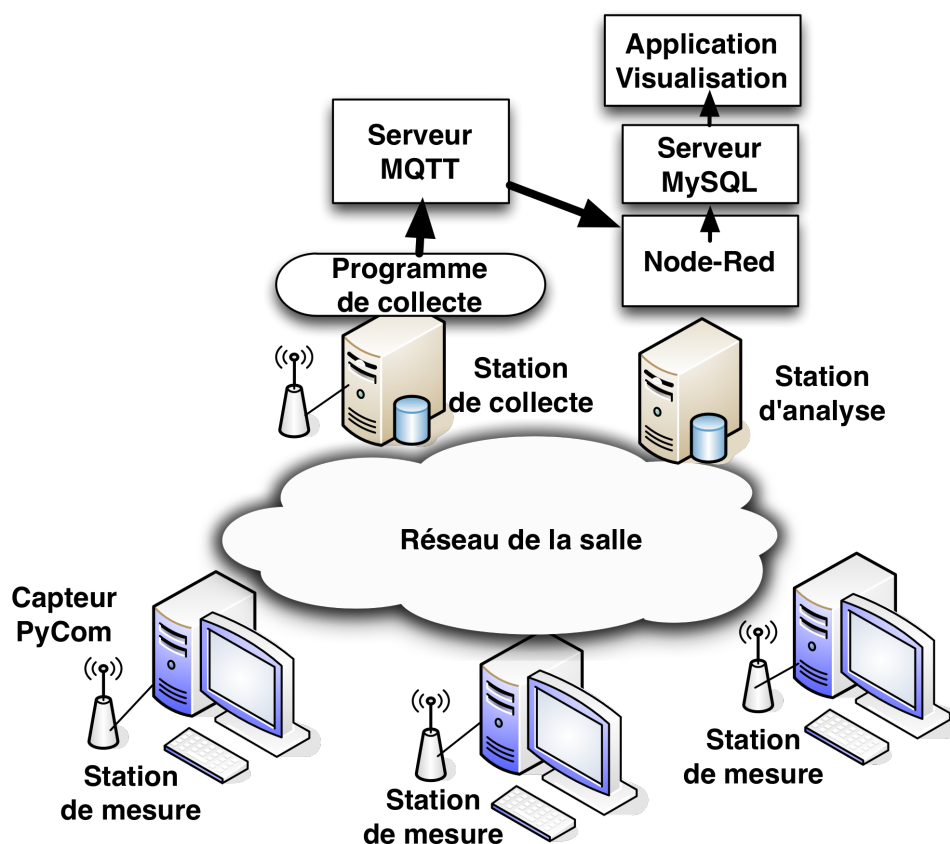


# Bureau d'étude : Mise en œuvre d'une application IoT

## Introduction

L'objectif de ce module est de construire une application IoT de bout en bout pour collecter des grandeurs physiques (températures, humidité et pression) en utilisant un réseau de capteurs sans fil, stocker les données collectées dans une base, afficher des dashboard et enfin afficher un plan thermique en utilisant les valeurs de températures collectées.

Cette application vise l'instrumentation de la salle 206 rouge de l'ENSEM pour collecter grâce aux capteurs déployés les valeurs de température et d'humidité. L'architecture de cette solution est la suivante :



Le travail à réaliser est découpé en plusieurs activités et se déroule sur une séquence de 15h (3h\*5 séances TP) :

- Activité 1 (3H) :
  - programmation des capteurs Pycom pour collecter les données et l'envoyer par un réseau LORA vers une station de collecte
  - collecte de données avec un programme Python exécuté sur la machine de collecte et publication des données vers un serveur MQTT
- Activité 2 (3 H) :
  - mise en place de l'outil Node-Red sur une station d'analyse et de supervision pour récupérer les données depuis le serveur MQTT

- création d'un dashboard Node-Red pour visualiser en temps réel les données de chaque capteur
- Activité 3 (6H) :
  - Création d'une base de données pour stocker les mesures collectées
  - stockage de données dans un serveur MySQL
  - publication des données vers la plateforme *thethingsnetwork*
- Activité 4 (3H) : Création d'un programme Python qui va interroger la base MySQL et visualiser un plan thermique de la salle 206 rouge en utilisant les valeurs de capteurs et des fonctions d'interpolation de la bibliothèque scipy

**Vous allez travailler en binôme. La durée indiquée pour chaque activité est donné à titre indicatif, vous avez la liberté de mettre plus ou moins du temps pour chaque activité en fonction de votre avancement.**

**La dernière séance TP est réservée aux démonstrations de vos réalisations et l'attribution des notes.**

### Activité 1 : Programmation des capteurs et collecte des données

Dans cette première activité vous allez reprendre le code que vous avez utilisé dans les TP4 et TP5 du module 'Réseaux de capteurs' du semestre S7 pour programmer les cartes Pycom et mesurer les valeurs de températures et d'humidités (SI7006A20) et l'envoyer en broadcast sur un réseau LORA.

Chaque capteur agit seulement comme émetteur pour envoyer périodiquement avec le protocole LORA un message qui contient les deux valeurs température et humidité, et également son identifiant (à définir sous un format CAPTx par exemple en remplaçant x par un numéro du capteur) sur le réseau LORA.

```
Temperature: 20.87899 deg C and Relative Humidity: 54.23789 %RH
Temperature: 20.82537 deg C and Relative Humidity: 53.68857 %RH
Temperature: 20.80392 deg C and Relative Humidity: 53.27658 %RH
Temperature: 20.69667 deg C and Relative Humidity: 52.93326 %RH
Temperature: 20.63232 deg C and Relative Humidity: 52.68912 %RH
Temperature: 20.63232 deg C and Relative Humidity: 52.50601 %RH
Temperature: 20.61086 deg C and Relative Humidity: 52.36868 %RH
Temperature: 20.56796 deg C and Relative Humidity: 52.29239 %RH
```

Ensuite, vous allez programmer un de capteur Pycom en mode récepteur seulement pour lire depuis le réseau LORA les valeurs des autres capteurs et les écrire sur la console avec l'instruction *print*.

Brancher ce capteur sur l'une de machines de la salle qui va représenter votre station de collecte.

```
b'CAPT1,20.83609,53.80301'
b'CAPT1,20.79319,53.36051'
b'CAPT1,20.70739,52.97141'
b' '
b'CAPT1,20.66449,52.71964'
b'CAPT1,20.63232,52.54416'
b' '
b'CAPT1,20.60014,52.3992'
b'CAPT1,20.58942,52.29239'
b'CAPT1,20.54652,52.25424'
b' '
b'CAPT1,20.54652,52.18558'
```

Sur cette station de collecte, installez un serveur MQTT en tapant les commande suivante dans une console Linux :

```
sudo apt install mosquitto
```

Dans une deuxième étape de cette activité et sur la station de collecte, vous allez également utiliser le contenu des TP4 et TP5 du module **Réseau de capteurs** pour coder un programme Python qui va lire le port série sur lequel est connecté le capteur Pycom récepteur, afficher les valeurs lues et publier ces valeurs vers votre serveur MQTT dans des topic spécifiques par capteur et par valeur (par exemple /ensem/206R/capt1/temp, /ensem/206R/capt1/hum).

Donnée reçue en provenance du capteur : CAPT1,20.36419,52.50601

Donnée reçue en provenance du capteur : CAPT1,20.39636,52.46787

Donnée reçue en provenance du capteur : CAPT1,20.38564,52.46787

Donnée reçue en provenance du capteur : CAPT1,20.38564,52.44498

**Tester le fonctionnement de votre installation en affichant dans la console les valeurs de capteurs publiées sur le serveur MQTT (utiliser la commande *mosquitto\_sub*)**

*Voici un exemple du résultat en utilisant broker.hivemq.com*

```
MacBook-Air-de-lahmadi:SCADA AbdelkaderLahmadi$ mosquitto_sub -h broker.hivemq.com -t /ensem/206R/temp
```

CAPT1,20.46072,52.04062

CAPT1,20.42854,52.03299

CAPT1,20.40709,52.00247

CAPT1,20.44999,52.04062

CAPT1,20.40709,52.00247

CAPT1,20.41782,51.99484

CAPT1,20.42854,52.01773

## Activité 2 : Visualisation des données

Dans une première étape, vous allez configurer node-red pour lire les topics MQTT et afficher les valeurs des capteurs sur sa console

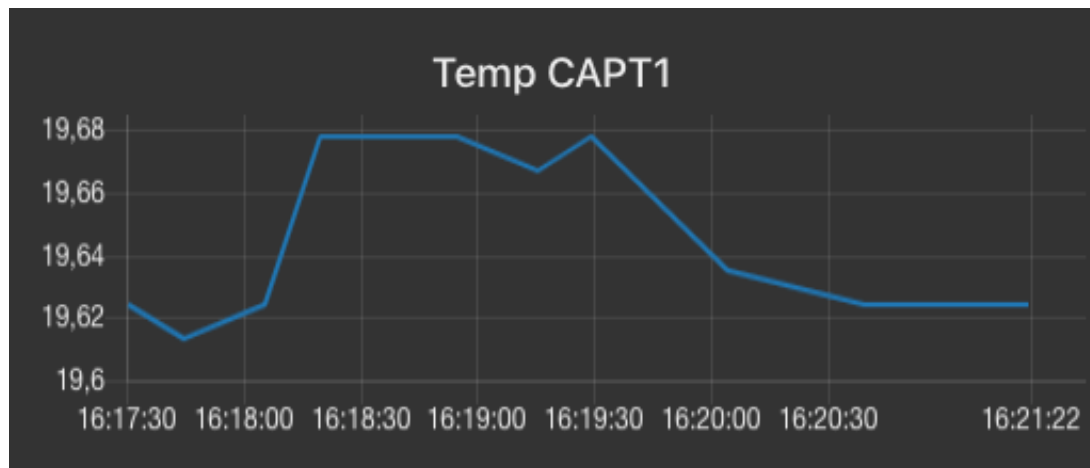
```
04/02/2023, 15:49:51 node: 44d86e9f.251ad
/ensem/206R/CAPT1/temp : msg.payload : string[8]
"19.45255"

04/02/2023, 15:49:58 node: 44d86e9f.251ad
/ensem/206R/CAPT1/temp : msg.payload : string[8]
"19.45255"

04/02/2023, 16:14:58 node: 44d86e9f.251ad
/ensem/206R/CAPT1/temp : msg.payload : string[8]
"19.42038"

04/02/2023, 16:15:05 node: 44d86e9f.251ad
/ensem/206R/CAPT1/temp : msg.payload : string[8]
"19.47401"
```

Dans une deuxième étape, vous allez développer dans node-red un tableau de bord qui affiche les valeurs de la température et l'humidité de chaque capteur



### Activité 3 : Stockage des données

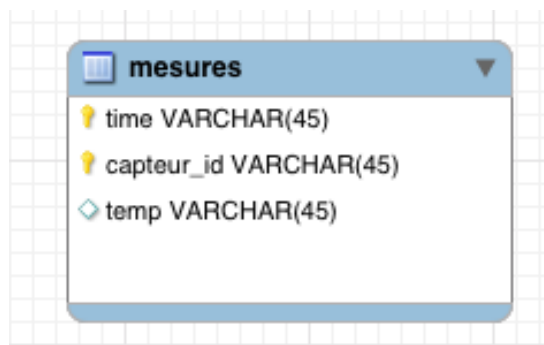
Dans une première étape, vous allez construire une base de données en utilisant l'outil mysqlworkbench (<https://www.mysql.com/products/workbench/>).

Si vous êtes sous Linux, vous pouvez consulter ce lien pour installer un serveur MySQL : <https://doc.ubuntu-fr.org/mysql>

Cette base de données contient une seule table *mesures* avec 3 attributs :

- capteur\_id : identifiant du capteur
- time : instant de la mesure
- temp : valeur de la température

La clé de cette table est composée de capteur\_id et time

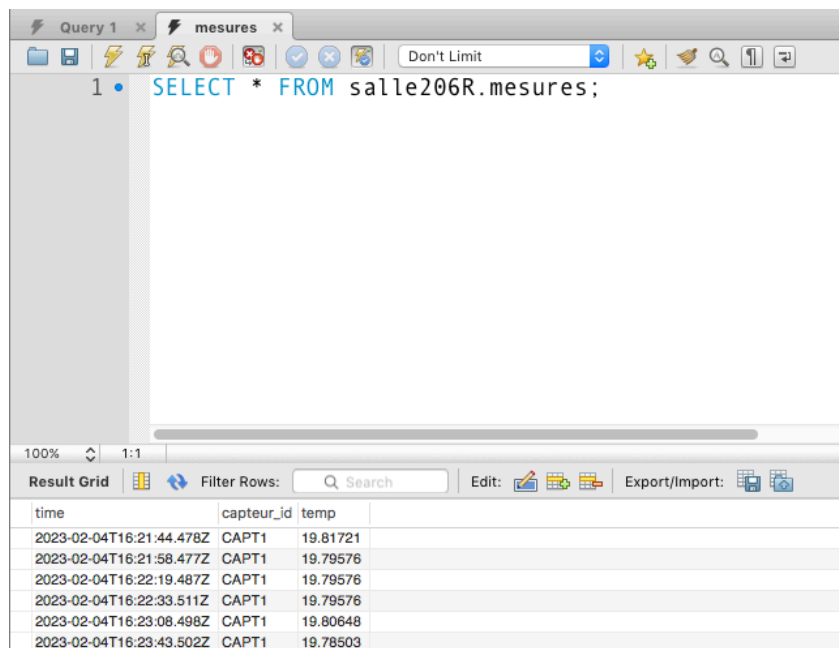


Maintenant, dans Node-Red, installer l'extension **node-red-node-mysql** en utilisant le menu Node-RED Menu - Manage Palette - Install

- Il faut configurer le node mysql avec le nom du host , le nom de la base de données l'identifiant user et son mot de passe
- Il faut ajouter une fonction node pour insérer les données de MQTT dans la table *mesures*



Si cela fonctionne correctement, vous pouvez afficher les valeurs stockées en utilisant MySQLworkbench



### Activité 3 (bis) : publication des données vers la plateforme *thethingsnetwork*

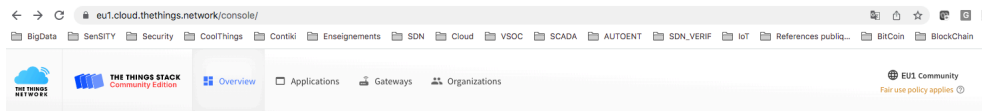
Dans cette activité, vous allez programmer l'un de vos capteurs Pycom pour envoyer les valeurs de température vers un serveur LORA publique hébergé par le réseau thethingsnetwork (<https://www.thethingsnetwork.org/>). Vous aurez également besoin de créer avec un Pycom une gateway pour envoyer les données vers la plateforme *thethingsnetwork*.

Vous pouvez utiliser le code disponible ici pour la gateway :

<https://docs.pycom.io/tutorials/networks/lora/lorawan-nano-gateway/>

Dans une première étape, vous devez créer un compte sur le site de thethingsnetwork en suivant ce lien : <https://www.thethingsindustries.com/tts-plans/?tab=community>

Choisir Student, puis s'enregistrer sur le site. Après accès au site, vous pouvez créer une application sur leur plateforme en cliquant sur *Create an application* :



Après création de l'application, vous devez enregistrer votre capteur LORA. Pour son enregistrement sur la plateforme, vous avez besoin de récupérer son adresse MAC en utilisant ce code dans un terminal Python de votre capteur :

```
>>> from network import LoRa
>>> import ubinascii
>>> lora = LoRa(mode=LoRa.LORA, region=LoRa.EU868)
>>> print("DevEUI: %s" % (ubinascii.hexlify(lora.mac()).decode('ascii')))
DevEUI: 70b3d5499a364b6a
>>> █
```

Ensuite dans l'interface web de TTN, cliquer sur *Register end device* et renseigner les informations suivantes (il faut mettre dans joinEUI la valeur de l'adresse MAC de votre capteur) :

## Register end device

Does your end device have a QR code? Scan it to speed up onboarding.

 Scan end device QR code  [Device registration help](#)

### End device type

#### Input Method

- ☐ Select the end device in the LoRaWAN Device Repository
- ☒ Enter end device specifics manually

#### Frequency plan \*

Europe 863-870 MHz (SF9 for RX2 - recommended) | 

#### LoRaWAN version \*

LoRaWAN Specification 1.0.2 | 

#### Regional Parameters version \*

RP001 Regional Parameters 1.0.2 | 


[Show advanced activation, LoRaWAN class and cluster settings](#) 

### Provisioning information

#### JoinEUI \*

70 B3 D5 49 9A 36 4B 6A


Cliquer ensuite sur *Confirm*. Dans AppKey, cliquer sur Generate. Enfin cliquer sur Register end device.


 **eui-70b3d57ed005a533**  
ID: eui-70b3d57ed005a533


↑ n/a ↓ n/a • No activity yet 


[Overview](#) [Live data](#) [Messaging](#) [Location](#) [Payload formatters](#) [Claiming](#) [General settings](#)

#### General information

End device ID  



Frequency plan  



LoRaWAN version  



Regional Parameters version  

Created at Feb 11, 2023 12:42:13

#### Activation information

AppEUI   

DevEUI   

AppKey   

#### Session information

This device has not joined the network yet

#### MAC data

 Download MAC data

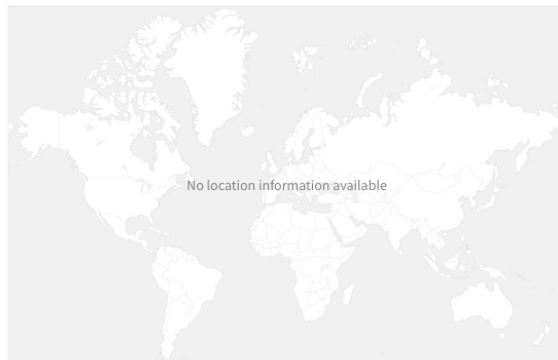
#### Live data

[See all activity](#) →

 12:42:13 Create end device

#### Location

[Change location settings](#) →





Maintenant, vous allez coder un programme sur votre capteur pour envoyer périodiquement la valeur de sa température vers TTN. Le programme doit au début s'authentifier auprès du réseau TTN en utilisant ce code :

```
from network import LoRa
import binascii
import socket
import time
import pycom
from pycoproc_1 import Pycoproc
from SI7006A20 import SI7006A20

pycom.heartbeat(False)
pycom.rgbled(0x0A0A08) # white

py = Pycoproc(Pycoproc.PYSENSE)

lora = LoRa(mode=LoRa.LORAWAN, rx_iq=True, region=LoRa.EU868)

dev_eui = binascii.unhexlify('70B3D5499A36486A')
app_eui = binascii.unhexlify('70B3D5499A36486A')
app_key = binascii.unhexlify('2B2D0EB266805F990BBFB030D31CFE4C')

# set the 3 default channels to the same frequency (must be before sending the OTAA join request)
lora.add_channel(0, frequency=868100000, dr_min=0, dr_max=5)
lora.add_channel(1, frequency=868100000, dr_min=0, dr_max=5)
lora.add_channel(2, frequency=868100000, dr_min=0, dr_max=5)

lora.join(activation=LoRa.OTAA, auth=(dev_eui, app_eui, app_key), timeout=0, dr=5)

# wait until the module has joined the network
while not lora.has_joined():
    time.sleep(2.5)
    print('Not joined yet...')

print('Network joined!')
# remove all the non-default channels
for i in range(3, 16):
    lora.remove_channel(i)
```

Remplacer les valeurs de app\_eui et app\_key par les valeurs d'enregistrement de votre capteur. Pour envoyer les données, utiliser le même code que l'activité 1 avec les sockets et la fonction send.


```
# create a raw LoRa socket
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
s.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)

# make the socket non-blocking
s.setblocking(False)

time.sleep(5.0)

si = SI7006A20(py)
while True:
    print("Temperature: " + str(si.temperature()) + " deg C and Relative Humidity: " + str(si.humidity()) + " %RH")
    #s.setblocking(True)
    s.send(str(si.temperature()))
    time.sleep(5)
```

Exécuter votre code sur le Pycom et observer les données reçues sur l'application de TTN (Live Data).


**eui-70b3d5499a364b6a**  
ID: eui-70b3d5499a364b6a

↑ 13 ↓ n/a • Last activity 5 seconds ago ⓘ

[Overview](#) [Live data](#) [Messaging](#) [Location](#) [Payload formatters](#) [Claiming](#) [General settings](#)

Time	Type	Data preview		Verbose stream		Export as JSON	Pause	Clear
↑ 19:04:13	Forward uplink data message	DevAddr: 26 08 DF C8	<>	Payload: { temp: "23.31359" }	32 33 2E 33 31 33 35 39	<>	FPort: 2 Data rate: SF7BW125 SNR: 6 RSSI: -76	
↑ 19:04:13	Successfully processed data mes...	DevAddr: 26 08 DF C8	<>					
↑ 19:04:06	Forward uplink data message	DevAddr: 26 08 DF C8	<>	Payload: { temp: "23.31359" }	32 33 2E 33 31 33 35 39	<>	FPort: 2 Data rate: SF7BW125 SNR: 7 RSSI: -78	
↑ 19:04:06	Successfully processed data mes...	DevAddr: 26 08 DF C8	<>					
↑ 19:03:59	Forward uplink data message	DevAddr: 26 08 DF C8	<>	Payload: { temp: "23.32432" }	32 33 2E 33 32 34 33 32	<>	FPort: 2 Data rate: SF7BW125 SNR: 6 RSSI: -77	
↑ 19:03:59	Successfully processed data mes...	DevAddr: 26 08 DF C8	<>					
↑ 19:03:52	Forward uplink data message	DevAddr: 26 08 DF C8	<>	Payload: { temp: "23.30287" }	32 33 2E 33 30 32 38 37	<>	FPort: 2 Data rate: SF7BW125 SNR: 6 RSSI: -77	

Pour afficher les valeurs de la température en Float, vous aurez besoin de configurer sur l'application TTN, un formater de messages Uplink de cette façon :

**eui-70b3d5499a364b6a**  
ID: eui-70b3d5499a364b6a

↑ 22 ↓ n/a • Last activity just now ⓘ

[Overview](#) [Live data](#) [Messaging](#) [Location](#) [Payload formatters](#) [Claiming](#)

[Uplink](#) [Downlink](#)

### Setup

Formatter type \*

Custom Javascript formatter

Formatter code \*

```
1 function decodeUplink(input) {  
2   return {  
3     data: {  
4       temp: String.fromCharCode.apply(null, input.bytes)  
5     },  
6     warnings: [],  
7     errors: []  
8   };  
9 }
```

## Activité 4 : Analyse des données

Dans une première étape de cette activité, vous allez coder en Python un programme pour se connecter sur votre base de données et récupérer les valeurs de température de capteurs.

Un exemple du code pour se connecter à la base de données et lire une table est ci-dessous :

```
import mysql.connector
import random
import numpy as np
from scipy.interpolate import Rbf

import matplotlib
import matplotlib.pyplot as plt
from matplotlib import cm
#remplacer le database par le nom de votre base
cnx = mysql.connector.connect(user='root', password='root',
host='127.0.0.1', database='salle206R',port=8889)

cursor = cnx.cursor()

query = ("SELECT capteur_id , time , temp FROM mesures")

cursor.execute(query)

for (capteur_id,time,temp) in cursor:
    print(time,capteur_id,temp)
```

Ensuite, vous allez étendre ce programme pour stocker dans deux tableaux X et Y les coordonnées (à définir manuellement) des capteurs de la salle, et dans un tableau Z leurs valeurs de température respectives. Grâce à la fonction Rbf de Scipy (<https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.Rbf.html>) et en utilisant une interpolation linéaire, vous allez interpoler différents points de la salle afin de visualiser son plan thermique.

Ci-dessous, un exemple de code pour générer aléatoirement des coordonnées et leur associer une 20 de valeurs d'un capteur pour générer un exemple d'une carte thermique par interpolation :

```
query = ("SELECT capteur_id , time , temp FROM mesures")

cursor.execute(query)
values = []
for (capteur_id,time,temp) in cursor:
    #print(time,capteur_id,temp)
    values.append(float(temp)-19)

z = random.sample(values,20)
print(z)
x = np.random.rand(20)
y = np.random.rand(20)
ti = np.linspace(0, 1, 100)
XI, YI = np.meshgrid(ti, ti)
rbf = Rbf(x, y, z,function='linear')
ZI = rbf(XI, YI)

# plot the result
plt.subplot(1, 1, 1)
plt.pcolor(XI, YI, ZI, cmap=cm.jet)
plt.scatter(x, y, 100, z, cmap=cm.jet)
plt.title('RBF interpolation - linear')
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.colorbar()
plt.show()
```

