Industrial Specific Fire Management system

Team ID	PNT2022TMID48285
Reg No	913019104017
Name	PREETHA.S

```
#importing the required module
#/usr/bin/python -p
import time
import subprocess
import threading
"""
This is a class for blinking the onboard ACT LED on the Raspberry PI 3 based on
shell commands. There should be a way to do this via GPIO, but for me it did not
work so far. Here are some relevant links for blinking via GPIO:
* https://kofler.info/on-board-leds-des-raspberry-pi-steuern/
* https://www.raspberrypi.org/forums/viewtopic.php?f=44&t=144787
Example:
import time
l = LED()
# turn on LED
l.led(True)
# turn off LED
l.led(False)
# turn on LED and then turn it off after 1 second
l.led(True, 1)
# some blinking patterns
l.start()
time.sleep(5)
l.stop()
l.start_simple(3,0.1,1)
time.sleep(5)
l.stop()
l.start_offset(3,0.1,0.2,1)
time.sleep(5)
l.stop()
l.start_simple([0.1,0.1,0.1,0.25,1])
time.sleep(5)
l.stop()
Credits:
* https://stackoverflow.com/questions/18018033/how-to-stop-a-looping-thread-in-python
* https://kofler.info/on-board-leds-des-raspberry-pi-steuern
class LED:
def __init__(self):
subprocess.call("echo \"none\" > /sys/class/leds/led0/trigger", shell=True)
def start_loop(self, loop_body):
def run():
```

```
while not self.stop_event.is_set():
loop_body()
self.stop_event = threading.Event()
thread = threading.Thread(target=run)
thread.start()
return self
def start(self):
self.start_loop(lambda: self.blink_offset(1, 1, 1, 1))
def start_simple(self, blinks, quick, slow):
self.start_loop(lambda: self.blink_offset(blinks, quick, quick, slow))
def start_offset(self, blinks, on, off, out):
self.start_loop(lambda: self.blink_offset(blinks, on, off, out))
def start_generic(self, onoff):
self.start_loop(lambda: self.blink_generic(onoff))
def led(self, on, reset = -1):
if on:
subprocess.call("echo 1 > /sys/class/leds/led0/brightness", shell=True)
else:
subprocess.call("echo 0 > /sys/class/leds/led0/brightness", shell=True)
if reset > 0:
time.sleep(reset)
self.led(not on)
def blink_offset(self, blinks, on, off, out):
time.sleep(out)
for i in range(blinks):
self.led(True, on)
time.sleep(off)
def blink_generic(self, onoff):
for i in range(len(onoff) / 2):
on = onoff[i * 2]
off = onoff[i * 2 + 1]
self.led(True, on)
time.sleep(off)
def stop(self):
self.stop_event.set()
```