모바일 스마트 시스템

project

컴퓨터공학부 1592033 지수민

컴퓨터공학부 1592039 황정하

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 번호 | 기능 | 가중치 | 지수민 | 황정하 | 관련코드 |
| 1 | Tensorflow를 이용한 web에 동영상 스트리밍 |  | 50 | 50 | det.py의 206~268라인  jmlee\_flask.py의 12~15라인  pro\_img.html의 4~5라인 |
| 2 | 버튼 누르면 동영상 tensorflow화면 그대로 캡쳐 |  | 50 | 50 | det.py의 250~257라인  jmlee\_flask.py의 33~40라인  pro\_capture.html 전체 |
| 3 | 버튼 누를 때 일정조도 아래일 시 led 켜짐 |  | 50 | 50 | det.py의 250~257라인  jmlee\_flask.py의 29~31라인  pro\_img.html의 6, 9라인 |
| 4 | 일정 온도 이상일 시 led 꺼짐 |  | 50 | 50 | det.py의 246~248라인  jmlee\_flask.py의 21~23라인  pro\_img.html의 7라인 |
| 5 | 습도 측정 |  | 50 | 50 | det.py의 56~60라인  jmlee\_flask.py의 25~27라인  pro\_img.html 8라인 |
| 6 | Nginx 사용 가능 |  | 50 | 50 |  |
| 합계 | LED, SWITCH, HTU21D, CDS, CAMERA, tensorflow, Web |  | 300 | 300 |  |

det.py

|  |
| --- |
| 1. from \_\_future\_\_ import absolute\_import 2. from \_\_future\_\_ import division 3. from \_\_future\_\_ import print\_function 4. import argparse 5. import io 6. import os 7. import re 8. import time 9. import RPi.GPIO as GPIO 10. import board 11. from adafruit\_htu21d import HTU21D 12. import busio 13. from annotation import Annotator 14. import numpy as np 15. import picamera 16. from PIL import Image 17. from PIL import ImageDraw 18. from PIL import ImageFont 19. from tflite\_runtime.interpreter import Interpreter 20. GPIO.setwarnings(False) 21. GPIO.setmode(GPIO.BCM) 22. btn = 24 23. led = 23 24. led\_status = 0 25. btn\_status = 1 26. GPIO.setup(btn, GPIO.IN, GPIO.PUD\_UP) 27. GPIO.setup(led, GPIO.OUT) 28. GPIO.output(led, led\_status) 29. id = [] 30. score = [] 31. minx = [] 32. maxx = [] 33. miny = [] 34. maxy = [] 35. count = 0 36. capture\_img = "ggg.jpeg" 37. fname = "gg.jpeg" 38. try: 39. i2c = busio.I2C(board.SCL, board.SDA) #busio 모듈 내에서 busio.I2C 클래$ 40. sensor = HTU21D(i2c) 41. except: 42. i2c = busio.I2C(board.SCL, board.SDA) 43. sensor = HTU21D(i2c) 44. def getTemperature(): #온도리턴 45. f = open("/home/pi/mobile/project/temp.txt", "wt") 46. f.write(str(sensor.temperature)) 47. f.close() 48. return float(sensor.temperature) 49. def getHumidity(): 50. f = open("/home/pi/mobile/project/humi.txt", "wt") 51. f.write(str(sensor.relative\_humidity)) 52. f.close() 53. return float(sensor.relative\_humidity) 54. def dinit(cs, mosi, miso, clk): #조도 측정에 쓸 변수 초기화 55. GPIO.setup(cs, GPIO.OUT) 56. GPIO.setup(mosi, GPIO.OUT) 57. GPIO.setup(miso, GPIO.IN) 58. GPIO.setup(clk,GPIO.OUT) 59. def sendCmd(cs, mosi, clk, command): 60. GPIO.output(cs, GPIO.HIGH) 61. GPIO.output(cs, GPIO.LOW) 62. GPIO.output(clk, GPIO.LOW) 63. for i in range (4): 64. if(command & 0x80): 65. GPIO.output(mosi, GPIO.HIGH) 66. else: 67. GPIO.output(mosi, GPIO.LOW) 68. command = command << 1 69. GPIO.output(clk, GPIO.HIGH) 70. GPIO.output(clk, GPIO.LOW) 71. def readData(miso, clk): # 조도 읽어오기 72. value = 0 73. for i in range(13): 74. GPIO.output(clk, GPIO.HIGH) 75. GPIO.output(clk, GPIO.LOW) 76. value = value << 1 77. if GPIO.input(miso): 78. value = value | 0x1 79. else: 80. value = value | 0x0 81. value = value >> 1 # discard the first bit 82. return value 83. def getIlluminance(): #조도 측정 함수 84. global current 85. cs = 8 # board.SPI\_CE0\_N 86. mosi = 10 # board.MOSI 87. miso = 9 # board.MISO 88. clk = 11 # board.CLK 89. dinit(cs, mosi, miso, clk) 90. sendCmd(cs, mosi, clk, ((0<<1)|0x0d)<<4) 91. f = open("/home/pi/mobile/project/ill.txt", "wt") 92. f.write(str(readData(miso, clk))) 93. f.close() 94. return int(readData(miso, clk)) 95. def lastfname(): 96. global fname 97. f = open("/home/pi/mobile/project/lastfname.txt", "wt") 98. f.write(fname) 99. f.close() 100. return fname 101. def currentledstatus(): 102. global led\_status 103. f = open("/home/pi/mobile/project/led.txt", "wt") 104. if(led\_status == 1): 105. f.write("LED ON") 106. elif(led\_status == 0): 107. f.write("LED OFF") 108. f.close() 109. def capture(): 110. global capture\_img 111. f = open("/home/pi/mobile/project/capture\_img.txt", "wt") 112. f.write(fname) 113. f.close() 114. return fname 115. CAMERA\_WIDTH = 640 116. CAMERA\_HEIGHT = 480 117. def init(): 118. global id, score, minx, miny, maxx, maxy 119. id = [] 120. score = [] 121. minx = [] 122. maxx = [] 123. miny = [] 124. maxy = [] 125. def load\_labels(path): 126. with open(path, 'r', encoding='utf-8') as f: 127. lines = f.readlines() 128. labels = {} 129. for row\_number, content in enumerate(lines): 130. pair = re.split(r'[:\s]+', content.strip(), maxsplit=1) 131. if len(pair) == 2 and pair[0].strip().isdigit(): 132. labels[int(pair[0])] = pair[1].strip() 133. else: 134. labels[row\_number] = pair[0].strip() 135. return labels 136. def set\_input\_tensor(interpreter, image): 137. tensor\_index = interpreter.get\_input\_details()[0]['index'] 138. input\_tensor = interpreter.tensor(tensor\_index)()[0] 139. input\_tensor[:, :] = image 140. def get\_output\_tensor(interpreter, index): 141. output\_details = interpreter.get\_output\_details()[index] 142. tensor = np.squeeze(interpreter.get\_tensor(output\_details['index'])) 143. return tensor 144. def detect\_objects(interpreter, image, threshold): 145. set\_input\_tensor(interpreter, image) 146. interpreter.invoke() 147. boxes = get\_output\_tensor(interpreter, 0) 148. classes = get\_output\_tensor(interpreter, 1) 149. scores = get\_output\_tensor(interpreter, 2) 150. count = int(get\_output\_tensor(interpreter, 3)) 151. con = count 152. results = [] 153. for i in range(count): 154. if scores[i] >= threshold: 155. result = { 156. 'bounding\_box': boxes[i], 157. 'class\_id': classes[i], 158. 'score': scores[i] 159. } 160. results.append(result) 161. return results 162. def annotate\_objects(annotator, results, labels): 163. global id, score, minx, miny, maxx, maxy 164. init() 165. for obj in results: 166. ymin, xmin, ymax, xmax = obj['bounding\_box'] 167. xmin = int(xmin \* CAMERA\_WIDTH) 168. xmax = int(xmax \* CAMERA\_WIDTH) 169. ymin = int(ymin \* CAMERA\_HEIGHT) 170. ymax = int(ymax \* CAMERA\_HEIGHT) 171. annotator.bounding\_box([xmin, ymin, xmax, ymax]) 172. annotator.text([xmin, ymin], 173. '%s\n%.2f' % (labels[obj['class\_id']], obj['score'])) 174. minx.append(xmin) 175. maxx.append(xmax) 176. miny.append(ymin) 177. maxy.append(ymax) 178. id.append(labels[obj['class\_id']]) 179. score.append(obj['score']) 180. def main(): 181. global id, score, minx, miny, maxx, maxy, count 182. global fname, capture\_img, led\_status, btn\_status 183. labels = load\_labels("./tmp/coco\_labels.txt") 184. interpreter = Interpreter("./tmp/detect.tflite") 185. interpreter.allocate\_tensors() 186. \_, input\_height, input\_width, \_ = interpreter.get\_input\_details()[0]['shape'] 187. with picamera.PiCamera( 188. resolution=(CAMERA\_WIDTH, CAMERA\_HEIGHT), framerate=30) as camera: 189. try: 190. stream = io.BytesIO() 191. annotator = Annotator(camera) 192. for \_ in camera.capture\_continuous( 193. stream, format='jpeg', use\_video\_port=True): 194. stream.seek(0) 195. image = Image.open(stream).convert('RGB').resize( 196. (input\_width, input\_height), Image.ANTIALIAS) 197. start\_time = time.monotonic() 198. results = detect\_objects(interpreter, image, 0.4) 199. elapsed\_ms = (time.monotonic() - start\_time) \* 1000 200. annotator.clear() 201. annotate\_objects(annotator, results, labels) 202. annotator.text([5, 0], '%.1fms' % (elapsed\_ms)) 203. annotator.update() 204. stream.seek(0) 205. fname = "%09d.jpg" % count 206. count += 1 207. f= open(fname, "wb") 208. ill = getIlluminance() 209. print("current ill %d" % ill) 210. print("Current temp %4.1d" % getTemperature()) 211. print("Current humi %4.1d%%" % getHumidity()) 212. if(getTemperature() > 30): 213. led\_status = 0 214. GPIO.output(led, led\_status) 215. if(GPIO.input(btn) == 0): 216. if(ill < 3000): 217. led\_status = 1 218. GPIO.output(led, led\_status) 219. btn\_status = 0 220. capture\_img = fname 221. capture() 222. print (capture\_img) 223. f.write(stream.read()) 224. f.close() 225. img = Image.open(fname) 226. draw = ImageDraw.Draw(img) 227. for i in range(len(id)): 228. draw.rectangle(((minx[i],miny[i]),(maxx[i],maxy[i])), outline= "red") 229. font = ImageFont.truetype("arial.ttf", 20) 230. draw.text((minx[i]+5, miny[i]), id[i] + str(score[i]), (255,0,0), font = font) 231. img.save("../static/"+fname) 232. lastfname() 233. currentledstatus() 234. stream.seek(0) 235. stream.truncate() 236. finally: 237. camera.stop\_preview() 238. if \_\_name\_\_ == '\_\_main\_\_': 239. main() |

Jmlee\_flask.py

|  |
| --- |
| 1. import sys 2. sys.path.append("..") 3. import importlib 4. from flask import Flask, render\_template, Response, request, redirect, url\_for 5. import project.det as det\_ 6. app = Flask(\_\_name\_\_,template\_folder = '../templates', static\_url\_path='/', static\_folder= '../static') 7. @app.route("/det") 8. def det(): 9. f = open("/home/pi/mobile/project/lastfname.txt", "r") 10. fname = f.read() 11. print ("%s" %fname) 12. f.close() 13. f = open("/home/pi/mobile/project/ill.txt", "r") 14. ill = f.read() 15. f.close() 16. f = open("/home/pi/mobile/project/temp.txt", "r") 17. temp = f.read() 18. f.close() 19. f = open("/home/pi/mobile/project/humi.txt", "r") 20. humi = f.read() 21. f.close() 22. f = open("/home/pi/mobile/project/led.txt", "r") 23. led = f.read() 24. f.close() 25. return render\_template("pro\_img.html", path = request.path, fname = fname, ill = ill, temp = temp, humi = humi, led = led) 26. @app.route("/det\_cap") 27. def capture(): 28. i = open("/home/pi/mobile/project/capture\_img.txt", "r") 29. fname = i.read() 30. print("%s" % fname) 31. i.close() 32. return render\_template("pro\_capture.html", path = request.path, fname = fname) 33. if \_\_name\_\_ == "\_\_main\_\_": 34. app.run(host='0.0.0.0', port=8000, threaded = True) #flask 속도 원활 |

pro\_img.html

|  |
| --- |
| 1. <html> 2. <meta http-equiv="refresh" content="0.1;url={%print(path)%}"> 3. <body> 4. <img src='{{url\_for('static', filename=fname) }}'> 5. <p> {%print(fname)%} </p> 6. <p> 조도:{%print(ill)%} </p> 7. <p> 온도:{%print(temp)%} </p> 8. <p> 습도:{%print(humi)%} </p> 9. <p> {%print(led)%} </p> 10. </body> 11. </html> |

pro\_capture.html

|  |
| --- |
| 1. <html> 2. <meta http-equiv="refresh" content="0.1;url={%print(path)%}"> 3. <body> 4. <img src='{{url\_for('static', filename=fname) }}'> 5. <p> {%print(fname)%} </p> 6. </body> 7. </html> |