Arobito Documentation

Release 0.0

The Arobito Project

October 05, 2014

Contents

Welcome to the Project Arobito Documentation. This is your primary source of information for everything about the Arobito Project.

- Project Website: arobito.github.io (still empty)
- Development Website: github.com/arobito
- Travis CI Code: travis-ci.org/arobito/arobito
- Travis CI Docs: travis-ci.org/arobito/arobito-docs
- Documentation: arobito-docs.rtfd.org
- Twitter Account: @arobito_project
- IRC Channel #arobito on chat.freenode.net

CHAPTER 1

Table of Contents

1.1 Repository Overview

The Arobito Project uses a few repository to keep code, documentation and other things organized. This is an overview over our active repositories. If you want to participate, just fork one (or all) of the repositories, and send us pull requests. Need a kick start? Check our *Welcome new User* quick start guide and start contributing.

All repository name start with “arobito”, you should be able to fork us whatever repository names are already under your user account.

1.1.1 arobito/arobito

The arobito/arobito repository hosts the sources for the “heart” of the Arobito. It contains all the stuff that is about to be run on the Raspberry Pi main board and is mainly written in Python. You might want to check the documentation page “*Coding for Arobito*” for details.

1.1.2 arobito/arobito-mcu

The arobito/arobito-mcu repository contains the sources for the micro controller programming for the Arobito. It also contains some experiments (in the `experiments` folder) with Arduinos and some parts and backups our research and development efforts in this field. Want to know more? Check the *Arobito MCU development page* for details.

1.1.3 arobito/arobito-docs

The arobito/arobito-docs repository is the place for the project documentation. The page you are currently reading is from there. We use Sphinx with reStructured Text files for documenting virtually anything, not only the source code of the Python projects. Our documentation gets automatically built and published by the cool guys of ReadTheDocs.org, so that you can find the most current version always at arobito-docs.rtfd.org. At this time, we’re doing all documentation in English, even if English is not our native language. We’re sure that this lowers the barrier for other developers to join us.

1.1.4 arobito/arobito.github.io

The repository at arobito/arobito.github.io contains the sources of our (still empty) project website, hosted by GitHub.com under arobito.github.io. Initially, the page was created with the GitHub Page Creator, but in the future, we should build our own thing.

1.1.5 arobito/arobito-assets

The arobito/arobito-assets repository contains all the stuff that does not match somewhere else, for example the sources of the Arobito logo, the Arobito LaTeX document class and other things.

1.2 Coding for Arobito

1.2.1 Development Environments and Systems

We're not limited to a special operating system. Of course, we're recommending a free and open source OS (like Debian or SuSE Linux).

The same goes for IDEs. There is no rule that says one must use this or that IDE. For Python development, we're recommending PyCharm by JetBrains, which is available as Community Edition for free.

1.2.2 Programming Languages

For the Raspberry Pi backend driven by Arch Linux we prefer Python 3.4. It is the primary project language.

For the web frontend we're using XHTML5, JavaScript and CSS3. The backend is again driven by Python, powered by the famous CherryPy web framework.

The Arduino code is written in C. We're programming the Atmel micro controller directly over the ICSP pinout to save the space for the boot loader.

1.2.3 Coding Conventions

There is not yet a document that exactly describes how to code. The PyCharm IDE knows a lot about coding conventions and code formatting, so we take the hints there for real.

1.2.4 License Headers

One thing is a must: Every source code file needs a header, formatted as a comment. It has to be the first thing in every file (except a shebang line). The block tells the viewer about the license of the code.

```
Copyright {year} The Arobito Project

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

Additionaly, all Python files need the following license description:

```
__license__ = 'Apache License V2.0'
__copyright__ = 'Copyright {year} The Arobito Project'
```

1.2.5 Python Coding

Python files are encoded as UTF-8. We prefer Unix-style line endings.

At the very beginning of each file, there is the shebang line (when the file contains 'main' code) and the encoding:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

Then the license header follows. To make sure that the header is not parsed when creating the documentation, it must look like that:

```
# Copyright 2014 The Arobito Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
```

After the license header, the imports take place. Below the inputs and before the first line of code, the author of the file/module must be set:

```
__author__ = 'Max Musterman'
__credits__ = [ 'Max Mustermann' ]
```

With more than one author:

```
__author__ = 'Eva Musterfrau, Max Mustermann'
__credits__ = [ 'Eva Musterfrau', 'Max Mustermann' ]
```

It could be helpful to add a maintainer:

```
__maintainer__ = 'Max Mustermann'
```

Todo

Define how docstrings shall look like, how parameters are defined and how return types are marked.

1.2.6 Documentation

The project documentation is made with Sphinx using reStructuredText-Files. Additional papers are made with a to-be-created LaTeX class.

The documentation can be found in the `arobito/arobito-docs` repository.

1.3 Arobito MCU development

The following pages provide an overview over the micro controller development.

1.3.1 Experiments

Experiments are not part of the project code. But they backup our research and development efforts with Arduinos and some special parts, like sensors.

Ultrasonic distance measurement with HC-SR04

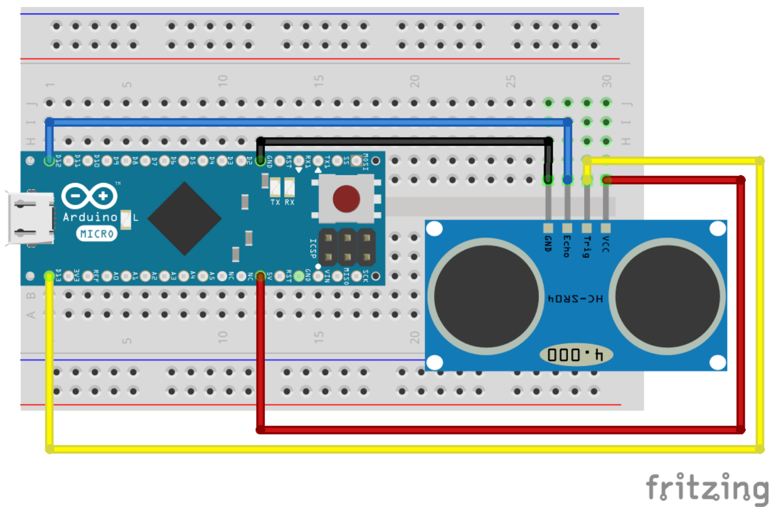
The HC-SR04 is a complete ultrasonic distance measurement device that is very simple to use and costs just about 3 EUR. The detection range as written in the data sheet is between 2 cm to 4 m, but experiments show that 3 cm to 2 m is more realistic.

Basic setup

The basic setup consists of only these parts:

- a half-length breadboard
- an Arduino Micro v3
- an HC-SR04 ultrasonic distance sensor
- a few wires

The HC-SR04 works simply by sending a pulse to the trigger input and waiting for a pulse from the echo output. The length of this pulse allows the calculation of the distance to the detected object. We selected the Arduino pin 13 for sending the trigger pulse and pin 12 for receiving the echo pulse.



The power is delivered over the USB connection of the Arduino.

Programming the Arduino

The code for this experiment can be found under arobito/arobito-mcu/experiments/usonic/Basic_Distance_Measuring as Atmel Studio Project.

The definitions:

```
#define TRIGGER_PIN 13
#define ECHO_PIN 12
#define USONIC_DIV 58.0
#define MEASURE_SAMPLE_DELAY 5
#define MEASURE_SAMPLES 25
#define MEASURE_DELAY 250
```

Trigger pin and echo pin are the two pins where the HC-SR04 is connected to. From the data sheet we know that we have to divide the measured microseconds by 58 to get centimeters. We delay our sample measurements by 5 µs, and take 25 samples to calculate the average. Between two measuring cycles the controller should wait for 1/4 second.

```
void setup()
{
   // Serial monitoring
   Serial.begin(9600);

   // Initializing Trigger Output and Echo Input
   pinMode(TRIGGER_PIN, OUTPUT);
   pinMode(ECHO_PIN, INPUT);

   // Reset the trigger pin and wait a half a second
   digitalWrite(TRIGGER_PIN, LOW);
   delayMicroseconds(500);
}
```

During setup, the serial (USB) port get initialized and the trigger and the echo pin get assigned. To make sure we have a defined starting point, the trigger pin is put to low explicitly. After waiting half a second, the loop may start.

The loop simply triggers the measurement:

```
void loop()
{
  delay(MEASURE_DELAY);
  long distance = measure();
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" mm");
}
```

The method `measure()` contains the sampling of single measurements:

```
long measure()
{
  long measureSum = 0;
  for (int i = 0; i < MEASURE_SAMPLES; i++)
  {
    delay(MEASURE_SAMPLE_DELAY);
    measureSum += singleMeasurement();
  }
  return measureSum / MEASURE_SAMPLES;
}
```

One single measurement sequence looks like the following:

```
long singleMeasurement()
{
  long duration = 0;
  // Measure: Put up Trigger...
  digitalWrite(TRIGGER_PIN, HIGH);
  // ... wait for 11 µs ...
  delayMicroseconds(11);
  // ... put the trigger down ...
  digitalWrite(TRIGGER_PIN, LOW);
  // ... and wait for the echo ...
  duration = pulseIn(ECHO_PIN, HIGH);
  return (long) (((float) duration / USONIC_DIV) * 10.0);
}
```

The programm's output looks like the following:

```
Distance: 90 mm
Distance: 90 mm
Distance: 89 mm
Distance: 92 mm
Distance: 90 mm
Distance: 197 mm
Distance: 207 mm
Distance: 1371 mm
Distance: 209 mm
Distance: 159 mm
Distance: 125 mm
Distance: 125 mm
Distance: 127 mm
Distance: 128 mm
Distance: 124 mm
Distance: 124 mm
Distance: 125 mm
Distance: 126 mm
```

Using a Matrix Keypad

There a several cheap 4x4 matrix keypads available that can be easily controlled with an Arduino MCU. Those keypads don't have own controllers. They are connected by their rows and columns. When putting one row to high, you can read at the column lines which of the buttons are pressed in this row. We need to cycle through the rows and read the columns to get all buttons that are pressed.

Basic setup

The basic setup consists of only these parts:

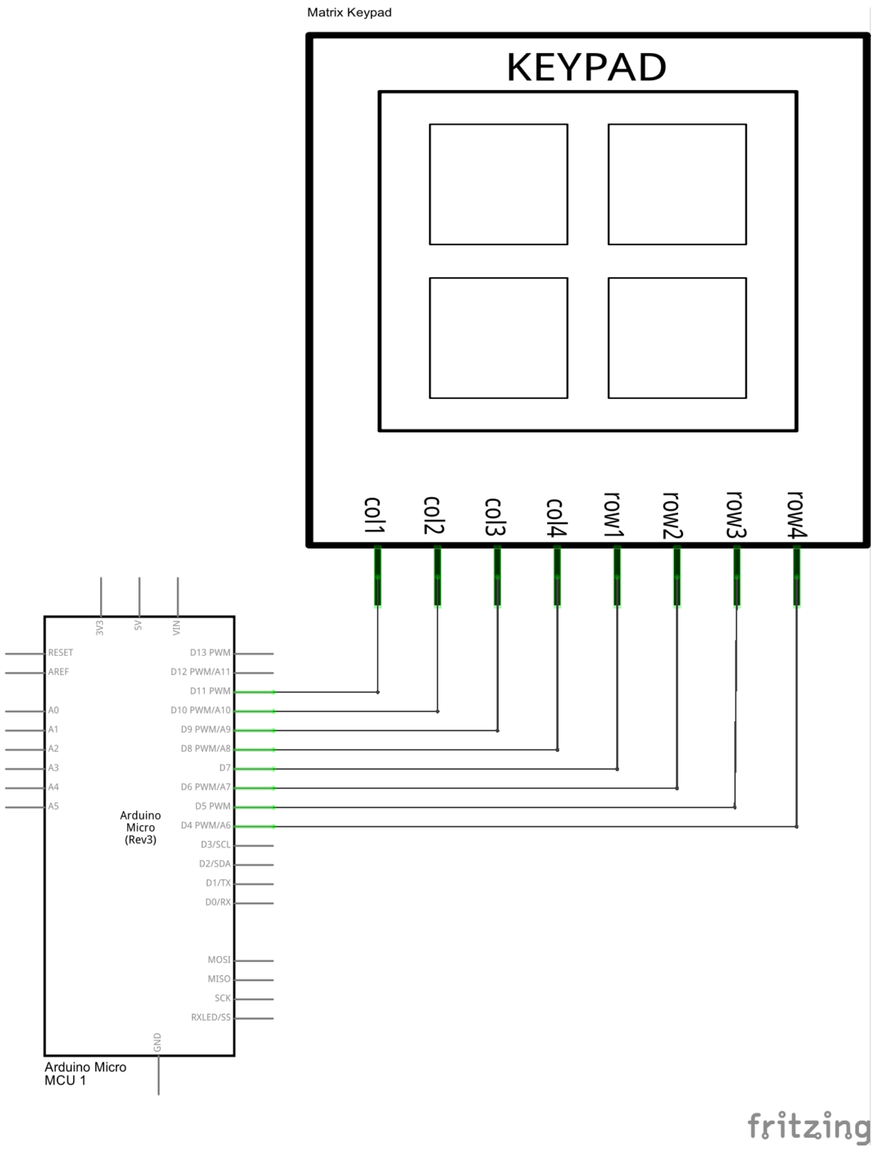
- a half-length breadboard
- an Arduino Micro v3
- a 4x4 membrane keypad
- a few wires

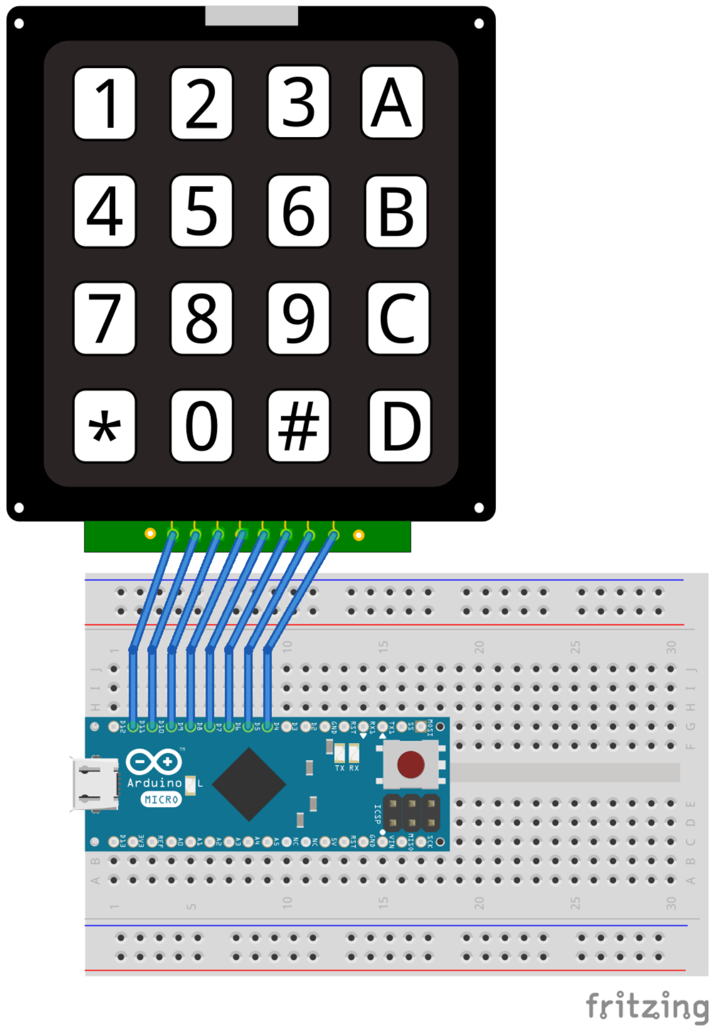
The circuit is very easy. We can connect the keypad directly to the Arduino. No extra parts are required.

For the experiments, we put everything together on a breadboad.

The power is delivered over the USB connection of the Arduino. The keypad does not need a dedicated power source.

Warning: The hardware keypad in our experiment has a different pin-out assignment as in some data sheets around the web. The Arduino program may not match the circuit.


Matrix Keypad
KEYPAD
col1
col2
col3
col4
row1
row2
row3
row4
3V3
5V
VIN
RESET
AREF
A0
A1
A2
A3
A4
A5
D13 PWM
D12 PWM/A11
D11 PWM
D10 PWM/A10
D9 PWM/A9
D8 PWM/A8
D7
D6 PWM/A7
D5 PWM
D4 PWM/A6
D3/SCL
D2/SDA
D1/TX
D0/RX
MOSI
MISO
SCK
RXLED/SS
GND
Arduino Micro (Rev3)
Arduino Micro MCU 1
fritzing


1
2
3
A
4
5
6
B
7
8
9
C
*
0
#
D
Arduino
MICRO
TX RX
ICSP
fritzing

```
PIN | Assignment
----+-----------
  1 | Column 4
  2 | Column 3
  3 | Column 2
  4 | Column 1
  5 | Row 4
  6 | Row 3
  7 | Row 2
  8 | Row 1
```

What's next?

Before we start programming, we need a strategy to debounce the inputs. This could be done with software in the MCU or hardware. We evaluate both possibilities.

1.4 Source code documentation

This chapter contains our source code documentation, separated into the API documentation and the Test Case documentation.

1.4.1 Arobito API Documentation

arobito package

arobito.Base module

This module contains some very basic stuff needed all over the whole project. It is kind of a base library.

class `arobito.Base.`**`SingletonMeta`**

Bases: `builtins.type`

Make a class a singleton.

Add `metaclass=SingletonMeta` to the class' options to make it a singleton.

`_SingletonMeta__instances` **= {}**

`arobito.Base.`**`create_salt`**(*length: int=128*) → str

Create a random string that works as a standard salt.

It selects `length` chars form letters, digest and punctuation chars excluding the percent sign. By today, 128 chars seems to be a sufficient salt length.

Parameters **length** – Length of the random string

Returns A random string

`arobito.Base.`**`create_simple_key`**(*length: int=64*) → str

Create a random key

This key can be used for creating session IDs. The default length is 64 to prevent collisions.

Parameters **length** – The length of the random string

Returns A random string

`arobito.Base.`**`find_root_path`**`()` → str

Find the application's root path in the file system

Returns The root path as string

`arobito.Base.`**`hash_password`**`(`*password: str*, *salt: str=None*, *rounds: int=1000*, *secret: str=None*`)` → str

Hash a given password

The hash algorithm used is SHA-512. The hash is returned as a string of hex numbers ('a1b2c3d4e5f6aa...'). If a salt is defined, it is added to the password/the hash on each round of hashing. The secret is added only once. To make it harder to compromise, there are a lot of rounds of re-hashing hash+salt. 1000 rounds is the default.

Attention: Using zero rounds is one hundred percent useless. You'll receive an exception. You might want to create a salt using the `create_salt()` function in this module.

Parameters

- **password** – The clear text password
- **salt** – The salt (a random string)
- **rounds** – The number of rounds of hashing
- **secret** – The application secret

Returns A string of hex numbers

arobito.FsTools module

This module contains functionality to work with the local file system.

We need those functions to use configuration files and find out about their location.

`arobito.FsTools.`**`get_config_file`**`(`*filename: str='arobito.ini'*`)` → str

Find the location of a configuration file

Parameters **filename** – The name of the file to find

Returns The location of the file

Raises IOError when no file can be found and created

`arobito.FsTools.`**`get_config_folder`**`()` → str

Find a folder for the configuration files

The function first tries

- to find the environment variable `AROBITO_CONF`.
- /etc/arobito
- ~/.arobito
- the current directory

Returns The folder

Raises IOError when no location is sufficient.

arobito.Helper module

This module provides some basic helper functions needed globally.

`arobito.Helper.`**`shutdown`**`(`*code: int=0*, *delay: int=0*`)` → None

Shutdown the CherryPy engine and exit the program.

The shutdown can be delayed.

Parameters

- **code** – Return code
- **delay** – Delay in seconds

arobito.controlinterface package

arobito.controlinterface.BackendManager module

This module contains the base structure of the CherryPy based web interface. It configures CherryPy and mounts some basic applications.

Additionally, it provides an User and a Session Manager to handle login requests, user privileges and associated sessions.

class `arobito.controlinterface.BackendManager.`**`SessionManager`**

Bases: `builtins.object`

This class, a singleton, manages the sessions.

Loads session defaults from the 'controller.ini' configuration file.

`cleanup``()` → None

Clean left-over and old sessions from the SessionManager

`get_current_sessions``()` → int

Get the amount of active sessions.

It calls the `cleanup()` method first to throw old sessions away.

Returns The count of active sessions.

`get_user``(`*session: str*`)` → dict

Get the user data dict by session ID

It calls the `cleanup()` method first to throw old sessions away.

Parameters **session** – The session ID to look up

Returns The dict or None if not there or already invalid.

`login``(`*username: str*, *password: str*`)` → str

Log a user in and create a session key on success.

It calls the `cleanup()` method first to throw old sessions away.

Parameters

- **username** – The username
- **password** – The password

Returns A session key or None on login failed

logout(*session: str*) → None

Log a user out

Parameters **session** – The session to log out

class `arobito.controlinterface.BackendManager.`**UserManager**

Bases: `builtins.object`

This class manages the users. It is used to grant access.

This is a singleton.

Load the user basic configuration from the users.ini file.

Raises IOError When there can be no users.ini can be loaded or created.

get_user_by_username_and_password(*username: str*, *password: str*) → dict

Try to find a user and check the password. If a match is found, create a dict that contains the username, the userlevel, the timestamp of login and the timestamp of the last access.

Parameters

- **username** – The username
- **password** – The password

Returns The described dict or None on invalid credentials or inactive user account.

username_regex = re.compile('^[a-zA-Z0-9]{1,64}$')

arobito.controlinterface.ControlInterface module

Parts of the web application

class `arobito.controlinterface.ControlInterface.`**ArobitoControlInterface**(*bind_ip: str='0.0.0.0'*, *listen_port: int=9812*)

Bases: `builtins.object`

Main Class to control the Robi Web Based Interface

Configure a control server for Robi

Parameters

- **bind_ip** – The IP to bind to
- **listen_port** – The port to listen to

startup() → int

Start the Server

Returns The exit status code

class `arobito.controlinterface.ControlInterface.`**ArobitoControlInterfaceRedirect**

Bases: `builtins.object`

On the web interface root, make a simple redirect to the static index page.

index() → None

Do only the redirect to `/static/index.html`.

class `arobito.controlinterface.ControlInterface.`**`ArobitoControlInterfaceStatics`**

Bases: `builtins.object`

This class delivers static content over the web interface, e.g. JavaScript, HTML files, CSS files and images.

Look in the config file 'controller.ini' for the folder with the static contents to serve.

`default`(**args*) → bytes

Serve static content directly out of the package

`default_mime_type` **= 'application/octet-stream'**

when no mime type could be identified

`mime_extract_regex` **= re.compile('\\.(?P<attr>[^\\.]+?)$', re.IGNORECASE)**

find the mime type by file ending

`mime_types` **= {'js': 'text/javascript', 'jpeg': 'image/jpeg', 'png': 'image/png', 'jpg': 'image/jpeg', 'gif': 'image/gif', 'ht**

supported mime types by file ending

arobito.controlinterface.ControllerBackend module

This module implements the methods for the main controller app. By decoupling them from the CherryPy exposed methods, they are better to test without a mock.

class `arobito.controlinterface.ControllerBackend.`**`App`**

Bases: `builtins.object`

This is the backend class for `ControllerFrontend.App`.

Initialize the main App with the `SessionManager` instance.

`auth`(*json_req: dict*) → dict

Backend method for `ControllerFrontend.App.auth`

Parameters **json_req** – The JSON request dict

Returns Response as dictionary

`auth_default_response` **= {'auth': {'reason': 'User unknown or password wrong', 'status': 'failed', 'success': False**

The default response when authorization fails.

`get_session_count`(*json_req: dict*) → dict

Backend method for `ControllerFrontend.App.get_session_count`

Parameters **json_req** – The JSON request dict

Returns Response as dictionary

`logout`(*json_req: dict*) → dict

Backend method for `ControllerFrontend.App.logout`

Parameters **json_req** – The JSON request dict

Returns Response as dictionary

`shutdown`(*json_req: dict*) → dict

Backend method for `ControllerFrontend.App.shutdown`

Parameters **json_req** – The JSON request dict

Returns Response as dictionary

arobito.controlinterface.ControllerFrontend module

This module contains the mapping between the CherryPy methods and the `ControllerBackend` module.

class `arobito.controlinterface.ControllerFrontend.`**App**

Bases: `builtins.object`

The Arobito Controlling Application

Initialize the backend

auth() → dict

Answer to an authorization request.

The request is a JSON post request that must look like this:

```
{
  'username': 'The User Name',
  'password': 'The Password'
}
```

In case of a successful login, the response will look like this:

```
{
  'auth':
  {
    'success': true,
    'status': 'Login successful',
    'key': 'The Session Key'
  }
}
```

Use the session key for all later requests.

In case of a failed login, the default fail response will look like this:

```
{
  'auth':
  {
    'success': false,
    'status': 'failed',
    'reason': 'User unknown or password wrong'
  }
}
```

This method returns a dict. The dict is automatically converted to JSON by CherryPy.

This method refers to the backend method `ControllerBackend.App.auth`.

Returns The authorization response as a dict

get_session_count() → dict

Get the current session count.

If the logged in user is an `Administrator`, the count of currently active sessions is returned. The request must be a JSON post and looks like this:

```
{
  'key': 'The Session Key'
}
```

In case of success, the number is returned (1 in this example):

```
{
  'session_count': 1
}
```

If there are insufficient rights, -1 is returned as result:

```
{
  'session_count': -1
}
```

The dict returned by this method is converted to JSON by CherryPy.

This method refers to the backend method `ControllerBackend.App.get_session_count`.

Returns The response as dict

`logout``()` → dict

Perform a logout with the given session key.

A logout request must be a JSON post request that looks like the following:

```
{
  'key': 'The Session Key'
}
```

A logout request gets always a positive response:

```
{
  'logout': true
}
```

This method returns a dict that is converted to JSON by CherryPy.

This method refers to the backend method `ControllerBackend.App.logout`.

Returns The positive response as a dict

`shutdown``()` → dict

Initiate the shutdown for the controlling application.

This is only available to users of the level `Administrator`. To initiate the shutdown, the following JSON needs to be posted:

```
{
  'key': 'The Session Key'
}
```

The shutdown will be initiated within the next 10 seconds.

In case of success, the response looks like the following:

```
{
  'shutdown': true
}
```

The request fails on insufficient rights. The response is in this cases:

```
{
  'shutdown': false
}
```

The dict that is returned by this method is converted to JSON by CherryPy.

This method refers to the backend method `ControllerBackend.App.shutdown`.

Returns The response as dict

controlinterface module

Module contents

This is the main startup module for the Arobito Control Panel.

The panel itself is in charge of starting all other required services, like the connection to the micro controllers.

It all starts below the `if __name__ == '__main__':` line: First, parse all arguments and then activate the web based control interface on a CherryPy foundation.

`controlinterface.`**`control_interface`**(*bind_ip: str='0.0.0.0'*, *listen_port: int=9812*) → int

Launch the Arobito Control Interface - a web based remote control solution

Parameters

- **bind_ip** – IP address to bind the control interface to. Use '0.0.0.0' for all available IP addresses.
- **listen_port** – The port number for the control interface to listen. Default is 9812.

Returns A return code. 0 means 'everything is ok'

1.4.2 Arobito Webinterface JavaScript Documentation

This part of our documentation is created manually. At this time, Sphinx does not have features that allow an automatic generation of JavaScript API documentation. All functions/methods/classes documented here can be found in the index also.

robi-app.js

This is an overview over the `/src/web-static/robi-assets/robi-app.js` file contents. It contains the main stuff for the Arobito communication between the backend and the web frontend.

Namespace

The code in the `robi-app.js` creates the namespace `window.robi` (or just `robi` for short) to make it's functions public. It also has the internal namespace `local` which is not available outside the module's code.

Initialization

The module initialize itself on document load time. There is no additional loading necessary.

API documentation

Public Methods The following methods allow an easy access to the growl implementation used.

`robi.`**`error`**(*title*, *message*)

Create a error growl message (the little pop-up in the upper right window corner)

See also `robi.warn`, `robi.notice`, `robi.message`.

Arguments

- **title** (*string*) – The title of the pop-up
- **message** (*string*) – The message to display

`robi.warn(title, message)`
Create a warn growl message (the little pop-up in the upper right window corner)

See also `robi.error`, `robi.notice`, `robi.message`.

Arguments

- **title** (*string*) – The title of the pop-up
- **message** (*string*) – The message to display

`robi.notice(title, message)`
Create a notice growl message (the little pop-up in the upper right window corner)

See also `robi.error`, `robi.warn`, `robi.message`.

Arguments

- **title** (*string*) – The title of the pop-up
- **message** (*string*) – The message to display

`robi.message(title, message)`
Create a growl message (the little pop-up in the upper right window corner)

See also `robi.error`, `robi.warn`, `robi.notice`.

Arguments

- **title** (*string*) – The title of the pop-up
- **message** (*string*) – The message to display

The next methods are helper for opening specific pagelets.

`robi.statusPage()`
Open the StatusPage pagelet.

The next methods are used for login, logout and session handling.

`robi.login(form_element)`
Initiate a login with credentials from the given form jQuery element.

Arguments

- **form_element** (*FormElement*) – The form element from which to take the user credentials

`robi.logout()`
Initiate a logout.

`robi.shutdown()`
Send a shutdown request to the server. The user logged-in needs the permissions to initiate the shutdown.

`robi.getSessionCount()`
Query the server for the current count of logged-in sessions. The user needs the matching permissions to get an useful answer.

Private Methods

`local.`**`loadPagelet`**`(`*container*, *pagelet*`)`
Load a pagelet into the given container (identified by a jQuery selector)

Arguments

- **jQuerySelector** (*container*) – The container element to fill
- **pagelet** (*string*) – The pagelet to load - it must be an existing HTML file in the `/src/web-static/pagelets` folder.

`local.`**`login`**`(`*data*`)`
This method is used as the success callback for a user login.

Arguments

- **data** (*JSON*) – The JSON data from the server

`local.`**`logout`**`(`*data*`)`
This method is used as the success callback for a user logout.

Arguments

- **data** (*JSON*) – The JSON data from the server

`local.`**`shutdown`**`(`*data*`)`
This method is used as the success callback for a shutdown request.

Arguments

- **data** (*JSON*) – The JSON data from the server

`local.`**`getSessionCount`**`(`*data*`)`
This method is used as the success callback for a getSessionCount request.

Arguments

- **data** (*JSON*) – The JSON data from the server

`local.`**`shutdown`**`(`*data*`)`
This method is used as the success callback for a shutdown request.

Arguments

- **data** (*JSON*) – The JSON data from the server

`local.`**`initLoginForm`**`()`
Initialize the login form for the web application. This method binds the submit handler and setups the jQuery UI stuff for the form elements.

`local.`**`initMainToolbar`**`()`
Initialize the main toolbar, even if it's not visible right from the application startup.

`local.`**`init`**`()`
Initialize this JavaScript module: Set up AJAX parameters and the whole application.

1.4.3 Arobito Test Case Documentation

To run the tests, simply execute the `testrunner.py` script in the `test` folder.

testlibs package

testlibs.Lister module

This is the place to put functions that we need in many tests.

`testlibs.Lister.`**`__enlist_all_modules`**(*list_of_modules: list*, *package: str='arobito'*) → None
: List all modules within a given package and all subpackages.

Parameters

- **list_of_modules** – A list that is filled with the modules found
- **package** – The name of the package

`testlibs.Lister.`**`enlist_all_classes`**(*package: str='arobito'*) → list
: List all classes in the given package and all sub packages.

Parameters **package** – The package name

Returns List of all classes

`testlibs.Lister.`**`enlist_all_modules`**(*package: str='arobito'*) → list
: Return a list of all modules of the given package and all sub packages.

Parameters **package** – The package name

Returns The list of modules

testrunner module

Module contents

This module is our primary test runner.

It iterates through all modules under the package test and searches for unit test test cases and runs them. This allows adding tests simply by adding packages and classes. Only rule: All test case classes must reside in modules under the `tests` module.

`testrunner.`**`run_tests`**() → int
: Get all classes and run the tests.

Returns 0 when all tests succeeded, 1 otherwise.

`testrunner.`**`test_suite_setup`**() → None
: Setup the test runner: Make the Arobito source code accessible.

tests package

tests.Documentation module

The Documentation Module contains tests about the source code documentation.

class `tests.Documentation.`**`DefaultFields`**(*methodName='runTest'*)
: Bases: `unittest.case.TestCase`

This test case checks if every module has the required fields like

- `__license__`

- `__copyright__`
- `__author__`
- `__credits__`
- `__maintainer__`

This ensures a minimum of documentation.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_DefaultFields__check_author`(*mod: str*, *fields: list*) → None

Check if the __author__ field is set up correctly.

Parameters

- **mod** – Name of the module to check
- **fields** – List of fields in the module

`_DefaultFields__check_copyright`(*mod: str*, *fields: list*) → None

Check if the __copyright__ field contains our standard text.

Parameters

- **mod** – Name of the module to check
- **fields** – List of fields in the module

`_DefaultFields__check_credits`(*mod: str*, *fields: list*) → None

Check if the __credits__ field is set up correctly.

Parameters

- **mod** – Name of the module to check
- **fields** – List of fields in the module

`_DefaultFields__check_license`(*mod: str*, *fields: list*) → None

Check if the __license__ field contains our standard text.

Parameters

- **mod** – Name of the module to check
- **fields** – List of fields in the module

`_DefaultFields__check_maintainer`(*mod: str*, *fields: list*) → None

Check if the __maintainer__ field is set up correctly.

Parameters

- **mod** – Name of the module to check
- **fields** – List of fields in the module

`runTest`() → None

Go though all modules and launch the tests for every single one.

tests.arobito package

tests.arobito.Base module

class `tests.arobito.Base.`**`CreateSalt`**`(`*methodName='runTest'*`)`

Bases: `unittest.case.TestCase`

Test the `create_salt` function

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest``()` → None

Create some salts and check the size of the returning string. Check if the string contains the percent sign. Also, check the default length explicitly.

Also checks if the correct error is raised on bad arguments.

class `tests.arobito.Base.`**`CreateSimpleKey`**`(`*methodName='runTest'*`)`

Bases: `unittest.case.TestCase`

Test the `create_simple_key` function

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest``()` → None

A simple key contains only numbers and letters. One would miss punctuation chars when comparing it to the `create_salt` method.

The test checks the correct length and the correct char output. It also checks if the correct errors are raised when one tries to create keys with a zero or negative length.

class `tests.arobito.Base.`**`DifferentSingletons`**`(`*methodName='runTest'*`)`

Bases: `unittest.case.TestCase`

Find out if two different singletons are really independent

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest``()`

Try to create two different singletons and check if their value is different, getting and setting works on different objects and if the objects are different.

class `tests.arobito.Base.`**`FindRootPath`**`(`*methodName='runTest'*`)`

Bases: `unittest.case.TestCase`

Test the `find_root_path` function

This function is not easy to test. It detects the path from where this application runs. But when testing, there are quite different results. So the only thing we could test is, that this function delivers a valid path name.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_FindRootPath__validate_path``(`*path*`)` → None

`runTest``()` → None

Test that the function result is not None, is a path, that path is absolute and exists. It does a few calls to the function to make sure it always comes up with the same result.

class `tests.arobito.Base.`**`HashPassword`**(*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Test the `hash_password` function

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest`() → None

Test if the method behaves correctly when fed with illegal parameters. Then it checks if the return value looks like a hash. And last but not least try some hashes that have been built with the `sha512sum` command on a Linux machine.

class `tests.arobito.Base.`**`Singleton`**(*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Test the singleton meta class (`SingletonMeta`)

Many things within this project rely on a working singleton implementation. It's very important that this behaviour is checked in every test run, especially when the language (Python) version changes.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest`() → None

Try to create two instances of the singleton (the test uses the `SingletonMock1` class) and tests if values and instances are equal.

class `tests.arobito.Base.`**`SingletonMock1`**

Bases: `builtins.object`

A mock class that is meant to be handled as a singleton. It is used by the `Singleton` test case.

`get_value`() → str

Get the value

Returns The value

`set_value`(*value: str*) → None

Set the value

Parameters **value** – The value

class `tests.arobito.Base.`**`SingletonMock2`**

Bases: `builtins.object`

A mock class that is meant to be handled as a singleton. It is used by the `Singleton` test case. This is roughly the same thing as `SingletonMock1`, but it should be another Singleton instance.

`get_value`() → str

Get the value

Returns The value

`set_value`(*value: str*) → None

Set the value

Parameters **value** – The value

tests.arobito.FsTools module

class `tests.arobito.FsTools.`**`GetConfigFile`**(*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Test the `get_config_file` function

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_GetConfigFile__check_default`() → None
Check with defaults, and clean up afterwards

`_GetConfigFile__check_with_name`() → None
Check with a dedicated file name, and clean up afterwards

`runTest`() → None
For now, simply test if the files are created

class `tests.arobito.FsTools.`**`GetConfigFolder`**(*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Test the `get_config_folder` function

For this test, it is very important, that there are no env variables are set and no special folders are created.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_GetConfigFolder__check_default`() → None
Check the default folder returned, without additional configuration. It expects the current folder.

`runTest`() → None
This very basic test checks if the folder returned for the configuration files.

tests.arobito.controlinterface package

tests.arobito.controlinterface.BackendManager module

Tests for the `BackendManager` module.

class `tests.arobito.controlinterface.BackendManager.`**`SessionManagerIsSingleton`**(*methodName='runTes*
Bases: `unittest.case.TestCase`

Find out if the `SessionManager` is created as singleton.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest`() → None
Simply create two instances of `UserManager` and compare them.

class `tests.arobito.controlinterface.BackendManager.`**`SessionManagerMultiTest`**(*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Test the `SessionManager` class.

It works only with the standard settings of the `controller.ini` file.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_SessionManagerMultiTest__check_count`(*count: int*, *expected: int*) → None
Check if count matches the expected count (and that the type matches and it's not None)

Parameters

- **count** – The count to check

- **expected** – The expected count

runTest () → None
Simulate a complete workflow through the class' methods

class tests.arobito.controlinterface.BackendManager.**UserManagerGetUserByUsernameAndPassword** (*m*
Bases: unittest.case.TestCase

Test the method get_user_by_username_and_password from class UserManager.

It works only with the standard settings of the users.ini file.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

runTest () → None
Just try to get user data by the default settings.

class tests.arobito.controlinterface.BackendManager.**UserManagerIsSingleton** (*methodName='runTest'*)
Bases: unittest.case.TestCase

Find out if the UserManager is created as singleton.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

runTest () → None
Simply create two instances of UserManager and compare them.

tests.arobito.controlinterface.BackendManager.**evaluate_user_object** (*test: unittest.case.TestCase*, *user_object: dict*) → None

Helper function for evaluating an user object

Parameters

- **test** – The currently running test case
- **user_object** – The user object to evaluate

tests.arobito.controlinterface.ControllerBackend module

Tests for the ControllerBackend module.

class tests.arobito.controlinterface.ControllerBackend.**AppAuth** (*methodName='runTest'*)
Bases: unittest.case.TestCase

Test the App.auth method.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

_AppAuth__check_failed_response (*response: dict*) → None
Check if the response is a failing one

Parameters response – The response from the auth method call

_AppAuth__check_response_basics (*response: dict*) → None
Check basic response characteristics

Parameters response – The response from the auth method call

`_AppAuth__check_success_response`(*response: dict*) → None
Check if the response is a success one

Parameters **response** – The response from the auth method call

`runTest`() → None
Try to use the auth method with working credentials and with wrong credentials.

class `tests.arobito.controlinterface.ControllerBackend.`**`AppGetSessionCount`**(*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Test the `App.get_session_count` method.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_AppGetSessionCount__check_basic_response`(*response: dict*) → None
Check the basics of a response objects

Parameters **response** – The response from the get_session_count method

`_AppGetSessionCount__check_invalid_response`(*response: dict*) → None
Check an invalid response object

Parameters **response** – The object to check

`_AppGetSessionCount__check_valid_response`(*response: dict, expected: int*) → None
Check a valid response

Parameters

- **response** – The object to check
- **expected** – The session count expected

`runTest`() → None
Test the method with bad and valid input

class `tests.arobito.controlinterface.ControllerBackend.`**`AppLogout`**(*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Test the `App.logout` method.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`_AppLogout__check_response`(*response: dict*) → None
Verify the response object

Parameters **response** – The response object

`runTest`() → None
Test the logout method with real and wrong keys

class `tests.arobito.controlinterface.ControllerBackend.`**`AppShutdown`**(*methodName='runTest'*)
Bases: `unittest.case.TestCase`

Test the `App.shutdown` method.

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

`runTest`() → None
Testing the shutdown method would cancel the program, so we can only check the behaviour with bad input.

`tests.arobito.controlinterface.ControllerBackend.`**`create_app`**(*test: unittest.case.TestCase*) → arobito.controlinterface.ControllerBackend.App

Create an `App` instance

Parameters test – The currently running unit test case

Returns An instance of App

`tests.arobito.controlinterface.ControllerBackend.`**`get_valid_key`**(*test: unittest.case.TestCase*, *app: arobito.controlinterface.ControllerBackend.*) → str

Produce a valid key by using the arobito default credentials against the `App.auth` method

Parameters test – The currently running unit test case

Returns A valid key

1.5 Equipment

Our experiments are built to run on the following platform:

- Some Arduinos (Arduino Micro, Arduino Nano and Arduino Due)
- Raspberry Pi Model B first generation

1.6 License

The sources of the Arobito Project are released under the Apache License 2.0 license.

```
                              Apache License
                        Version 2.0, January 2004
                     http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.
   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this section) patent license to make, have made,
   use, offer to sell, sell, import, and otherwise transfer the Work,
   where such license applies only to those patent claims licensable
   by such Contributor that are necessarily infringed by their
   Contribution(s) alone or by combination of their Contribution(s)
   with the Work to which such Contribution(s) was submitted. If You
   institute patent litigation against any entity (including a
   cross-claim or counterclaim in a lawsuit) alleging that the Work
   or a Contribution incorporated within the Work constitutes direct
   or contributory patent infringement, then any patent licenses
   granted to You under this License for that Work shall terminate
   as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
   Work or Derivative Works thereof in any medium, with or without
   modifications, and in Source or Object form, provided that You
   meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works
       that You distribute, all copyright, patent, trademark, and
       attribution notices from the Source form of the Work,
       excluding those notices that do not pertain to any part of
       the Derivative Works; and

   (d) If the Work includes a "NOTICE" text file as part of its
       distribution, then any Derivative Works that You distribute must
       include a readable copy of the attribution notices contained
       within such NOTICE file, excluding those notices that do not
       pertain to any part of the Derivative Works, in at least one
       of the following places: within a NOTICE text file distributed
       as part of the Derivative Works; within the Source form or
       documentation, if provided along with the Derivative Works; or,
       within a display generated by the Derivative Works, if and
       wherever such third-party notices normally appear. The contents
       of the NOTICE file are for informational purposes only and
       do not modify the License. You may add Your own attribution
       notices within Derivative Works that You distribute, alongside
       or as an addendum to the NOTICE text from the Work, provided
       that such additional attribution notices cannot be construed
       as modifying the License.

   You may add Your own copyright statement to Your modifications and
   may provide additional or different license terms and conditions
   for use, reproduction, or distribution of Your modifications, or
   for any such Derivative Works as a whole, provided Your use,
   reproduction, and distribution of the Work otherwise complies with
   the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
   any Contribution intentionally submitted for inclusion in the Work
   by You to the Licensor shall be under the terms and conditions of
   this License, without any additional terms or conditions.
   Notwithstanding the above, nothing herein shall supersede or modify
   the terms of any separate license agreement you may have executed
   with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
   names, trademarks, service marks, or product names of the Licensor,
   except as required for reasonable and customary use in describing the
   origin of the Work and reproducing the content of the NOTICE file.
```

```
7. Disclaimer of Warranty. Unless required by applicable law or
   agreed to in writing, Licensor provides the Work (and each
   Contributor provides its Contributions) on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
   implied, including, without limitation, any warranties or conditions
   of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
   PARTICULAR PURPOSE. You are solely responsible for determining the
   appropriateness of using or redistributing the Work and assume any
   risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
   whether in tort (including negligence), contract, or otherwise,
   unless required by applicable law (such as deliberate and grossly
   negligent acts) or agreed to in writing, shall any Contributor be
   liable to You for damages, including any direct, indirect, special,
   incidental, or consequential damages of any character arising as a
   result of this License or out of the use or inability to use the
   Work (including but not limited to damages for loss of goodwill,
   work stoppage, computer failure or malfunction, or any and all
   other commercial damages or losses), even if such Contributor
   has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "{}"
   replaced with your own identifying information. (Don’t include
   the brackets!)  The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright {yyyy} {name of copyright owner}

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
```

```
limitations under the License.
```

Please refer to the LICENSE.txt file in the project root folder.

1.7 Connect with us

1.7.1 Development Contact

- Development Main Page: github.com/arobito
- Issue Tracker: github.com/arobito/arobito/issues
- Issue Tracker for Documentation: github.com/arobito/arobito-docs/issues
- IRC Channel #arobito on chat.freenode.net

1.7.2 Social Networks

- Twitter: @arobito_project
- Facebook: eeeeh, Face-what?

1.7.3 Other

- E-Mail: Put `arobito` in front of the "At"-sign and `freenet` after it. The top level domain is `.de`.

1.7.4 Additional Resources

Welcome New User

This document should help you getting started with our project. We're glad you're with us. If you need help or if you have questions, please do not hesitate to *drop us a line*.

Prepare yourself

1. Create a GitHub.com account You need a GitHub.com account to commit to the project. If you already have one, you can skip this step. Just make sure that we know about your username.

Open github.com and you will see the registration form right on the front page. Make sure you read the terms and conditions and sign up. Send us your new username, so we can add you to the list of collaborators for the repositories you're going to contribute to.

2. Install Git Client Software Most Linuxes and Unixes have a Git client in their package repositories. So you might have to run one of the following commands, depending on your operating system:

```
apt-get install git           # Debian, Ubuntu
yum install git               # CentOS
zypper install git            # OpenSUSE
```

If you live in the Windows world, there is a nice all-in-one installer, that contains everything you need, including GUI and command line tools, a SSH client and much more. Check out git-scm.com for details.

3. Create a SSH key pair We strongly encourage developers to access our repositories via SSH. This has several pros and cons [1], but from a developer perspective, the pros win, because GitHub allows anonymous access always via HTTPS.

Long story short: You need a key pair (a private and a public key). This is not the place to discuss facets of authorization, authentication and encryption. If you already have such a key pair, you can skip this step.

Execute this command in a shell:

```
ssh-keygen -b 8192
```

Almost every Linux and Unix have `ssh-keygen` preinstalled. Under Windows, the Git package above provides the tool.

You will see the following on your screen for a few seconds up to minutes...

```
Generating public/private rsa key pair.
```

... and a little later, you'll be ask to enter the filename to save the private key to. Default would be something like `$HOME/.ssh/id_rsa` or, under Windows, `%USERPROFILE%\.ssh\id_rsa`.

```
Enter file in which to save the key (~/.ssh/id_rsa):
```

We recommend calling your file `id_rsa_github` and put it in the standard directories mentioned above.

You will be asked for a passphrase next. This passphrase will be needed each time you use your key. We encourage using a passphrase, but it is not required. If you don't enter one, the key is unprotected and can be used anytime without prior unlocking.

When you see such a picture...

```
00:11:22:33:44:55:66:77:88:99:aa:bb:cc:dd:ee:ff documentation@doc.lan
The key's randomart image is:
+--[ RSA 8192]----+
|*            o.++|
| +           o*E+|
|  -.        .+=*.|
|    =    .   +=+ |
|       S      o..|
|     o o       . |
|          . . .  |
|           . .   |
|            .    |
+-----------------+
```

... your key is ready to rumble.

4. Add your public key to your GitHub account Open the GitHub Settings/SSH Keys page for your account. Click "Add SSH key" and paste the contents of your `id_rsa_github.pub` file to the "key" field. The "title" field can be filled with some name that helps you to identify the key.

> **Warning:** Make sure you paste the contents of the `.pub` file of your key! The private key (in the file without the `.pub` ending) is really meant to be kept secret!

After you clicked the "Add key" button, you'll receive an e-mail that confirms the action.

[1] see Git on the Server: The Protocols

5. Configure your SSH client You can easily configure your SSH client to automatically use your key when you connect with GitHub. Go to your `$HOME/.ssh` (`%USERPROFILE%\.ssh` under Windows) directory and create a file simply named `config` (if it does not already exist). Put the following information in this file:

```
Host github.com
User git
IdentityFile ~/.ssh/id_rsa_github
```

So, each time, you connect to GitHub, SSH automatically uses your key.

Note: The username `git` is correct and must not be replaced with your GitHub username! You are identified by your key.

6. Setup basic Git configuration There are a few global settings that should be applied to your Git installation. Skip this step when you already did that, but you can't destroy anything if you do it twice.

```
git config --global user.name "Your Real Name"
git config --global user.email your.mail@address.org
```

Note: The e-mail address should be the address you used to register with GitHub. If you want to use another one, you need to add this address to your GitHub account. This makes sure that your contributions are correctly associated with your account.

Note: Your real name and your e-mail address you enter here will be visible to anyone that takes a closer look at the contributions.

7. Fork the repositories Now you have everything you need to get started and you can fork our repositories. Forking means, that you get an independent copy of one or all of our repositories. Just log in with GitHub, and navigate to our project page under github.com/arobito. Select the repository you want to fork and click the "fork" button in the upper right corner.

Now you can clone your fork to your own machine:

```
git clone git@github.com:<username>/arobito.git
```

For the documentation, use

```
git clone git@github.com:<username>/arobito-docs.git
```

Make sure you replaced `<username>` with your GitHub username!

So, that's it! You are ready to go! We have some extra tips for you about working with your fork, refreshing it and making pull requests - just below the next step.

Note: When you are a team member of the Arobito Project organization you *could* access our repositories directly. There are several situations where this might be the right thing, but we strongly encourage you to still work on your own fork and make pull requests.

8. Setup signing key (optional) If you do not know what we are talking about here, simply skip this step.

If you are familiar with the ideas of commit and tag signing via GPG, you may want to setup your GPG key. This can be done globally for all repositories (ABCD1234 is your key ID)...

```
git config --global user.signingkey ABCD1234
```

... or local to a specific clone:

```
cd myclone
git config user.signingkey ABCD1234
```

Refer to the Tagging Section of the Pro Git e-book for more information on tag signing.

Note: It does not make sense to sign every commit. Basically, it would be enough to sign tags for releases. But currently, we have not established any rules on that.

Work with the code

We're not going to much into detail here. If you need more information, visit the Pro Git e-book.

Basic Branching and Merging

Configure Upstream Repository When you are about to develop a new feature or a patch, make sure your fork is up-to-date. To make your clone "updatable", you may need to add the Arobito repository as upstream repository. You need to do this only once on a clone. You can verify the upstream configuration with the command `git remote -v`, executed in your clone's root path. If the upstream repository is still missing, the output looks like this:

```
origin  git@github.com:<username>/arobito.git (fetch)
origin  git@github.com:<username>/arobito.git (push)
```

Add our repository as upstream repository:

```
git remote add upstream https://github.com/arobito/arobito.git
```

Use the HTTPS repository URL here to circumvent authentication issues when you are not a team member. Team members can use the `git@github.com:arobito/arobito.git` SSH connection with their key. To verify the upstream configuration you can execute the `git remote -v` command again. You should see something like that:

```
origin  git@github.com:<username>/arobito.git (fetch)
origin  git@github.com:<username>/arobito.git (push)
upstream        https://github.com/arobito/arobito.git (fetch)
upstream        https://github.com/arobito/arobito.git (push)
```

To sync your clone with our repository, simply fetch and merge the upstream:

```
git fetch upstream
```

For each branch that you want to base your work on (in most cases, this would be `master` only), execute the following:

```
git checkout master
git merge upstream/master
```

Use `git push --all` now to sync your remote repository also.

Most times, you would like to switch now to the `master` branch (just execute `git checkout master`) and create your own feature branch and start working.

Note: The GitHub help page Managing Remotes has more detailed information on that.

Create a feature or patch branch If you are about to create a feature, create a new feature branch:

```
git checkout -b features/<your_feature_name>
```

Work, commit and push as much as you need. When your feature is done, proceed with a pull request.

When you are about to develop a patch or a fix, create a patch branch:

```
git checkout -b patches/<your_patch_name>
```

When your patch refers to an issue from our issue tracker, name the patch accordingly, e. g. “Issue_213”. Proceed with development, commit and push your changes.

Create a pull request When you’re done with your work, you might want to create a pull request. This notifies the project administrators of your work and allows them to review and merge your contribution.

Creating pull requests with GitHub is very easy and straight-forward. The GitHub help pages contain a chapter about creating pull requests.

Delete a branch When your pull request has been accepted, you might want to delete your branch. First, switch to the master branch:

```
git checkout master
```

Now you can delete the old branch:

```
git branch -D patches/<your_patch_name>
```

or

```
git branch -D features/<your_feature_name>
```

Now, delete the remote branch:

```
git push origin :patches/<your_patch_name>
```

or

```
git push origin :features/<your_feature_name>
```

To get the results of the pull request back to your clone, just update your clone as described above:

```
git fetch upstream
git checkout master
git merge upstream/master
git push --all
```

Submodules In the documentation repository, we’re using a submodule under `code/` that contains a defined version of the code from the main repository. This kind of decouples the documentation from the commits in the main repository.

When you clone the documentation repository for the first time, the directory `code/` is empty.

Checking out a submodule for the first time To get the code, simply execute the following command from the root directory of your documentation repository clone.

```
git submodule update --init
```

This brings the `code` directory to the defined version.

Updating a submodule It may happen that you need to update the contents of the submodule to the current version from the main repository. Simply go to the `code` directory and execute

```
git pull origin master
```

The contents will be updated to the most current version. Change back to the root directory of the your documentation repository clone and enter

```
git commit
```

Now you can work with the new code base.

Todo

Detailed instructions

More stuff to read

To get familiar with Git, we recommend taking a look at the great Pro Git e-book, which is available for free in several formats and as web page.

CHAPTER 2

FAQ

2.1 So, what is Arobito?

Arobito is a little project created by some computer scientists who left university years ago. They try to build some kind of a robot from scratch.

We're trying to learn about controlling hardware and software, detecting obstacles and moving around.

It's all an experiment. Nothing to lead to a productive usage. Just for fun.

2.2 Sounds good, can I join?

Yes, of course. This is an open source project. The easy way: Fork the source and build your own stuff on it. We're welcoming pull requests.

If you want to join our developer circle please don't hesitate to send us a message via github.com. For new users, we created a *quick guide to the project*.

2.3 Where can I download stuff?

This is not something you can install on your PC and simply run it. Not yet. So grab the source and go on.

2.4 Where are the building and running instructions?

We're still working on very basic code and very basic documentation. Please be patient.

2.5 What languages are you using?

The primary project language is Python 3.4 (see *Coding for Arobito* for details). But there's also a lot of HTML, CSS, JavaScript, Shell Scripts. A nice zoo.