BitVoicer 1.2

User Manual

English

Copyright © 2013 BitSophia Tecnologia Ltda - ME. All rights reserved.

Table of Contents

1 Introduction

BitVoicer is the speech recognition software that brings to the microcontrollers' universe all the processing and voice analyzing power of the PC. The BitVoicer's speech recognition technology and the use of communication standards common in the industry (TCP/IP and Serial/UART) make it extremely easy and fast to design complex speech recognition functionalities for virtually any microcontroller available in the market.

This manual will guide you through the process of installing and configuring BitVoicer, show you the incredible flexibility of the BitSophia's Voice Schemas and introduce you all the tools available in the application.

1.1 Minimum Requirements

- Microsoft Windows Vista, 7, 8 or 10 (except Windows 10 IoT Core and Windows 10 Mobile)
- Microsoft .NET framework 4.0 or greater
- One of the following communication interfaces:
  - Serial Port
  - USB (used as a virtual COM port)
  - Ethernet Adapter
  - Wireless Adapter
- Internet connection (for activation and installation of additional languages)

1.2 Compatibility

BitVoicer should be compatible with any programmable microcontroller or shield which uses TCP/IP communication based on sockets or serial communication (UART interface). However, you should refer to the documentation of your device and make sure it supports these communication standards.

During the development of BitVoicer some products of the following manufacturers were used and are more likely to be compatible:

- Arduino
- Roving Networks
- Digi International
- Atmel Corporation
- Sparkfun Electronics

1.3 Safety Warning

The BitVoicer must not be used in any critical mission applications, life support devices, the conduction or operation of machinery or automotive vehicles, or, any application whose malfunction can cause damage to property, the environment or endanger the health of persons or animals. The BitSophia does not authorize the use of its software in any of these applications. The user assumes all liability for any misuse or violation of this prohibition.

2 First steps

2.1 Installation

To perform the following installation procedures, it is necessary to log on to Windows using an account with administrator privileges:

1. Download the installation file **BitVoicerSetup_v12_xXX.exe** from BitSophia (www.bitsophia.com).
2. Execute the installation file and follow the instructions on the screen. You must agree to the BitVoicer License Agreement and choose an installation folder.

3. Launch the application by using the shortcut created on your desktop or through the shortcut located at Start → All Programs → BitSophia → BitVoicer. The activation of BitVoicer will not be requested until you try to press the speech recognition engine start button.

2.2 Graphic Interface

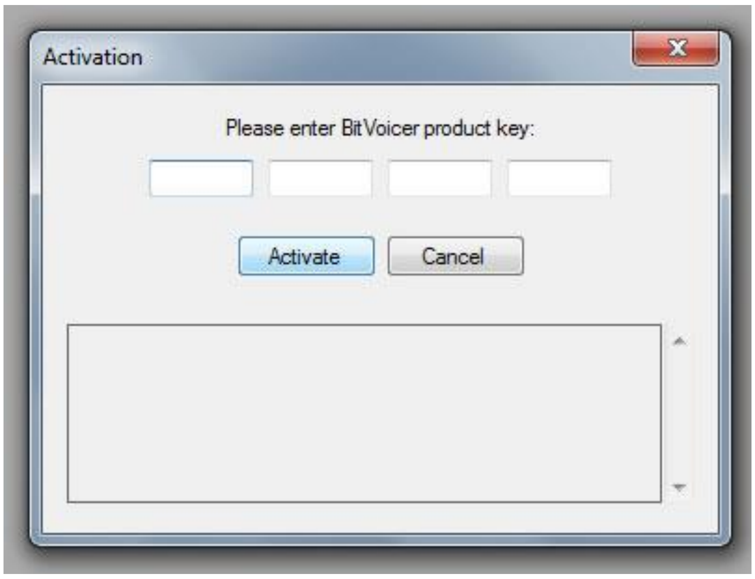
The BitVoicer graphic user interface (GUI) is basically composed of two areas: menus and workspace. The menus, which are described below and found on the top of the screen, access features always displayed in the workspace:

- **File → New:** starts the design of a new Voice Schema and loads the editor in the workspace area.
- **File → Open:** enables a previously designed Voice Schema to load. By default, the initial folder shown in the dialog box is the same as in **File → Preferences → Default output folder**.
- **File → Close:** closes the Voice Schema currently opened. In case the current Voice Schema had been altered, a dialog box will ask the user to save the current Voice Schema.
- **NOTE:** the use of any of the options above stops the speech recognition engine.
- **File → Save:** saves the current Voice Schema. In case the current Voice Schema had not been previously saved, a dialog box will prompt the user for the name and location of the file.
- **File → Save As:** saves the current Voice Schema in a file and location specified by the user.
- **File → Preferences:** opens the form where all the default settings of the BitVoicer must be entered.
- **File → Exit:** closes the application and stops the recognition engine. If the opened Voice Schema has been altered, but it has not been saved, BitVoicer will prompt the user to save the Voice Schema before closing the application.
- **Help → BitVoicer Manual:** opens the reference manual of the BitVoicer.
- **Help → Activation:** opens the form where the user can start the activation process of their copy of BitVoicer. In case the activation has already been done, this form will remain unavailable.
- **Help → About BitVoicer:** displays all the information about the copy of BitVoicer plus contact details.

2.3 Activation

To be able to use the communication interfaces, you must have your copy of BitVoicer activated by a valid product key. In case you do not have this key, get yours at www.bitsophia.com.

With the product key in hand, run the BitVoicer as an administrator (click on the BitVoicer shortcut using the right button and select "Run as administrator") and click on *Activation* in the *Help* menu. You should see the *Activation* form below:



After you enter a valid product key, click on *Activate*. At this point, your computer must be connected to the Internet, so that the BitSophia Activation Service can check the informed product key.

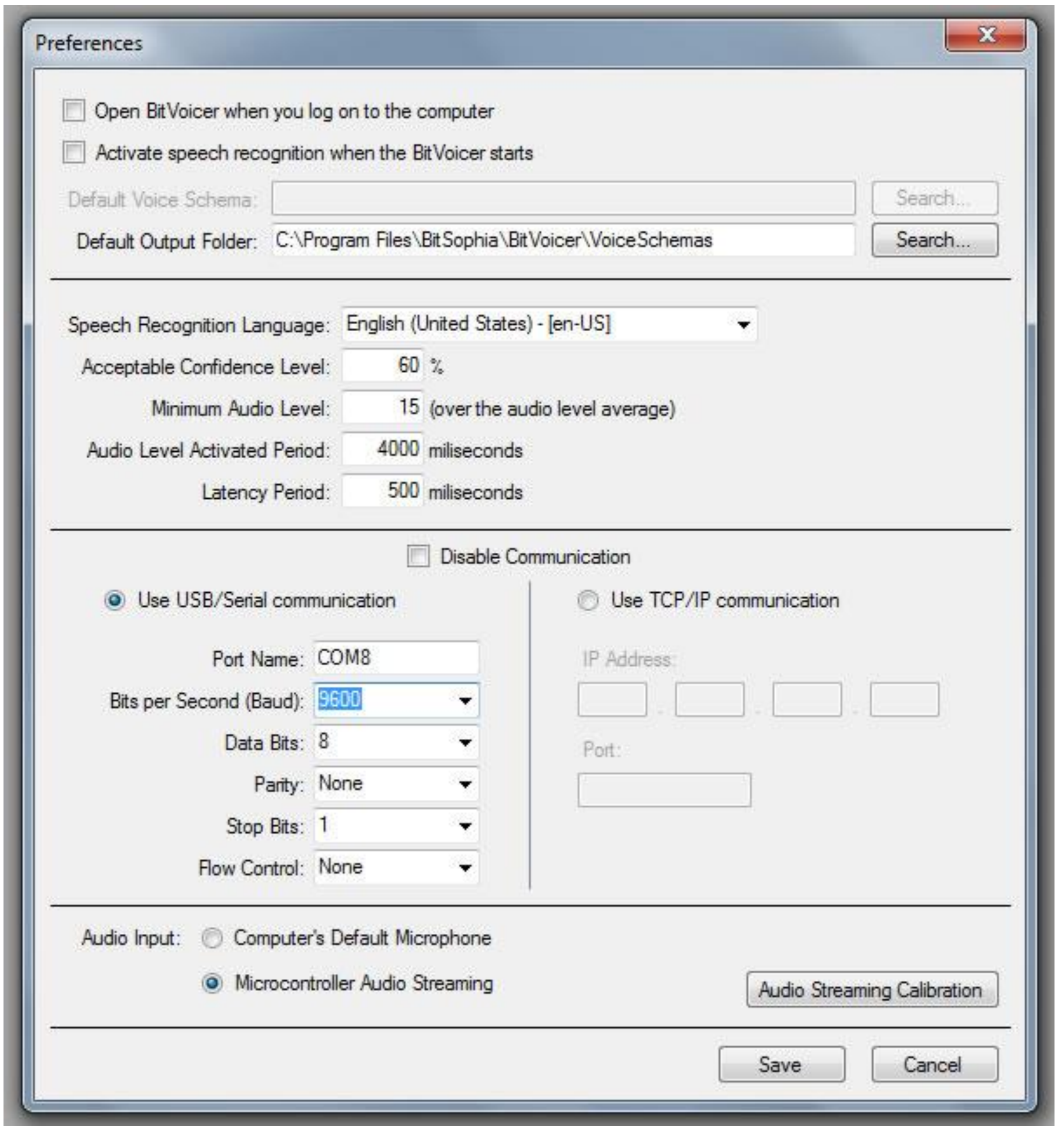
In the text box below the *Activate* button, it is possible to follow the activation process and visualize any error messages regarding this process. Once the BitVoicer activation succeeds, a message box will inform you that the activation process has been completed and the speech recognition engine can now be started.

During the activation process, BitVoicer will generate an exclusive code for your computer. This code will be encrypted and stored on BitSophia's servers. None of your computer's information will be used to contact you in the future, and this information will not be shared with others.

Once a product key is activated, you will not be able to use it to activate any other copy of BitVoicer on other computer. For this reason, activate your copy of BitVoicer only on the machine on which it is intended to be used.

3 Configuration

To access the settings form of the BitVoicer, click on *Preferences* in the *File* menu. You should see the following *Preferences* form:



[ ] Open BitVoicer when you log on to the computer
[ ] Activate speech recognition when the BitVoicer starts
Default Voice Schema:
Default Output Folder: C:\Program Files\BitSophia\BitVoicer\VoiceSchemas
Speech Recognition Language: English (United States) - [en-US]
Acceptable Confidence Level: 60 %
Minimum Audio Level: 15 (over the audio level average)
Audio Level Activated Period: 4000 miliseconds
Latency Period: 500 miliseconds
[ ] Disable Communication
● Use USB/Serial communication
○ Use TCP/IP communication
Port Name: COM8
Bits per Second (Baud): 9600
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: None
IP Address:
Port:
Audio Input: ○ Computer's Default Microphone
● Microcontroller Audio Streaming

The BitVoicer already comes with a default set of configurations and the speech recognition engine can be started at any time without the need of any initial setup.

3.1 Initialization

The following initialization options can be changed in the *Preferences* form:

- **Open BitVoicer when you log on to the computer:** when this option is checked, the BitVoicer will be loaded with Windows and will appear minimized on the task bar of your computer.
- **Activate speech recognition when the BitVoicer starts:** when this option is checked, the BitVoicer will start its speech recognition engine right after it is opened. With this option enabled, you must indicate the default Voice Schema to be loaded. To select the default schema, you can use the file search tool (*Search* button) located to the right of the *Default Voice Schema* textbox.
- **Default output folder:** this is the default folder used to save the Voice Schemas designed with BitVoicer. To select the default folder, you can use the folder search tool (*Search* button) located to the right of the *Default output folder* textbox.

3.2 Speech Recognition

This section of the preferences is responsible for the proper functioning of the speech recognition engine. Although the BitVoicer comes with a default set of configurations, you need to read carefully the items 3.2.1 to 3.2.6 in case you want to enhance its performance.

3.2.1 Speech Recognition Language

The *Speech Recognition Language* dropdown list displays the recognition languages installed on your computer. The following languages are currently available:

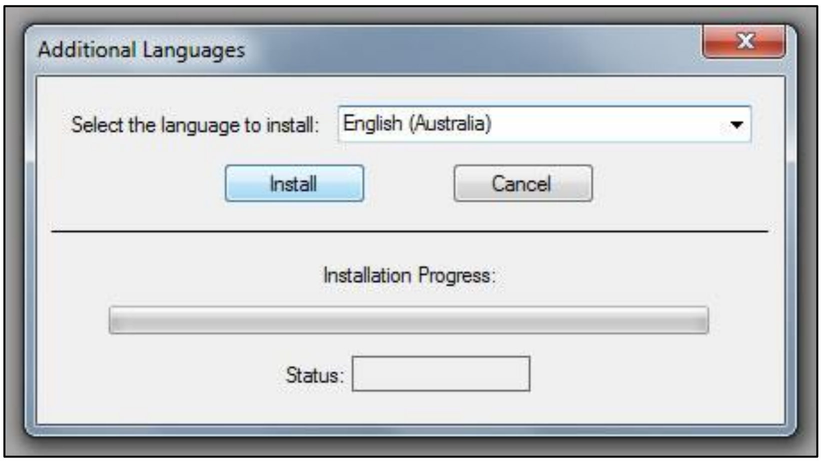
- Catalan
- Chinese (China, Honk Kong and Taiwan)
- Danish (Denmark)
- Dutch (Netherlands)
- English (Australia, Canada, India, United Kingdom and United States)
- Finnish (Finland)
- French (Canada and France)
- German (Germany)
- Italian (Italy)
- Japanese (Japan)
- Korean (Korea)
- Norwegian, Bokmål (Norway)
- Polish (Poland)
- Portuguese (Brazil and Portugal)
- Russian (Russia)
- Spanish (Mexico and Spain)
- Swedish (Sweden)

By default, BitVoicer is installed with English (US) only. If you need to install any additional language, refer to the section “Additional Languages” (item 3.2.1.1).

3.2.1.1 Additional Languages

To be able to install additional recognition languages, you must start BitVoicer with administrative privileges (“*Run as Administrator*”) and you must be connected to the internet.

First, click on *Additional Languages* in the *File* menu. You should see the form below:



The language dropdown list only shows the languages that are not installed on your computer. Select one language in the list and click on the *Install* button. After the download and installation of the language package, you will be able to select the recently installed language in the preferences form (refer to section 3).

3.2.2 Acceptable Confidence Level

The Acceptable Confidence Level represents the relative minimum probability for the speech recognition engine to accept a detected speech. The confidence level does not indicate the absolute probability that an alternative corresponds to a given speech. The confidence level indicates that the alternative is more likely to be the correct match among the multiple alternatives available in the Voice Schema. Suppose you have a Voice Schema with ten recognition alternatives: during the recognition process, each alternative will receive a “grade” (1 to 100%) which represents the probability of this alternative be the correct match among all available alternatives. The alternative with the highest “grade” will be the one accepted by the speech recognition engine. When you enter an acceptable confidence level, you set a minimum limit for this “grade”. All other recognitions with a lower “grade” than the Acceptable Confidence Level will be rejected, and therefore, no command will be sent to the microcontroller.

The Acceptable Confidence Level is a percentage value from one to one hundred and the use of decimals is significant for determining if a speech will be rejected or accepted.

When you set the confidence level, you must consider that different interlocutors will get different confidence levels for the same speech. Therefore, it is necessary to set the acceptable confidence level to meet the “grades” achieved by all the users of the application. However, excessively lowering the confidence level can lead BitVoicer to accept a recognition that should actually be rejected.

The best way to determine a base for the acceptable confidence level is by making tests of recognition. To do that, disable the BitVoicer communication (item 3.3), load a Voice Schema and start the recognition engine (item 4.5). Watch the confidence levels in the activity panel (item 7.1). In this panel it is possible to see the “grade” attributed to each recognition and obtain a base for the acceptable confidence level.

It is important to point out that changes in the environment also affect the levels of confidence achieved by the user. When there are significant changes in the environment (such as increase in background noise or changes in the average audio level), it may be necessary to review the settings of the acceptable confidence level.

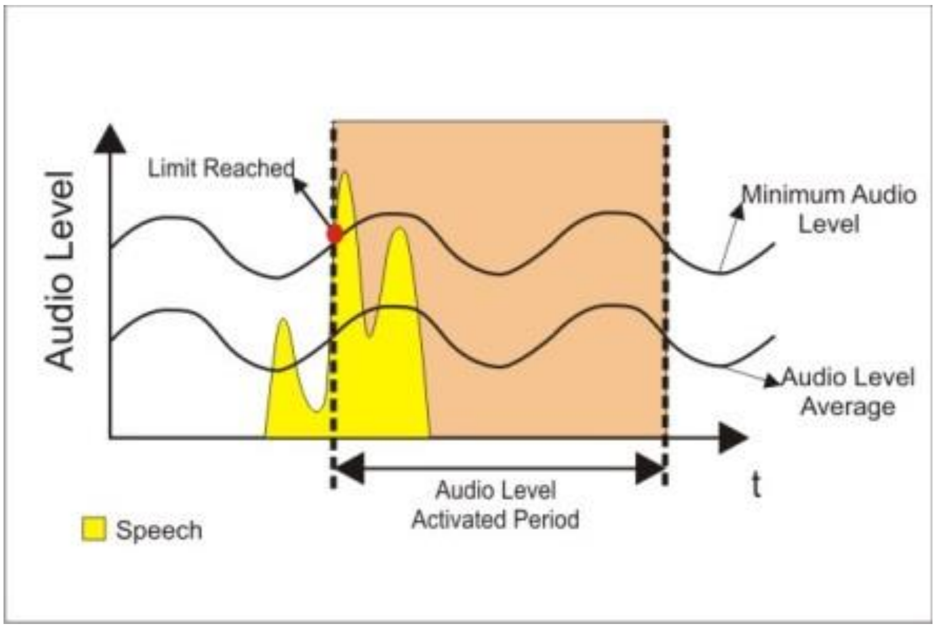
For further information about the Acceptable Confidence Level, please refer to the section “Speech Recognition Performance” (item 3.2.6).

3.2.3 Minimum Audio Level

The Minimum Audio Level works similar to the Acceptable Confidence Level. It represents a lowest limit that must be reached, before the BitVoicer accepts a particular speech.

As soon as the speech recognition engine is started, it begins recording the audio level captured from the environment. Based on the setting of the following item (*Audio level activated period*), an average audio level is calculated for the period. In case the audio level, during the recognition, does not reach this average plus the minimum audio level, the recognition will be rejected. In other words, a speech only will be recognized and accepted if the audio level reaches or overcomes the value represented by the audio level average plus the minimum audio level.

The following picture illustrates the concept behind this setting:



3.2.4 Audio Level Activated Period

The Audio Level Activated Period starts when the minimum audio level is reached. This period renews itself whenever the minimum audio level is reached, and all recognized speech in this period is accepted by the BitVoicer (regardless of the audio level at the moment of recognition) unless:

- The confidence level is lower than the acceptable confidence level;
- The moment of recognition is inside the latency period (item 3.2.5);
- The activation word (item 4.4) has not been identified, in case this feature had been enabled;
- The moment of recognition is outside the activation word period.

To better understand the behavior of this filter, you must know in what order all the BitVoicer's filters act and their levels of priority:

Filter	Acting	Priority
Confidence level	1st to act	5th
Minimum audio level	2nd to act	4th
Audio level activated period	3rd to act	3rd
Latency period	4th to act	2nd
Activation word	5th to act	1st

According to the above table, the higher levels of priority (1st being the highest and 5th the lowest) override a recognition accepted by filters of lower priority. For example: if all the filters accept a speech, but the fifth filter rejects it, the BitVoicer will consider the speech rejected.

Copyright © 2013 BitSophia Tecnologia Ltda - ME. All rights reserved.

The audio level activated period must be entered in milliseconds (1000 milliseconds = 1 second) and you should consider the time that is necessary to pronounce all the words of the longest sentence in the Voice Schema.

Further information about the Audio Level Activated Period is in the section “Speech Recognition Performance” (item 3.2.6).

3.2.5 Latency Period

The latency period starts right after a speech is accepted by the BitVoicer (all filters must have accepted the recognition). In this period, any recognition will be rejected even if it has passed all other filters. The latency period avoids new recognitions from being accepted right after a valid recognition. This Latency Period prevents sending commands in sequence which could interfere with the execution of a command previously sent to the microcontroller.

3.2.6 Speech Recognition Performance

This section gives you some tips on how to improve the performance of the BitVoicer’s speech recognition engine. You may have to use more than one of these tips to achieve the desired performance:

- Speak clearly and steadily. The longer the speech recognition engine keeps running, the better its accuracy will become at establishing the appropriate confidence level.
- Pronouncing the words using the correct accent is critical to improve the confidence level. If you are using a language that is not your native language, it is necessary to give special attention to the correct pronunciation of the words.
- Keep the microphone at least one meter away from any audio source such as speakers or TVs. If the interference from other audio sources persists, you may have to consider the use of a directional microphone or headset.
- Avoid using very short sentences that have words with only one or two syllables. The speech recognition engine may identify these words as parts of everyday speech and lead to “false positives”, i.e. an accepted speech that should have been rejected.
- In case you are using an *Activation Word* (item 4.4), we recommend the use of words with three or more syllables and whose occurrence is not frequent in everyday speech.
- To avoid “false positives”, i.e. an accepted speech that should have been rejected, raise the *Acceptable confidence level* (item 3.2.2) by small increments.
- In wide-open areas, with high noise levels or many other speakers, consider the use of a headset (there are wireless options available) and increase the *Minimum Audio Level* (item 3.2.3). When you use a headset, your mouth stays closer to the microphone, so that the audio level and quality improves. Increasing the audio level along with the use of a headset prevents surrounding speech from being recognized by the speech recognition engine.
- Consider gradually increasing the *Minimum Audio Level* if you are getting “false positives”, i.e. an accepted speech that should have been rejected, from other audio sources whose levels are lower.
- Consider increasing the *Latency period* to avoid “false positives”, so that the BitVoicer will ignore recognitions right after a previous recognition. Please note that if you are using an activation word, the *Latency Period* will have to be smaller, because, in most cases, you do want the BitVoicer to recognize a speech right after the activation word.
- Establishing a correct *Audio Level Activated Period* may also avoid “false positives”. If this period is much longer than the time needed to pronounce the sentences in the Voice Schema, you will increase the probability of getting accepted recognitions with insufficient audio level.
- If the audio capture is performed by a microphone wired to the microcontroller, make sure that the sample rate is not lower than 8000 or higher than 8200 samples per second. The speech recognition performance can be unpredictably compromised if the sample rate is out of this range. Also make sure that the audio stream sent to BitVoicer is free of interference or noise

caused by the microcontroller's CPU, communication errors, voltage variations or port fluctuations.

3.3 Communication

In this section of the preferences, you set the communication interface used by the BitVoicer to send data to your microcontroller:

- **Using USB/Serial Communication:** if using this interface, it is necessary to enter which serial port your microcontroller is connected to (e.g. COM4, COM8, etc.) and the parameters of serial communication. It is possible that higher communication speeds (baud rate) cause data corruption on some computers or microcontrollers. If you believe you are receiving corrupted data in your microcontroller, try reducing the communication speed.
- **Using TCP/IP Communication:** if using this interface, it is necessary to enter the IP and port to where the commands will be sent. The destination microcontroller must support communication based on sockets (Berkeley Sockets).

As soon as the speech recognition engine is started, BitVoicer will block the port in use for both communication interfaces. No other application on the computer will have access to this port. However, it is possible to send data from your microcontroller to the PC and see this data in the communication monitor panel (item 7.2).

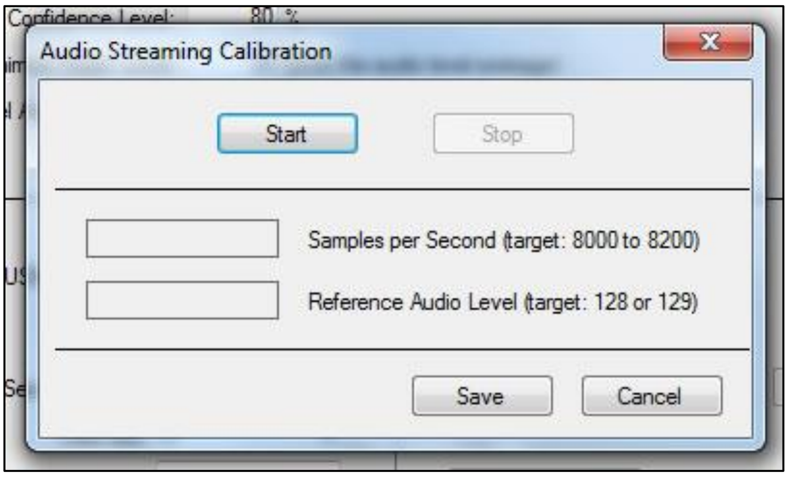
Also, you can disable the transfer of data to the microcontroller by checking *Disable communications*. Even if you check this option, it is possible to start the speech recognition engine and monitor the BitVoicer's activity in the activity panel (item 7.1).

3.4 Audio Input

BitVoicer can process audio captured by the microphone installed on the computer or by a microphone wired to the microcontroller. The *Audio Input* option allows you to select the desired audio source. For further information about audio capture through the microcontroller, refer to the section "Audio Streaming" (item 6).

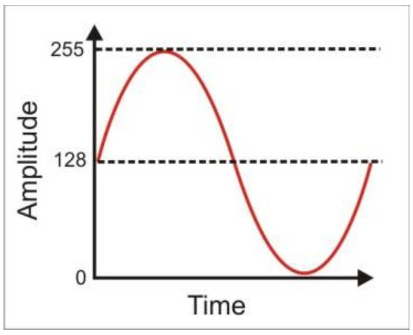
3.4.1 Audio Streaming Calibration

If the audio capture is made by the microcontroller and sent to BitVoicer as audio stream, you must run the Audio Streaming Calibration Tool, so that BitVoicer can correctly establish the audio level it receives. The information panel that shows the audio level (item 7.1) and the audio level filter (item 3.2.3) directly depends on the values obtained by this tool.


Audio Streaming Calibration
Start
Stop
Samples per Second (target: 8000 to 8200)
Reference Audio Level (target: 128 or 129)
Save
Cancel

The *Samples per Second* field displays the number of samples received by BitVoicer in each second. This value can be used as reference to adjust the number of audio captures performed by the micro-controller.

The *Reference Audio Level* field displays the average value of the received samples. In a sound wave, this value represents the center of the amplitude of a wave as shown below. Because BitVoicer processes 8-bit samples, the target values for this field are 128 or 129. For further information about audio streaming, refer to the section 6 of this manual.



When you click on the *Start* button, BitVoicer sends a “Start Sampling” command to the micro-controller. The *Stop* button sends a “Stop Sampling” command when it is clicked. Refer to the section Communication (item 5) for further information about BitVoicer’s commands.

4 Creating, Editing and Using Voice Schemas

4.1 The Voice Schema

The Voice Schema is the structure upon which the BitVoicer works to bind a specific command to a sentence recognized by the speech recognition engine. To do that, the BitVoicer uses two main features of the Voice Schema, which are the creation of all possible permutations (called anagrams) for a given sentence and the mapping of each anagram to the corresponding command.

The anagrams are generated by calculating all possible permutations from the sentence items (item 4.3). Suppose you have the sentence model below:

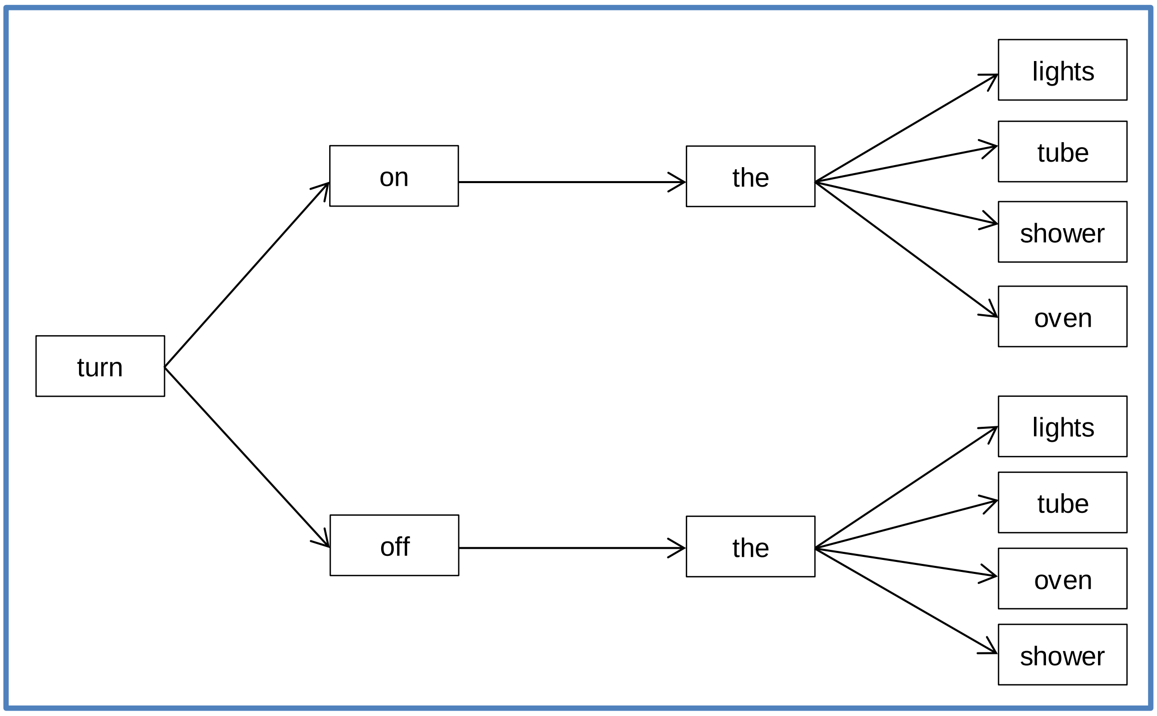
Simgle Item	Option Item	Single Item	Option Item
turn	on	the	lights
	off		tube
			oven
			shower

The BitVoicer would create the following anagrams in the Voice Schema:

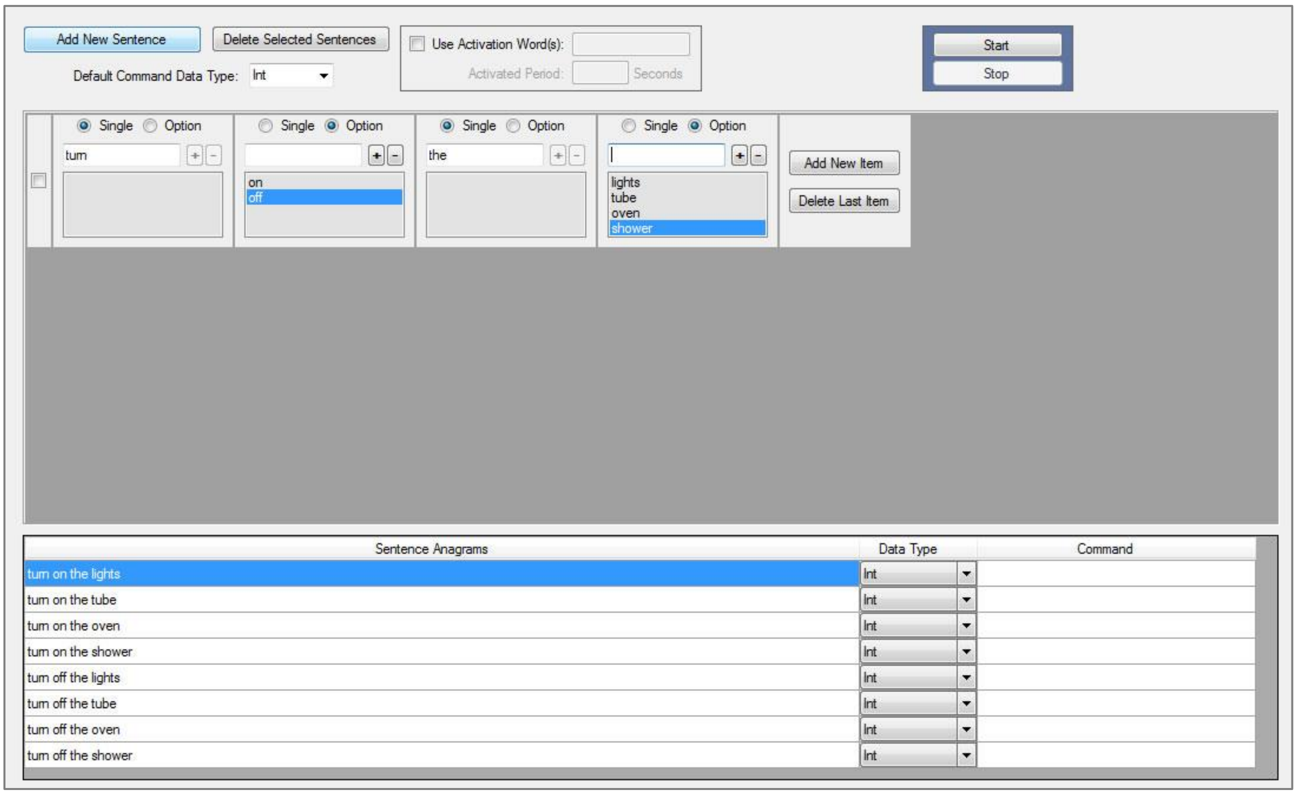
Anagrams			
turn	on	the	lights
turn	on	the	tube
turn	on	the	oven
turn	on	the	shower
turn	off	the	lights
turn	off	the	tube
turn	off	the	oven

turn	off	the	shower

For a better comprehension, this is the tree of possibilities generated from the above Voice Schema:



In the Voice Schema Editor, this would be the representation of the Voice Schema:



Default Command Data Type: Int

Column	Mode	Entry	Items
1	Single	turn	
2	Option		on, off
3	Single	the	
4	Option		lights, tube, oven, shower

Sentence Anagrams	Data Type	Command
turn on the lights	Int	
turn on the tube	Int	
turn on the oven	Int	
turn on the shower	Int	
turn off the lights	Int	
turn off the tube	Int	
turn off the oven	Int	
turn off the shower	Int	

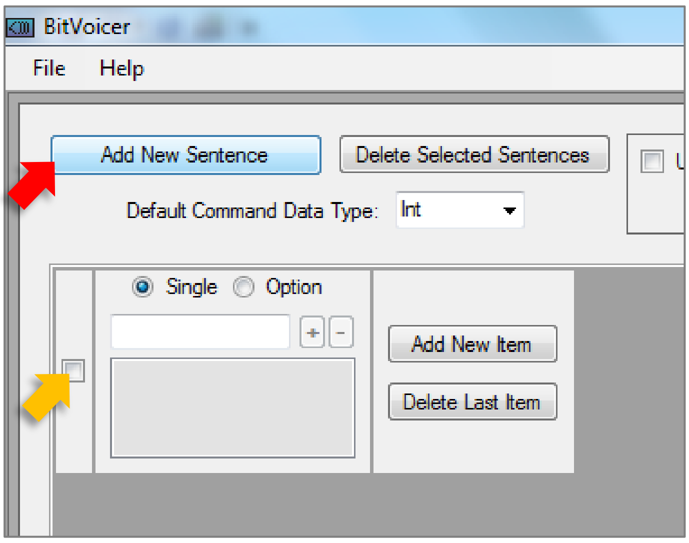
After generating all possible anagrams, the BitVoicer allows the mapping of an individual command to each anagram. This information is visible on the lower portion of the screen.

Another important attribute of the BitSophia's Voice Schema is its flexibility. At any time you can add, edit or delete sentence items or whole sentences. The BitVoicer will immediately recalculate the entire possibilities tree and create the new corresponding anagrams. The capacity of including sentences or sentence items has not been limited in the application and the size of a Voice Schema virtually has no limits. The only restrictions for the Voice Schema are the available memory and the processing power of your computer.

When you start the speech recognition, the BitVoicer loads all sentences and analyses the audio input looking for the best match (anagram) among all available sentences. For this reason, you can say that these sentences make up the "vocabulary" of the speech recognition engine. In case a user pronounces a word not present in this "vocabulary", at any position in the sentence, the speech recognition will make an approximation with other words at the same position. If this process does not find a relevant match, which is the result in most cases, the speech recognition engine will reject the speech.

4.2 Adding Sentences

The first step to start building a Voice Schema is to add a sentence. To do so, click on the button *Add New Sentence* (see red arrow in the picture below) located at the upper left corner of the Voice Schema Editor.



When you click this button, a new sentence is added to the Voice Schema. By default, this sentence is created with a blank *Single* item. Also, two buttons (add and delete) are added on the right side of the sentence item (refer to item 4.3).

The *Delete Selected Sentences* button, which is located to the right of the *Add New Sentence* button, removes the sentences marked in the check box located at the beginning of each sentence (see orange arrow in the picture above). Removing any sentence from the Voice Schema also removes all commands associated to the sentence. In case your commands have data you do not want to lose, remember to copy this data before removing a sentence.

Please note that when you add a sentence, an anagram is created in the column *Sentence Anagrams* on the command board. This anagram has the *{EMPTY}* description because there is no data at this sentence position that allows the creation of a complete anagram. Different from a *Single* item, a sentence item of the *Option* type (refer to item 4.3 right below) will show the description *[{EMPTY}]* when it has no data.

A Voice Schema can contain several sentences and the anagrams based on each of these sentences will be ordered in the command board. Suppose you have a Voice Schema with two sentences and each sentence has four possible anagrams. The command board will display the four anagrams from the first sentence and then the four anagrams from the second one, that is, the anagrams do not shuffle. The last anagram from a sentence marks the beginning of the anagrams from the following one.

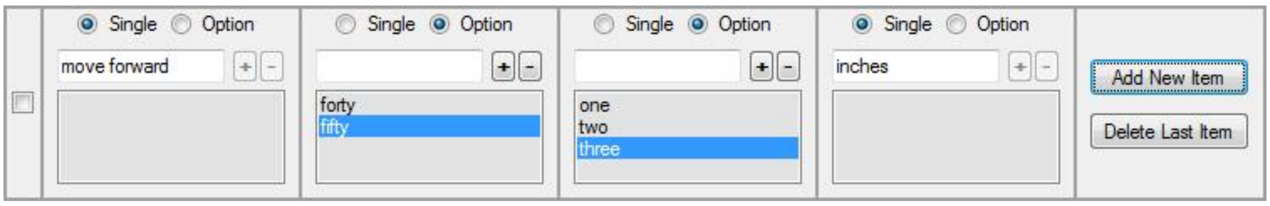
An important aspect of the BitVoicer's speech recognition engine is that it always recognizes a sentence as a whole and not just part of it. Suppose you have the following sentence in your Voice Schema: "go forward ten inches". The speech recognition engine expects to hear the whole sentence and not just part of it as in: "go forward". In this situation, the speech recognition engine should reject the speech. If you want to have both sentences accepted ("go forward" and "go forward ten inches"), it will be necessary to create two sentences, one for each statement.

4.3 Sentence Items

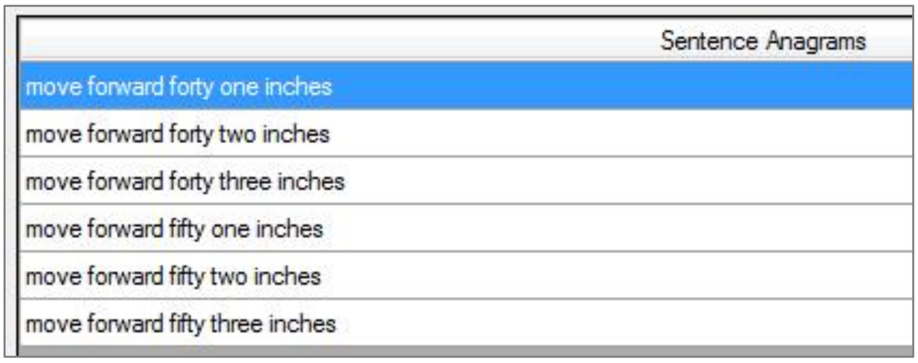
Each sentence in a Voice Schema consists of a sequence of sentence items. These items can be of two types:

- **Single**: the *Single* items represent a position in a sentence whose content is common to all possible permutations. A *Single* item can contain one word or a set of words considered as a unit.
- **Option**: the *Option* items consist of a set of options that may vary at that specific position. If an *Option* item contains only one option, it will have the same behavior as a *Single* item. Similarly to a *Single* item, each option in an *Option* item can contain one word or a set of words considered as a unit.

The sentence below uses some *Single* and *Option* items:



This set of anagrams is based on the sentence above:



Sentence Anagrams
move forward forty one inches
move forward forty two inches
move forward forty three inches
move forward fifty one inches
move forward fifty two inches
move forward fifty three inches

Note that the *Single* items (first and forth) repeat themselves in all anagrams. Even for the first *Single* item, which is composed of two words, there is no creation of permutations for each word.

The second and third items are *Option* items and, in this case, the anagrams undergo permutations at the exact position of each *Option* item. In these *Option* items, it is possible to note that numbers must not be used in their numeric form (e.g. 1, 2, 3, etc.), but rather in their literal form (e.g. one, two, three, etc.).

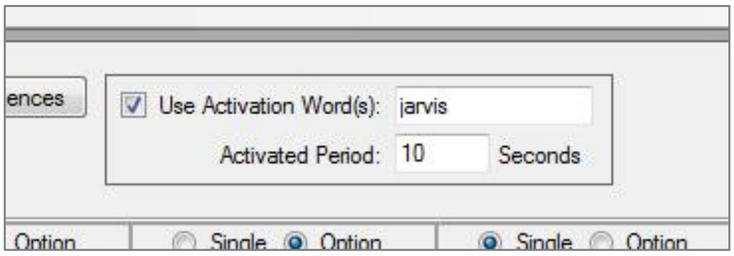
To set a sentence item as *Single* or *Option*, check the corresponding radial button on the top of the item. In items of the *Single* type, the "+" (add) and "–" (remove) buttons and the list box of options are disabled. When you set an item as *Option*, these controls become available.

To add a word to an *Option* item, type this word in the item textbox and then click on the "+" (add) button. You can also type the word and just press ENTER that the BitVoicer will automatically add it to the list box of options.

To remove an option from an *Option* item, select the option in the list box and then click on the "–" (remove) button. Removing any options from an *Option* item also removes all commands associated to the option. In case your commands have data you do not want to lose, remember to copy this data before removing an option.

4.4 Activation word

The Activation Word is one word, or a set of words, that must be said before the sentence itself. You can say the Activation Word and the sentence in a continuous manner or you can say only the activation word and, in the activation period, the sentence itself. When you check *Use Activation Word(s)* on the editor, BitVoicer will accept a speech only if the activation word is pronounced before the sentence.



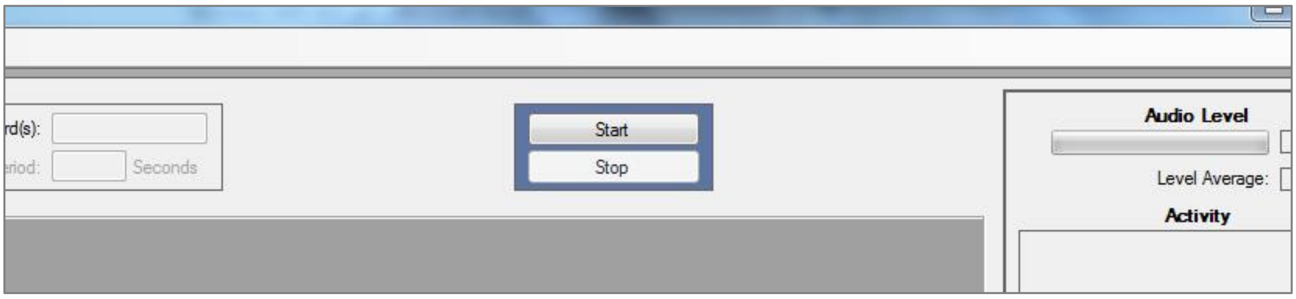
The activated period by Activation Word represents the period during which you do not need to repeat this word, so that BitVoicer considers the recognition as valid. This period must be entered in seconds.

The Activation Word can be used to personalize a Voice Schema or as a strategy to avoid "false positives". In this last case, we recommend the use of words with three or more syllables and whose occurrence is not frequent in everyday speech.

It is worth remembering that the activation word is one of the filters used by BitVoicer to accept or reject the recognition (refer to item 3.2.4).

4.5 Starting and Stopping Speech Recognition

To start the speech recognition, you just have to click on the *Start* button located at the center upper portion of the Voice Schema Editor.



You must have an opened Voice Schema to be able to start the speech recognition and this schema cannot have any anagram with the *{EMPTY}* or *[{EMPTY}]* description in it. In other words, it cannot have any blank sentence items. It is not necessary to have the commands filled out on the command board, but when the speech recognition engine identifies an anagram corresponding to this command, the activity panel will show an error message regarding the missing command (refer to item 6.1).

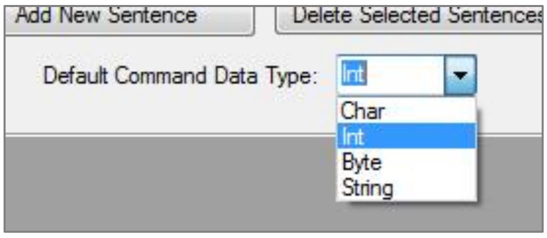
It is not possible to use any communication interface if you do not have an activated copy of BitVoicer. To activate your copy, refer to the item 2.3 of this manual.

BitVoicer will block the communication port specified in the preferences and this port will remain blocked until the speech recognition engine stops. When the speech recognition engine starts, you will not be able to edit the Voice Schema and the information panels (refer to item 7) will display information about the state of the application and the communication. To regain access to the Voice Schema, you must stop the speech recognition engine.

When you click on the *Start* or *Stop* button, BitVoicer sends a status update to the microcontroller. This status can be "Engine Running" or "Engine Stopped". For further information about the data exchange between BitVoicer and the microcontroller, refer to the Communication section (item 5).

4.6 Default Command Data Type

Every time a new anagram is created in the Voice Schema, it is added to the command board. The command data type of this anagram is defined by the value in the *Default Command Data Type* list box located right below the *Add New Sentence* button on the Voice Schema Editor.



This feature is especially useful when you are creating a Voice Schema with many anagrams, and it will save you a lot of work when the commands are being defined.

4.7 Practical Example

In the installation package of BitVoicer there is an example of a complete Voice Schema (Sample.vsc). It can be found in the *VoiceSchemas* folder of the BitVoicer installation folder. This schema could be used to control an automated hospital bed. It would be very useful in a medical center where the patients are in a state of reduced mobility.

5 Communication

BitVoicer can communicate with the microcontroller over serial ports or TCP/IP ports (refer to item 3.3). Whatever communication interface you choose, most of the data exchanged between BitVoicer and the microcontroller is wrapped in a specific protocol which is described on item 5.1. This protocol is not used just when the BitVoicer is receiving audio stream from the microcontroller (item 6).

The communication between BitVoicer and the microcontroller occurs in five situations:

- While using the Audio Streaming Calibration tool (item 3.4.2), when you click on the *Start* or *Stop* button;
- When you start or stop the speech recognition engine;
- When a speech is recognized and accepted;
- When information is sent from the microcontroller to be displayed on the Communication Monitor panel;
- During the period in which the BitVoicer is receiving audio stream from the microcontroller.

5.1 BitVoicer Protocol

The BitVoicer Protocol consists of a set of formatting parameters that must be implemented in order to exchange information between BitVoicer and the microcontrollers. This protocol must be used in both communication directions: BitVoicer → microcontroller and microcontroller → BitVoicer.

The table below describes the format of the datagrams sent and received by BitVoicer:

Name	Byte Position	Value	Description
Start of Transmission	1	1	Start of Transmission mark
Data Type	2	1 = Char 2 = Int 3 = Byte 4 = String 254 = Command 255 = Status	Indicates the data type carried by the datagram
Data Length	3	X	Length of the data field in bytes
Data	4 a n	X	Contains the data itself n = 3 + Data Length
End of Transmission	n + 1	4	End of Transmission mark

Although commands are wrapped according to these parameters, in essence, all exchanged data is a sequence of bytes whose length will vary depending on the data type and the amount of data exchanged.

The first byte in a datagram always must have the value 1 (0x01). This byte marks the beginning of a datagram. The second byte, whose value can be any of those in the Value column of the table above, specifies the data type contained within the Data field. The third byte indicates the length, in bytes, of the Data field. Because the Data Length field is only one byte long, the longest length for the Data field is 255 (the biggest number in only one byte). The fourth field contains the data itself, and its length is the one specified in the third byte. The fifth field must have the value 4 (0x04), and it marks the end of the datagram. This field can be used to check if a complete datagram was received.

The data contained within the Data field is formatted according to the following description:

- **Char data:** corresponds to only one alpha-numeric character in the ASCII table (http://en.wikipedia.org/wiki/ASCII). The length of this data type will always be one byte, and its value will be the decimal value of the character in the ASCII table;
- **Int data:** contains a 16-bit integer between -32768 and 32767. The length of this data type will always be two bytes (little-endian: http://en.wikipedia.org/wiki/Endianness) that constitute the integer value;
- **Byte data:** corresponds to an 8-bit unsigned integer between 0 and 255. When you enter a byte value in the BitVoicer's Command Panel (located write below the sentence constructor), you must follow the format: b00000000, where 0 can be 1 or 0 (command examples can be found in the item 5.3);
- **String data:** corresponds to a sequence of characters whose maximum length varies according to the direction of the communication. Because of the input buffer size limitation on some microcontrollers, the longest sequence of characters that can be sent to the microcontroller is fifty-nine characters long. In the other direction, when a string is sent to BitVoicer, the longest sequence of characters is limited to the maximum length of the Data field in the BitVoicer Protocol (255 characters). The value of each character will be the decimal value of the character in the ASCII table (http://en.wikipedia.org/wiki/ASCII);
- **Command data:** this data type must not be confused with the commands sent to the microcontroller in response to a valid speech recognition whose value is defined by the user (item 5.3). Currently, BitVoicer sends only two values of the command data type to the microcontroller: *Start Sampling* or *Stop Sampling*. This command is sent only by the Audio

Streaming Calibration tool (item 3.4.2) and it will have the value 1 (0x01) for start sampling or the value 0 (0x00) for stop sampling.

- **Status data:** every time the speech recognition engine is started or stopped, one datagram with this data type is sent to the microcontroller. When you start the speech recognition by clicking on the *Start* button (item 4.1), the value 1 (0x01 – *Engine Running*) is wrapped in a datagram and sent to the microcontroller. When you click on the *Stop* button, the value 0 (0x00 – *Engine Stopped*) is sent in the same manner.

5.2 Sending Data to BitVoicer

BitVoicer constantly monitors all data received at the port used to establish communication with the microcontroller. All data received at this port must be formatted according to the BitVoicer Protocol and its content is displayed in the Communication Monitor panel (item 7.2). The data types that can be sent to BitVoicer are: Char (1 character), Int (16-bits), Byte (unsigned 8-bit integer) or String (sequence of characters).

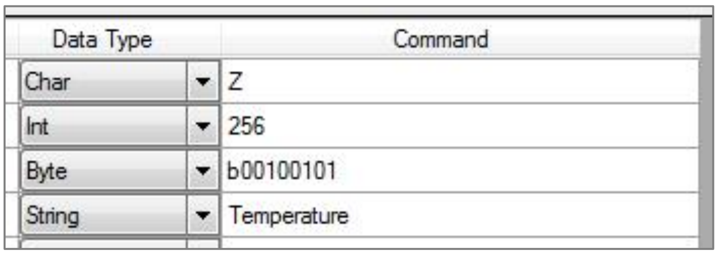
If the audio capture is made by a microphone wired to the microcontroller, any data sent to BitVoicer when it is processing audio stream is considered to be part of the audio. For this reason, in case you need to send data to be displayed in the Communication Monitor, you need to send a signal to BitVoicer, so that it stops processing the audio stream and starts handling data as datagrams. Additional information about signaling to BitVoicer can be found in item 6.

5.3 Commands

Commands are data that BitVoicer sends to the microcontroller in response to one valid speech recognition. This data is wrapped in a specific protocol (refer to item 5.1) and can be of the following types:

- **Byte:** one byte written in the format b00000000, where 0 can be 1 or 0;
- **Char**: only one alpha-numeric character;
- **Int**: 16 bits integer number from -32768 to 32767;
- **String**: a sequence of characters containing a maximum of fifty-nine characters.

The command board, which is located right below the sentence constructor, is where you enter the commands for each anagram in the Voice Schema. The picture below shows part of the command board and some example commands.



Data Type	Command
Char	Z
Int	256
Byte	b00100101
String	Temperature

5.4 BitVoicer Arduino Library

The installation package of BitVoicer contains an open source library that can be used as reference to your sketches. This library can be found in the *Library* folder of the BitVoicer installation folder. You can also include this library directly into your code to retrieve data sent from BitVoicer or to perform audio capture.

To be able to use the BitVoicer Library, you need to copy the whole BitVoicer11 folder into the ***libraries*** sub-folder located in the installation folder of the Arduino IDE. Then add the reference to the BitVoicer Library at the beginning of your code as shown below:

```
#include <BitVoicer11.h>
```

For further information about how to use libraries in your sketches, refer to the library tutorial available at the Arduino's website (http://arduino.cc/en/Hacking/LibraryTutorial).

Note 1: The BitVoicer Arduino Library was design to work with all Arduino boards, except the Arduino Due board.

Note 2: By default, the BitVoicer library uses the *Serial* class to implement serial communication. If you need to use any other serial interfaces (*Serial1*, *Serial2* or *Serial3*), you must replace all occurrences of the *Serial* class in the BitVoicer Library with the desired serial interface.

The core of the BitVoicer Arduino Library is the BitVoicerSerial class. This class contains several variables and functions that allow you to communicate with BitVoicer and retrieve data sent from the application. To gain access to these functions, create a global instance of the BitVoicerSerial class by adding the piece of code below before the *setup()* and *loop()* functions in your sketch:

```
BitVoicerSerial bvSerial = BitVoicerSerial();
```

Before you use the instance created above, you will need to start the serial communication of your Arduino board. To do so, enter the piece of code below inside the *setup()* function of your sketch:

```
Serial.begin(115200);
```

Remember that the above baud rate must be the same as the one entered in the BitVoicer's preferences (item 3.3) if the communication is performed through a serial interface.

At this point, all the required configurations to start interacting with BitVoicer are concluded. Refer to the following topic for further information about all the variables and functions available in the BitVoicerSerial class. Item 5.5 in this manual contains two practical examples about how to turn a LED wired to your board on and off. The first example uses audio captured by a microphone installed on the PC and the second one uses a microphone wired to the microcontroller.

5.4.1 BitVoicer Arduino Library Reference

- **Constructor:**
  - **BitVoicerSerial()**: returns an instance of the BitVoicerSerial class with all the public variables set to their default values.
- **Constants:**
  - **BUFFER_SIZE**: contains the default value for the internal buffer of the BitVoicerSerial class. Value = 64;
  - **BV_DEFAULT**: used to set the reference voltage on the analog input pins. Value = 3;
  - **BV_EXTERNAL**: used to set the reference voltage on the analog input pins. Value = 1;
  - **BV_UNDEFINED**: can be used to check the data type returned by the *getData()* function. Value = 0;
  - **BV_CHAR**: can be used to check the data type returned by the *getData()* function. Value = 1;
  - **BV_INT**: can be used to check the data type returned by the *getData()* function. Value = 2;
  - **BV_BYTE**: can be used to check the data type returned by the *getData()* function. Value = 3;
  - **BV_STR**: can be used to check the data type returned by the *getData()* function. Value = 4;
  - **BV_COMMAND**: can be check to test the data type returned by the *getData()* function. Value = 254;

    - **BV_STATUS**: can be used to check the data type returned by the *getData()* function. Value = 255;
    - **BV_STARTSAMPLING**: can be used to check the data in a command datagram. Value = 1;
    - **BV_STOPSAMPLING**: can be used to check the data in a command datagram. Value = 0;
    - **BV_ENGINERUNNING**: can be used to check the status of the speech recognition engine set in the public variable *engineRunning* of the BitVoicerSerial class. Value = 1;
    - **BV_ENGINESTOPPED**: can be used to check the status of the speech recognition engine set in the public variable *engineRunning* of the BitVoicerSerial class. Value = 0.
- **Public Variables**:
    - **byte byteData**: stores the result of the last reading made by the *getData()* function if there is any data in the input buffer, and this data is of the **byte** type. Can be accessed and changed at any time by the user's code. Default = 0;
    - **char charData**: stores the result of the last reading made by the *getData()* function if there is any data in the input buffer, and this data is of the **char** type. Can be accessed and changed at any time by the user's code. Default = 0;
    - **int intData**: stores the result of the last reading made by the *getData()* function if there is any data in the input buffer, and this data is of the **int** type. Can be accessed and changed at any time by the user's code. Default = 0;
    - **String strData**: stores the result of the last reading made by the *getData()* function if there is any data in the input buffer, and this data is of the **String** type. Can be accessed and changed at any time by the user's code. Default = 0;
    - **byte cmdData**: stores the result of the last reading made by the *getData()* function if there is any data in the input buffer, and this data is of the **command** type. Can be accessed and changed at any time by the user's code. This variable will only be changed by the *getData()* function while running the Audio Streaming Calibration tool (item 3.4.2). Default = 0;
    - **boolean engineRunning**: stores the result of the last reading made by the *getData()* function if there is any data in the input buffer, and this data is of the **status** type. Can be accessed and changed at any time by the user's code. This variable will only be changed by the *getData()* function when the speech recognition engine is started or stopped (item 4.5). Default = false;
- **Functions:**
    - **getData()**: performs one reading of the input serial buffer of the Arduino, stores the data of the BitVoicer datagram in the proper public variable and returns the data type contained in the datagram. If the datagram is corrupted or does not match the format of the BitVoicer Protocol, the data in the buffer is disposed until the beginning of the next datagram and the function returns ZERO (BV_UNDEFINED);
    - **setAudioInput(byte pin)**: sets up the Arduino to perform fast analog readings at the analog input pin passed to the function. If you do not call this function, the analog-to-digital converter (ADC) of the ATMEL microcontroller will work with a slow clock. Once BitVoicer requires only 8-bit samples to process audio stream, increasing the clock speed of the ADC will not compromise its accuracy, and it will enable Arduino to capture at least 8000 samples per second;
    - **undoAudioInput()**: sets the ADC parameters to their factory values;
    - **setAnalogReference (byte mode)**: changes the reference voltage used by the analog-to-digital converter (ADC). By default, Arduino boards use 5 or 3.3 volts as voltage reference when performing analog readings. If any reference voltage is connected to the AREF pin, it is mandatory to call this function passing the constant BV_EXTERNAL as parameter before calling *setAudioInput()* or *processAudio()*. **If you fail to comply with**

this mandatory requirement, the internal reference voltage of the Arduino and the voltage at AREF will be shorted, which may permanently damage your board.

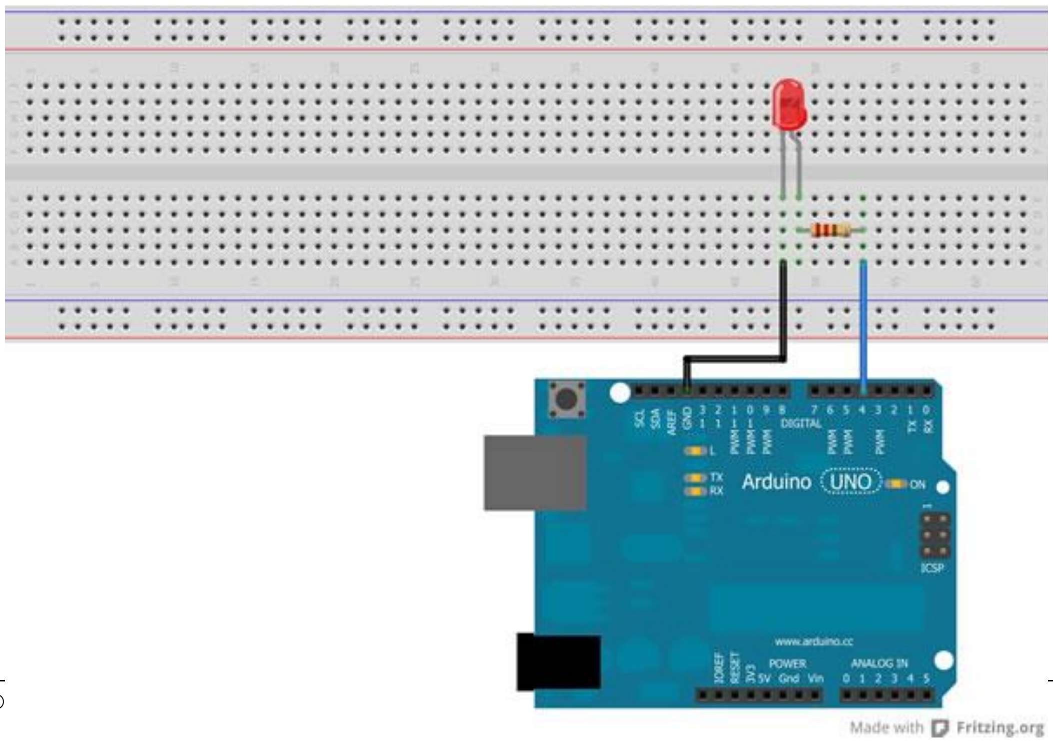
- **startStopListening()**: signalizes to BitVoicer to start or stop processing data received from the microcontroller as audio stream. Before starting the audio stream transmission, you must call this function so that the BitVoicer starts directing all the received data to the speech recognition engine and not to the Communication Monitor. Once the transmission was started, if you want to send any data to be displayed in the Communication Monitor, you must call this function again so that BitVoicer stops processing data as audio stream and directs it to the Communication Monitor. If you do not call this function before sending data to the Communication Monitor and BitVoicer is "listening" to audio stream, the data you send will be considered part of the audio and sent to the speech recognition engine. The signal sent to BitVoicer is a sequence of bytes with the following values: { 255, 255, 255, 255, 0, 0, 0, 0 }.
- **processAudio(unsigned int wait)**: performs one fast reading at the analog pin passed to the *setAudioInput()* function. The value passed to this function is the time the function will wait before performing one reading. It must be passed in microseconds (1 microsecond = 1000 milliseconds). This time can be used to calibrate the readings so that 8000 readings per second are achieved. As soon as the reading is complete, the function sends the result to BitVoicer (1 byte – unsigned int from 0 to 255). For this reason, this function must be called at regular intervals to prevent distortions in the digitalized audio wave.
- **sendToBV(var data)**: wraps the data passed to the function in a BitVoicer datagram and sends this data to BitVoicer in order to display it in the Communication Monitor. The accepted date types are char, byte, int or String. In case BitVoicer is processing audio stream, it is necessary to call the *startStopListening()* function before and after calling this function.

5.5 BitVoicer Library Examples

5.5.1 Audio Captured by the Computer's Microphone

In the example below, a LED connected to the digital pin 4 of the Arduino will turn on and off in response to a voice command captured by the computer's microphone.

First, wire your Arduino as shown in the picture below:



Then, copy and paste the sketch below in the Arduino IDE and upload it to the board. Remember that you need to copy the folder with the files of the BitVoicer Library to the ***library*** sub-folder of the Arduino IDE before compiling and uploading your sketch (refer to item 5.4).

```
//Imports the BitVoicer library to the sketch
#include <BitVoicer11.h>

//Instantiates the BitVoicerSerial class
BitVoicerSerial bvSerial = BitVoicerSerial();
//Stores the data type retrieved by getData()
byte dataType = 0;
//Stores the state of pin 4
byte pinVal = 0;

void setup()
{
  //Starts serial communication at 9600 bps
  Serial.begin(9600);
  //Sets digital pin 4 as OUTPUT
  pinMode(4, OUTPUT);
  //Turns off pin 4
  digitalWrite(4, pinVal);
}

void loop()
{
  //Updates the state of pin 4 on every loop
  digitalWrite(4, pinVal);
}

//This function runs every time serial data is available
//in the serial buffer after a loop
void serialEvent()
{
  //Reads the serial buffer and stores the received data type
  dataType = bvSerial.getData();

  //Checks if the data type is the same as the one in the
  //Voice Schema
  if (dataType == BV_BYTE)
  {
    //Checks the stored value in byteData by getData() and
    //changes the value of the pin
    if (bvSerial.byteData == 0)
      pinVal = LOW;
    else
      pinVal = HIGH;
  }
}
```

Finally, open the Voice Schema named *Sample2.vsc*, located in the sub-folder VoiceSchemas of the BitVoicer installation folder, and click on the *Start* button to start the speech recognition. The LED connected to the Arduino board should go on if you say "turn LED on" and go off if you say "turn LED off".

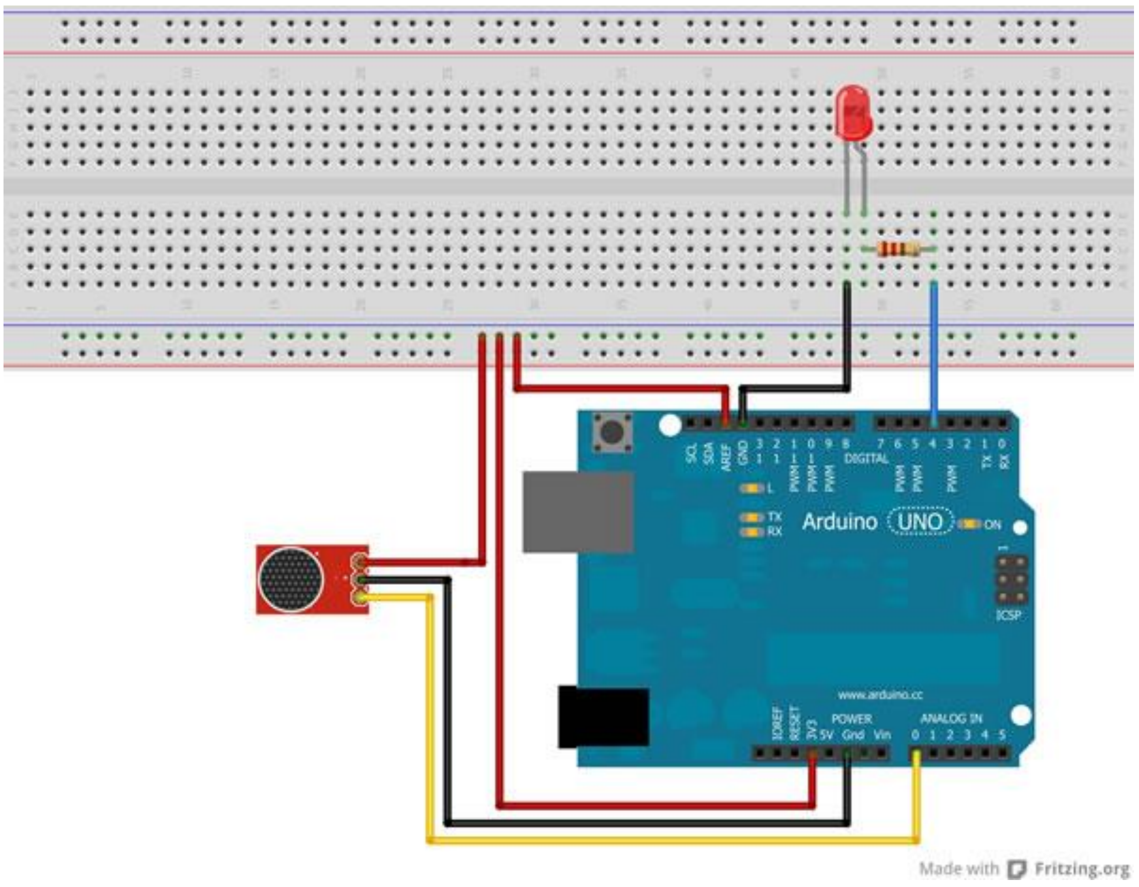
If you do not see any changes on the LED, make sure that:

- You selected English as the recognition language in the preferences form. If English is not available on your computer, select the language of your choice and translate the sentence in the Voice Schema to the selected language;
- The communication settings are correct in the preferences form (item 3.3);
- The serial speed (baud rate) is the same in the sketch and in the preferences;
- The audio captured by the computer's microphone has an appropriate level for recognition (items 3.2.3 and 7.1).

5.5.2 Audio Captured by a Microphone Wired to the Microcontroller

In the example below, a LED connected to the digital pin 4 of the Arduino will turn on and off in response to a voice command captured by a microphone wired to the microcontroller. The microphone used is an electret microphone from Sparkfun (BOB-09964 - https://www.sparkfun.com/products/9964).

First, wire your Arduino as shown in the picture below:



Important: in the above wiring notice that the 3.3 volt pin of the Arduino is used as analog reference voltage (AREF) for the analog-to-digital converter (ADC). This set up was necessary because, although Sparkfun states that this microphone works with 2.7V to 5.5V, it was not possible to get reliable readings from this microphone when it was connected to the 5 volt pin. For this reason, it was necessary to connect the microphone to the 3.3V pin on the board and the AREF pin (analog reference) to the same 3.3V source. Using this setup, it was possible the get readings from 0 to 255 (8-bits) and silence readings varying between 128 and 129. **Warning: do not use the Arduino's analogRead() function if any voltage**

reference is connected to AREF! It can damage your board if the *analogReference()* function is not called before *analogRead()*. In the code below, the *setAnalogReference()* function of the BitVoicer Library is called to properly set up the voltage reference for the ADC. **This function MUST be called before any other function of the BitVoicerSerial class if you are using any voltage reference at AREF.**

Then, copy and paste the sketch below in the Arduino IDE and upload it to the board. Remember that you need to copy the folder with the files of the BitVoicer Library to the ***library*** sub-folder of the Arduino IDE before compiling and uploading your sketch (refer to item 5.4).

```
//Imports the BitVoicer library to the sketch
#include <BitVoicer11.h>

//Instantiates the BitVoicerSerial class
BitVoicerSerial bvSerial = BitVoicerSerial();
//Stores the data type retrieved by getData()
byte dataType = 0;
//Stores the state of pin 4
byte pinVal = 0;
//Stores true if the Audio Streaming Calibration tool
//is running
boolean sampleTest = false;

void setup()
{
  //Sets the analog reference to external (AREF pin)
  //WARNING!!! If anything is conected to the AREF pin,
  //this function MUST be called first. Otherwise, it will
  //damage the board.
  bvSerial.setAnalogReference(BV_EXTERNAL);
  //Sets up the microcontroller to perform faster analog reads
  //on the specified pin
  bvSerial.setAudioInput(0);
  //Starts serial communication at 115200 bps
  Serial.begin(115200);
  //Sets digital pin 4 as OUTPUT
  pinMode(4, OUTPUT);
  //Turns off pin 4
  digitalWrite(4, pinVal);
}

void loop()
{
  //Captures audio and sends it to BitVoicer if the Audio
  //Streaming Calibration Tool is running
  if (sampleTest == true)
  {
    //The value passed to the function is the time
    //(in microseconds) that the function has to wait before
    //performing the reading. It is used to achieve about
    //8000 readings per second.
    bvSerial.processAudio(46);
  }

  //Captures audio and sends it to BitVoicer if the Speech
  //Recognition Engine is running
  if (bvSerial.engineRunning)
  {
    //The value passed to the function is the time
    //(in microseconds) that the function has to wait before
    //performing the reading. It is used to achieve about
    //8000 readings per second.
    bvSerial.processAudio(46);
  }

  //Updates the pin 4 state on every loop
  digitalWrite(4, pinVal);
}

//This function runs every time serial data is available
//in the serial buffer after a loop
void serialEvent()
{
  //Reads the serial buffer and stores the received data type
  dataType = bvSerial.getData();

  //Changes the value of sampleTest if the received data was
  //the start/stop sampling command
  if (dataType == BV_COMMAND)
      sampleTest = bvSerial.cmdData;

  //Signals BitVoicer's Speech Recognition Engine to start
  //listening to audio stream after the engineRunning status
  //was received
  if (dataType == BV_STATUS && bvSerial.engineRunning == true)
    bvSerial.startStopListening();

  //Checks if the data type is the same as the one in the
  //Voice Schema
  if (dataType == BV_BYTE)
  {
    //Checks the value stored in byteData by getData() and
    //changes the pin value
    if (bvSerial.byteData == 0)
      pinVal = LOW;
    else
      pinVal = HIGH;
  }
}
```

Finally, open the Voice Schema named *Sample2.vsc*, located in the sub-folder VoiceSchemas of the BitVoicer installation folder, and click on the *Start* button to start the speech recognition. The LED connected to the Arduino board should go on if you say “turn LED on” and go off if you say “turn LED off”.

If you do not see any changes on the LED, make sure that:

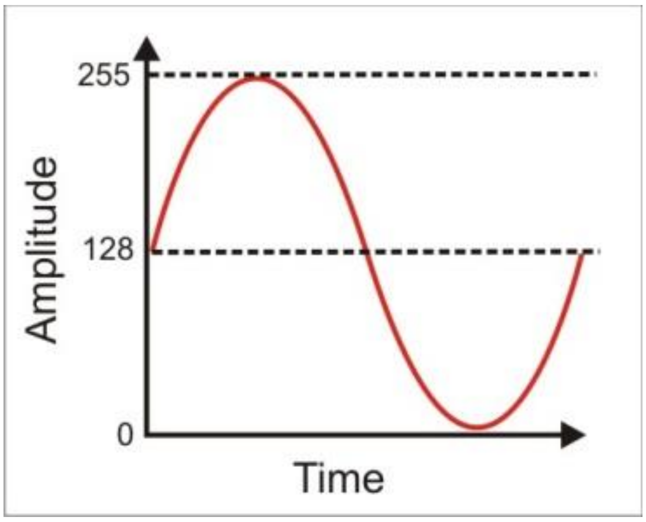
- You selected English as the recognition language in the preferences form. If English is not available on your computer, select the language of your choice and translate the sentence in the Voice Schema to the selected language;
- The communication settings are correct in the preferences form (item 3.3);
- The serial speed (baud rate) is the same in the sketch and in the preferences;
- The audio captured by the microphone has an appropriate level for recognition (items 3.2.3, 3.4.1 and 7.1);
- BitVoicer is receiving between 8000 and 8200 audio samples per second (items 3.4.2 and 6).

6 Audio Streaming

BitVoicer can process speech captured by a microphone wired to the microcontroller and sent to BitVoicer as audio streams. Combining audio streaming and wireless communication, it is possible to implement advanced speech recognition features on any microcontroller and keep the microcontroller physically apart from the computer.

The audio streams sent to BitVoicer must be digitalized using 8-bit PCM (http://en.wikipedia.org/wiki/Pulse-code_modulation) at 8000 samples per second. In simple terms, it is necessary to make 8000 readings per second of the voltage at the microphone, convert this reading (ADC = analog-to-digital converter) into an 8-bit value (from 0 to 255) and send this value to BitVoicer at regular intervals. Therefore, in essence, audio streaming is a sequence of bytes, sent one by one at regular intervals, that describe the pressure of a sound wave through time.

To keep the quality of the speech recognition, the phase (http://en.wikipedia.org/wiki/Sine_wave), that is, the digitalized value captured by a microphone in a silent environment, must be equal to 128 or 129. The number of samples per second, on the other hand, must be between 8000 and 8200.



Other factors may significantly affect the performance of the BitVoicer's speech recognition engine:

- The readings must be taken at regular intervals. To achieve 8000 readings per second, one reading must be taken each 125 microseconds. In this interval you must consider:
  - The time to perform the reading;
  - The time to convert the reading;
  - The time spent by the microcontroller while running other procedures between the readings.
- Oscillations in the voltage supplied to the microphone;

- Voltage fluctuation at the reading port;
- Inappropriate grounding;
- Interference from other circuits;
- Communication errors.

6.1 Signaling to BitVoicer

By default, BitVoicer expects to receive datagrams from the microcontroller. These datagrams must be formatted according to the parameters of the BitVoicer Protocol (item 5.1). For the BitVoicer to stop processing data as datagrams and start processing it as audio stream, it is necessary to send a signal to the speech recognition engine. This signal consists of an 8-byte sequence with the following values: { 255, 255, 255, 255, 0, 0, 0, 0 }.

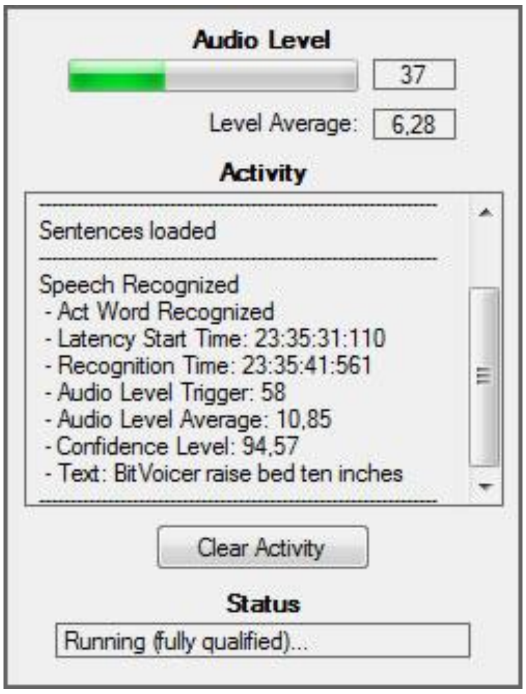
For the BitVoicer to resume to its initial state, when the received data are considered to be datagrams and its content is displayed in the Communication Monitor, another identical signal ({ 255, 255, 255, 255, 0, 0, 0, 0 }) must re-sent to BitVoicer. Therefore, if BitVoicer is processing audio stream and you need to send data to the Communication Monitor, the following bytes must be sent to the application: { 255, 255, 255, 255, 0, 0, 0, 0 } + BitVoicer datagram + { 255, 255, 255, 255, 0, 0, 0, 0 }. This cycle tells BitVoicer to: stop processing audio stream, process the anagram, and then resume "listening" to the audio stream.

7 Information Panels

The information panels are located on the right portion of the Voice Schema Editor. These panels remain inactive while editing a schema, but when the speech recognition is started, they show valuable information about the application.

7.1 Audio Level, Activity, Status and Errors

Displays information generated by the speech recognition engine.



```
Sentences loaded

Speech Recognized
 - Act Word Recognized
 - Latency Start Time: 23:35:31:110
 - Recognition Time: 23:35:41:561
 - Audio Level Trigger: 58
 - Audio Level Average: 10,85
 - Confidence Level: 94,57
 - Text: BitVoicer raise bed ten inches
```

Running (fully qualified)...

Note: if the BitVoicer is receiving audio stream captured in a silent environment, and nevertheless, the audio level constantly reaches a value greater than 4, it means that BitVoicer is receiving excessive noise (http://en.wikipedia.org/wiki/Noise). This noise can significantly compromise the performance of the speech recognition (refer to item 6).

These are the descriptions of the items in this panel:

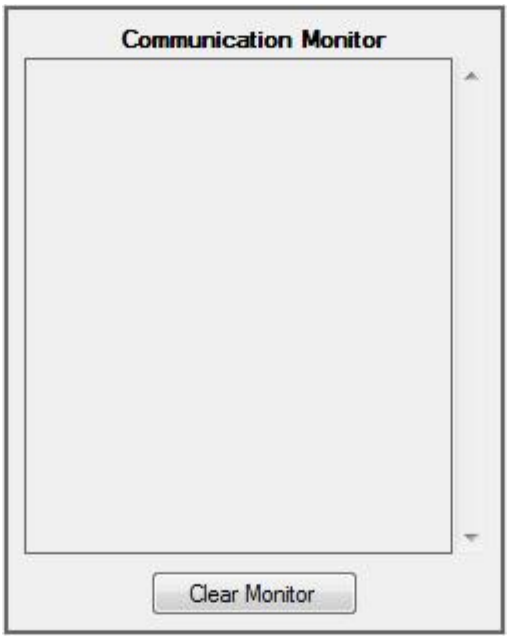
- ***Audio Level***: the audio level bar and the field by its side display the audio level captured by the speech recognition engine in real time. This information is updated ten times per second.
- ***Level Average***: this average is calculated based on the period entered in the option *Audio Level Activated Period* of the preferences, that is, the average is calculated for this period and the readings, are taken each 100 milliseconds.
- ***Activity***: this panel displays information about errors, initialization and disposing of the speech recognition engine, as well as information about the sending commands and the results of speech recognitions. These are the messages you can get in this panel:
    - ***Loading recognition engine...***
    - ***Error: unable to load recognition engine***: check the settings in the preferences, save them again and try to restart the speech recognition engine. If it does not help, contact the BitSophia's support team (contact information available in the item 7 of this manual).
    - ***Loading sentences...***
    - ***Error: unable to load sentences***: check if your Voice Schema has any blank sentence item and make sure you only used valid characters for the language selected in the preferences. If it does not help, contact the BitSophia's support team (contact information available in the item 7 of this manual).
    - ***Sentences loaded***
    - ***Error: unable to open serial port***: check if your microcontroller is properly connected and make sure the serial port entered in the preferences is correct and unblocked. If you do not have a microcontroller, disable the communication at *File* → *Preferences*.
    - ***Error: unable to establish IP connection***: check if the destination IP and the TCP port are available and unblocked. Also check the settings about the destination IP at *File* → *Preferences*. If you do not have a microcontroller, disable the communication at *File* → *Preferences.*
    - ***Disposing recognition engine...***
    - ***Engine disposed (stopped)***
    - ***Error: unable to create command data***: the command for the recognized anagram is empty.
    - ***Error: unable to write to serial port***: check if your microcontroller is properly connected and make sure the serial port entered in the preferences is correct and unblocked.
    - ***Command sent to serial port***
    - ***Error: unable to send TCP/IP packet***: check if the destination IP and the TCP port are available and unblocked. Also check the settings about the destination IP at *File* → *Preferences*.
    - ***Command sent to destination IP***
    - ***Speech Rejected***
        - *- Confidence: XXX %*
        - *- Text: XXX*
    - ***Speech Rejected***
        - *- Low Confidence Level*: the acceptable confidence level was not reached.
        - *- Confidence Level: XXX*

    - - *Text: XXX:* anagram whose confidence level was too low.
- ***Speech Rejected***
    - - *Audio Level Limit Not Reached*: refer to item 3.2.3
    - - *Max Audio Level: XX*: maximum audio level during the recognition.
    - - *Audio Level Average: XX*: audio level average at the moment of recognition.
    - - *Confidence Level: XXX*
    - - *Text: XXX*: rejected anagram.
- ***Speech Rejected***
    - - *In Latency Period*: refer to item 3.2.5
    - - *Latency Start Time: [HH:MM:SS:MMM]*: beginning of the latency period in hour, minute, second and millisecond.
    - - *Recognition Time: [HH:MM:SS:MMM]*: moment of recognition in hour, minute, second and millisecond.
    - - *Audio Level Trigger: XX*: audio level that started the *Audio level activated period* (refer to item 3.2.3).
    - - *Audio Level Average: XX*: audio level average at the moment of recognition.
    - - *Confidence Level: XXX*
    - - *Text: XXX*: rejected anagram.
- ***Speech Rejected***
    - - *Act Word Not Recognized*: the Voice Schema requires the recognition of the activation word (refer to item 4.4), but this word was not identified.
    - - *Latency Start Time: [HH:MM:SS:MMM]*: beginning of the latency period in hour, minute, second and millisecond.
    - - *Recognition Time: [HH:MM:SS:MMM]*: moment of recognition in hour, minute, second and millisecond.
    - - *Audio Level Trigger: XX*: audio level that started the *Audio level activated period* (refer to item 3.2.3).
    - - *Audio Level Average: XX*: audio level average at the moment of recognition.
    - - *Confidence Level: XXX*
    - - *Text: XXX*: rejected anagram.
- ***Speech Recognized***
    - - *Latency Start Time: [HH:MM:SS:MMM]*: beginning of the latency period in hour, minute, second and millisecond.
    - - *Recognition Time: [HH:MM:SS:MMM]*: moment of recognition in hour, minute, second and millisecond.
    - - *Audio Level Trigger: XX*: audio level that started the *Audio Level Activated Period* (refer to item 3.2.3). It will always be the value that started the period and not the value of the current recognition.
    - - *Audio Level Average: XX*: audio level average at the moment of recognition. It will always be the value at the beginning of the period and not the value of the current recognition.
    - - *Confidence Level: XXX*
    - - *Text: XXX*: rejected anagram.
- ***Speech Recognized***
    - - *Under Act Word Period*: the activation word was previously recognized and the current recognition happened inside the activated period by activation word. (item 4.4).
    - - Act Word Start Time: *[HH:MM:SS:MMM]*: beginning of the activated period by activation word in hour, minute, second and millisecond.
    - - *Latency Start Time: [HH:MM:SS:MMM]*: beginning of the latency period in hour, minute, second and millisecond.

    - - *Recognition Time: [HH:MM:SS:MMM]*: moment of recognition in hour, minute, second and millisecond.
    - - *Audio Level Trigger: XX*: audio level that started the *Audio Level Activated Period* (refer to item 3.2.3). It will always be the value that started the period and not the value of the current recognition.
    - - *Audio Level Average: XX*: audio level average at the moment of recognition. It will always be the value at the beginning of the period and not the value of the current recognition.
    - - *Confidence Level: XXX*
    - - *Text: XXX*: rejected anagram.
  - ***Speech Recognized***
    - - Act Word Recognized: the activation word was identified during the current recognition. The activated period by activation word was started (item 4.4).
    - - *Latency Start Time: [HH:MM:SS:MMM]*: beginning of the latency period in hour, minute, second and millisecond.
    - - *Recognition Time: [HH:MM:SS:MMM]*: moment of recognition in hour, minute, second and millisecond.
    - - *Audio Level Trigger: XX*: audio level that started the *Audio Level Activated Period* (refer to item 3.2.3). It will always be the value that started the period and not the value of the current recognition.
    - - *Audio Level Average: XX*: audio level average at the moment of recognition. It will always be the value at the beginning of the period and not the value of the current recognition.
    - - *Confidence Level: XXX*
    - - *Text: XXX*: rejected anagram.
- ***Status:*** displays status information about the speech recognition engine:
  - ***Stopped***: speech recognition engine stopped.
  - ***Starting...***: speech recognition engine is running the initialization process.
  - ***Connecting...***: connecting to the microcontroller.
  - ***Running...***: speech recognition engine up and running.
  - ***Running (in activation word period)...***: refer to item 4.4.
  - ***Running (audio level limit reached)...***: refer to item 3.2.3.
  - ***Running (fully qualified)...***: the *Minimum audio level* was reached and the *Activation word* was identified.

7.2 Communication Monitor

Through the Communication Monitor it is possible to visualize text information sent from your microcontroller. This panel can be used to receive status information, sensor readings or code debugging.



The *Clear Monitor* button, located at the bottom of the panel, clears all information sent from the microcontroller.

Note: if the BitVoicer is receiving audio stream sent from the microcontroller, it is necessary to signal to BitVoicer before sending information to be displayed on this panel (refer to item 6.1 for further information about signaling).

8 Support and Contact

- Support: support@bitsophia.com
- Sales: sales@bitvoicer.com
- Feedback: feedback@bitsophia.com