# ardas-tools Documentation

***Release 1.0***

**Christophe Bastin**

**Jan 22, 2019**

# Contents:

CHAPTER 1

# Getting Started

Datas are stored locally and can be sent to an influxdb database an visualized into grafana.

These tools don't cover influxdb and grafana installation, see dedicated websites for tutorial on how to do this.

## 1.1 Preparation

- Prepare a raspberry pi with a RASPBIAN LITE image and an arduino board.
- Make sure the pi and arduino are turned-off
- Insert the ardas shield in the arduino board
- Connect the arduino board to the raspberry pi with a USB cable (no external power cable for the arduino board)
- Power up the raspberry pi and wait for the raspberry pi to boot
- Log on the raspberry pi (using ssh)

**Warning:** Make sure you don't have a random cron task activated

## 1.2 For visualization(optional)

Log on you influxdb server and create a new database:

```
create database DATABASE_NAME
```

Create a user for this database and grant him permissions on this database:

```
create user USERNAME with password 'MYPASSWORD'
grant all on "DATABASE_NAME" to "USERNAME"
```

Add the database as data source on grafana.

## 1.3 Install Git

```
sudo apt install git
```

## 1.4 Download the code

To deploy ardas-tools including ardas:

```
rm -rf * .git .gitignore && git init && git remote add origin https://github.
↪com/UMONS-GFA/ardas-tools.git && git pull origin master
```

## 1.5 Install the required packages

```
./raspbian_upgrade.sh
```

## 1.6 Initialization and setup

- For production

  To initiate stable branch of ardas:

  ```
  bash init_ardas.sh
  ```

  To update stable branch:

  ```
  bash update_ardas.sh
  ```

  These scripts can take quite a long time to retrieve the python dependencies, espacially when using a rapsberry pi one. See the *Options* option for an alternative...

- For developers

  To initiate development branch of ardas

  ```
  bash init_ardas.sh --dev
  ```

  To update development branch

  ```
  bash update_ardas.sh --dev
  ```

  Follow the instructions to set the approriate sensors and acquisition settings.

## 1.7 Configure the PYTHONPATH

```
nano ~/.bashrc
```

add at the end:

```
export PYTHONPATH=/home/pi/ardas
```

Refresh your profile:

```
source ~/.bashrc
```

## 1.8 Generate the sensors

You can use the *uncalibrated_sensors_generator.py* script to generate 4 uncalibrated sensors:

```
python3 /home/pi/ardas/ardas/uncalibrated_sensors_generator.py
```

## 1.9 Configure the ardas

- Set the default arduino config, default netid will be 255:

  ```
  bash reset_arduino_EEPROM.sh
  ```

- Upload the sketch in the arduino:

  ```
  bash upgrade_sketch.sh
  ```

## 1.10 Define your settings

Make a copy of the *settings_example.py* and name it *settings.py*:

```
cp /home/pi/ardas/ardas/settings_example.py /home/pi/ardas/ardas/settings.py
```

Configure this file for your needs.

## 1.11 Add a cron task

```
crontab -e
```

and add:

```
PYTHONPATH=/home/pi/ardas

# m h  dom mon dow   command
@reboot /usr/bin/python3 /home/pi/ardas/ardas/raspardas.py > /home/pi/ardas/cronlog.
↪log 2>&1 &
```

## 1.12 Start logging

To reboot the raspberry.py and start logging:

```
bash reboot.sh
```

Don't forget to type a message to explain in which circumstances (and eventually by whom) the system is rebooted. It show in the logs when the system will restart.

## 1.13 Options

- Update code without automatically rebooting(only for update_ardas script):

```
--noreboot
```

- Prevent creating a new virtual environment and installing all dependencies

  Installing dependencies could be slow on a raspberry pi, espacially a . If you plan to install several identical loggers, it could be advisable to create a disk image with the right dependencies installed in the virtual environment. The following option could then be used to prevent the system from creating a new virtual environment and retrieving dependencies:

  ```
  --nopip
  ```

## 1.14 Debug

If there is a communication problem with the Arduino, you can try to debug with picocom:

```
apt install picocom
```

Make sure the python cron task is not running:

```
ps -aux | grep python3
```

Launch picocom:

```
picocom -b 57600 /dev/ttyACM0
```

Use Ctr+a then Ctrl+c to enable echo.

To quit, use Ctrl+x

Call the ardas with his netid:

```
-255
```

Set naïve data mode:

```
#ND
```

You can now change the default config with ZR command:

```
#ZR station netId integrationPeriod nbInst sensor1 sensor2 sensor3 sensor4 code

Ex: #ZR 1111 222 3333 4 0001 0002 0003 0004 31
```

Set raspardas data mode:

```
#RD
```

Change your settings accordingly

CHAPTER 2

# Advanced scripts

## 2.1 clean.sh

Clean the log and data directory

## 2.2 purge.sh

Remove all files from the current directory and get latest ardas release

## 2.3 reboot.sh

Write a message to influx event database before rebooting the computer

## 2.4 reset_arduino_EEPROM.sh

Build arduino-reset sketch and upload to the arduino. Arduino-reset flushes the serial connection, reconfigure the arduino and set raspardas mode to true in EEPROM.

## 2.5 update_ardas.sh

Update ardas code version.

## 2.6 upgrade_sketch.sh

Build a new version of arduino sketch and upload it to the arduino.

CHAPTER 3

# Indices and tables

- genindex
- modindex
- search