SPRINT 2

ALGORITHM:

- ➢ Import Packages
- ➢ Create ‘myConfig’ location
- ➢ Implement the wiotp.sdk.device.DeviceClient
- ➢ Run a while Loop
- ➢ Finally set the latitude and longitude range
- ➢ Desired result Obtained

Modified Version of Code according to main project:

```
import json import
wiotp.sdk.device
import time myConfig={
   "identity":{
   "orgId": "hj5fmy", "typeid": "NodeMCU",
   "deviceId": "12345"
   },
   "auth": {
      "token": "12345678"
```

```
    }
}
client = wiotp.sdk.device.DeviceClient (config=myConfig, logHandlers=None)
client.connect()

while True:
    name= "Smartbridge"
    #in area location
    #latitude- 17.4225176 longitude 78.5450842
    #out area location
latitude = 17.4219272
longitude =70.5400783
    myData = {'name':name, 'lat':latitude, 'lon': longitude}
    client.publishEvent (eventId="Status", msgformat="json", data=myData,
        qos=0, onPublish=None)
    print ("Data published to IM IoT platfrom: ",myData)
time.sleep(5) client.disconnect()
```

__Reference Code:__

```
import time def
stopwatch(seconds,d,lspoint):
        start = time.time()
time.clock()
elapsed = 0  flag = False
```

```
          num = 0       while elapsed < seconds:                    elapsed =
time.time() - start           print "%02d" % elapsed              if elapsed >
d[num] and elapsed < d[num+1] and flag == False:
                              x = lspoint[num][0]
y = lspoint[num][1]
createpoint(x,y)                          flag = True
print "Shot Taken"                                  print
point_in_poly(x,y,polygon)                          if elapsed >
d[num+1]:                          print "Shot Taken"
          flag == False                             num =
num+1
                              x = lspoint[num][0]
y = lspoint[num][1]
createpoint(x,y)                          print
point_in_poly(x,y,polygon)
time.sleep(1)


def createpoint(x,y):
   crs = "point?crs=epsg:27700&field=id:integer"
layer =  QgsVectorLayer(crs, 'points' , "memory")
pr = layer.dataProvider()                pt =
QgsFeature()
   point1 = QgsPoint(x,y)
   pt.setGeometry(QgsGeometry.fromPoint(point1))
pr.addFeatures([pt])       layer.updateExtents()      pt
= QgsFeature()
```

```
   QgsMapLayerRegistry.instance().addMapLayers([layer])

def point_in_poly(x,y,poly):

   n = len(poly)
inside = False

   p1x,p1y = poly[0]
for i in range(n+1):
      p2x,p2y = poly[i % n]
if y > min(p1y,p2y):
         if y <= max(p1y,p2y):
if x <= max(p1x,p2x):
if p1y != p2y:
                  xints = (y-p1y)*(p2x-p1x)/(p2y-p1y)+p1x
if p1x == p2x or x <= xints:
                  inside = not inside
p1x,p1y = p2x,p2y     return
inside polygon =
[(512882.78819722467,120811.83924772343),(512960.84437170526,120809.
7007223952),(512960.84437170526,120809.7007223952),(512959.77510904
1         13,120754.09906386107),(512882.78819722467,120756.2375891893)]
time_seconds = 70 x = 512915 y = 120728
intervals = int(time_seconds / 10)
lspoint = [] for i in
range(0,intervals+1):      y1 = y +
(i*12.5)

lspoint.append([x,y1])

f = 10 a = 0 b =

intervals+1

d = [x * f for x in range(a, b)] stopwatch(time_seconds,d,lspoint)
```