Version Lab10

Makefile Content

ghc: ghc.o ghcontrol.o pisensehat.o

gcc -g -o ghc ghc.o ghcontrol.o pisensehat.o -lwiringPi

ghc.o: ghc.c ghcontrol.h pisensehat.h

gcc -g -c ghc.c

ghcontrol.o: ghcontrol.c ghcontrol.h pisensehat.h

gcc -g -c ghcontrol.c

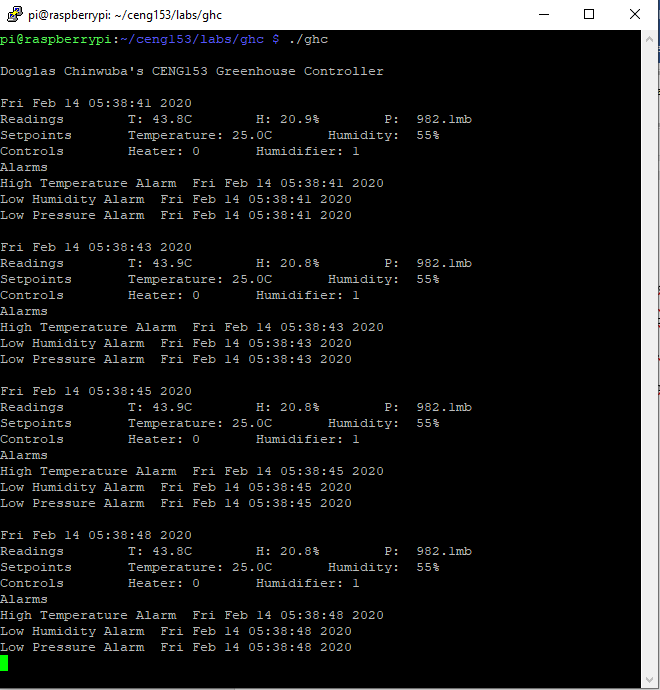
pisensehat.o: pisensehat.c pisensehat.h

gcc -g -c pisensehat.c

clean:

touch \*

rm \*.o



# Table of Contents

File Index 1

File Documentation 1

ghc/ghc.c 1

ghc/ghcontrol.c 2

ghc/ghcontrol.h 11

ghc/pisensehat.c 15

ghc/pisensehat.h 22

Index 26

# File Index

## File List

Here is a list of all files with brief descriptions:

**ghc/ghc.c**

**ghc/ghcontrol.c**

**ghc/ghcontrol.h**

**ghc/pisensehat.c**

**ghc/pisensehat.h**

# File Documentation

## ghc/ghc.c File Reference

#include "ghcontrol.h"

### Functions

* int **main** (void)

### Function Documentation

#### int main (void )

Defines the entry point for the console application. \*

##### Version:

**ghc.c**

7 {

8 time\_t now;

9 struct readings creadings = {0};

10 struct controls ctrl = {0};

11 struct setpoints sets = {0};

12 int logged;

13 alarmlimit\_s alimits = {0};

14 alarm\_s warn[NALARMS] = {0};

15 sets = GhSetSetpoints();

16 alimits = GhSetAlarmLimits();

17 GhControllerInit();

18 while(1)

19 {

20 now = time(NULL);

21 GhGetSetpoints();

22 creadings = GhGetReadings();

23 logged = GhLogData("ghdata.txt",creadings);

24 ctrl = GhSetControls(sets,creadings);

25 GhSetAlarms(warn,alimits,creadings);

26 GhDisplayAll(creadings,sets);

27 GhDisplayReadings(creadings);

28 GhDisplaySetpoints(sets);

29 GhDisplayControls(ctrl);

30 GhDisplayAlarms(warn);

31 GhDelay(GHUPDATE);

32 }

33 return 1;

34 }

## ghc/ghcontrol.c File Reference

#include <stdio.h>

#include "ghcontrol.h"

### Functions

* void **GhDisplayHeader** (const char \*sname)
* int **GhGetRandom** (int range)
* void **GhDelay** (int milliseconds)
* void **GhControllerInit** (void)
* void **GhGetControls** ()
* **control\_s** **GhSetControls** (**setpoint\_s** target, **reading\_s** rdata)
* void **GhGetSetpoints** (void)
* **setpoint\_s** **GhSetSetpoints** (void)
* void **GhDisplayReadings** (**reading\_s** rdata)
* void **GhDisplaySetpoints** (**setpoint\_s** sdata)
* void **GhDisplayControls** (**control\_s** ctrl)
* double **GhGetTemperature** (void)
* double **GhGetHumidity** (void)
* double **GhGetPressure** (void)
* **reading\_s** **GhGetReadings** (void)
* int **GhLogData** (char \*fname, **reading\_s** ghdata)
* int **GhSaveSetpoints** (char \*fname, **setpoint\_s** spts)
* **setpoint\_s** **GhRetrieveSetpoints** (char \*fname)
* void **GhDisplayAll** (**reading\_s** rd, **setpoint\_s** sd)
* **alarmlimit\_s** **GhSetAlarmLimits** (void)
* void **GhSetAlarms** (**alarm\_s** calarm[**NALARMS**], **alarmlimit\_s** alarmpt, **reading\_s** rdata)
* void **GhDisplayAlarms** (**alarm\_s** alrm[**NALARMS**])

### Variables

* const char **alarmnames** [**NALARMS**][**ALARMNMSZ**] = {"No Alarms","High Temperature","Low Temperature","High Humidity","Low Humidity","High Pressure","Low Pressure"}

### Function Documentation

#### void GhControllerInit (void )

Passes operator name to GhDisplayHeader

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

void

64 {

65 srand((unsigned) time(NULL));

66 GhDisplayHeader("Douglas Chinwuba");

67 #if SENSEHAT

68 wiringPiSetup();

69 ShInit();

70 #endif

71 }

#### void GhDelay (int *milliseconds*)

Delays program for two seconds

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *int* | milliseconds |

##### Returns:

void

43 {

44 long wait;

45 clock\_t now,start;

46

47 wait = milliseconds\*(CLOCKS\_PER\_SEC/1000);

48 start = clock();

49 now = start;

50 while( (now-start) < wait )

51 {

52 now = clock();

53 }

54 }

#### void GhDisplayAlarms (alarm\_s *alrm*[NALARMS])

Displays alarms

##### Version:

09APR2020

##### Author:

Paul Maggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *array* | alarm\_s alrm[NALARMS] |

##### Returns:

void

424 {

425 fprintf(stdout,"Alarms\n");

426 for(int i = 1;i < 7;i++)

427 {

428 if(alrm[i].code != NOALARM)

429 {

430 fprintf(stdout,"%s Alarm %s",alarmnames[i],ctime(&alrm[i].atime));

431 }

432 }

433 }

#### void GhDisplayAll (reading\_s *rd*, setpoint\_s *sd*)

Displays bar graph on pisensehat

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *struct* | readings rd |
| *struct* | setpoints sd |

##### Returns:

void

306 {

307 int rv = 0;

308 int sv = 0;

309 int avh = 0;

310 int avl = 0;

311 fbpixel\_s pxc = {0};

312 ShClearMatrix();

313

314 rv = (8.0 \* ((rd.temperature / (USTEMP - LSTEMP)) + 0.05)) - 1.0;

315 sv = (8.0 \* ((rd.temperature / (USTEMP - LSTEMP)) + 0.05)) - 1.0;

316 pxc.red = 0x00;

317 pxc.green = 0xFF;

318 pxc.blue = 0x00;

319 ShSetVerticalBar(TBAR,pxc,rv);

320 pxc.red = 0x0F;

321 pxc.green = 0x0F;

322 pxc.blue = 0x00;

323 ShSetPixel(TBAR,sv,pxc);

324

325 rv = (8.0 \* ((rd.humidity / (USHUMID - LSHUMID)) + 0.05)) - 1.0;

326 sv = (8.0 \* ((rd.humidity / (USHUMID - LSHUMID)) + 0.05)) - 1.0;

327 pxc.red = 0x00;

328 pxc.green = 0xFF;

329 pxc.blue = 0x00;

330 ShSetVerticalBar(HBAR,pxc,rv);

331 pxc.red = 0x0F;

332 pxc.green = 0x0F;

333 pxc.blue = 0x00;

334 ShSetPixel(HBAR,sv,pxc);

335

336 rv = (8.0 \* ((rd.pressure / (USPRESS - LSPRESS)) + 0.05)) - 1.0;

337 pxc.red = 0x00;

338 pxc.green = 0xFF;

339 pxc.blue = 0x00;

340 ShSetVerticalBar(PBAR,pxc,rv);

341 }

#### void GhDisplayControls (control\_s *ctrl*)

Prints heater and humidifier on or off status

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *struct* | controls ctrl |

##### Returns:

void

155 {

156 fprintf(stdout,"Controls Heater: %d Humidifier: %d\n", ctrl.heater,ctrl.humidifier);

157 }

#### void GhDisplayHeader (const char \* *sname*)

Prints Gh Controller Title

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *sname* | string with Operator's name |

##### Returns:

void

19 {

20 fprintf(stdout,"\n%s's CENG153 Greenhouse Controller\n",sname);

21 }

#### void GhDisplayReadings (reading\_s *rdata*)

Prints time and readings

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *struct* | readings rdata |

##### Returns:

void

131 {

132 fprintf(stdout,"\n%sReadings\tT: %4.1lfC\tH: %4.1lf%%\tP: %6.1lfmb", ctime(&rdata.rtime), rdata.temperature, rdata.humidity, rdata.pressure);

133 }

#### void GhDisplaySetpoints (setpoint\_s *sdata*)

Prints setpoints

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *struct* | setpoints sdata |

##### Returns:

void

143 {

144 fprintf(stdout,"\nSetpoints Temperature: %3.1lfC Humidity: %3.0lf%\n",sdata.temperature, sdata.humidity);

145 }

#### void GhGetControls ()

74 {

75 }

#### double GhGetHumidity (void )

Gets humidity value

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

double humidity data

186 {

187 #if SIMHUMIDITY

188 return GhGetRandom(USHUMID - LSHUMID) + LSHUMID;

189 #else

190 ht221sData\_s ch = {0};

191

192 ch = ShGetHT221SData();

193 return ch.humidity;

194 #endif

195 }

#### double GhGetPressure (void )

Gets pressure value

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

double pressure data

205 {

206 #if SIMPRESSURE

207 return GhGetRandom(USPRESS - LSPRESS) + LSPRESS;

208 #else

209 lps25hData\_s cp = {0};

210

211 cp = ShGetLPS25HData();

212 return cp.pressure;

213 #endif

214 }

#### int GhGetRandom (int *range*)

Gets random integer

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *int* | range |

##### Returns:

int rand()range

31 {

32 return rand()%range;

33 }

#### reading\_s GhGetReadings (void )

218 {

219 reading\_s now = {0};

220 now.rtime = time(NULL);

221 now.temperature = GhGetTemperature();

222 now.humidity = GhGetHumidity();

223 now.pressure = GhGetPressure();

224 return now;

225 }

#### void GhGetSetpoints (void )

103 {

104 }

#### double GhGetTemperature (void )

Gets temperature value

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

double temperature data

167 {

168 #if SIMTEMPERATURE

169 return GhGetRandom(USTEMP - LSTEMP) + LSTEMP;

170 #else

171 ht221sData\_s ct = {0};

172

173 ct = ShGetHT221SData();

174 return ct.temperature;

175 #endif

176 }

#### int GhLogData (char \* *fname*, reading\_s *ghdata*)

Logs data into file

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *fname* | string with ghdata |
| *struct* | readings ghdata |

##### Returns:

0 or 1

236 {

237 FILE \*fp;

238 char ltime[25];

239 fp = fopen(fname,"w");

240 if (fp == NULL)

241 {

242 printf("\nCan't open file. data not retrived!\n");

243 return 0;

244 }

245 strcpy(ltime,ctime(&ghdata.rtime));

246 ltime[3] = ',';

247 ltime[7] = ',';

248 ltime[10] = ',';

249 ltime[19] = ',';

250 fprintf(fp,"\n%.24s,%3.1lf,%3.1lf,%5.1lf",ltime,ghdata.temperature,ghdata.humidity,ghdata.pressure);

251 fclose(fp);

252 return 1;

253 }

#### setpoint\_s GhRetrieveSetpoints (char \* *fname*)

281 {

282 setpoint\_s spts = {0.0};

283 FILE \*fp;

284 fp = fopen(fname,"r");

285 if (fp == NULL)

286 {

287 return spts;

288 }

289 else

290 {

291 fread(&spts,sizeof(setpoint\_s),1,fp);

292 fclose(fp);

293 return spts;

294 }

295 }

#### int GhSaveSetpoints (char \* *fname*, setpoint\_s *spts*)

Saves setpoints data in file

##### Version:

09ARP2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *fnmae* | string |
| *struct* | setpoints spts |

##### Returns:

0 or 1

264 {

265 FILE \*fp;

266 fp = fopen(fname,"w");

267 if (fp == NULL)

268 {

269 return 0;

270 }

271 else

272 {

273 fwrite(&spts,sizeof(setpoint\_s),1,fp);

274 fclose(fp);

275 return 1;

276 }

277 }

#### alarmlimit\_s GhSetAlarmLimits (void )

345 {

346 alarmlimit\_s calarm = {0};

347 calarm.hight = UPPERATEMP;

348 calarm.lowt = LOWERATEMP;

349 calarm.highh = UPPERAHUMID;

350 calarm.lowh = LOWERAHUMID;

351 calarm.highp = UPPERAPRESS;

352 calarm.lowp = LOWERAPRESS;

353 return calarm;

354 }

#### void GhSetAlarms (alarm\_s *calarm*[NALARMS], alarmlimit\_s *alarmpt*, reading\_s *rdata*)

Sets alarms

##### Version:

09APR2020

##### Author:

Paul Moggach

Douglas Chinwuba

##### Parameters:

|  |  |
| --- | --- |
| *array* | alarm\_s calarm[NALARM] |
| *struct* | alarmlimits alarmpt |
| *struct* | readings rdata |

##### Returns:

void

366 {

367 int i;

368 for(i = 0;i < 7;i++)

369 {

370 calarm[i].code = NOALARM;

371 }

372

373 if(rdata.temperature >= alarmpt.hight)

374 {

375 calarm[HTEMP].code = HTEMP;

376 calarm[HTEMP].atime = rdata.rtime;

377 calarm[HTEMP].value = rdata.temperature;

378 }

379

380 if(rdata.temperature <= alarmpt.lowt)

381 {

382 calarm[LTEMP].code = LTEMP;

383 calarm[LTEMP].atime = rdata.rtime;

384 calarm[LTEMP].value = rdata.temperature;

385 }

386

387 if(rdata.humidity >= alarmpt.highh)

388 {

389 calarm[HHUMID].code = HHUMID;

390 calarm[HHUMID].atime = rdata.rtime;

391 calarm[HHUMID].value = rdata.humidity;

392 }

393

394 if(rdata.humidity <= alarmpt.lowh)

395 {

396 calarm[LHUMID].code = LHUMID;

397 calarm[LHUMID].atime = rdata.rtime;

398 calarm[LHUMID].value = rdata.humidity;

399 }

400

401 if(rdata.pressure >= alarmpt.highp)

402 {

403 calarm[HPRESS].code = HPRESS;

404 calarm[HPRESS].atime = rdata.rtime;

405 calarm[HPRESS].value = rdata.pressure;

406 }

407

408 if(rdata.pressure <= alarmpt.lowp)

409 {

410 calarm[LPRESS].code = LPRESS;

411 calarm[LPRESS].atime = rdata.rtime;

412 calarm[LPRESS].value = rdata.pressure;

413 }

414 }

#### control\_s GhSetControls (setpoint\_s *target*, reading\_s *rdata*)

80 {

81 control\_s cset = {0};

82 if (rdata.temperature < target.temperature)

83 {

84 cset.heater = ON;

85 }

86 else

87 {

88 cset.heater = OFF;

89 }

90

91 if (rdata.humidity < target.humidity)

92 {

93 cset.humidifier = ON;

94 }

95 else

96 {

97 cset.humidifier = OFF;

98 }

99 return cset;

100 }

#### setpoint\_s GhSetSetpoints (void )

108 {

109 setpoint\_s cpoints = {0};

110 cpoints.temperature = STEMP;

111 cpoints.humidity = SHUMID;

112 cpoints = GhRetrieveSetpoints("setpoints.dat");

113 if (cpoints.temperature == 0)

114 {

115 cpoints.temperature = STEMP;

116 cpoints.humidity = SHUMID;

117 GhSaveSetpoints("setpoints.dat",cpoints);

118 }

119

120 return cpoints;

121 }

### Variable Documentation

#### const char alarmnames[NALARMS][ALARMNMSZ] = {"No Alarms","High Temperature","Low Temperature","High Humidity","Low Humidity","High Pressure","Low Pressure"}

Function definitions for Gh control code

##### Version:

**ghcontrol.c**

## ghc/ghcontrol.h File Reference

#include <stdlib.h>

#include <time.h>

#include <string.h>

#include <wiringPi.h>

#include "pisensehat.h"

### Data Structures

* struct **readings**
* struct **setpoints**
* struct **controls**
* struct **alarmlimits**
* struct **alarms**

### Macros

* #define **GHUPDATE**  2000
* #define **SENSORS**  3
* #define **TEMPERATURE**  0
* #define **HUMIDITY**  1
* #define **PRESSURE**  2
* #define **SIMULATE**  0
* #define **USTEMP**  50
* #define **LSTEMP**  -10
* #define **USHUMID**  100
* #define **LSHUMID**  0
* #define **USPRESS**  1016
* #define **LSPRESS**  985
* #define **STEMP**  25.0
* #define **SHUMID**  55.0
* #define **ON**  1
* #define **OFF**  0
* #define **SIMTEMPERATURE**  0
* #define **SIMHUMIDITY**  0
* #define **SIMPRESSURE**  0
* #define **NUMBERS**  8
* #define **NUMPTS**  8.0
* #define **TBAR**  7
* #define **HBAR**  5
* #define **PBAR**  3
* #define **SENSEHAT**  1
* #define **NALARMS**  7
* #define **LOWERATEMP**  10
* #define **UPPERATEMP**  30
* #define **LOWERAHUMID**  25
* #define **UPPERAHUMID**  70
* #define **LOWERAPRESS**  985
* #define **UPPERAPRESS**  1016
* #define **ALARMNMSZ**  18

### Typedefs

* typedef struct **readings** **reading\_s**
* typedef struct **setpoints** **setpoint\_s**
* typedef struct **controls** **control\_s**
* typedef struct **alarmlimits** **alarmlimit\_s**
* typedef struct **alarms** **alarm\_s**

### Enumerations

* enum **alarm\_e** { **NOALARM**, **HTEMP**, **LTEMP**, **HHUMID**, **LHUMID**, **HPRESS**, **LPRESS** }

### Data Structure Documentation

#### struct readings

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | humidity |  |
| double | pressure |  |
| time\_t | rtime |  |
| double | temperature |  |

#### struct setpoints

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | humidity |  |
| double | temperature |  |

#### struct controls

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| int | heater |  |
| int | humidifier |  |

#### struct alarmlimits

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | highh |  |
| double | highp |  |
| double | hight |  |
| double | lowh |  |
| double | lowp |  |
| double | lowt |  |

#### struct alarms

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| time\_t | atime |  |
| **alarm\_e** | code |  |
| double | value |  |

### Macro Definition Documentation

#### #define ALARMNMSZ  18

#### #define GHUPDATE  2000

Constants, structures, function prototype for Gh control code

##### Version:

**ghcontrol.h**

#### #define HBAR  5

#### #define HUMIDITY  1

#### #define LOWERAHUMID  25

#### #define LOWERAPRESS  985

#### #define LOWERATEMP  10

#### #define LSHUMID  0

#### #define LSPRESS  985

#### #define LSTEMP  -10

#### #define NALARMS  7

#### #define NUMBERS  8

#### #define NUMPTS  8.0

#### #define OFF  0

#### #define ON  1

#### #define PBAR  3

#### #define PRESSURE  2

#### #define SENSEHAT  1

#### #define SENSORS  3

#### #define SHUMID  55.0

#### #define SIMHUMIDITY  0

#### #define SIMPRESSURE  0

#### #define SIMTEMPERATURE  0

#### #define SIMULATE  0

#### #define STEMP  25.0

#### #define TBAR  7

#### #define TEMPERATURE  0

#### #define UPPERAHUMID  70

#### #define UPPERAPRESS  1016

#### #define UPPERATEMP  30

#### #define USHUMID  100

#### #define USPRESS  1016

#### #define USTEMP  50

### Typedef Documentation

#### typedef struct alarms alarm\_s

#### typedef struct alarmlimits alarmlimit\_s

#### typedef struct controls control\_s

#### typedef struct readings reading\_s

#### typedef struct setpoints setpoint\_s

### Enumeration Type Documentation

#### enum alarm\_e

##### Enumerator:

|  |  |
| --- | --- |
| NOALARM |  |
| HTEMP |  |
| LTEMP |  |
| HHUMID |  |
| LHUMID |  |
| HPRESS |  |
| LPRESS |  |

48 {NOALARM,HTEMP,LTEMP,HHUMID,LHUMID,HPRESS,LPRESS}alarm\_e;

## ghc/pisensehat.c File Reference

#include "pisensehat.h"

### Functions

* int **ShInit** (void)
* int **ShExit** (void)
* void **ShClearMatrix** (void)
* uint8\_t **ShSetPixel** (int x, int y, **fbpixel\_s** px)
* **fbpixel\_s** **ShGetPixel** (int x, int y)
* int **ShSetVerticalBar** (int bar, **fbpixel\_s** px, uint8\_t value)
* **lps25hData\_s** **ShGetLPS25HData** (void)
* **ht221sData\_s** **ShGetHT221SData** (void)

### Variables

* static int **fbfd**
* static uint16\_t \* **map**
* static int **HTS221fd**
* static int **LPS25Hfd**

### Function Documentation

#### void ShClearMatrix (void )

Clears Sensehat 8X8 RGB LED display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

void

100 {

101 memset(map, 0, FILESIZE);

102 }

#### int ShExit (void )

Closes Down the Sensehat

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

status an int, 1 if successful

76 {

77 int status = 1;

78 ShClearMatrix();

79 /\* un-map and close \*/

80 if (munmap(map, FILESIZE) == -1)

81 {

82 perror("Error un-mmapping the file");

83 status=0;

84 return status;

85 }

86 close(fbfd);

87 close(HTS221fd);

88 close(LPS25Hfd);

89 return status;

90 }

#### ht221sData\_s ShGetHT221SData (void )

Gets HT221S Sensehat sensor data

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

ht221sData\_s temperature and humidity data

248 {

249 ht221sData\_s rd = {0};

250 int status;

251 uint8\_t t0\_out\_l,t0\_out\_h,t1\_out\_l,t1\_out\_h;

252 uint8\_t t0\_degC\_x8,t1\_degC\_x8,t1\_t0\_msb;

253 int16\_t T0\_OUT,T1\_OUT;

254 uint16\_t T0\_DegC\_x8,T1\_DegC\_x8;

255 double T0\_DegC,T1\_DegC;

256 double t\_gradient\_m,t\_intercept\_c;

257 uint8\_t t\_out\_l,t\_out\_h;

258 int16\_t T\_OUT;

259 uint8\_t h0\_out\_l,h0\_out\_h,h1\_out\_l,h1\_out\_h,h0\_rh\_x2,h1\_rh\_x2,h\_t\_out\_l,h\_t\_out\_h;

260 int16\_t H0\_T0\_OUT,H1\_T0\_OUT,H\_T\_OUT;

261 double H0\_rH,H1\_rH,h\_gradient\_m,h\_intercept\_c;

262

263 // Power down the device (clean start)

264 wiringPiI2CWriteReg8(HTS221fd, CTRL\_REG1, 0x00);

265 // Turn on the humidity sensor analog front end in single shot mode

266 wiringPiI2CWriteReg8(HTS221fd, CTRL\_REG1, 0x84);

267 // Run one-shot measurement (temperature and humidity). The set bit will be reset by the

268 // sensor itself after execution (self-clearing bit)

269 wiringPiI2CWriteReg8(HTS221fd, CTRL\_REG2, 0x01);

270

271 // Wait until the measurement is completed

272 do

273 {

274 usleep(HTS221DELAY); // 25 ms

275 status = wiringPiI2CReadReg8(HTS221fd, CTRL\_REG2);

276 }

277 while (status != 0);

278

279 // Read calibration temperature LSB (ADC) data

280 // (temperature calibration x-data for two points)

281 t0\_out\_l = wiringPiI2CReadReg8(HTS221fd, T0\_OUT\_L);

282 t0\_out\_h = wiringPiI2CReadReg8(HTS221fd, T0\_OUT\_H);

283 t1\_out\_l = wiringPiI2CReadReg8(HTS221fd, T1\_OUT\_L);

284 t1\_out\_h = wiringPiI2CReadReg8(HTS221fd, T1\_OUT\_H);

285

286 // Read calibration relative humidity LSB (ADC) data

287 // (humidity calibration x-data for two points)

288 h0\_out\_l = wiringPiI2CReadReg8(HTS221fd, H0\_T0\_OUT\_L);

289 h0\_out\_h = wiringPiI2CReadReg8(HTS221fd, H0\_T0\_OUT\_H);

290 h1\_out\_l = wiringPiI2CReadReg8(HTS221fd, H1\_T0\_OUT\_L);

291 h1\_out\_h = wiringPiI2CReadReg8(HTS221fd, H1\_T0\_OUT\_H);

292

293 // Read calibration temperature (°C) data

294 // (temperature calibration y-data for two points)

295 t0\_degC\_x8 = wiringPiI2CReadReg8(HTS221fd, T0\_degC\_x8);

296 t1\_degC\_x8 = wiringPiI2CReadReg8(HTS221fd, T1\_degC\_x8);

297 t1\_t0\_msb = wiringPiI2CReadReg8(HTS221fd, T1\_T0\_MSB);

298

299 // Read relative humidity (% rH) data

300 // (humidity calibration y-data for two points)

301 h0\_rh\_x2 = wiringPiI2CReadReg8(HTS221fd, H0\_rH\_x2);

302 h1\_rh\_x2 = wiringPiI2CReadReg8(HTS221fd, H1\_rH\_x2);

303

304 // make 16 bit values (bit shift)

305 // (temperature calibration x-values)

306 T0\_OUT = t0\_out\_h << 8 | t0\_out\_l;

307 T1\_OUT = t1\_out\_h << 8 | t1\_out\_l;

308

309 // make 16 and 10 bit values (bit mask and bit shift)

310 T0\_DegC\_x8 = (t1\_t0\_msb & 3) << 8 | t0\_degC\_x8;

311 T1\_DegC\_x8 = ((t1\_t0\_msb & 12) >> 2) << 8 | t1\_degC\_x8;

312

313 // Calculate calibration values

314 // (temperature calibration y-values)

315 T0\_DegC = T0\_DegC\_x8 / 8.0;

316 T1\_DegC = T1\_DegC\_x8 / 8.0;

317

318 // Solve the linear equasions 'y = mx + c' to give the

319 // calibration straight line graphs for temperature and humidity

320 t\_gradient\_m = (T1\_DegC - T0\_DegC) / (T1\_OUT - T0\_OUT);

321 t\_intercept\_c = T1\_DegC - (t\_gradient\_m \* T1\_OUT);

322

323 // Read the ambient temperature measurement (2 bytes to read)

324 t\_out\_l = wiringPiI2CReadReg8(HTS221fd, TEMP\_OUT\_L);

325 t\_out\_h = wiringPiI2CReadReg8(HTS221fd, TEMP\_OUT\_H);

326

327 // make 16 bit value

328 T\_OUT = t\_out\_h << 8 | t\_out\_l;

329

330 // make 16 bit values (bit shift)

331 // (humidity calibration x-values)

332 H0\_T0\_OUT = h0\_out\_h << 8 | h0\_out\_l;

333 H1\_T0\_OUT = h1\_out\_h << 8 | h1\_out\_l;

334

335 // Humidity calibration values

336 // (humidity calibration y-values)

337 H0\_rH = h0\_rh\_x2 / 2.0;

338 H1\_rH = h1\_rh\_x2 / 2.0;

339 h\_gradient\_m = (H1\_rH - H0\_rH) / (H1\_T0\_OUT - H0\_T0\_OUT);

340 h\_intercept\_c = H1\_rH - (h\_gradient\_m \* H1\_T0\_OUT);

341

342 // Read the ambient humidity measurement (2 bytes to read)

343 h\_t\_out\_l = wiringPiI2CReadReg8(HTS221fd, H\_T\_OUT\_L);

344 h\_t\_out\_h = wiringPiI2CReadReg8(HTS221fd, H\_T\_OUT\_H);

345

346 // make 16 bit value

347 H\_T\_OUT = h\_t\_out\_h << 8 | h\_t\_out\_l;

348

349 // Power down the device

350 wiringPiI2CWriteReg8(HTS221fd, CTRL\_REG1, 0x00);

351

352 // Calculate and return ambient temperature

353 rd.temperature = (t\_gradient\_m \* T\_OUT) + t\_intercept\_c;

354 rd.humidity = (h\_gradient\_m \* H\_T\_OUT) + h\_intercept\_c;

355 return rd;

356 }

#### lps25hData\_s ShGetLPS25HData (void )

Gets LPS25H Sensehat sensor information

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

lps25hData\_s pressure and temperature data

188 {

189 lps25hData\_s rd = {0};

190

191 uint8\_t temp\_out\_l = 0, temp\_out\_h = 0;

192 int16\_t temp\_out = 0;

193 uint8\_t press\_out\_xl = 0;

194 uint8\_t press\_out\_l = 0;

195 uint8\_t press\_out\_h = 0;

196 int32\_t press\_out = 0;

197 uint8\_t status = 0;

198

199 // Power down the device (clean start)

200 wiringPiI2CWriteReg8(LPS25Hfd, CTRL\_REG1, 0x00);

201

202 // Turn on the humidity sensor analog front end in single shot mode

203 wiringPiI2CWriteReg8(LPS25Hfd, CTRL\_REG1, 0x84);

204

205 // Run one-shot measurement (temperature and humidity). The set bit will be reset by the

206 // sensor itself after execution (self-clearing bit)

207 wiringPiI2CWriteReg8(LPS25Hfd, CTRL\_REG2, 0x01);

208

209 // Wait until the measurement is completed

210 do

211 {

212 usleep(HTS221DELAY); // 25 ms

213 status = wiringPiI2CReadReg8(LPS25Hfd, CTRL\_REG2);

214 }

215 while (status != 0);

216

217 /\* Read the temperature measurement (2 bytes to read) \*/

218 temp\_out\_l = wiringPiI2CReadReg8(LPS25Hfd, TEMP\_OUT\_L);

219 temp\_out\_h = wiringPiI2CReadReg8(LPS25Hfd, TEMP\_OUT\_H);

220

221 /\* Read the pressure measurement (3 bytes to read) \*/

222 press\_out\_xl = wiringPiI2CReadReg8(LPS25Hfd, PRESS\_OUT\_XL);

223 press\_out\_l = wiringPiI2CReadReg8(LPS25Hfd, PRESS\_OUT\_L);

224 press\_out\_h = wiringPiI2CReadReg8(LPS25Hfd, PRESS\_OUT\_H);

225

226 /\* make 16 and 24 bit values (using bit shift) \*/

227 temp\_out = temp\_out\_h << 8 | temp\_out\_l;

228 press\_out = press\_out\_h << 16 | press\_out\_l << 8 | press\_out\_xl;

229

230 /\* calculate output values \*/

231 rd.temperature = 42.5 + (temp\_out / 480.0);

232 rd.pressure = press\_out / 4096.0;

233

234 // Power down the device

235 wiringPiI2CWriteReg8(LPS25Hfd, CTRL\_REG1, 0x00);

236

237 return rd;

238 }

#### fbpixel\_s ShGetPixel (int *x*, int *y*)

Gets pixel information from the Sensehat display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *x* | an integer position value |
| *y* | an integer position value |

##### Returns:

cpx fbpixel\_s pixel colour data

135 {

136 fbpixel\_s cpx = {0};

137 int i;

138

139 if (x >= 0 && x < 8 && y >= 0 && y < 8)

140 {

141 i = (y\*8)+x; // offset into array

142 cpx.red = (map[i] & RGB565\_RED) >> 11;

143 cpx.green = (map[i] & RGB565\_GREEN) >> 5;

144 cpx.blue = (map[i] & RGB565\_BLUE);

145 }

146 return cpx;

147 }

#### int ShInit (void )

Initialize Sensehat

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *void* |  |

##### Returns:

status an int, 1 if successful

20 {

21 int status = 1;

22 struct fb\_fix\_screeninfo fix\_info;

23

24 // Frame Buffer Initialization for 8X8 LED Matrix

25 /\* open the led frame buffer device \*/

26 fbfd = open(FILEPATH, O\_RDWR);

27 if (fbfd == -1)

28 {

29 perror("Error (call to 'open')");

30 exit(EXIT\_FAILURE);

31 }

32

33 /\* read fixed screen info for the open device \*/

34 if (ioctl(fbfd, FBIOGET\_FSCREENINFO, &fix\_info) == -1)

35 {

36 perror("Error (call to 'ioctl')");

37 close(fbfd);

38 exit(EXIT\_FAILURE);

39 }

40

41 /\* now check the correct device has been found \*/

42 if (strcmp(fix\_info.id, "RPi-Sense FB") != 0)

43 {

44 printf("%s\n", "Error: RPi-Sense FB not found");

45 close(fbfd);

46 exit(EXIT\_FAILURE);

47 }

48

49 /\* map the led frame buffer device into memory \*/

50 map = mmap(NULL, FILESIZE, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fbfd, 0);

51 if (map == MAP\_FAILED)

52 {

53 close(fbfd);

54 perror("Error mmapping the file");

55 exit(EXIT\_FAILURE);

56 }

57

58 // Sensor Initialization

59 HTS221fd = wiringPiI2CSetup(HTS221I2CADDRESS);

60 LPS25Hfd = wiringPiI2CSetup(LPS25HI2CADDRESS);

61

62 // Power down the device (clean start)

63 wiringPiI2CWriteReg8(HTS221fd, CTRL\_REG1, 0x00);

64 wiringPiI2CWriteReg8(LPS25Hfd, CTRL\_REG1, 0x00);

65 return status;

66 }

#### uint8\_t ShSetPixel (int *x*, int *y*, fbpixel\_s *px*)

Sets a pixel on the Sensehat display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *x* | an integer position value |
| *y* | an integer position value |
| *fbpixel\_s* | pixel colour data |

##### Returns:

uint8\_t 1 if successful

114 {

115 int i;

116

117 if (x >= 0 && x < 8 && y >= 0 && y < 8)

118 {

119 i = (y\*8)+x; // offset into array

120 map[i] = (px.red << 11) | (px.green << 5) | (px.blue);

121 return 1;

122 }

123 return 0;

124 }

#### int ShSetVerticalBar (int *bar*, fbpixel\_s *px*, uint8\_t *value*)

Sets a vertical bar on the Sensehat display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-01-14

##### Parameters:

|  |  |
| --- | --- |
| *int* | bar to light |
| *fbpixel\_s* | pixel colour data |
| *uint8\_t* | value how many pixels to light in bar |

##### Returns:

int 1 if successful

159 {

160 int i;

161

162 if (bar >= 0 && bar < 8 && value >= 0 && value < 8)

163 {

164 for(i=0; i< value; i++)

165 {

166 ShSetPixel(bar,i,px);

167 }

168 px.red = 0x00;

169 px.green = 0x00;

170 px.blue = 0x00;

171 for(i=value; i< 8;i++)

172 {

173 ShSetPixel(bar,i,px);

174 }

175 return 1;

176 }

177 return 0;

178 }

### Variable Documentation

#### int fbfd[static]

RPi Sensehat functions

##### Version:

**pisensehat.c** 2020-01-15

#### int HTS221fd[static]

#### int LPS25Hfd[static]

#### uint16\_t\* map[static]

#### 

## ghc/pisensehat.h File Reference

#include <stdlib.h>

#include <stdint.h>

#include <unistd.h>

#include <stdio.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <string.h>

#include <linux/fb.h>

#include <sys/ioctl.h>

#include <wiringPi.h>

#include <wiringPiI2C.h>

#include <poll.h>

#include <dirent.h>

#include <linux/input.h>

#include <time.h>

### Data Structures

* struct **fbpixel**
* struct **lps25hData**
* struct **ht221sData**

### Macros

* #define **LPS25HI2CADDRESS**  0x5c
* #define **PRESS\_OUT\_XL**  0x28
* #define **PRESS\_OUT\_L**  0x29
* #define **PRESS\_OUT\_H**  0x2A
* #define **HTS221I2CADDRESS**  0x5F
* #define **HTS221DELAY**  25000
* #define **WHO\_AM\_I**  0x0F
* #define **CTRL\_REG1**  0x20
* #define **CTRL\_REG2**  0x21
* #define **T0\_OUT\_L**  0x3C
* #define **T0\_OUT\_H**  0x3D
* #define **T1\_OUT\_L**  0x3E
* #define **T1\_OUT\_H**  0x3F
* #define **T0\_degC\_x8**  0x32
* #define **T1\_degC\_x8**  0x33
* #define **T1\_T0\_MSB**  0x35
* #define **TEMP\_OUT\_L**  0x2A
* #define **TEMP\_OUT\_H**  0x2B
* #define **H0\_T0\_OUT\_L**  0x36
* #define **H0\_T0\_OUT\_H**  0x37
* #define **H1\_T0\_OUT\_L**  0x3A
* #define **H1\_T0\_OUT\_H**  0x3B
* #define **H0\_rH\_x2**  0x30
* #define **H1\_rH\_x2**  0x31
* #define **H\_T\_OUT\_L**  0x28
* #define **H\_T\_OUT\_H**  0x29
* #define **FILEPATH**  "/dev/fb1"
* #define **NUM\_WORDS**  64
* #define **FILESIZE**  (**NUM\_WORDS** \* sizeof(uint16\_t))
* #define **RGB565\_RED**  0xF800
* #define **RGB565\_GREEN**  0x07E0
* #define **RGB565\_BLUE**  0x001F

### Typedefs

* typedef struct **fbpixel** **fbpixel\_s**
* typedef struct **lps25hData** **lps25hData\_s**
* typedef struct **ht221sData** **ht221sData\_s**

### Data Structure Documentation

#### struct fbpixel

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| uint8\_t | blue |  |
| uint8\_t | green |  |
| uint8\_t | red |  |

#### struct lps25hData

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | pressure |  |
| double | temperature |  |

#### struct ht221sData

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | humidity |  |
| double | temperature |  |

### Macro Definition Documentation

#### #define CTRL\_REG1  0x20

#### #define CTRL\_REG2  0x21

#### #define FILEPATH  "/dev/fb1"

#### #define FILESIZE  (NUM\_WORDS \* sizeof(uint16\_t))

#### #define H0\_rH\_x2  0x30

#### #define H0\_T0\_OUT\_H  0x37

#### #define H0\_T0\_OUT\_L  0x36

#### #define H1\_rH\_x2  0x31

#### #define H1\_T0\_OUT\_H  0x3B

#### #define H1\_T0\_OUT\_L  0x3A

#### #define H\_T\_OUT\_H  0x29

#### #define H\_T\_OUT\_L  0x28

#### #define HTS221DELAY  25000

#### #define HTS221I2CADDRESS  0x5F

#### #define LPS25HI2CADDRESS  0x5c

RPi Sensehat constants, structures, function prototypes

##### Version:

**pisensehat.h** 2020-01-15

#### #define NUM\_WORDS  64

#### #define PRESS\_OUT\_H  0x2A

#### #define PRESS\_OUT\_L  0x29

#### #define PRESS\_OUT\_XL  0x28

#### #define RGB565\_BLUE  0x001F

#### #define RGB565\_GREEN  0x07E0

#### #define RGB565\_RED  0xF800

#### #define T0\_degC\_x8  0x32

#### #define T0\_OUT\_H  0x3D

#### #define T0\_OUT\_L  0x3C

#### #define T1\_degC\_x8  0x33

#### #define T1\_OUT\_H  0x3F

#### #define T1\_OUT\_L  0x3E

#### #define T1\_T0\_MSB  0x35

#### #define TEMP\_OUT\_H  0x2B

#### #define TEMP\_OUT\_L  0x2A

#### #define WHO\_AM\_I  0x0F

### Typedef Documentation

#### typedef struct fbpixel fbpixel\_s

#### typedef struct ht221sData ht221sData\_s

#### typedef struct lps25hData lps25hData\_s

# Index

alarm\_e

ghcontrol.h, 15

alarm\_s

ghcontrol.h, 15

alarmlimit\_s

ghcontrol.h, 15

alarmlimits, 13

alarmnames

ghcontrol.c, 11

ALARMNMSZ

ghcontrol.h, 13

alarms, 13

control\_s

ghcontrol.h, 15

controls, 12

CTRL\_REG1

pisensehat.h, 25

CTRL\_REG2

pisensehat.h, 25

fbfd

pisensehat.c, 22

fbpixel, 24

fbpixel\_s

pisensehat.h, 26

FILEPATH

pisensehat.h, 25

FILESIZE

pisensehat.h, 25

ghc.c

main, 1

ghc/ghc.c, 1

ghc/ghcontrol.c, 2

ghc/ghcontrol.h, 11

ghc/pisensehat.c, 15

ghc/pisensehat.h, 22

ghcontrol.c

alarmnames, 11

GhControllerInit, 2

GhDelay, 3

GhDisplayAlarms, 3

GhDisplayAll, 4

GhDisplayControls, 4

GhDisplayHeader, 5

GhDisplayReadings, 5

GhDisplaySetpoints, 5

GhGetControls, 6

GhGetHumidity, 6

GhGetPressure, 6

GhGetRandom, 7

GhGetReadings, 7

GhGetSetpoints, 7

GhGetTemperature, 7

GhLogData, 8

GhRetrieveSetpoints, 8

GhSaveSetpoints, 8

GhSetAlarmLimits, 9

GhSetAlarms, 9

GhSetControls, 10

GhSetSetpoints, 10

ghcontrol.h

alarm\_e, 15

alarm\_s, 15

alarmlimit\_s, 15

ALARMNMSZ, 13

control\_s, 15

GHUPDATE, 13

HBAR, 14

HHUMID, 15

HPRESS, 15

HTEMP, 15

HUMIDITY, 14

LHUMID, 15

LOWERAHUMID, 14

LOWERAPRESS, 14

LOWERATEMP, 14

LPRESS, 15

LSHUMID, 14

LSPRESS, 14

LSTEMP, 14

LTEMP, 15

NALARMS, 14

NOALARM, 15

NUMBERS, 14

NUMPTS, 14

OFF, 14

ON, 14

PBAR, 14

PRESSURE, 14

reading\_s, 15

SENSEHAT, 14

SENSORS, 14

setpoint\_s, 15

SHUMID, 14

SIMHUMIDITY, 14

SIMPRESSURE, 14

SIMTEMPERATURE, 14

SIMULATE, 14

STEMP, 14

TBAR, 14

TEMPERATURE, 14

UPPERAHUMID, 15

UPPERAPRESS, 15

UPPERATEMP, 15

USHUMID, 15

USPRESS, 15

USTEMP, 15

GhControllerInit

ghcontrol.c, 2

GhDelay

ghcontrol.c, 3

GhDisplayAlarms

ghcontrol.c, 3

GhDisplayAll

ghcontrol.c, 4

GhDisplayControls

ghcontrol.c, 4

GhDisplayHeader

ghcontrol.c, 5

GhDisplayReadings

ghcontrol.c, 5

GhDisplaySetpoints

ghcontrol.c, 5

GhGetControls

ghcontrol.c, 6

GhGetHumidity

ghcontrol.c, 6

GhGetPressure

ghcontrol.c, 6

GhGetRandom

ghcontrol.c, 7

GhGetReadings

ghcontrol.c, 7

GhGetSetpoints

ghcontrol.c, 7

GhGetTemperature

ghcontrol.c, 7

GhLogData

ghcontrol.c, 8

GhRetrieveSetpoints

ghcontrol.c, 8

GhSaveSetpoints

ghcontrol.c, 8

GhSetAlarmLimits

ghcontrol.c, 9

GhSetAlarms

ghcontrol.c, 9

GhSetControls

ghcontrol.c, 10

GhSetSetpoints

ghcontrol.c, 10

GHUPDATE

ghcontrol.h, 13

H\_T\_OUT\_H

pisensehat.h, 25

H\_T\_OUT\_L

pisensehat.h, 25

H0\_rH\_x2

pisensehat.h, 25

H0\_T0\_OUT\_H

pisensehat.h, 25

H0\_T0\_OUT\_L

pisensehat.h, 25

H1\_rH\_x2

pisensehat.h, 25

H1\_T0\_OUT\_H

pisensehat.h, 25

H1\_T0\_OUT\_L

pisensehat.h, 25

HBAR

ghcontrol.h, 14

HHUMID

ghcontrol.h, 15

HPRESS

ghcontrol.h, 15

ht221sData, 24

ht221sData\_s

pisensehat.h, 26

HTEMP

ghcontrol.h, 15

HTS221DELAY

pisensehat.h, 25

HTS221fd

pisensehat.c, 22

HTS221I2CADDRESS

pisensehat.h, 25

HUMIDITY

ghcontrol.h, 14

LHUMID

ghcontrol.h, 15

LOWERAHUMID

ghcontrol.h, 14

LOWERAPRESS

ghcontrol.h, 14

LOWERATEMP

ghcontrol.h, 14

LPRESS

ghcontrol.h, 15

lps25hData, 24

lps25hData\_s

pisensehat.h, 26

LPS25Hfd

pisensehat.c, 22

LPS25HI2CADDRESS

pisensehat.h, 25

LSHUMID

ghcontrol.h, 14

LSPRESS

ghcontrol.h, 14

LSTEMP

ghcontrol.h, 14

LTEMP

ghcontrol.h, 15

main

ghc.c, 1

map

pisensehat.c, 22

NALARMS

ghcontrol.h, 14

NOALARM

ghcontrol.h, 15

NUM\_WORDS

pisensehat.h, 26

NUMBERS

ghcontrol.h, 14

NUMPTS

ghcontrol.h, 14

OFF

ghcontrol.h, 14

ON

ghcontrol.h, 14

PBAR

ghcontrol.h, 14

pisensehat.c

fbfd, 22

HTS221fd, 22

LPS25Hfd, 22

map, 22

ShClearMatrix, 16

ShExit, 16

ShGetHT221SData, 17

ShGetLPS25HData, 18

ShGetPixel, 19

ShInit, 20

ShSetPixel, 21

ShSetVerticalBar, 21

pisensehat.h

CTRL\_REG1, 25

CTRL\_REG2, 25

fbpixel\_s, 26

FILEPATH, 25

FILESIZE, 25

H\_T\_OUT\_H, 25

H\_T\_OUT\_L, 25

H0\_rH\_x2, 25

H0\_T0\_OUT\_H, 25

H0\_T0\_OUT\_L, 25

H1\_rH\_x2, 25

H1\_T0\_OUT\_H, 25

H1\_T0\_OUT\_L, 25

ht221sData\_s, 26

HTS221DELAY, 25

HTS221I2CADDRESS, 25

lps25hData\_s, 26

LPS25HI2CADDRESS, 25

NUM\_WORDS, 26

PRESS\_OUT\_H, 26

PRESS\_OUT\_L, 26

PRESS\_OUT\_XL, 26

RGB565\_BLUE, 26

RGB565\_GREEN, 26

RGB565\_RED, 26

T0\_degC\_x8, 26

T0\_OUT\_H, 26

T0\_OUT\_L, 26

T1\_degC\_x8, 26

T1\_OUT\_H, 26

T1\_OUT\_L, 26

T1\_T0\_MSB, 26

TEMP\_OUT\_H, 26

TEMP\_OUT\_L, 26

WHO\_AM\_I, 26

PRESS\_OUT\_H

pisensehat.h, 26

PRESS\_OUT\_L

pisensehat.h, 26

PRESS\_OUT\_XL

pisensehat.h, 26

PRESSURE

ghcontrol.h, 14

reading\_s

ghcontrol.h, 15

readings, 12

RGB565\_BLUE

pisensehat.h, 26

RGB565\_GREEN

pisensehat.h, 26

RGB565\_RED

pisensehat.h, 26

SENSEHAT

ghcontrol.h, 14

SENSORS

ghcontrol.h, 14

setpoint\_s

ghcontrol.h, 15

setpoints, 12

ShClearMatrix

pisensehat.c, 16

ShExit

pisensehat.c, 16

ShGetHT221SData

pisensehat.c, 17

ShGetLPS25HData

pisensehat.c, 18

ShGetPixel

pisensehat.c, 19

ShInit

pisensehat.c, 20

ShSetPixel

pisensehat.c, 21

ShSetVerticalBar

pisensehat.c, 21

SHUMID

ghcontrol.h, 14

SIMHUMIDITY

ghcontrol.h, 14

SIMPRESSURE

ghcontrol.h, 14

SIMTEMPERATURE

ghcontrol.h, 14

SIMULATE

ghcontrol.h, 14

STEMP

ghcontrol.h, 14

T0\_degC\_x8

pisensehat.h, 26

T0\_OUT\_H

pisensehat.h, 26

T0\_OUT\_L

pisensehat.h, 26

T1\_degC\_x8

pisensehat.h, 26

T1\_OUT\_H

pisensehat.h, 26

T1\_OUT\_L

pisensehat.h, 26

T1\_T0\_MSB

pisensehat.h, 26

TBAR

ghcontrol.h, 14

TEMP\_OUT\_H

pisensehat.h, 26

TEMP\_OUT\_L

pisensehat.h, 26

TEMPERATURE

ghcontrol.h, 14

UPPERAHUMID

ghcontrol.h, 15

UPPERAPRESS

ghcontrol.h, 15

UPPERATEMP

ghcontrol.h, 15

USHUMID

ghcontrol.h, 15

USPRESS

ghcontrol.h, 15

USTEMP

ghcontrol.h, 15

WHO\_AM\_I

pisensehat.h, 26