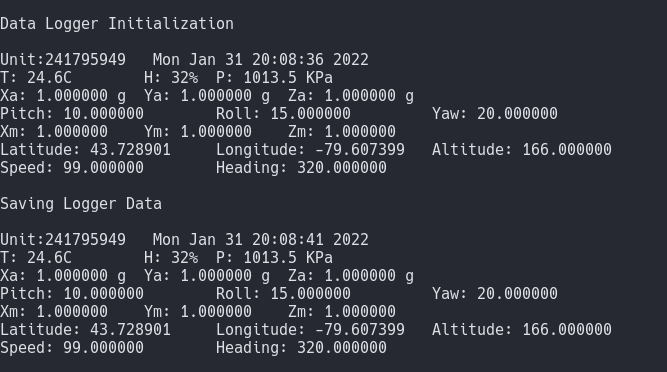
CENG252 Vehicle Data Logger

Robert R Miller

Version Lab03

DocInformation:Created (fixed)

Contents of Makefile:



vdl: vdl.o logger.o serial.o nmea.o dlgps.o sensehat.o

g++ -g -o vdl vdl.o logger.o serial.o nmea.o dlgps.o sensehat.o-lm -IRIMULib

vdl.o: vdl.cpp vdl.h logger.h serial.h nmea.h dlgps.h

g++ -g -c vdl.cpp

logger.o: logger.cpp logger.h serial.h nmea.h dlgps.h

g++ -g -c logger.cpp

serial.o: serial.cpp serial.h

g++ -g -c serial.cpp

dlgps.o: dlgps.cpp dlgps.h

g++ -g -c dlgps.cpp

nmea.o: nmea.cpp nmea.h

g++ -g -c nmea.cpp

sensehat.o: sensehat.cpp sensehat.h

g++ -g -c sensehat.cpp

clean:

touch \*

rm \*.o

Output of ‘vdl’:

# Table of Contents

[Data Structure Index 1](#1%19Data_Structure_Index%19C)

[File Index 1](#1%19File_Index%19C)

[Data Structure Documentation 1](#1%19Data_Structure_Documentation%19C)

[SenseHat 1](#1%19SenseHat%19C)

[File Documentation 18](#1%19File_Documentation%19C)

[dlgps.cpp 18](#1%19dlgps.cpp%19C)

[dlgps.h 22](#1%19dlgps.h%19C)

[font.h 23](#1%19font.h%19C)

[logger.cpp 24](#1%19logger.cpp%19C)

[logger.h 27](#1%19logger.h%19C)

[nmea.cpp 31](#1%19nmea.cpp%19C)

[nmea.h 34](#1%19nmea.h%19C)

[sensehat.cpp 36](#1%19sensehat.cpp%19C)

[sensehat.h 40](#1%19sensehat.h%19C)

[serial.cpp 44](#1%19serial.cpp%19C)

[serial.h 47](#1%19serial.h%19C)

[vdl.cpp 47](#1%19vdl.cpp%19C)

[vdl.h 48](#1%19vdl.h%19C)

[Index 48](#1%19Index%19C)

# Data Structure Index

## Data Structures

Here are the data structures with brief descriptions:

**SenseHat**  1

# File Index

## File List

Here is a list of all files with brief descriptions:

**dlgps.cpp (Data logger gps Functions )** 18

**dlgps.h (Constants, structures, function prototypes for gps functions )** 22

**font.h**  23

**logger.cpp (Vehicle Data logger functions )** 23

**logger.h**  27

**nmea.cpp**  29

**nmea.h (Constants, structures, function prototypes for NMEA functions )** 33

**sensehat.cpp**  35

**sensehat.h**  38

**serial.cpp (Serial Functions )** 42

**serial.h (Constants, structures, function prototypes for serial functions )** 45

**vdl.cpp (Vehicle Data Logger main function )** 45

**vdl.h**  46

# Data Structure Documentation

## SenseHat Class Reference

#include <sensehat.h>

### Public Member Functions

**SenseHat** (void)

***SenseHat::SenseHat****.*

**~SenseHat** (void)

***SenseHat::~SenseHat****.*

**SenseHat** & **operator<<** (**SenseHat** &(\*)(**SenseHat** &))

**SenseHat** & **operator<<** (const std::string &)

**SenseHat** & **operator<<** (const int)

**SenseHat** & **operator<<** (const double)

**SenseHat** & **operator<<** (const char)

**SenseHat** & **operator<<** (const char \*)

**SenseHat** & **operator<<** (const bool)

void **ViewMessage** (const std::string message, int vitesseDefilement=100, uint16\_t colorText=**BLUE**, uint16\_t colorBackground=**BLACK**)

void **ViewLetter** (char lettre, uint16\_t colorText=**BLUE**, uint16\_t colorBackground=**BLACK**)

*SenseHat::AfficherLettre.*

void **LightPixel** (int row, int column, uint16\_t **color**)

***SenseHat::LightPixel****.*

uint16\_t **GetPixel** (int row, int column)

***SenseHat::GetPixel****.*

void **ViewPattern** (uint16\_t pattern[][8])

***SenseHat::ViewPattern****.*

void **RotatePattern** (int **rotation**)

***SenseHat::RotatePattern****.*

char **ScannerJoystick** (void)

char **ScanJoystick** (void)

***SenseHat::ScannerJoystick****.*

**COLOR\_SENSEHAT** **ConvertRGB565** (uint8\_t red, uint8\_t green, uint8\_t blue)

***SenseHat::ConvertRGB565****.*

**COLOR\_SENSEHAT** **ConvertRGB565** (uint8\_t **color**[])

*surcharge de SenseHat::ConvertirRGB565*

**COLOR\_SENSEHAT** **ConvertRGB565** (std::string **color**)

*surcharge de SenseHat::ConvertirRGB565*

void **WipeScreen** (uint16\_t **color**=**BLACK**)

***SenseHat::WipeScreen****.*

float **GetTemperature** (void)

***SenseHat::GetTemperature****.*

float **correctTemperature** (float senseHatTemp, float cpuTemp)

float **getRawTemperature** (void)

***SenseHat::getRawTemperature****.*

float **getCpuTemperature** (void)

***SenseHat::getCpuTemperature****.*

float **GetPressure** (void)

***SenseHat::GetPressure****.*

float **GetHumidity** (void)

***SenseHat::GetHumidity****.*

void **GetOrientation** (float &pitch, float &roll, float &yaw)

***SenseHat::GetOrientation****.*

void **GetAcceleration** (float &x, float &y, float &z)

***SenseHat::GetAcceleration****.*

void **GetMagnetism** (float &x, float &y, float &z)

***SenseHat::GetMagnetism****.*

void **GetSphericalMagnetism** (float &ro, float &teta, float &delta)

***SenseHat::GetSphericalMagnetism****.*

void **Version** (void)

***SenseHat::Version****.*

void **Flush** (void)

void **SetColor** (uint16\_t)

void **SetRotation** (uint16\_t)

### Private Member Functions

void **InitializeLeds** (void)

***SenseHat::InitializeLeds****.*

void **InitializeJoystick** (void)

***SenseHat::InitializeJoystick****.*

void **InitializePressure** (void)

*SenseHat::InitialiserPression.*

void **InitializeHumidity** (void)

*SenseHat::Initialiserhumidity @detail initialise le capteur d'humidité*

void **InitializeOrientation** (void)

void **InitializeAcceleration** (void)

void **ConvertCharacterToPattern** (char c, uint16\_t image[8][8], uint16\_t colorText, uint16\_t colorBackground)

*SenseHat::ConvertCharactereToPattern.*

bool **EmptyColumn** (int numcolumn, uint16\_t image[8][8], uint16\_t colorBackground)

void **ImageContainment** (int numcolumn, uint16\_t image[][8][8], int taille)

### Private Attributes

struct **fb\_t** \* **fb**

int **joystick**

RTIMUSettings \* **settings**

RTIMU \* **imu**

RTPressure \* **pressure**

RTHumidity \* **humidity**

std::string **buffer**

uint16\_t **color**

int **rotation**

### Constructor & Destructor Documentation

#### SenseHat::SenseHat (void )

Constructeur de la classe, initialise les attributs par défaut imu, leds, Joystick, buffer.

146 {

147 #if SENSEHAT\_EMULATOR

148          Py\_Initialize();

149 #else

150      int tries;

151      settings = new RTIMUSettings("RTIMULib");

152

153      tries = 0;

154      while(true)

155      {

156          imu = RTIMU::createIMU(settings);

157          if ((imu == NULL) || (imu->IMUType() == RTIMU\_TYPE\_NULL))

158          {

159              tries++;

160              usleep(100);

161          }

162          else

163          {

164              break;

165          }

166          if (tries > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

167          {

168              exit (EXIT\_FAILURE);

169          }

170      }

171

172      imu->IMUInit();

173      imu->setSlerpPower(0.02);

174      imu->setGyroEnable(true);

175      imu->setAccelEnable(true);

176      imu->setCompassEnable(true);

177      InitializeLeds();

178      InitializeJoystick();

179      InitializeHumidity();

180      InitializePressure();

181 #endif

182      buffer=" ";

183      color=BLUE;

184      rotation = 0;

185 }

#### SenseHat::~SenseHat (void )

Destructeur de la classe

192 {

193 #if SENSEHAT\_EMULATOR

194          Py\_Finalize();

195 #else

196          delete settings;

197 #endif

198 }

### Member Function Documentation

#### SenseHat & SenseHat::operator<< (SenseHat & \*)(SenseHat &)

#### SenseHat & SenseHat::operator<< (const std::string &    *message*)

1005 {

1006          buffer += message;

1007          return \*this;

1008 }

#### SenseHat & SenseHat::operator<< (const int    *valeur*)

1011 {

1012          buffer += std::to\_string(valeur);

1013          return \*this;

1014 }

#### SenseHat & SenseHat::operator<< (const double    *valeur*)

1017 {

1018          std::stringstream ss;

1019          ss << std::fixed << std::setprecision(2) << valeur;

1020          buffer += ss.str();

1021          return \*this;

1022 }

#### SenseHat & SenseHat::operator<< (const char    *caractere*)

1025 {

1026          buffer += std::string(1, caractere);

1027          return \*this;

1028 }

#### SenseHat & SenseHat::operator<< (const char \*    *message*)

1031 {

1032          buffer += std::string(message);

1033          return \*this;

1034 }

#### SenseHat & SenseHat::operator<< (const bool    *valeur*)

1037 {

1038          buffer +=    std::to\_string(valeur);

1039          return \*this;

1040 }

#### void SenseHat::ViewMessage (const std::string    *message*, int    *vitesseDefilement* = 100, uint16\_t    *colorText* = BLUE, uint16\_t    *colorBackground* = BLACK)

928 {

929          int taille=message.length();

930          uint16\_t chaine[taille][8][8]; /\* Le tableau de pattern (image/caractère) à afficher \*/

931          int i=0,j=0,k=0,l=0,nombreDecolumnVide=0;

932          int isuivant=0,ksuivant=0,nombreDecolumns=0;

933

934          /\* Convertion de tout le message en tableau de patterns

935           \* format caractère : 1 column vide + 5 columns réellement utilisées

936           \* + 2 columns vides \*/

937          for( i=0,j=0; i<taille; i++,j++)

938          {

939                  if(message[i]==195)    // les lettres accentuées sont codées sur deux octets    (195 167 pour ç)

940                  {

941                          i++;

942                          k++;

943                  }

944          // row suivante à décommenter pour obtenir le code des caractères UTF8

945          // std::cout << "code : " << (int)message[i] << std::endl;

946                  ConvertCharacterToPattern(message[i],chaine[j],colorText,colorBackground);

947          }

948          taille = taille - k;

949          nombreDecolumns=(taille)\*8-2;

950          k = 0;

951          // Parcours de toutes les columns de tous les patterns qui compose

952          // la chaine de caractères à afficher pour supprimer les columns vides sauf celle

953          // qui sépare les patterns (caractères). + gestion du caractère espace.

954          for( l=0;l<nombreDecolumns;l++)

955          {

956                  i=l/8;

957                  k=l%8;

958

959 //            if(columnVide(k,chaine[i],colorBackground)) // Une column Vide avant chaque caractère à ne pas supprimer

960                  if(EmptyColumn(k,chaine[i],colorBackground)) // Une column Vide avant chaque caractère à ne pas supprimer

961                  {

962                          isuivant=(++l)/8;

963                          ksuivant=(l)%8;

964                          nombreDecolumnVide=1;

965

966                          // compter les columns vide après la première afin de les supprimer

967                          // si plus de 4 c'est le caractère espace que l'on doit garder

968 //                    while(columnVide(ksuivant,chaine[isuivant],colorBackground) && nombreDecolumnVide++ < 6)

969                          while(EmptyColumn(ksuivant,chaine[isuivant],colorBackground) && nombreDecolumnVide++ < 6)

970                          {

971                                  ImageContainment(l,chaine,taille);

972                                  nombreDecolumns--;

973                          }

974

975                  }

976          }

977

978          // Parcours de toutes les columns de tous les patterns qui composent

979          // la chaine de caractères à afficher (sans les columns vides superflues).

980          for( l=0; l<nombreDecolumns; l++)

981          {

982                  // Decalage des columns vers la gauche sur l'image Numero 0 (celle qu'on affiche sur la matrice de LED

983                  for( i=0;i<taille;i++)

984                  {

985                          // Cas Normal, les columns sont sur le même pattern

986                          for(j=0;j<8;j++)

987                          {

988                                  for(k=0;k<7;k++)

989                                          chaine[i][j][k]=chaine[i][j][k+1];

990

991                          }

992                          // Cas où l'on doit changer de pattern

993                          for(j=0;j<8;j++)

994                          {

995                                  chaine[i][j][7]=chaine[i+1][j][0];

996                          }

997                  }

998                  usleep(1000\*vitesseDefilement);

999                  ViewPattern(chaine[0]);

1000          }

1001

1002 }

#### void SenseHat::ViewLetter (char    *lettre*, uint16\_t    *colorText* = BLUE, uint16\_t    *colorBackground* = BLACK)

##### Parameters

|  |  |
| --- | --- |
| *lettre* | char |
| *colorText* | uint16\_t une color au format 565 |
| *colorBackground* | uint16\_t une color au format 565 |

affiche un caractère (lettre ponctuation signe) sur l'afficheur

236 {

237          uint16\_t chr[8][8];

238

239          ConvertCharacterToPattern(lettre,chr,colorText,colorBackground);

240          ViewPattern(chr);

241 }

#### void SenseHat::LightPixel (int    *row*, int    *column*, uint16\_t    *color*)

##### Parameters

|  |  |
| --- | --- |
| *row* | integer |
| *column* | integer |
| *color* | uint16\_t une color au format 565 |

250 {

251 #if SENSEHAT\_EMULATOR

252          char ltime [120];

253          float conv;

254          uint8\_t red,green,blue = 0;

255

256          red = ((color & RED) >> 11)/31.0\*255.0;

257          green = ((color & GREEN) >> 5)/63.0\*255.0;

258          blue = ((color & BLUE))/31.0\*255.0;

259          sprintf(ltime,

260                  "from sense\_emu import SenseHat\n"

261                  "sense=SenseHat()\n"

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

263                  ,column,row,red,green,blue);

264          PyRun\_SimpleString(ltime);

265 #else

266          if(row < 0)

267          row = 0;

268          if(column < 0)

269          column = 0;

270

271          fb->pixel[row%8][column%8] = color;

272 #endif

273 }

#### uint16\_t SenseHat::GetPixel (int    *row*, int    *column*)

##### Parameters

|  |  |
| --- | --- |
| *row* | integer |
| *column* | integer |

##### Returns

uint16\_t color in 565 format

282 {

283          if(row < 0) { row = 0; }

284          if(column < 0) { column = 0; }

285          return fb->pixel[row%8][column%8] ;

286 }

#### void SenseHat::ViewPattern (uint16\_t    *pattern*[][8])

##### Parameters

|  |  |
| --- | --- |
| *pattern* | uint16\_t 8x8 arrays |

293 {

294          for(int row=0; row <8 ; row++)

295          {

296                  for(int column=0 ; column <8 ; column++)

297                  {

298                          switch(this->rotation)

299                          {

300                                  case      90:

301                                  case -270:

302                                          fb->pixel[7 - column][row] = pattern[row][column];

303                                          break;

304                                  case    180:

305                                  case -180:

306                                          fb->pixel[7 - row][7 - column] = pattern[row][column];

307                                          break;

308                                  case    270:

309                                  case    -90:

310                                          fb->pixel[column][7 - row] = pattern[row][column];

311                                          break;

312                                  default:

313                                          fb->pixel[row][column] = pattern[row][column];

314                          }

315                  }

316          }

317 }

#### void SenseHat::RotatePattern (int    *angle*)

##### Parameters

|  |  |
| --- | --- |
| *int* | angle de rotation 90, 180, 270, -90, -180, -270 |

324 {

325          uint16\_t tabAux[8][8];

326

327          for(int row=0; row <8 ; row++)

328          {

329                  for(int column=0 ; column <8 ; column++)

330                  {

331                          switch(angle)

332                          {

333                                  case      90:

334                                  case -270:

335                                          tabAux[7 - column][row] = fb->pixel[row][column];

336                                          break;

337                                  case    180:

338                                  case -180:

339                                          tabAux[7 - row][7 - column] = fb->pixel[row][column];

340                                          break;

341                                  case    270:

342                                  case    -90:

343                                          tabAux[column][7 - row] = fb->pixel[row][column];

344                                          break;

345                                  default:

346                                          tabAux[row][column] = fb->pixel[row][column];

347                          }

348                  }

349          }

350          ViewPattern(tabAux);

351 }

#### char SenseHat::ScannerJoystick (void )

#### char SenseHat::ScanJoystick (void )

##### Returns

le code équivalent aux touches du clavier enter, fleche droite, gauche, haut et bas.

394 {

395          return handle\_events(joystick);

396 }

#### uint16\_t SenseHat::ConvertRGB565 (uint8\_t    *red*, uint8\_t    *green*, uint8\_t    *blue*)

##### Parameters

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

##### Returns

uint16\_t RGB565 color code

406 {

407          blue &= 0xF8;

408          green &= 0xFC;

409          red &= 0xF8;

410          return ((red<<8) + (green<<3) + (blue>>3));

411 }

#### uint16\_t SenseHat::ConvertRGB565 (uint8\_t    *color*[])

##### Parameters

|  |  |
| --- | --- |
| *un* | tableau de trois uint8\_t |

##### Returns

uint16\_t une color codée en RGB565

permet de convertir une color exprimer sous la forme d'un tableaude trois entiers non signés sur 8 bits en un entier représentant la color codée en RGB565

422 {

423          return ConvertRGB565(color[1],color[2],color[3]);

424 }

#### COLOR\_SENSEHAT SenseHat::ConvertRGB565 (std::string    *hexCode*)

##### Parameters

|  |  |
| --- | --- |
| *string* | chaine de caratère représentant une color au format hexa |

##### Returns

uint16\_t une color codée en RGB565

permet de convertir une color exprimer sous la forme d'une chaîne de caractère au format hexa en un entier représentant la color codée en RGB565

435 {

436        int r, g, b;

437

438        // Retire le hashtag ...

439        if(hexCode.at(0) == '#')

440        {

441              hexCode = hexCode.erase(0, 1);

442        }

443        // puis extraction des valeurs r g b.

444        std::istringstream(hexCode.substr(0,2)) >> std::hex >> r;

445        std::istringstream(hexCode.substr(2,2)) >> std::hex >> g;

446        std::istringstream(hexCode.substr(4,2)) >> std::hex >> b;

447

448        return ConvertRGB565(r,g,b);

449 }

#### void SenseHat::WipeScreen (uint16\_t    *color* = BLACK)

##### Parameters

|  |  |
| --- | --- |
| *color* | uint16\_t RG%565 |

358 {

359 #if SENSEHAT\_EMULATOR

360          if (numReadings >=12)

361          {

362                  numReadings=0;

363                  fprintf(stdout,"12 readings is about the limit for the emulator\n"

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

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

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

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

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

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

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

371                  exit(EXIT\_FAILURE);

372          }

373          else

374          {

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

376                  numReadings++;

377          }

378                  PyRun\_SimpleString(

379                  "from sense\_emu import SenseHat\n"

380                  "sense=SenseHat()\n"

381                  "sense.clear()\n"

382                  );

383 #else

384          memset(fb, color, 128);

385 #endif

386 }

#### float SenseHat::GetTemperature (void )

##### Returns

float la valeur de la température exprimée en °C,

456 {

457          float cpuTemp;

458          float correctedTemp;

459          float senseHatTemp;

460

461          senseHatTemp = getRawTemperature();

462          cpuTemp = getCpuTemperature();

463

464          //temp\_calibrated = temp - ((cpu\_temp - temp)/FACTOR)

465          correctedTemp = correctTemperature(senseHatTemp, cpuTemp);

466          return (correctedTemp);

467 }

#### float SenseHat::correctTemperature (float    *senseHatTemp*, float    *cpuTemp*)

470 {

471          float correctedTemp;

472          float TEMPERATUREFACTOR = 1.2;

473

474          //temp\_calibrated = temp - ((cpu\_temp - temp)/FACTOR)

475          correctedTemp = senseHatTemp - ((cpuTemp - senseHatTemp)/TEMPERATUREFACTOR);

476          return (correctedTemp);

477 }

#### float SenseHat::getRawTemperature (void )

##### Returns

float la valeur de la température exprimée en °C,

484 {

485          float senseHatTemp;

486 #if SENSEHAT\_EMULATOR

487          double reading=0;

488          FILE \*fp;

489

490          PyRun\_SimpleString(

491                  "from sense\_emu import SenseHat\n"

492                  "sense=SenseHat()\n"

493                  "temp=sense.temp\n"

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

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

496                  "f.close()\n"

497                  );

498

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

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

501          fclose(fp);

502          senseHatTemp = reading;

503 #else

504          RTIMU\_DATA data;

505

506          pressure->pressureRead(data);

507          senseHatTemp = data.temperature;

508 #endif

509          return senseHatTemp;

510 }

#### float SenseHat::getCpuTemperature (void )

##### Returns

float la valeur de la température exprimée en °C,

517 {

518          FILE \*temperatureFile;

519          double T;

520

521          temperatureFile = fopen ("/sys/class/thermal/thermal\_zone0/temp", "r");

522          if (temperatureFile == NULL)

523          {

524 //            printf ("Error getting Core Temperature!")    ; //print some message

525              return (0);

526          }

527          fscanf (temperatureFile, "%lf", &T);

528          T /= 1000;

529 //        printf ("The temperature is %6.3f C.\n", T);

530          fclose (temperatureFile);

531          return(T);

532 }

#### float SenseHat::GetPressure (void )

##### Returns

float la valeur de la Pression exprimée en hPa,

540 {

541          float pression = nan("");    // initialise la valeur à Not-A-Number

542 #if SENSEHAT\_EMULATOR

543          double reading=0;

544          FILE \*fp;

545

546          PyRun\_SimpleString(

547                  "from sense\_emu import SenseHat\n"

548                  "sense=SenseHat()\n"

549                  "temp=sense.pressure\n"

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

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

552                  "f.close()\n"

553                  );

554

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

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

557          fclose(fp);

558          pression = reading;

559 #else

560          RTIMU\_DATA data;

561

562          if (pressure->pressureRead(data))

563          {

564                  if (data.pressureValid)

565                  {

566                          pression = data.pressure;

567                  }

568          }

569 #endif

570          return pression;

571 }

#### float SenseHat::GetHumidity (void )

##### Returns

float la valeur de l'humidité relative exprimée en %,

578 {

579          float humidi = nan("");    // initialise la valeur à Not-A-Number

580 #if SENSEHAT\_EMULATOR

581          double reading=0;

582          FILE \*fp;

583

584          PyRun\_SimpleString(

585                  "from sense\_emu import SenseHat\n"

586                  "sense=SenseHat()\n"

587                  "humid=sense.humidity\n"

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

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

590                  "f.close()\n"

591                  );

592

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

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

595          fclose(fp);

596          humidi = reading;

597 #else

598          RTIMU\_DATA data;

599

600          if (humidity->humidityRead(data))

601          {

602                  if (data.humidityValid)

603                  {

604                          humidi = data.humidity;

605                  }

606          }

607 #endif

608          return humidi;

609 }

#### void SenseHat::GetOrientation (float &    *pitch*, float &    *roll*, float &    *yaw*)

##### Returns

float la valeur de l'accélération angulaire suivant pitch roll et yaw

616 {

617 #if SENSEHAT\_EMULATOR

618          FILE \*fp;

619          float p,r,y;

620

621          PyRun\_SimpleString

622          (

623                  "from sense\_emu import SenseHat\n"

624                  "sense=SenseHat()\n"

625                  "p, r, y = sense.get\_orientation().values()\n"

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

627                  "f.write(\"%s %s %s\" % (p,r,y))\n"

628                  "f.close()\n"

629          );

630

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

632          fscanf(fp, "%f %f %f", &pitch,&roll,&yaw);

633          fclose(fp);

634 #else

635          while (imu->IMURead())

636          {

637                  RTIMU\_DATA imuData = imu->getIMUData();

638                  pitch = imuData.gyro.x();

639                  roll    = imuData.gyro.y();

640                  yaw      = imuData.gyro.z();

641          }

642 #endif

643 }

#### void SenseHat::GetAcceleration (float &    *x*, float &    *y*, float &    *z*)

##### Returns

float la valeur de l'accélération linéaire suivant X,Y,Z

650 {

651 #if SENSEHAT\_EMULATOR

652          FILE \*fp;

653          float ax,ay,az;

654

655          PyRun\_SimpleString

656          (

657                  "from sense\_emu import SenseHat\n"

658                  "sense=SenseHat()\n"

659                  "ax, ay, az = sense.get\_accelerometer\_raw().values()\n"

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

661                  "f.write(\"%s %s %s\" % (ax,ay,az))\n"

662                  "f.close()\n"

663          );

664

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

666          fscanf(fp, "%f %f %f", &z,&y,&z);

667          fclose(fp);

668 #else

669          while (imu->IMURead())

670          {

671                  RTIMU\_DATA imuData = imu->getIMUData();

672                  x = imuData.accel.x();

673                  y = imuData.accel.y();

674                  z = imuData.accel.z();

675          }

676 #endif

677 }

#### void SenseHat::GetMagnetism (float &    *x*, float &    *y*, float &    *z*)

##### Returns

float la valeur du champ magnétique terrestre suivant X,Y,Z

685 {

686 #if SENSEHAT\_EMULATOR

687          FILE \*fp;

688          float mx,my,mz;

689

690          PyRun\_SimpleString

691          (

692                  "from sense\_emu import SenseHat\n"

693                  "sense=SenseHat()\n"

694                  "mx, my, mz = sense.get\_compass\_raw().values()\n"

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

696                  "f.write(\"%s %s %s\" % (mx,my,mz))\n"

697                  "f.close()\n"

698          );

699

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

701          fscanf(fp, "%f %f %f", &z,&y,&z);

702          fclose(fp);

703 #else

704          while (imu->IMURead())

705          {

706                  RTIMU\_DATA imuData = imu->getIMUData();

707                  x = imuData.compass.x();

708                  y = imuData.compass.y();

709                  z = imuData.compass.z();

710          }

711 #endif

712 }

#### void SenseHat::GetSphericalMagnetism (float &    *ro*, float &    *teta*, float &    *delta*)

##### Returns

la valeur du vecteur champ magnétique en coordonnées sphérique

719 {

720          float x,y,z;

721 #if SENSEHAT\_EMULATOR

722          ro = teta = delta = 1.0;

723 #else

724          GetMagnetism(x,y,z);

725          teta = atan2 (y,x) \* 180/PI;

726          ro      = sqrt(x\*x + y\*y + z\*z);

727          delta =    atan2 (z,sqrt(x\*x + y\*y)) \* 180/PI;

728 #endif

729 }

#### void SenseHat::Version (void )

affiche la version de la classe

214 {

215          std::cout << "SenseHat PCT,PSR,CGO Version 1.2.0" << std::endl;

216 }

#### void SenseHat::Flush (void )

1043 {

1044          buffer += "    ";

1045          ViewMessage(buffer, 80, color);

1046          buffer = " ";

1047 }

#### void SenseHat::SetColor (uint16\_t    *\_color*)

219 {

220          color = \_color;

221 }

#### void SenseHat::SetRotation (uint16\_t    *\_rotation*)

224 {

225          rotation = \_rotation;

226 }

#### void SenseHat::InitializeLeds (void )[private]

737 {

738          int fbfd ;

739          int tries;

740          int tries2;

741

742          tries = 0;

743          tries2 = 0;

744          while(true)

745          {

746              fbfd = open\_fbdev("RPi-Sense FB");

747              if (fbfd > 0)

748              {

749                      while (true)

750                      {

751                          fb = (struct fb\_t\*)mmap(0, 128, PROT\_READ | PROT\_WRITE, MAP\_SHARED, fbfd, 0);

752                          if (!fb)

753                          {

754                                  tries2++;

755                                  usleep(100);

756                                  if (tries2 > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

757                                  {

758                                      printf("Failed to mmap.\n");

759                                      exit(EXIT\_FAILURE);

760                                  }

761                          }

762                          else

763                          {

764

765                              memset(fb, 0, 128);

766                              return;

767                          }

768

769                      }

770              }

771              else

772              {

773                  close(fbfd);

774                  tries++;

775                  usleep(100);

776                  if (tries > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

777                  {

778                      printf("Error: cannot open framebuffer device.\n");

779                      exit (EXIT\_FAILURE);

780                  }

781              }

782

783          }

784

785 }

#### void SenseHat::InitializeJoystick (void )[private]

791 {

792          joystick = open\_evdev("Raspberry Pi Sense HAT Joystick");

793 }

#### void SenseHat::InitializePressure (void )[private]

799 {

800          int tries;

801

802          tries = 0;

803          while(true)

804          {

805                  pressure = RTPressure::createPressure(settings);

806                  if (pressure == NULL)

807                  {

808                          tries++;

809                          usleep(100);

810                  }

811                  else

812                  {

813                          break;

814                  }

815                  if (tries > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

816                  {

817                          fprintf(stdout,"Pas de mesure de pression/température \n");

818                          exit (EXIT\_FAILURE);

819                  }

820          }

821          pressure->pressureInit();

822 }

#### void SenseHat::InitializeHumidity (void )[private]

829 {

830          int tries;

831

832          tries = 0;

833          while(true)

834          {

835                  humidity = RTHumidity::createHumidity(settings);

836                  if(humidity == NULL)

837                  {

838                          tries++;

839                          usleep(100);

840                  }

841                  else

842                  {

843                          break;

844                  }

845                  if (tries > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

846                  {

847                          fprintf(stdout,"Pas de mesure de pression/température \n");

848                          exit (EXIT\_FAILURE);

849                  }

850          }

851          humidity->humidityInit();

852 }

#### void SenseHat::InitializeOrientation (void )[private]

855 {

856

857 }

#### void SenseHat::InitializeAcceleration (void )[private]

860 {

861          imu->setAccelEnable(true);

862 }

#### void SenseHat::ConvertCharacterToPattern (char    *c*, uint16\_t    *image*[8][8], uint16\_t    *colorText*, uint16\_t    *colorBackground*)[private]

Fait par Grilo Christophe

870 {

871          int i=0;

872          int j,k;

873          int tailleTableDeConvertion=sizeof(font)/sizeof(Tfont);

874

875          // Recherche si le caractere existe dans la table de convertion (cf font.h)

876          while(c!=font[i].caractere && i < tailleTableDeConvertion )

877          i++;

878

879          // Si le caractere est dans la table on le converti

880          if(i < tailleTableDeConvertion)

881          {

882                  for (j=0;j<8;j++)

883                  {

884                          for(k=0;k<8;k++)

885                          {

886                                  if(font[i].binarypattern[j][k]) { image[j][k]=colorText; }

887                                  else { image[j][k]=colorBackground; }

888                          }

889                  }

890          }

891          else // caractère inexistant on le remplace par un glyphe inconnu

892 //    ConvertirCaractereEnpattern(255,image,colorText,colorBackground);

893          {

894                  ConvertCharacterToPattern(255,image,colorText,colorBackground);

895          }

896 }

#### bool SenseHat::EmptyColumn (int    *numcolumn*, uint16\_t    *image*[8][8], uint16\_t    *colorBackground*)[private]

899 {

900          int i=0;

901

902          for(i=0;i<8;i++)

903          {

904                  if(image[i][numcolumn]!=colorBackground) { return false; }

905          }

906          return true;

907 }

#### void SenseHat::ImageContainment (int    *numcolumn*, uint16\_t    *image*[][8][8], int    *taille*)[private]

910 {

911          int i=0,j=0,k=0,l=0,isuivant,ksuivant;

912          int nombredecolumns=taille\*8; //8 columns par pattern

913

914          for(l=numcolumn;l<nombredecolumns-1;l++)

915          {

916                  i=l/8;

917                  k=l%8;

918                  isuivant=(l+1)/8;

919                  ksuivant=(l+1)%8;

920                  for(j=0;j<8;j++)

921                  {

922                          image[i][j][k]=image[isuivant][j][ksuivant];

923                  }

924          }

925 }

### Field Documentation

#### struct fb\_t\* SenseHat::fb[private]

#### int SenseHat::joystick[private]

#### RTIMUSettings\* SenseHat::settings[private]

#### RTIMU\* SenseHat::imu[private]

#### RTPressure\* SenseHat::pressure[private]

#### RTHumidity\* SenseHat::humidity[private]

#### std::string SenseHat::buffer[private]

#### uint16\_t SenseHat::color[private]

#### int SenseHat::rotation[private]

#### The documentation for this class was generated from the following files:

**sensehat.h**

**sensehat.cpp**

# File Documentation

## dlgps.cpp File Reference

Data logger gps Functions.

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <cmath>

#include "dlgps.h"

#include "nmea.h"

#include "serial.h"

### Functions

void **DlGpsInit** (void)

*Initializes GPS Module.*

void **DlGpsOn** (void)

*Turns on GPS Module.*

**loc\_t** **DlGpsLocation** (void)

*Compute the GPS location using decimal scale.*

void **DlGpsOff** (void)

*Turns Off GPS Module.*

void **DlGpsConvertDegToDec** (double \*latitude, char ns, double \*longitude, char we)

*Convert lat e lon to decimals (from deg)*

double **DlGpsDegDec** (double deg\_point)

*Convert GPS points to decimal.*

### Variables

FILE \* **fpgps** = NULL

### Function Documentation

#### void DlGpsInit (void )

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *None* |  |

##### Returns

void

24 {

25 #if SIMGPS

26          fpgps = fopen("gpstestdata.txt","r");

27          if(fpgps == NULL) { fprintf(stdout,"Unable to open gps test data file\n"); }

28 #else

29          // Serial GPS device or GPSD server

30          serial\_init();

31          serial\_config();

32 #endif

33 }

#### void DlGpsOn (void )

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *None* |  |

##### Returns

void

42 {

43          //Write on

44 }

#### loc\_t DlGpsLocation (void )

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

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

##### Returns

coord loc\_t data structure

53 {

54          loc\_t cloc = {0.0};

55          char buffer[GPSDATASZ] = {0};

56          uint8\_t status = \_EMPTY;

57          gpgga\_t gpgga;

58          gprmc\_t gprmc;

59

60          while(status != \_COMPLETED)

61          {

62 #if SIMGPS

63                  fgets(buffer,NMEAMSGSZ,fpgps);

64                  if(feof(fpgps)) { rewind(fpgps); }

65 #else

66                  serial\_readln(buffer,GPSDATASZ);

67 #endif

68                  switch (nmea\_get\_message\_type(buffer))

69                  {

70                          case NMEA\_GPGGA:

71                                  nmea\_parse\_gpgga(buffer, &gpgga);

72                                  cloc.utc = gpgga.utc;

73                                  DlGpsConvertDegToDec(&(gpgga.latitude), gpgga.lat, &(gpgga.longitude), gpgga.lon);

74                                  cloc.latitude = gpgga.latitude;

75                                  cloc.longitude = gpgga.longitude;

76                                  cloc.altitude = gpgga.altitude;

77                                  status |= NMEA\_GPGGA;

78                                  break;

79                          case NMEA\_GPRMC:

80                                  nmea\_parse\_gprmc(buffer, &gprmc);

81                                  cloc.speed = gprmc.speed;

82                                  cloc.course = gprmc.course;

83                                  cloc.date = gprmc.date;

84                                  status |= NMEA\_GPRMC;

85                                  break;

86                  }

87          }

88          return cloc;

89 }

#### void DlGpsOff (void )

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *None* |  |

##### Returns

void

98 {

99 #if SIMGPS

100          fclose(fpgps);

101 #endif

102 //        serial\_close();

103 }

#### void DlGpsConvertDegToDec (double \*    *latitude*, char    *ns*, double \*    *longitude*, char    *we*)

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *latitude* | double \* |
| *ns* | char |
| *longitude* | double \* |
| *we* | char |

##### Returns

void

115 {

116          double lat = (ns == 'N') ? \*latitude : -1 \* (\*latitude);

117          double lon = (we == 'E') ? \*longitude : -1 \* (\*longitude);

118

119          \*latitude = DlGpsDegDec(lat);

120          \*longitude = DlGpsDegDec(lon);

121 }

#### double DlGpsDegDec (double    *deg\_point*)

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *deg\_point* | double |

##### Returns

double

130 {

131          double ddeg;

132          double sec = modf(deg\_point, &ddeg)\*60;

133          int deg = (int)(ddeg/100);

134          int min = (int)(deg\_point-(deg\*100));

135

136          double absmlat = round(min \* 1000000.);

137          double absslat = round(sec \* 1000000.);

138          double absdlat = round(deg \* 1000000.);

139

140          return round(absdlat + (absmlat/60) + (absslat/3600)) /1000000;

141 }

### Variable Documentation

#### FILE\* fpgps = NULL

#### 

## dlgps.h File Reference

Constants, structures, function prototypes for gps functions.

#include <cmath>

### Data Structures

struct **location**

### Macros

#define **round**(x)  ((x < 0) ? (ceil((x)-0.5)) : (floor((x)+0.5)))

#define **SIMGPS**  1

#define **GPSSERIAL**  0

#define **GPSDATASZ**  256

### Typedefs

typedef struct **location** **loc\_t**

### Data Structure Documentation

#### struct location

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | utc |  |
| double | date |  |
| double | latitude |  |
| double | longitude |  |
| double | speed |  |
| double | altitude |  |
| double | course |  |

### Macro Definition Documentation

#### #define round( x)  ((x < 0) ? (ceil((x)-0.5)) : (floor((x)+0.5)))

#### #define SIMGPS  1

#### #define GPSSERIAL  0

#### #define GPSDATASZ  256

### Typedef Documentation

#### typedef struct location loc\_t

#### 

## font.h File Reference

### Data Structures

struct **Tfont**

### Variables

const **Tfont** **font** []

### Data Structure Documentation

#### struct Tfont

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| char | caractere |  |
| bool | binarypattern[8][8] |  |

### Variable Documentation

#### const Tfont font[]

#### 

## logger.cpp File Reference

Vehicle Data logger functions.

#include "logger.h"

#include <cinttypes>

#include <cstdio>

#include <cstring>

#include <ctime>

#include <unistd.h>

### Functions

int **DlInitialization** (void)

*Initialize data logger, currently prints header.*

uint64\_t **DlGetSerial** (void)

*Gets the serial number from the raspi from /proc/cpuinfo.*

void **DlDisplayLoggerReadings** (**reading\_s** dreads)

*Prints the logger info to the console.*

**reading\_s** **DlGetLoggerReadings** (void)

*Gets the reading results from the system, and adds items (system time, currently default data)*

int **DlSaveLoggerData** (**reading\_s** creads)

*Currently just prints a message, will save logger data to storage.*

### Detailed Description

##### Author

Robert Miller

##### Date

23Jan2022

### Function Documentation

#### int DlInitialization (void )

##### Author

Robert Miller

##### Date

23Jan2022

##### Parameters

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

##### Returns

int

27                                                        {

28      fprintf(stdout, "\nData Logger Initialization\n");

29      return 1;

30 }

#### uint64\_t DlGetSerial (void )

##### Author

Robert Miller

##### Date

23Jan2022

##### Parameters

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

##### Returns

uint64\_t 'serial'

38                                                        {

39      static uint64\_t serial = 0;

40      FILE \*fp;

41      char buf[SYSINFOBUFSZ];

42      char searchstring[] = SEARCHSTR;

43      fp = fopen("/proc/cpuinfo", "r");

44      if (fp != NULL) {

45          while (fgets(buf, sizeof(buf), fp) != NULL) {

46              if (!strncasecmp(searchstring, buf, strlen(searchstring))) {

47                  sscanf(buf + strlen(searchstring), "%Lx", &serial);

48              }

49          }

50          fclose(fp);

51      }

52      if (serial == 0) {

53          system("uname -a");

54          system("ls --fu /usr/lib/codeblocks | grep -Po '\\.\\K[^ ]+'>stamp.txt");

55          fp = fopen("stamp.txt", "r");

56          if (fp != NULL) {

57              while (fgets(buf, sizeof(buf), fp) != NULL) {

58                  sscanf(buf, "%Lx", &serial);

59              }

60              fclose(fp);

61          }

62      }

63      return serial;

64 }

#### void DlDisplayLoggerReadings (reading\_s    *dreads*)

##### Author

Robert Miller

##### Date

23Jan2022

##### Parameters

|  |  |
| --- | --- |
| *struct* | reading\_s |

##### Returns

void

72                                                                                                {

73      fprintf(stdout, "\nUnit:%lu \t", DlGetSerial());

74      fprintf(stdout, " %s", ctime(&dreads.rtime));

75      fprintf(stdout, "T: %0.1fC \t", dreads.temperature);

76      fprintf(stdout, "H: %0.0f%% \t", dreads.humidity);

77      fprintf(stdout, "P: %0.1f KPa\n", dreads.pressure);

78      fprintf(stdout, "Xa: %f g\t", dreads.xa);

79      fprintf(stdout, "Ya: %f g\t", dreads.ya);

80      fprintf(stdout, "Za: %f g\n", dreads.za);

81          fprintf(stdout, "Pitch: %f \t", dreads.pitch);

82          fprintf(stdout, "Roll: %f \t", dreads.roll);

83          fprintf(stdout, "Yaw: %f \t \n", dreads.yaw);

84          fprintf(stdout, "Xm: %f \t", dreads.xm);

85          fprintf(stdout, "Ym: %f \t", dreads.ym);

86          fprintf(stdout, "Zm: %f \t \n", dreads.zm);

87      fprintf(stdout, "Latitude: %f \t", dreads.latitude);

88      fprintf(stdout, "Longitude: %f \t", dreads.longitude);

89      fprintf(stdout, "Altitude: %f \n", dreads.altitude);

90      fprintf(stdout, "Speed: %f \t", dreads.speed);

91      fprintf(stdout, "Heading: %f \n", dreads.heading);

92 }

#### reading\_s DlGetLoggerReadings (void )

##### Author

Robert Miller

##### Date

23Jan2022

##### Parameters

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

##### Returns

struct reading\_s

101                                                                          {

102      reading\_s creads;

103      creads.rtime = time(NULL);

104 #if SENSEHAT == 1

105      usleep(IMUDELAY);

106      creads.temperature = Sh.GetTemperature();

107      creads.humidity = Sh.GetHumidity();

108      creads.pressure = Sh.GetPressure();

109      Sh.GetAcceleration(creads.xa, creads.ya, creads.za);

110      Sh.GetOrientation(creads.pitch, creads.roll, creads.yaw);

111      Sh.GetMagnetism(creads.xm, creads.ym, creads.zm);

112 #else

113      creads.temperature = DTEMP;

114      creads.humidity = DHUMID;

115          creads.pressure = DPRESS;

116      creads.xa = DXA;

117      creads.ya = DYA;

118      creads.za = DZA;

119      creads.pitch = DPITCH;

120      creads.roll = DROLL;

121      creads.yaw = DYAW;

122      creads.xm = DXM;

123      creads.ym = DYM;

124          creads.zm = DZM;

125 #endif

126      creads.latitude = DLAT;

127      creads.longitude = DLONG;

128      creads.altitude = DALT;

129      creads.speed = DSPEED;

130      creads.heading = DHEADING;

131      return creads;

132 }

#### int DlSaveLoggerData (reading\_s    *creads*)

##### Author

Robert Miller

##### Date

23Jan2022

##### Parameters

|  |  |
| --- | --- |
| *struct* | reading\_s creads |

##### Returns

int

140                                                                                {

141      fprintf(stdout, "\nSaving Logger Data\n");

142      return 0;

143 }

## logger.h File Reference

#include <cinttypes>

#include <cstdlib>

### Data Structures

struct **readings**

### Macros

#define **DTEMP**  24.6

#define **DHUMID**  32

#define **DPRESS**  1013.5

#define **DXA**  1

#define **DYA**  1

#define **DZA**  1

#define **DPITCH**  10

#define **DROLL**  15

#define **DYAW**  20

#define **DXM**  1

#define **DYM**  1

#define **DZM**  1

#define **DLAT**  43.7289

#define **DLONG**  -79.6074

#define **DALT**  166

#define **DSPEED**  99

#define **DHEADING**  320

#define **SEARCHSTR**  "serial\t\t:"

#define **SYSINFOBUFSZ**  512

#define **SENSEHAT**  0

### Typedefs

typedef struct **readings** **reading\_s**

### Detailed Description

Serial: e69836d

##### Author

Robert Miller

##### Date

22Jan2022

### Data Structure Documentation

#### struct readings

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| time\_t | rtime | Reading time. |
| float | temperature | Degrees Celsius. |
| float | humidity | Per cent relative humidity. |
| float | pressure | Kilo Pascals. |
| float | xa | X-axis accelaration. |
| float | ya | Y-axis accelaration. |
| float | za | Z-axis accelaration. |
| float | pitch | Pitch angle. |
| float | roll | Roll angle. |
| float | yaw | Yaw angle. |
| float | xm | X axis micro Teslas. |
| float | ym | Y axis micro Teslas. |
| float | zm | Z axis micro Teslas. |
| float | latitude | Latitude. |
| float | longitude | Longitude. |
| float | altitude | Altitude. |
| float | speed | Speed kph. |
| float | heading | Heading degrees True. |

### Macro Definition Documentation

#### #define DTEMP  24.6

#### #define DHUMID  32

#### #define DPRESS  1013.5

#### #define DXA  1

#### #define DYA  1

#### #define DZA  1

#### #define DPITCH  10

#### #define DROLL  15

#### #define DYAW  20

#### #define DXM  1

#### #define DYM  1

#### #define DZM  1

#### #define DLAT  43.7289

#### #define DLONG  -79.6074

#### #define DALT  166

#### #define DSPEED  99

#### #define DHEADING  320

#### #define SEARCHSTR  "serial\t\t:"

#### #define SYSINFOBUFSZ  512

#### #define SENSEHAT  0

### Typedef Documentation

#### typedef struct readings reading\_s

#### 

## nmea.cpp File Reference

#include <cstdio>

#include <cstdlib>

#include <cstdint>

#include <cstring>

#include <cmath>

#include "nmea.h"

### Functions

void **nmea\_parse\_gpgga** (char \*nmea, **gpgga\_t** \*loc)

*Parses GPGGA Message.*

void **nmea\_parse\_gprmc** (char \*nmea, **gprmc\_t** \*loc)

*Parses GPRMC Message.*

uint8\_t **nmea\_get\_message\_type** (const char \*message)

*Get the message type (GPGGA, GPRMC, etc..)*

uint8\_t **nmea\_valid\_checksum** (const char \*message)

*Checks Message Checksum.*

### Function Documentation

#### void nmea\_parse\_gpgga (char \*    *nmea*, gpgga\_t \*    *loc*)

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *char* | \* nmea Pointer to message |
| *gpgg\_t* | \* loc Pointer to output data structure |

18 {

19          char \*p = nmea;

20

21          p = strchr(p, ',')+1; // time

22          loc->utc = atof(p);

23

24          p = strchr(p, ',')+1;

25          loc->latitude = atof(p);

26

27          p = strchr(p, ',')+1;

28          switch (p[0])

29          {

30                  case 'N':

31                          loc->lat = 'N';

32                          break;

33                  case 'S':

34                          loc->lat = 'S';

35                          break;

36                  case ',':

37                          loc->lat = '\0';

38                          break;

39          }

40

41          p = strchr(p, ',')+1;

42          loc->longitude = atof(p);

43

44          p = strchr(p, ',')+1;

45          switch (p[0])

46          {

47                  case 'W':

48                          loc->lon = 'W';

49                          break;

50                  case 'E':

51                          loc->lon = 'E';

52                          break;

53                  case ',':

54                          loc->lon = '\0';

55                          break;

56          }

57

58          p = strchr(p, ',')+1;

59          loc->quality = (uint8\_t)atoi(p);

60

61          p = strchr(p, ',')+1;

62          loc->satellites = (uint8\_t)atoi(p);

63

64          p = strchr(p, ',')+1;

65

66          p = strchr(p, ',')+1;

67          loc->altitude = atof(p);

68 }

#### void nmea\_parse\_gprmc (char \*    *nmea*, gprmc\_t \*    *loc*)

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *char* | \* nmea Pointer to message |
| *gpgg\_t* | \* loc Pointer to output data structure |

77 {

78          char \*p = nmea;

79

80          p = strchr(p, ',')+1; // skip time

81          p = strchr(p, ',')+1; //skip status

82          p = strchr(p, ',')+1;

83          loc->latitude = atof(p);

84          p = strchr(p, ',')+1;

85          switch (p[0])

86          {

87                  case 'N':

88                          loc->lat = 'N';

89                          break;

90                  case 'S':

91                          loc->lat = 'S';

92                          break;

93                  case ',':

94                          loc->lat = '\0';

95                          break;

96          }

97

98          p = strchr(p, ',')+1;

99          loc->longitude = atof(p);

100          p = strchr(p, ',')+1;

101          switch (p[0])

102          {

103                  case 'W':

104                          loc->lon = 'W';

105                          break;

106                  case 'E':

107                          loc->lon = 'E';

108                          break;

109                  case ',':

110                          loc->lon = '\0';

111                          break;

112          }

113

114          p = strchr(p, ',')+1;

115          loc->speed = atof(p);

116

117          p = strchr(p, ',')+1;

118          loc->course = atof(p);

119

120          p = strchr(p, ',')+1;

121          loc->date = atof(p);

122 }

#### uint8\_t nmea\_get\_message\_type (const char \*    *message*)

##### Parameters

|  |  |
| --- | --- |
| *message* | The NMEA message |

##### Returns

The type of message if it is valid

129 {

130          uint8\_t checksum = 0;

131          if ((checksum = nmea\_valid\_checksum(message)) != \_EMPTY){ return checksum; }

132

133          if (strstr(message, NMEA\_GPGGA\_STR) != NULL) { return NMEA\_GPGGA; }

134

135          if (strstr(message, NMEA\_GPRMC\_STR) != NULL) { return NMEA\_GPRMC; }

136

137          return NMEA\_UNKNOWN;

138 }

#### uint8\_t nmea\_valid\_checksum (const char \*    *message*)

##### Author

Paul Moggach

##### Date

25MAR2019

##### Parameters

|  |  |
| --- | --- |
| *char* | \* nmea Pointer to message |

##### Returns

uint8\_t Checksum

147 {

148          uint8\_t checksum= (uint8\_t)strtol(strchr(message, '\*')+1, NULL, 16);

149

150          char p;

151          uint8\_t sum = 0;

152          ++message;

153          while ((p = \*message++) != '\*') { sum ^= p; }

154

155          if (sum != checksum) { return NMEA\_CHECKSUM\_ERR; }

156

157          return \_EMPTY;

158 }

## nmea.h File Reference

Constants, structures, function prototypes for NMEA functions.

#include <cstdio>

#include <cstdlib>

#include <cstdint>

### Data Structures

struct **gpgga**

struct **gprmc**

struct **nmeamsg**

### Macros

#define **\_EMPTY**  0x00

#define **NMEA\_GPRMC**  0x01

#define **NMEA\_GPRMC\_STR**  "$GPRMC"

#define **NMEA\_GPGGA**  0x02

#define **NMEA\_GPGGA\_STR**  "$GPGGA"

#define **NMEA\_UNKNOWN**  0x00

#define **\_COMPLETED**  0x03

#define **NMEA\_CHECKSUM\_ERR**  0x80

#define **NMEA\_MESSAGE\_ERR**  0xC0

#define **NMEAMSGSZ**  82

### Typedefs

typedef struct **gpgga** **gpgga\_t**

typedef struct **gprmc** **gprmc\_t**

typedef struct **nmeamsg** **nmeamsg\_s**

### Data Structure Documentation

#### struct gpgga

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | utc | UTC Time. |
| double | latitude | Latitude eg: 4124.8963 (XXYY.ZZKK.. DEG, MIN, SEC.SS) |
| char | lat | Latitude eg: N. |
| double | longitude | Longitude eg: 08151.6838 (XXXYY.ZZKK.. DEG, MIN, SEC.SS) |
| char | lon | Longitude eg: W. |
| uint8\_t | quality | Quality 0, 1, 2. |
| uint8\_t | satellites | Number of satellites: 1,2,3,4,5... |
| double | altitude | Altitude eg: 280.2 (Meters above mean sea level) |

#### struct gprmc

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| double | utc | UTC Time. |
| double | latitude | Latitude eg: 4124.8963 (XXYY.ZZKK.. DEG, MIN, SEC.SS) |
| char | lat | Latitude eg: N. |
| double | longitude | Longitude eg: 08151.6838 (XXXYY.ZZKK.. DEG, MIN, SEC.SS) |
| char | lon | Longitude eg: W. |
| double | speed | Speed. |
| double | course | Direction. |
| double | date | Date. |

#### struct nmeamsg

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| char | msgstr[**NMEAMSGSZ**+1] |  |

### Macro Definition Documentation

#### #define \_EMPTY  0x00

#### #define NMEA\_GPRMC  0x01

#### #define NMEA\_GPRMC\_STR  "$GPRMC"

#### #define NMEA\_GPGGA  0x02

#### #define NMEA\_GPGGA\_STR  "$GPGGA"

#### #define NMEA\_UNKNOWN  0x00

#### #define \_COMPLETED  0x03

#### #define NMEA\_CHECKSUM\_ERR  0x80

#### #define NMEA\_MESSAGE\_ERR  0xC0

#### #define NMEAMSGSZ  82

### Typedef Documentation

#### typedef struct gpgga gpgga\_t

#### typedef struct gprmc gprmc\_t

#### typedef struct nmeamsg nmeamsg\_s

#### 

## sensehat.cpp File Reference

#include <iostream>

#include <stdio.h>

#include <fcntl.h>

#include "sensehat.h"

#include "font.h"

### Macros

#define **NUMBER\_OF\_TRIES\_BEFORE\_FAILURE**  1000

### Functions

static int **is\_framebuffer\_device** (const struct dirent \*dir)

static int **open\_fbdev** (const char \*dev\_name)

static int **is\_event\_device** (const struct dirent \*dir)

static int **open\_evdev** (const char \*dev\_name)

uint16\_t **handle\_events** (int evfd)

**SenseHat** & **endl** (**SenseHat** &os)

**SenseHat** & **flush** (**SenseHat** &os)

**\_Setcolor** **setcolor** (int n)

**SenseHat** & **operator<<** (**SenseHat** &os, **\_Setcolor** color)

**\_SetRotation** **setrotation** (int n)

**SenseHat** & **operator<<** (**SenseHat** &os, **\_SetRotation** rotation)

### Detailed Description

##### Date

4 mars 2018

##### Version

1.2

##### Authors

Philippe SIMIER Philippe CRUCHET Christophe GRILLO

Classe **SenseHat** : Gestion de la carte **SenseHat**

##### Version

1.3

##### Date

30 July 2019

##### Authors

Jon Dellaria bug fixes, method translation to English and Temperature Measurement adjustments required for the Raspberry for a true temperature reading.

##### Version

1.3a

##### Date

05 May 2021

##### Authors

Paul Moggach removed french to compact files and make it more readable. Made changes to conform with Allman style., Python Emulator support

### Macro Definition Documentation

#### #define NUMBER\_OF\_TRIES\_BEFORE\_FAILURE  1000

### Function Documentation

#### static int is\_framebuffer\_device (const struct dirent \*    *dir*)[static]

26 {

27          return strncmp(FB\_DEV\_NAME, dir->d\_name,strlen(FB\_DEV\_NAME)-1) == 0;

28 }

#### static int open\_fbdev (const char \*    *dev\_name*)[static]

31 {

32          struct dirent \*\*namelist;

33          int i, ndev;

34          int fd = -1;

35          struct fb\_fix\_screeninfo fix\_info;

36

37          ndev = scandir(DEV\_FB, &namelist, is\_framebuffer\_device, versionsort);

38          if (ndev <= 0) { return ndev; }

39

40          for (i = 0; i < ndev; i++)

41          {

42                  char fname[64];

43

44                  snprintf(fname, sizeof(fname),"%s/%s", DEV\_FB, namelist[i]->d\_name);

45                  fd = open(fname, O\_RDWR);

46                  if (fd < 0) { continue; }

47                  ioctl(fd, FBIOGET\_FSCREENINFO, &fix\_info);

48                  if (strcmp(dev\_name, fix\_info.id) == 0){ break; }

49                  close(fd);

50                  fd = -1;

51          }

52          for (i = 0; i < ndev; i++) { free(namelist[i]); }

53          return fd;

54 }

#### static int is\_event\_device (const struct dirent \*    *dir*)[static]

57 {

58          return strncmp(EVENT\_DEV\_NAME, dir->d\_name,strlen(EVENT\_DEV\_NAME)-1) == 0;

59 }

#### static int open\_evdev (const char \*    *dev\_name*)[static]

62 {

63          int tries;

64          struct dirent \*\*namelist;

65          int i, ndev;

66          int fd = -1;

67          int sortie = false;

68          char fname[64];

69          char name[256];

70

71          tries = 0;

72          while(true)

73          {

74                  ndev = scandir(DEV\_INPUT\_EVENT, &namelist, is\_event\_device, versionsort);

75                  if (ndev <= 0)

76                  {

77                          tries++;

78                          usleep(100);

79                  }

80                  else

81                  {

82                          break;

83                  }

84                  if (tries > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

85                  {

86                          exit (EXIT\_FAILURE);

87                  }

88          }

89

90          i=0;

91          do

92          {

93                  snprintf(fname, sizeof(fname),"%s/%s", DEV\_INPUT\_EVENT, namelist[i++]->d\_name);

94                  tries = 0;

95                  while(true)

96                  {

97                          fd = open(fname, O\_RDONLY );

98                          if (fd < 0)

99                          {

100                                  tries++;

101                                  usleep(100);

102                          }

103                          else

104                          {

105                                  break;

106                          }

107                          if (tries > NUMBER\_OF\_TRIES\_BEFORE\_FAILURE)

108                          {

109                                  exit (EXIT\_FAILURE);

110                          }

111                  }

112                  ioctl(fd, EVIOCGNAME(sizeof(name)), name);

113                  if (strcmp(dev\_name, name) != 0) { close(fd); }

114                  else { sortie = true; }

115                  }

116          while( i<ndev && sortie != true);

117

118          for (i = 0; i < ndev; i++) { free(namelist[i]); }

119          return fd;

120 }

#### uint16\_t handle\_events (int    *evfd*)

123 {

124          struct input\_event ev;

125          int rd;

126          uint16\_t retour = 0 ;

127          int flag;

128

129          flag = fcntl(evfd,F\_GETFL,0);

130          fcntl(evfd,F\_SETFL,flag | O\_NONBLOCK | O\_NOCTTY);

131

132          rd = read(evfd, &ev, sizeof(struct input\_event));

133          if(rd > 0)

134          {

135                  if(ev.type == EV\_KEY && ev.value == 1) { retour = ev.code; }

136          }

137          return retour;

138 }

#### SenseHat & endl (SenseHat &    *os*)

1052 {

1053        os.Flush();

1054        return os;

1055 }

#### SenseHat & flush (SenseHat &    *os*)

1058 {

1059          os.Flush();

1060          return os;

1061 }

#### \_Setcolor setcolor (int    *n*)

1064 {

1065          return { n };

1066 }

#### SenseHat & operator<< (SenseHat &    *os*, \_Setcolor    *color*)

1069 {

1070          os.SetColor(color.val);

1071          return os;

1072 }

#### \_SetRotation setrotation (int    *n*)

1075 {

1076          return { n };

1077 }

#### SenseHat & operator<< (SenseHat &    *os*, \_SetRotation    *rotation*)

1080 {

1081          os.SetRotation(rotation.val);

1082          return os;

1083 }

## sensehat.h File Reference

#include <stdio.h>

#include <stdlib.h>

#include <stdint.h>

#include <dirent.h>

#include <linux/fb.h>

#include <sys/mman.h>

#include <sys/ioctl.h>

#include <string>

#include <fcntl.h>

#include <unistd.h>

#include <termios.h>

#include <linux/input.h>

#include <sstream>

#include <math.h>

#include <RTIMULib.h>

#include <iostream>

#include <iomanip>

### Data Structures

struct **fb\_t**

class **SenseHat**

struct **\_Setcolor**

struct **\_SetRotation**

### Macros

#define **SENSEHAT\_EMULATOR**  0

#define **DEV\_FB**  "/dev"

#define **FB\_DEV\_NAME**  "fb"

#define **DEV\_INPUT\_EVENT**  "/dev/input"

#define **EVENT\_DEV\_NAME**  "event"

#define **IMUDELAY**  200000

#define **COLOR\_SENSEHAT**  uint16\_t

#define **PI**  3.14159265

#define **RED**  0xF800

#define **BLUE**  0x001F

#define **GREEN**  0x07E0

#define **WHITE**  0xFFFF

#define **BLACK**  0x0000

#define **ORANGE**  0xFC00

#define **CYAN**  0x87FF

#define **MAGENTA**  0xF81F

#define **YELLOW**  0xFFE0

### Functions

**SenseHat** & **endl** (**SenseHat** &os)

**SenseHat** & **flush** (**SenseHat** &os)

**\_Setcolor** **setcolor** (int n)

**SenseHat** & **operator<<** (**SenseHat** &os, **\_Setcolor** color)

**\_SetRotation** **setrotation** (int n)

**SenseHat** & **operator<<** (**SenseHat** &os, **\_SetRotation** rotation)

### Detailed Description

##### Date

4 mars 2018

##### Version

1.2

##### Authors

Philippe SIMIER Philippe CRUCHET Christophe GRILLO

Classe **SenseHat** : Gestion de la carte **SenseHat**

##### Version

1.3

##### Date

30 July 2019

##### Authors

Jon Dellaria bug fixes, method translation to English and Temperature Measurement adjustments required for the Raspberry for a true temperature reading.

##### Version

1.3a

##### Date

01 October 2020

##### Authors

Paul Moggach removed french to compact files and make it more readable. Made changes to conform with Allman style.

### Data Structure Documentation

#### struct fb\_t

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| uint16\_t | pixel[8][8] |  |

#### struct \_Setcolor

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| int | val |  |

#### struct \_SetRotation

##### Data Fields:

|  |  |  |
| --- | --- | --- |
| int | val |  |

### Macro Definition Documentation

#### #define SENSEHAT\_EMULATOR  0

#### #define DEV\_FB  "/dev"

#### #define FB\_DEV\_NAME  "fb"

#### #define DEV\_INPUT\_EVENT  "/dev/input"

#### #define EVENT\_DEV\_NAME  "event"

#### #define IMUDELAY  200000

#### #define COLOR\_SENSEHAT  uint16\_t

#### #define PI  3.14159265

#### #define RED  0xF800

#### #define BLUE  0x001F

#### #define GREEN  0x07E0

#### #define WHITE  0xFFFF

#### #define BLACK  0x0000

#### #define ORANGE  0xFC00

#### #define CYAN  0x87FF

#### #define MAGENTA  0xF81F

#### #define YELLOW  0xFFE0

### Function Documentation

#### SenseHat & endl (SenseHat &    *os*)

1052 {

1053        os.Flush();

1054        return os;

1055 }

#### SenseHat & flush (SenseHat &    *os*)

1058 {

1059          os.Flush();

1060          return os;

1061 }

#### \_Setcolor setcolor (int    *n*)

1064 {

1065          return { n };

1066 }

#### SenseHat & operator<< (SenseHat &    *os*, \_Setcolor    *color*)

1069 {

1070          os.SetColor(color.val);

1071          return os;

1072 }

#### \_SetRotation setrotation (int    *n*)

1075 {

1076          return { n };

1077 }

#### SenseHat & operator<< (SenseHat &    *os*, \_SetRotation    *rotation*)

1080 {

1081          os.SetRotation(rotation.val);

1082          return os;

1083 }

## serial.cpp File Reference

serial Functions

#include <cstdio>

#include <cstdlib>

#include <cinttypes>

#include <cstring>

#include <unistd.h>

#include <fcntl.h>

#include <termios.h>

#include "serial.h"

### Functions

void **serial\_init** (void)

*Serial port setup.*

void **serial\_config** (void)

*Serial port configuration.*

void **serial\_println** (const char \*line, int len)

*Writes a line to the serial port.*

void **serial\_readln** (char \*buffer, int len)

*Reads a line from the serial port.*

void **serial\_close** (void)

*Closes serial port.*

### Variables

int **uart0\_filestream** = -1

### Function Documentation

#### void serial\_init (void )

##### Author

Paul Moggach

##### Date

01JAN2019

20 {

21          uart0\_filestream = open(PORTNAME, O\_RDWR | O\_NOCTTY | O\_NDELAY);

22

23          if (uart0\_filestream == -1)

24          {

25                  //TODO error handling...

26          }

27 }

#### void serial\_config (void )

##### Author

Paul Moggach

##### Date

01JAN2019

34 {

35          struct termios options;

36          tcgetattr(uart0\_filestream, &options);

37          options.c\_cflag = B9600 | CS8 | CLOCAL | CREAD;

38          options.c\_iflag = IGNPAR;

39          options.c\_oflag = 0;

40          options.c\_lflag = 0;

41          tcflush(uart0\_filestream, TCIFLUSH);

42          tcsetattr(uart0\_filestream, TCSANOW, &options);

43 }

#### void serial\_println (const char \*    *line*, int    *len*)

##### Author

Paul Moggach

##### Date

01JAN2019

##### Parameters

|  |  |
| --- | --- |
| *line* | char \* to buffer |
| *len* | number of characters to write |

52 {

53          if (uart0\_filestream != -1)

54          {

55                  char \*cpstr = (char \*)malloc((len+1) \* sizeof(char));

56                  strcpy(cpstr, line);

57                  cpstr[len-1] = '\r';

58                  cpstr[len] = '\n';

59

60                  int count = write(uart0\_filestream, cpstr, len+1);

61                  if (count < 0)

62                  {

63                          //TODO: handle errors...

64                  }

65                  free(cpstr);

66          }

67 }

#### void serial\_readln (char \*    *buffer*, int    *len*)

##### Author

Paul Moggach

##### Date

01JAN2019

##### Parameters

|  |  |
| --- | --- |
| *buffer* | char \* |
| *len* | int number of characters to read |

76 {

77          char c;

78          char \*b = buffer;

79          int rx\_length = -1;

80          while(1)

81          {

82                  rx\_length = read(uart0\_filestream, (void\*)(&c), 1);

83

84                  if (rx\_length <= 0)

85                  {

86                          //wait for messages

87                          sleep(1);

88                  } else

89                  {

90                          if (c == '\n')

91                          {

92                                  \*b++ = '\0';

93                                  break;

94                          }

95                          \*b++ = c;

96                  }

97          }

98 }

#### void serial\_close (void )

##### Author

Paul Moggach

##### Date

01JAN2019

105 {

106          close(uart0\_filestream);

107 }

### Variable Documentation

#### int uart0\_filestream = -1

#### 

## serial.h File Reference

Constants, structures, function prototypes for serial functions.

#include <inttypes.h>

### Macros

#define **PORTNAME**  "/dev/ttyS0"

### Macro Definition Documentation

#### #define PORTNAME  "/dev/ttyS0"

#### 

## vdl.cpp File Reference

Vehicle Data Logger main function.

#include <unistd.h>

#include "vdl.h"

#include "logger.h"

### Functions

int **main** (void)

*Vehicle Data Logger main function.*

### Detailed Description

Serial: e69836d

##### Author

Robert Miller

##### Date

16Jan2022

### Function Documentation

#### int main (void )

##### Author

Robert Miller

##### Date

16Jan2022

##### Parameters

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

##### Returns

int program status

17 {

18          reading\_s reads = {0};

19      DlInitialization();

20      while (1) {

21          reads = DlGetLoggerReadings();

22          DlDisplayLoggerReadings(reads);

23          DlSaveLoggerData(reads);

24          sleep(5);

25          }

26 }

## vdl.h File Reference

### Macros

#define **SLEEPTIME**  500000

#define **LOGCOUNT**  10

### Macro Definition Documentation

#### #define SLEEPTIME  500000

#### #define LOGCOUNT  10

# Index

INDEX