How to use the software package for the SensorTile Wireless Industrial Node based on STM32Cube

**Introduction**

The STSW-STWINKT01 firmware package for the SensorTile Wireless Industrial Node (STWIN) development kit provides sample projects that you can use to develop custom predictive maintenance, smart industry, IoT and remote monitoring applications. The package is based on our STM32Cube software technology, and includes all the low level drivers to manage the on-board devices and system-level interfaces.

The package features two projects that demonstrate data logging functionality. One involves streaming data via the USB Virtual COM Port class, which you can subsequently display directly on a PC terminal, and the other is for high speed data storage at maximum sensor sampling rates. In the second case, sensor data can be either stored on micro SD card or can be streamed via USB (WinUSB class).

There are also three projects regarding audio. The OnboardMics application streams audio signals from both the on-board analog and digital microphones via USB, so you can record and process them on a host PC. The MicArrayCoupon application instead requires the STEVAL-STWINMAV1 microphone array expansion board (not included in the kit), from which the application can simultaneously acquire up to four external microphone audio signals through a USB interface.

The UltrasoundFFT example calculates the FFT of the on-board analog microphone signal and streams the result to a PC GUI via USB. The microphone sampling rate is set by default to 192 kHz whereas the microphone bandwidth is up to 80 kHz.

A further two projects demonstrate wireless connectivity using Bluetooth and Wi-Fi. The Bluetooth project allows you to stream environmental sensor data via the Bluetooth Low Energy (BLE) protocol, and is compatible with our freely available ST BLE Sensor app on Android and iOS stores, so you can read and manipulate the data from your mobile device. The Wi-Fi project requires the STEVAL-STWINWFV1 Wi-Fi expansion board (not included in the kit) to implement basic network functionality including pinging a remote station, connecting to a TLS secure server, sending data to an echo server and verifying returned data, and running a server that a remote client can connect to.

**Figure 1. SensorTile Wireless Industrial Node application diagram**
The **STSW-STWINKT01** firmware expands the functionality of the STM32Cube environment with many features and functions needed to build wireless predictive maintenance and condition monitoring applications and provides some working code examples to help you become familiar with the underlying middleware.

The key features of the package are:

- Set of firmware examples that show how to implement basic functions on the STWIN (**STEVALL-STWINKT1**) kit:
  - Sensor data streaming example via USB terminal (VCP)
  - Analog and digital microphone signal acquisition and streaming via USB
  - Wi-Fi network functionality using the connectivity framework (with Wi-Fi expansion board)
  - Sensor data streaming via BLE
  - High speed sensor data logger to SD card or via USB
  - Microphone array audio acquisition example (with microphone expansion board)
  - Ultrasound FFT analysis demonstration
  - Source code freely available from the ST website with developer-friendly license terms

- Embedded software, middleware and drivers:
  - FatFS third party FAT file system module for small embedded systems
  - FreeRTOS third party RTOS kernel for embedded devices
  - STWIN Low-Level BSP drivers

- Based on **STM32Cube** software development environment for STM32 microcontrollers
2 How to program the board

Follow the procedure below to program the STWIN core system board.

**Step 1.** Connect the STWIN core system board to the STLINK-V3MINI programmer using the 14-pin flat cable. The programmer and the cable are included in the STEVAL-STWINKT1 hardware kit.

**Step 2.** Connect both the boards to a PC using micro USB cables.

**Step 3.** Download the firmware onto the core system board; you can either:
- download one of the sample application binaries provided
- recompile one of the projects with your preferred IDE (EWARM, Keil, STM32CubeIDE)

**Figure 3.** STLINK-V3MINI connected to STWIN core system board
3 Sample applications

3.1 Serial_DataLog application

The Serial DataLog application shows how to stream sensor data via USB Virtual COM Port class, so the data can be viewed using any serial terminal software like TeraTerm or PuTTY.

After reset, the firmware performs the following initial tasks:
1. configures HAL and clocks
2. configures LED1 and LED2
3. initializes the USB peripheral
4. creates the threads and activate FreeRTOS scheduler

The GetData_Thread and WriteData_Thread threads are scheduled by FreeRTOS with different priorities and communicate with each other through a message queue:
- **GetData_Thread**: a high priority task that configures the sensors, reads data at a given frequency and pushes new data in the queue. An OS timer triggers the execution of the thread at a given frequency.
- **WriteData_Thread**: a low priority task that writes sensor data as soon as they are available in the queue.

After you have downloaded the firmware, you can follow the procedure below to run the Serial DataLog application:

**Step 1.** Connect the board to a PC via micro-USB cable.
   The PC will recognize the board as a Virtual COM Port.

**Step 2.** Open the COM port with a serial terminal like Putty or TeraTerm.
   Use the following parameters: 8N1, 115200 bauds, no HW flow control, line endings LF or CR-LF (Transmit) and LF (receive).
   When connected to the PC, the board configures the sensors and starts streaming data to the PC.

### RELATED LINKS

*Appendix A* Serial Terminal configuration using TeraTerm on page 16

If required, you can download the Windows driver from the ST website: VCP driver

3.2 OnboardMics application

The OnBoardMics application sends the audio signals acquired by both the analog and the digital microphone via USB. The board is recognized as a stereo USB microphone, allowing the user to easily record the signals on a host device using any recording software.

After reset, the firmware performs the following initial tasks:
1. configures HAL and clocks
2. configures LED1 and LED2
3. configures the digital filter peripheral (DFSDM), the DMA and the Analog-to-Digital Converter (ADC)
4. initializes USB audio class
5. starts audio acquisition

The main loop is empty in this application because all the operations needed to copy the audio stream acquired from the microphone to the serial audio interface are executed in the DFSDM Interrupt Service Routine. For this reason, the `AudioProcess()` function is called by `BSP_AUDIO_IN_TransferComplete_CallBack()` and `BSP_AUDIO_IN_HalfTransfer_CallBack()`.

After you have downloaded the firmware, you can follow the procedure below to run the OnBoardMics application:

**Step 1.** Connect the board to a PC with a micro-USB cable.
   The PC will recognize the board as a stereo USB Microphone (2 channels).

**Step 2.** Open your preferred audio editing software.
Step 3. Select [Windows WASAPI] or [MME] driver. Item (1) in following figure.

Step 4. Ensure that the recording device in the software is [STM32 AUDIO...]. Item (2) in following figure.

Figure 4. Audio editing software configuration for OnBoardMics application
1. Driver selection (WASAPI or MME)
2. Recording device (STM32 AUDIO…)

Step 5. Start recording the audio.

--- RELATED LINKS ---
To record the audio signal, you can use any audio editor software like Audacity

3.2.1 Microphone acquisition process
A digital MEMS microphone can be acquired via different peripherals like SPI, I²S, GPIO or DFSDM. It requires an input clock and it outputs a PDM stream at the same frequency of the input clock. This PDM stream is further filtered and decimated for conversion into PCM standard for audio transmission.

Two digital MEMS microphones can be connected on the same data line, configuring the first to generate valid data on the rising edge of the clock and the other on the falling edge by setting the L/R pin of each microphone accordingly.

The STWIN Core System board has one digital (IMP34DT05) microphone and one analog (MP23ABS1) microphone.

The digital microphone output signal is acquired via the digital filter for sigma delta modulators (DFSDM) peripheral, which generates the precise clock needed by the microphone and reads the PDM signal on the rising edge of the CLOCK line. The acquired signal is then passed to the DFSDM filter for hardware filtering and decimation to generate a standard PCM stream. An additional software high pass filtering stage removes any DC offset in the output stream. DMA is used to reduce MCU load.

The analog microphone output signals are amplified through an audio optimized operational amplifier and acquired via the ADC peripheral available on the STM32 microcontroller. The oversampled signal is then input to another DFSDM filter for hardware filtering and decimation to generate a standard PCM stream. DMA is used to reduce MCU load.
3.3 UltrasoundFFT

The UltrasoundFFT example calculates the FFT of the on-board MP23ABS1 analog microphone signal and streams the result to a PC GUI (available in 'Utilities/UltrasoundFFT') via USB. The microphone sampling rate is set by default to 192 kHz whereas the microphone bandwidth is up to 80 kHz.

Thanks to the very high sampling frequency available, the application can be used to perform condition analysis in the ultrasound frequency domain on any kind of machinery.

After the startup sequence, the board is in idle state, waiting for the 'start' command from the PC GUI.

When the PC sends the command to the STEVAL-STWINKT1 board through USB, the STM32 starts calculating and streaming the audio FFT values. Power Spectrum Density is plotted into the GUI and the user can choose thresholds in both energy and frequency ranges to easily find the maximum energetic bin.

Figure 5. UltrasoundFFT - power spectrum density vs. frequency
Through the GUI it is also possible to store the FFT values into a raw data file by checking the flag 'Save to file'. Files are saved in the folder 'Acquisition'; the file name is 'YYYYMMDD_HHMMSS.dat' (i.e: 20200117_155823.dat)
The folder 'Utilities/UltrasoundFFT' contains also a Matlab and a Python scripts called 'ReadFFT' to plot the spectrogram of the data saved into the 'Acquisition' folder. 'ReadFFT.py' has been tested using Python 3.7 on Linux and Windows 10 (Anaconda environment). 'ReadFFT.m' has been tested using MATLABv2019a.

### 3.4 WiFi_Connectivity application

This sample application requires STEVAL-STWINWFV1 Wi-Fi expansion plugged on the CN3 connector of the core system board.

The application implements basic network functionality using the connectivity framework.
WiFi Connectivity demonstrates how to set up the STM32_Connect_Library Middleware to allow the STWIN core system board to:

- Ping a remote station
- Connect to a TLS secure server without server identification check
- Connect to a TLS secure server with server identification check
- Send data to an echo server and check return data
- Run a Server, waiting for connection from a remote client

Following reset, the firmware performs the following initial tasks:
1. configures HAL and clocks
2. configures LED1 and LED2
3. configures USB peripheral
4. configures the network interface and initialize the Wi-Fi module
5. starts the connectivity examples

After you have downloaded the firmware, you can follow the procedure below to run the WiFi Connectivity application:

**Step 1.** Connect the core system board with Wi-Fi module expansion board to a PC with a micro-USB cable. The PC will recognize it as a Virtual COM Port.

**Step 2.** Open the COM port with a serial terminal like Putty or TeraTerm. Use the following parameters: 8N1, 115200 bauds, no HW flow control, line endings LF or CR-LF (Transmit) and LF (receive).
Step 3. Run the program.
Your terminal console will return output status information.

Figure 10. Serial terminal output for Wi-Fi_Connectivity app status

Step 4. Change default IPs and access credentials (optional).
By default, the STWIN tries to connect to the following Wi-Fi network:
- SSID: stwintest
- Password: stwintest
- Security: WPA2 PSK
- Band: 2.4GHz

The default SSID and Password may be changed by modifying the following defines in main.c file:
- #define SSID "stwintest"
- #define PASSWORD "stwintest"

You can also edit the test_client_server.c file to set up a different address for the TCP echo server. By default, the ARM-Mbed echo server is used:
- #define REMOTE_IP_ADDR "52.215.34.155"
- #define REMOTE_PORT 7

---

RELATED LINKS

Appendix A Serial Terminal configuration using TeraTerm on page 16
3.5 **BLE_SampleApp application**

BLE_SampleApp provides an example of Bluetooth Low Energy configuration that enables the STWIN core system board to stream environmental sensor data; it is compatible with the ST BLE Sensor mobile app available for Android and iOS. BLE_SampleApp also shows how to correctly configure the STBC02 Battery Charger, providing a good example of how to manage power on/off routines for the STWIN board.

Following reset, the firmware performs the following initial tasks:

- configures HAL, clocks and buttons
- configures Battery Charger
- initializes the target platform:
  - USB peripheral (for debugging)
    - LED1 and LED2
    - environmental sensors
    - initializes the Bluetooth Low Energy stack
- initializes the Bluetooth Low Energy services
- initializes timers
- starts the main loop:
  - LED management
  - BLE event management
  - environmental sensors data management

After you have downloaded the firmware, you can follow the procedure below to run the BLE SampleApp application:

**Step 1.** Once downloaded, the STWIN LED starts blinking. This signals that the device is waiting for a connection via Bluetooth.

**Step 2.** Open the STBLESensor app on your Android or iOS smartphone and use it to connect to the STWIN core system board.

---

**RELATED LINKS**

Visit the ST website for information regarding ST BLE Sensor mobile app.

---

3.6 **HS_Datalog application**

Before you begin, you can choose which sensors to log data from by editing the `Enable_Sensors()` function in `sdcard_manager.c`.

The HS_DataLog (High-Speed DataLog) allows you to save data from any combination of sensors and microphones configured at their maximum sampling rate. Sensor data are stored on a micro SD Card, SDHC (Secure Digital High Capacity) formatted with the FAT32 file system, or can be streamed to a PC via USB.

At startup, the application tries to load the device configuration from the SD card (if any) and then goes to Idle state, waiting for the start command either via USB or push button.
Together with HS_Datalog application, inside the Utilities folder, a MATLAB and a Python script are available to automatically read and plot the data saved by the application. The script has been successfully tested with MATLAB v2017a and Python 3.7. The 'ReadSensorDataApp.mlapp' MATLAB app is also available, developed and tested using the App Designer tool available in MATLAB v2019a.
The script performs the following actions:
- Reads and decodes the JSON file
- Reads the raw data and uses the information from the JSON to translate them into readable data (data + timestamp)
- Plots the data

Note: The handling of JSON scripts requires MATLAB v2017a or above.

3.6.1 SD Card considerations
A SensorBuffer.xlsx file is available to help you calculate the amount memory needed for temporary data storage in STM32 RAM, based on the sensor ODR, the bytes per sample and the SD Write period. Using large buffers is far more efficient than small ones when writing data to the SD Card.

As the data logging application may involve large volumes of sensor data, the micro SD card must be capable of handling the data rates without issues. SD card performance vary significantly depending on the size, speed class, and even on the manufacturer.

Our sample high speed data logging application was tested with the following cards, formatted FAT32 with 32KB allocation table:

- Verbatim 16 GB Class 10 U1 (p/n 44082)
- Transcend Premium 16 GB U1 C10 (TS16GUSDCU1)
- Kingston 8 GB HC C4 (SDC4/8 GB)

Note: Smaller allocation tables may impact performance.

- Verbatim 16 GB Class 10 U1 (p/n 44082)
- Transcend Premium 16 GB U1 C10 (TS16GUSDCU1)
- Kingston 8 GB HC C4 (SDC4/8 GB)

RELATED LINKS
See FatFS application note for more details.

3.7 MicArrayCoupon application
This sample application requires STEVAL-STWINMAV1 microphone array expansion plugged on the CN4 connector of the core system board.
Following reset, the firmware performs the following initial tasks:
1. configures HAL and clocks
2. configures LED1 and LED2
3. configures the STM32 Serial Audio Interface peripheral (SAI) in I²S mode and the DMA
4. configures the external Analog-to-Digital Converter (ADC)
5. initializes USB audio class
6. starts audio acquisition

The main loop is empty in this application because all the operations needed to copy the audio stream acquired from the microphone to the serial audio interface are executed in the DMA interrupt. For this reason, the `AudioProcess()` function is called by `AMIC_ARRAY_AUDIO_IN_TransferComplete_CallBack()` and `AMIC_ARRAY_AUDIO_IN_HalfTransfer_CallBack()`.

After you have downloaded the firmware, you can follow the procedure below to run the MicArrayCoupon application:

**Step 1.** Connect the core system board with microphone coupon array expansion board to a PC with a micro-USB cable.
   The PC will recognize the board as a USB Microphone with 4 channels.

**Step 2.** Open your preferred audio editing software.

**Step 3.** Select [Windows WASAPI] or [MME] driver.
   - Select [WASAPI] if it is available.
     This driver allows you to select 4 as recording channels and record all the 4 microphones available on the board.
   - If [WASAPI] is not available, select [MME].
     This driver only allows you to record 2 channels.

Item (1) in following figure.
Step 4. Ensure that the recording device in the software is [STM32 AUDIO...].
Item (2) in following figure.

Figure 14. Audio editing software configuration for MicArrayCoupon application
1. Driver selection (WASAPI or MME)
2. Recording device (STM32 AUDIO…)

Step 5. Start recording the audio.

RELATED LINKS
To record the audio signal, you can use any audio editor software like Audacity
Appendix A  Serial Terminal configuration using TeraTerm

Figure 15. 8N1, 115200 bauds, no HW flow control configuration

![TeraTerm configuration screenshot]

Figure 16. Line endings to LF or CR-LF (Transmit) and LF (receive)

![TeraTerm configuration screenshot]
# Revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>05-Sep-2019</td>
<td>1</td>
<td>Initial release.</td>
</tr>
<tr>
<td>15-Nov-2019</td>
<td>2</td>
<td>Updated Introduction, Section 1 Overview, Section 2 How to program the board and Section 3.5 HS_Datalog application.</td>
</tr>
<tr>
<td>10-Feb-2020</td>
<td>3</td>
<td>Updated Introduction and Section 1 Overview. Added Section 3.3 UltrasoundFFT.</td>
</tr>
</tbody>
</table>
Contents

1 Overview .......................................................... 2
2 How to program the board .............................................. 3
3 Sample applications .................................................. 4
   3.1 Serial_DataLog application ...................................... 4
   3.2 OnboardMics application .......................................... 4
      3.2.1 Microphone acquisition process .......................... 5
   3.3 UltrasoundFFT ..................................................... 6
   3.4 WiFi_Connectivity application ................................... 8
   3.5 BLE_SampleApp application ..................................... 11
   3.6 HS_Datalog application ........................................... 11
      3.6.1 SD Card considerations .................................... 13
   3.7 MicArrayCoupon application ..................................... 13
Appendix A Serial Terminal configuration using TeraTerm .................... 16
Revision history ......................................................... 17
# List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SensorTile Wireless Industrial Node application diagram</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>STSW-STWINKT01 software block diagram</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>STLINK-V3MINI connected to STWIN core system board</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Audio editing software configuration for OnBoardMics application</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>UltrasoundFFT - power spectrum density vs. frequency</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>UltrasoundFFT - maximum energetic bin</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>UltrasoundFFT - save raw data</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>FFT values saved into a raw data file</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>STEVAL-STWINWFV1 expansion on core system board CN3 connector</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>Serial terminal output for Wi-Fi_Connectivity app status</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>HS_DataLog data flow</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>Folder structure in the SD card</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>STEVAL-STWINMAV1 expansion on core system board CN3 connector</td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>Audio editing software configuration for MicArrayCoupon application</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>8N1, 115200 bauds, no HW flow control configuration</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>Line endings to LF or CR-LF (Transmit) and LF (receive)</td>
<td>16</td>
</tr>
</tbody>
</table>