TECH153 Greenhouse Controller

**Sarva Jeyapalan**

Student # N00674064

CENG-153-0NB

Version Lab10

8/11/2020 11:41:00 PM

**makefile File Documentation**

#makefile

ghc: ghc.o ghcontrol.o pisensehat.o

gcc -g -o ghc ghc.o ghcontrol.o pisensehat.o -lwiringPi -lpython2.7

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

gcc -g -c ghc.c

ghcontrol.o: ghcontrol.c ghcontrol.h

gcc -g -c ghcontrol.c

pisensehat.o: pisensehat.c pisensehat.h

gcc -g -c pisensehat.c

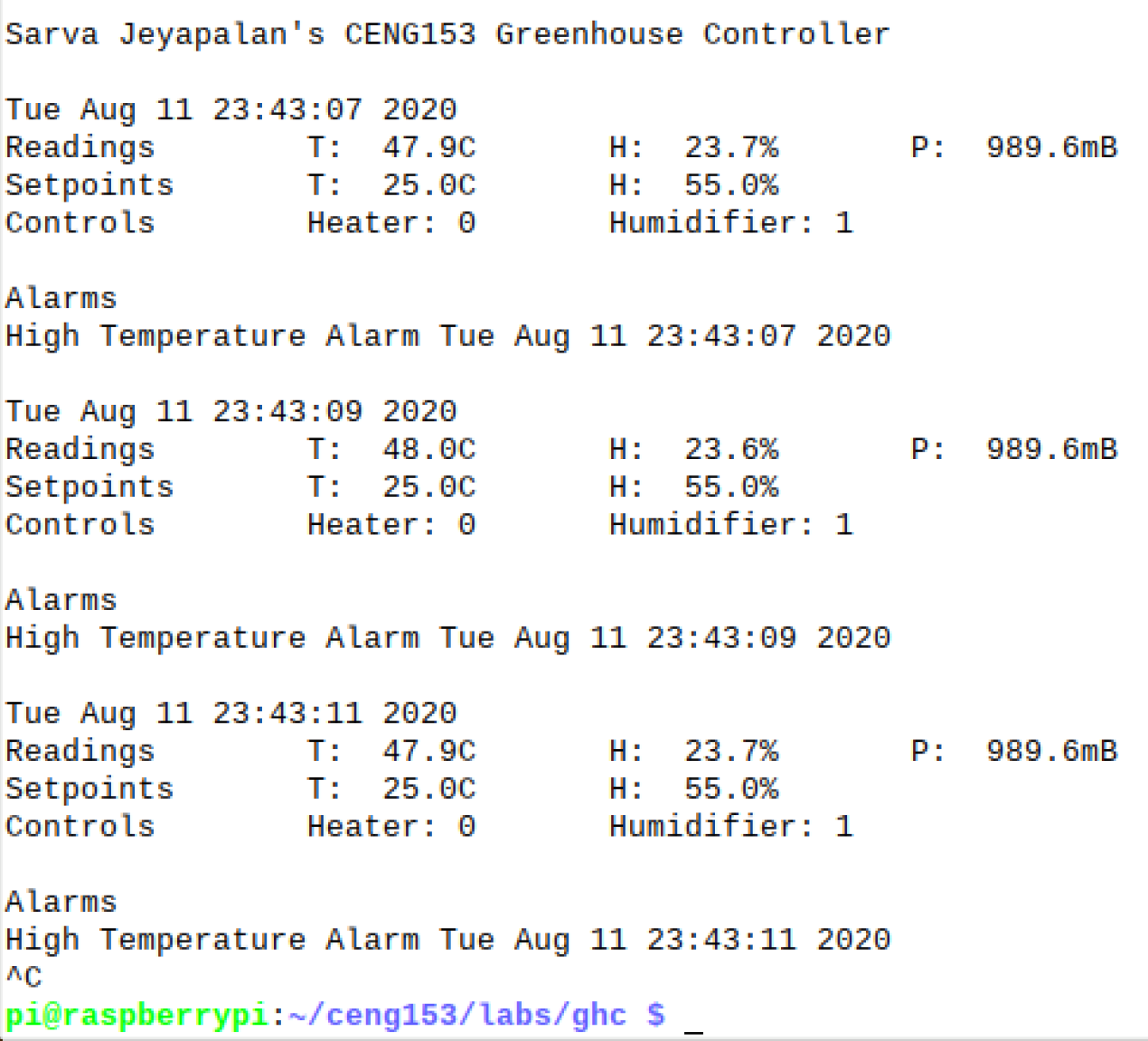
clean:

touch \*

rm \*.o

# Sample Output – Terminal

Output results when using physical SenseHat device



# Table of Contents

File Index 1

File Documentation 1

ghc.c 1

ghcontrol.c 2

ghcontrol.h 13

pisensehat.c 18

pisensehat.h 25

Index 29

# File Index

## File List

Here is a list of all files with brief descriptions:

**ghc.c**  1

**ghcontrol.c**  2

**ghcontrol.h**  13

**pisensehat.c**  18

**pisensehat.h**  25

# File Documentation

## ghc.c File Reference

#include "ghcontrol.h"

#include <stdlib.h>

### Functions

int **main** (void)

### Detailed Description

Defines entry point for the console application.

##### Author:

Paul Moggach

Kristian Medri

Sarva Jeyapalan

##### Version:

2020-08-11

### Function Documentation

#### int main (void )

Gh main function

##### Author:

Paul Moggach

Kristian Medri

Sarva Jeyapalan

##### Version:

2020-08-11

##### Parameters:

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

##### Returns:

exit status

21 {

22 int logged;

23 alarmlimit\_s alimits = {0};

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

25 control\_s ctrl = {0};

26 reading\_s creadings = {0.0};

27 setpoint\_s sets = {0};

28

29 GhControllerInit(); //Prints header to the screen

30 sets = GhSetTargets();

31 alimits = GhSetAlarmLimits();

32 while(1)

33 {

34 creadings = GhGetReadings();

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

36 ctrl = GhSetControls(sets,creadings);

37 GhSetAlarms(warn,alimits,creadings);

38 GhDisplayAll(creadings,sets);

39 GhDisplayReadings(creadings);

40 GhDisplayTargets(sets);

41 GhDisplayControls(ctrl);

42 GhDisplayAlarms(warn);

43 GhDelay(GHUPDATE);

44 }

45 return EXIT\_FAILURE;

46 }

## ghcontrol.c File Reference

#include "ghcontrol.h"

#include "pisensehat.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

### Functions

**alarmlimit\_s** **GhSetAlarmLimits** (void)

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

double **GhGetHumidity** (void)

double **GhGetPressure** (void)

double **GhGetTemperature** (void)

int **GhGetRandom** (int range)

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

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

**reading\_s** **GhGetReadings** (void)

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

**setpoint\_s** **GhSetTargets** (void)

void **GhControllerInit** (void)

void **GhDelay** (int milliseconds)

void **GhDisplayAlarms** (**alarm\_s** calarm[**NALARMS**])

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

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

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

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

void **GhDisplaySetpoints** (void)

void **GhDisplayTargets** (**setpoint\_s** spts)

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

### Variables

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

### Detailed Description

Gh control functions

##### Author:

Paul Moggach

Kristian Medri

Sarva Jeyapalan

##### Version:

2020-08-11

### Function Documentation

#### void GhControllerInit (void )

Controller Initialization

##### Version:

2020-07-23

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

void

260 {

261 #if SIMULATE

262 srand((unsigned) time(NULL));

263 #endif

264 #if SENSEHAT

265 ShInit();

266 #endif

267 GhDisplayHeader("Sarva Jeyapalan");

268 }

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

Delay clock interval for controller updates

##### Version:

2020-06-27

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *milliseconds* | integer delay value |

##### Returns:

void

277 {

278 long wait;

279 clock\_t now, start;

280

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

282 start = clock();

283 now = start;

284 while((now - start) < wait)

285 {

286 now = clock();

287 }

288 }

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

Displays environmental alarms

##### Version:

2020-08-11

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *calarm* | alarm\_s alarm code |
| *NALARMS* | number of alarms |

##### Returns:

void

298 {

299 int i;

300

301 fprintf(stdout,"\nAlarms\n");

302 for( i=0; i<= NALARMS; i++)

303 {

304 if(calarm[i].code != NOALARM)

305 {

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

307 }

308 }

309 }

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

Displays environmental data on 8x8 matrix

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *rd* | reading\_s reading value |
| *sd* | setpoint\_s setpoint value |

##### Returns:

void

319 {

320 int rv,sv,avh,avl;

321 fbpixel\_s pxc = {0};

322

323 ShClearMatrix();

324

325 // Set Temprature data bar and Setpoints pixel

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

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

328

329 pxc.red = 0x00;

330 pxc.green = 0xFF;

331 pxc.blue = 0x00;

332 ShSetVerticalBar(TBAR,pxc,rv);

333 pxc.red = 0xF0;

334 pxc.green = 0x0F;

335 pxc.blue = 0xF0;

336 ShSetPixel(TBAR,sv,pxc);

337

338 // Set Humidity data bar and Setpoints pixel

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

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

341

342 pxc.red = 0x00;

343 pxc.green = 0xFF;

344 pxc.blue = 0x00;

345 ShSetVerticalBar(HBAR,pxc,rv);

346 pxc.red = 0xF0;

347 pxc.green = 0x0F;

348 pxc.blue = 0xF0;

349 ShSetPixel(HBAR,sv,pxc);

350

351 // Set Pressure data bar and Setpoints pixel

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

353

354 pxc.red = 0x00;

355 pxc.green = 0xFF;

356 pxc.blue = 0x00;

357 ShSetVerticalBar(PBAR,pxc,rv);

358 }

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

Prints current On/Off status of Heater and Humidifier.

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *ctrl* | control\_s with control values |

##### Returns:

void

367 {

368 fprintf(stdout,"Controls\tHeater: %1.1ld\tHumidifier: %1.1ld\n",ctrl.heater,ctrl.humidifier);

369 }

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

Prints Gh Controller Title

##### Version:

2020-06-24

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *sname* | char with Operetor's name |

##### Returns:

void

378 {

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

380 }

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

Prints environmental readings and time.

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *rdata* | reading\_s of environmental readings data |

##### Returns:

void

389 {

390 fprintf(stdout,"\n%sReadings\tT: %5.1lfC\tH: %5.1lf%%\tP: %6.1lfmB",ctime(&rdata.rtime),rdata.temperature,rdata.humidity,rdata.pressure);

391 }

#### void GhDisplaySetpoints (void )

Not completed

##### Version:

2020-06-27

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

void

400 {

401 //Not Completed

402 }

#### void GhDisplayTargets (setpoint\_s *spts*)

Prints current target levels for temperature and humidity.

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *spts* | setpoint\_s environment values |

##### Returns:

void

411 {

412 fprintf(stdout, "\nSetpoints\tT: %5.1lfC\tH: %5.1lf\%\t\n", spts.temperature, spts.humidity);

413 }

#### double GhGetHumidity (void )

Returns Radomized Humidity Value

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

double value of humidity

79 {

80 #if SIMHUMIDITY

81 return GhGetRandom(USHUMID + 1);

82 #else

83 ht221sData\_s ct = {0};

84

85 ct = ShGetHT221SData();

86 return ct.humidity;

87 #endif

88 }

#### double GhGetPressure (void )

Returns Radomized Pressure Value

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

double value of pressure

97 {

98 #if SIMPRESSURE

99 return (GhGetRandom(USPRESS - LSPRESS + 1) + LSPRESS);

100 #else

101 lps25hData\_s ct = {0};

102

103 ct = ShGetLPS25HData();

104 return ct.pressure;

105 #endif

106 }

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

Returns Radomized Value

##### Version:

2020-06-27

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *range* | integer value of range |

##### Returns:

integer random value within range

133 {

134 return rand() %range;

135 }

#### reading\_s GhGetReadings (void )

Assigns environmental values to readings

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

reading\_s with environmental data

198 {

199 reading\_s now = {0};

200

201 now.rtime = time(NULL);

202 now.humidity = GhGetHumidity();

203 now.pressure = GhGetPressure();

204 now.temperature = GhGetTemperature();

205 return now;

206 }

#### double GhGetTemperature (void )

Returns Radomized Temperature Value

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

double value of temperature

115 {

116 #if SIMTEMPERATURE

117 return GhGetRandom(USTEMP - LSTEMP + 1) + LSTEMP;

118 #else

119 ht221sData\_s ct = {0};

120

121 ct = ShGetHT221SData();

122 return ct.temperature;

123 #endif

124 }

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

Data Logging

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *fname* | string file name pointer |
| *ghdata* | reading\_s with environment readings |

##### Returns:

integer 1

145 {

146 FILE \* fp;

147 char ltime[25];

148

149 fp = fopen(fname, "a");

150 if (fp == NULL)

151 {

152 return 0;

153 }

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

155

156 ltime[3] = ',';

157 ltime[7] = ',';

158 ltime[10] = ',';

159 ltime[19] = ',';

160

161 fprintf(fp,"\n%.24s,%5.1lf,%5.1lf,%6.1lf",ltime,ghdata.temperature,ghdata.humidity,ghdata.pressure);

162

163 fclose(fp);

164

165 return 1;

166 }

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

Retrieve new setpoints

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *fname* | char file name pointer |

##### Returns:

setpoint\_s update targets

215 {

216 setpoint\_s spts = {0.0};

217

218 FILE \* fp;

219

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

221 if (fp == NULL)

222 {

223 return spts;

224 }

225

226 fread(&spts, sizeof(spts), 1, fp);

227 fclose(fp);

228 return spts;

229 }

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

Saving new setpoint

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *fname* | string file name pointer |
| *spts* | setpoint\_s with environment setpoint |

##### Returns:

integer setpoint status

176 {

177 FILE \* fp;

178

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

180 if (fp == NULL)

181 {

182 return 0;

183 }

184

185 fwrite(&spts, sizeof(spts), 1, fp);

186

187 fclose(fp);

188 return 1;

189 }

#### alarmlimit\_s GhSetAlarmLimits (void )

Set upper and lower alarm limits of environment controllers

##### Version:

2020-08-07

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

alarmlimit\_s provides alarm limit values for the environment controller

27 {

28 alarmlimit\_s calarm = {0};

29

30 calarm.hight = UPPERATEMP;

31 calarm.lowt = LOWERATEMP;

32 calarm.highh = UPPERAHUMID;

33 calarm.lowh = LOWERATEMP;

34 calarm.highp = UPPERAPRESS;

35 calarm.lowp = LOWERAPRESS;

36

37 return calarm;

38 }

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

Compares readings with alarm limits and updates status

##### Version:

2020-08-07

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *calarm* | alarm\_s alarm code |
| *NALARMS* | number of alarms |
| *alarmpt* | alarmlimit\_s set values |
| *rdata* | reading\_s of environmental readings data |

##### Returns:

void

425 {

426 int i;

427

428 for(i=0; i<=NALARMS; i++)

429 {

430 calarm[i].code = NOALARM;

431 }

432

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

434 {

435 calarm[HHUMID].code = HHUMID;

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

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

438 }

439

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

441 {

442 calarm[HHUMID].code = HHUMID;

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

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

445 }

446

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

448 {

449 calarm[HPRESS].code = HPRESS;

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

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

452 }

453

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

455 {

456 calarm[HPRESS].code = HPRESS;

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

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

459 }

460

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

462 {

463 calarm[HTEMP].code = HTEMP;

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

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

466 }

467

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

469 {

470 calarm[HTEMP].code = HTEMP;

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

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

473 }

474 }

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

Sets On/Off condition of Hearter and/or Humidifier as per settings.

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

|  |  |
| --- | --- |
| *target* | setpoint\_s for target environment |
| *rdata* | reading\_s with environmental readings |

##### Returns:

control\_s to set operation mode of environment controllers

48 {

49 control\_s cset = {0};

50

51 if(rdata.humidity<SHUMID)

52 {

53 cset.humidifier = ON;

54 }

55 else

56 {

57 cset.humidifier = OFF;

58 }

59

60 if(rdata.temperature<STEMP)

61 {

62 cset.heater = ON;

63 }

64 else

65 {

66 cset.heater = OFF;

67 }

68

69 return cset;

70 }

#### setpoint\_s GhSetTargets (void )

Set targets for the environment

##### Version:

2020-08-03

##### Author:

Sarva Jeyapalan

##### Parameters:

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

##### Returns:

cpoints setpoint\_s with environmental targets

238 {

239 setpoint\_s cpoints = {0};

240

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

242

243 if(cpoints.temperature == 0)

244 {

245 cpoints.humidity = SHUMID;

246 cpoints.temperature = STEMP;

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

248 }

249

250 return cpoints;

251 }

### Variable Documentation

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

#### 

## ghcontrol.h File Reference

#include <time.h>

### Data Structures

struct **alarms**

struct **alarmlimits**

struct **controls**

struct **readings**

struct **setpoints**

### Macros

#define **ALARMNMSZ**  18

#define **GHUPDATE**  2000

#define **HUMIDITY**  1

#define **LOWERAHUMID**  25

#define **LOWERAPRESS**  985

#define **LOWERATEMP**  10

#define **LSHUMID**  0

#define **LSPRESS**  975

#define **LSTEMP**  -10

#define **NALARMS**  7

#define **OFF**  0

#define **ON**  1

#define **PRESSURE**  2

#define **SENSORS**  3

#define **SHUMID**  55.0

#define **SIMHUMIDITY**  0

#define **SIMPRESSURE**  0

#define **SIMTEMPERATURE**  0

#define **SIMULATE**  0

#define **STEMP**  25.0

#define **TEMPERATURE**  0

#define **UPPERAHUMID**  70

#define **UPPERAPRESS**  1016

#define **UPPERATEMP**  30

#define **USHUMID**  100

#define **USPRESS**  1016

#define **USTEMP**  50

#define **HBAR**  5

#define **NUMBARS**  8

#define **NUMPTS**  8.0

#define **PBAR**  3

#define **SENSEHAT**  1

#define **TBAR**  7

### Typedefs

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**

### Enumerations

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

### Detailed Description

Gh control constants, structures, function prototypes

##### Author:

Paul Moggach

Kristian Medri

Sarva Jeyapalan

##### Version:

2020-08-11

### Data Structure Documentation

#### struct alarms

##### Data Fields:

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

#### struct alarmlimits

##### Data Fields:

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

#### struct controls

##### Data Fields:

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

#### struct readings

##### Data Fields:

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

#### struct setpoints

##### Data Fields:

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

### Macro Definition Documentation

#### #define ALARMNMSZ  18

#### #define GHUPDATE  2000

#### #define HBAR  5

#### #define HUMIDITY  1

#### #define LOWERAHUMID  25

#### #define LOWERAPRESS  985

#### #define LOWERATEMP  10

#### #define LSHUMID  0

#### #define LSPRESS  975

#### #define LSTEMP  -10

#### #define NALARMS  7

#### #define NUMBARS  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 |  |

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

## 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)

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**

int **numReadings** =0

### Detailed Description

RPi Sensehat functions

##### Version:

2020-05-03

### Function Documentation

#### void ShClearMatrix (void )

Clears Sensehat 8X8 RGB LED display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-03

##### Parameters:

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

##### Returns:

void

108 {

109 #if EMULATOR

110 if (numReadings >=12){

111 numReadings=0;

112 printf("12 readings is about the limit for the emulator\n"

113 "the way that the current code is written since\n"

114 "it spawns too many threads and using Py\_Finalize\n"

115 "causes a decref segmentation fault. In addition,\n"

116 "it doesn't respond to Ctrl-C thus exiting gracefully.\n");

117 /\* Note that if you want to exit sooner you can stop the ghc process

118 by using Ctrl-Z, find the PID of ghc by using the command ps, and

119 use kill -9 PID# to end the process. \*/

120 exit(EXIT\_FAILURE);

121 }

122 else{

123 //printf("numReadings= %d\n",numReadings);

124 numReadings++;

125 }

126 PyRun\_SimpleString(

127 "from sense\_emu import SenseHat\n"

128 "sense=SenseHat()\n"

129 "sense.clear()\n"

130 );

131 #else

132 memset(map, 0, FILESIZE);

133 #endif

134 }

#### int ShExit (void )

Closes Down the Sensehat

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-01

##### Parameters:

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

##### Returns:

exit status

82 {

83 #if EMULATOR

84 Py\_Finalize();

85 #else

86 ShClearMatrix();

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

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

89 {

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

91 return EXIT\_FAILURE;

92 }

93 close(fbfd);

94 close(HTS221fd);

95 close(LPS25Hfd);

96 #endif

97 return EXIT\_SUCCESS;

98 }

#### ht221sData\_s ShGetHT221SData (void )

Gets HT221S Sensehat sensor data

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-03

##### Parameters:

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

##### Returns:

ht221sData\_s temperature and humidity data

286 {

287 ht221sData\_s rd = {0};

288 #if EMULATOR

289 PyRun\_SimpleString(

290 "from sense\_emu import SenseHat\n"

291 "#from time import time,ctime\n"

292 "#print('Today is '+ctime(time))\n"

293 "sense=SenseHat()\n"

294 "temp=sense.temp\n"

295 "humid=sense.humidity\n"

296 "#print(temp)\n"

297 "#print(humid)\n"

298 "f=open(\"tempfileforpython.txt\",\"w\")\n"

299 "f.write(repr(temp))\n"

300 "f.close()\n"

301 "f=open(\"humifileforpython.txt\",\"w\")\n"

302 "f.write(repr(humid))\n"

303 "f.close()\n"

304 );

305 double reading=0;

306 FILE \*fp;

307 fp=fopen("tempfileforpython.txt","r");

308 fscanf(fp, "%lf", &reading);

309 fclose(fp);

310 rd.temperature = reading;

311 //fprintf(stdout, "%lf\n", reading);

312 fp=fopen("humifileforpython.txt","r");

313 fscanf(fp, "%lf", &reading);

314 fclose(fp);

315 //fprintf(stdout, "%lf\n", reading);

316 rd.humidity = reading;

317 #else

318 int status;

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

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

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

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

323 double T0\_DegC,T1\_DegC;

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

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

326 int16\_t T\_OUT;

327 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;

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

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

330

331 // Power down the device (clean start)

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

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

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

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

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

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

338

339 // Wait until the measurement is completed

340 do

341 {

342 usleep(HTS221DELAY); // 25 ms

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

344 }

345 while (status != 0);

346

347 // Read calibration temperature LSB (ADC) data

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

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

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

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

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

353

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

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

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

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

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

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

360

361 // Read calibration temperature (°C) data

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

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

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

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

366

367 // Read relative humidity (% rH) data

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

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

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

371

372 // make 16 bit values (bit shift)

373 // (temperature calibration x-values)

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

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

376

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

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

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

380

381 // Calculate calibration values

382 // (temperature calibration y-values)

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

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

385

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

387 // calibration straight line graphs for temperature and humidity

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

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

390

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

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

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

394

395 // make 16 bit value

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

397

398 // make 16 bit values (bit shift)

399 // (humidity calibration x-values)

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

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

402

403 // Humidity calibration values

404 // (humidity calibration y-values)

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

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

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

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

409

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

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

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

413

414 // make 16 bit value

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

416

417 // Power down the device

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

419

420 // Calculate and return ambient temperature

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

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

423 #endif

424 return rd;

425 }

#### lps25hData\_s ShGetLPS25HData (void )

Gets LPS25H Sensehat sensor information

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-01

##### Parameters:

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

##### Returns:

lps25hData\_s pressure and temperature data

210 {

211 lps25hData\_s rd = {0};

212 #if EMULATOR

213 PyRun\_SimpleString(

214 "from sense\_emu import SenseHat\n"

215 "sense=SenseHat()\n"

216 "temp=sense.pressure\n"

217 "f=open(\"tempfileforpython.txt\",\"w\")\n"

218 "f.write(repr(temp))\n"

219 "f.close()\n"

220 );

221 double reading=0;

222 FILE \*fp;

223 fp=fopen("tempfileforpython.txt","r");

224 fscanf(fp, "%lf", &reading);

225 fclose(fp);

226 rd.pressure = reading;

227 rd.temperature = 5; //placeholder, use the temperature from the ht221s

228 #else

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

230 int16\_t temp\_out = 0;

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

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

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

234 int32\_t press\_out = 0;

235 uint8\_t status = 0;

236

237 // Power down the device (clean start)

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

239

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

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

242

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

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

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

246

247 // Wait until the measurement is completed

248 do

249 {

250 usleep(HTS221DELAY); // 25 ms

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

252 }

253 while (status != 0);

254

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

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

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

258

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

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

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

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

263

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

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

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

267

268 /\* calculate output values \*/

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

270 rd.pressure = press\_out / 4096.0;

271

272 // Power down the device

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

274 #endif

275 return rd;

276 }

#### int ShInit (void )

Initialize Sensehat

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-01

##### Parameters:

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

##### Returns:

exit status

22 {

23 #if EMULATOR

24 Py\_Initialize();

25 #else

26 wiringPiSetup();

27 struct fb\_fix\_screeninfo fix\_info;

28

29 // Frame Buffer Initialization for 8X8 LED Matrix

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

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

32 if (fbfd == -1)

33 {

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

35 exit(EXIT\_FAILURE);

36 }

37

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

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

40 {

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

42 close(fbfd);

43 exit(EXIT\_FAILURE);

44 }

45

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

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

48 {

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

50 close(fbfd);

51 exit(EXIT\_FAILURE);

52 }

53

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

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

56 if (map == MAP\_FAILED)

57 {

58 close(fbfd);

59 perror("Error mmapping the file");

60 exit(EXIT\_FAILURE);

61 }

62

63 // Sensor Initialization

64 HTS221fd = wiringPiI2CSetup(HTS221I2CADDRESS);

65 LPS25Hfd = wiringPiI2CSetup(LPS25HI2CADDRESS);

66

67 // Power down the device (clean start)

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

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

70 #endif

71 return EXIT\_SUCCESS;

72 }

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

Sets a pixel on the Sensehat display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-01

##### Parameters:

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

##### Returns:

uint8\_t exit status

146 {

147 #if EMULATOR

148 char ltime [120];

149 sprintf(ltime,

150 "from sense\_emu import SenseHat\n"

151 "sense=SenseHat()\n"

152 "sense.set\_pixel(%d,%d,%d,%d,%d)\n"

153 ,x,y,px.red,px.green,px.blue);

154 PyRun\_SimpleString(ltime);

155 return EXIT\_SUCCESS;

156 #else

157 int i;

158

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

160 {

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

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

163 return EXIT\_SUCCESS;

164 }

165 #endif

166 return EXIT\_FAILURE;

167 }

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

Sets a vertical bar on the Sensehat display

##### Author:

Paul Moggach

Kristian Medri

##### Version:

2020-05-01

##### Parameters:

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

##### Returns:

exit status

179 {

180 int i;

181 if (value>7){

182 value=7;

183 }

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

185 {

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

187 {

188 ShSetPixel(bar,i,px);

189 }

190 px.red = 0x00;

191 px.green = 0x00;

192 px.blue = 0x00;

193 for(i=value+1; i< 8;i++)

194 {

195 ShSetPixel(bar,i,px);

196 }

197 return EXIT\_SUCCESS;

198 }

199 return EXIT\_FAILURE;

200 }

### Variable Documentation

#### int fbfd[static]

#### int HTS221fd[static]

#### int LPS25Hfd[static]

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

#### int numReadings =0

#### 

## 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 <poll.h>

#include <dirent.h>

#include <linux/input.h>

#include <time.h>

#include <wiringPi.h>

#include <wiringPiI2C.h>

### Data Structures

struct **fbpixel**

struct **lps25hData**

struct **ht221sData**

### Macros

#define **EMULATOR**  0

#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**

### Detailed Description

RPi Sensehat constants, structures, function prototypes

##### Version:

2020-05-03

### 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 EMULATOR  0

#### #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

#### #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, 17

alarm\_s

ghcontrol.h, 17

alarmlimit\_s

ghcontrol.h, 17

alarmlimits, 14

alarmnames

ghcontrol.c, 13

ALARMNMSZ

ghcontrol.h, 16

alarms, 14

control\_s

ghcontrol.h, 17

controls, 14

CTRL\_REG1

pisensehat.h, 28

CTRL\_REG2

pisensehat.h, 28

EMULATOR

pisensehat.h, 28

fbfd

pisensehat.c, 25

fbpixel, 27

fbpixel\_s

pisensehat.h, 29

FILEPATH

pisensehat.h, 28

FILESIZE

pisensehat.h, 28

ghc.c, 1

main, 1

ghcontrol.c, 2

alarmnames, 13

GhControllerInit, 3

GhDelay, 3

GhDisplayAlarms, 4

GhDisplayAll, 4

GhDisplayControls, 5

GhDisplayHeader, 5

GhDisplayReadings, 6

GhDisplaySetpoints, 6

GhDisplayTargets, 6

GhGetHumidity, 7

GhGetPressure, 7

GhGetRandom, 7

GhGetReadings, 8

GhGetTemperature, 8

GhLogData, 9

GhRetrieveSetpoints, 9

GhSaveSetpoints, 10

GhSetAlarmLimits, 10

GhSetAlarms, 10

GhSetControls, 11

GhSetTargets, 12

ghcontrol.h, 13

alarm\_e, 17

alarm\_s, 17

alarmlimit\_s, 17

ALARMNMSZ, 16

control\_s, 17

GHUPDATE, 16

HBAR, 16

HHUMID, 17

HPRESS, 17

HTEMP, 17

HUMIDITY, 16

LHUMID, 17

LOWERAHUMID, 16

LOWERAPRESS, 16

LOWERATEMP, 16

LPRESS, 17

LSHUMID, 16

LSPRESS, 16

LSTEMP, 16

LTEMP, 17

NALARMS, 16

NOALARM, 17

NUMBARS, 16

NUMPTS, 16

OFF, 16

ON, 16

PBAR, 16

PRESSURE, 16

reading\_s, 17

SENSEHAT, 16

SENSORS, 16

setpoint\_s, 17

SHUMID, 16

SIMHUMIDITY, 16

SIMPRESSURE, 16

SIMTEMPERATURE, 16

SIMULATE, 17

STEMP, 17

TBAR, 17

TEMPERATURE, 17

UPPERAHUMID, 17

UPPERAPRESS, 17

UPPERATEMP, 17

USHUMID, 17

USPRESS, 17

USTEMP, 17

GhControllerInit

ghcontrol.c, 3

GhDelay

ghcontrol.c, 3

GhDisplayAlarms

ghcontrol.c, 4

GhDisplayAll

ghcontrol.c, 4

GhDisplayControls

ghcontrol.c, 5

GhDisplayHeader

ghcontrol.c, 5

GhDisplayReadings

ghcontrol.c, 6

GhDisplaySetpoints

ghcontrol.c, 6

GhDisplayTargets

ghcontrol.c, 6

GhGetHumidity

ghcontrol.c, 7

GhGetPressure

ghcontrol.c, 7

GhGetRandom

ghcontrol.c, 7

GhGetReadings

ghcontrol.c, 8

GhGetTemperature

ghcontrol.c, 8

GhLogData

ghcontrol.c, 9

GhRetrieveSetpoints

ghcontrol.c, 9

GhSaveSetpoints

ghcontrol.c, 10

GhSetAlarmLimits

ghcontrol.c, 10

GhSetAlarms

ghcontrol.c, 10

GhSetControls

ghcontrol.c, 11

GhSetTargets

ghcontrol.c, 12

GHUPDATE

ghcontrol.h, 16

H\_T\_OUT\_H

pisensehat.h, 28

H\_T\_OUT\_L

pisensehat.h, 28

H0\_rH\_x2

pisensehat.h, 28

H0\_T0\_OUT\_H

pisensehat.h, 28

H0\_T0\_OUT\_L

pisensehat.h, 28

H1\_rH\_x2

pisensehat.h, 28

H1\_T0\_OUT\_H

pisensehat.h, 28

H1\_T0\_OUT\_L

pisensehat.h, 28

HBAR

ghcontrol.h, 16

HHUMID

ghcontrol.h, 17

HPRESS

ghcontrol.h, 17

ht221sData, 27

ht221sData\_s

pisensehat.h, 29

HTEMP

ghcontrol.h, 17

HTS221DELAY

pisensehat.h, 28

HTS221fd

pisensehat.c, 25

HTS221I2CADDRESS

pisensehat.h, 28

HUMIDITY

ghcontrol.h, 16

LHUMID

ghcontrol.h, 17

LOWERAHUMID

ghcontrol.h, 16

LOWERAPRESS

ghcontrol.h, 16

LOWERATEMP

ghcontrol.h, 16

LPRESS

ghcontrol.h, 17

lps25hData, 27

lps25hData\_s

pisensehat.h, 29

LPS25Hfd

pisensehat.c, 25

LPS25HI2CADDRESS

pisensehat.h, 28

LSHUMID

ghcontrol.h, 16

LSPRESS

ghcontrol.h, 16

LSTEMP

ghcontrol.h, 16

LTEMP

ghcontrol.h, 17

main

ghc.c, 1

map

pisensehat.c, 25

NALARMS

ghcontrol.h, 16

NOALARM

ghcontrol.h, 17

NUM\_WORDS

pisensehat.h, 28

NUMBARS

ghcontrol.h, 16

NUMPTS

ghcontrol.h, 16

numReadings

pisensehat.c, 25

OFF

ghcontrol.h, 16

ON

ghcontrol.h, 16

PBAR

ghcontrol.h, 16

pisensehat.c, 18

fbfd, 25

HTS221fd, 25

LPS25Hfd, 25

map, 25

numReadings, 25

ShClearMatrix, 18

ShExit, 19

ShGetHT221SData, 19

ShGetLPS25HData, 22

ShInit, 23

ShSetPixel, 24

ShSetVerticalBar, 24

pisensehat.h, 25

CTRL\_REG1, 28

CTRL\_REG2, 28

EMULATOR, 28

fbpixel\_s, 29

FILEPATH, 28

FILESIZE, 28

H\_T\_OUT\_H, 28

H\_T\_OUT\_L, 28

H0\_rH\_x2, 28

H0\_T0\_OUT\_H, 28

H0\_T0\_OUT\_L, 28

H1\_rH\_x2, 28

H1\_T0\_OUT\_H, 28

H1\_T0\_OUT\_L, 28

ht221sData\_s, 29

HTS221DELAY, 28

HTS221I2CADDRESS, 28

lps25hData\_s, 29

LPS25HI2CADDRESS, 28

NUM\_WORDS, 28

PRESS\_OUT\_H, 28

PRESS\_OUT\_L, 28

PRESS\_OUT\_XL, 28

RGB565\_BLUE, 28

RGB565\_GREEN, 28

RGB565\_RED, 28

T0\_degC\_x8, 29

T0\_OUT\_H, 29

T0\_OUT\_L, 29

T1\_degC\_x8, 29

T1\_OUT\_H, 29

T1\_OUT\_L, 29

T1\_T0\_MSB, 29

TEMP\_OUT\_H, 29

TEMP\_OUT\_L, 29

WHO\_AM\_I, 29

PRESS\_OUT\_H

pisensehat.h, 28

PRESS\_OUT\_L

pisensehat.h, 28

PRESS\_OUT\_XL

pisensehat.h, 28

PRESSURE

ghcontrol.h, 16

reading\_s

ghcontrol.h, 17

readings, 15

RGB565\_BLUE

pisensehat.h, 28

RGB565\_GREEN

pisensehat.h, 28

RGB565\_RED

pisensehat.h, 28

SENSEHAT

ghcontrol.h, 16

SENSORS

ghcontrol.h, 16

setpoint\_s

ghcontrol.h, 17

setpoints, 15

ShClearMatrix

pisensehat.c, 18

ShExit

pisensehat.c, 19

ShGetHT221SData

pisensehat.c, 19

ShGetLPS25HData

pisensehat.c, 22

ShInit

pisensehat.c, 23

ShSetPixel

pisensehat.c, 24

ShSetVerticalBar

pisensehat.c, 24

SHUMID

ghcontrol.h, 16

SIMHUMIDITY

ghcontrol.h, 16

SIMPRESSURE

ghcontrol.h, 16

SIMTEMPERATURE

ghcontrol.h, 16

SIMULATE

ghcontrol.h, 17

STEMP

ghcontrol.h, 17

T0\_degC\_x8

pisensehat.h, 29

T0\_OUT\_H

pisensehat.h, 29

T0\_OUT\_L

pisensehat.h, 29

T1\_degC\_x8

pisensehat.h, 29

T1\_OUT\_H

pisensehat.h, 29

T1\_OUT\_L

pisensehat.h, 29

T1\_T0\_MSB

pisensehat.h, 29

TBAR

ghcontrol.h, 17

TEMP\_OUT\_H

pisensehat.h, 29

TEMP\_OUT\_L

pisensehat.h, 29

TEMPERATURE

ghcontrol.h, 17

UPPERAHUMID

ghcontrol.h, 17

UPPERAPRESS

ghcontrol.h, 17

UPPERATEMP

ghcontrol.h, 17

USHUMID

ghcontrol.h, 17

USPRESS

ghcontrol.h, 17

USTEMP

ghcontrol.h, 17

WHO\_AM\_I

pisensehat.h, 29