```python
import json
import socket
import threading

from RPi import GPIO


class TcpServer(threading.Thread):
    def __init__(self, ip, port, byte_num, led1, led2, button1, button2):
        """
        TcpServer fragt GPIOs ab und sendet Daten an TcpClient
        :param ip:
        :param port:
        :param byte_num:
        """
        super().__init__()
        self.__v_sock = None
        self.k_sock = None
        self.__ip = ip
        self.__port = port
        self.__bytes = byte_num
        self.leds_state = {}
        GPIO.setmode(GPIO.BOARD)
        self.led1_pin = led1
        self.button1_pin = button1
        self.led2_pin = led2
        self.button2_pin = button2
        #Festlegung der Ausgänge
        GPIO.setup(self.led1_pin, GPIO.OUT)
        GPIO.setup(self.led2_pin, GPIO.OUT)
        #Festlegung der Eingänge
        GPIO.setup(self.button1_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        GPIO.setup(self.button2_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
        # Ereignis bei Betätigung des Buttons in Raum 1
        GPIO.add_event_detect(self.button1_pin, GPIO.RISING, callback=self.button1_interrupt, bouncetime=500)
        # Ereignis bei Betätigung des Buttons in Raum 1
        GPIO.add_event_detect(self.button2_pin, GPIO.RISING, callback=self.button2_interrupt, bouncetime=500)

    def init_states(self):

        if GPIO.input(self.led1_pin) == GPIO.HIGH:
            self.leds_state["led1"] = 0
        else:
            self.leds_state["led1"] = 1

```

```
46          if GPIO.input(self.led2_pin) == GPIO.HIGH:
47              self.leds_state["led2"] = 0
48          else:
49              self.leds_state["led2"] = 1
50          self.__send()
51
52      def run(self):
53          """
54          Send current LED pin state to connected clients.
55
56          v_sock: Verbindunssocket
57          k_sock: Kommunikationssocket
58
59          :return: None
60          """
61          self.__v_sock = socket.socket(socket.AF_INET, socket.
    SOCK_STREAM)
62          self.__v_sock.bind((self.__ip, self.__port))
63          self.__v_sock.listen(1)
64          try:
65              while True:
66                  self.k_sock, addr = self.__v_sock.accept()
67
68                  while True:
69                      data = self.k_sock.recv(self.__bytes)
70                      if len(data) > 0:
71                          print("Server data: {}".format(data.
    decode()))
72                          data_json = json.loads(data.decode())
73                          if 'init' in data_json.keys():
74                              #Schlüssel (Element) im dictionary?
75                              self.init_states()
76                          else:
77                              self.leds_state["led1"] = data_json
    ["R1"]
78                              self.leds_state["led2"] = data_json
    ["R2"]
79                              self.change_led()
80                              # self.k_sock.send(json.dumps(
    data_dict).encode())
81                      else:
82                          self.k_sock.close()
83                          break
84          finally:
85              self.__v_sock.close()
86
87      def change_led(self):
88          if self.leds_state["led1"] == 1:
89              self.licht_an(self.led1_pin)
90          if self.leds_state["led2"] == 1:
```

```
            self.licht_an(self.led2_pin)
        if self.leds_state["led1"] == 0:
            self.licht_aus(self.led1_pin)
        if self.leds_state["led2"] == 0:
            self.licht_aus(self.led2_pin)

    def licht_an(self, pin):
        """
        Turn LED on
        :param pin: RPi board pin number where LED is
    connected to.
        """
        if GPIO.input(pin) == GPIO.HIGH:
            GPIO.output(pin, GPIO.LOW)
            print("Licht ist bei Pin {} an".format(pin))
            self.__send()


    def licht_aus(self, pin):
        if GPIO.input(pin) == GPIO.LOW:
            GPIO.output(pin, GPIO.HIGH)
            print("Licht ist bei Pin {} aus".format(pin))
            self.__send()

    # Interrupt- Funktion für den Button in Raum 1
    def button1_interrupt(self, channel):
        if GPIO.input(self.led1_pin) == GPIO.HIGH:
            self.leds_state["led1"] = 1
            self.licht_an(self.led1_pin)
        else:
            self.leds_state["led1"] = 0
            self.licht_aus(self.led1_pin)

    def button2_interrupt(self, channel):
        if GPIO.input(self.led2_pin) == GPIO.HIGH:
            self.leds_state["led2"] = 1
            self.licht_an(self.led2_pin)
        else:
            self.leds_state["led2"] = 0
            self.licht_aus(self.led2_pin)

    def __send(self):
        msg = json.dumps(self.leds_state)
        self.k_sock.send(msg.encode())

```