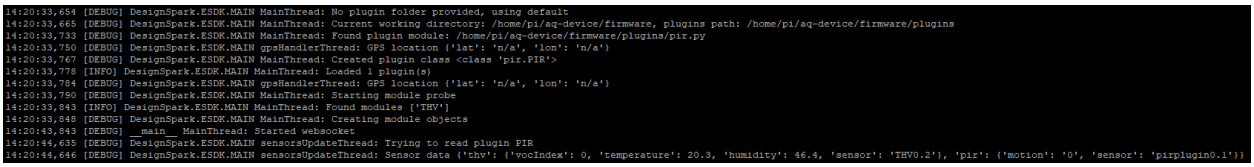
DesignSpark.ESDK

DesignSpark

Nov 21, 2022

CONTENTS



DesignSpark Python library to support the Environmental Sensing Development Kit hardware.

Features include:

- Simple interface to supported ESDK modules
- Support for plugins to extend the ecosystem with new sensors
- Control over mainboard features including buzzer, sensor power rails and GPS

CHAPTER

ONE

SUPPORTED MODULES

Currently supported sensor modules include:

- ESDK-THV (temperature, humidity and VOC)
- ESDH-CO2 (CO2)
- ESDK-PM2 (PM1.0, PM2.5, PM4.0 and PM10 particulates)
- ESDK-NO2 (NO2)
- ESDK-NRD (nuclear radiation)
- ESDK-FDH (formaldehyde)

CHAPTER

TWO

INSTALLATION

DesignSpark.ESDK can be installed from PyPI using the command-line tool *pip*. See the documentation for more information.

CHAPTER

THREE

DOCUMENTATION

Installation, API documentation and examples are available at:

https://docs.designspark.io/projects/esdk-library/en/latest/index.html

3.1 Writing Plugins

Plugins are currently an experimental feature, and only support for reading sensors has been implemented.

A plugin is a Python module that expands the ESDK ecosystem to support additional sensors.

3.1.1 Requirements

At the bare minimum, a plugin contains a class with a function called `readSensors` that returns a dictionary containing the sensor data. Should no data be available, a value of -1 should be returned in place of the dictionary.

The sensor data dictionary **should** contain certain data:

```
{
        "sensorname": {
                "sensor": "sensorversion"
        }
}
```

The `sensorname` key should be a descriptive, short name for the sensor. For example, the ESDK sensor boards are `thv`, `co2` and `pm2`.

The `sensorversion` value should be a string containing a revision of the sensor board or plugin module. For example, the ESDK sensor boards are `THV0.2`, `CO20.2` and `PM20.2` (board name plus a hardware version number).

Sensor data can be inserted into the innermost object, alongside the `sensor` key. For example, the sensor data structure for a PIR sensor plugged into the ESDK-EEA board would look like the following:

```
{
        "pir": {
                "motion": "1",
                "sensor": "pirplugin0.1"
        }
}
```

3.1.2 Example

A simple example that reads a GPIO pin is provided below:

```
import RPi.GPIO as GPIO
GPIO1 = 20

class PIR:
def __init__(self):
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(GPIO1, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

def readSensors(self):
        pirState = GPIO.input(GPIO1)
        return {"pir": {"motion": str(pirState), "sensor": "pirplugin0.1"}}
```

3.1.3 Usage

By default, `ModMAIN` looks for a `plugins` directory in the current working directory of the Python script that has instantiated the class, but this can be overridden by providing the `pluginDir` argument when instantiating the class.

```
mb = MAIN.ModMAIN(config=config, debug=True, loggingLevel='off', pluginDir="/home/pi/
↪plugins")
```

Plugins are loaded using the `loadPlugins` function, which attempts to import each module found in the specified directory.

Calling the function `readAllModules` attempts to read all the default sensor modules (i.e. THV, CO2 and PM2), and then attempts to read the plugin sensors. The function then returns a dictionary containing all the available sensor data.

3.1.4 Exceptions

Exceptions are handled internally within the ModMAIN class, and errors are logged where necessary. If a plugin isn't working, check for errors in the logs to pinpoint where the issue lies.

3.2 API

3.2.1 DesignSpark.ESDK.MAIN

ESDK Main board interface

class DesignSpark.ESDK.MAIN.**ModMAIN**(*config*, *debug=False*, *loggingLevel='full'*, *pluginDir=None*)

This class handles the ESDK mainboard, and it's various features.

Parameters **config** – A dictionary containing configuration data with a minimum of:

```
{
    "esdk":{
        "GPS":False
    }
}
```

Parameters

- **debug** (*bool, optional*) – Debug logging control, defaults to False
- **loggingLevel** (*str, optional*) – One of 'off', 'error' or 'full' to control file logging, defaults to 'full'
- **pluginDir** (*str, optional*) – A string value containing a file path to a plugin directory, defaults to None

`createModules()`
Discovers and instantiates module objects for use with `readAllModules()`.

`getGPSStatus()`
Returns a dictionary containing GPS status.

Returns A dictionary containing:

```
{
    "gpsStatus":{
        "mode":0,
        "satellites":13,
        "satellitesUsed":5
    }
}
```

Return type dict

`getLocation()`
Returns a dictionary containing GPS location, or configuration file location if GPS is disabled.

Returns A dictionary containing:

```
{
    "lat":0.0,
    "lon":0.0
}
```

Return type dict

`getModuleVersion()`
Returns a dictionary containing ESDK module version.

Returns A dictionary containing:

```
{
    "moduleVerson":"0.0.1"
}
```

Return type dict

`getSerialNumber()`
Returns a dictionary containing the Raspberry Pi serial number.

Returns A dictionary containing:

```
{
    "serialNumber":"RPI0123456789"
}
```

Return type dict

getUndervoltageStatus()
Returns a dictionary containing the Raspberry Pi throttle status and code.

Returns A dictionary containing (throttle_state is optional, and only populated should a nonzero code exist)

```
{
    "throttle_state":{
        "code":0,
        "throttle_state":""
    }
}
```

Return type dict

loadPlugins()
Attempts to load and instantiate plugins from a specified folder.

readAllModules()
Reads all sensor modules and returns a dictionary containing sensor data.

setBuzzer(*freq=0*)
Sets a PWM frequency on the buzzer output.

Parameters **freq** (*int, optional*) – Buzzer frequency, 0 stops the buzzer

setPower(*vcc3=False*, *vcc5=False*)
Switches 3.3V and 5V sensor power supply rails according to supplied arguments.

Parameters

- **vcc3** (*bool, optional*) – 3.3V sensor power supply status, defaults to False
- **vcc5** (*bool, optional*) – 5V sensor power supply status, defaults to False

3.2.2 DesignSpark.ESDK.THV

ESDK THV board interface

class DesignSpark.ESDK.THV.**ModTHV**
This is a class that handles interfacing with the ESDK-THV board.

readSensors()
Reads sensors and returns a dictionary containing module version, and all readings.

Returns A dictionary containing

```
{
    "thv":{
        "sensor":"THV0.2",
        "temperature":21.2,
```

(continues on next page)

(continued from previous page)

```
        "humidity":50.3,
        "vocIndex":100
    }
}
```

Or -1 if data is unavailable

Return type dict, int

readTempAndHumidity()

Queries SHT31 and returns a dictionary of temperature and humidity values.

Returns A dictionary containing

```
{
    "temperature":12.3,
    "humidity":50.3
}
```

Return type dict

readVocIndex()

Returns a calculated VOC index value.

Returns An integer VOC index value, or -1 if unavailable

Return type int

readVocRaw()

Returns a compensated raw VOC value.

Returns An integer VOC value

Return type int

3.2.3 DesignSpark.ESDK.CO2

ESDK CO2 board interface

class DesignSpark.ESDK.CO2.**ModCO2**

This is a class that handles interfacing with the ESDK-CO2 board.

readCO2()

Reads a CO2 value from the sensor

Returns A CO2 reading in ppm, or -1 if the sensor is not ready

Return type int

readSensors()

Reads sensors and returns a dictionary containing module version, and all readings.

Returns A dictionary containing

```
{
    "co2":{
        "sensor":"CO20.2",
        "co2":453
```

(continues on next page)

(continued from previous page)

```
    }
}
```

Or -1 if data is unavailable

Return type dict, int

readTempAndHumidity()
Reads temperature and humidity from the sensor

Returns A dictionary containing:

```
{
    "temp":12.3,
    "humidity":50.3
}
```

Or -1 if sensor data is unavailable

Return type dict, int

3.2.4 DesignSpark.ESDK.PM2

ESDK PM2 board interface

class DesignSpark.ESDK.PM2.**ModPM2**
This is a class that handles interfacing with the ESDK-PM2 board.

readSensors()
Reads sensors and returns a dictionary containing module version, and all readings.

Returns A dictionary containing

```
{
    "pm2":{
        "sensor":"PM20.2",
        "pm1.0":0,
        "pm2.5":0,
        "pm4.0":0,
        "pm10":0
    }
}
```

Or -1 if data is unavailable

Return type dict, int

startFanCleaning()
Starts fan cleaning procedure.

startMeasurement()
Starts measurement, configures readings to be unsigned 16-bit integers.

3.2.5 DesignSpark.ESDK.NO2

ESDK NO2 board interface

class DesignSpark.ESDK.NO2.**ModNO2**(*sensitivity=- 20.86*, *tia_gain=499*, *voffset=0*)
This is a class that handles interfacing with the ESDK-PM2 board.

Parameters

- **sensitivity** (*float*) – Sensitivity code from barcode on sensor
- **tia_gain** (*float*) – Transimpedance amplifier gain from sensor datasheet
- **voffset** (*float*) – Offset voltage used in gas calculation

readNO2()
Read ADC and calculate an NO2 reading.

Returns The NO2 concentration.

Return type float

readSensors()
Reads sensor and returns a dictionary containing module version, and all readings.

Returns A dictionary containing

```
{
    "no2":{
        "sensor":"NO20.1",
        "no2":2.1,
    }
}
```

Or -1 if data is unavailable

Return type dict, int

3.2.6 DesignSpark.ESDK.NRD

ESDK NRD board interface

class DesignSpark.ESDK.NRD.**ModNRD**
This is a class that handles interfacing with the ESDK-NRD board.

disableEventGpio()
Disable the event detection GPIO output.

disableEventLed()
Disable the event detection LED.

disableI2CWatchdog()
Disable NRD I2C watchdog functionality.

enableEventGpio()
Enable the event detection GPIO output.

enableEventLed()
Enable the event detection LED.

enableI2CWatchdog()
Enable NRD I2C watchdog functionality.

getEventGpioEnabledState()
Get the event GPIO enabled state.

Returns A boolean value indicating the event GPIO enabled status

Return type bool

getEventLedEnabledState()
Get the event LED enabled state.

Returns A boolean value indicating the event LED enabled status

Return type bool

getI2CWatchdogEnabledState()
Get the NRD I2C watchdog enabled state.

Returns A boolean value indicating the watchdog enabled status

Return type bool

readCountsPerMinute()
Get the current CPM rate.

Returns An integer representing counts-per-minute

Return type int

readCountsPerSecond()
Get the current CPS rate.

Returns An integer representing counts-per-second

Return type int

readSensors()
Reads sensors and returns a dictionary containing module version and readings.

Returns A dictionary containing

```
{
        "nrd":{
                "sensor":"NRD0.1",
                "cps":23,
                "cpm":483,
                "totalCounts":9465
        }
}
```

Return type dict

readTotalCounts()
Get the total accumulated event count

Returns An integer representing accumulated event counts

Return type int

resetCounts()
Reset all event counters.

3.2.7 DesignSpark.ESDK.FDH

ESDK FDH board interface

class DesignSpark.ESDK.FDH.**ModFDH**
This is a class that handles interfacing with the ESDK-FDH board.

readFormaldehyde()
Reads a formaldehyde value from the sensor

Returns A HCHO reading in ppb

Return type int

readSensors()
Reads sensors and returns a dictionary containing module version and readings.

Returns A dictionary containing

```
{
        "fdh":{
                "sensor":"FDH0.1",
                "formaldehyde":25
        }
}
```

Return type dict

3.2.8 DesignSpark.ESDK.AppLogger

Logging functions

DesignSpark.ESDK.AppLogger.**getErrorLogFileHandler**(*errorFile='/aq/log/error.log'*)
Returns a file logging handler with the log level set to ERROR.

Parameters **errorFile** (*str, optional*) – Log file path (default is /aq/log/error.log).

DesignSpark.ESDK.AppLogger.**getLogFileHandler**(*debug*, *logFile='/aq/log/aq.log'*)
Returns a file logging handler with the log level set to DEBUG or INFO.

Parameters **logFile** (*str, optional*) – Log file path (default is /aq/log/aq.log).

DesignSpark.ESDK.AppLogger.**getLogger**(*name*, *debug=False*, *loggingSetup='full'*)
Returns a logger object.

Parameters

- **debug** (*bool, optional*) – Controls whether the debug log level is enabled (default is False)
- **loggingSetup** (*str, optional*) – One of 'off', 'error' or 'full' log levels (default is 'full')

DesignSpark.ESDK.AppLogger.**getStreamHandler**(*debug*)
Returns a stream handler.

Parameters **debug** (*bool*) – Enables debug logging.

CHAPTER

FOUR

LICENSE

CHAPTER

FIVE

THE MIT LICENSE (MIT)

PYTHON MODULE INDEX

d

INDEX

R

S