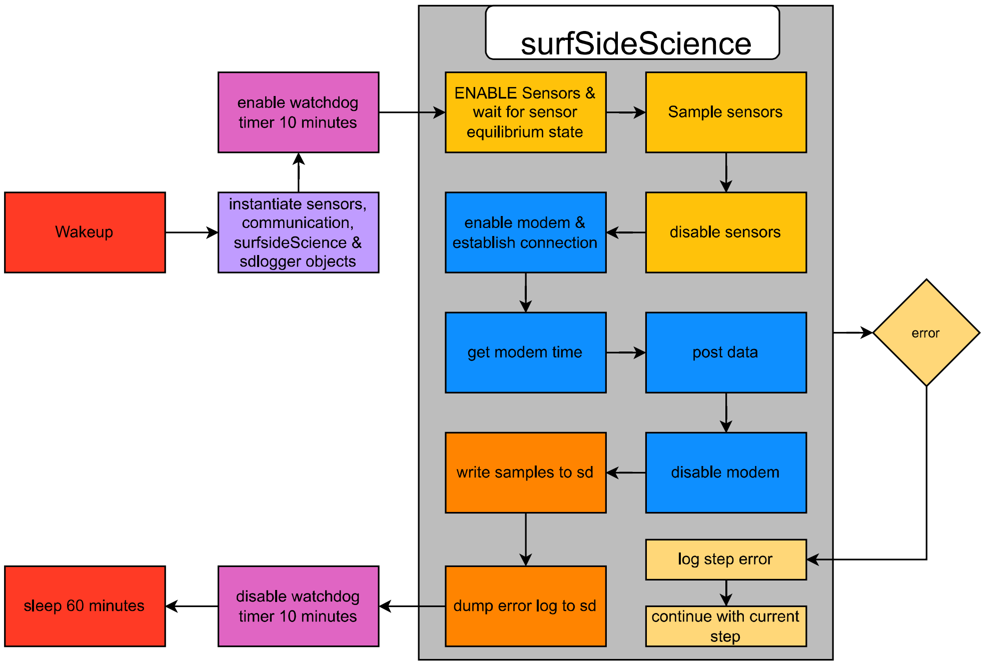
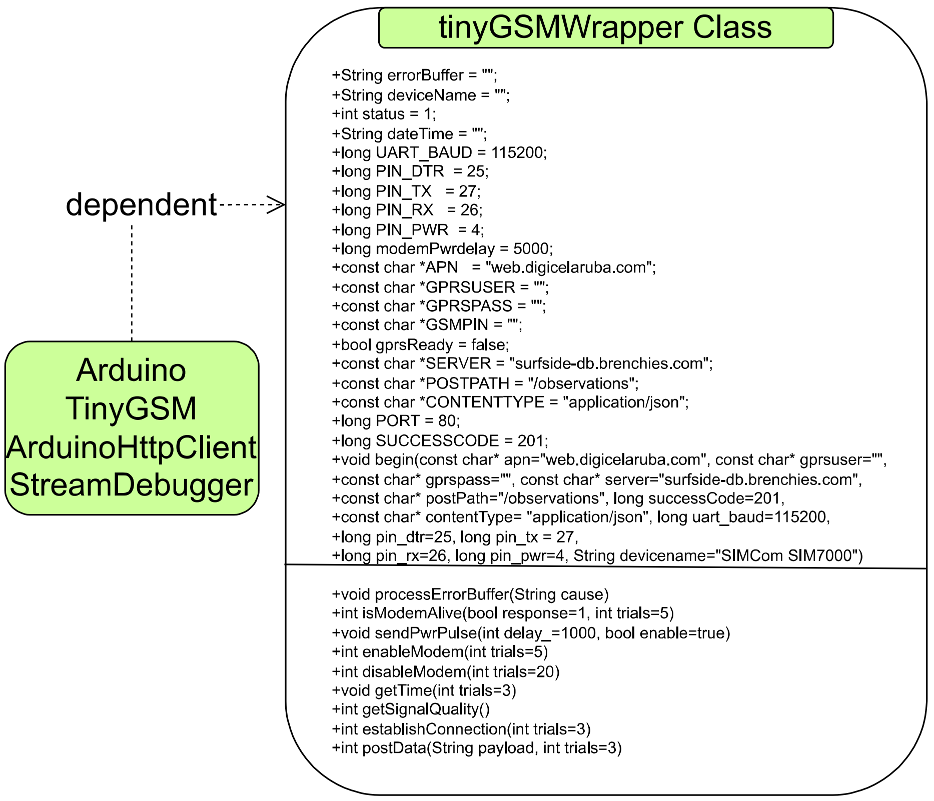

surfSideScience
Wakeup
instantiate sensors, communication, surfsideScience & sdlogger objects
enable watchdog timer 10 minutes
ENABLE Sensors & wait for sensor equilibrium state
Sample sensors
disable sensors
enable modem & establish connection
get modem time
post data
disable modem
write samples to sd
dump error log to sd
disable watchdog timer 10 minutes
sleep 60 minutes
error
log step error
continue with current step

DATA PAYLOAD JSON FORMAT

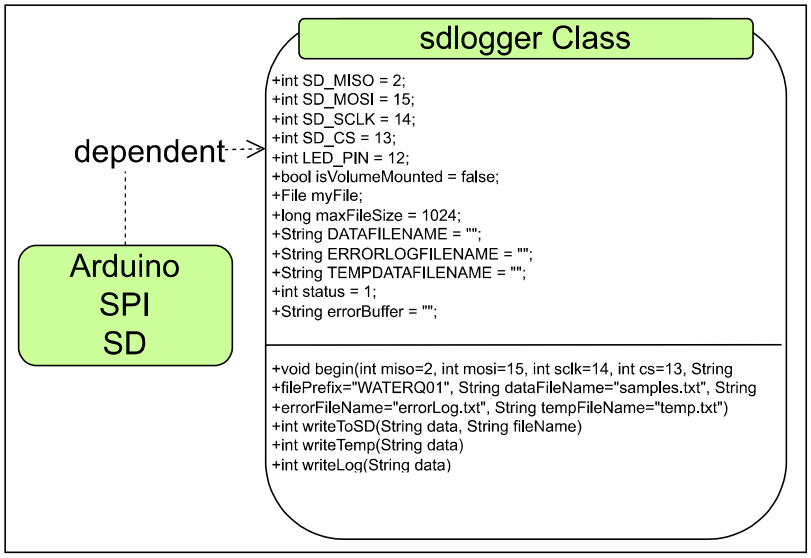
```
{"deviceName": "string", "timestamp": "dateTime",
 "sesnors": [
                  {"sensorName": "string", "value": float, "unit": "string"}
                  {...},
                  {"sensorName": "string", "value": float, "unit": "string"}
              ]
}
```

ERROR PAYLOAD JSON FORMAT

```
{"deviceName": "string", "timestamp": "dateTime",
 "errors": [
            {"sensorName": "string", "error": "String"},
            {...},
            {"sensorName": "string", "error": "String"},
      ]
}
```

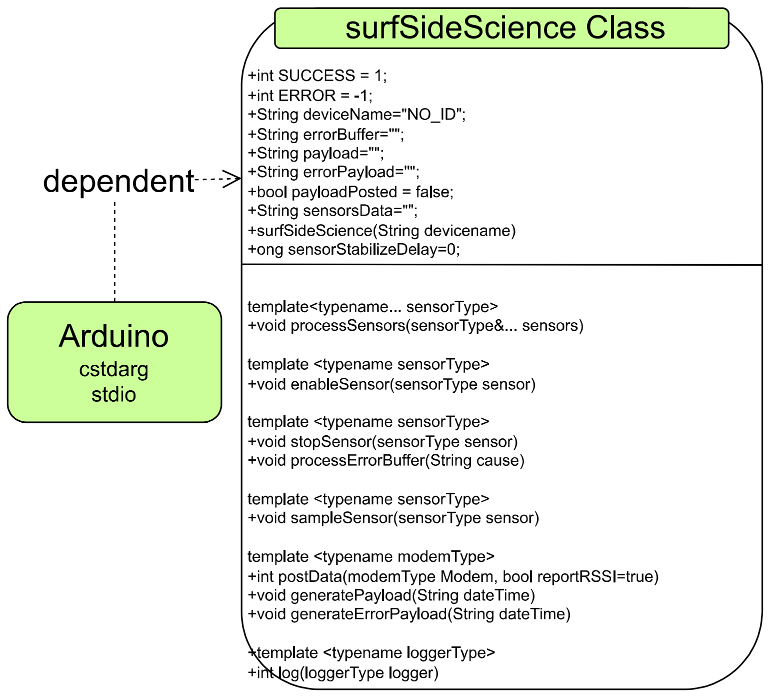


```
+String errorBuffer = "";
+String deviceName = "";
+int status = 1;
+String dateTime = "";
+long UART_BAUD = 115200;
+long PIN_DTR  = 25;
+long PIN_TX   = 27;
+long PIN_RX   = 26;
+long PIN_PWR  = 4;
+long modemPwrdelay = 5000;
+const char *APN   = "web.digicelaruba.com";
+const char *GPRSUSER = "";
+const char *GPRSPASS = "";
+const char *GSMPIN = "";
+bool gprsReady = false;
+const char *SERVER = "surfside-db.brenchies.com";
+const char *POSTPATH = "/observations";
+const char *CONTENTTYPE = "application/json";
+long PORT = 80;
+long SUCCESSCODE = 201;
+void begin(const char* apn="web.digicelaruba.com", const char* gprsuser="",
+const char* gprspass="", const char* server="surfside-db.brenchies.com",
+const char* postPath="/observations", long successCode=201,
+const char* contentType= "application/json", long uart_baud=115200,
+long pin_dtr=25, long pin_tx = 27,
+long pin_rx=26, long pin_pwr=4, String devicename="SIMCom SIM7000")

+void processErrorBuffer(String cause)
+int isModemAlive(bool response=1, int trials=5)
+void sendPwrPulse(int delay_=1000, bool enable=true)
+int enableModem(int trials=5)
+int disableModem(int trials=20)
+void getTime(int trials=3)
+int getSignalQuality()
+int establishConnection(int trials=3)
+int postData(String payload, int trials=3)
```



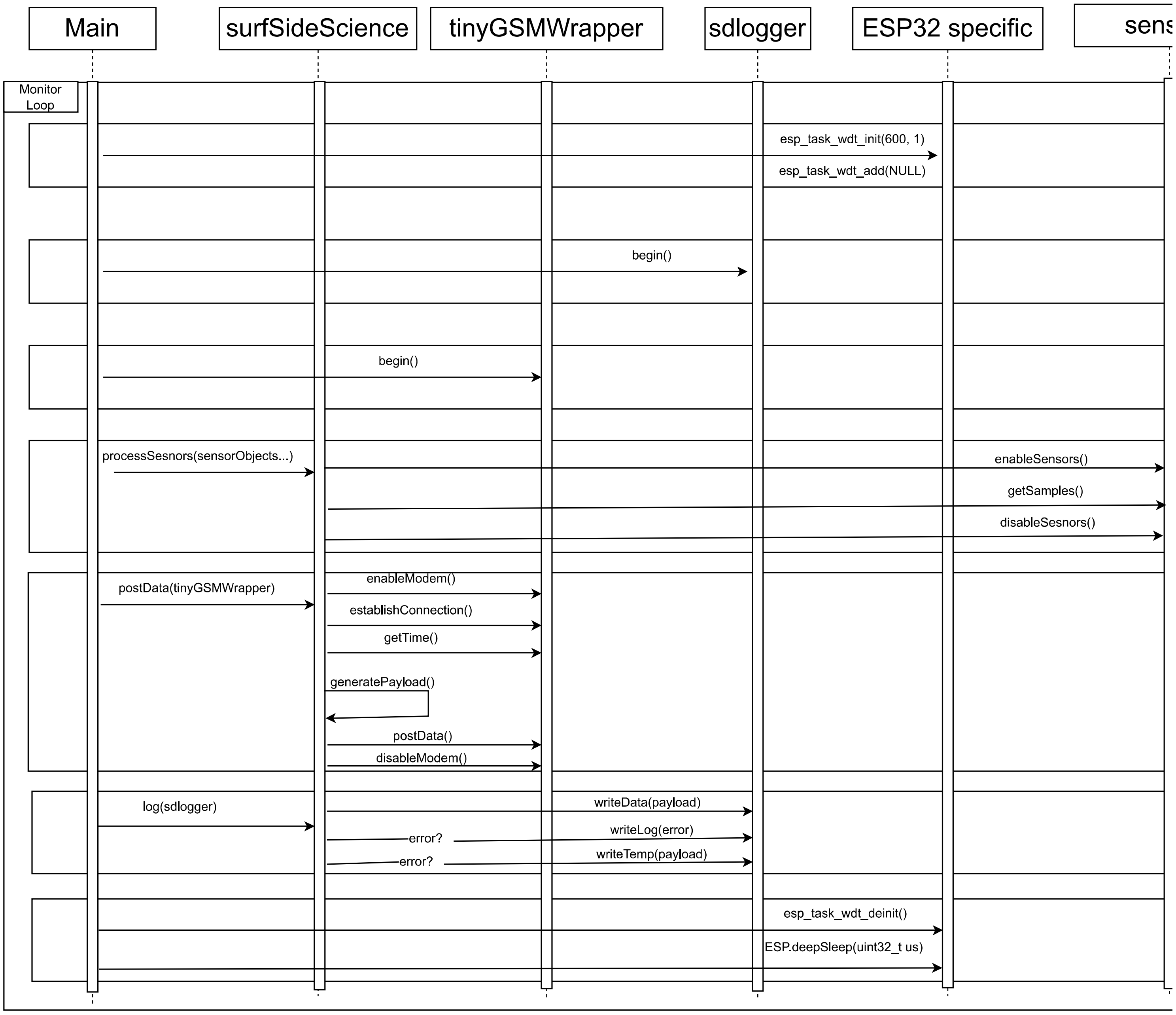
```
+int SD_MISO = 2;
+int SD_MOSI = 15;
+int SD_SCLK = 14;
+int SD_CS = 13;
+int LED_PIN = 12;
+bool isVolumeMounted = false;
+File myFile;
+long maxFileSize = 1024;
+String DATAFILENAME = "";
+String ERRORLOGFILENAME = "";
+String TEMPDATAFILENAME = "";
+int status = 1;
+String errorBuffer = "";
```

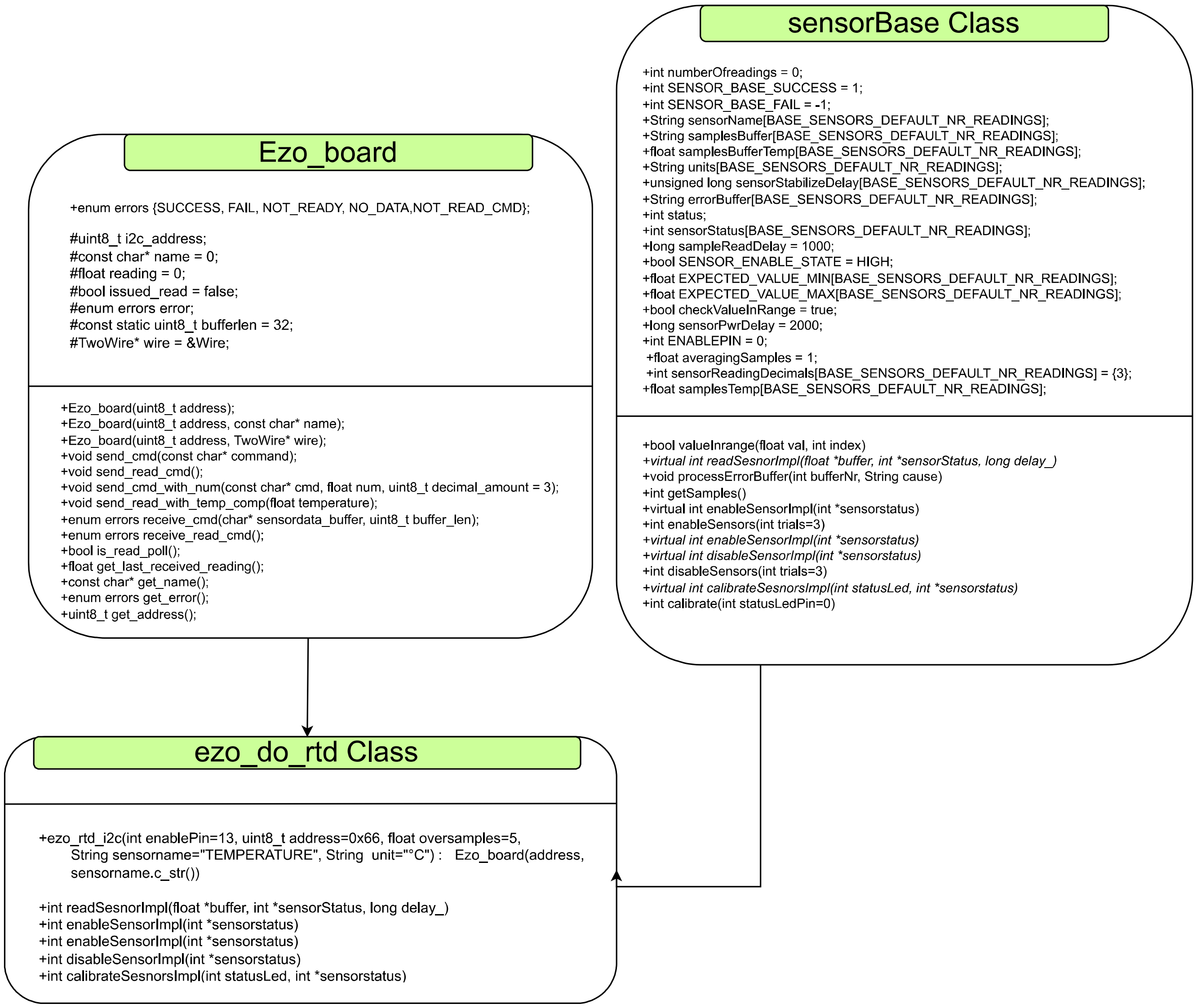
```
+void begin(int miso=2, int mosi=15, int sclk=14, int cs=13, String
+filePrefix="WATERQ01", String dataFileName="samples.txt", String
+errorFileName="errorLog.txt", String tempFileName="temp.txt")
+int writeToSD(String data, String fileName)
+int writeTemp(String data)
+int writeLog(String data)
```



```
+int SUCCESS = 1;
+int ERROR = -1;
+String deviceName="NO_ID";
+String errorBuffer="";
+String payload="";
+String errorPayload="";
+bool payloadPosted = false;
+String sensorsData="";
+surfSideScience(String devicename)
+ong sensorStabilizeDelay=0;

template<typename... sensorType>
+void processSensors(sensorType&... sensors)

template <typename sensorType>
+void enableSensor(sensorType sensor)

template <typename sensorType>
+void stopSensor(sensorType sensor)
+void processErrorBuffer(String cause)

template <typename sensorType>
+void sampleSensor(sensorType sensor)

template <typename modemType>
+int postData(modemType Modem, bool reportRSSI=true)
+void generatePayload(String dateTime)
+void generateErrorPayload(String dateTime)

+template <typename loggerType>
+int log(loggerType logger)
```

Sequence diagram

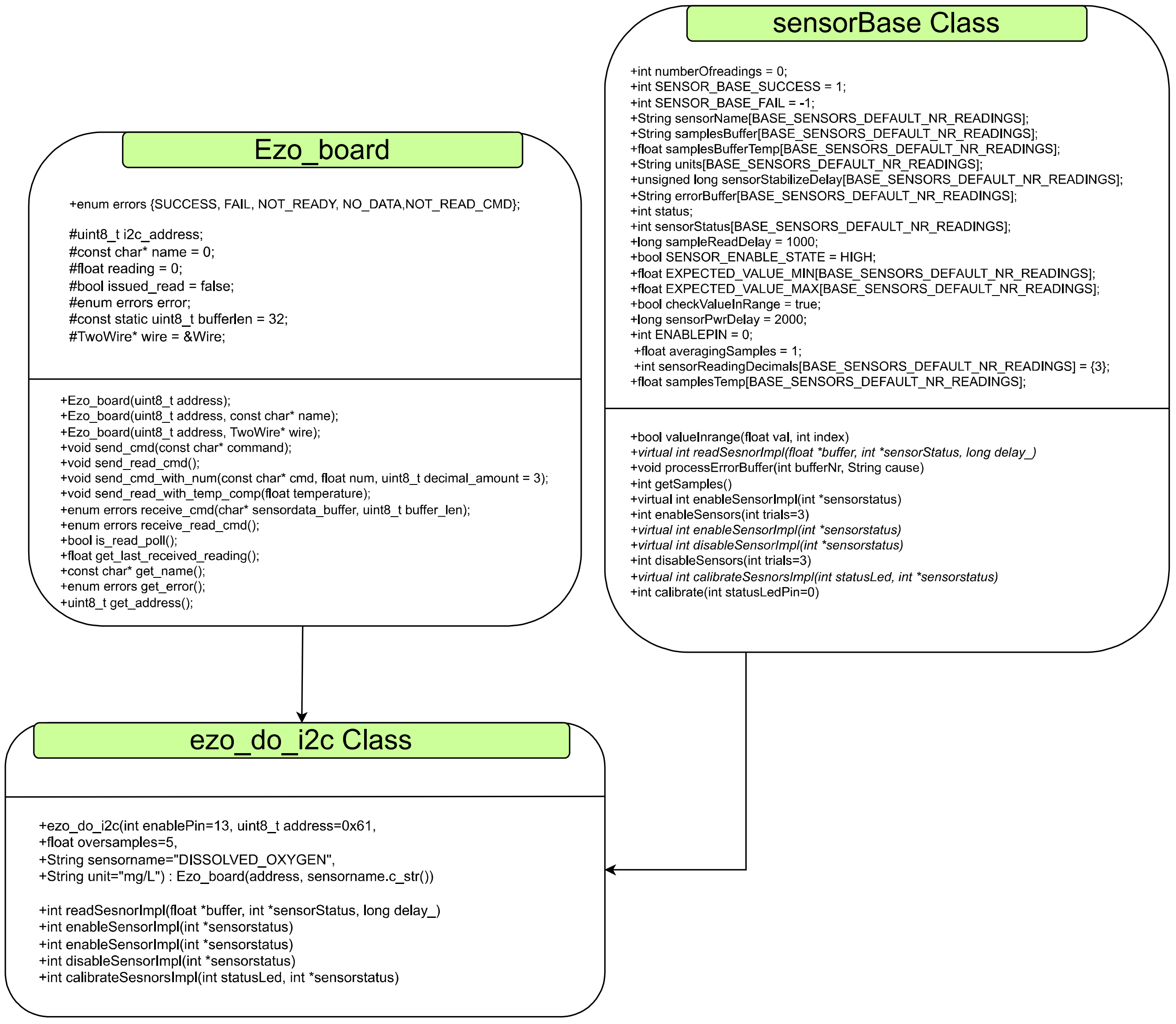




```
Ezo_board

+enum errors {SUCCESS, FAIL, NOT_READY, NO_DATA,NOT_READ_CMD};

#uint8_t i2c_address;
#const char* name = 0;
#float reading = 0;
#bool issued_read = false;
#enum errors error;
#const static uint8_t bufferlen = 32;
#TwoWire* wire = &Wire;

+Ezo_board(uint8_t address);
+Ezo_board(uint8_t address, const char* name);
+Ezo_board(uint8_t address, TwoWire* wire);
+void send_cmd(const char* command);
+void send_read_cmd();
+void send_cmd_with_num(const char* cmd, float num, uint8_t decimal_amount = 3);
+void send_read_with_temp_comp(float temperature);
+enum errors receive_cmd(char* sensordata_buffer, uint8_t buffer_len);
+enum errors receive_read_cmd();
+bool is_read_poll();
+float get_last_received_reading();
+const char* get_name();
+enum errors get_error();
+uint8_t get_address();
```

```
sensorBase Class

+int numberOfreadings = 0;
+int SENSOR_BASE_SUCCESS = 1;
+int SENSOR_BASE_FAIL = -1;
+String sensorName[BASE_SENSORS_DEFAULT_NR_READINGS];
+String samplesBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+float samplesBufferTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
+String units[BASE_SENSORS_DEFAULT_NR_READINGS];
+unsigned long sensorStabilizeDelay[BASE_SENSORS_DEFAULT_NR_READINGS];
+String errorBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+int status;
+int sensorStatus[BASE_SENSORS_DEFAULT_NR_READINGS];
+long sampleReadDelay = 1000;
+bool SENSOR_ENABLE_STATE = HIGH;
+float EXPECTED_VALUE_MIN[BASE_SENSORS_DEFAULT_NR_READINGS];
+float EXPECTED_VALUE_MAX[BASE_SENSORS_DEFAULT_NR_READINGS];
+bool checkValueInRange = true;
+long sensorPwrDelay = 2000;
+int ENABLEPIN = 0;
 +float averagingSamples = 1;
 +int sensorReadingDecimals[BASE_SENSORS_DEFAULT_NR_READINGS] = {3};
+float samplesTemp[BASE_SENSORS_DEFAULT_NR_READINGS];

+bool valueInrange(float val, int index)
+virtual int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+void processErrorBuffer(int bufferNr, String cause)
+int getSamples()
+virtual int enableSensorImpl(int *sensorstatus)
+int enableSensors(int trials=3)
+virtual int enableSensorImpl(int *sensorstatus)
+virtual int disableSensorImpl(int *sensorstatus)
+int disableSensors(int trials=3)
+virtual int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
+int calibrate(int statusLedPin=0)
```

```
ezo_do_rtd Class

+ezo_rtd_i2c(int enablePin=13, uint8_t address=0x66, float oversamples=5,
      String sensorname="TEMPERATURE", String  unit="°C") :   Ezo_board(address,
      sensorname.c_str())

+int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+int enableSensorImpl(int *sensorstatus)
+int enableSensorImpl(int *sensorstatus)
+int disableSensorImpl(int *sensorstatus)
+int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
```



Ezo_board

```
+enum errors {SUCCESS, FAIL, NOT_READY, NO_DATA,NOT_READ_CMD};

#uint8_t i2c_address;
#const char* name = 0;
#float reading = 0;
#bool issued_read = false;
#enum errors error;
#const static uint8_t bufferlen = 32;
#TwoWire* wire = &Wire;
```

```
+Ezo_board(uint8_t address);
+Ezo_board(uint8_t address, const char* name);
+Ezo_board(uint8_t address, TwoWire* wire);
+void send_cmd(const char* command);
+void send_read_cmd();
+void send_cmd_with_num(const char* cmd, float num, uint8_t decimal_amount = 3);
+void send_read_with_temp_comp(float temperature);
+enum errors receive_cmd(char* sensordata_buffer, uint8_t buffer_len);
+enum errors receive_read_cmd();
+bool is_read_poll();
+float get_last_received_reading();
+const char* get_name();
+enum errors get_error();
+uint8_t get_address();
```

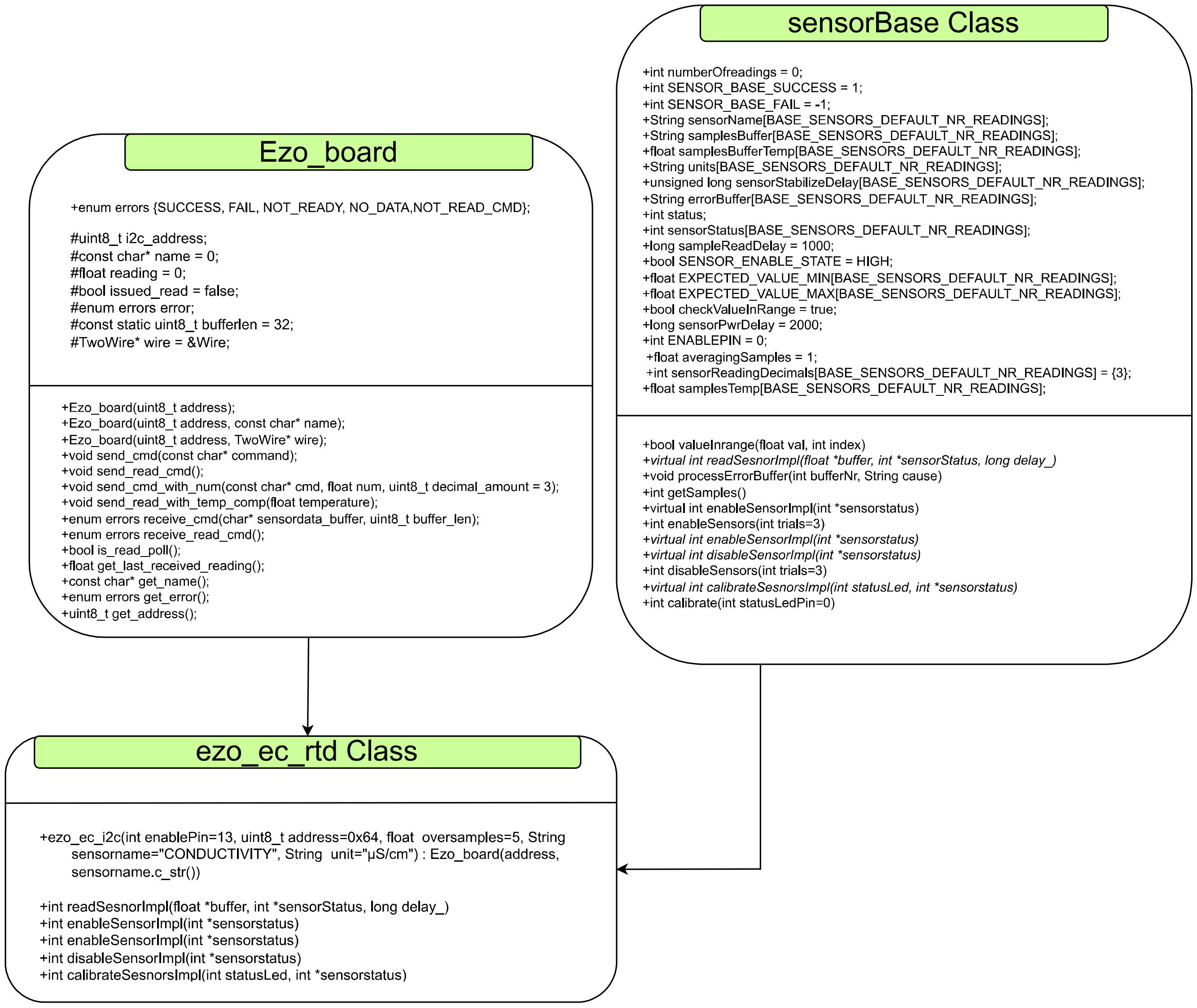
sensorBase Class

```
+int numberOfreadings = 0;
+int SENSOR_BASE_SUCCESS = 1;
+int SENSOR_BASE_FAIL = -1;
+String sensorName[BASE_SENSORS_DEFAULT_NR_READINGS];
+String samplesBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+float samplesBufferTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
+String units[BASE_SENSORS_DEFAULT_NR_READINGS];
+unsigned long sensorStabilizeDelay[BASE_SENSORS_DEFAULT_NR_READINGS];
+String errorBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+int status;
+int sensorStatus[BASE_SENSORS_DEFAULT_NR_READINGS];
+long sampleReadDelay = 1000;
+bool SENSOR_ENABLE_STATE = HIGH;
+float EXPECTED_VALUE_MIN[BASE_SENSORS_DEFAULT_NR_READINGS];
+float EXPECTED_VALUE_MAX[BASE_SENSORS_DEFAULT_NR_READINGS];
+bool checkValueInRange = true;
+long sensorPwrDelay = 2000;
+int ENABLEPIN = 0;
 +float averagingSamples = 1;
 +int sensorReadingDecimals[BASE_SENSORS_DEFAULT_NR_READINGS] = {3};
+float samplesTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
```

```
+bool valueInrange(float val, int index)
+virtual int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+void processErrorBuffer(int bufferNr, String cause)
+int getSamples()
+virtual int enableSensorImpl(int *sensorstatus)
+int enableSensors(int trials=3)
+virtual int enableSensorImpl(int *sensorstatus)
+virtual int disableSensorImpl(int *sensorstatus)
+int disableSensors(int trials=3)
+virtual int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
+int calibrate(int statusLedPin=0)
```

ezo_do_i2c Class

```
+ezo_do_i2c(int enablePin=13, uint8_t address=0x61,
+float oversamples=5,
+String sensorname="DISSOLVED_OXYGEN",
+String unit="mg/L") : Ezo_board(address, sensorname.c_str())

+int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+int enableSensorImpl(int *sensorstatus)
+int enableSensorImpl(int *sensorstatus)
+int disableSensorImpl(int *sensorstatus)
+int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
```



sensorBase Class

```
+int numberOfreadings = 0;
+int SENSOR_BASE_SUCCESS = 1;
+int SENSOR_BASE_FAIL = -1;
+String sensorName[BASE_SENSORS_DEFAULT_NR_READINGS];
+String samplesBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+float samplesBufferTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
+String units[BASE_SENSORS_DEFAULT_NR_READINGS];
+unsigned long sensorStabilizeDelay[BASE_SENSORS_DEFAULT_NR_READINGS];
+String errorBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+int status;
+int sensorStatus[BASE_SENSORS_DEFAULT_NR_READINGS];
+long sampleReadDelay = 1000;
+bool SENSOR_ENABLE_STATE = HIGH;
+float EXPECTED_VALUE_MIN[BASE_SENSORS_DEFAULT_NR_READINGS];
+float EXPECTED_VALUE_MAX[BASE_SENSORS_DEFAULT_NR_READINGS];
+bool checkValueInRange = true;
+long sensorPwrDelay = 2000;
+int ENABLEPIN = 0;
 +float averagingSamples = 1;
 +int sensorReadingDecimals[BASE_SENSORS_DEFAULT_NR_READINGS] = {3};
+float samplesTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
```

```
+bool valueInrange(float val, int index)
+virtual int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+void processErrorBuffer(int bufferNr, String cause)
+int getSamples()
+virtual int enableSensorImpl(int *sensorstatus)
+int enableSensors(int trials=3)
+virtual int enableSensorImpl(int *sensorstatus)
+virtual int disableSensorImpl(int *sensorstatus)
+int disableSensors(int trials=3)
+virtual int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
+int calibrate(int statusLedPin=0)
```

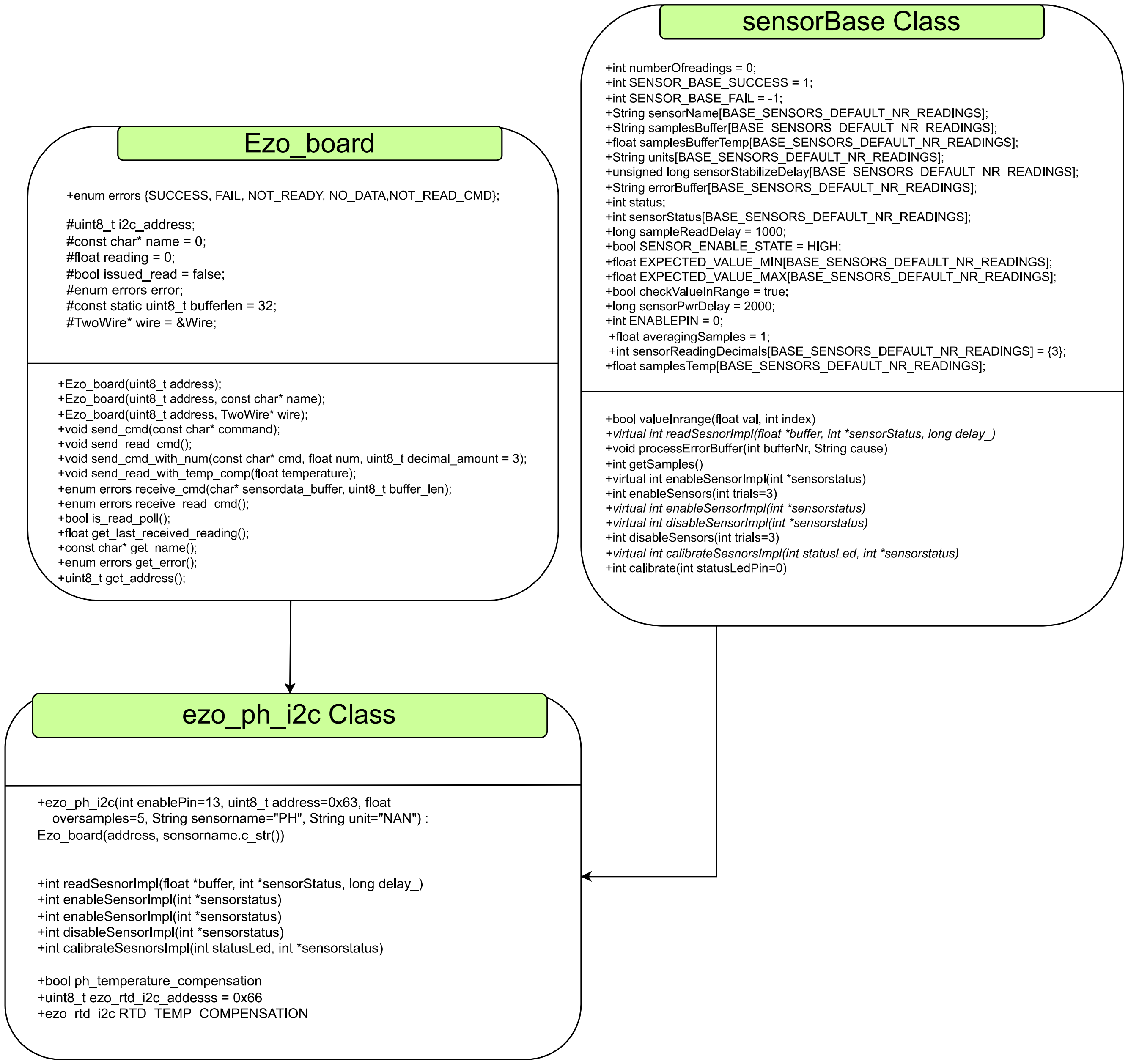
Ezo_board

```
+enum errors {SUCCESS, FAIL, NOT_READY, NO_DATA,NOT_READ_CMD};

#uint8_t i2c_address;
#const char* name = 0;
#float reading = 0;
#bool issued_read = false;
#enum errors error;
#const static uint8_t bufferlen = 32;
#TwoWire* wire = &Wire;
```

```
+Ezo_board(uint8_t address);
+Ezo_board(uint8_t address, const char* name);
+Ezo_board(uint8_t address, TwoWire* wire);
+void send_cmd(const char* command);
+void send_read_cmd();
+void send_cmd_with_num(const char* cmd, float num, uint8_t decimal_amount = 3);
+void send_read_with_temp_comp(float temperature);
+enum errors receive_cmd(char* sensordata_buffer, uint8_t buffer_len);
+enum errors receive_read_cmd();
+bool is_read_poll();
+float get_last_received_reading();
+const char* get_name();
+enum errors get_error();
+uint8_t get_address();
```

ezo_ec_rtd Class

```
+ezo_ec_i2c(int enablePin=13, uint8_t address=0x64, float  oversamples=5, String
      sensorname="CONDUCTIVITY", String  unit="µS/cm") : Ezo_board(address,
      sensorname.c_str())

+int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+int enableSensorImpl(int *sensorstatus)
+int enableSensorImpl(int *sensorstatus)
+int disableSensorImpl(int *sensorstatus)
+int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
```



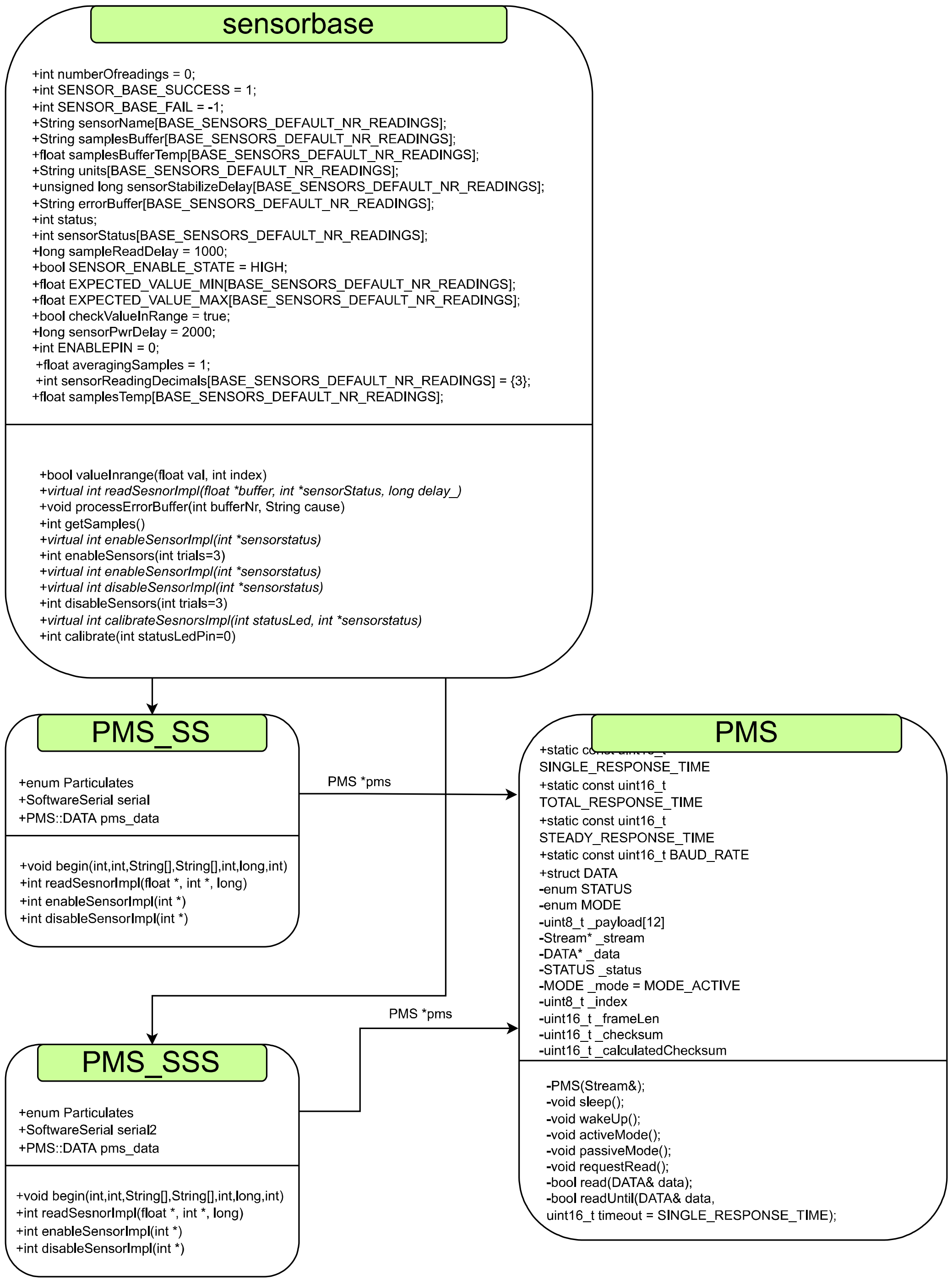
```
+enum errors {SUCCESS, FAIL, NOT_READY, NO_DATA,NOT_READ_CMD};

#uint8_t i2c_address;
#const char* name = 0;
#float reading = 0;
#bool issued_read = false;
#enum errors error;
#const static uint8_t bufferlen = 32;
#TwoWire* wire = &Wire;
```

```
+Ezo_board(uint8_t address);
+Ezo_board(uint8_t address, const char* name);
+Ezo_board(uint8_t address, TwoWire* wire);
+void send_cmd(const char* command);
+void send_read_cmd();
+void send_cmd_with_num(const char* cmd, float num, uint8_t decimal_amount = 3);
+void send_read_with_temp_comp(float temperature);
+enum errors receive_cmd(char* sensordata_buffer, uint8_t buffer_len);
+enum errors receive_read_cmd();
+bool is_read_poll();
+float get_last_received_reading();
+const char* get_name();
+enum errors get_error();
+uint8_t get_address();
```

```
+int numberOfreadings = 0;
+int SENSOR_BASE_SUCCESS = 1;
+int SENSOR_BASE_FAIL = -1;
+String sensorName[BASE_SENSORS_DEFAULT_NR_READINGS];
+String samplesBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+float samplesBufferTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
+String units[BASE_SENSORS_DEFAULT_NR_READINGS];
+unsigned long sensorStabilizeDelay[BASE_SENSORS_DEFAULT_NR_READINGS];
+String errorBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+int status;
+int sensorStatus[BASE_SENSORS_DEFAULT_NR_READINGS];
+long sampleReadDelay = 1000;
+bool SENSOR_ENABLE_STATE = HIGH;
+float EXPECTED_VALUE_MIN[BASE_SENSORS_DEFAULT_NR_READINGS];
+float EXPECTED_VALUE_MAX[BASE_SENSORS_DEFAULT_NR_READINGS];
+bool checkValueInRange = true;
+long sensorPwrDelay = 2000;
+int ENABLEPIN = 0;
 +float averagingSamples = 1;
 +int sensorReadingDecimals[BASE_SENSORS_DEFAULT_NR_READINGS] = {3};
+float samplesTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
```

```
+bool valueInrange(float val, int index)
+virtual int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+void processErrorBuffer(int bufferNr, String cause)
+int getSamples()
+virtual int enableSensorImpl(int *sensorstatus)
+int enableSensors(int trials=3)
+virtual int enableSensorImpl(int *sensorstatus)
+virtual int disableSensorImpl(int *sensorstatus)
+int disableSensors(int trials=3)
+virtual int calibrateSesnorsImpl(int statusLed, int *sensorstatus)
+int calibrate(int statusLedPin=0)
```

```
+ezo_ph_i2c(int enablePin=13, uint8_t address=0x63, float
   oversamples=5, String sensorname="PH", String unit="NAN") :
Ezo_board(address, sensorname.c_str())


+int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)
+int enableSensorImpl(int *sensorstatus)
+int enableSensorImpl(int *sensorstatus)
+int disableSensorImpl(int *sensorstatus)
+int calibrateSesnorsImpl(int statusLed, int *sensorstatus)

+bool ph_temperature_compensation
+uint8_t ezo_rtd_i2c_addesss = 0x66
+ezo_rtd_i2c RTD_TEMP_COMPENSATION
```


sensorbase
+int numberOfreadings = 0;
+int SENSOR_BASE_SUCCESS = 1;
+int SENSOR_BASE_FAIL = -1;
+String sensorName[BASE_SENSORS_DEFAULT_NR_READINGS];
+String samplesBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+float samplesBufferTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
+String units[BASE_SENSORS_DEFAULT_NR_READINGS];
+unsigned long sensorStabilizeDelay[BASE_SENSORS_DEFAULT_NR_READINGS];
+String errorBuffer[BASE_SENSORS_DEFAULT_NR_READINGS];
+int status;
+int sensorStatus[BASE_SENSORS_DEFAULT_NR_READINGS];
+long sampleReadDelay = 1000;
+bool SENSOR_ENABLE_STATE = HIGH;
+float EXPECTED_VALUE_MIN[BASE_SENSORS_DEFAULT_NR_READINGS];
+float EXPECTED_VALUE_MAX[BASE_SENSORS_DEFAULT_NR_READINGS];
+bool checkValueInRange = true;
+long sensorPwrDelay = 2000;
+int ENABLEPIN = 0;
+float averagingSamples = 1;
+int sensorReadingDecimals[BASE_SENSORS_DEFAULT_NR_READINGS] = {3};
+float samplesTemp[BASE_SENSORS_DEFAULT_NR_READINGS];
+bool valueInrange(float val, int index)
+*virtual int readSesnorImpl(float *buffer, int *sensorStatus, long delay_)*
+void processErrorBuffer(int bufferNr, String cause)
+int getSamples()
+*virtual int enableSensorImpl(int *sensorstatus)*
+int enableSensors(int trials=3)
+*virtual int enableSensorImpl(int *sensorstatus)*
+*virtual int disableSensorImpl(int *sensorstatus)*
+int disableSensors(int trials=3)
+*virtual int calibrateSesnorsImpl(int statusLed, int *sensorstatus)*
+int calibrate(int statusLedPin=0)
PMS_SS
+enum Particulates
+SoftwareSerial serial
+PMS::DATA pms_data
+void begin(int,int,String[],String[],int,long,int)
+int readSesnorImpl(float *, int *, long)
+int enableSensorImpl(int *)
+int disableSensorImpl(int *)
PMS *pms
PMS_SSS
+enum Particulates
+SoftwareSerial serial2
+PMS::DATA pms_data
+void begin(int,int,String[],String[],int,long,int)
+int readSesnorImpl(float *, int *, long)
+int enableSensorImpl(int *)
+int disableSensorImpl(int *)
PMS *pms
PMS
+static const uint16_t
SINGLE_RESPONSE_TIME
+static const uint16_t
TOTAL_RESPONSE_TIME
+static const uint16_t
STEADY_RESPONSE_TIME
+static const uint16_t BAUD_RATE
+struct DATA
-enum STATUS
-enum MODE
-uint8_t _payload[12]
-Stream* _stream
-DATA* _data
-STATUS _status
-MODE _mode = MODE_ACTIVE
-uint8_t _index
-uint16_t _frameLen
-uint16_t _checksum
-uint16_t _calculatedChecksum
-PMS(Stream&);
-void sleep();
-void wakeUp();
-void activeMode();
-void passiveMode();
-void requestRead();
-bool read(DATA& data);
-bool readUntil(DATA& data,
uint16_t timeout = SINGLE_RESPONSE_TIME);