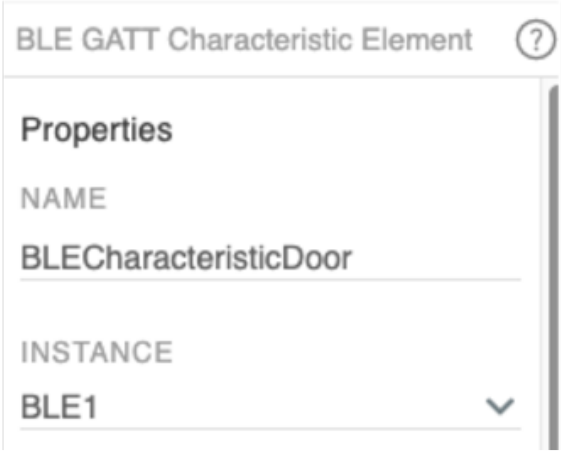
- Add a connector between IntervalFast and GPIOPin15
- Change the connector's ability to Read

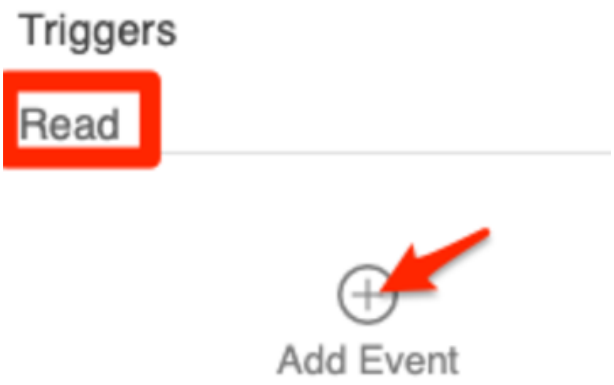




Next, you will write the state of the door to a BLE Characteristic so that it can be read in the mobile application.

- Create a new BLE GATT Characteristic element
- Name the element BLECharacteristicDoor
- Change the Read Data Type to Boolean
- Change the Write Data Type to Boolean
- Change the Notify Data Type to Boolean

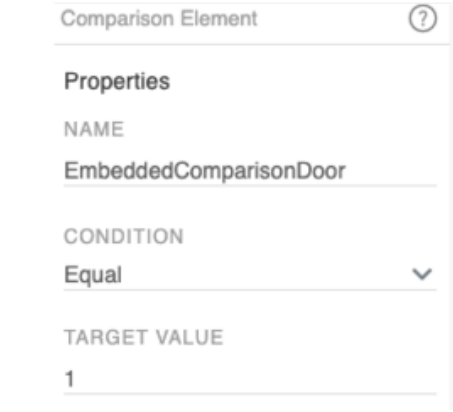


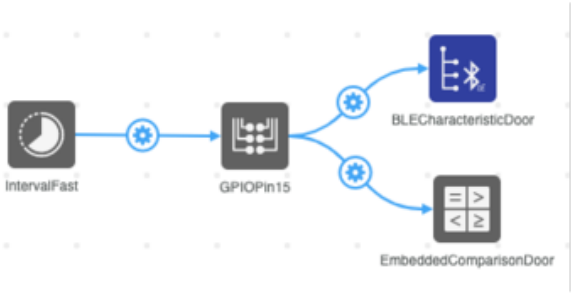


- In the GPIOPin15 Element's properties, change its trigger to Read
- Add a new connector between GPIOPin15 and BLECharacteristicDoor









Next, let's add an element to evaluate if the door is open or closed.

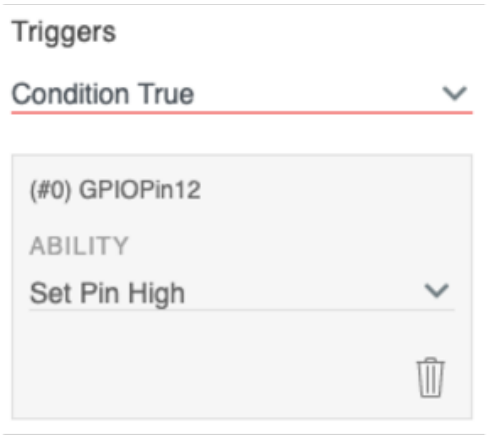
- From the Element Toolbox, select the Comparison Element
- In the comparison element's properties, change the element's name to EmbeddedComparisonDoor
- Change the target value to 1
- Then, add a connector between the GPIOPin15 element and the embedded comparison element.

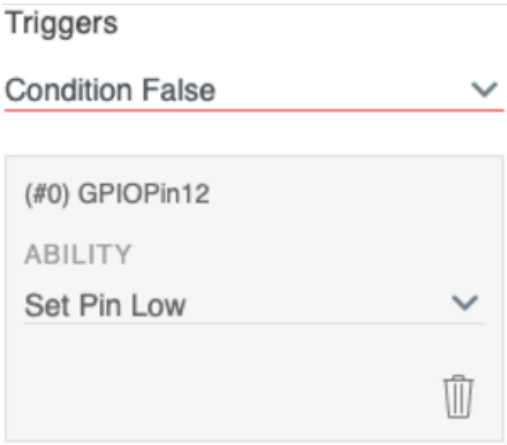


You need a way to notify our neighbors that an intruder has entered our home. Luckily, you installed a buzzer (conveniently connected to ESP32 Pin 12) in the smart home hub!

- Add a GPIO Pin Element
- Name the element GPIOPin12
- Change the GPIO Pin to IO12







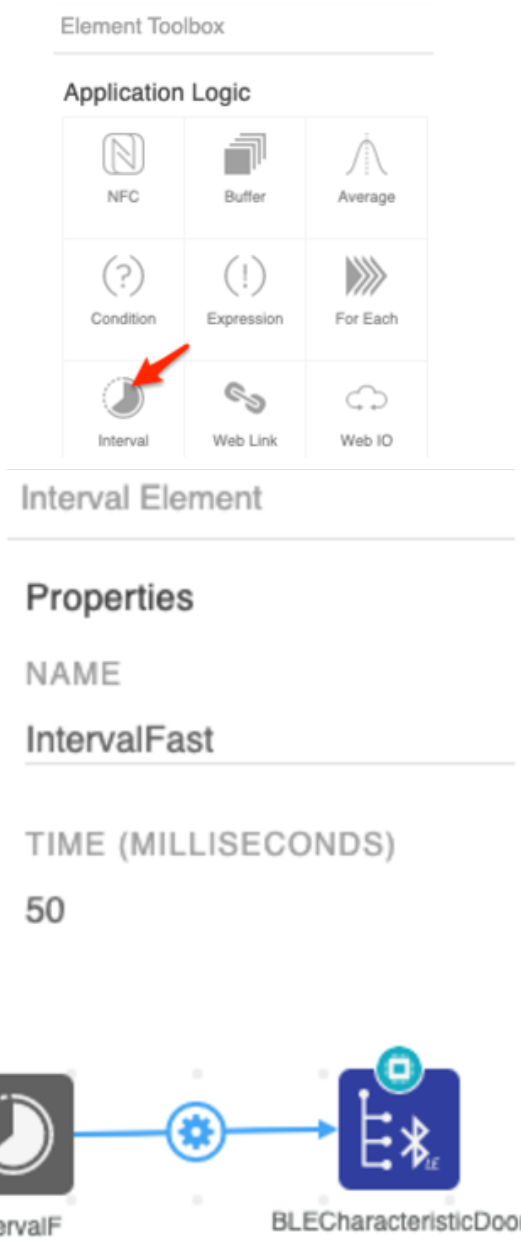
Using the comparison element, your code will evaluate if the door is open or not. Based on the door's state, the buzzer will turn on or off.

- Click the EmbeddedComparisonDoor element
- Under triggers -> Condition True, click Add Event
- Add a new connector between the embedded comparison element and GPIO Pin 12.
- Change the connector's ability to Set Pin High
- Next, change the EmbeddedComparisonDoor Element's trigger to Condition False
- Change the connector's ability to Set Pin Low

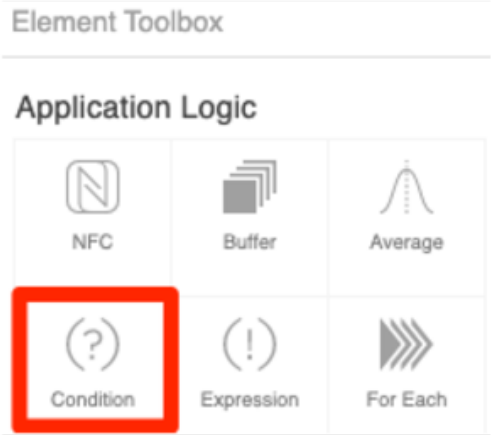


Step-by-Step: Application Workspace

Let's add an icon to the mobile application to let us know if the door is open, even if you're away from home.



- From the Application Workspace, add a new interval element
- Name this interval IntervalFast
- Change the time to 50 milliseconds
- Add a new connector between IntervalFast and BLECharacteristicDoor



To evaluate if the door is open or closed, a comparison element needs to be added.

- Add a Condition Element to the Application Workspace
- Add a new connector between the BLECharacteristicDoor and Condition elements



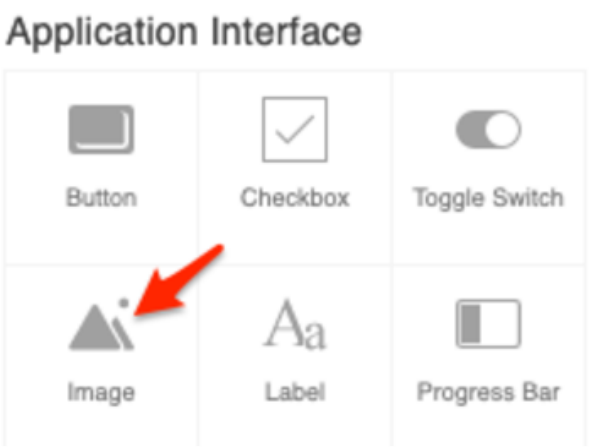
Download the two door images by clicking the buttons below. Save them to your desktop

Download door-closed.png

https://adafru.it/HEP

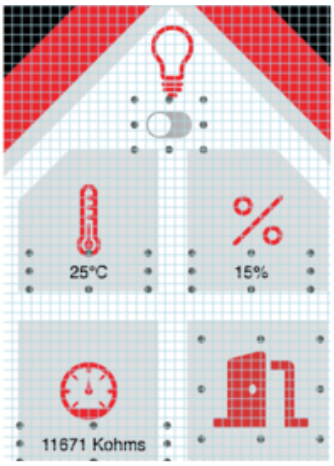
Download door-open.png

https://adafru.it/HEQ



Ok, here comes the tricky part. Two icons are used to display the door's status - one for if it is closed, and one for if it is open. This will look great in our application, but is a bit tricky to implement as you will need to build some code to show one icon at a time.

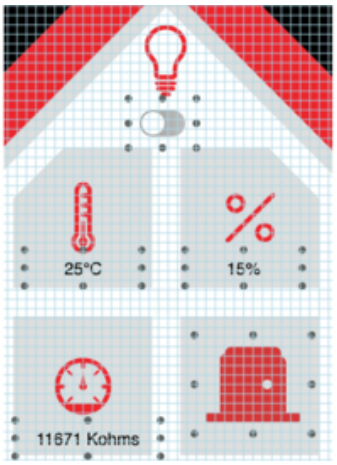
- Add two image elements to the Application Workspace
- Name the first image element ImageDoorOpen
- Underneath Image, click Browse.
- From the files you downloaded earlier, select Door-Open.png



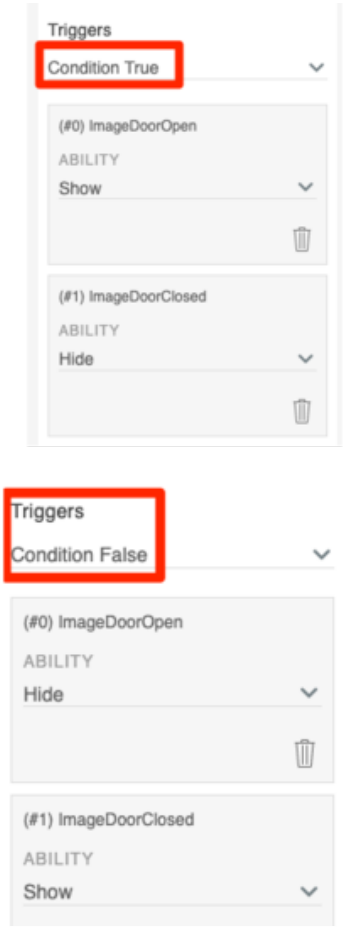
In the Application Builder Workspace, bring the door open image to the front of the application and move it to the bottom right of the application's interface.



- Click the second image element on the workspace.
- From the Image Element's properties, name the element ImageDoorClosed
- Click the Browse button
- From the files you downloaded earlier, select Door-Closed.png



In the Application Builder Workspace, bring the door closed image to the front of the application and move it to the bottom right of the application's interface. It should overlap the door open image.



Let's set up the condition element to selectively show and hide images based on the state of the door.

- From the Workspace, click the Condition Element.
- Under triggers, select Condition True
- Add a new connector between the Condition Element and ImageDoorOpen
- Change the ability of the connector to Show
- Add a new connector between the Condition Element and ImageDoorClosed
- Change the ability of the connector to Hide

Now, let's set up the condition element for the opposite condition.

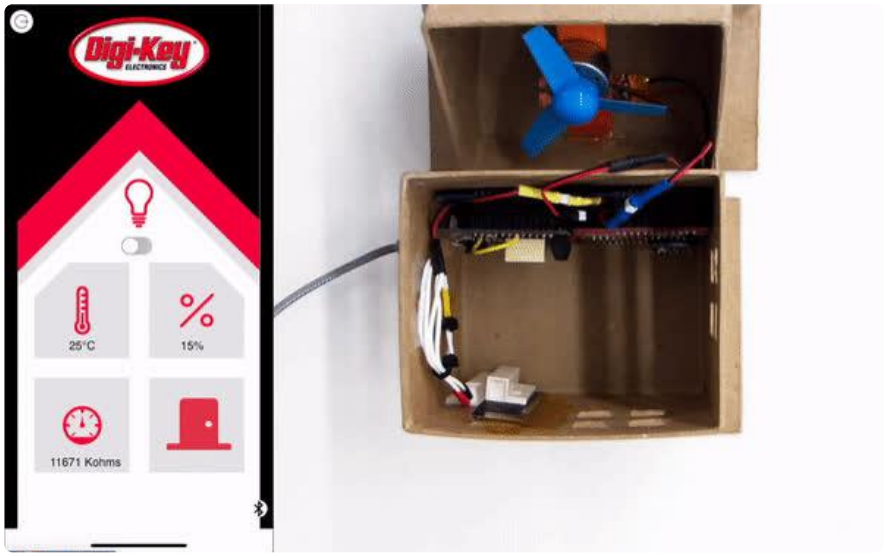
- Under triggers, select Condition False
- Add a new connector between the Condition Element and ImageDoorClosed
- Change the ability of the connector to Show
- Add a new connector between the Condition Element and ImageDoorOpen
- Change the ability of the connector to Hide

Code Usage

Compile the project. Then, upload the project to the Feather HUZZAH32.

Open the door on your IoT Home. You should hear a loud BEEEEEEP from the buzzer. The icon on the mobile application should show the door's state.

Closing the door on your IoT Home should turn off the buzzer. The icon on the mobile application should display the door's status.



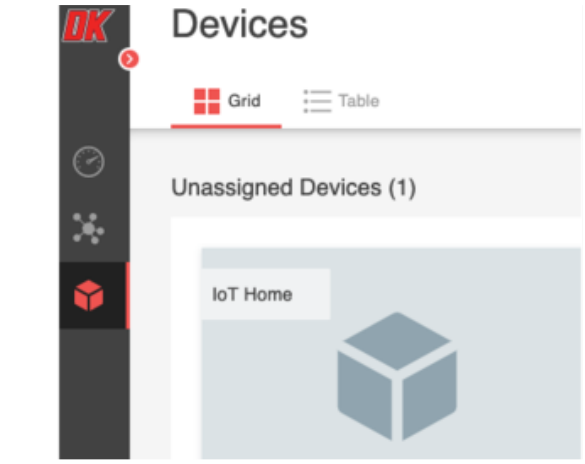
Dashboards

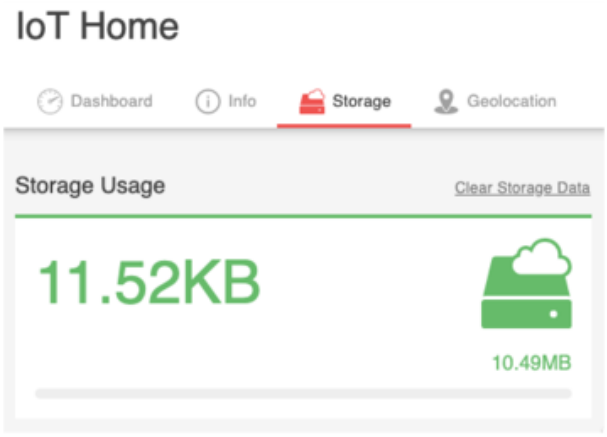
Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

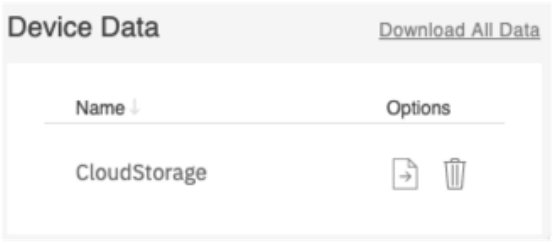
Now that data is streaming into both your device over Bluetooth and it's connected to Digi-Key IoT Cloud Storage, you can now visualize our home's data in real-time. While you could use an API or framework to display this data, there's an even simpler way.

Digi-Key IoT Studio is an all-in-one solution and includes a Dashboard front-end. You're going to build a dashboard to help visualize the sensors in your home over time. Just like before, there's no programming. You'll drag and drop widgets to create a dashboard.

As mentioned before, IoT Studio provides 10.49MB of data per-device. You can verify how much data your device is using by navigating to the device panel.







Before jumping into building a dashboard, it's important to know where our data is located. IoT Studio provides 10.49MB of data per-device, for up to 5 devices.

To verify how much data your IoT Home is using,

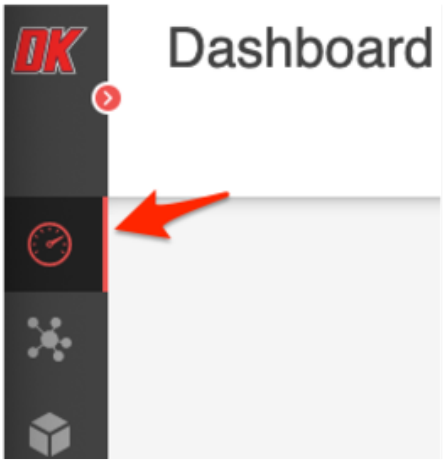
- Navigate to the Device Manager
- From the Device Manager, click the IoT Home device.
- Click Storage from the device to navigate to the device storage page.

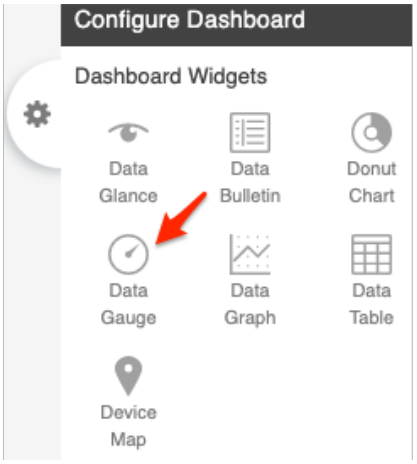
The storage page will display how much data your device uses and which database its stored on. Data is stored in the CloudStorage element you created in the Monitoring your Home page.

You may also download or delete all your device's data at any point. Also, you may verify how much data your device is using by navigating to the device panel.

Step-by-Step: Building a Dashboard

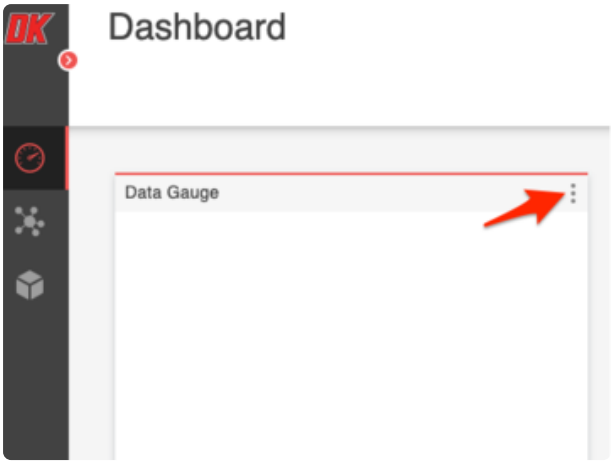
Digi-Key IoT Studio's Dashboard "allows you to display your devices data through a variety of widgets". Widgets are visual components that allow you to display device data in different forms. There are widgets for displaying data as a gauge, a text block, a graph, or even as a pie chart.





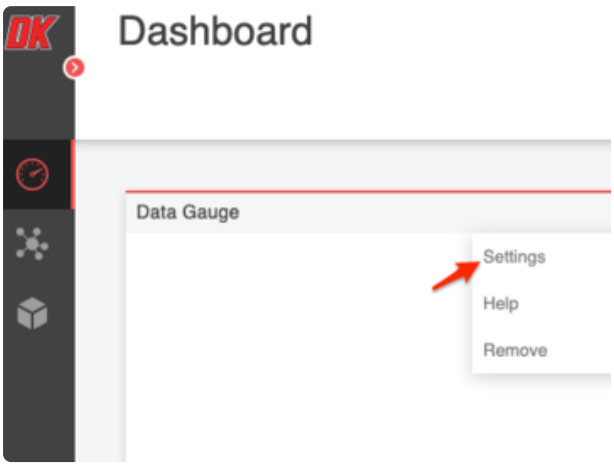
Let's start by adding a gauge widget to display the temperature from the ADT7410 temperature sensor in the IoT Home.

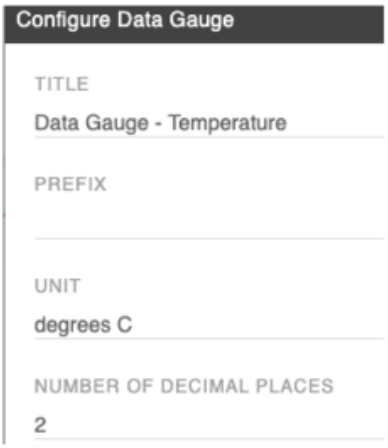
- From Digi-Key IoT Studio, navigate to the Dashboard page.
- Click the Gear button on the right-hand side of the page to bring up the configuration menu.
- Click the Data Gauge widget.



A new Data Gauge widget will appear on your dashboard.

- Click the ... button to bring up a dropdown for configuring its settings.
- Click Settings to bring up the widget's properties

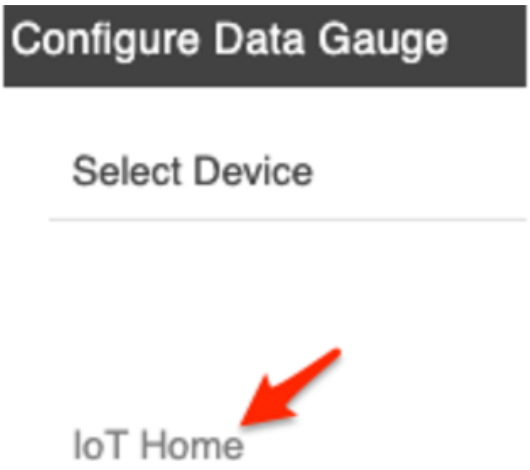




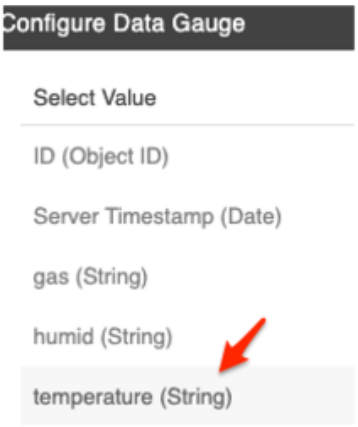
In the widget's properties window,

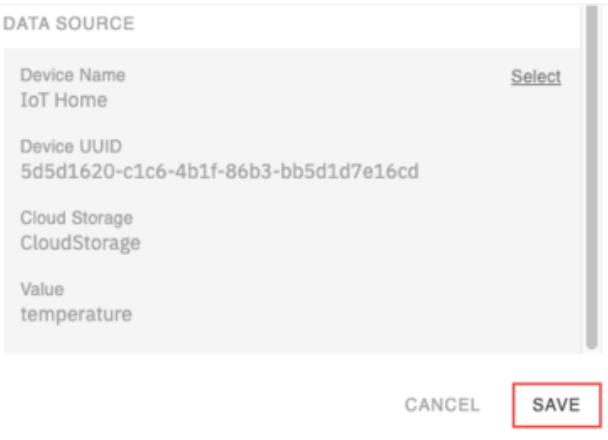
- Change the title to Data Gauge - Temperature
- Change the unit to degrees C
- Change the number of decimal places to 2
- Under Data Source, click Select





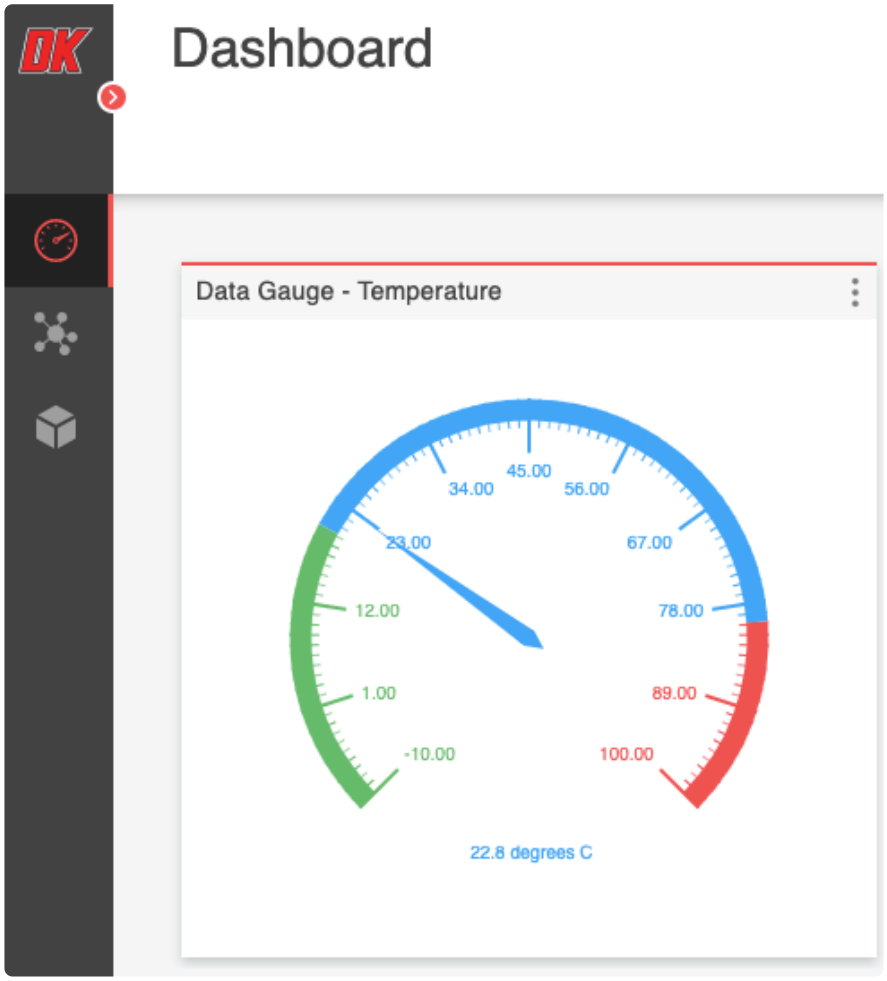


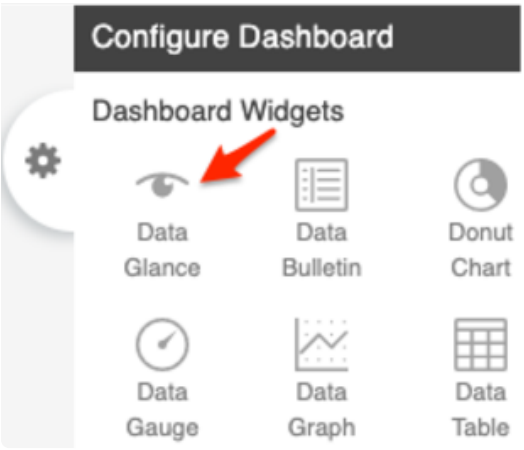




- Select IoT Home as the device
- Select CloudStorage as the Cloud Storage Provider
- Select Temperature as the value
- Click Confirm to configure the data source.
- Then, click Save

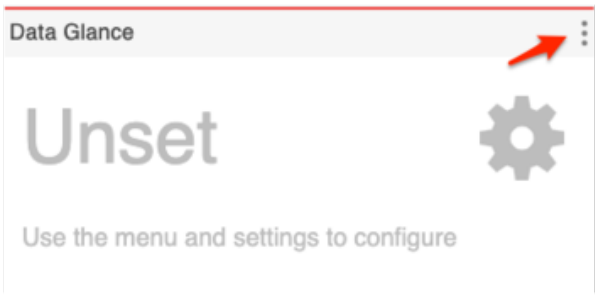
The gauge will automatically appear and the needle will move, reflecting the value read by your temperature sensor.

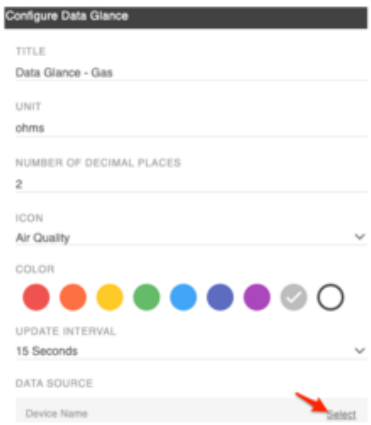




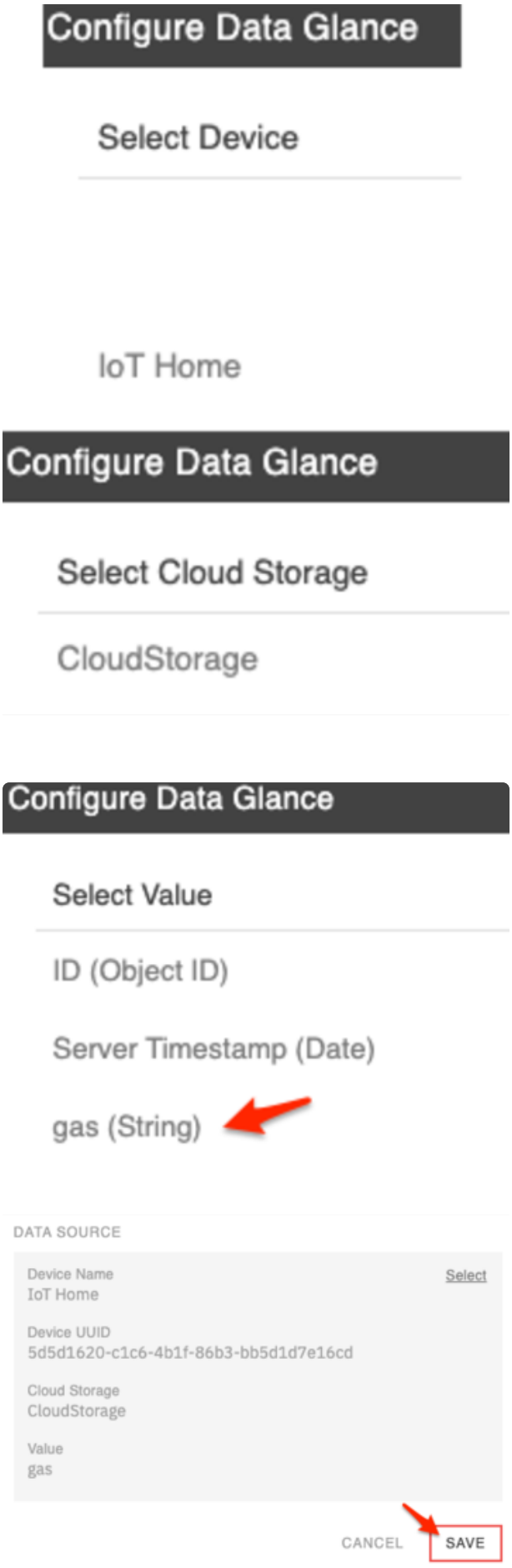
Let's add another element type to display the resistance of the gas sensor.

- From the Configure Dashboard menu, select the Data Glance widget.
- Click the ... button to bring up a dropdown for configuring it settings.
- Click Settings



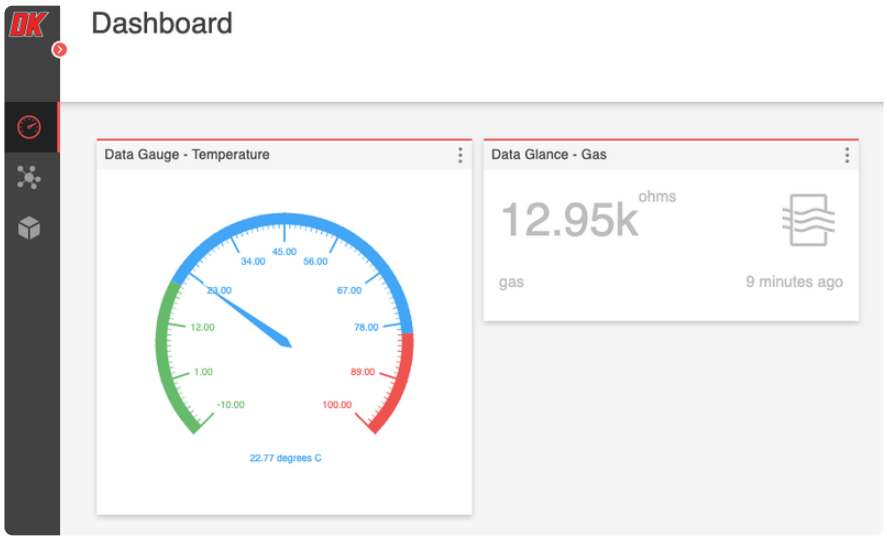


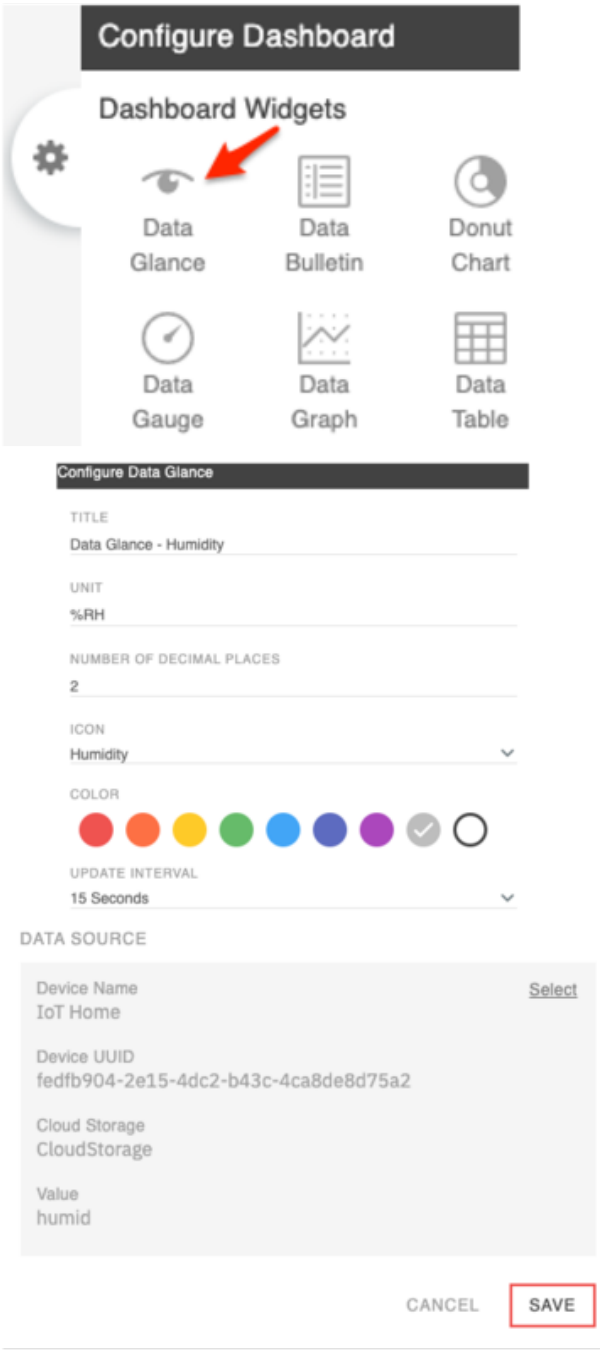
- Change the title to Data Glance - Gas
- Change the unit to ohms
- Change the icon to reflect what is being measured measuring. In this case, it's the air quality.
- You may also optionally change the update interval, ranging from 1 to 30 seconds.
- Under data source, click select



- Under Select Device, click IoT Home
- Under Select Cloud Storage, click CloudStorage
- Under Select Value, click gas (string)
- Click Save

The data glance widget should populate with the IoT Home's gas reading.

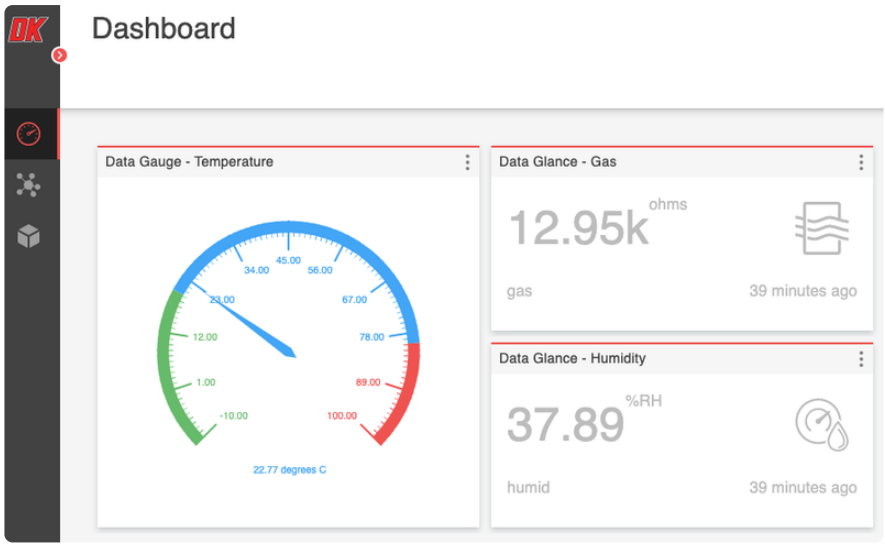




Let's complete this dashboard by adding a second Data Glance widget to display the relative humidity in the IoT Home.

- Add a Data Glance Widget to your dashboard
- In the Data Glance configuration, change the title to Data Glance - Humidity
- Change the unit to %RH
- Change the icon to Humidity
- Set the data source to humid
- Click Save

You've successfully created a dashboard to display data about your IoT Home's sensors! You can view this dashboard on any web browser or mobile device, from anywhere in the world.



For more information about the DK IoT Studio Dashboard and its elements, visit the developer documentation pages (https://adafru.it/HmC).

Conclusion

Digi-Key IoT Studio Web and Mobile Applications have been discontinued by Digi-Key. The links and instructions in this guide may be outdated or invalid. For support and more information about the future of the Digi-Key IoT Studio application, please visit https://forum.digikey.com/t/dk-iot-studio-has-migrated/8463

We've created a complete home automation system with our IoT Studio Kit - but why stop there? You can take this project further by utilizing the kit's Feather Huzzah and sensors in the real world. For example - we have a guide about using a Huzzah with the door sensor included in the kit to text you when a door in your home has been opened (https://adafru.it/jdq).

It's never been a better time to play with IoT - there are so many protocols, boards and sensors available. What only a few decades ago was literally space-age technology is now only a few cents on Digi-Key's website (https://adafru.it/lxf).

Even though this series is over - we're not saying good-bye forever. There's a wealth of resources on Adafruit, from the thousands of tutorials on the learning system (https://adafru.it/dlu), to our hundreds of Python and Arduino libraries, all free for you to use (https://adafru.it/aYH). Digi-Key has all the newest tools and parts (https://adafru.it/BJr), with links to app notes, CAD files and engineering resources when you're ready to go to production.

And if you're still in the prototyping phase, check out all the Adafruit breakout boards and Feathers that Digi-Key stocks, they're a great match for IoT Studio (https://adafru.it/IxA) - you could order this evening and get started by tomorrow!