Kadir Gürhan Kaya

250206057

CENG 314 Embedded Computer Systems Term Project Report

Subject: Push messaging (Heartbeat) and Monitoring

Group:17

1.Introduction

The general principle of push messaging is controlling a system, a function or a server with heartbeat pulse like humans. The system produces a heartbeat signal and another system or controller controls that signal and decide whether it is alive. The objective was creating that signal with python.

2. The Main Heartbeat Function

```
from Heartbeat import Monitor
from Heartbeat import Heartbeat_Maker
import time
import multiprocessing
import queue



True_Message=6 #count of true message
None_Message=2 #count of false message
q = multiprocessing.Queue()
if __name__ == "__main__":

    po=Heartbeat_Maker(q,True_Message,None_Message)
    pq=Monitor(q)

    #creating multiprocessing
    t2 = multiprocessing.Process(target=pq.run)
    t1 = multiprocessing.Process(target=po.run)

    #starting
    t2.start()
    t1.start()

    t1.join()
    t2.join()

```

Figure 1: The runnable of the component.

The runnable function is started with implementing necessary modules. The heartbeat class is place where the heartbeat maker and monitor class are placed. The time module is imported because the functions work time based. Multiprocessor module is necessary to work with processor more than one. Queue module is imported because the processes must share a signal.

In 10th and 11th line the true and false message signal length is identified for simulating heartbeat with heartbeat maker.

In main function the functions are identified from classes and multiprocessing process started.

3- The Heartbeat Classes

3.1 The Heartbeat Maker Class

```
62  #heartbeat creator
63  class Heartbeat_Maker:
64      def __init__(self,q,a,b):
65
66          self.a=a #true signal count
67          self.b=b #false signal count
68          self.q=q #sended signal
69      def run(self):
70          message=3
71          timeout_start=time()
72          while(True==True):
73              sleep(1)
74              elapsed=(time()-timeout_start)
75              if(elapsed<self.a):
76
77                  self.q.put(message)
78
79              elif(elapsed>self.a and elapsed<self.b+self.a):
80                  self.q.put(None)
81              else:
82                  timeout_start=time()
83                  self.q.put(message)
84
85              #print(self.q.get())
86
```

Figure 2: The heartbeat maker class

The init function created for use q, a, b variables external. a and b values are signal count in second. In the run function, the message is identified 3 (0000 …. 0011 in 32 bit) and time is logged. While function runs infinite. True message is created in first if block and the false (None) message is created in elif block. Else block is reset that period and started again. The q.put function is used for send the message signal.

3.2 The Monitoring Class

```
7   treshold=5
8   timeout_start = time()
9   #monitoring heartbeat
10  class Monitor:
11      def __init__(self, q1):
12          self.q1=q1
13      def run(self):
14          flag=False
15          message=3
16
17          while(True==True):
18
19              value=self.q1.get()
20              if(flag==False):
21                  if value!=message and value!=None:
22                      print("message missmatch")
23                  elif value==None:
24
25                      a=time()
26                      while(value ==None):
27                          print("Status=OK2")
28                          value=self.q1.get()
29                          b=time()
30                          if(treshold<(b-a)):
31                              flag=True
32                              sleep(1)
33                              print("Error=The function is not working.")
34                              break
35
36                  elif value == message:
37                      print("Status=OK")
38                  else:
39                      print("unknown error")
40
41              elif(flag==True):
42                  if value!=message and value!=None:
43                      print("message missmatch")
44                  elif value == message:
45                      a=time()
46                      while(value == message):
47                          value=self.q1.get()
48                          b=time()
49                          print("Error=The function is not working")
50                          if(treshold<(b-a)):
51                              print("Status changed=OK")
52                              flag=False
53                              break
54
55                  elif value==None:
56                      print("Error=The function is not working.")
57
58                  else:
59                      print("unknown error")
60
```

Figure 3: The Monitoring class

The monitoring class starts with a threshold which means the decide the failure with fail of continuous threshold count. The value is identified with get function for prevent race condition. The first if block whose flag is false is run if there is no failure decided before. The elif block in line 41 is run when the failure decided. When we continue first block, first if block controls the mismatch of send and received message. The elif blocks controls the message is none or true. When the message is none, the another while block is started until the message is true. If the message is not true for threshold time (in this example it is 5), the status is turn error and flag is false. When flag is false the elif block (line 41) started to work. The elif block which condition flag is false is much same as if block which condition flag is true. This block continues to control the message signal. If the heartbeat is started again, it must be continuing to send true message along threshold time. If it is true for greater time than threshold, the flag will be true again.

4-Test and Conclusion

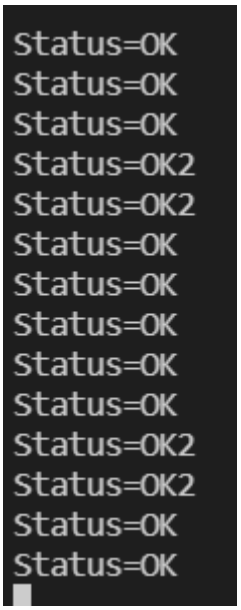

Figure 4: The monitor output. The condition is none message under threshold.

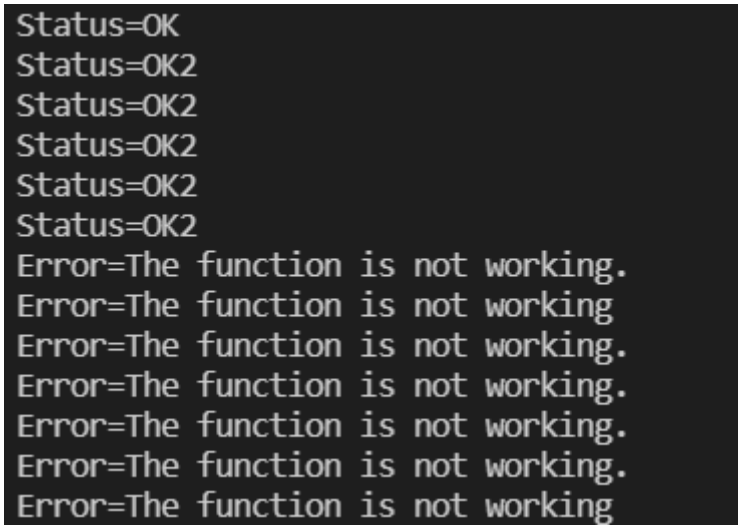

Figure 5: The monitor output. The condition is none message greater than threshold.

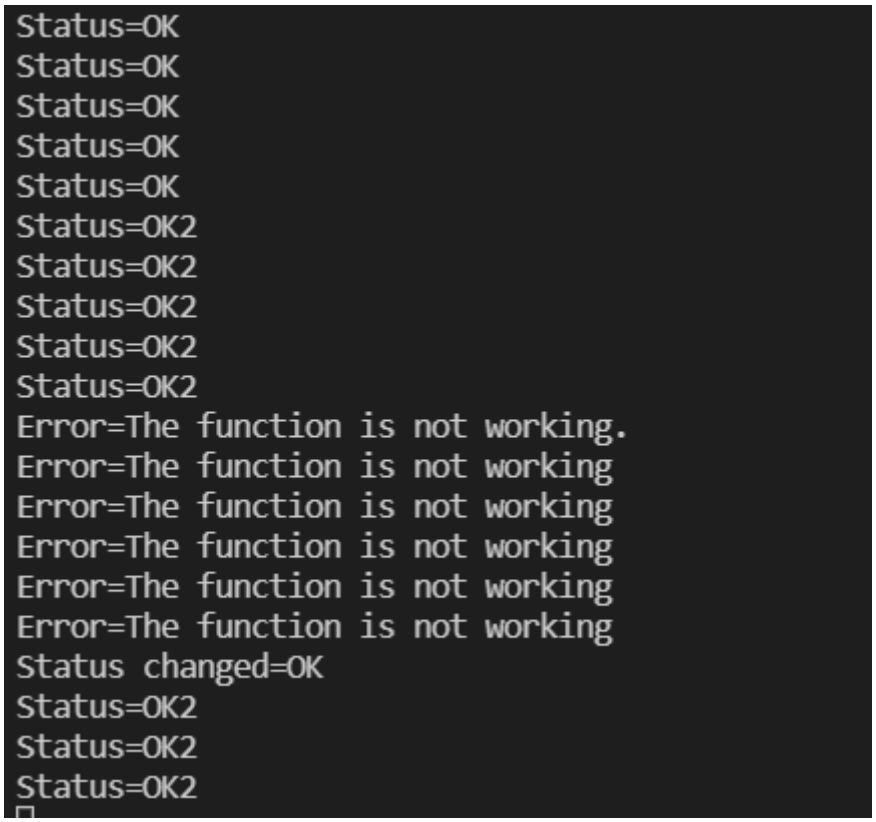

Figure 6: The monitor output. The condition is none message is greater than threshold. After that, the true message is coming greater than threshold.

The OK2 status indicates the heartbeat is not working but it is still under threshold. It looks if the threshold is exceeded when the status is OK2.

The function and class worked almost true for the conditions. There are some time delays under condition which put and get functions works at the same time. But the delays can be prevented with arranging sleep function.