

Getting started with the AlgoBuilder application for the graphical design of algorithms

Introduction

AlgoBuilder is a graphical design application to build and use algorithms.

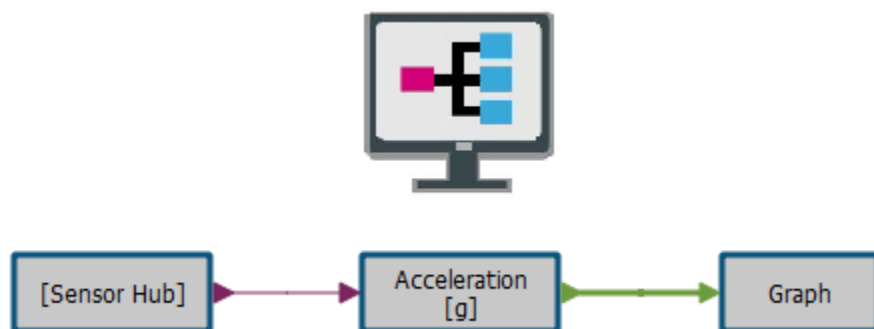
It quickly elaborates prototypes of applications for STM32 microcontrollers and MEMS sensors, including already existing algorithms (i.e. sensor fusion or pedometer), user-defined data processing blocks and additional functionalities.

The application eases the process of implementing proof of concept using a graphical interface without writing the code.

AlgoBuilder reuses previously defined blocks, combines multiple functionalities in a single project and visualizes data using **Unicleo-GUI** in real time using plot and display.

AlgoBuilder utilizes the STM32 ODE (Open Development Environment) ecosystem which combines hardware like **STM32 Nucleo** boards (**NUCLEO-F401RE** or **NUCLEO-L476RG**), **X-NUCLEO-IKS01A2** expansion board and software (STM32 HAL drivers, BSP structure, low and high-level sensor drivers) and **Unicleo-GUI**.

Figure 1. AlgoBuilder application diagram



1 Description

1.1 Overview

The main objectives of [AlgoBuilder](#) are:

- quick prototyping of applications for STM32 microcontrollers and MEMS sensors which already include existing algorithms (i.e., sensor fusion or pedometer), user-defined data processing blocks and additional functionalities
- easier process of implementing proof of concept using graphical interface without writing the code
- reuse of previously defined blocks
- combination of multiple functionalities in a single project
- visualization of data in [Unicleo-GUI](#) in real time using plot and display

The key features of the application include:

- Simple graphical design of algorithms (drag and drop, connect, set properties, build, upload)
- Wide range of function blocks available in libraries, including motion sensor algorithms (sensor fusion, gyroscope, magnetometer calibration and pedometer, for example)
- Building function blocks
- Automatic validation of design rules
- C code generation from the graphical design
- Use of external compilers (System Workbench for STM32, IAR EWARM, Keil μ Vision[®])
- Generated firmware output displayed through [Unicleo-GUI](#)
- Open XML format for function blocks and design storage
- Support for [NUCLEO-F401RE](#) or [NUCLEO-L476RG](#) with connected [X-NUCLEO-IKS01A2](#) expansion board and SensorTile [STEVAL-STLKT01V1](#)
- Network updates with automatic notification of new releases
- Free user-friendly licensing terms

[AlgoBuilder](#) utilizes STM32 ODE (Open Development Environment) ecosystem which combines [STM32 Nucleo](#) ([NUCLEO-F401RE](#) or [NUCLEO-L476RG](#)), [X-NUCLEO-IKS01A2](#) expansion board and software (STM32 HAL drivers, BSP structure, low and high-level sensor drivers) and [Unicleo-GUI](#).

1.2 Prerequisites

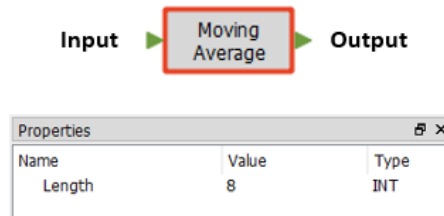
The following software and hardware are needed to fully exploit the functions of [AlgoBuilder](#).

- One of the following IDEs:
 - System Workbench for STM32 ([SW4STM32](#)) v1.13.1 or newer
 - [IAR-EWARM](#) 7.80.4 or newer
 - Keil μ Vision 5.22 or newer
- [Unicleo-GUI](#)
- STM32 ST-LINK utility ([STSW-LINK004](#))
- STM32 Virtual COM Port Driver ([STSW-STM32102](#))
- [NUCLEO-F401RE](#) or [NUCLEO-L476RG](#) with [X-NUCLEO-IKS01A2](#) or SensorTile [STEVAL-STLKT01V1](#)

1.3 Terms and references

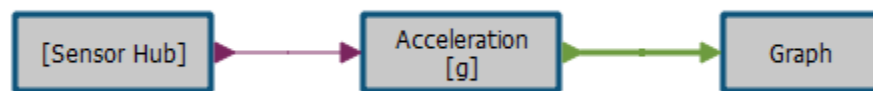
- **Function block** is a data processing element with one or multiple inputs or outputs. It processes inputs and generates outputs and can have one or more properties.

Figure 2. AlgoBuilder function block



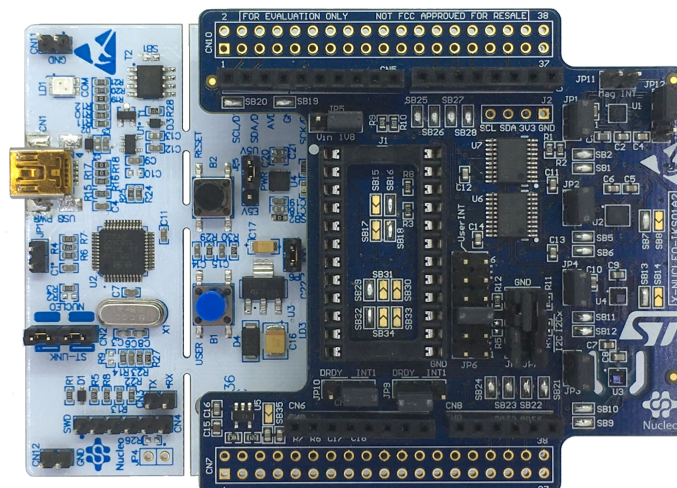
- **Design** is a set of several function blocks connected together.

Figure 3. AlgoBuilder design



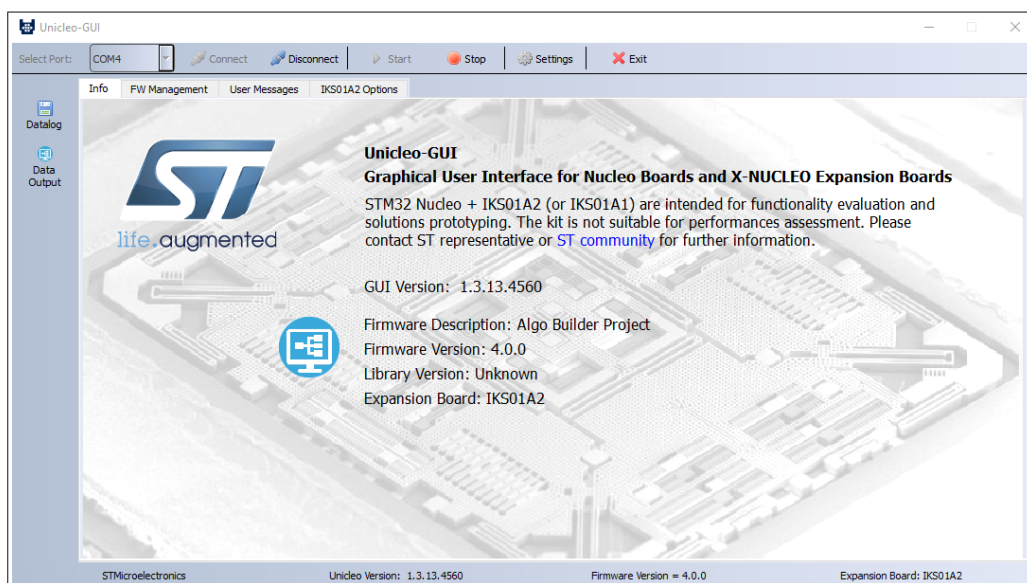
- **Node** represents the connection between two function blocks.
- **Firmware** for STM32 microcontroller can be built from the design.
- **STM32 Nucleo** development board with an STM32 microcontroller used for design testing.
- **X-NUCLEO-IKS01A2** motion MEMS and environmental sensor expansion board which embeds accelerometer, gyroscope, magnetometer, temperature, humidity and pressure sensors.

Figure 4. STM32 Nucleo (NUCLEO-F401RE) plus X-NUCLEO-IKS01A2



- **Unicleo-GUI** can be used to display the firmware outputs.

Figure 5. Unicleo-GUI



1.4 Principle of operation

The workflow starts from the graphical design of the desired functionality by using a simple "drag and drop" approach.

You can use the predefined function blocks provided in the form of libraries.

You can also create a custom function block. Some function block properties can or must be adjusted in order to run (in the example, filter coefficients are defined in the filter function block properties). Then, you can interconnect the compatible function blocks using nodes.

AlgoBuilder automatically checks the compatibility between input and output and allows connecting only terminals with the same type and dimension.

When the design is finished, AlgoBuilder generates the C code from the defined graphical design.

The final firmware project is created from the C code generator combined with pre-prepared firmware templates and binary libraries.

The project can be compiled using an external compiler tool and the most common Integrated Development Environments (IDEs) are supported (System Workbench for STM32 with GCC compiler, Keil µVision, IAR Embedded Workbench).

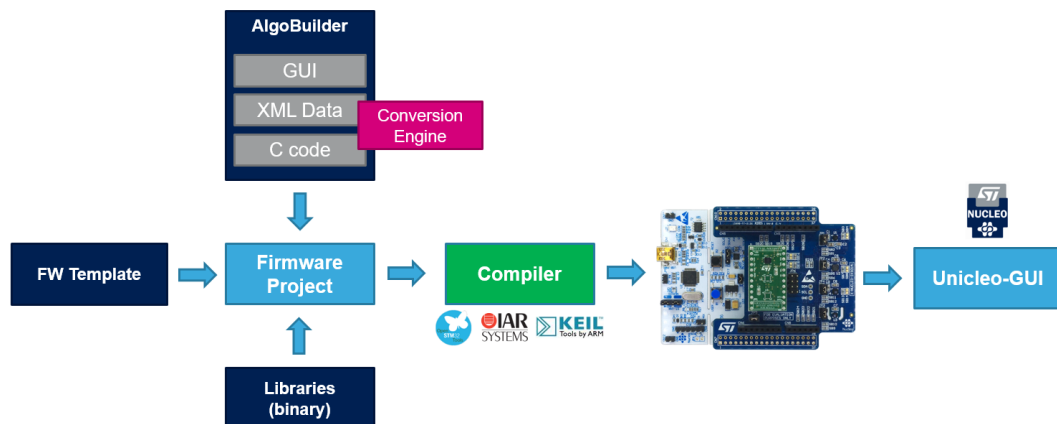
An STM32 Nucleo board is then programmed by the generated binary file. When the firmware is executed it starts reading data from the selected sensor, process the data via the algorithm and sends results to Unicleo-GUI application.

During the graphical design, you can select how to see the results. Graphs, logical analyzer, bar charts, 3D plot, scatter plot, histogram, teapot, FFT plot and text values are supported.

During the startup, the firmware configures the Unicleo-GUI to display in the desired format.

The graphical designs as well as the libraries are stored as XML files.

Figure 6. Algebuidler principle of operation

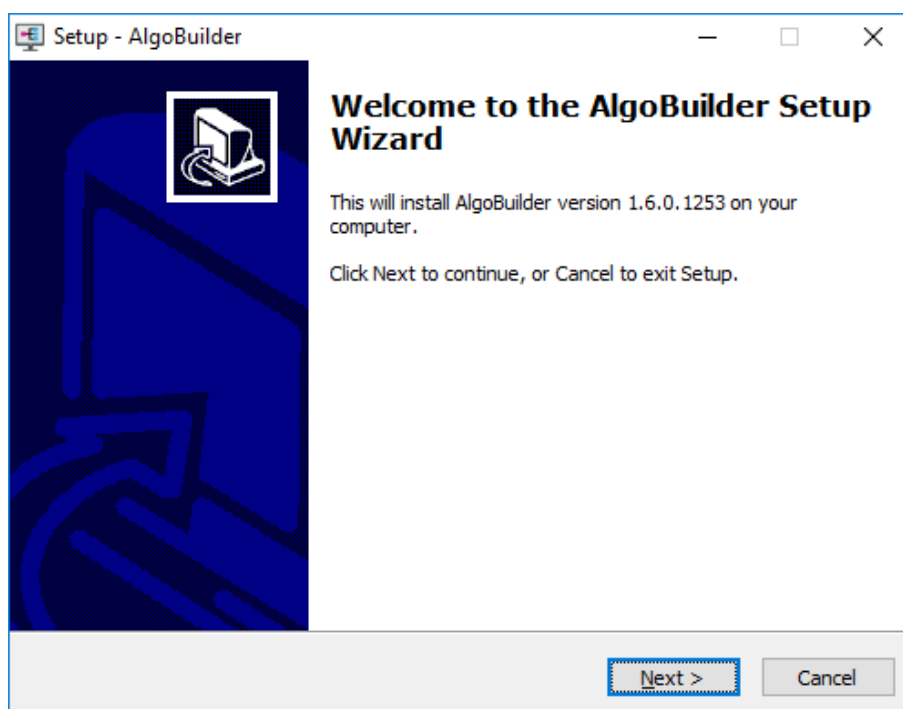


2 Getting started

2.1 Installing the software

The [AlgoBuilder](#) software is designed to run in Microsoft® Windows. To install the application, run *Setup_AlgoBuilder.exe*, follow the instructions and execute AlgoBuilder once the installation is complete.

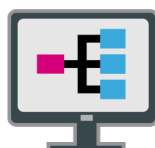
Figure 7. AlgoBuilder installer



2.2 Running the software for the first time

The installer may have created a shortcut on your Windows desktop and/or Windows start menu. The AlgoBuilder can be run by double clicking on the shortcut. If the shortcuts were not created, you can run the AlgoBuilder by executing *AlgoBuilder.exe* file which is located in the directory where the application was installed (default location is C:\Program Files (x86)\STMicroelectronics\AlgoBuilder).

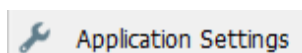
Figure 8. AlgoBuilder icon



2.3 Application settings

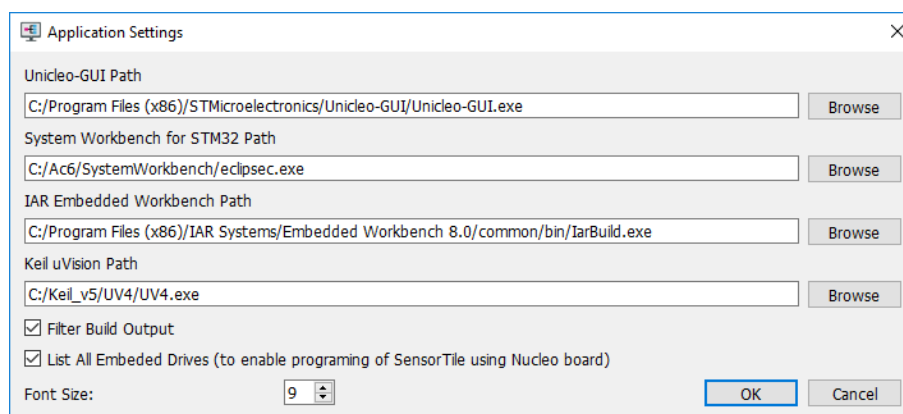
You can adjust the **AlgoBuilder** configuration in File→Application Settings.

Figure 9. Application Settings menu option



Step 1. Specify the path to **Unicleo-GUI**.

Figure 10. Application Settings window



If the path (**Unicleo-GUI.exe**) is properly set, the Unicleo-GUI can be quickly executed from the toolbar or the AlgoBuilder menu. If the path is not set, the corresponding icon in the toolbar and item in the menu are disabled.

Step 2. Specify the path to at least one IDE.

For System Workbench for STM32, put the path to **eclipse.exe**, for IAR Embedded Workbench to **IarBuild.exe** and for Keil μ Vision to **UV4.exe**.

Step 3. Set up the application behavior.

If **Filter Build Output** is enabled, AlgoBuilder automatically filters outputs from the external compiler and makes them more readable in the console.

If you are going to use SensorTile, select **List All Embedded Drives** to allow SensorTile programming through various STM32 NUCLEO boards (ST-LINK V2.1 programmer).

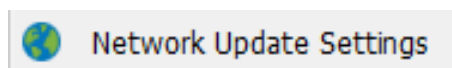
The **font size** can be adjusted as well, for better readability on high-resolution monitors.

2.4 Network update settings

The application is able to check and notify if a new version is available. You can then decide whether to download and install the new version.

Some functional network parameters have to be properly set in **Network Update Settings** in the **File** menu.

Figure 11. Network Update Settings menu option



The setting dialog is divided in different sections:

1. In the first section you can choose between manual and automatic check. In the latter, you can adjust the periodicity. If an interval of zero days is set, a check for updates is performed at every application start. To run an immediate check for updates, click on **Check Now**.
2. The second section contains options for proxy server type settings.
*Tip: When you select **Use System Proxy Parameters**, it is usually necessary to open a web browser to run all security scripts before running a check for updates.*
3. The third section contains the proxy manual configuration field where the proxy HTTP name and port number can be entered.
4. The last section contains the authentication credential fields (if required)

The **Check Connection** button can be used to check if the update server is accessible.

Figure 12. Network Update Settings window

A screenshot of the 'Network Update Settings' dialog box. The window has a title bar with a close button. It is divided into four main sections: 1. 'Check and Update Settings' with radio buttons for 'Manual Check' and 'Automatic Check' (selected), a 'Check Now' button, and a text field for 'Interval between two checks [days]' set to '1'. 2. 'Proxy Server Type' with radio buttons for 'No Proxy', 'Use System Proxy Parameters' (selected), and 'Manual Configuration of Proxy Server'. 3. 'Manual Configuration of Proxy Server' with text fields for 'Proxy HTTP' and 'Port'. 4. 'Authentication' with checkboxes for 'Require Authentication' and 'Remember My Credentials', and text fields for 'User Login' and 'Password'. At the bottom, there are buttons for 'OK', 'Cancel', and a red 'X' 'Check Connection' button.

3 Using AlgoBuilder

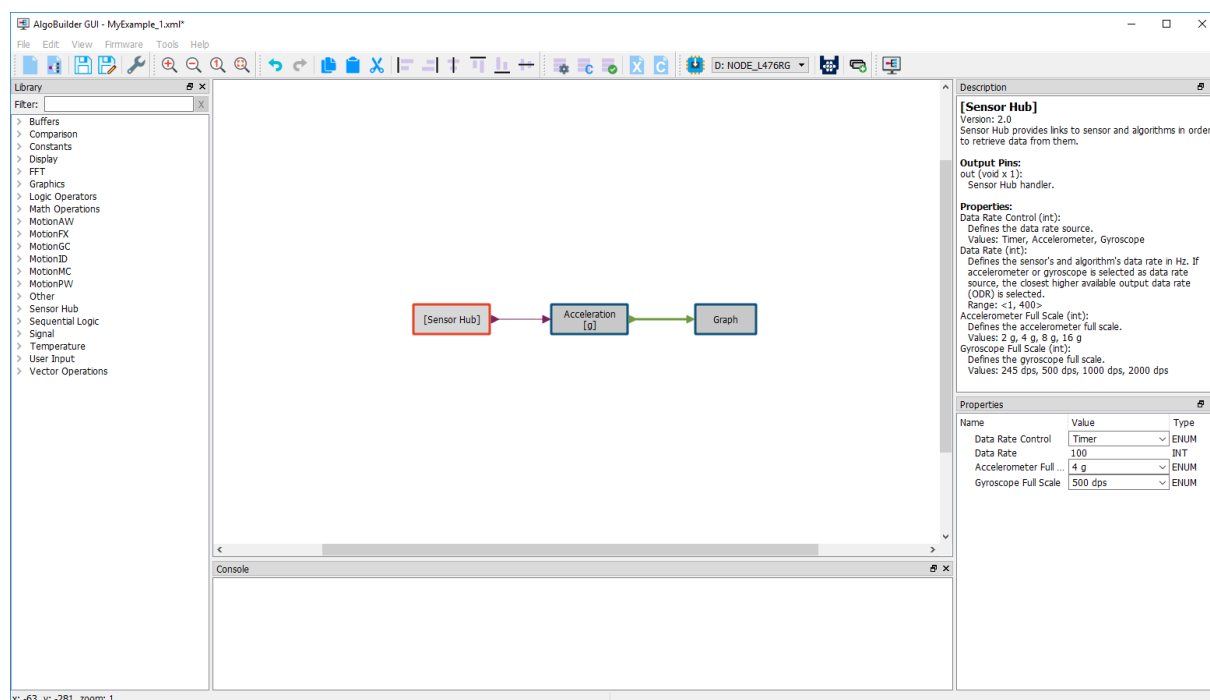
The **AlgoBuilder** main window contains:

- a central **Workspace** where the algorithm is designed using function blocks
- a **Library** dock with a list of available libraries and their function blocks which can be dragged and dropped to the workspace window
- a **Description** dock which displays information about the selected component (function block, connection, etc.)
- a **Properties** dock which displays all available properties of the selected function block
- a **Console** dock which displays messages from the AlgoBuilder or an external compiler

The AlgoBuilder application has a standard menu and a toolbar to speed up access to frequently used functions.

Note: You can change the position of all docks and the toolbar. Docks can be opened and closed in the View menu.

Figure 13. AlgoBuilder main window



3.1 Workspace

The developed algorithm design is created in the workspace area.

- Step 1.** Place the necessary function blocks on the workspace
Note: Function blocks can be simply dragged from the library dock and dropped in the workspace.
- Step 2.** Set their properties
- Step 3.** Connect them by clicking and holding the mouse left button on the output you want to connect and move the cursor to the input where the connection should be made. The connection can be created also in the opposite way from input to output.
Note: You can connect only inputs and outputs of the same type and size. If you try to connect different types or sizes, the console displays an error message.
You can change the number of inputs for some function blocks (e.g. MUX, Sum, And, Or...).
- Step 4.** Use **Delete** to remove any component, **Cut**, **Copy** and **Paste** for any part of the design through the **Edit** menu, the **Toolbar** or the shortcut.
- Step 5.** Align the function blocks to the right, left, top or bottom.
The last selected function block determines the final position.
- Step 6.** Use **Do** and **Undo** to go back and forward in the performed operations in the workspace.
- Step 7.** To **Zoom In** or **Zoom Out** use Ctrl and the mouse wheel or the appropriate function in **View** menu or in **Toolbar**.
- Step 8.** Select **Fit All** to fit the whole design on the screen.
- Step 9.** Select **Zoom 1:1** to set zoom factor to 1.
- Step 10.** Right click and hold on the workspace area to explore the content of the design.

3.2 Library dock

The **Library** dock gives you access to all the available libraries and function blocks located in a particular library. **AlgoBuilder** scans *[Install path]/Library/* and the user's home directory *\STMicroelectronics\AlgoBuilder\Library* during startup and loads all valid libraries located there.

The **Graphics** library is not stored in an xml file but it is automatically added by AlgoBuilder.

3.3 Description dock

The **Description** dock provides information about the component selected in the **Workspace** or in the **Library** dock.

If you select a function block, the following information is shown:

- Name
- Version
- Description of the function block functionality
- Type, size and functionality of all inputs
- Type, size and functionality of all outputs
- Description of all function block properties

3.4 Properties dock

If a function block has a property or properties, they are displayed in the **Properties** dock.

Each property has name, value and type fields.

The values can be modified.

The **AlgoBuilder** automatically checks if the value is valid and does not allow setting an invalid one (for example, a value out of an available range).

For the **STRING** type, the **%** character is forbidden and is automatically deleted.




3.5

Toolbar

The **Toolbar** provides quick access to the most commonly used functions. The position of the toolbar and the order of the function can be adjusted.

Table 1. AlgoBuilder toolbar default functions

Toolbar icon	Function
	Creates new design
	Open existing design
	Save design
	Save design as different file
	Open Application Settings window
	Zoom In
	Zoom Out
	Set zoom to 1:1 ratio
	Fit all design into screen
	Undo
	Redo
	Copy
	Paste
	Cut
	Align to the left
	Align to the right
	Align to the top
	Align to the bottom

Toolbar icon	Function
	Align to center horizontally
	Align to center vertically
	Open Firmware Settings window
	Generates C Code from the graphical design
	Build firmware
	Show the design xml source file in the default text editor
	Show C Code in the default text editor
	Program STM32 Nucleo board
	Run Unicleo-GUI application
	Open About window
	Open Function Block Creator

4 Data types

AlgoBuilder works with four data types:

- **FLOAT** represents real numbers and is used for floating-point arithmetic (for example, in the acceleration function block output). In C code, the representation float variable is used. The size is 4 bytes.
- **INT** represents integer numbers (for example, in the counter function block). In C code, int32_t variable is used. The size is 4 bytes.
- **VARIANT** is used for inputs of a set of function blocks; the variant changes its type on the basis of the type of output connected to this input (for example, the variant type is used for inputs of comparison function blocks).
- **VOID** is used exclusively for the connection between Sensor Hub and its data outputs. This type cannot be visualized.

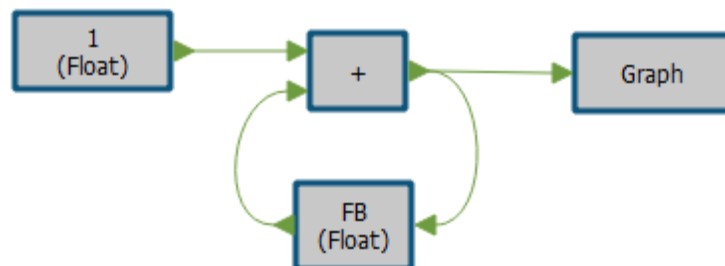
Each input or output is characterized by its type and size: the thickness of the connection line indicates the size of input and output and color of the connection line indicated the type.

The size value can be changed in the properties.

Important: Only input and output with the exact same type and size can be connected together. The only exception is the VARIANT input which gets the type of connected output.

It is not possible to connect the input and output of the same function block. If this is desired, the **Feedback** function block for the particular data type needs to be used. The Feedback function block has an Init value, which defines the output value of the block for the first run.

Figure 14. Using the Feedback function block



5 Conditional Execution

In some cases, it is needed to execute the function block operation only if a certain condition is valid. For this case it is possible to add **Conditional Execution Input** to the selected function block. This input then defines if the function block code will be executed or not. This is represented as an if statement in the generated C code. To add or remove the conditional execution input, click on the function block using the right mouse button.

Figure 15. Conditional Execution



6 Libraries

The following libraries are available for a fresh [AlgoBuilder](#) installation.

Table 2. AlgoBuilder installation libraries

Library	Content
Buffers	Function blocks for operations with data buffers
Comparison	Function blocks for two value comparison (e.g. >, <, =, etc.)
Constants	Function blocks for constant definition
Display	Function blocks for data visualization in Unicleo-GUI application
FFT	Function blocks related to FFT analysis
Graphics	Text note to annotate the design
Logic Operators	Function blocks for logic operations (e.g. And, Or, ..., etc.)
Math Operations	Function blocks for various mathematical operations (e.g. +, -, /, etc.)
Other	Auxiliary function blocks (e.g. mux, demux, type conversion, etc.)
Sensor Hub	Main Sensor Hub function block, which provides access to the sensors and pre-build algorithm and function blocks for data acquisitions from connected sensors.
Sequential Logic	Function block for sequential logic (flip flops)
Signal	Function blocks for signal processing (e.g. filters, etc.)
Temperature	Function blocks for temperature units conversion
User Input	Function blocks which allow user to send arbitrary data to the running firmware at real-time through Unicleo-GUI application
Vector Operations	Function blocks for vector operation (e.g. calculate magnitude, etc.)

Already prepared algorithms in binary form can be also integrated in the user design.

Table 3. AlgoBuilder supported libraries

Library	Functionality
MotionAW	Activity recognition algorithm for wrist-worn devices
MotionFX	Sensor fusion algorithm
MotionGC	Real-time gyroscope calibration algorithm
MotionID	Motion intensity detection algorithm
MotionMC	Real-time magnetometer calibration algorithm
MotionPW	Pedometer algorithm for wrist-worn devices

7 Creating your first design

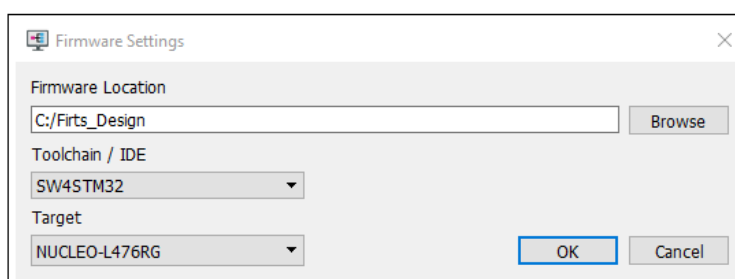
As a first example, you can create a design to read acceleration from the accelerometer sensor at a selected data rate and send data to [Unicleo-GUI](#) to be visualized in a time chart.

Step 1. Start with blank design by clicking on the icon (Create new design) in the toolbar or the **File** menu.

The Firmware Settings window is opened automatically or you can open it by clicking on the icon (Firmware settings) in the toolbar or **Firmware** menu.

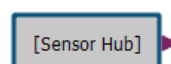
Step 2. Set the path to the directory where the output firmware is located, the IDE to be used to build the firmware and the target to be used for testing. It is important to select the correct target ([NUCLEO-F401RE](#) or [NUCLEO-L476RG](#) or [SensorTile](#)).

Figure 16. AlgoBuilder Firmware Settings window



Step 3. Drag the [Sensor Hub] function block from the Sensor Hub library and drop it into the workspace. *Important: Each design must start with the Sensor Hub function block, which provides access to the sensors and pre-build algorithm.*

Figure 17. Sensor Hub function block



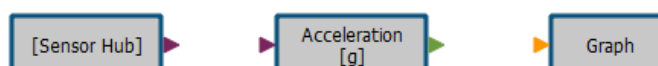
Step 4. Adjust Sensor Hub properties, select Timer as the source for Data Rate Control, set the Data Rate to 50 Hz and Accelerometer Full Scale to 2 g.

Figure 18. Sensor Hub properties

Properties		
Name	Value	Type
Data Rate Control	Timer	ENUM
Data Rate	50	INT
Accelerometer Full ...	2 g	ENUM
Gyroscope Full Scale	500 dps	ENUM

Step 5. Add Acceleration [g] function block from the Sensor Hub library and Graph function block from the Display library to the workspace.

Figure 19. Sensor Hub, Acceleration[g], Graph function blocks



Step 6. Adjust Graph properties.

The Number of Curves in Graph defines the size of its input. The value needs to be changed to 3 to match the Acceleration [g] output size. The Graph, Waveform and Unit names can be also changed.

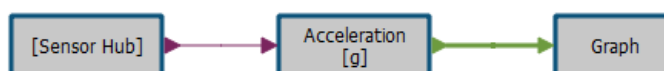
Figure 20. Graph properties

Name	Value	Type
Number of Curves	3	INT
Graph Name	Acceleration	STRING
Waveform 1 Name	X axis	STRING
Waveform 2 Name	Y axis	STRING
Waveform 3 Name	Z axis	STRING
Unit Name	[g]	STRING
Zero axis position	Middle	ENUM
Auto-scale	ON	ENUM
Full Scale	4	STRING

Step 7. Connect the function blocks by clicking on [Sensor Hub] output, holding and moving your mouse to the input of the Acceleration [g] block.

Step 8. Repeat the previous step to connect Acceleration [g] to Graph block.

Figure 21. Sensor Hub, Acceleration[g], Graph function blocks



Your design is ready and you can generate C code from it.

Step 9. Click on the icon (Generate C Code) in the toolbar or **Firmware** menu.


This function copies the firmware template into the previously selected file and creates the algo_builder.c file which is C code representation of the graphical design. If the operation is successful, the message "Code generation finished successfully" appears in the console. You can check the generated C code by clicking on the icon  (Show C Code) in the toolbar or **Firmware** menu.

Figure 22. Generated code in algo_builder.c file

```

#include "algo_builder.h"


void *Sensor_Hub_2_out;
float Acceleration_g_2_data[3];

sDISPLAY_INFO display_info_list[] = {
{INFO_TYPE_GRAPH,1,3,VAR_TYPE_FLOAT,0,"graph|Acceleration|[g]|1|19|X axis|Y axis|Z axis|||1",0},
{0,0,0,0,0,0,0}};

void AB_Init(void)
{
    Sensor_Hub_Init(0, 100, 1, 1);
    Accelerometer_Init();
    Message_Length = 12;
}

void AB_Handler(void)
{
    Sensor_Hub_Handler(&Sensor_Hub_2_out);
    Accelerometer_GetData(Sensor_Hub_2_out, Acceleration_g_2_data);
    Display_Update(Acceleration_g_2_data, &display_info_list[0]);
}

```

*Note: In case the firmware template in the target directory is accidentally broken or deleted you can invoke re-initialization of the firmware template by clicking on the icon  (Re-initialize Firmware) in the **Firmware** menu.*


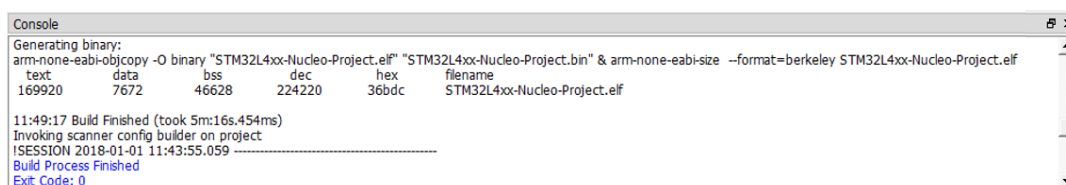
- Step 10.** Click on the icon  (Build Firmware) to call the external IDE to build the firmware project and generate a binary file for the STM32 microcontroller.
- The console shows an output from the compiler. Once the compilation finishes, a “Build Process Finished” message appears. If there is no error message from the compiler, the firmware is ready to be programmed in the STM32 Nucleo board.

Figure 23. System Workbench for STM32 output



*Note: Only files which were changed are compiled during the firmware building process. To use the external tool to recompile all files, select the icon  (Rebuild Firmware) in the **Firmware** menu.*

- Step 11.** Save the design by clicking on the icon  (Save Design) in the toolbar of the **File** menu.

8 Programming the target

8.1 STM32 Nucleo board

AlgoBuilder automatically scans for connected STM32 Nucleo boards.


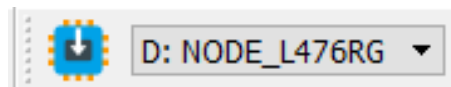
The list of connected STM32 Nucleo boards is available in the toolbar next to the "Program STM32 Nucleo" icon. If the firmware is successfully built and an STM32 Nucleo board is selected, you can program the board by pressing the button  (Program STM32 Nucleo).

Figure 24. Program STM32 Nucleo selection box



8.2 SensorTile

SensorTile does not have a built-in programmer. To program the board, connect an external ST-LINK to the SWD connector on the cradle using the 5-pin flat cable which is provided in the SensorTile Kit package.

The easiest way to obtain an ST-LINK device is to get an STM32 Nucleo board which bundles an ST-LINK V2.1 debugger and programmer. Both CN2 jumpers on the STM32 Nucleo board have to be removed to program SensorTile.


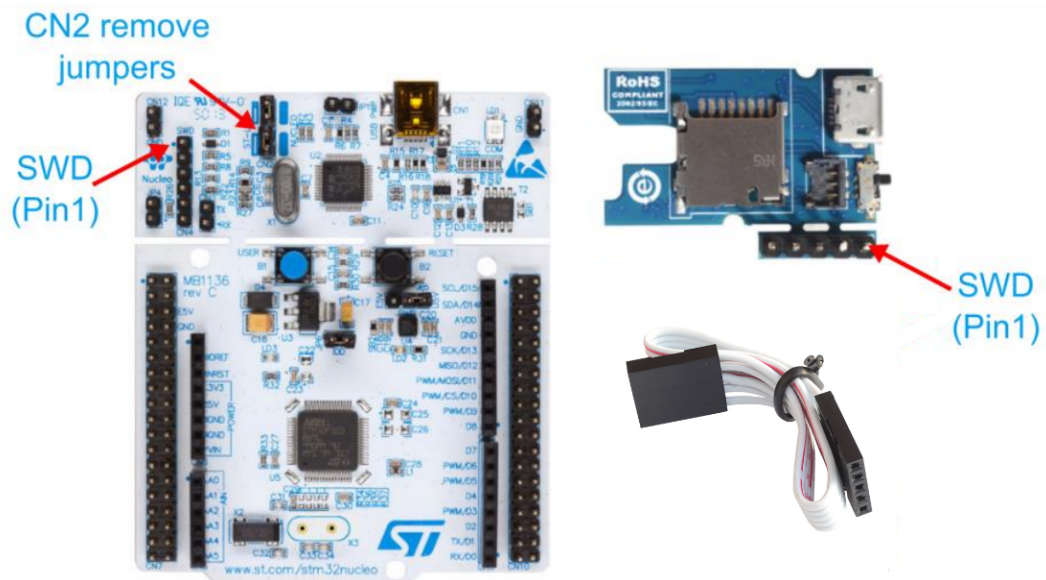
To see all STM32 Nucleo boards in the AlgoBuilder selection box select **List All Embedded Drives** in the AlgoBuilder settings. Then you can select the STM32 Nucleo board which you are going to use to program SensorTile and press the program button .

Figure 25. STM32 Nucleo board, SensorTile cradle SWD connectors



9 Using Unicleo-GUI

Unicleo-GUI can be used to check the functionality of the firmware. You can visualize data coming from the firmware and send data from Unicleo-GUI to the running firmware.

9.1 Data visualization

There are eight types of data visualization:

- **Bar Graph**
- **3D Teapot Model**
- **Graph**
- **Histogram**
- **Logic Analyzer**
- **3D Plot**
- **Scatter Plot**
- **Text Value**

To send data to Unicleo-GUI, the appropriate function block needs to be added to the design.

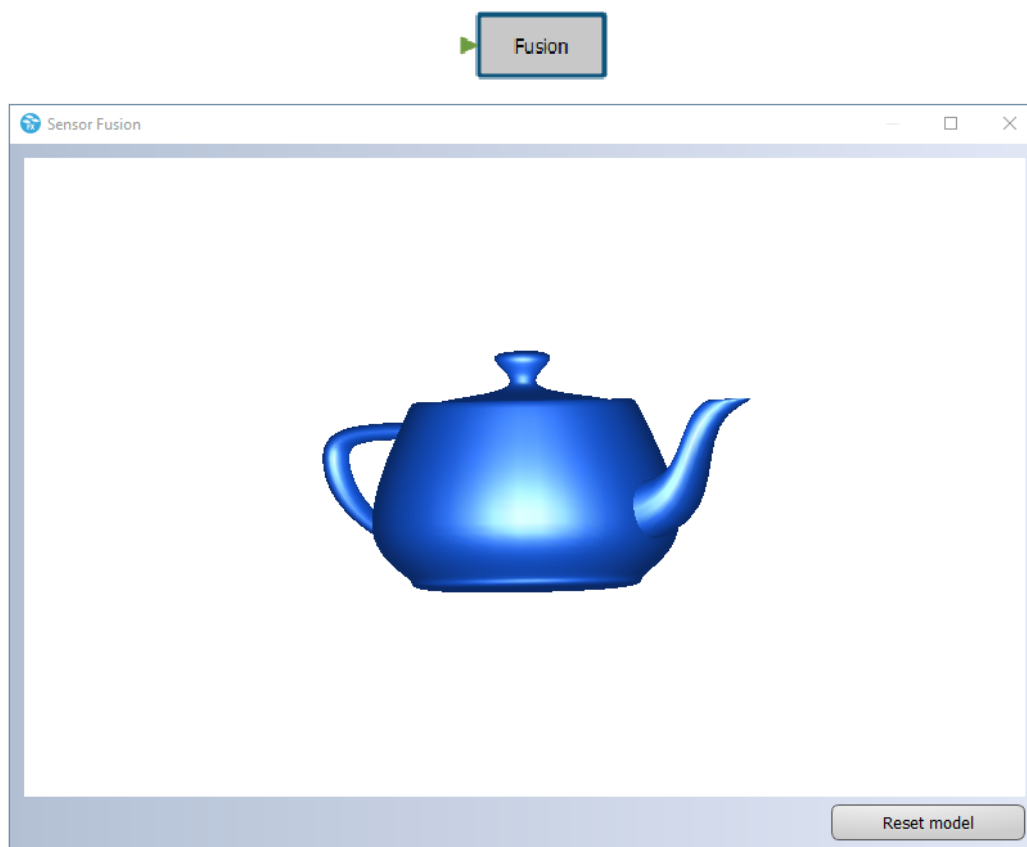
- Add the **Bar** function block from the **Display** library to your design to display data as a bar graph. A bar graph is suitable for a quick check of an actual value without needing to see the history. This graph works with any floating or integer value and each of them can have up to 6 bars. In the properties field, you can set the name of the graph, of each bar, unit on the Y-axis, position of the 0 on Y axis, full scale and enable or disable auto-scale.

Figure 26. Bar function block and example of data visualization in Unicleo-GUI



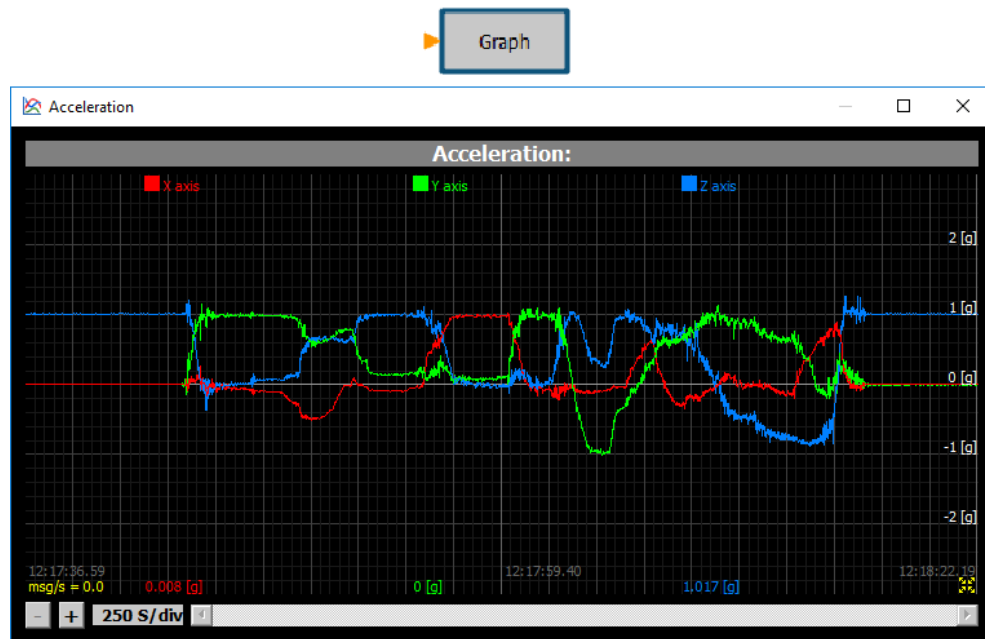
- Add the **Fusion** function block from the **Display** library to your design to display quaternion data as the teapot 3D model.
The 3D Teapot Model is usually used to check device orientation in 3D space. This graph requires quaternions which can be obtained for example from the sensor fusion (MotionFX) algorithm.

Figure 27. Fusion function block and example of data visualization in Unicleo-GUI



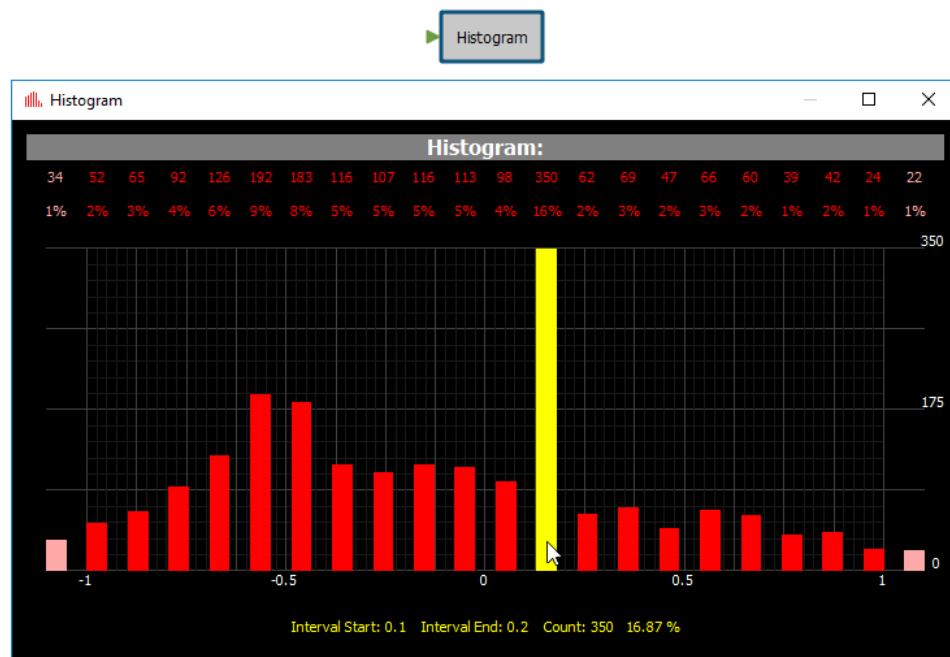
- Add the **Graph** function block from the **Display** library to your design to display data as a time graph. This graph works with any floating or integer value and each of them can have up to 6 waveforms. In the properties field, you can set the name of the graph, of each waveform, unit on the Y-axis, position of the 0 on Y axis, full scale and enable or disable auto-scale.

Figure 28. Graph function block and example of data visualization in Unicleo-GUI



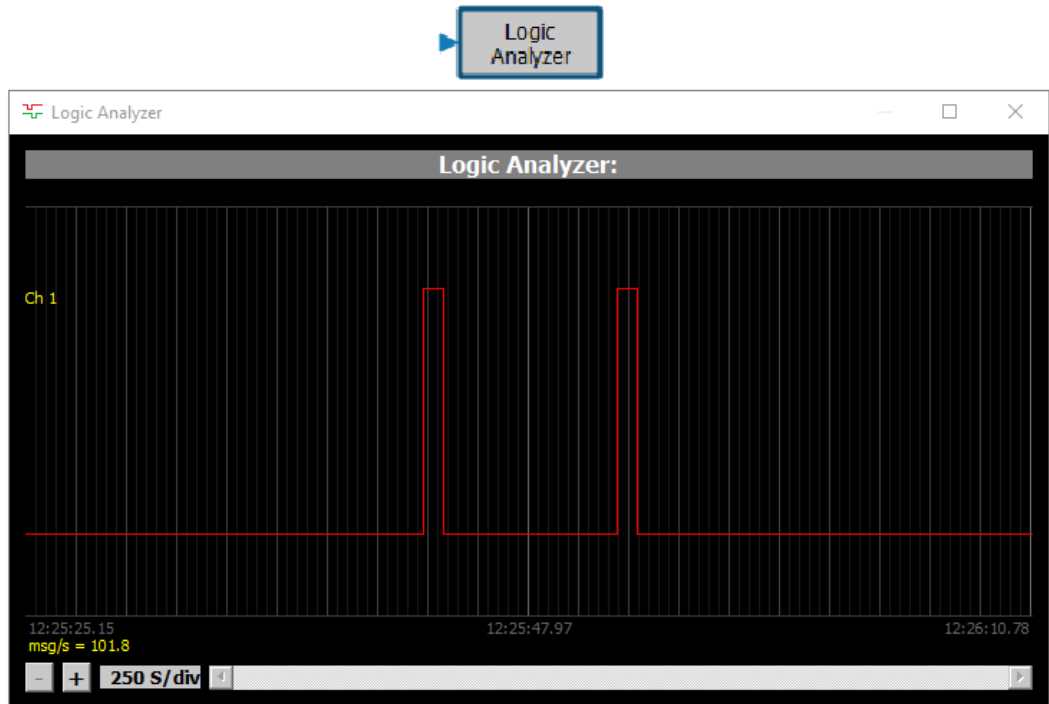
- Add the **Histogram** function block from the **Display** library to your design to display the data distribution chart. The histogram can be used to see the distribution of the selected value. In the properties field, you can set the name of the graph, number of intervals, zero axis position, full scale and enable or disable auto-scale.

Figure 29. Histogram function block and example of data visualization in Unicleo-GUI



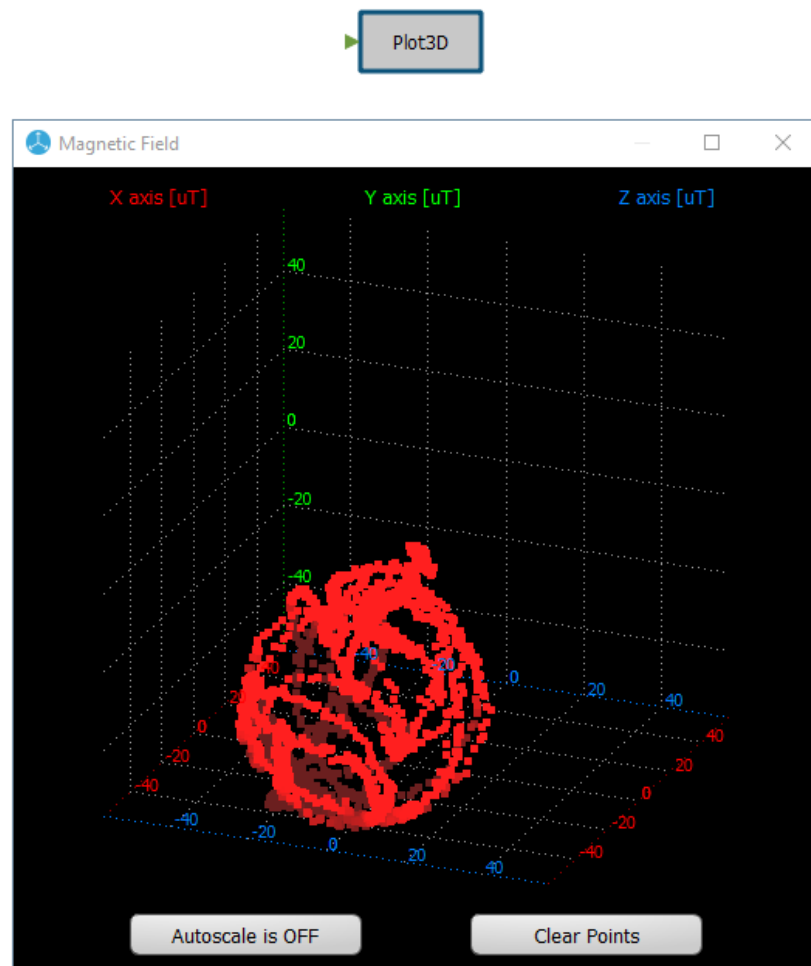
- Add the **Logic Analyzer** function block from the **Display** library to display logic signals which can have values of only 0 or 1.
The logic analyzer can have up to 8 channels. In properties, you can change the name of each channel.

Figure 30. Logic Analyzer function block and example of data visualization in Unicleo-GUI



- Add the **Plot3D** function block from the **Display** library to display X,Y,Z data in the 3D chart

Figure 31. 3D Plot function block and example of data visualization in Unicleo-GUI



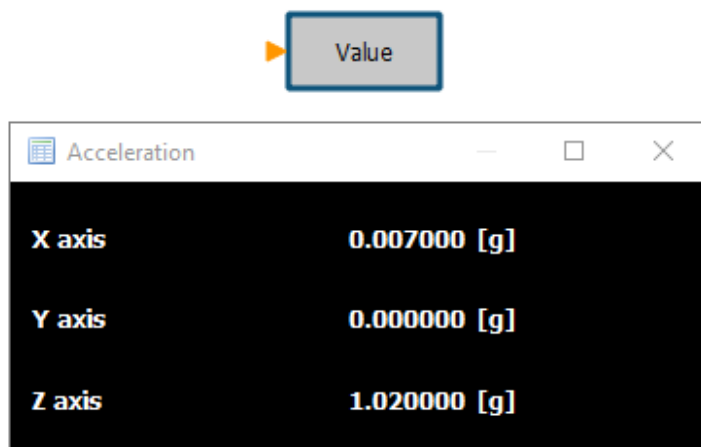
- Add the **Scatter Plot** function block from the **Display** library to display X, Y, Z data in the 2D X-Y, X-Z, Y-Z chart

Figure 32. Scatter Plot function block and example of data visualization in Unicleo-GUI



- Add the **Value** function blocks from the **Display** library to display the exact float or integer value
Each function block can display up to 8 values. In properties, you can change the name of the value and unit for each item.

Figure 33. Value function block and example of data visualization in Unicleo-GUI



9.2 Data input

Three types of data can be sent from **Unicleo-GUI** to running firmware: **Binary**, **Integer** and **Float**.

Add the Input Value function block (with appropriate type) from the **User Input** library to your design.

Each Input Value function block can represent up to 4 values. In properties, you can set the name of each value and default value. The Input Value function block output size is defined by the Number of Values property.

Figure 34. Input Value function blocks

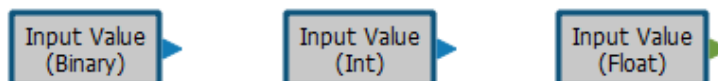


Figure 35. Input Values in Unicleo-GUI

Integer Values

Value 1	0
Value 2	0
Value 3	0
Value 4	0

Float Values

Value 1	0
Value 2	0
Value 3	0
Value 4	0

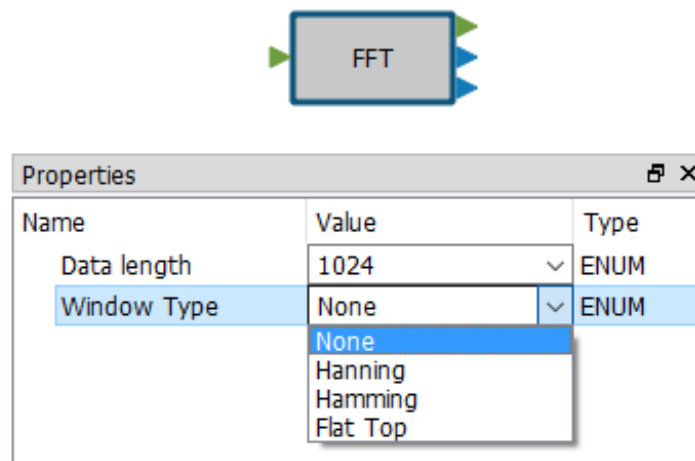
Binary Values

Value 1
Value 2
Value 3
Value 4

10 FFT (Fast Fourier Transform)

AlgoBuilder offers also a function block for frequency analysis of the sensor's output signal using FFT (Fast Fourier Transform). Fourier analysis converts a signal from the time domain to a representation in the frequency domain.

Figure 36. FFT function block



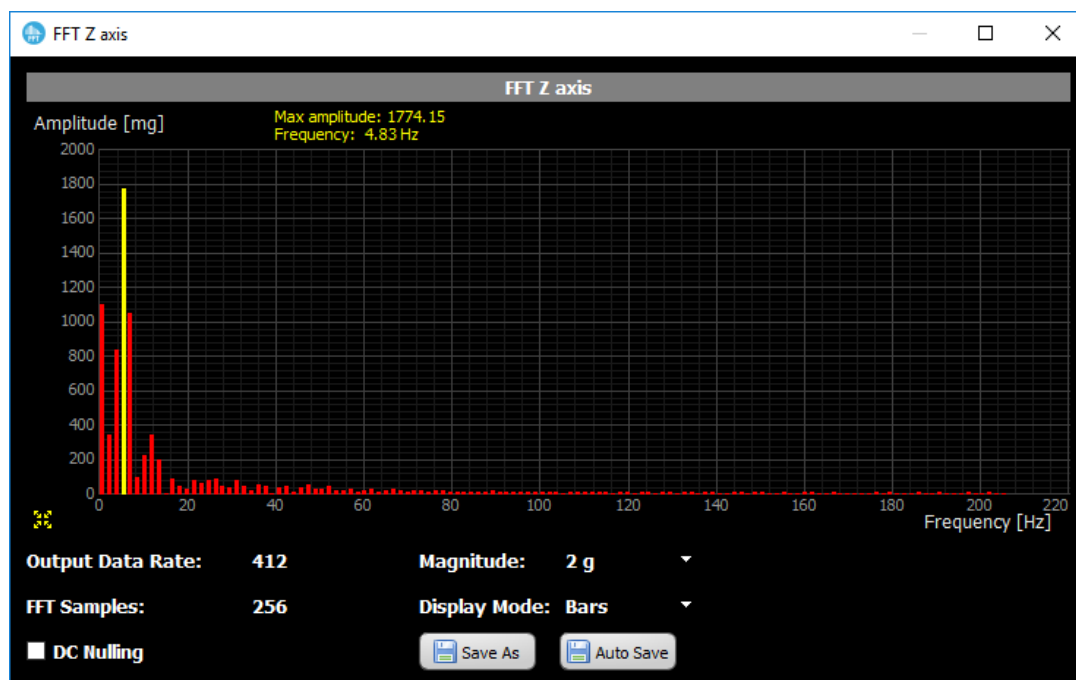
The **FFT** function block offers frequency analysis from 32, 64, 128, 256, 512 and 1024 samples. It is also possible to enable window usage to eliminate spectrum leakage. Hanning, Hamming, and Flat Top windows can be used.

Output from the FFT function block can be connected to the **FFT plot** function block, which will send the results to Unicleo-GUI. Unicleo-GUI then displays the data as a frequency spectrum. To send data to Unicleo-GUI only when the FFT calculation is finished, conditional execution input must be added to the FFT plot and connected to the FFT function block.

Figure 37. FFT and FFT Plot connections



Figure 38. Frequency spectrum in Unicleo-GUI



Note: To get the correct frequency values, it is necessary to select the sensor whose data are analyzed in the data rate control device in the Sensor Hub. Real sensor ODR (output data rate) is measured during firmware initialization.

11 Creating your own function block

A function block is defined as XML records in an XML file.

Each XML file represents an [AlgoBuilder](#) library, which may contain one or more function blocks.

A library has the following structure, where each function block is described inside the `<Block></Block>` tags.

```
<?xml version="1.0" encoding="UTF-8"?>
<Library>
<Version>1.0</Version>
<Name>Comparison</Name>
<Block>
</Block>
</Library>
```

Each function block must have the XML tags listed below.

Table 4. Function block XML tags

Tag	Description
<code><Name></Name></code>	Name of the function block displayed in the list of the Library dock.
<code><DisplayName></DisplayName></code>	Text displayed in the function block rectangle when placed in the workspace. Two hash signs (##) can be used to break the text in a new line.
<code><Version></Version></code>	Version of the function block.
<code><Description></Description></code>	Description of the overall function block functionality displayed in the Description dock.
<code><Width></Width></code>	Width of the function block rectangle.
<code><Height></Height></code>	Height of the function block rectangle.
<code><X></X></code>	X position of the function block: it must be 0.
<code><Y></Y></code>	Y position of the function block: it must be 0.
<code><Instances></Instances></code>	The maximum number of a particular function block in one design. Put 0 value for the unlimited number of instances.
<code><Inputs></Inputs></code>	List of all inputs.
<code><Input></Input></code>	Input definition with <code><Type></code> , <code><Size></code> , <code><Name></code> , <code><Description></code> tags.
<code><Type></Type></code>	Data type of the particular input. The following types can be used: INT, FLOAT, VARIANT
<code><Size></Size></code>	Data size of the particular input.
<code><Name></Name></code>	Name of the particular input.
<code><Description></Description></code>	Description of the particular input.
<code><Outputs></Outputs></code>	List of all outputs.
<code><Output></Output></code>	Output definition with <code><Type></code> , <code><Size></code> , <code><Name></code> , <code><Description></code> tags.
<code><Type></Type></code>	Data type of the particular output. The following types can be used: INT, FLOAT, VARIANT
<code><Size></Size></code>	Data size of the particular output.
<code><Name></Name></code>	Name of the particular output.
<code><Description></Description></code>	Description of the particular output.
<code><Properties></Properties></code>	List of all properties.

Tag	Description
<Property></Property>	Property definition with <Type>, <Name>, <Value>, <Description> tags.
<Type></Type>	Data type of the particular property. The following types can be used: INT, FLOAT, STRING
<Name></Name>	Name of the particular property.
<Value></Value>	Default value of the particular property.
<Description></Description>	Description of the particular property.
<Memory></Memory>	List of all memory items.
<Item></Item>	Memory item definition.
<Type></Type>	Data type of the particular memory item. The following types can be used: INT, FLOAT
<Size></Size>	Data size of the particular memory item.
<Name></Name>	Name of the particular memory item.
<Command></Command>	C language commands representing the function block. Each command must be placed on a new line between <Line></Line> tags. The value of any input, output or property can be used in the commands. For this put % before and after the name of the input, output or property.
<Line></Line>	
<Function></Function>	It is possible to define a complete function to be used in the <Command> tag. Each command must be placed on a new line between <Line></Line> tags.
<Line></Line>	
<InitCode></InitCode>	C language commands executed once, during firmware initialization.

11.1 Function Block Creator

AlgoBuidler offers a Function Block Creator which allows defining a function block without knowledge of the XML tags. The Function Block Creator can be executed from the Tools menu or from the Toolbar by pressing the button

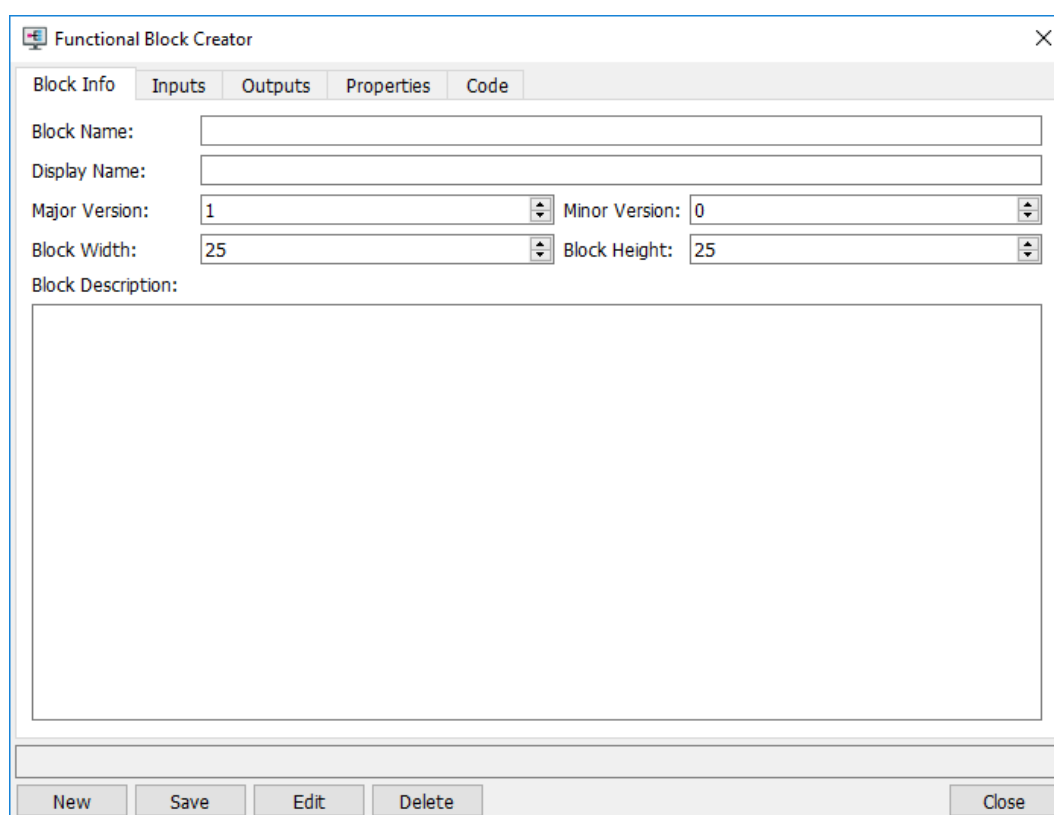


(Function Block Creator).

User-defined function blocks are stored in the libraries which are stored in the user's home directory \STMicroelectronics\AlgoBuilder\Library.

The Function Block Creator also offers the possibility to edit or delete an already existing function block.

Figure 39. Function Block Creator



The screenshot shows the 'Functional Block Creator' dialog box with the 'Block Info' tab selected. The dialog has a title bar with a close button. Below the title bar are five tabs: 'Block Info', 'Inputs', 'Outputs', 'Properties', and 'Code'. The 'Block Info' tab contains the following fields:

- Block Name:** A text input field.
- Display Name:** A text input field.
- Major Version:** A spin box with the value '1'.
- Minor Version:** A spin box with the value '0'.
- Block Width:** A spin box with the value '25'.
- Block Height:** A spin box with the value '25'.
- Block Description:** A large text area.

At the bottom of the dialog are five buttons: 'New', 'Save', 'Edit', 'Delete', and 'Close'.

Revision history

Table 5. Document revision history

Date	Version	Changes
26-Mar-2018	1	Initial release
25-Jul-2018	2	<p>Updated Figure 1. AlgoBuilder application diagram, Figure 3. AlgoBuilder design, and Figure 7. AlgoBuilder installer</p> <p>Added SensorTile STEVAL-STLKT01V1 to Section 1.1 Overview and Section 1.2 Prerequisites</p> <p>Updated Section 2.3 Application settings and Figure 10. Application Settings window</p> <p>Updated Figure 13. AlgoBuilder main window</p> <p>Updated Section 3.2 Library dock</p> <p>Updated Table 1. AlgoBuilder toolbar default functions</p> <p>Updated Table 2. AlgoBuilder installation libraries</p> <p>Updated Section 6 Creating your first design</p> <p>Updated Section 7 Programming the target</p> <p>Added Section 7.2 SensorTile</p> <p>Updated Section 8 Using Unicleo-GUI</p> <p>Added Section 9.1 Function Block Creator</p>
27-Sep-2018	3	<p>Updated Figure 7. AlgoBuilder installer</p> <p>Updated Figure 13. AlgoBuilder main window</p> <p>Added Figure 14. Using the Feedback function block</p> <p>Added Section 5 Conditional Execution</p> <p>Updated Table 2. AlgoBuilder installation libraries</p> <p>Updated Figure 18. Sensor Hub properties and Figure 22. Generated code in algo_builder.c file</p> <p>Added Section 10 FFT (Fast Fourier Transform)</p> <p>Minor textual updates</p>

Contents

1	Description	2
1.1	Overview	2
1.2	Prerequisites	2
1.3	Terms and references	3
1.4	Principle of operation	5
2	Getting started	6
2.1	Installing the software	6
2.2	Running the software for the first time	6
2.3	Application settings	7
2.4	Network update settings	8
3	Using AlgoBuilder	9
3.1	Workspace	10
3.2	Library dock	10
3.3	Description dock	10
3.4	Properties dock	10
3.5	Toolbar	11
4	Data types	13
5	Conditional Execution	14
6	Libraries	15
7	Creating your first design	16
8	Programming the target	19
8.1	STM32 Nucleo board	19
8.2	SensorTile	19
9	Using Unicleo-GUI	20
9.1	Data visualization	20
9.2	Data input	27
10	FFT (Fast Fourier Transform)	28
11	Creating your own function block	30
11.1	Function Block Creator	32

Revision history	33
------------------------	----

List of tables

Table 1.	AlgoBuilder toolbar default functions.	11
Table 2.	AlgoBuilder installation libraries	15
Table 3.	AlgoBuilder supported libraries	15
Table 4.	Function block XML tags.	30
Table 5.	Document revision history	33

List of figures

Figure 1.	AlgoBuilder application diagram	1
Figure 2.	AlgoBuilder function block	3
Figure 3.	AlgoBuilder design	3
Figure 4.	STM32 Nucleo (NUCLEO-F401RE) plus X-NUCLEO-IKS01A2	3
Figure 5.	Unicleo-GUI	4
Figure 6.	Algobuilder principle of operation	5
Figure 7.	AlgoBuilder installer	6
Figure 8.	AlgoBuilder icon	6
Figure 9.	Application Settings menu option	7
Figure 10.	Application Settings window	7
Figure 11.	Network Update Settings menu option	8
Figure 12.	Network Update Settings window	8
Figure 13.	AlgoBuilder main window	9
Figure 14.	Using the Feedback function block	13
Figure 15.	Conditional Execution	14
Figure 16.	AlgoBuilder Firmware Settings window	16
Figure 17.	Sensor Hub function block	16
Figure 18.	Sensor Hub properties	16
Figure 19.	Sensor Hub, Acceleration[g], Graph function blocks	16
Figure 20.	Graph properties	17
Figure 21.	Sensor Hub, Acceleration[g], Graph function blocks	17
Figure 22.	Generated code in algo_builder.c file	17
Figure 23.	System Workbench for STM32 output	18
Figure 24.	Program STM32 Nucleo selection box	19
Figure 25.	STM32 Nucleo board, SensorTile cradle SWD connectors	19
Figure 26.	Bar function block and example of data visualization in Unicleo-GUI	20
Figure 27.	Fusion function block and example of data visualization in Unicleo-GUI	21
Figure 28.	Graph function block and example of data visualization in Unicleo-GUI	22
Figure 29.	Histogram function block and example of data visualization in Unicleo-GUI	22
Figure 30.	Logic Analyzer function block and example of data visualization in Unicleo-GUI	23
Figure 31.	3D Plot function block and example of data visualization in Unicleo-GUI	24
Figure 32.	Scatter Plot function block and example of data visualization in Unicleo-GUI	25
Figure 33.	Value function block and example of data visualization in Unicleo-GUI	26
Figure 34.	Input Value function blocks	27
Figure 35.	Input Values in Unicleo-GUI	27
Figure 36.	FFT function block	28
Figure 37.	FFT and FFT Plot connections	28
Figure 38.	Frequency spectrum in Unicleo-GUI	29
Figure 39.	Function Block Creator	32

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved