**PyZMQ Documentatie**

1. Wat is ZMQ
2. Waarom ZMQ?
3. Installatie ZMQ
   1. Windows
   2. Linux
   3. Mac
4. ZMQ-broker
5. ZMQ-publisher
6. ZMQ-ontvanger
7. ZMQ in PSoC blocking issue
8. Oplossing
9. Extra Bronnen
10. **Wat is ZMQ?**

ZeroMQ is een asynchrone berichtenbibliotheek wat gebruikt wordt in gedistribueerde of gelijktijdige toepassingen. Het ondersteunt veel programmeertalen maar wij gebruiken de Python versie van ZMQ. Vandaar de naam PyZMQ.

1. **Waarom ZMQ?**

Omdat het een snelle en lightweighted berichtenbibliotheek waarbij waarmee u zonder veel moeite een complex communicatiesysteem kunt ontwerpen.

1. **Installatie PyZMQ**

PyZMQ is mogelijk om te installeren op MAC, Windows en Linux. Het is belangrijk dat je Python en de laatste versie van Python pip hebt geïnstalleerd.

**Windows**

Je kan PyZMQ installeren op Windows a.d.h.b.v. de command

“pip install pyzmq”

Je moet ook een IDE installeren om in te kunnen programmeren. Wij hebben tijdens dit onderzoek gebruik gemaakt van PyCharm.

Ondervonden probleem “import zmq does not exist”

Deze error geeft dus aan dat zmq niet goed geïnstalleerd is. Ook al is dit vreemd want als we het nachecken is zmq wel degelijk geïnstalleerd. Om dit op te lossen moet je ervoor zorgen dat Python in de default locatie is geïnstalleerd (geen idee waarom). Vervolgens installeer je pyzmq en kopieer je de mappen “zmq” en “pyzmq-20.0.0.dist-info” op de locatie:

“C:\Users\(naam gebruiker)\AppData\Local\Programs\Python\Python39\Lib\site-packages”

En plaats je deze bij je pycharm project op locatie:

“(path van je pycharm project)\venv\Lib\site-packages”

**Linux**

Je kan PyZMQ installeren op Linux a.d.h.v. de volgende commando’s:

“pip install pyzmq”

Bij het compileren van pyzmq (bijv. Installeren met pip op Linux), wordt over het algemeen aanbevolen om zeromq apart te installeren, via homebrew, apt, yum, enzv:

# Debian-based

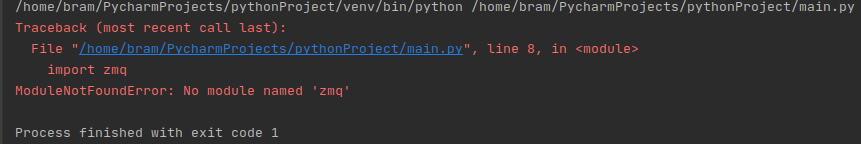
“sudo apt-get install libzmq3-dev”

# RHEL-based

“sudo yum install libzmq3-devel”

Je moet ook een IDE installeren om in te kunnen programmeren. Wij hebben tijdens dit onderzoek gebruik gemaakt van PyCharm.

Ondervonden probleem



Deze error geeft aan dat de pyzmq library nog niet correct geïnstalleerd is. Om dit te verhelpen zijn er al verschillende dingen geprobeerd, zoals de path toevoegen aan sys, de library kopiëren naar de external libraries folder in pycharm en de code proberen runnen via command line. Geen van bovenstaande oplossing werkt en dus zitten we even vast bij het runnen van zmq code in python.

Om deze error op te lossen moet je alle files in deze map kopiëren naar de library map in je project.





Als je nu op run klikt zal je programma kunnen runnen.

**Mac**

Je kan PyZMQ installeren op Windows a.d.h.b.v. de command

“pip install pyzmq”

Je moet ook een IDE installeren. Maar tijdens dit onderzoek heeft niemand van onze groep gebruik gemaakt van PyZMQ op Mac.

**Bron voor de installatie:** <https://github.com/zeromq/pyzmq>

1. **ZMQ-broker**

Een message broker is programma waarmee applicaties, systemen en services met elkaar kunnen communiceren en informatie kunnen uitwisselen. De message broker doet dit door berichten te vertalen tussen formele berichtenprotocollen. Hierdoor kunnen onderling afhankelijke services rechtstreeks met elkaar 'praten', zelfs als ze in verschillende talen zijn geschreven of op verschillende platforms zijn geïmplementeerd.

Code:

import time  
import zmq  
  
ms\_TO\_WAIT\_IN\_POLL = 100  
  
def cliShow(aText="no parameter was fed in"):  
 print("{0:}:: {1:}".format(time.ctime(), aText))  
  
context = zmq.Context()  
  
push = context.socket(zmq.PUB) # Socket facing out - THE zmq.PUSH ARCHETYPE is a right-enough one  
push.bind("tcp://\*:24242") # push acquires all ports 5556 to be PUSH-served  
push.setsockopt(zmq.LINGER, 0) # .SET always explicitly, even if "defaults" promise  
  
pull = context.socket(zmq.PULL) # Socket facing in - THE zmq.PULL ARCHETYPE is a right-enough one  
pull.bind("tcp://\*:24241") # pull acquires all ports 5555 to be PULL-served  
pull.setsockopt(zmq.LINGER, 0) # .SET always explicitly, even if "defaults" promise  
  
try:  
 while True:  
 if (0 == pull.poll(ms\_TO\_WAIT\_IN\_POLL, zmq.POLLIN)):  
 # cliShow("No message arrived yet") # CLI - GUI STUB print()  
 dud = 1  
 else:  
 msg = pull.recv(zmq.NOBLOCK) # /NEVER/ USE A BLOCKING .recv()  
 cliShow("recieved message...") # CLI - GUI STUB print()  
  
 push.send(msg) # .send()  
 cliShow("And sent it to all subscribers") # CLI - GUI STUB print()  
  
except KeyboardInterrupt:  
 pass  
  
except:  
 pass  
  
finally:  
  
 pull.close()  
 push.close()  
  
 context.term()

Het broker adres is tcp://mc1337server.ddns.net:(hier komt de poort) .De poort “24242” is voor requesten en de poort “24241” is voor subscribers.

1. **ZMQ-Publisher**

Code:

# Verzenden API gegevens ZMQ-packet  
  
import zmq  
import time  
  
context = zmq.Context()  
socket = context.socket(zmq.PUSH)  
socket.connect("tcp://mc1337server.ddns.net:24242")  
  
while True:  
 # Declareren van voorbeeld waardes die zogezegd van de api komen  
 # In de toekomst worden in deze variabelen de echte uitgelezen waardes van de api gestopt  
  
 #topic = ID (x,y)  
 x = 0  
 y = 1  
  
 #message  
 mode = 2  
 freq = 25.000 #in Hz  
 R1 = 255  
 G1 = 65  
 B1 = 38  
 W1 = 0  
 R2 = 44  
 G2 = 0  
 B2 = 120  
 W2 = 33  
 R3 = 0  
 G3 = 25  
 B3 = 255  
 W3 = 0  
 hoekAlpha = 20  
 hoekTheta = 170  
  
 # Send to reciever  
 socket.send\_string("%u,%u>{%u,%f,[%u,%u,%u,%u],[%u,%u,%u,%u],[%u,%u,%u,%u],[%u,%u]}"  
 % (x, y, mode, freq, R1, G1, B1, W1, R2, G2, B2, W2, R3, G3, B3, W3, hoekAlpha, hoekTheta))  
 time.sleep(1)

Deze code is bevat de test-variabelen bedoelt voor team 5. Wanneer wij (team 6) deze code dus gaan runnen zal deze de test data blijven sturen tot we de code stoppen. De data wordt gestuurd naar de broker "tcp://mc1337server.ddns.net:24242". De test data heeft de ID als topic. In dit geval 0,1.

1. **ZMQ-Subcriber**

Code:

# ontvanger random zmq packet  
  
import zmq  
import time  
  
ms\_TO\_WAIT\_IN\_POLL = 100  
  
def cliShow(aText="no parameter was fed in"):  
 print("{0:}:: {1:}".format(time.ctime(), aText))  
  
  
# Socket to talk to server  
context = zmq.Context()  
pull = context.socket(zmq.SUB)  
print("Collecting updates from server...")  
pull.connect("tcp://mc1337server.ddns.net:24241")  
  
topicfilter = "0,1"  
pull.subscribe(topicfilter)  
while True:  
 msg = pull.recv()  
 cliShow(msg)  
  
pull.close()  
  
context.term()

Deze code subt op de broker met de topic “0,1”. Dit wil zeggen dat hij alle berichten die de broker naar hem doorstuurt ontvangt en naargelang van topic dropt of binnenlaat. We hebben de code van de broker, publisher en subscriber eens samen gerunt en het resultaat was dat de subscriber effectief het juiste bericht binnenkrijgt.

1. **ZMQ in PSoC blocking issue**

Om ZMQ te kunnen gebruiken op de PSoC zal eerst de zmq lib geïnstalleerd moeten worden op de Zerynth studio software. Hierbij stuitte we op verschillende problemen.

De installatie van de librarie ging als volgt:

Om deze library te installeren heb ik eerst volgende forum bekeken: <https://community.zerynth.com/t/how-to-install-python-libraries-mpu-6050-to-esp-32-fall-detection/2459>

<https://community.zerynth.com/t/adding-lightweight-python-libraries-into-zerynth-studio/1903>

Ze zeggen dus om de lib te downloaden en in de juiste folder te zetten.

Ik heb daarna eerst volgende link toegepast: <https://pypi.org/project/pyzmq/>

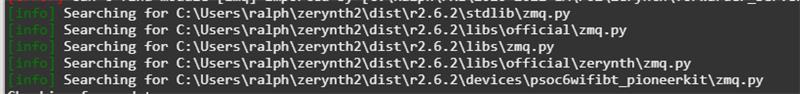
Daarna heb ik de pyzmq lib gedownload van Github en in de juiste folder gegzet: <https://github.com/zeromq/pyzmq>

Probleem

Dan komt het volgende probleem kijken: Hij gaat opzoek naar het zmq.py bestand. Dit bestand zit echter niet in die lib. Ik heb ook nog nergens gevonden welke file de library ‘start’ zodat ik die kan oproepen in de code. Want volgens mij moet ik in Zerynth studio de bestandnaam geven van de code om die uit te kunnen voeren. Dus bijvoorbeeld import zmq als er een zmq.py bestand beschikbaar is.

In 1 van de forums antwoord ook een ingenieur van Zerynth het volgende:   
*However, Unsupported Python libraries may not work in Zerynth studio.  
If these libraries either depend on other libraries that are not supported or due to the difference in operation/Restrictions in the Micro-controllers environment.*

Het is dus mogelijk dat de zmq lib niet kan werken op hun software.



Er is ook contact opgenomen met Bart Stukken en Dieter Vanrykel. Tot op heden hebben Bart en Dieter ook geen oplossing gevonden voor dit probleem. Ook is er een mail verstuurd over dit probleem naar Zerynth zelf als ook naar Cypress. Zerynth heeft ondertussen de vraag beantwoord met ons vermoeden. Zerynth studio kan niet werken met ZMQ.

1. **Oplossing**

Na dit probleem zijn we meteen beginnen zoeken naar een oplossing. Uiteindelijk hebben we twee mogelijke oplossingen.

* Een Raspberry Pi
* MQTT

Raspberry Pi

We hebben ZMQ + de nodige codes op de Raspberry Pi gerunt met succes. Alles werkt succescvol op de Raspberry. Maar toch wouden we de PSoC alsnog een kans geven en hebben we ZMQ proberen de vervangen naar MQTT.

MQTT

Tussen MQTT en ZMQ zijn niet veel verschillen. Het grote verschil tussen beide is dat je bij MQTT subscribed op topics en bij ZMQ subscribbed op message prefixes. Voor ons was het feit dat MQTT wel samenwerkt met Zyrenth ook een belangrijk verschil.

Uiteindelijk zijn we dus ook code beginnen schrijven voor een MQTT communicatie.

1. **Extra Bronnen**

[**https://zeromq.org/**](https://zeromq.org/)

**https://learning-0mq-with-pyzmq.readthedocs.io/en/latest/pyzmq/patterns/pubsub.html**

[**https://blog.stefan-koch.name/2017/11/05/message-queues-brokers-overview-part-1#:~:text=The%20publish%2Dsubscribe%20pattern%20works,will%20do%20in%20this%20example**](https://blog.stefan-koch.name/2017/11/05/message-queues-brokers-overview-part-1#:~:text=The%20publish%2Dsubscribe%20pattern%20works,will%20do%20in%20this%20example)