User manual

Getting started with InfraredAL presence detection library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

InfraredAL is a middleware library component of the X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about the presence of a person in the field of view of the sensor.

This library is intended to work with the STHS34PF80 sensor only.

The algorithm is provided in static library format and is designed to be used on STM32 microcontrollers based on the ARM® Cortex® -M0+, ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 or ARM® Cortex®-M7 architectures.

It is built on top of STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with sample implementation running on X-NUCLEO-IKS01A3, X-NUCLEO-IKS02A1, and X-NUCLEO-IKS4A1 expansion boards on a NUCLEO-F401RE, NUCLEO-L073RZ, NUCLEO-L152RE, or NUCLEO-U575ZI-Q development board.

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
API	Application programming interface
BSP	Board support package
GUI	Graphical user interface
HAL	Hardware abstraction layer
IDE	Integrated development environment

2 InfraredAL middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 InfraredAL overview

The InfraredAL library expands the functionality of the X-CUBE-MEMS1 software.

The library acquires data from the infrared sensor and provides real-time information about the presence of a person in the field of view of the sensor with its confidence.

This library is intended to work with the STHS34PF80 sensor only. Functionality and performance when using other sensors are not analyzed and can be significantly different from what described in the document.

A sample implementation is available on X-NUCLEO-IKS01A3, X-NUCLEO-IKS02A1, and X-NUCLEO-IKS4A1 expansion boards on a NUCLEO-F401RE, NUCLEO-L073RZ, NUCLEO-L152RE, or NUCLEO-U575ZI-Q development board.

2.2 InfraredAL library

Technical information fully describing the functions and parameters of the InfraredAL APIs can be found in the InfraredAL_Package.chm compiled HTML file located in the Documentation folder.

2.2.1 InfraredAL library description

The InfraredAL sensor fusion library manages data acquired from the infrared sensor; it features:

- the presence detection and its level of confidence
- recommended sensor data sampling frequency of 1 Hz to 30 Hz
- resources requirements:flash ram
- Cortex-M0+: 3.23 kB of code and up to 0.52 kB of data memory
- Cortex-M33: 3.25 kB of code and up to 0.51 kB of data memory
- Cortex-M3: 3.17 kB of code and up to 0.51 kB of data memory
- Cortex-M4: 3.24 kB of code and up to 0.51 kB of data memory
- Cortex-M7: 3.25 kB of code and up to 0.51 kB of data memory
- available for ARM Cortex-M0+, Cortex-M3, Cortex-M33, Cortex-M4 and Cortex-M7 architecture

Note: The size of dynamically allocated data memory is dependent on algorithm setup.

2.2.2 InfraredAL library operation

The InfraredAL library implements human presence detection for workstation applications. The algorithms of the library are compatible with Microsoft Windows's Wake-on-Approach and Lock-on-Leave features.

The library is designed for the STHS34PF80 TMOS IR sensor only. Its functionality and performance with other sensors have not been not analyzed and can differ significantly different from what is described here.

The library implements algorithms that provide an output flag which reports whether or not a user is present inside the field of view of the sensor (presence or absence states). The algorithms also provide an output value which reports the level of confidence on the output flag of presence detection. The confidence can range from 50% (complete uncertainty) to 100% (complete certainty) for both states.

The library works by leveraging on two different algorithms: a “motion-based” presence detection algorithm and a “true” presence detection algorithm. The only input needed by the algorithms is the uncompensated object temperature signal (TOBJECT) which is exposed by the STHS34PF80 IR sensor.

The “true” presence detection algorithm is always enabled, and it is triggered when the user enters the field of view. When this algorithm is triggered, the output flag reports the presence state and the confidence is locked at 100%, and the “motion-based” presence detection algorithm is disabled.

The “true” presence detection algorithm can be configured by changing the parameter representing the threshold for presence detection. This threshold should be fine-tuned depending on the final application, and it should be configured to be lower than the change in the input signal baseline caused by a user entering the field of view and then positions themself at the distance from the sensor where it is expected they will be usually positioned.

The “motion-based” presence detection algorithm is enabled at start-up or whenever the “true” presence detection algorithm is not triggered. In particular, at startup the user may be already present inside the sensor's field of view, so the “true” presence detection algorithm will not be triggered until they exit and re-enter the field of view. When the “motion-based” presence detection algorithm is enabled, an internal timer gradually decreases the confidence on the presence state from 100% downward; when confidence reaches 50%, the output flag changes to the absence state and the confidence on absence state gradually increases up to 100%. Whenever small movements of the user inside the field of view are detected, the presence flag changes back to the presence state (if it was previously set to the absence state) and the confidence on the presence state is set back to 100%, the timer is reset and the cycle restarts. This value of the timer for the confidence on presence state to decrease from 100% to 50%, and for the output flag to be set to the absence state, can be configured with a specific configuration parameter.

The “motion-based” presence detection algorithm can be configured by changing a parameter representing the threshold for motion detection. This threshold should be fine-tuned depending on the final application, and it should be configured to be low enough to be triggered by small movements of the user inside the field of view when the user is positioned at the distance from the sensor where it is expected they will be usually positioned, but high enough so that it is not triggered by environmental infrared noise.

As mentioned, the algorithms of the library have been designed to be compatible with the requirements of the Wake-on-Approach and Lock-on-Leave features of Microsoft Windows. For the Wake-on-Approach feature, this means that the entrance / approach of the user into the field of view is detected within 1 second (i.e., if the output flag was set to the absence state, it will be set to the presence state within 1 second). For the Lock-on-Leave feature, the exit / leave of the user from the field of view is detected within 5 seconds (i.e., if the output flag was set to the presence state, it will be set to the absence state within 5 seconds) if the timer of the “motion-based” presence detection algorithm is set at a value equal or lower to the default value (5000 ms); if the timer is set to a higher value, the library may not be compatible with the requirements of the Lock-on-Leave feature.

If the algorithms are confidently started with an empty field of view, a specific configuration flag can be used to initialize the motion detection algorithm in the absence state instead of the presence state, as by default. In both cases, confidence is set at 100% at the start-up of the algorithms.

The algorithms have been designed to work for workstation applications, where the distance from the sensor is limited (in the order of 0.5 m to 1.5 m). Moreover, the algorithms have been designed to be resilient against the entrance into and the exit from the field of view of multiple people in addition to the main user.

2.2.3 InfraredAL library parameters

In the following, the types that are defined in the header file of the library.

```
typedef void *IAL_Instance_t;
```

- pointer to the library instance loaded in data memory

```
typedef enum
{
  IAL_MCU_STM32 = 0,
  IAL_MCU_BLUE_NRG1,
  IAL_MCU_BLUE_NRG2,
  IAL_MCU_BLUE_NRG_LP,
} IAL_mcu_type_t;
```

- used MCU type

```
typedef enum
{
  IAL_INIT_OK = 0,       /* No error */
  IAL_INIT_ERR_ALLOC,    /* Instance NULL */
  IAL_INIT_ERR_ODR,      /* Wrong ODR value */
  IAL_INIT_ERR_RES,      /* Reserved error code */
} IAL_init_err_t;
```

- library status – error code returned by InfraredAL_Start API function

```
typedef enum
{
  IAL_RUN_OK = 0,        /* No error */
  IAL_RUN_ERR_ALLOC,     /* Instance NULL */
  IAL_RUN_ERR_UNINIT,    /* Instance uninitialised*/
  IAL_RUN_ERR_RES,       /* Reserved error code */
} IAL_run_err_t;
```

- library status – error code returned by InfraredAL_Update API function

```
typedef struct
{
  uint8_t  odr;
} IAL_device_conf_t;
```

- the parameters of the device, that must be configured and/or retrieved in the application code and passed to the algorithm during initialization:
  - `odr` – ODR in Hz, possible values are from 1 Hz to 30 Hz

```
typedef struct
{
  uint16_t mot_ths;
  uint16_t pres_ths;
  uint16_t abs_timer;
  bool empty_fov_init;
} IAL_algo_conf_t;
```

- the parameters of the algorithm that can be configured from the application code:
  - `mot_ths` – threshold for the “motion-based” presence detection algorithm [LSB]
  - `pres_ths` – threshold for the “true” presence detection algorithm [LSB]
  - `abs_timer` – timer for the confidence on presence state to decrease from 100% to 50% of the “motion-based” presence detection algorithm [ms]
  - `empty_fov_init`– flag reporting whether algorithm is confidently started with an empty field of view

```
typedef struct
{
  int16_t t_obj;
} IAL_input_t;
```

- the inputs of the algorithms that must be provided to the algorithms at each iteration:
  - `t_obj` – raw object temperature data [LSB]

```
typedef struct
{
  uint8_t pres_flag;     /* Presence detection flag [0: absence, 1: presence] */
  uint8_t pres_conf;     /* Presence detection confidence [%] */
} IAL_output_t;
```

- the outputs of the algorithms that are provided by the algorithms at each iteration:
  - `pres_flag` – presence detection flag [0: absence, 1: presence]
  - `pres_conf` – presence detection confidence [%]

2.2.4 InfraredAL APIs

In the following, the API functions that are defined in the header file of the library.

```
uint8_t InfraredAL_GetLibVersion(char *version)
```

- Retrieve the version of the library
  - `version` – pointer to an array of up to 35 characters
- Return the number of characters in the version string

```
void InfraredAL_Initialize(IAL_mcu_type_t mcu_type)
```

- Perform InfraredAL library initialization and setup of the internal mechanism

Note: This function must be called before using the presence detection library and the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled.

```
IAL_Instance_t InfraredAL_CreateInstance(IAL_algo_conf_t *algo_conf)
```

- Create an instance of the InfraredAL library:
  - Allocate the memory for a library instance
  - Fill the structure pointed by `algo_conf` with the default values of the parameters for the configuration of the algorithms
  - Return a pointer to its memory location

Note: After calling this function, the algorithms can be initialized with the `InfraredAL_Start()` API function

- `algo_conf` – configuration of the algorithms
- Return pointer to new instance of the algorithms

```
void InfraredAL_DeleteInstance(IAL_Instance_t instance)
```

- Delete instance of the InfraredAL library:
  - De-initialize the algorithms
  - Free the memory allocated for the instance
- `instance` – pointer to instance of the library to be deleted

```
IAL_init_err_t InfraredAL_Start(IAL_Instance_t instance, IAL_device_conf_t
                                *device_conf, IAL_algo_conf_t *algo_conf)
```

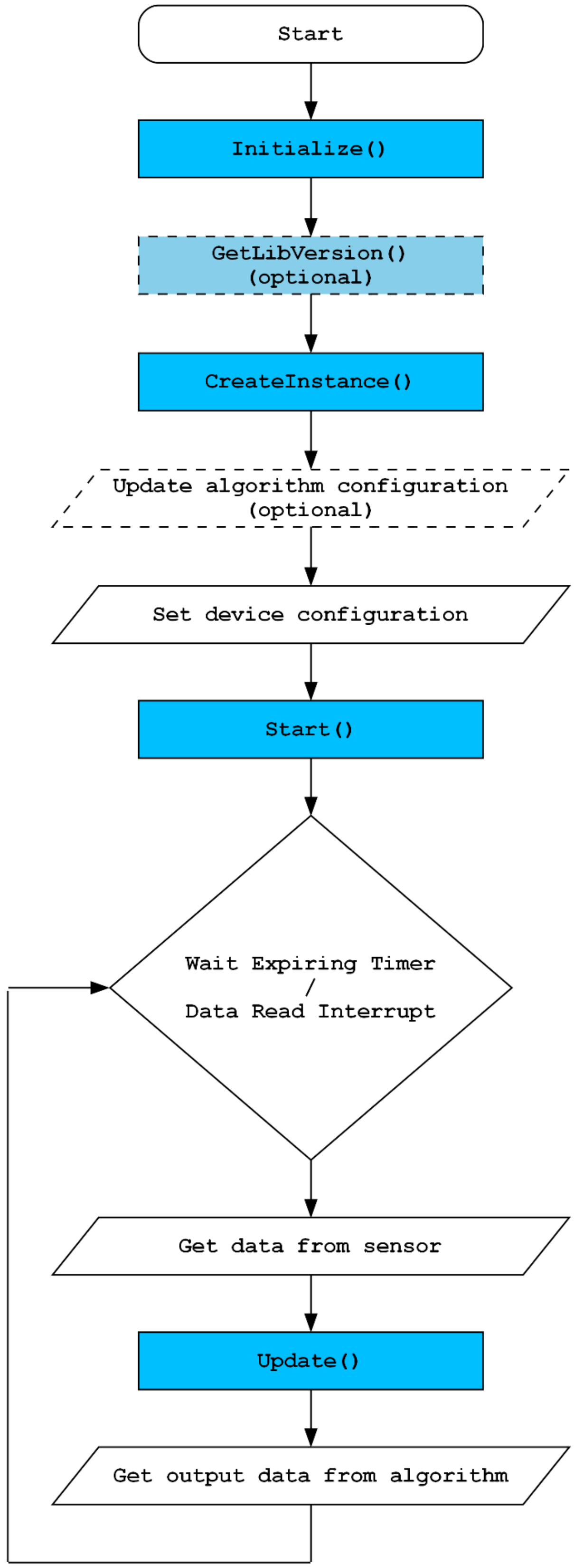
- Start the InfraredAL engine:
  - Initialize (or re-initialize) the algorithms of the instance following the parameters set in the two structures pointed by `device_conf` and `algo_ conf`
  - Return an initialization error code (e.g.: if invalid device parameters were set)
- `instance` – pointer to instance of the library to be started
- `device_conf` – configuration of the device
- `algo_conf` – configuration of the algorithms
- Return an initialization error code

```
IAL_run_err_t InfraredAL_Update(IAL_Instance_t instance, IAL_input_t *data_in, IAL_output_t *
data_out);
```

- Execute one step of the algorithm
- `instance` – pointer to instance of the library
- `data_in` – input data
- `data_out` – output data
- Return an execution error code

2.2.5 API flow chart

Figure 1. InfraredAL API logic sequence



2.2.6 Demo code

```
#define IAL_STR_LENG  35

/*** Initialization ***/
char lib_version[IAL_STR_LENG];
IAL_mcu_type_t mcu = IAL_MCU_STM32;
IAL_Instance_t IAL_Instance;
IAL_device_conf_t device_conf;
IAL_algo_conf_t algo_conf;
IAL_init_err_t init_err;
IAL_run_err_t run_err;
uint8_t pres_flag_old;

/* Library API initialization function */
InfraredAL_Initialize(mcu);

/* Optional: Get version */
InfraredAL_GetLibVersion(lib_version);

/* Create library algorithm instance */
IAL_Instance = InfraredAL_CreateInstance(&algo_conf);

/* Setup device configuration */
device_conf.odr = 30;

/* Optional: Modify algorithm configuration */
algo_conf.mot_ths = 250;
algo_conf.pres_ths = 1000;
algo_conf.abs_timer = 5000;
algo_conf.empty_fov_init = false;

/* Start the algorithm engine */
init_err = InfraredAL_Start(IAL_Instance, &device_conf, &algo_conf);
if (algo_conf.empty_fov_init == true)
{
  pres_flag_old = 0;
}
else
{
  pres_flag_old = 1;
}

/*** Using Presence Detection algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
  IAL_input_t data_in;
  IAL_output_t data_out;
  uint8_t pres_flag;
  uint8_t pres_conf;

  /* Get data from sensor */
  ReadSensor(&data_in.t_obj);

  /* Execute one step of the algorithms */
  run_err = InfraredAL_Update(IAL_Instance, &data_in, &data_out);

  /* Get output data from algorithm */
  pres_flag = data_out.pres_flag;
  pres_conf = data_out.pres_conf;

  if ((pres_flag_old == 0) && (pres_flag == 1))
  {
    /* Report approach event */
  }
  else if ((pres_flag_old == 1) && (pres_flag == 0))
  {
    /* Report leave event */
  }
  else
```

```
  {
    /* No event */
  }

  pres_flag_old = pres_flag;
}
```

2.2.7 Algorithm performance

Table 2. **Elapsed time (μs) algorithm**

MCU	Min	Average	Max
Cortex-M4 STM32F401RE at 84 MHz	1	1.88	3
Cortex-M3 STM32L152RE at 32 MHz	<1	<1	1
Cortex-M33 STM32U575ZI-Q at 160 MHz	<1	<1	1
Cortex-M0+ STM32L073RZ at 32 MHz	<1	<1	1

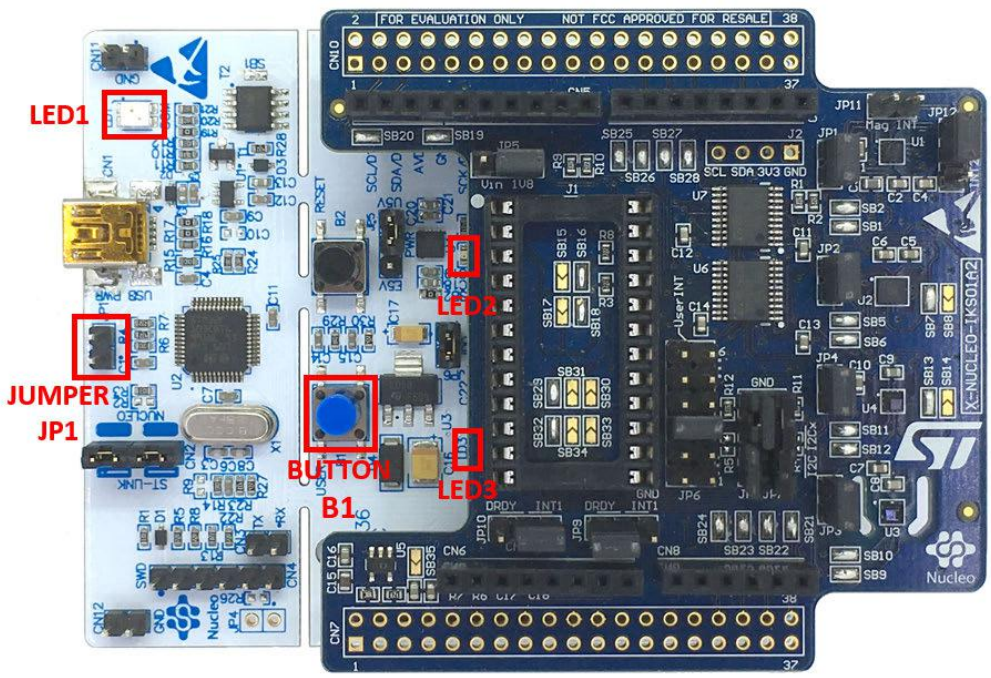
2.3 Sample application

The InfrearedAL middleware can be easily manipulated to build user applications. A sample application is provided in the Application folder.

It is designed to run on X-NUCLEO-IKS01A3, X-NUCLEO-IKS02A1, and X-NUCLEO-IKS4A1 expansion board on a NUCLEO-F401RE, NUCLEO-L073RZ, NUCLEO-L152RE, or NUCLEO-U575ZI-Q development board.

The application provides real-time information about the presence of a person in the field of view of the sensor and its confidence.

Figure 2. **STM32 Nucleo LEDs, button, and jumpers**



The above figure shows the user button B1 and the three LEDs of the NUCLEO-F401RE board. Once the board is powered, LED LD3 (PWR) turns ON.

Note: After powering the board, LED LD2 blinks once indicating the application is ready.

2.3.1 MEMS Studio application

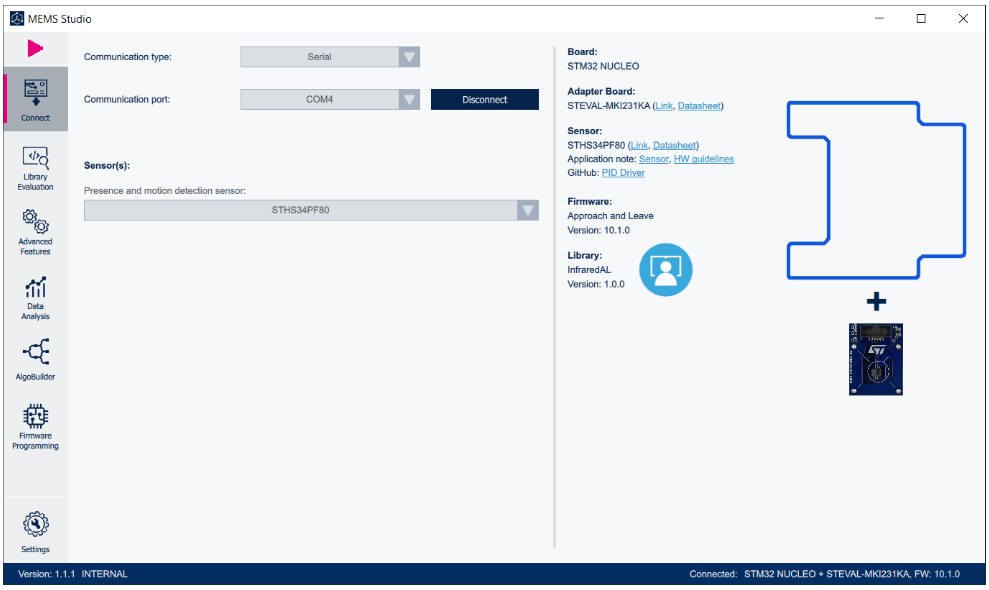
The sample application uses the MEMS Studio application, which can be downloaded from www.st.com.

Step 1. Ensure that the necessary drivers are installed and the STM32 Nucleo board with appropriate expansion board is connected to the PC.

Step 2. Launch the MEMS Studio application to open the main application window.
If an STM32 Nucleo board with supported firmware is connected to the PC, it is automatically detected and the appropriate COM port is opened.

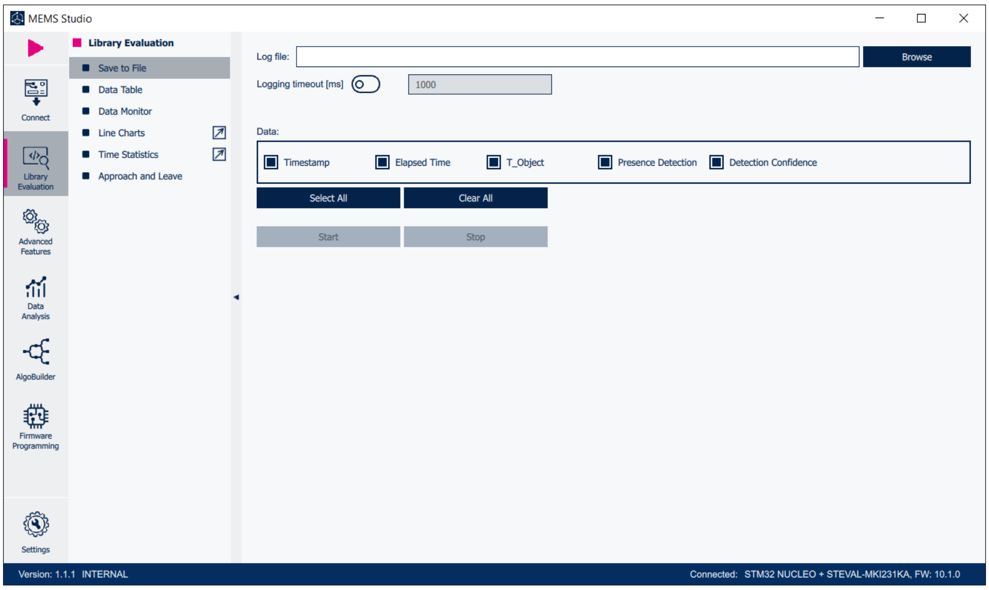
Step 3. Press the [**Connect**] button to start communicating.

Figure 3. MEMS Studio Connect window



Step 4. Start and stop data streaming by using the appropriate buttons on the vertical tool bar.
The data coming from the connected sensor can be viewed in the [**Data table**] and [**Data monitor**] tabs.

Figure 4. Library Evaluation window

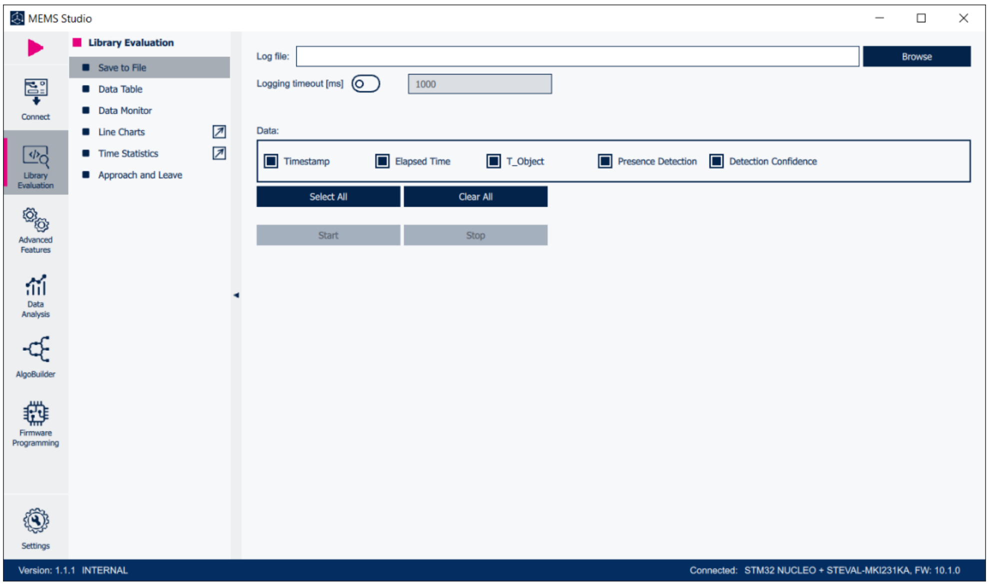


Step 5. Click on the [**Approach and Leave**] icon in the vertical toolbar to open the dedicated application window to see the temperature of the object in LSB, presence detection flag (H = presence, L = absence), and the detection confidence in %.

Figure 5. Approach and Leave window

Step 6. Click on the [**Save to file**] icon in the vertical toolbar to open the datalog configuration window: you can select the data to be saved in the files. You can start or stop saving by clicking on the corresponding button.

Figure 6. Datalog window



3 References

All of the following resources are freely available on www.st.com.

[1] Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube (UM1859)

[2] STM32 Nucleo-64 board (UM1724)

[3] Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube (UM2128)

Revision history

Table 3. **Document revision history**

Date	Version	Changes
05-Aug-2024	1	Initial release.
18-Feb-2025	2	Update: Section 2.2.2: InfraredAL library operation, Section 2.2.3: InfraredAL library parameters, Section 2.2.4: InfraredAL APIs, Section 2.2.6: Demo code
14-May-2025	3	Update: Section 2.2.6: Demo code.

Contents

List of figures

List of tables