Edge Fall Detection

Step-By-Step Tutorial

IoT Cart

Contents

[Change Log 2](#_Toc36825460)

[Purpose 3](#_Toc36825461)

[Azure Services Used 3](#_Toc36825462)

[Required Hardware 3](#_Toc36825463)

[How to set up hardware 4](#_Toc36825464)

[Setting up the Cloud 4](#_Toc36825465)

[Getting Ready 5](#_Toc36825466)

[How to do it 5](#_Toc36825467)

[How it works 6](#_Toc36825468)

[Appendix 6](#_Toc36825469)

# Change Log

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Update** |
| 4/3/2020 | Mike Roshak | Created template |
| 4/29/2020 | Mike Roshak | Created V1 |

# Purpose

This document will show you how to create a IoT Edge module that using an accelerometer, Raspberry Pi, and Azure resources such as IoT Hub, Stream Analytics, and Cosmos DB can detect falls.

# Azure Services Used

1. IoT Hub
2. Stream Analytics
3. Cosmos DB
4. Function App
5. Azure Container Registry

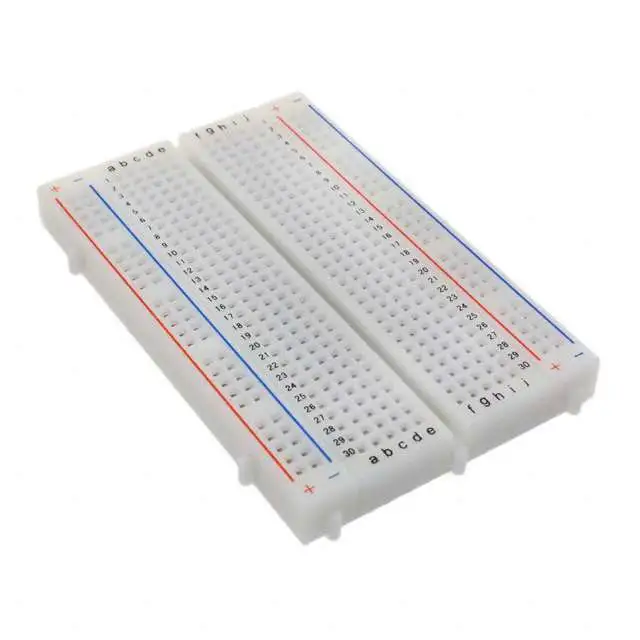
# Required Hardware

You will need:

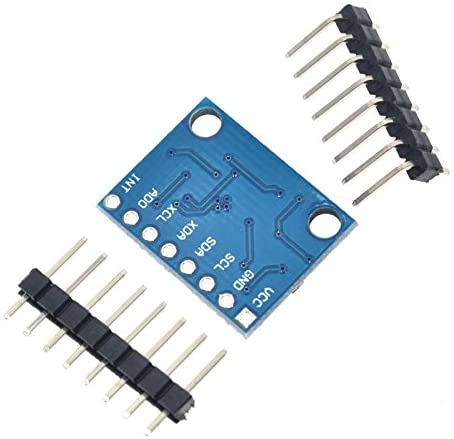
1. Raspberry Pi 4



1. Breadboard



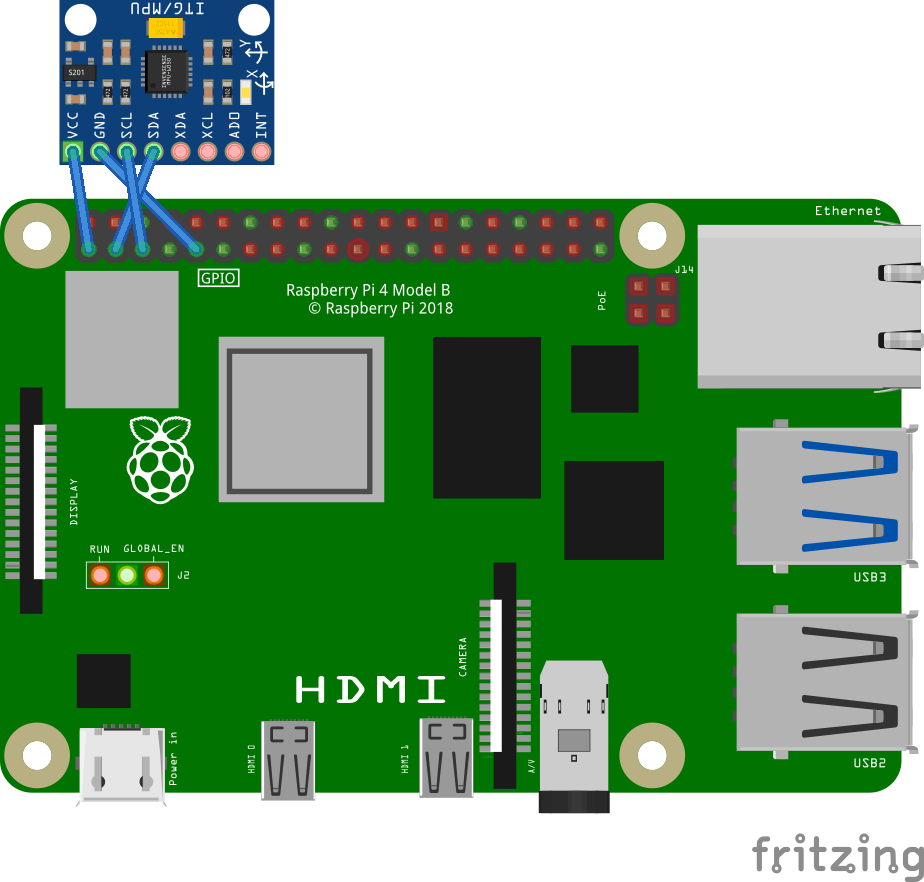
1. Wire
2. MPU6050 3 Axis Accelerometer Gyroscope Module 6 DOF 6-axis Accelerometer Gyroscope



# How to set up the hardware

To make this you will need to connect the motion sensor to the breadboard.

1. Connect the 3.3 volt pin of the Raspberry Pi to the VCC slot on the sensor.
2. Pin 3 of the Raspberry Pi to the SDA pin on the sensor
3. Connect pin 5 to the SCL pin on the sensor.
4. Finally connect the ground on the Pi to that of the sensor.



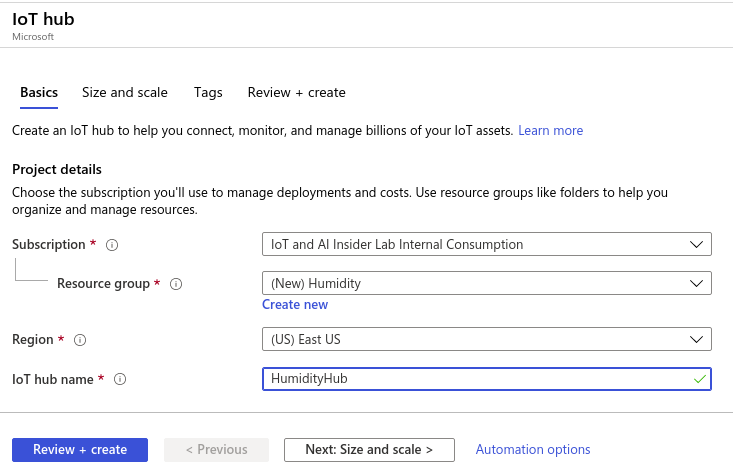
|  |  |
| --- | --- |
| **Raspberry Pi** | **MPU 6050** |
| 3.3V - Pin 1 | VCC |
| SDA - Pin 3 | SDA |
| SCL - Pin 5 | SCL |
| GND – Pin 9 | GND |

# Setting up the Cloud

## Set up IoT Hub

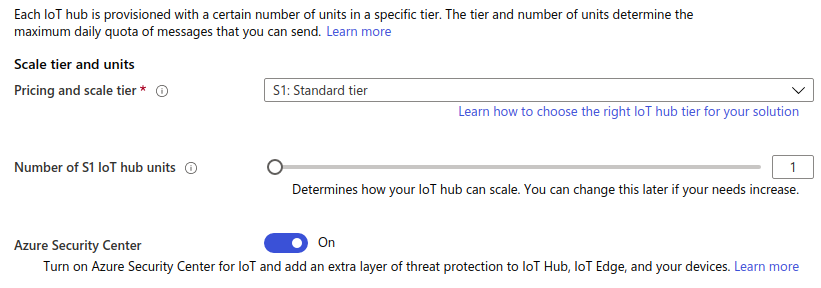
We are going to set up an IoT Hub with the default values.

**Step 1:** Set up the basics. Chose a resource group, name of project and subscription.



Step 2: Size and Scale:

For this exercise we are going to use S1 of IoT Hub and have 1 unit of S1. This will allow us to send 400,000 messages a day.

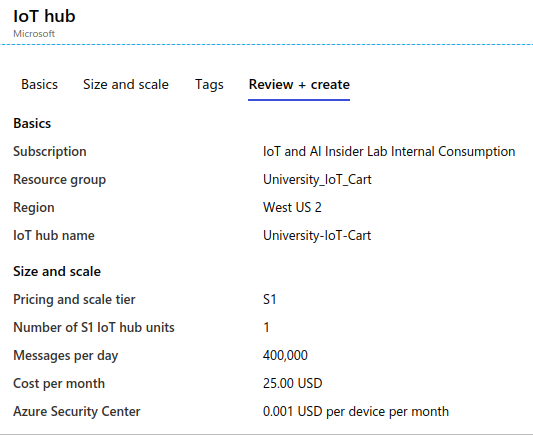


Step 3: Meta tags

Next, we are going to skip the tags section and go over to Create and Review.

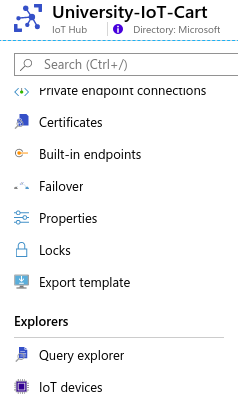
Step 4: Review and create

The final step is to review then create the IoT Hub Resource.



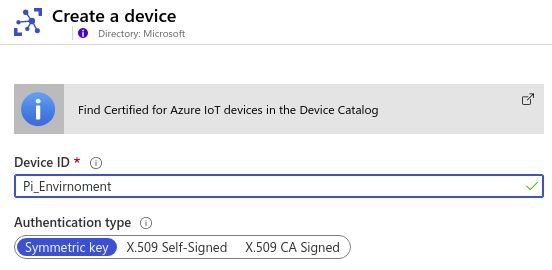
Step 5: Getting a device key

After creating the IoT Hub you will need to get an IoT Edge Connection string. To do this go into your ioT Hub and get the IoT Edge device key. You can do this by clicking on devices in the side menu and selecting “Edge Devices”.

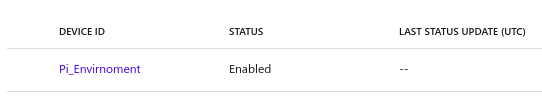


From here select Add Device.

Next give your device a name and select Save.



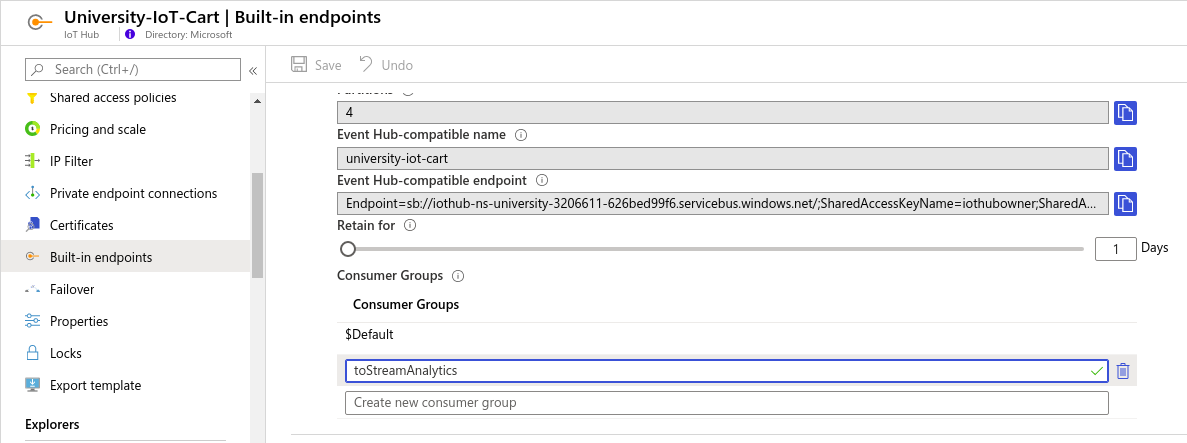
From here select the device in the device list



And save the device connection string for later use.

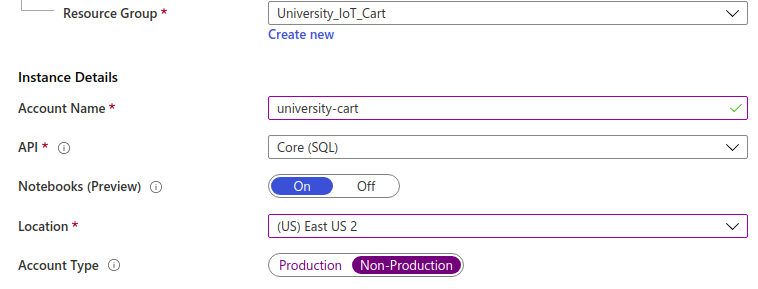
Step 6: Create a consumer group.

A consumer group give you the ability to have a distributed system read from the same IoT Hub. We will need a consumer group for Steam Analytics.



## Setting up Cosmos DB

Step 1: In the Azure portal create a new Cosmos DB. Select Core(SQL) as the API, and enable notebooks.



Step 2: Create and Review

Select Create and Review. Then select Create.

Step 3: Connect to your database

After you create the resource group you will need to create a collection to store the data. Go to <https://cosmos.azure.com>, login, select your subscription, and the Cosmos Account you just created.

Step 4: Create a Database and Collection

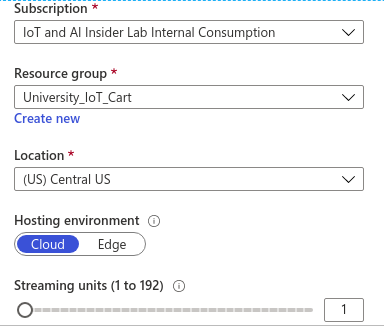
In the GitHub Repo for this project there are 1 iPython notebook. Go into the Notebooks of the repository and upload the 1 notebook. Then run the notebook titled Edge-Fall-Detection-Setup.ipynb by clicking on the Run command. This notebook will create the database, collection, and user defined function.

## Stream Analytics Job

The next step is to create a Stream Analytics Job. Stream Analytics is a distributed stream processing and aggregation service. We will use it to put data from IoT Hub to Cosmos DB.

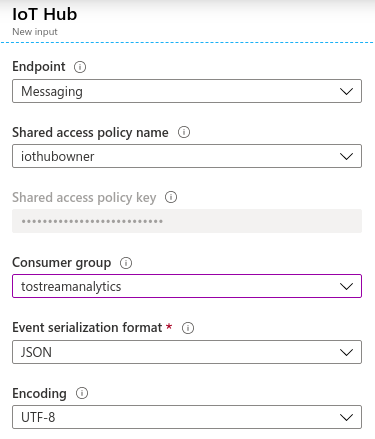
Step 1: Create the Stream Analytics Job

In the Azure Portal create a Stream Analytics Job. From here give it a name, assign it to a resource group, Gie it a



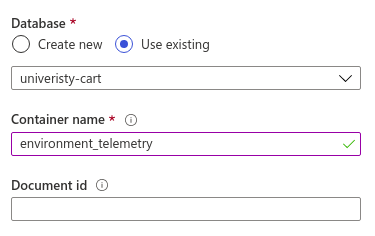
Step 2: Connect to IoT Hub

We are going to add IoT Hub as an input. Do to this we will go to the inputs section in Stream Analytics and select the IoT Hub you created and the consumer group you created for that IoT Hub. For this exercise we are going to ingest the uncompressed JSON telemetry messaging.



Step 3 Output to Cosmos DB

Next, we are going to add an output from Stream Analytics to Cosmos DB. To do this click on output and add a new Cosmos DB output. Select the database and collection you created in the previous step. Leave Document id blank.



Step 4: Query

Next select the “Query” in the menu and in the query editor window put in the following query.

**SELECT** GetMetadataPropertyValue(IoTHub, '[IoTHub].[ConnectionDeviceId]') + **cast**( **MONTH**(EventEnqueuedUtcTime) **as** **nvarchar**(**max**)) + **cast**(**Year**(EventEnqueuedUtcTime) **as** **nvarchar**(**max**)) **as** partitionkey  
      , GetMetadataPropertyValue(IoTHub, '[IoTHub].[ConnectionDeviceId]') **as** deviceid  
      ,alerttype  
      , alertseverty  
      , EventEnqueuedUtcTime **as** time  
      ,device\_type  
**INTO**  
    falldetection  
**FROM**  
IoTHub   
**WHERE** device\_type = 'edge\_fall\_detection';

In the code we are creating a partition key so that Cosmos DB knows where to store the data. Next, we are adding the fall detection alert type and severity.

Next, select Save.

Step 5: Running the Job

After you have saved your query, go back to the “Overview” page and click Start.

# Getting Ready

The first thing you will need to do is set up the IoT Hub Edge tooling on the Respberry Pi. Follow Microsoft instructions on the Pi to get it up <https://docs.microsoft.com/en-us/azure/iot-edge/how-to-install-iot-edge-linux>.

# How to do it

Create a new file called telemetry.py. Raspherry Pi has a code editor called Thorney you can use.

First import the libraries

**import** random  
**import** time  
**import** threading  
from azure.iot.device **import** IoTHubDeviceClient, Message, MethodResponse

Next set the variables. We got the device connection string in the section about creating a device in IoT Hub.

CONNECTION\_STRING = "The device connection stringCONNECTION\_STRING = "The device connection string"  
  
TEMPERATURE = 20.0  
HUMIDITY = 60  
INTERVAL = 1  
readings = {'temperature':0,'humidity':0, 'device\_type':'pi\_environment\_monitor'}

Next, we are going to get the readings from the sensors

def getReadings():  
readings['temperature'] = TEMPERATURE + (random.random() \* 15)  
readings['humidity'] = HUMIDITY + (random.random() \* 20)  
msg\_txt\_formatted = json.dumps(readings)  
message = Message(msg\_txt\_formatted)  
return(message)

Then we are going to have an infinate loop of getting readings and sending it to IoT Hub.

def iothub\_client\_telemetry\_run():  
   
    try:  
        client = IoTHubDeviceClient.create\_from\_connection\_string(CONNECTION\_STRING)  
        print ( "IoT Hub device sending periodic messages, press Ctrl-C to exit" )  
   
        **while** True:  
            message = getReadings()  
   
            # Send the message.  
            print( "Sending message: {}".format(message) )  
            client.send\_message(message)  
            print( "Message sent" )  
            time.sleep(INTERVAL)  
  
    except KeyboardInterrupt:  
        print ( "IoTHubClient sample stopped" )

Finally, we are going to start our program

**if** \_\_name\_\_ == '\_\_main\_\_':   
    print ( "Press Ctrl-C to exit" )   
    **iothub\_client\_telemetry\_run**()

You can find the code for this project listed in the appendix.

# Appendix

IoT Hub: [Reference](https://docs.microsoft.com/en-us/azure/iot-hub/) and [Tutorial](https://iotschool.microsoft.com/en-us/iot/learning-paths/7zy2NeaeYwsoaCuIueeoSa)

Stream Analytics Job: [Reference](https://docs.microsoft.com/en-us/azure/stream-analytics/) and [Tutorial](https://iotschool.microsoft.com/en-us/iot/learning-paths/6qJe4ohYd2EsAEo6gw6C2G/ILFm6wfcKyeUy2Mk2iAE0?selectedItemId=2G4AQkOJVuAcmuaQiIQ0mm)

CosmosDB: [Reference](https://docs.microsoft.com/en-us/azure/cosmos-db/) and [Tutorial](https://iotschool.microsoft.com/en-us/iot/learning-paths/1z75cTRBNqEA2EigQoQKKe/27PKfzBeLuYCCy6qkA2MaQ)