

```

1  """
2  #####
3  Project: Sensor Subsystem
4
5  Authors: Team Spatium Lucis
6
7  Version: v2.0
8
9  Target Device: Raspberry Pi 3
10
11 Files: circadian.py, pir_sensor.py, rgb_sensor.py, send_circadian_values.py, usr_sensor.py,
12 wait_for_cmd.py, start.py, stop.py, compensate.txt, config.txt, sensor_data.txt
13
14 Last edited: June 15, 2017
15
16 #####
17
18 (I) start.py:
19 This is the init script for the sensor subsystem. Usage:
20
21 $ python start.py
22
23 *****DO NOT USE sudo python!!!
24
25 (II) circadian.py:
26 This file serves as a custom Python module that houses functions that are common to all the various sensor subsystem
27 files.
28
29 (a) Module imports:
30     (1) import subprocess
31     (2) import socket
32     (3) import MySQLdb
33     (4) import time
34     (5) import math
35     (6) import datetime
36
37 (b) Functions:
38     (1) def init_circadian_table():
39         This function creates base MASTER_CIRCADIAN_TABLE. Returns a list of lists containing RGB brightnesses
40         percentages for each minute of the day for 7 AM wake up and 11 PM sleep. Example index (not relative to the
41         actual table):
42         MASTER_CIRCADIAN_TABLE[420] == [ 30, 40, 50 ]. Therefore, MASTER_CIRCADIAN_TABLE[420][0] == 30.
43
44     (2) def init_offset_table():
45         This function creates the base MASTER_OFFSET_TABLE. Returns a list of lists containing rgb sensor offset
46         values for each minute of the day. These are needed because the rgb sensor cannot properly pick up the values
47         of the lights so they are offset with these. This function calls init_red_offset(), init_green_offset(),
48         and init_blue_offset() to generate the values for the MASTER_OFFSET_TABLE. This table is for 7 AM to 11 PM
49         cycle. Example index (not relative to the actual table ):
50         MASTER_OFFSET_TABLE[420] == [ 120, 130, 140 ]. Therefore, MASTER_OFFSET_TABLE[420][0] == 120.
51
52     (3) def init_red_offset(MASTER_OFFSET_TABLE):
53         This function generates the red offset values for the MASTER_OFFSET_TABLE.
54
55     (4) def init_green_offset(MASTER_OFFSET_TABLE):
56         This function generates the green offset values for the MASTER_OFFSET_TABLE.
57
58     (5) def init_blue_offset(MASTER_OFFSET_TABLE):
59         This function generates the blue offset values for the MASTER_OFFSET_TABLE.
60
61     (6) def init_master_lux_table():
62         This function creates the lux offset values for the MASTER_LUX_TABLE. Returns a list containing lux offset
63         values. These offset values are needed for the rgb sensor to calculate lux values because it doesn't naturally
64         do so. The values are for every minute of the day base on 7 AM/11 PM cycle. Example index (not relative to
65         the actual table):
66         MASTER_LUX_TABLE[420] == 50.
67
68     (7) def calc_user_tables(WAKE_UP_TIME, MASTER_CIRCADIAN_TABLE, MASTER_OFFSET_TABLE, MASTER_LUX_TABLE):
69         This function takes the user WAKE_UP_TIME, MASTER_CIRCADIAN_TABLE, MASTER_OFFSET_TABLE,
70         and the MASTER_LUX_TABLE then shifts these tables based on the WAKE_UP_TIME. Returns a tuple with the newly
71         shifted tables.
72
73     (8) def calc_Illuminance(lux, distance, angle):
74         This function takes a lux value, distance (in meters), and a viewing angle (in degrees). Returns the lumens
75         value. Uses toArea(), toSr(), and toRad() in calculation. Source:
76
77     (9) def get_pids():
78         Returns a list with all of the process ids (pids) of the following scripts:
79         pir_sensor.py, rgb_sensor.py, usr_sensor.py, wait_for_cmd.py, and send_circadian_values.py
80
81     (10) def get_system_time():
82         Returns the system time in mintues.
83
84     (11) def get_ip():
85         Returns the local IP address of the Raspberry Pi.
86
87     (12) def create_log(cursor, db, message, user_name):
88         This function takes a database cursor object, database object, a message string, and username string. Stores
89         the message into the database using execute_db_query().

```

```

90
91     (13) def execute_dB_query(cursor, db, sql, sql_args):
92         This function takes a database cursor object, database object, an sql string, and a tuple of lists that
93         contain the sql query arguments and executes the query.
94
95     (14) def get_circadian_cmd(USER_CIRCADIANTABLE, PREV_PRIMARY_COLORS, PREV_SECONDARY_COLORS, IS_PRIMARY_DEG,
96         IS_SEC_ON, IS_SEC_DEG):
97         This function takes the USER_CIRCADIANTABLE, PREV_PRIMARY_COLORS, PREV_SECONDARY_COLORS, IS_PRIMARY_DEG,
98         IS_SEC_ON, and IS_SEC_DEG lists as input. Returns a tuple containing the circadian string, list for new
99         previous primary colors, and list for new previous secondary colors.
100
101 (III) pir_sensor.py:
102 This file is for the usage of the PIR motion sensor.
103
104 (a) Module imports:
105     (1) import time
106     (2) import os
107     (3) import signal
108     (4) import subprocess
109     (5) import circadian
110     (6) import MySQLdb
111     (7) import socket
112     (8) import datetime
113     (9) import RPi.GPIO as GPIO
114
115 (b) Signal handling:
116     These are the signal handler setups. When a kill -<number> is issued to the LINUX system, if it is one of the
117     following numbers then it will be handled differently in the Python script. These are basically software
118     interrupts.
119     signal.signal(3, handle_change_cmd)
120     signal.signal(4, catch_other_signals)
121     signal.signal(5, catch_other_signals)
122     signal.signal(6, handle_send_compensation)
123     signal.signal(7, handle_send_circadian)
124     signal.signal(8, handle_wait_for_cmd_dB_connect)
125     signal.signal(10, catch_other_signals)
126     signal.signal(11, handle_rgb_dB_connect)
127     signal.signal(12, handle_usr_dB_connect)
128     signal.signal(15, handle_send_circadian_dB_connect)
129
130     You will notice that throughout the scripts that some signal handler functions don't do anything and that may
131     seem redundant. This was done on purpose for (1) to catch the signal and (2) to keep the signal handling
132     consistent among the scripts.
133
134 (c) Functions:
135     (1) def catch_other_signals(signum, stack):
136         Does nothing but catch signals. Used with signal handling.
137
138     (2) def handle_change_cmd(signum, stack):
139         Catches kill -3. Simply performs time.sleep(3).
140
141     (3) def handle_send_compensation(signum, stack):
142         Catches kill -6. Simply performs time.sleep(3).
143
144     (4) def handle_send_circadian(signum, stack):
145         Catches kill -7. Simply performs time.sleep(3).
146
147     (5) def handle_wait_for_cmd_dB_connect(signum, stack):
148         Catches kill -8. Alerts the pir_sensor.py script that the wait_for_cmd.py script has connected to the database.
149
150     (6) def handle_rgb_dB_connect(signum, stack):
151         Catches kill -11. Alerts the pir_sensor.py script that the rgb_sensor.py script has connected to the database.
152
153     (7) def handle_usr_dB_connect(signum, stack):
154         Catches kill -12. Alerts the pir_sensor.py script that the usr_sensor.py script has connected to the database.
155
156     (8) def handle_send_circadian_dB_connect(signum, stack):
157         Catches kill -15. Alerts the pir_sensor.py script that the send_circadian_values.py script has connected to
158         the database.
159
160     (9) def handle_motion_detection(PIR_PIN):
161         This function is the hardware interrupt handler for the PIR sensor.
162
163 (IV) rgb_sensor.py:
164 This file is for the usage of the RGB sensor.
165
166 (a) Module imports;
167     (1) import time
168     (2) import os
169     (3) import signal
170     (4) import subprocess
171     (5) import circadian
172     (6) import MySQLdb
173     (7) import socket
174     (8) import datetime
175     (9) import smbus
176
177 (b) Signal handling: (See section in pir_sensor.py for more info on signal handling.)
178     signal.signal(3, handle_change_cmd)

```

```

179     signal.signal(4, handle_sleep_mode)
180     signal.signal(5, handle_wake_up)
181     signal.signal(6, catch_other_signals)
182     signal.signal(7, handle_send_circadian)
183     signal.signal(8, handle_wait_for_cmd_dB_connect)
184     signal.signal(10, handle_pir_dB_connect)
185     signal.signal(11, catch_other_signals)
186     signal.signal(12, handle_usr_dB_connect)
187     signal.signal(15, handle_send_circadian_dB_connect)
188

```

(c) Functions:

```

190     (1) def catch_other_signals(signum, stack):
191         Does nothing but catch signals. Used with signal handling.
192
193     (2) def handle_change_cmd(signum, stack):
194         Catches kill -3. Handles when the user changes a parameter on the website.
195
196     (3) def handle_sleep_mode(signum, stack):
197         Catches kill -4. Updates database with sensor reading of 0 when system goes into sleep mode.
198
199     (4) def handle_wake_up(signum, stack):
200         Catches kill -5. Used in waking from sleep mode.
201
202     (5) def handle_send_circadian(signum, stack):
203         Catches kill -7. Simply does time.sleep(3).
204
205     (6) def handle_wait_for_cmd_dB_connect(signum, stack):
206         Catches kill -8. Tells rgb_sensor.py that wait_for_cmd.py connected to the database.
207
208     (7) def handle_pir_dB_connect(signum, stack):
209         Catches kill -10. Tells rgb_sensor.py that pir_sensor.py connected to the database.
210
211     (8) def handle_usr_dB_connect(signum, stack):
212         Catches kill -12. Tells rgb_sensor.py that usr_sensor.py connected to the database.
213
214     (9) def handle_send_circadian_dB_connect(signum, stack):
215         Catches kill -15. Tells rgb_sensor.py that sends_circadian_values.py connected to the database.
216

```

(V) send_circadian_values.py:

This file is responsible for sending circadian values to the lighting subsystem. This sends compensation values as well.

```

219
220     (a) Module imports:
221         (1) import time
222         (2) import os
223         (3) import signal
224         (4) import subprocess
225         (5) import circadian
226         (6) import MySQLdb
227         (7) import socket
228         (8) import datetime
229
230     (b) Signal handling: (See section in pir_sensor.py for more info on signal handling.)
231     signal.signal(3, handle_change_cmd)
232     signal.signal(4, handle_sleep_mode)
233     signal.signal(5, handle_wake_up)
234     signal.signal(6, handle_send_compensation)
235     signal.signal(7, catch_other_signals)
236     signal.signal(8, handle_wait_for_cmd_dB_connect)
237     signal.signal(10, handle_pir_dB_connect)
238     signal.signal(11, handle_rgb_dB_connect)
239     signal.signal(12, handle_usr_dB_connect)
240     signal.signal(15, catch_other_signals)
241

```

(c) Functions:

```

243     (1) def catch_other_signals(signum, stack):
244         Catches signals. Does nothing else.
245
246     (2) def handle_change_cmd(signum, stack):
247         Catches kill -3. Used for when a user changes something from the website.
248
249     (3) def handle_sleep_mode(signum, stack):
250         Catches kill -4. Sends values to the lighting subsystem to put it to sleep.
251
252     (4) def handle_wake_up(signum, stack):
253         Catches kill -5. Makes the script send a value to wake the lights up.
254
255     (5) def handle_send_compensation(signum, stack):
256         Catches kill -6. Sends compensation values to the lighting subsystem.
257
258     (6) def handle_wait_for_cmd_dB_connect(signum, stack):
259         Catches kill -8. Alerts the send_circadian_values.py that wait_for_cmd.py connected to the DB.
260
261     (7) def handle_pir_dB_connect(signum, stack):
262         Catches kill -10. Alerts the send_circadian_values.py that pir_sensor.py connected to the DB.
263
264     (8) def handle_rgb_dB_connect(signum, stack):
265         Catches kill -11. Alerts the send_circadian_values.py that rgb_sensor.py connected to the DB.
266
267     (9) def handle_usr_dB_connect(signum, stack):

```

268 Catches kill -12. Alerts the send_circadian_values.py that usr_sensor.py connected to the DB.

269

270 (VI) usr_sensor.py:

271 This file is responsible for the usage of the ultra sonic range sensor.

272

273 (a) Module imports:

274 (1) import time

275 (2) import os

276 (3) import signal

277 (4) import subprocess

278 (5) import circadian

279 (6) import MySQLdb

280 (7) import socket

281 (8) import datetime

282 (9) import math

283 (10) import RPi.GPIO as GPIO

284

285 (b) Signal handling: (See section in pir_sensor.py for more info on signal handling.)

286 signal.signal(3, catch_other_signals)

287 signal.signal(4, catch_other_signals)

288 signal.signal(5, catch_other_signals)

289 signal.signal(6, catch_other_signals)

290 signal.signal(7, catch_other_signals)

291 signal.signal(8, handle_wait_for_cmd_dB_connect)

292 signal.signal(10, handle_pir_dB_connect)

293 signal.signal(11, handle_rgb_dB_connect)

294 signal.signal(12, catch_other_signals)

295 signal.signal(15, handle_send_circadian_dB_connect)

296

297 (c) Functions:

298 (1) def catch_other_signals(signum, stack):

299 Catches signals. Does nothing else.

300

301 (2) def handle_wait_for_cmd_dB_connect(signum, stack):

302 Catches kill -8. Tells usr_sensor.py that wait_for_cmd.py connected to the DB.

303

304 (3) def handle_pir_dB_connect(signum, stack):

305 Catches kill -10. Tells usr_sensor.py that pir_sensor.py connected to the DB.

306

307 (4) def handle_rgb_dB_connect(signum, stack):

308 Catches kill -11. Tells usr_sensor.py that rgb_sensor.py connected to the DB.

309

310 (5) def handle_send_circadian_dB_connect(signum, stack):

311 Catches kill -15. Tells usr_sensor.py that send_circadian_values.py connected to the DB.

312

313 (VII) wait_for_cmd.py:

314 This file is responsible for receiving commands from the website.

315

316 (a) Module imports:

317 (1) import time

318 (2) import os

319 (3) import signal

320 (4) import subprocess

321 (5) import circadian

322 (6) import MySQLdb

323 (7) import socket

324 (8) import datetime

325

326 (b) Signal handling: (See section in pir_sensor.py for more info on signal handling.)

327 signal.signal(3, catch_other_signals)

328 signal.signal(4, handle_sleep_mode)

329 signal.signal(5, handle_wake_up)

330 signal.signal(6, catch_other_signals)

331 signal.signal(7, catch_other_signals)

332 signal.signal(8, catch_other_signals)

333 signal.signal(10, handle_pir_dB_connect)

334 signal.signal(11, handle_rgb_dB_connect)

335 signal.signal(12, handle_usr_dB_connect)

336 signal.signal(15, handle_send_circadian_dB_connect)

337

338 (c) Functions:

339 (1) def catch_other_signals(signum, stack):

340 Catches signals and does nothing else.

341

342 (2) def handle_pir_dB_connect(signum, stack):

343 Catches kill -10. Tells wait_for_cmd.py that pir_sensor.py connected to the DB.

344

345 (3) def handle_rgb_dB_connect(signum, stack):

346 Catches kill -11. Tells wait_for_cmd.py that rgb_sensor.py connected to the DB.

347

348 (4) def handle_usr_dB_connect(signum, stack):

349 Catches kill -12. Tells wait_for_cmd.py that usr_sensor.py connected to the DB.

350

351 (5) def handle_send_circadian_dB_connect(signum, stack):

352 Catches kill -15. Tells wait_for_cmd.py that send_circadian_values.py connected to the DB.

353

354 (6) def handle_sleep_mode(signum, stack):

355 Catches kill -4. Tells the script that the system entered sleep mode.

356

```

357         (7) def handle_wake_up(signum, stack):
358             Catches kill -5. Tells the script that the system exited sleep mode.
359
360         (8) def boot_up():
361             Initial database check to pair with the lighting subsystem and retrieve previous sensor subsystem settings if
362             any.
363
364 (VIII) stop.py
365 This script is used for killing the sensor subsystem. Usage:
366
367 $ python stop.py
368
369 ****DO NOT USE sudo python!!!
370
371 (IX) pause.py
372 This script is used for suspending the sensor and lighting subsystems. Usage:
373
374 $ python pause.py
375
376 ****DO NOT USE sudo python!!!
377
378 (X) compensate.txt:
379 Holds the compensation data.
380
381 (XI) sensor_data.txt:
382 Holds the sensor readings. Uses in compensation.
383
384 (XII) config.txt:
385 Holds the values that were sent by the user. Needed because DB was misbehaving.
386 """

```