Copyright 2026 Synthiam Inc.

Python API

Overview

Python in Synthiam ARC

Python, known for its simplicity and wide ecosystem, is fully integrated into Synthiam ARC, a powerful platform for robot programming and automation. ARC combines a user-friendly interface with advanced robotics tools for hobbyists and professionals alike.

By offering Python as a language option, ARC lets you use clear syntax and popular libraries to implement complex behaviors, algorithms, and integrations. This combination makes it easier to develop dynamic, intelligent, and responsive robots while keeping the platform accessible and flexible.

Python version: ARC uses the Python 3 compiler.

Additional Python Modules

Additional Python module files (*.py) can be installed into your local ARC module folder:

`My Documents\ARC\Python Modules`
Place custom .py files in this folder to make them available to ARC scripts.

Installing Modules with pip

pip is the standard Python package installer. By default, pip installs packages into the system/user Python site-packages directory, which is different from ARC's module folder. To have pip install directly into ARC's Python Modules folder, configure pip's default target in a pip configuration file.

pip configuration file locations

- **Unix / macOS:** `$HOME/.pip/pip.conf`
- **Windows:** `%HOME%\pip\pip.ini` (typically `C:\Users\<username>\pip\pip.ini`)

Steps to configure pip for ARC

1. Create the pip directory if it does not exist (for example, `C:\Users\Bob\pip`).
2. Create or edit `pip.ini` (Windows) or `pip.conf` (Unix/macOS) and set the default target to ARC's Python Modules folder.

Example pip.ini content (Windows)

```
[global]
target=C:\Users\Bob\My Documents\ARC\Python Modules
```

Replace `C:\Users\Bob\My Documents\ARC\Python Modules` with your actual ARC Python Modules path.

ARC Python Helper Classes

In addition to Python's standard libraries, ARC exposes several helper classes and methods for robot control, sensor access, OS interaction, and more. You can use the editor's intellisense to discover available ARC classes and methods.

ADC — EZB analog-to-digital converter functions

Audio — streaming audio functions

COM — local serial COM port functions

Digital — EZB digital I/O functions

EZB — EZB-specific functions (voltage, connection status, etc.)

File — reading and writing local files

I2C — EZB I2C helper functions

Movement — control movement panel directions

Net — networking functions

Ping — EZB ultrasonic distance sensor functions

PWM — pulse width modulation on digital I/O pins

Servo — EZB servo movement functions

UART — EZB hardware UART and software serial functions

Utility — general utility functions

Tip: Enable intellisense in the ARC editor, type a class name, press the period (.) and browse available methods.

Root Commands (Global Helper Functions)

Root commands are global functions provided by ARC. They are not methods of a specific class and are useful for accessing ARC variables, sending commands to controls and other robot skills, and printing to the ARC console.

getVar(variableName)

Retrieve a value from ARC's public global variable storage. Robot skills publish these variables (for example, Camera, Auto Position, Speech Recognition, etc.).

- **variableName** — string name of the global variable (for example, "$Direction").
- **returns** — value of the global variable.

Example

```
// Get the current direction the robot is moving
var direction = getVar("$Direction")
```

setVar(variableName, value)

Store a value in ARC's public global variable storage so other robot skills or scripts can access it with getVar().

- **variableName** — string name of the global variable.
- **value** — value to store (string, number, boolean, etc.).

Example

```
// Set a value of 5 to be accessible by other robot skills
setVar("$MyValue", 5)
```

setVarObject(variableName, value)

Store an object in ARC's public variable storage without translation. Use this to pass entire objects such as functions, classes, or multidimensional arrays between robot skills. See the variable picker documentation for more details.

Read More

- **variableName** — string name of the global variable.
- **value** — object to store (array, object, function, etc.).

Example

```
// Set the multidimensional array to be accessible by other robot skills
x = [[0, 1, 2], [3, 4, 5]]
setVarObject("$MyValue", x)
```

varExists(variableName)

Check whether a given global variable exists in ARC's public storage.

- **variableName** — string name of the global variable.
- **returns** — boolean true/false indicating existence.

Example

```
// Print whether the variable exists
print(varExists("$MyValue"))
```

print(txt)

Write the given value to the ARC console output.

- **txt** — value or data to print (string, number, variable, etc.).

Example

```
// Print a string to the console output
print("Hello World")
```

sleep(timeMs)

Pause script execution for the specified number of milliseconds.

- **timeMs** — time in milliseconds to pause.

Example

```
// Print a string, pause for 5 seconds, and print another string
print("Hello")
sleep(5000)
print(" World")
```

sleepRandom(minTimeMs, maxTimeMs)

Pause execution for a random number of milliseconds between the minimum and maximum values.

- **minTimeMs** — minimum pause time in milliseconds.
- **maxTimeMs** — maximum pause time in milliseconds.

Example

```
// Print a string, pause for a random time between 1 and 5 seconds, then print another string
print("Hello")
sleepRandom(1000, 5000)
print(" World")
```

controlCommand(controlName, command, [parameters])

Send a command to a named control with optional parameters. Use the ARC cheat sheet or control interrogation to view supported commands for controls in your project.

- **controlName** — the name of the control (for example, "Camera").
- **command** — command string the control understands (for example, "StartCamera").
- **parameters** — optional parameters for the command.

Example

```
// Start the camera control
controlCommand("Camera", "StartCamera")
print("Camera has started")
```

showControl(controlName)

Display the specified user interface control and add it to the display stack.

Example

```
// Show the camera control
showControl("Camera")
```

closeControl(controlName)

Close the specified control on the interface and return to the previous control in the display stack.

Example

```
// Close the current control
closeControl()
```

Variables

Python variables are local to each script engine namespace. A variable created in one script engine (for example, the Camera control) is not automatically available in another script engine (for example, the WiiMote control).

To share data across robot skills and compilers, use ARC's public/global variable manager via `setVar`, `setVarObject`, and `getVar`.

Python Constants (Predefined)

ARC defines a set of constants that are available throughout the Python environment. These constants are numeric and do not require quotes when referenced. Common constants include:

D0 ... D23

V0 ... V99

ADC0 ... ADC7

Note: Numeric constants may be written with or without quotes, but string values must be quoted.

Add Custom Python Modules

Where to Install Additional Python Modules

Additional Python modules (`.py` files) can be installed to your user folder: **`My Documents\ARC\Python Modules`**. Place custom modules here so Synthiam ARC Python can import them directly.

Important Rules & Guidelines

- **Pure Python Only:** Install only modules written entirely in Python. Modules that require C extensions or compiled components are not compatible with Synthiam ARC Python.

- **Proper File Structure:** Each module should be either a single `.py` file or a properly structured package (a folder containing an `__init__.py` file).

- **Maintain Isolation:** Keep all custom modules inside the designated folder so they remain separate from system or application modules.

- **Version Compatibility:** Verify that any module you install is compatible with the version of Synthiam ARC Python you are using.

Using PIP with Synthiam ARC Python

PIP is the standard Python package installer. By default, PIP installs packages to a global location, which is different from the ARC folder. To instruct PIP to install modules into your **`My Documents\ARC\Python Modules`** folder, follow these steps.

1. **Locate or create the PIP configuration file**

   Windows: Create or edit `%HOME%\pip\pip.ini`
   Example: `C:\Users\Bob\pip\pip.ini` (replace `Bob` with your username).

2. **Edit the configuration file**
   Open `pip.ini` (or `pip.conf`) in a text editor and add the following lines, replacing

the example path with the path to your ARC Python folder:

```
[global]
target=C:\Users\Bob\My Documents\ARC\Python Modules
```

This tells PIP to install packages into the specified target directory.

3. **Install modules with PIP**
   Run PIP as you normally would. For example:
   `pip install requests`
   PIP will install the package files into the target directory defined in your configuration file.

4. **Verify the installation**
   Confirm that the installed package files are present in **`My Documents\ARC\Python Modules`**.

Note: When using PIP with Synthiam ARC Python, ensure all packages are pure Python. Packages that require compilation or include C extensions will not work properly.

Creating a Simple Example Module

A basic example module prints "Hello, World!" to the console. Create a file named `hello_world_module.py` inside **`My Documents\ARC\Python Modules`** with the following content:

```
# hello_world_module.py

def say_hello():
    print("Hello, World!")
```

To use this module in an ARC Python script, add a Script robot skill, switch to the Python tab, and include:

```
import hello_world_module

hello_world_module.say_hello()
```

When you run the script in ARC, it will import `hello_world_module` and execute `say_hello()`, printing "Hello, World!" to ARC's console output.

Example Module — TCP Client (ARCClient)

The following example demonstrates a simple TCP client module you can place in **`My Documents\ARC\Python Modules`** and save as `ARCClient.py` (case-sensitive name recommended).

This client connects to the Synthiam ARC TCP Server (enable the TCP Server in the Connection robot skill). The ARC TCP Server uses EZ-Script; see the ez-script manual for commands.

Create `ARCClient.py` with this code:

```
import socket

class ARCClient:
    def __init__(self, server_ip, server_port):
        self.server_ip = server_ip
        self.server_port = server_port
        self.sock = None
        self._version = "unknown"

    def connect(self):
        self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            self.sock.connect((self.server_ip, self.server_port))
            self._version = self._receive_response()
            print("Version:", self._version.decode() if isinstance(self._version, bytes) else
self._version)
        except Exception as e:
            print("Connection error:", str(e))

    def send_command(self, command):
        if self.sock:
            self.sock.sendall(command.encode())
            response = self._receive_response()
            return response.decode() if isinstance(response, bytes) else response
        return None

    def close(self):
        if self.sock:
            self.sock.close()
            self.sock = None

    def _receive_response(self):
        response = b''
        while True:
            data = self.sock.recv(1024)
            if not data:
                break
            response += data
            if b'\n' in data:
                break
        return response

    def get_version(self):
        return self._version

if __name__ == "__main__":
    server_ip = '127.0.0.1'
    server_port = 6666
    command = 'SayEZB("I am connected")\n'

    arc_client = ARCClient(server_ip, server_port)
    arc_client.connect()
    response = arc_client.send_command(command)
    print("Received response:", response)
    arc_client.close()
```

To use this module from an ARC Python script, add a Script robot skill and include:

```
from ARCClient import ARCClient

server_ip = '127.0.0.1'
server_port = 6666
command = 'SayEZB("I am connected")\n'

arc_client = ARCClient(server_ip, server_port)
```

```
arc_client.connect()
response = arc_client.send_command(command)
print("Received response:", response)
arc_client.close()
```

Remember to enable the TCP Server in the Connection robot skill so ARC will accept TCP connections.

Enable the ARC TCP Server in the Connection robot skill to allow remote TCP clients to connect.

ADC

get12Bit

```
ADC.get12Bit(port, [ezbIndex])
```

Parameters

port	Analog port to read from.
ezbIndex (optional)	Board index of the EZB to read from.

Returns
The value read from the specified ADC port as a 12 bit number.

Description
Returns the value read from the specified ADC port as a 12 bit number which ranges between 0 and 4095.

waitForBetween

```
ADC.waitForBetween(port, low, high, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
low	Inclusive lower bound on value to wait for.
high	Inclusive upper bound on value to wait for.
frequencyMs (optional)	How often the ADC port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The value read from the specified ADC port that is between or low and high. Returns -1 if timeout

Description
Suspends execution of the script until the value read from the specified ADC port is between low (inclusive) and high (inclusive). The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

waitForEquals

```
ADC.waitForEquals(port, value, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
value	Value to wait for.
frequencyMs (optional)	How often the port is checked in milliseconds.

ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The value read from the specified ADC port that is equal to value. Returns -1 if timeout

Description
Suspends execution of the script until the value read from the specified ADC port is equal to value. The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

waitForHigher

```
ADC.waitForHigher(port, value, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
value	Exclusive lower bound on value to wait for.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The value read from the specified ADC port that higher than value. Returns -1 if timeout

Description
Suspends execution of the script until the value read from the specified ADC port is greater than value. The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

waitForLower

```
ADC.waitForLower(port, value, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Analog port to read from.
value	Exclusive upper bound on value to wait for.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The value read from the specified ADC port that is lower than value. Returns -1 if timeout

Description
Suspends execution of the script until the value read from the specified ADC port is lower than value. The value read from the ADC port ranges between 0 and 255. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

get

```
ADC.get(port, [ezbIndex])
```

Parameters

port	Analog port to read from.
ezbIndex (optional)	Board index of the EZB to read from.

Returns

The value read from the specified ADC port.

Description

Returns the value read from the specified ADC port. The value read from the ADC port ranges between 0 and 255.

Audio

waitForSpeech

```
Audio.waitForSpeech(timeout, s1, [s2], [s3], [s4], [s5], [s6])
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
s1	String to wait for.
s2 (optional)	String to wait for.
s3 (optional)	String to wait for.
s4 (optional)	String to wait for.
s5 (optional)	String to wait for.
s6 (optional)	String to wait for.

Returns
The phrase heard if the timeout did not occur in a lowercase string.

Description
Suspends execution of the script until one of the strings specified by s1, s2, s3, s4, s5, or s6 is heard, or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout".
***Note: the returned string is in lowercase**

stopAudioFile

```
Audio.stopFile()
```

Returns
Nothing

Description
Stops the playback of the audio file started by playAudioFile(filename) that is playing through the computer's audio system.

speakVolume

```
Audio.speakVolume(volume)
```

Parameters

volume	Level to set the text-to-speech volume to.

Returns
Nothing

Description
Sets the text-to-speech volume level of the computer to volume. The volume level ranges between 0 and 100.

getVolume

```
Audio.getVolume([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to get volume level from.

Returns
The volume level of the specified EZB.

Description
Returns the volume level of the specified EZB. The volume level ranges between 0 (quite), 100 (loud), 200 (2x over drive).

stop

```
Audio.stop([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to stop playing audio.

Returns
Nothing

Description
Stops all audio currently playing on the EZB.

isConnected

```
Audio.isConnected([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to check if audio is connected.

Returns
True if audio is connected to the EZB. False otherwise.

Description
Returns true if audio is connected to the EZB. Returns false otherwise.

say

```
Audio.say(txt)
```

Parameters

txt	Text to say.

Returns
Nothing

Description
Speak the text given in the string txt using text-to-speech through the computer's audio system. Execution of the script will continue while the audio is playing.

sayEZB

```
Audio.sayEZB(txt, [ezbIndex])
```

Parameters

txt	Text to say.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns
Nothing

Description
Speak the text given in the string txt using text-to-speech through the EZB's audio system. Execution of the script will continue while the audio is playing.

sayEZBWait

```
Audio.sayEZBWait(txt, [ezbIndex])
```

Parameters

txt	Text to say.
ezbIndex(optional)	Board index of the EZB to output the audio from.

Returns
Nothing

Description
Speak the text given in the string txt using text-to-speech through the EZB's audio system. Execution of the script will be suspended until the audio is finished playing.

sayWait

```
Audio.sayWait(txt)
```

Parameters

txt	Text to say.

Returns
Nothing

Description
Speak the text given in the string txt using text-to-speech through the computer's audio system. Execution of the script will be suspended until the audio is finished playing.

setVolume

```
Audio.setVolume(volume, [ezbIndex])
```

Parameters

volume	Level to set the volume to.
ezbIndex (optional)	Board index of the EZB to set the volume level for.

Returns
Nothing

Description
Set the volume level of the EZB to volume. The volume level ranges between 0 (quite), 100 (loud), 200 (2x over drive).

soundNote

```
Audio.soundNote(note, length, [signalType], [ezbIndex])
```

Parameters

note	Note to play as a string.
length	How long to play the note for in milliseconds.
signalType (optional)	Type of signal to use when playing the note as a string.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns
Nothing

Description
Plays the specified note for length milliseconds. Notes are inputted as strings which range from "C1" to "Bb3". The note is played using the signal specified by signalType. If signalType is not provided a sine wave is used. Accepted signal types are "Sine", "Square", "Triangle", "Pulse", "Sawtooth", "WhiteNoise", "GaussNoise", "DigitalNoise". This function does not wait for the note to end to continue executing the script. In case of playing multiple notes in sequence, call the sleep() function to ensure the notes don't play over previous notes.

playAudioFile

```
Audio.playAudioFile(filename)
```

Parameters

filename	Filename of the audio file to play.

Returns
Nothing

Description
Plays the audio file at the location given by filename through the computer’s audio system.

setVolumePC

```
Audio.setVolumePC(volume)
```

Parameters

volume	Level to set the PC speaker volume to between 0-255

Returns
Nothing

Description
Set the volume level of the default audio output device (Soundcard). The volume level ranges between 0 and 255. With 0 (mute) and 255 is loudest

waitForSpeechRange

```
Audio.waitForSpeechRange(timeout, start, end, [increment])
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
start	numeric start value of the range
end	numeric end value of the range
increment	the increment of the range (optional)

Returns
The value within the range as a number or the word "timeout" if timeout is reached.

Description
Suspends execution of the script until one of the numbers within the range (start, end) is detected in the microphone by speech, or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout."
The returned value is a number (i.e., 23, 10, 5) and not a string (i.e., twenty-three, ten, five).

Each option will be displayed if the number of options within the range is less than 10. For example, if the start is 10 and the end is 20, there will be 10 options for each option to be displayed.

If there are more than ten options, the range will be displayed.

Large Number Ranges

This style of speech recognition populates the pre-defined responses it expects to hear. In this case, a range of numbers.
This dramatically increases the accuracy of the speech recognition system by limiting the detected phrase/words to the list.
By providing the list, the speech recognition system will generate definitions of the sound waveforms it is looking for.

While this method dramatically improves detection accuracy, it also can consume a lot of PC resources if an extensive number range is used.
For example, a range higher than 100 is considered an extensive range. You may notice a significant delay every time this function is called, and that is due to the vast range.

A recommended solution is to use the waitForAnyNumberSpeech() command.

An additional solution is to prompt for multiple digits that will generate a more significant number when added together with simple math. For example, if you require 1000 positions, consider prompting for each digit.

Example

```
// Get a number from the user within the range between 10 and 20
response = Audio.waitForSpeechRange(10, 10, 20);
Audio.say("You selected " + response);

// Get a number from the user within the range between 10 and 20 incremented by 2 (every even
number)
response = Audio.waitForSpeechRange(10, 10, 20, 2);
Audio.say("You selected " + response);
```

waitForAnySpeech

```
Audio.waitForAnySpeech(timeout, prompt)
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
prompt	The text prompt to display to the user while listening for speech

Returns
The phrase heard if the timeout did not occur in a lowercase string.

Description
Suspends execution of the script until any speech is detected in the microphone or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout."
***Note: the returned string is in lowercase**
This command uses the entire dictionary of speech recognition words.
Generally, using this dictionary size will have a very low recognition quality. This may work okay if you are feeding a chatbot (such as OpenAI or PandoraBot).
But, if you are looking for specific phrases, this will most likely not work, and you should use the waitForSpeech() command instead.
Using this command for open conversations is discouraged.

Example

```
// Listen for any response from the user
response = Audio.waitForAnySpeech(30, "Say anything to me");

print("Heard: " + response);
```

waitForAnyNumberSpeech

```
Audio.waitForAnyNumberSpeech(timeout, prompt)
```

Parameters

timeout	Time to wait in seconds before stopping waiting for speech.
prompt	The text prompt to display to the user while listening for speech

Returns
The number heard if the timeout did not occur in a lowercase string.

Description
Suspends execution of the script until any number is detected in the microphone or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout" in lowercase. The number can include a minus sign and a decimal point.

Example

```
// Listen for any response from the user
response = Audio.waitForAnyNumberSpeech(30, "Give me a number");

print("Heard: " + response);
```

saySSML

```
Audio.saySSML(ssml)
```

Parameters

ssml	Text to say.

Returns
Nothing

Description
Speak the formatted SSML given using text-to-speech through the computer's audio system. Execution of the script will continue while the audio is playing.
This support document explains SSML here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSML(str);
```

```
// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSML(str);
```

saySSMLEZB

```
Audio.saySSMLEZB(ssml, [ezbIndex])
```

Parameters

ssml	Text to say.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns
Nothing

Description
Speak the SSML document given in the string using text-to-speech through the EZB's audio system. Execution of the script will continue while the audio is playing.
Read about SSML in this support document here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSMLEZB(str);
```

```
// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSMLEZB(str);
```

saySSMLEZBWait

```
Audio.saySSMLEZBWait(ssml, [ezbIndex])
```

Parameters

ssml	Text to say.
ezbIndex (optional)	Board index of the EZB to output the audio from.

Returns
Nothing

Description
Speak the SSML document given in the string using text-to-speech through the EZB's audio system. Execution of the script will be suspended until the audio is finished playing.
Read about SSML formatted speech here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSMLEZBWait(str);
```

```
// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSMLEZBWait(str);
```

saySSMLWait

```
Audio.saySSMLWait(ssml)
```

Parameters

ssml	Text to say.

Returns
Nothing

Description
Speak the text given in the provided SSML using text-to-speech through the computer's audio system. Execution of the script will be suspended until the audio is finished playing.
Read about how to format SSML in this document here: https://www.w3.org/standards/history/speech-synthesis/

Example

```
// This is an example using the say-as to specify an input format and speak the date
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "<say-as type="date:mdy"> 1/29/2009 </say-as>" +
          "</speak>";

Audio.saySSMLWait(str);
```

```
// This example includes a pause of 500ms between the two words hello and there
var str = "<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-
US">" +
          "hello <break time="500ms"> there" +
          "</break></speak>";

Audio.saySSMLWait(str);
```

COM

available

```
COM.available(port)
```

Parameters

port	COM port name as a string.

Returns
True if the specified COM port is available. False otherwise.

Description
Returns true if the specified COM port is available. Returns false otherwise. port is provided as a string (e.g. “COM3”).

availablePorts

```
COM.availablePorts()
```

Returns
Array of available COM ports.

Description
Returns an array of available COM ports. Available COM ports are represented in the array as strings (e.g. “COM3”).

clearInputBuffer

```
COM.clearInputBuffer(port)
```

Parameters

port	COM port name as a string.

Returns
Nothing

Description
Clears all data from the input buffer of the specified COM port.

close

```
COM.close(port)
```

Parameters

port	COM port name as a string.

Returns
Nothing

Description
Closes the specified COM port.

isPortOpen

```
COM.isPortOpen(port)
```

Parameters

port	COM port name as a string.

Returns
True if the COM port is open. False otherwise.

Description
Returns true if the specified COM port is open. Returns false otherwise.

open

```
COM.open(port, baud)
```

Parameters

port	COM port name as a string.
baud	Baud rate of the port

Returns
Nothing

Description
Opens the specified COM port with a baud rate of baud. This function must be called before calling any other read or write functions to the COM port.

readAllText

```
COM.readAllText(port)
```

Parameters

port	COM port name as a string.

Returns
All data in the COM port as a string.

Description
Reads all data from the specified COM port and returns it as a string.

readLine

```
COM.readLine(port)
```

Parameters

port	COM port name as a string.

Returns
One line of data from the COM port as a string.

Description
Reads all data from the specified COM port until an endline character is reached. Value is returned as a string.

write

```
COM.write(port, byte/byteArray)
```

Parameters

port	COM port name as a string.
byte/byteArray	Byte or byte array to write to the COM port.

Returns
Nothing

Description
Writes one byte given by byte to the specified COM port. If a byte array is provided instead, the byte array given by byteArray will be written to the specified COM port.

writeString

```
COM.writeString(port, str)
```

Parameters

port	COM port name as a string.
str	String to write to the COM port.

Returns
Nothing

Description
Writes the string given by str to the specified COM port.

writeStringNewLine

```
COM.writeStringNewLine(port, str)
```

Parameters

port	COM port name as a string.
str	String to write to the COM port.

Description
Writes the string given by str with a newline character appended to the end to the specified COM port.

read

```
COM.read(port, bytesToRead)
```

Parameters

port	COM port name as a string.
bytesToRead	Number of bytes to read from the COM port.

Returns
A byte array of length bytesToRead containing the data read from the COM port.

Description
Reads bytesToRead bytes from the COM port and returns it in a byte array.

readByte

```
COM.readByte(port)
```

Parameters

port	COM port name as a string.

Returns
The first byte read from the COM port.

Description
Reads and returns one byte from the specified COM port.

Digital

get

```
Digital.get(port, [ezbIndex])
```

Parameters

port	Digital port to read from.
ezbIndex (optional)	Board index of the EZB to read from.

Returns
True if the specified digital port is 1 (high). False if the digital port is 0 (low).

Description
Reads and returns the digital value from the specified digital port.

set

```
Digital.set(port, value, [ezbIndex])
```

Parameters

port	Digital port to set the value of.
value	Value to set the digital port to as a boolean or integer.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Sets the value of the specified digital port to value.

setRandom

```
Digital.setRandom(port, [ezbIndex])
```

Parameters

port	Digital port to set the value of.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The random value that the digital port was set to.

Description
Randomly sets the value of the specified digital port to either 0 or 1.

toggle

```
Digital.toggle(port, [ezbIndex])
```

Parameters

port	Digital port to toggle.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The value toggled to.

Description
Toggles the digital port. If the digital port was 0 then it is set to 1. If the digital port was 1 then it is set to 0.

wait

```
Digital.wait(port, valueToWaitFor, [frequencyMs], [ezbIndex], [timeoutMS])
```

Parameters

port	Digital port to set the value of.
valueToWaitFor	Digital value to wait for as boolean or integer.
frequencyMs (optional)	How often the port is checked in milliseconds.
ezbIndex (optional)	Board index of the EZB to read from.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
Nothing

Description
Suspends execution of the script until the value read from the specified digital port is equal to valueToWaitFor. If frequencyMs is provided the port is checked every frequencyMs milliseconds. Otherwise the port is checked as often as possible.

EZB

getCPUTemp

```
EZB.getCPUTemp([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to read from.

Returns
CPU temperature of the EZB in degrees celsius.

Description
Returns the CPU temperature of the EZB in degrees Celsius.

getVoltage

```
EZB.getCPUTemp([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to read from.

Returns
Input voltage of the EZB in volts.

Description
Returns the input voltage of the EZB in volts.

isConnected

```
EZB.isConnected(ezbIndex)
```

Parameters

ezbIndex	Board index of the EZB to check if connected.

Returns
True if the EZB at board index ezbIndex is connected. False otherwise.

Description
Checks if the EZB at board index ezbIndex is connected or not.

File

setReadPosition

```
File.setReadPosition(filename)
```

Parameters

filename	Path of file to set read position of.
position	Read position to set to.

Returns
Nothing

Description
Sets the read position of the file to position.

readReset

```
File.readReset(filename)
```

Parameters

filename	Path of file to reset read position of.

Returns
Nothing

Description
Resets the read position of the file to 0.

readRandomLine

```
File.readRandomLine(filename)
```

Parameters

filename	Path of file to read from.

Returns
A random line from the file as a string.

Description
Reads and returns a random line from the file, which starts from the end of one line and ends at the next endline character. This function does not affect the read position of the file.

readLine

```
File.readLine(filename)
```

Parameters

filename	Path of file to read from.

Returns
The string from the current read position to the next endline character.

Description
Reads and returns all characters as a string from the current read position to the next endline character. The returned string does not contain the endline character. The read

position advances to be immediately after the endline character.

readClose

```
File.readClose(filename)
```

Parameters

filename	Path of file to close.

Returns
Nothing

Description
Closes the file at path filename that was being read from.

append

```
File.append(filename, byte/byteArray)
```

Parameters

filename	Path of file to append to.
byte/byteArray	Byte or byte array to append to the file.

Returns
Nothing

Description
Appends the byte given by byte to the file. If a byte array is provided instead, the byte array given by byteArray will be appended to the file.

appendString

```
File.appendString(filename, str)
```

Parameters

filename	Path of file to append to.
str	String to append to the file.

Returns
Nothing

Description
Appends the string given by str to the file.

appendStringLine

```
File.appendStringLine(filename, str)
```

Parameters

filename	Path of file to append to.
str	String to append to the file.

Returns
Nothing

Description
Appends the string given by str with a newline character appended to the end to the file.

delete

```
File.delete(filename)
```

Parameters

filename	Path of file to delete.

Returns
Nothing

Description
Deletes the file at the path specified by filename.

exists

```
File.exists(filename)
```

Parameters

filename	Path of file to check if exists.

Returns
True if the file at path filename exists. False otherwise.

Description
Checks if the file at path filename exists.

getLength

```
File.getLength(filename)
```

Parameters

filename	Path of file to get length of.

Returns
Number of characters in the file.

Description
Returns the number of characters contained in the file.

getReadPosition

```
File.getReadPosition(filename)
```

Parameters

filename	Path of file to get read position of.

Returns
Read position of the file as the number of characters that have been read since it was opened.

Description
Returns the read position of the file as the number of characters that have been read since it was opened.

isFileOpenForReading

```
File.isFileOpenForReading(filename)
```

Parameters

filename	Path of file to check if open for reading.

Returns
True if file is open for reading. False otherwise.

Description
Checks if the file at filename is open for reading or not. If a read function has been called on the file this will return true. If the file has not been opened for reading, or the file is closed using the readClose funciton then this will return false. A file will remain open for reading even after the script finishes executing unless the file is closed using the readClose function.

isReadEnd

```
File.isReadEnd(filename)
```

Parameters

filename	Path of file to check if at end.

Returns
True if the read position of the file is as at the end of the file. False otherwise.

Description
Checks if the read position of the file at filename is at the end of the file.

readAllText

```
File.readAllText(filename)
```

Parameters

filename	Path of file to read.

Returns
All text contained within the file as a string.

Description
Reads and returns all the text contained with the file at filename as a string.

readByte

```
File.readByte(filename)
```

Parameters

filename	Path of file to read from.

Returns
The ASCII code representation for the character at the current read position of the file.

Description
Reads and returns the ASCII code representation (one byte of data) of the character located at the current read position of the file, and then advances the read position by one character.

readChar

```
File.readChar(filename)
```

Parameters

filename	Path of file to read from.

Returns
The character at the current read position of the file.

Description
Reads and returns the character located at the current read position of the file, and then advances the read position by one character.

I2C

read

```
I2C.read(address, bytesToExpect, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
bytesToExpect	Number of bytes to read from the device.
ezbIndex (optional)	Board index of the EZB to read from.

Returns
A byte array of length bytesToExpect containing the bytes read from the device.

Description
Reads bytesToExpect bytes from the device at address and returns them as a byte array.

readByte

```
I2C.readByte(address, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
ezbIndex (optional)	Board index of the EZB to read from.

Returns
One byte read from the device.

Description
Reads one byte of data from the device at address.

readString

```
I2C.readString(address, bytesToExpect, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
bytesToExpect	Number of bytes to read from the device (1 byte = 1 character).
ezbIndex (optional)	Board index of the EZB to read from.

Returns
A string of length bytesToExpect.

Description
Reads a string of data of length bytesToExpect from the device at address.

setClockSpeed

```
I2C.setClockSpeed(speed, [ezbIndex])
```

Parameters

speed	Clock speed to set the I2C interface to.
ezbIndex (optional)	Board index of the EZB to set the I2C clock speed of.

Returns
Nothing

Description
Sets the clock speed of the I2C interface. The default speed is 100000, which is 100khz. Many devices support faster speeds, up to 400000 (400khz).

write

```
I2C.write(address, byte/byteArray, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
byte/byteArray	Byte or byte array to write to the device.
ezbIndex (optional)	Board index of the EZB to write to.

Returns
Nothing

Description
Writes byte to the device at address. A byte array can instead be provided to write to the device at address.

writeString

```
I2C.write(address, str, [ezbIndex])
```

Parameters

address	Address of the device to read from as a hexadecimal number (e.g.0x5e).
str	String to write to the device.
ezbIndex (optional)	Board index of the EZB to write to.

Returns
Nothing

Description
Writes the string given by str to the device at address.

Movement

setSpeed

```
Movement.setSpeed(speed/speedLeft, [speedRight])
```

Parameters

speed/speedLeft	Global Movement Speed value for left and right wheel. If speedRight is provided this value will be used for the global Movement Speed value of the left wheel only.
speedRight (optional)	Global movement speed value for the right wheel.

Returns
Nothing

Description
Sets the global movement speed value for the left and right wheels. If speedLeft and speedRight are provided the values for the left and right wheels will be set respectively.

setSpeedLeft

```
Movement.setSpeedLeft(speed)
```

Parameters

speed	Global movement speed value for the left wheel.

Returns
Nothing

Description
Sets the global movement speed value for the left wheel. **speed** is a value between 0 and 255.

setSpeedRight

```
Movement.setSpeedRight(speed)
```

Parameters

speed	Global movement speed value for the right wheel.

Returns
Nothing

Description
Sets the global movement speed value for the right wheel. speed is a value between 0 and 255.

stop

```
Movement.stop()
```

Returns
Nothing

Description
Stops the robot using the project's Movement Panel skill.

takeoff

```
Movement.takeoff()
```

Returns
Nothing

Description
Tells the robot to takeoff using project's Movement Panel skill. The Movement Panel skill must have takeoff capability.

up

```
Movement.up([timeOut])
```

Parameters

timeOut (optional)	Duration to move up for in milliseconds.

Returns
Nothing

Description
Moves the robot up using the project's Movement Panel skill. The Movement Panel skill must have the capability to move the robot up. If timeOut is provided the robot will move up for timeOut milliseconds. Otherwise it will continue to move up until a different Movement function is called.

waitForDown

```
Movement.waitForDown()
```

Returns
Nothing

Description
Suspends execution of the script until down is executed from the project's movement panel either by the user or from another script.

waitForLeft

```
Movement.waitForLeft()
```

Returns
Nothing

Description
Suspends execution of the script until left is executed from the project's movement panel either by the user or from another script.

waitForReverse

```
Movement.waitForReverse()
```

Returns
Nothing

Description
Suspends execution of the script until reverse is executed from the project's movement panel either by the user or from another script.

waitForRight

```
Movement.waitForRight()
```

Returns
Nothing

Description
Suspends execution of the script until right is executed from the project's movement panel either by the user or from another script.

waitForStop

```
Movement.waitForStop()
```

Returns
Nothing

Description
Suspends execution of the script until stop is executed from the project's movement panel either by the user or from another script.

waitForUp

```
Movement.waitForUp()
```

Returns
Nothing

Description
Suspends execution of the script until up is executed from the project's movement panel either by the user or from another script.

waitForForward

```
Movement.waitForForward()
```

Returns
Nothing

Description
Suspends execution of the script until forward is executed from the project's movement panel either by the user or from another script.

rollRight

```
Movement.rollRight([timeOut])
```

Parameters

timeOut (optional)	Duration to roll right for in milliseconds.

Returns
Nothing

Description
Rolls the robot right using the project's Movement Panel skill. The Movement Panel skill must have the capability to roll the robot right. If timeOut is provided the robot will roll right for timeOut milliseconds. Otherwise it will continue to roll right until a different Movement function is called.

rollLeft

```
Movement.rollLeft([timeOut])
```

Parameters

timeOut (optional)	Duration to roll left for in milliseconds.

Returns
Nothing

Description
Rolls the robot left using the project's Movement Panel skill. The Movement Panel skill must have the capability to roll the robot left. If timeOut is provided the robot will roll left for timeOut milliseconds. Otherwise it will continue to roll left until a different Movement function is called.

right

```
Movement.right([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move right for in milliseconds.

Returns
Nothing

Description
Moves the robot right using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move right for timeOut milliseconds. Otherwise it will continue to move right until a different Movement function is called.

reverse

```
Movement.reverse([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move reverse for in milliseconds.

Returns
Nothing

Description
Moves the robot reverse using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move reverse for timeOut milliseconds. Otherwise it will continue to move reverse until a different Movement function is called.

forward

```
Movement.forward([speed], [timeOut])
```

Parameters

speed	Speed to move at.

timeOut (optional)	Duration to move forward for in milliseconds.

Returns
Nothing

Description
Moves the robot forward using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move forward for timeOut milliseconds. Otherwise it will continue to move forward until a different Movement function is called.

land

```
Movement.land()
```

Returns
Nothing

Description
Lands the robot using the project's Movement Panel skill. The Movement Panel skill must have the capability to land the robot.

getSpeed

```
Movement.getSpeed()
```

Returns
The global movement speed value as a value between 0 and 255.

Description
Returns the global movement speed value as a value between 0 and 255. The global movement speed value is the speed used for the left and right wheels by the project's Movement Panel skill if it has the capability to do so.

getSpeedLeft

```
Movement.getSpeedLeft()
```

Returns
The global movement speed value of the left wheel as a value between 0 and 255.

Description
Returns the global movement speed value of the left wheel as a value between 0 and 255. The global movement speed value of the left wheel is the speed used for the left wheel by the project's Movement Panel skill if it has the capability to do so.

getSpeedRight

```
Movement.getSpeedRight()
```

Returns
The global movement speed value of the right wheel as a value between 0 and 255.

Description
Returns the global movement speed value of the right wheel as a value between 0 and 255. The global movement speed value of the right wheel is the speed used for the right wheel by the project's Movement Panel skill if it has the capability to do so.

left

```
Movement.left([speed], [timeOut])
```

Parameters

speed (optional)	Speed to move at.
timeOut (optional)	Duration to move left for in milliseconds.

Returns
Nothing

Description
Moves the robot left using the project's Movement Panel skill. The speed can be specified by providing speed as a value between 0 and 255. If timeOut is provided the robot will move left for timeOut milliseconds. Otherwise it will continue to move left until a different Movement function is called.

down

```
Movement.down([timeOut])
```

Parameters

timeOut (optional)	Duration to move down for in milliseconds.

Returns
Nothing

Description
Moves the robot down using the project's Movement Panel skill. The Movement Panel skill must have the capability to move the robot down. If timeOut is provided the robot will move down for timeOut milliseconds. Otherwise it will continue to move down until a different Movement function is called.

Net

sendUDP

```
Net.sendUDP(hostname, port, byteArray)
```

Parameters

hostname	Hostname of receiver as a string.
port	Port to use.
byteArray	Data to send as a byte array.

Returns
The length of the byteArray being sent.

Description
Sends byteArray over the specified port to the hostname using UDP.

hTTPPost

```
Net.hTTPPost(url, postData, [timeout])
```

Parameters

url	URL to send HTTP POST request to.
postData	Data to send as part of POST request as a string.
timeout (optional)	Timeout in milliseconds.

Returns
The HTTP POST request response as a string.

Description
Sends an HTTP POST request to url with postData stored in the request body. If timeout is specified the request will timeout after timeout milliseconds. Suspends script execution until the request completes, or the request times out.

isInternetAvailable

```
Net.isInternetAvailable()
```

Returns
True if the computer is connected to the internet. False otherwise.

Description
Checks the internet connection of the computer.

hTTPGet

```
Net.hTTPGet(url, [timeout])
```

Parameters

url	URL to send HTTP Get request to.
timeout (optional)	Timeout in milliseconds.

Returns

HTTP contents from the provided URL as a string.

Description

Sends an HTTP Get request to url. If timeout is specified the request will timeout after timeout milliseconds. Suspends script execution until the request completes, or the request times out.

Ping

waitForBetween

```
Ping.waitForBetween(triggerPort, echoPort, low, high, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
low	Inclusive lower bound on value to wait for.
high	Inclusive upper bound on value to wait for.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The value measured from the ultrasonic distance sensor that is between low and high as a number between 0 and 255.

Description
Suspends execution of the script until the value read from the ultrasonic distance sensor is between low (inclusvie) and high (inclusive). The value read from the ultrasonic distance sensor is returned when the script resumes execution.

waitForEquals

```
Ping.waitForEquals(triggerPort, echoPort, distance, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
distance	Value to wait for.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The value measured from the ultrasonic distance sensor that is equal to distance as a number between 0 and 255.

Description
Suspends execution of the script until the value read from the ultrasonic distance sensor is equal to distance. The value read from the ultrasonic distance sensor is returned when the script resumes execution.

waitForHigher

```
Ping.waitForHigher(triggerPort, echoPort, distance, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
distance	Exclusive lower bound on value to wait for

ezbIndex (optional)	Board index of the EZB to use.

Returns
The value measured from the ultrasonic distance sensor that is higher than distance as a number between 0 and 255.

Description
Suspends execution of the script until the value read from the ultrasonic distance sensor is higher than distance. The value read from the ultrasonic distance sensor is returned when the script resumes execution.

waitForLower

```
Ping.waitForLower(triggerPort, echoPort, distance, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
distance	Exclusive upper bound on value to wait for.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The value measured from the ultrasonic distance sensor that is lower than distance as a number between 0 and 255.

Description
Suspends execution of the script until the value read from the ultrasonic distance sensor is lower than distance. The value read from the ultrasonic distance sensor is returned when the script resumes execution.

get

```
Ping.get(triggerPort, echoPort, [ezbIndex])
```

Parameters

triggerPort	Trigger port used by the ultrasonic distance sensor.
echoPort	Echo port used by the ultrasonic distance sensor.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The value measured from the ultrasonic distance sensor as a number between 0 and 255.

Description
Pings the ultrasonic distance sensor once and returns the value measured.

PWM

get

```
PWM.get(port, [ezbIndex])
```

Parameters

port	Port to get PWM duty cycle from.
ezbIndex (optional)	Board index of the EZB to get PWM duty cycle from.

Returns
The PWM duty cycle percentage of the specified port as a value between 0 and 100.

Description
Returns the PWM duty cycle percentage of the specified port. Duty cycle percentage is returned as a value between 0 and 100.

set

```
PWM.set(port, dutyCycle, [ezbIndex])
```

Parameters

port	Port to use.
dutyCycle	Duty cycle percentage to set PWM duty cycle to.
ezbIndex (optional)	Board index of the EZB to get PWM duty cycle from.

Returns
Nothing

Description
Sets the PWM duty cycle percentage of the specified port to dutyCycle which is a value between 0 and 100.

setRandom

```
PWM.setRandom(port, lowCycle, highCycle, [ezbIndex])
```

Parameters

port	Port to use.
lowCycle	Inclusive lower bound on duty cycle percentage to set PWM duty cycle to.
highCycle	Exclusive upper bound on duty cycle percentage to set PWM duty cycle to.
ezbIndex (optional)	Board index of the EZB to get PWM duty cycle from.

Returns
The random value that the PWM was set to.

Description
Sets the PWM duty cycle percentage of the specified port to a value randomly chosen between lowCycle (inclusive) and highCycle (exclusive). lowCycle and highCycle are values between 0 and 100.

Servo

getPosition

```
Servo.getPosition(port, [ezbIndex])
```

Parameters

port	Servo port to use.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The commanded position of the servo at the specified port as a value between 1 and 180.

Description
Returns the position of the servo at the specified port as a value between 1 and 180. This function only returns the position the servo was commanded to move to, not the actual position of the servo. This does not query a bi-directional servo to get the position. It only returns the last position the servo was moved to.
If you wish to get the servo position of a smart servo, use the getPositionRealtime() command

getPositionRealtime

```
Servo.getPositionRealtime(port, [ezbIndex])
```

Parameters

port	Servo port to use.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The actual position of the servo at the specified port as a value between 1 and 180.

Description
Returns the position of the servo at the specified port. This function returns the physical position of the servo returned by the servo. This function was meant to work with servos which accept bi-directional communication (e.g. Dynamixel, LewanSoul).

- If this function is called on a pwm servo or a servo that does not support bi-direction communication, the last set servo position will be returned
- If the servo being queried has a communication issue, the last set position will be returned

getSpeed

```
Servo.getSpeed(port, [ezbIndex])
```

Parameters

port	Servo port to get speed from.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The speed that the specified port is set to as a value

Description

Returns the speed that the specified port is set to as a value

increment

```
Servo.increment(port, count, [ezbIndex])
```

Parameters

port	Servo port to use.
count	Degrees to increment by.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Increments the position of the servo at the specified port by count. Servo position is between 0 and 180.

release

```
Servo.release(port, [ezbIndex])
```

Parameters

port	Servo port to release.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Stops the servo at the specified port from holding its position.

releaseAll

```
Servo.releaseAll([ezbIndex])
```

Parameters

ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Stops all servos at all ports from holding their position.
*Warning: this will affect all ports and reset their state. That means if you using some ports for digital i/o or UART, those ports will be reset. This resets ALL ports, not some. If you wish to only release servos on specific ports, create a script using Servo.release(port) instead of this.
i.e.

```
Servo.release(d0);
Servo.release(d1);
Servo.release(d3);
```

decrement

```
Servo.decrement(port, count, [ezbIndex])
```

Parameters

port	Servo port to use.
count	Degrees to decrement by.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Decrements the position of the servo at the specified port by count. Servo position is between 0 and 180.

setMaxPositionLimit

```
Servo.setMaxPositionLimit(port, position, [ezbIndex])
```

Parameters

port	Servo port to set max position limit of.
position	Max position limit.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Sets the global maximum position that the servo at the specified port can move to the position. The position is a value between 0 and 180 (or the global max servo position if overridden).
While robot skills may have their own Min/Max servo options, this sets the global value. This is used for safety to ensure servos don't move outside the specified range.

setMinPositionLimit

```
Servo.setMinPositionLimit(port, position, [ezbIndex])
```

Parameters

port	Servo port to set min position limit of.
position	Min position limit.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Sets the global minimum position that the servo at the specified port can move to the position. The position is a value between 0 and 180 (or the global max servo position if overridden).
While robot skills may have their own Min/Max servo options, this sets the global value. This is used for safety to ensure servos don't move outside the specified range.

setPositionRandom

```
Servo.setPositionRandom(port, lowPosition, highPosition, [ezbIndex])
```

Parameters

port	Servo port to set position of.
lowPosition	Inclusive lower bound on random position value.
highPosition	Exclusive upper bound on random position value.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The random position that the servo was commanded to move to.

Description
Commands the servo to move to a random position between lowPosition (inclusive) and highPosition (exclusive). lowPosition and highPositionis are values between 1 and 180.

setPosition

```
Servo.setPosition(port, position, [ezbIndex])
```

Parameters

port	Servo port to set position of.
position	Position to move the servo to.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Commands the servo to move to position. position is a value between 1 and 180.

setSpeed

```
Servo.setSpeed(port, speed, [ezbIndex])
```

Parameters

port	Servo port to use.
speed	Speed to set the servo port to.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Sets the speed of the servo port to speed. speed is a value between 0 (fastest) and 10 (slowest).
This tutorial explains how to re-initialize servo speeds on startup: Setting Servo speeds and Initialization Script Tutorial - Tutorials - Community - Synthiam

waitForMove

```
Servo.waitForMove(port, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to wait for movement.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The position of the servo at the specified port that has changed. Returns -1 if timeout

Description
Suspends execution of the script until the commanded position of the servo at the specified port changes. The commanded position can be changed either by the user from another skill, or another script.

waitForPositionEquals

```
Servo.waitForPositionEquals(port, position, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to use.
position	Position to wait for.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
Returns current servo position. Returns -1 if timeout

Description
Suspends execution of the script until the commanded position of the servo at the specified port is equal to position. The commanded position can be changed either by the user from another skill, or another script.

waitForPositionHigher

```
Servo.waitForPositionHigher(port, position, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to use.
position	Exclusive lower bound on the position to wait for.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The position of the servo at the specified port that is higher than position. Returns -1 if timeout.

Description
Suspends execution of the script until the commanded position of the servo at the specified port is higher than position. The commanded position can be changed either by the user from another skill, or another script.

waitForPositionLower

```
Servo.waitForPositionLower(port, position, [ezbIndex], [timeoutMS])
```

Parameters

port	Servo port to use.
position	Exclusive upper bound on the position to wait for.
ezbIndex (optional)	Board index of the EZB to use.
timeoutMS (optional)	Number of milliseconds to wait before timeout.

Returns
The position of the servo at the specified port that is lower than position. Returns -1 if timeout

Description
Suspends execution of the script until the commanded position of the servo at the specified port is lower than position. The commanded position can be changed either by the user from another skill, or another script.

setAcceleration

```
setAcceleration(port, value, ezb index);
```

Parameters

port	The servo port
value	The Acceleration value
ezb index (optional)	The ezb index

Description
Set the acceleration of a servo. The servo and servo driver must support this feature. For example, check the Dynamixel robot skill because some of their servos support the Acceleration feature.

Example

```
// Set the Acceleration of the servo v1 to 10. By default, this will use EZB #0 because no
ezb is specified
Servo.setAcceleration(v1, 10)
```

```
// Specify the Acceleration of servo V1 on ezb index #3
Servo.setAcceleration(v1, 10, 3)
```

setVelocity

```
setVelocity(port, velocity, ezb index);
```

Parameters

port	The servo port
velocity	The velocity value
ezb index (optional)	The ezb index

Description
Set the velocity of a servo. The servo and servo driver must support this feature. For example, check the Dynamixel robot skill because some of their servos support the Velocity feature.

Example

```
// Set the velocity of the servo v1 to 10. By default, this will use EZB #0 because no ezb is specified
Servo.setVelocity(v1, 10)
```

```
// Specify the velocity of servo V1 on ezb index #3
Servo.setVelocity(v1, 10, 3)
```

IsReleased

```
IsReleased(port, ezbIndex)
```

Parameters

port	The servo port
ezbIndex (optional)	The ezb index

Returns
boolean (true of false)

Description
Returns the released state of the servo (torque off or on)

Example

```
if (IsReleased(d0))
  print("Servo d0 is released")
```

setFineTuneOffset

```
setFineTuneOffset(port, offset, ezbIndex)
```

Parameters

port - Servo port (i.e. d0	v2
offset - Value of the offset. Can be a positive or negative value	
ezbIndex - The ezb to apply this offset value (optional)	

Description
Sets the global fine-tune offset in positions of the specified servo. For example, if you set the offset to +10, every servo position set by other robot skills or script commands will add 10 positions. If you query the servo position, it'll return 10, not 20.
One of the uses for this could be used to maintain servos upright from an IMU, such as in this post.

Example

```
// Set the servo D0 to an offset of +20 positions
setFineTuneOffset(D0, 20);
```

```
// Set the servo D0 to an offset of -15 positions
setFineTuneOffset(D0, -15);
```

getFineTuneOffset

```
getFineTuneOffset(port, ezbIndex)
```

Parameters

port - Servo port (i.e. d0	v2
ezbIndex - The ezb to apply this offset value (optional)	

Description
Gets the global fine-tuned offset in positions of the specified servo. The value is set by either loading a fine tune servo profile or calling setServoFileTune()
One of the uses for this could be used to maintain servos upright from an IMU, such as in this post.

Example

```
// Get the servo D0 offset and print it to the output
print("The offset of D0 is " + getFineTuneOffset(D0));
```

getAcceleration

```
Servo.getAcceleration (port, [ezbIndex])
```

Parameters

port	Servo port to get Acceleration from.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The Acceleration that the specified port is set to as a value

Description
Returns the Acceleration that the specified port is set to as a value

getVelocity

```
Servo.getVelocity(port, [ezbIndex])
```

Parameters

port	Servo port to get Velocity from.
ezbIndex (optional)	Board index of the EZB to use.

Returns
The Velocity that the specified port is set to as a value

Description
Returns the Velocity that the specified port is set to as a value

UART

sendSerialString

```
UART.sendSerialString(port, baud, str, [ezbIndex])
```

Parameters

port	Digital port to send data over.
baud	Baud rate to send data at.
str	String to send.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Sends a string of data over the specified digital port with a baud rate of baud.

hardwareUartAvailable

```
UART.hardwareUartAvailable(uartIndex, [ezbIndex])
```

Parameters

uartIndex	Index of UART to check.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Number of bytes available in the UART receive buffer at the specified UART.

Description
Returns the number of bytes available in the UART receive buffer at the specified UART given by uartIndex.

hardwareUartRead

```
UART.hardwareUartRead(uartIndex, bytesToRead, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
bytesToRead	Number of bytes to read from the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns
A byte array of length bytesToRead containing data received from the specified UART.

Description
Returns a byte array of length bytesToRead containing data received from the specified UART.

hardwareUartReadString

```
UART.hardwareUartReadString(uartIndex, bytesToRead, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
bytesToRead	Number of bytes to read from the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns
A string of length bytesToRead containing data received from the specified UART.

Description
Returns a string of length bytesToRead containing data received from the specified UART.

hardwareUartReadStringAvailable

```
UART.hardwareUartReadStringAvailable(uartIndex, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
ezbIndex (optional)	Board index of the EZB to use.

Returns
A string containing all available data received from the specified UART.

Description
Returns a string containing all available data received from the specified UART.

hardwareUartWrite

```
UART.hardwareUartWrite(uartIndex, byte/byteArray, [ezbIndex])
```

Parameters

uartIndex	Index of UART to write to.
byte/byteArray	Byte or byte array to write to the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Writes byte to the specified UART. A byte array can instead be provided to write to the specified UART.

hardwareUartWriteString

```
UART.hardwareUartWriteString(uartIndex, str, [ezbIndex])
```

Parameters

uartIndex	Index of UART to write to.
str	String to write to the UART.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description

Writes string given by str to the specified UART.

initHardwareUart

```
UART.initHardwareUart(uartIndex, baud, [ezbIndex])
```

Parameters

uartIndex	Index of UART to initialize.
baud	Baud rate to set the UART to.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Initializes the UART with the specified baud rate. The UART will stay initialized until the EZB is power cycled.

sendSerial

```
UART.sendSerial(port, baud, byte/byteArray, [ezbIndex])
```

Parameters

port	Digital port to send data over.
baud	Baud rate to send data at.
byte/byteArray	Byte or byte array to send.
ezbIndex (optional)	Board index of the EZB to use.

Returns
Nothing

Description
Sends a byte or a byte array of data over the specified digital port with a baud rate of baud.

hardwareUartReadAvailable

```
UART.hardwareUartReadAvailable(uartIndex, [ezbIndex])
```

Parameters

uartIndex	Index of UART to read from.
ezbIndex (optional)	Board index of the EZB to use.

Returns
A byte array containing all available data received from the specified UART.

Description
Returns a byte array containing all available data received from the specified UART.

Utility

closeControl

```
Utility.closeControl()
```

Returns
Nothing

Description
Used for mobile devices and the Interface Builder only, this function will close the current control, the same as pressing the BACK button on your device.

defineGlobalVariable

```
Utility.defineGlobalVariable(globalVariableName, size, [defaultValue])
```

Parameters

globalVariableName	Name of the global variable to define as a string. Must start with '$'
size	Size of the array to create.
defaultValue (optional)	Default value to fill the array.

Returns
Nothing

Description
Creates a global variable array with name globalVariableName and size size. If defaultValue is provided the array will be filled with defaultValue. Otherwise it will be filled with empty strings.

exec

```
Utility.exec(filename, [arguments])
```

Parameters

filename	File path of the executable file to execute.
arguments (optional)	Optional arguments to pass to the executable file as a string.

Returns
Nothing

Description
Execute the executable file at file path filename with optional arguments arguments.

fillGlobalArray

```
Utility.fillGlobalArray(globalVariableName, defaultValue)
```

Parameters

globalVariableName	Name of the global array as a string.
defaultValue	Value to fill the array with.

Returns
Nothing

Description
Fills the global array with name globalVariableName with the value defaultValue.

getBit

```
Utility.getBit(value, bitPosition)
```

Parameters

value	Number to get bit from.
bitPosition	Position to get bit from.

Returns
The bit (1 or 0) at position bitPosition in the number value.

Description
Returns the bit at position bitPosition in the number value. Position is 0 indexed starting from the least significant bit.

getGlobalArraySize

```
Utility.getGlobalArraySize(globalVariableName)
```

Parameters

globalVariableName	Name of the global array as a string.

Returns
Returns The size of the global array.

Description
Returns the size of the global array with the name globalVariableName.

clearGlobalVariable

```
Utility.clearGlobalVariable(globalVariableName)
```

Parameters

globalVariableName	Name of the global variable to clear as a string.

Returns
Nothing

Description
Deletes the specified global variable.

getRandom

```
Utility.getRandom(min, max)
```

Parameters

min	Inclusive lower bound on the random integer.
max	Exclusive upper bound on the random integer.

Returns
Random integer between min (inclusive) and max (exclusive).

Description
Returns a random integer between min (inclusive) and max (exclusive).

loadProject

```
Utility.loadProject(filename)
```

Parameters

filename	File path of project to load.

Returns
Nothing

Description
Loads the project at file path filename.

map

```
Utility.map(input, inputMin, inputMax, outputMin, outputMax)
```

Parameters

input	Value to map.
inputMin	Minimum value the input can be.
inputMax	Maximum value the input can be.
outputMin	Minimum value the output can be.
outputMax	Maximum value the output can be.

Returns
input mapped onto the provided range of outputs.

Description
Maps input onto the provided range of outputs. (e.g. if input is halfway between inputMin and inputMax, then this function will return the value halfway between ouputMin and outputMax).

The formula for calculating the return value is shown below.

setBits

```
Utility.setBits(b7, b6, b5, b4, b3, b2, b1, b0)
```

Parameters

b7	The bit in the 8th position (most significant bit).
b6	The bit in the 7th position.
b5	The bit in the 6th position.
b4	The bit in the 5th position.
b3	The bit in the 4th position.
b2	The bit in the 3rd position.
b1	The bit in the 2nd position.

b0	The bit in the 1st position (least significant bit).

Returns
The number in decimal that is equal to the binary number (b7 b6 b5 b4 b3 b2 b1 b0).

Description
Converts the bits given by b7, b6, b5, b4, b3, b2, b1, b0 into decimal and returns the number.

showControl

```
Utility.showControl(controlName)
```

Parameters

controlName	Name of the control to show.

Returns
Nothing

Description
Used for mobile devices and the Interface Builder only, this function will open the control with name controlName into the foreground.

showDesktop

```
Utility.showDesktop(desktopNumber)
```

Parameters

desktopNumber	Virtual desktop to show.

Returns
Nothing

Description
Switches to virtual desktop desktopNumber. desktopNumber can be either 1, 2 or 3.

sleepPCHibernate

```
Utility.sleepPCHibernate(force, wake)
```

Parameters

force	Boolean to force the computer to hibernate or not. If true other applications have no say in the decision.
wake	Boolean to allow the computer will wake up on Wake events or not.

Returns
Nothing

Description
Puts the computer in hibernation mode. If force is true, other applications will have no say in the decision to hibernate. If wake is true, the computer will be allowed to wake up on Wake events.

getRandomUnique

```
Utility.getRandomUnique(min, max)
```

Parameters

min	Inclusive lower bound on the random integer.
max	Exclusive upper bound on the random integer.

Returns
Random integer between min (inclusive) and max (exclusive) and unique from the last returned integer.

Description
Returns a random integer between min (inclusive) and max (exclusive). This function will never return the same integer twice.

checkForUpdate

```
Utility.checkForUpdate()
```

Returns
True if an ARC update is available. False otherwise.

Description
Checks if an ARC update is available.

browser

```
Utility.browser(url)
```

Parameters

url	URL to open.

Returns
Nothing

Description
Opens the provided URL in the computer's default browser.

sleepPCSuspend

```
Utility.sleepPCSuspend(force, wake)
```

Parameters

force	Boolean to force the computer to sleep or not. If true other applications have no say in the decision.
wake	Boolean to allow the computer will wake up on Wake events or not.

Returns
Nothing

Description
Puts the computer in sleep mode. If force is true, other applications will have no say in the decision to sleep. If wake is true, the computer will be allowed to wake up on Wake events.

waitUntilTime

```
Utility.waitUntilTime(hour, minute)
```

Parameters

hour	Hour to wait until in 24 hour format.
minute	Minute to wait until.

Returns
Nothing

Description
Suspends execution of the script until the time (from the computer's clock) reaches the specified time.

shutdownPC

```
shutdownPC();
```

Description
Shuts down the PC. Does not prompt to save the project. Forces windows to shut down all applications.

Navigation

updateLocation

```
Navigation.updateLocation(x, y, degreesHeading, [confidence]);
```

Parameters

x	Cartesian X (cm) coordinate of the robot
y	Cartesian Y (cm) coordinate of the robot
degreesHeading	Direction in degrees (0-359) that the robot is facing relative to the starting position
confidence	[optional] the confidence (0-255) of accuracy of the coordinate location

Description

Send the cartesian coordinate of the robot to the NMS (navigation messaging system).

Example

```
// Send to the NMS that the robot is moving forward 30 cm over 30 seconds

for x in range(0, 3):
  print("Position X:" + x);
  Navigation.updateLocation(x, y, degreesHeading);
  sleep(1000);
```

updateScan

```
updateScan(distance, confidence, degree)
```

Parameters

distance	the distance in CM of the detected obstacle
confidence	the confidence (0-255) of the detected obstacle
degree	the degree (0-359) of the obstacle

Description

Programmatically send the detected obstacle distance based on the current cartesian coordinate of the robot to the NMS (navigation messaging system).

SetNavigationStatusToNavigating

```
SetNavigationStatusToNavigating()
```

Description

Instruct the navigator in NMS Level #1 to continue navigating if previously paused. The variable $NavigationStatus will also update to Navigating.
*Note: the NMS Level #1 robot skill must support this function. Verify with its manual that this feature is supported. There should also be an option in the respective robot skill for support pausing with close distances.
Example NMS Level #1 controls are:

- EZ-Slam
- The Navigator

- The Better Navigator
- Other navigation robot skills

Example

```
// Pause navigating if the ping sensor detects an object less than 100 distance and continue
if greater
if (Ping.get(d0, d0) < 100)
  Navigation.SetNavigationStatusToPause();
else
  Navigation.SetNavigationStatusToNavigating();
```

SetNavigationStatusToPause

```
SetNavigationStatusToPause()
```

Description

Instruct the navigator in NMS Level #1 to pause navigating. The variable $NavigationStatus will also update to Paused.
*Note: the NMS Level #1 robot skill must support this function. Verify with its manual that this feature is supported. There should also be an option in the respective robot skill for support pausing with close distances.
Example NMS Level #1 controls are:

- EZ-Slam
- The Navigator
- The Better Navigator
- Other navigation robot skills

Example

```
// Pause navigating if the ping sensor detects an object less than 100 distance and continue
if greater
if (Ping.get(d0, d0) < 100)
  Navigation.SetNavigationStatusToPause();
else
  Navigation.SetNavigationStatusToNavigating();
```

SetNavigationStatusToStopCancel

```
SetNavigationStatusToStopCancel()
```

Description

Instruct the navigator in NMS Level #1 to stop/cancel navigating. The variable $NavigationStatus will also update to StoppedCancelled.
*Note: the NMS Level #1 robot skill must support this function. Verify with its manual that this feature is supported. There should also be an option in the respective robot skill for support pausing with close distances.
Example NMS Level #1 controls are:

- EZ-Slam
- The Navigator
- The Better Navigator
- Other navigation robot skills

Example

```
// Stop navigating if the ping sensor detects an object less than 100 distance
if (Ping.get(d0, d0) < 100)
  Navigation.SetNavigationStatusToStopCancel();
```

UI

waitForUserInput

```
UI.waitForUserInput(timeout, prompt)
```

Parameters

timeout	Time to wait in seconds before stopping waiting for their text input.
prompt	The text prompt to display to the user while waiting for their text input

Returns

The lowercase text entered by the user, or the word timeout.

Description

Suspends execution of the script and prompts the user to enter text or a timeout occurs after timeout seconds. If a timeout occurs, the method will return "timeout."
***Note: the returned string is in lowercase**

Example

```
// Wait for any input from the user
response = UI.waitForUserInput(30, "Type anything to me");

print("You entered: " + response);
```