```
import beaglebone_pru_adc as adc
import smbus
import time
import Adafruit_BBIO.GPIO as GPIO
from flask import Flask, render_template, request, jsonify
import threading

GPIO.setup("P8_6", GPIO.IN)
GPIO.setup("P8_7", GPIO.IN)


class captureThread(threading.Thread):
    def __init__(self, numSamples_ = 10000, delayCapture_ = 0, delayADC_ = 0, captureWindow_
    = 100, oneTime_ = False, offset_ = 10, interface_refreshTime_ = 100,
    interface_refreshType_ = 0, ymin_ = 0, ymax_ = 4095, ytype_ = 1):
        threading.Thread.__init__(self)
        self.numSamples = numSamples_
        self.delayCapture = delayCapture_
        self.delayADC  = delayADC_
        self.oneTime = oneTime_
        self.offset = offset_
        self.ytype = ytype_
        self.ymin = ymin_
        self.ymax = ymax_
        self.captureWindow = captureWindow_
        self.interface_refreshTime = interface_refreshTime_
        self.interface_refreshType = interface_refreshType_ #0 - continuo / 1 - uma vez
        self.captureSamples = ()
        print("ADC inicializado")
        print("Samples: ", self.numSamples, "Capture delay: ", self.delayCapture, "ADC
        speed: ", self.delayADC,"Offset: ", self.offset, "Interface refresh: ", self.
        interface_refreshTime, "interface_refreshType", self.interface_refreshType)
    def run(self):
        print("Iniciando captura do ADC interno...")
        while(self.oneTime == False):
            self.captureData()
            time.sleep(self.delayCapture/1000)
        print("Processo terminado.")
    def captureData(self):
        capture = adc.Capture()

        if self.delayADC != 0:
            capture.cap_delay = self.delayADC
        else:
            capture.cap_delay = 0

        capture.oscilloscope_init(adc.OFF_VALUES, self.numSamples) # captures AIN0 - the
        first elt in AIN array
        capture.start()

        while(True):
            if capture.oscilloscope_is_complete():
                break

        capture.stop()
        capture.wait()
        #print("Buffer atualizado com %d amostras."%(self.numSamples,))
        self.captureSamples = capture.oscilloscope_data(self.numSamples)
        capture.close()

class captureThreadExternal(threading.Thread):
    def __init__(self, numSamples = 10000, delayCapture = 10, ymin = 0, ymax = 65535, ytype =
     1):
        threading.Thread.__init__(self)
        self.numSamples = numSamples
        self.delayCapture = delayCapture
        self.ymin = ymin
        self.ymax = ymax
        self.ytype = ytype
```

```
        self.captureSamples = ()

    def run(self):
        bus = smbus.SMBus(1)
        data = [0x84,0x83]
        bus.write_i2c_block_data(0x48, 0x01, data)
        print("Iniciando captura do ADC externo...")
        time.sleep(0.5)
        i = 0
        captureSamplesBuffer = []
        while(True):
            data = bus.read_i2c_block_data(0x48, 0x00, 2)
            raw_adc = data[0] * 256 + data[1]
            time.sleep(self.delayCapture/1000)
            if (i< self.numSamples):
                captureSamplesBuffer.append(raw_adc)
                i += 1
            else:
                self.captureSamples = tuple(captureSamplesBuffer)
                i = 0

"""if (GPIO.input("P8_6")):

else:
    print("Pino P8_6 desconectado Thread do ADC interno n criado")"""

thread1 = captureThread()
thread1.start()
print("Thread do ADC interno criado.")

if (GPIO.input("P8_7")):
    thread2 = captureThreadExternal()
    thread2.start()
    adc_external = True
    print("Thread do ADC externo criado.")
else:
    adc_external = False
    print("Pino P8_7 desconectado Thread do ADC externo n criado")

app = Flask(__name__)
@app.route('/')
def toolbox():
    web_data = []
    for x in thread1.captureSamples[thread1.offset:len(thread1.captureSamples)]:
        web_data.append(int(x))
    if (adc_external == True):
        return render_template('toolbox.html', web_data = web_data, noSamples = thread1.
        numSamples, delayADC = thread1.delayCapture, offsetADC = thread1.offset, speedADC =
        thread1.delayADC, interface_refreshType = thread1.interface_refreshType,
        interface_frameRate = thread1.interface_refreshTime, windowSamples = thread1.
        captureWindow, ytype = thread1.ytype, ymin = thread1.ymin, ymax = thread1.ymax,
        adcExternal = True, web_data_external = thread2.captureSamples, noSamples_external =
        thread2.numSamples, delay_capture_external = thread2.delayCapture, ymin_external =
        thread2.ymin, ymax_external = thread2.ymax, ytype_external = thread2.ytype)
    else:
        return render_template('toolbox.html', web_data = web_data, noSamples = thread1.
        numSamples, delayADC = thread1.delayCapture, offsetADC = thread1.offset, speedADC =
        thread1.delayADC, interface_refreshType = thread1.interface_refreshType,
        interface_frameRate = thread1.interface_refreshTime, windowSamples = thread1.
        captureWindow, ytype = thread1.ytype, ymin = thread1.ymin, ymax = thread1.ymax,
        adcExternal = False, web_data_external = 0, noSamples_external = 0,
        delay_capture_external = 0, ymin_external = 0, ymax_external = 0, ytype_external = 0)

@app.route('/config', methods=['POST'])
def toolbox_config():
    thread1.numSamples =  int(request.form['adc_captureSamples'])
    thread1.delayCapture = int(request.form['adc_captureDelay'])
    thread1.offset = int(request.form['adc_captureOffset'])
    thread1.delayADC = int(request.form['adc_captureSpeed'])
```

```
    thread1.interface_refreshType = int(request.form['interface_refreshType'])

    thread1.ytype = int(request.form['ytype'])
    if thread1.ytype == 1:
        thread1.ymin = int(request.form['ymin'])
        thread1.ymax = int(request.form['ymax'])
    if thread1.interface_refreshType == 0:
        thread1.interface_refreshTime = int(request.form['interface_frameRate'])
    thread1.captureWindow = int(request.form['adc_windowSize'])
    print("Samples: ", thread1.numSamples, "Capture delay: ", thread1.delayCapture, "ADC
    speed: ", thread1.delayADC,"Offset: ", thread1.offset, "Interface refresh: ", thread1.
    interface_refreshTime, "interface_refreshType", thread1.interface_refreshType)
    return ''

@app.route('/senddata', methods=['POST'])
def send_data():
    web_data = []
    for x in thread1.captureSamples[thread1.offset:len(thread1.captureSamples)]:
        web_data.append(int(x))
    if (adc_external == False):
        return jsonify(web_data)
    else:
        web_data_complete = [web_data thread2.captureSamples]
        return jsonify(web_data_complete)


if __name__ == '__main__':
    app.run('0.0.0.0')
```