# Machinery Vibrodiagnostics with the IIOT

Generated by Doxygen 1.10.0

# 1 Topic Documentation

## 1.1 Accelerometer Data logger

Data flow control from the sensor to the memory card.

### Topics

- Accelerometer

  *Configuration of the accelerometer sensor.*
- Memory card

  *Filesystem operations of the SD card.*
- Button

  *Configuration of the recording button.*
- LED

  *Configuration of the indicator LED light.*

### Macros

- #define **MAX_FILENAME** 256

  *Maximum length of the file name buffer.*
- #define **MOUNT_POINT** "/sd"

  *Directory in virtual file system (VFS) where microSD card gets mounted.*
- #define **LOG_FOLDER** MOUNT_POINT"/"

  *Path prefix of the directory where recordings are saved.*
- #define **NO_WAIT** 10 / portTICK_PERIOD_MS

  *Minimal possible delay in milliseconds for using a synchronization primitive.*
- #define **SWITCH_DEBOUNCE** 2000 / portTICK_PERIOD_MS

  *Period in milliseconds for which the button is disabled after the press to prevent the bouncing effect.*
- #define **CARD_CLK_PIN** 14

  *SD/MMC bus GPIO pin for CLK.*
- #define **CARD_CMD_PIN** 15

  *SD/MMC bus GPIO pin for CMD.*
- #define **CARD_D0_PIN** 2

  *SD/MMC bus GPIO pin for D0.*
- #define **RECORD_SWITCH_PIN** 34

  *GPIO pin for a button that starts and stops recording.*
- #define **RECORD_LED_PIN** 32

  *GPIO pin for the indicator LED.*
- #define **SENSOR_MISO** 13

  *GPIO pin for accelerometer SPI Master In Slave Out.*
- #define **SENSOR_MOSI** 16

  *GPIO pin for accelerometer SPI Master Out Slave In.*
- #define **SENSOR_CLK** 4

  *GPIO pin for accelerometer SPI Clock.*
- #define **SENSOR_CS** 5

  *GPIO pin for accelerometer SPI Chip Select.*
- #define **SENSOR_INT1** 33

  *GPIO pin for accelerometer interrupt pin.*

- #define **SPI_BUS_FREQUENCY** SPI_MASTER_FREQ_8M

  *SPI master bus frequency.*
- #define **FIFO_LENGTH** 512

  *Length of accelerometer FIFO buffer.*
- #define **FIFO_WATERMARK** FIFO_LENGTH / 2

  *Half length of accelerometer FIFO buffer.*
- #define **NUM_OF_FIELDS** 4

  *Number of columns per acceleration vector.*
- #define **SENSOR_SPI_LENGTH** NUM_OF_FIELDS ∗ FIFO_LENGTH

  *Length of buffer for SPI transaction.*
- #define **QUEUE_LENGTH** 16

  *Number of FIFO buffers that can be pushed to Queue before file write.*

### Functions

- void panic (int delay)

  *Signal fatal error of system indiacted by blinking LED and halting execution.*

#### 1.1.1 Detailed Description

Data flow control from the sensor to the memory card.

#### 1.1.2 Function Documentation

**panic()**

```
void panic (
            int delay )
```

Signal fatal error of system indiacted by blinking LED and halting execution.

**Parameters**

| | | |
|---|---|---|
| in | *delay* | Interval in milliseconds for LED blink |

#### 1.1.3 Accelerometer

Configuration of the accelerometer sensor.

### Data Structures

- struct Acceleration

  *Unprocessed data packet for storing samples from accelerometer.*

### Macros

- #define **SAMPLE_RATE** 9000

  *Interval of the periodic timer in milliseconds that reads out circa half of accelerometer FIFO.*
- #define **SPI_BUS** SPI3_HOST

  *Hardware bus for accelerometer SPI interface.*
- #define **AUTO_TURN_OFF_US** 60000000

  *Duration of recoding in microseconds (60 s)*
- #define **ACC_RESOLUTION** IIS3DWB_4g

  *Resolution of the accelerometer in a unit of g.*

### Functions

- int sensor_enable (spi_device_handle_t ∗spi_dev, stmdev_ctx_t ∗dev)

  *Configure SPI bus and set accelerometer to required parameters.*
- void sensor_disable (spi_device_handle_t spi_dev)

  *Remove accelerometer from the SPI bus and disable it.*
- void sensor_events_enable (stmdev_ctx_t ∗dev)

  *Enable accelerometer interrupts.*
- void sensor_events_disable (stmdev_ctx_t ∗dev)

  *Disable accelerometer interrupts.*

#### 1.1.3.1 Detailed Description

Configuration of the accelerometer sensor.

#### 1.1.3.2 Function Documentation

##### sensor_disable()

```
void sensor_disable (
            spi_device_handle_t spi_dev )
```

Remove accelerometer from the SPI bus and disable it.

**Parameters**

| | | |
|---|---|---|
| in | *spi_dev* | SPI bus |

##### sensor_enable()

```
int sensor_enable (
            spi_device_handle_t * spi_dev,
            stmdev_ctx_t * dev )
```

Configure SPI bus and set accelerometer to required parameters.

**Parameters**

| | | |
|---|---|---|
| in | *spi_dev* | SPI bus |
| in | *dev* | Accelerometer sensor |

**Returns**

Status code of successful setup

### sensor_events_disable()

```
void sensor_events_disable (
            stmdev_ctx_t * dev )
```

Disable accelerometer interrupts.

**Parameters**

| | | |
|---|---|---|
| in | *dev* | Accelerometer sensor |

### sensor_events_enable()

```
void sensor_events_enable (
            stmdev_ctx_t * dev )
```

Enable accelerometer interrupts.

**Parameters**

| | | |
|---|---|---|
| in | *dev* | Accelerometer sensor |

## 1.1.4 Memory card

Filesystem operations of the SD card.

### Functions

- sdmmc_card_t * storage_enable (const char *mount_point)

  *Configure SD/MMC bus and mount SD memory card to FAT filesystem.*
- void storage_disable (sdmmc_card_t *card, const char *mount_point)

  *Disable and unmount SD memory card from FAT filesystem.*
- void get_recording_filename (char *filename, const char *path)

  *Get file name for new recording with sequentially higher unused number.*

#### 1.1.4.1 Detailed Description

Filesystem operations of the SD card.

#### 1.1.4.2 Function Documentation

**get_recording_filename()**

```
void get_recording_filename (
            char * filename,
            const char * path )
```

Get file name for new recording with sequentially higher unused number.

**Parameters**

| | | |
|---|---|---|
| out | *filename* | Available file name for new file |
| in | *path* | Base path prefix for saving the recording |

**storage_disable()**

```
void storage_disable (
            sdmmc_card_t * card,
            const char * mount_point )
```

Disable and unmount SD memory card from FAT filesystem.

**Parameters**

| | | |
|---|---|---|
| in | *card* | SD/MMC card information structure |
| in | *mount_point* | path where partition is registered |

**Returns**

c

**storage_enable()**

```
sdmmc_card_t * storage_enable (
            const char * mount_point )
```

Configure SD/MMC bus and mount SD memory card to FAT filesystem.

**Parameters**

| | | |
|---|---|---|
| in | *mount_point* | path where the partition will be registered |

**Returns**

SD/MMC card information structure

### 1.1.5 Button

Configuration of the recording button.

**Functions**

- void switch_enable (bool on, gpio_isr_t isr_handler)

  *Configure GPIO input pin and interrupt handler for button press.*
- void **switch_disable** (void)

  *Remove interrupt handler for button press.*

#### 1.1.5.1 Detailed Description

Configuration of the recording button.

#### 1.1.5.2 Function Documentation

**switch_enable()**

```
void switch_enable (
            bool on,
            gpio_isr_t isr_handler )
```

Configure GPIO input pin and interrupt handler for button press.

**Parameters**

| | | |
|---|---|---|
| in | *on* | decides whether the button is enabled or disabled |
| in | *isr_handler* | handler function for button press in interrupt context |

### 1.1.6 LED

Configuration of the indicator LED light.

**Functions**

- void **led_enable** (void)

  *Configure GPIO for LED to output mode.*
- void led_light (bool on)

  *Set the LED state.*

#### 1.1.6.1 Detailed Description

Configuration of the indicator LED light.

#### 1.1.6.2 Function Documentation

**led_light()**

```
void led_light (
            bool on )
```

Set the LED state.

**Parameters**

| | | |
|---|---|---|
| in | *on* | turns LED light to be either on or off |

## 1.2 Firmware Tasks

Main program of the firmware execution.

### Functions

- void **push_trigger** (void ∗args)

  *Task to start or stop recoding after siganl from button press.*
- void **read_accelerometer** (void ∗args)

  *Task to read FIFO buffer of the accelerometer and write it to Queue.*
- void **write_card** (void ∗args)

  *Task to write accelerations vectors from Queue to the memory card.*
- void **app_main** (void)

  *Entrypoint of firmware to setup hardware peripherals and run tasks.*

### Variables

- TaskHandle_t **trigger_task**

  *Task handler for notification of button press.*
- TaskHandle_t **sampler_task**

  *Task handler for notification from sampling timer.*
- QueueHandle_t **samples**

  *Queue for sending samples from the sensor read task to the memory card write task.*
- spi_device_handle_t **spi**

  *SPI bus handle.*
- stmdev_ctx_t **sensor**

  *Accelerometer sensor device.*
- sdmmc_card_t ∗ **card** = NULL

  *SD memory card handle.*

- FILE * **file** = NULL

  *Currently opened file handle.*
- SemaphoreHandle_t **file_mutex**

  *Mutex to protect file handle.*
- bool **is_recording** = false

  *Flag for active recording in progress.*
- int32_t **sensor_timestamp** = 0

  *Last seen accelerometer timestamp.*
- const esp_timer_create_args_t sampler_timer_conf
- esp_timer_handle_t **sampler_timer**

  *Periodic timer to signal when to read FIFO buffer from accelerometer.*
- const esp_timer_create_args_t stop_timer_conf
- esp_timer_handle_t **stop_timer**

  *Timer to stop recording after fixed amount of time.*

### 1.2.1 Detailed Description

Main program of the firmware execution.

### 1.2.2 Variable Documentation

#### sampler_timer_conf

```
const esp_timer_create_args_t sampler_timer_conf
```

**Initial value:**

```
= {
        .callback = &isr_sample
}
```

#### stop_timer_conf

```
const esp_timer_create_args_t stop_timer_conf
```

**Initial value:**

```
= {
    .callback = &stop_timer_run
}
```

# 2 Data Structure Documentation

## 2.1 Acceleration Struct Reference

Unprocessed data packet for storing samples from accelerometer.

```
#include <pinout.h>
```

### Data Fields

- uint16_t **len**

  *Number of samples in every array in the structure.*
- int32_t **t** [FIFO_LENGTH]

  *Array of timestamps relative to time when sensor was enabled.*
- int32_t **x** [FIFO_LENGTH]

  *Accelerometer samples for X axis.*
- int32_t **y** [FIFO_LENGTH]

  *Accelerometer samples for Y axis.*
- int32_t **z** [FIFO_LENGTH]

  *Accelerometer samples for Z axis.*

#### 2.1.1 Detailed Description

Unprocessed data packet for storing samples from accelerometer.

The documentation for this struct was generated from the following file:

- firmware/main/include/pinout.h

# Index


Acceleration, 8
Accelerometer, 2
    sensor_disable, 3
    sensor_enable, 3
    sensor_events_disable, 4
    sensor_events_enable, 4
Accelerometer Data logger, 1
    panic, 2

Button, 6
    switch_enable, 6

Firmware Tasks, 7
    sampler_timer_conf, 8
    stop_timer_conf, 8

get_recording_filename
    Memory card, 5

LED, 6
    led_light, 7
led_light
    LED, 7

Memory card, 4
    get_recording_filename, 5
    storage_disable, 5
    storage_enable, 5

panic
    Accelerometer Data logger, 2

sampler_timer_conf
    Firmware Tasks, 8
sensor_disable
    Accelerometer, 3
sensor_enable
    Accelerometer, 3
sensor_events_disable
    Accelerometer, 4
sensor_events_enable
    Accelerometer, 4
stop_timer_conf
    Firmware Tasks, 8
storage_disable
    Memory card, 5
storage_enable
    Memory card, 5
switch_enable
    Button, 6