



# SPARKCORE WORKSHOP

Fabian Morón Zirfas  
University of Applied Sciences Potsdam (Germany)  
2014

Vorstellen. Ggf für unbekannte

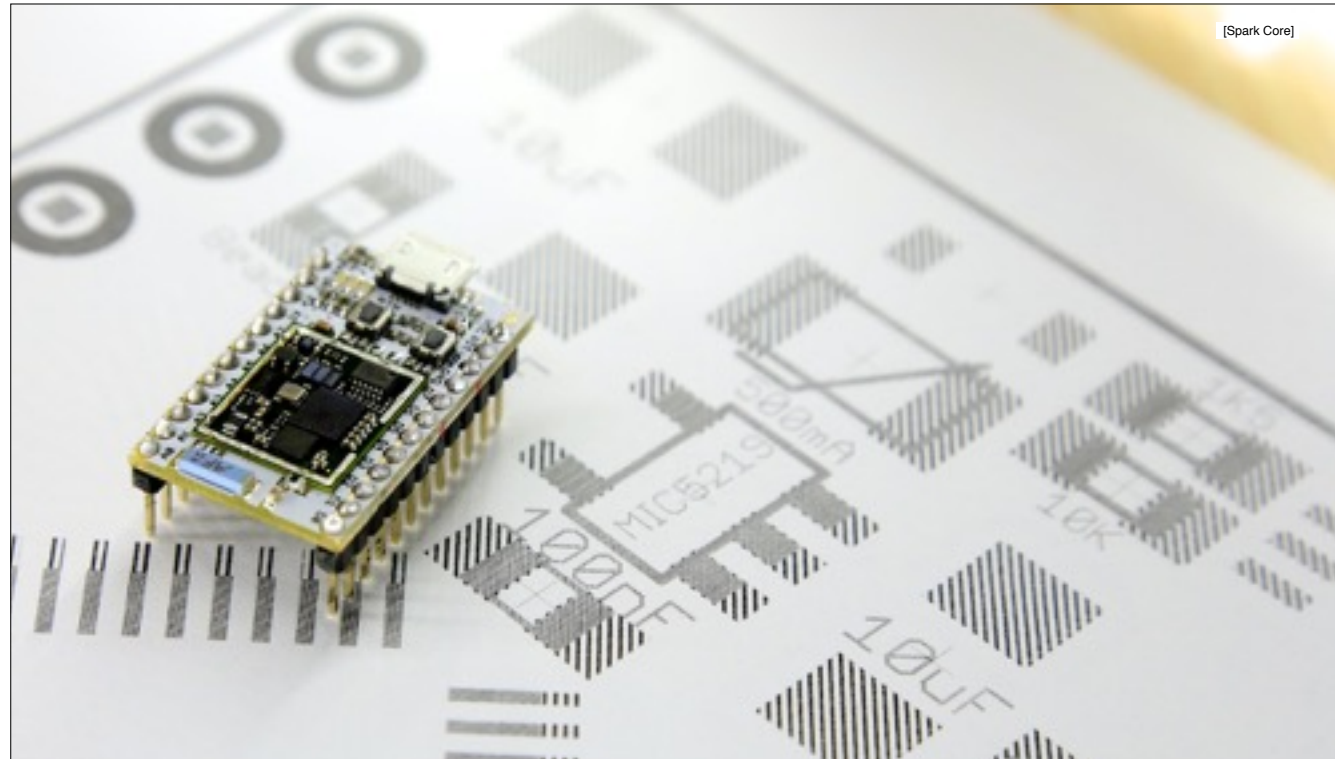
## TOPICS

1. What is Sparkcore?
2. How to get it?
3. Demo
4. Prerequisites
5. Command Line Interface
6. Setup, Claim & Blink
7. Walk through code examples (firmware & web)
8. Trix & Hints
9. Ups & Downs & ToDos
10. Hands On

# WHAT IS SPARKCORE?



Spark OS is a complete open source operating system for cloud-connected things.

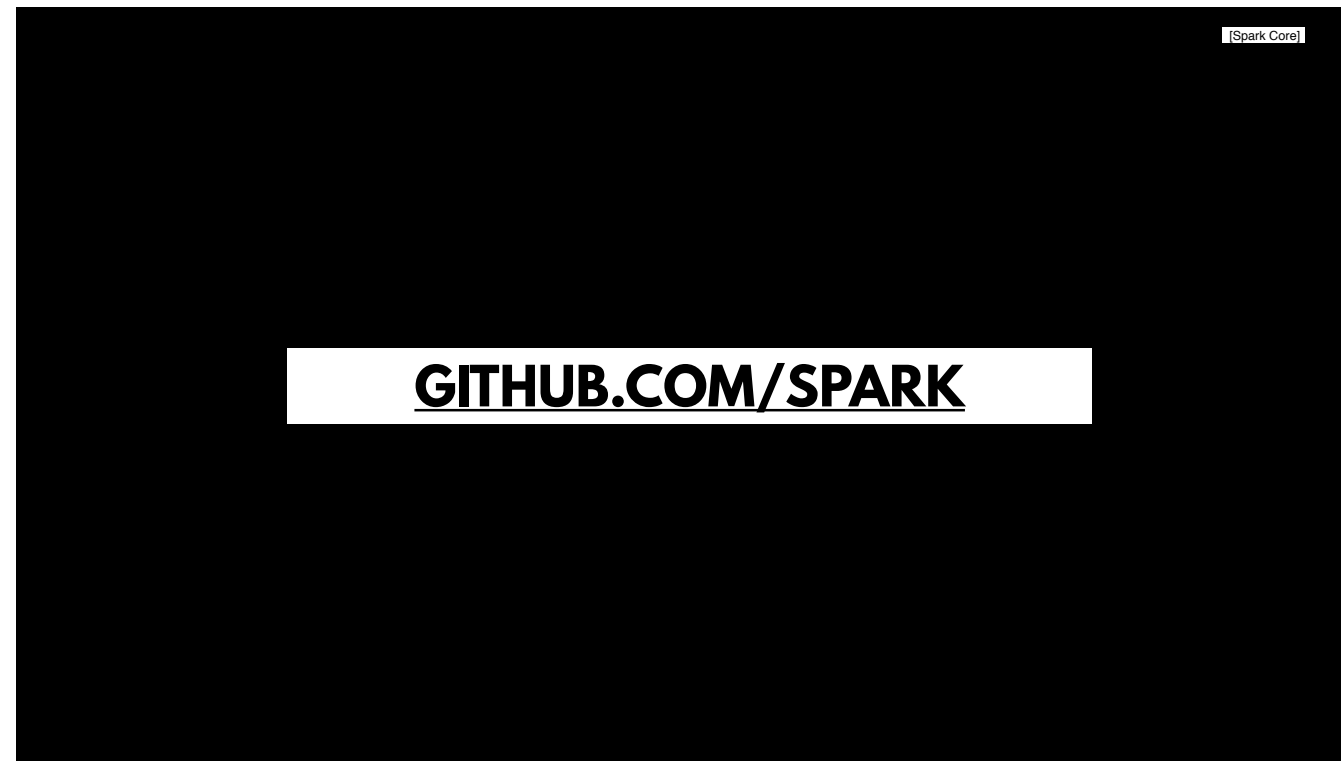


The spark core is programmed with the Arduino language and has some additional classes to connect to the cloud.



<https://www.spark.io/>

You can get all the information you need with code examples and other things from their website and their github account

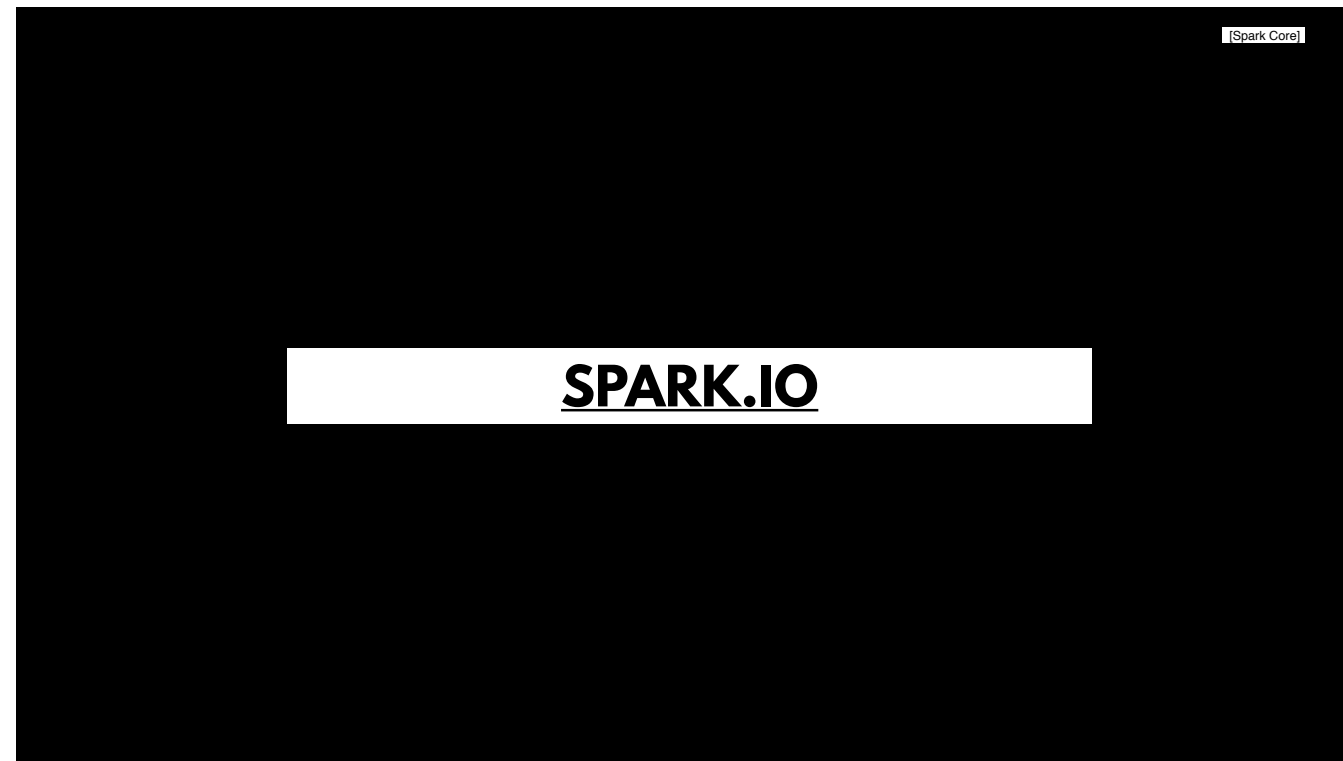


<https://www.spark.io/> <https://github.com/spark>

You can get all the information you need with code examples and other things from their website and their github account



**HOW TO GET IT?**



Cost: 39\$

If you order make a request to don't send it with USPS. Use UPS or FedEx

**DEMO**

## PREREQUISITES

## PREREQUISITES

- Terminal or CMD
- Homebrew (OSX)
- (Git installed via Homebrew XCode command line tools)
- Node.js
- Bower/Grunt
- spark-cli
- spark account
- CoolTerm (OSX optional)



Package Manager for OS X

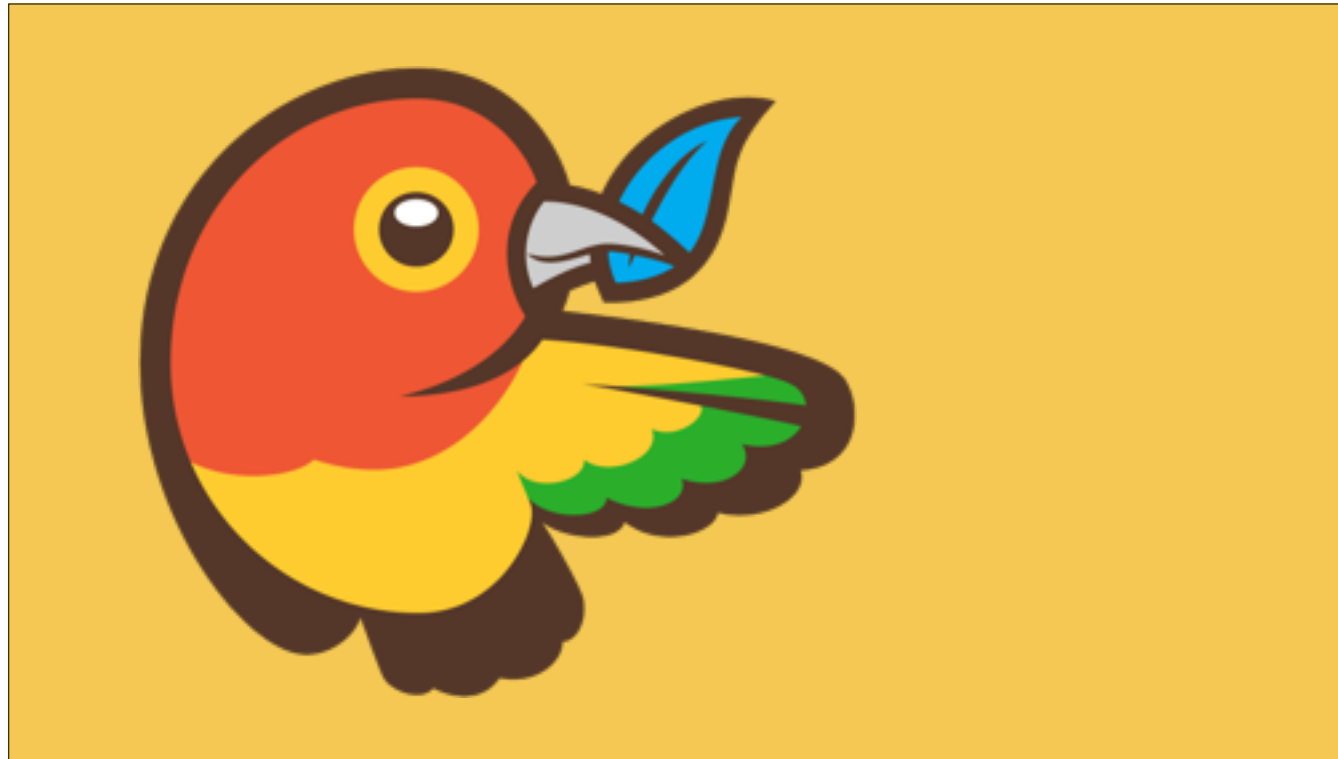


version control system



Server side JS





library package manager



js task runner



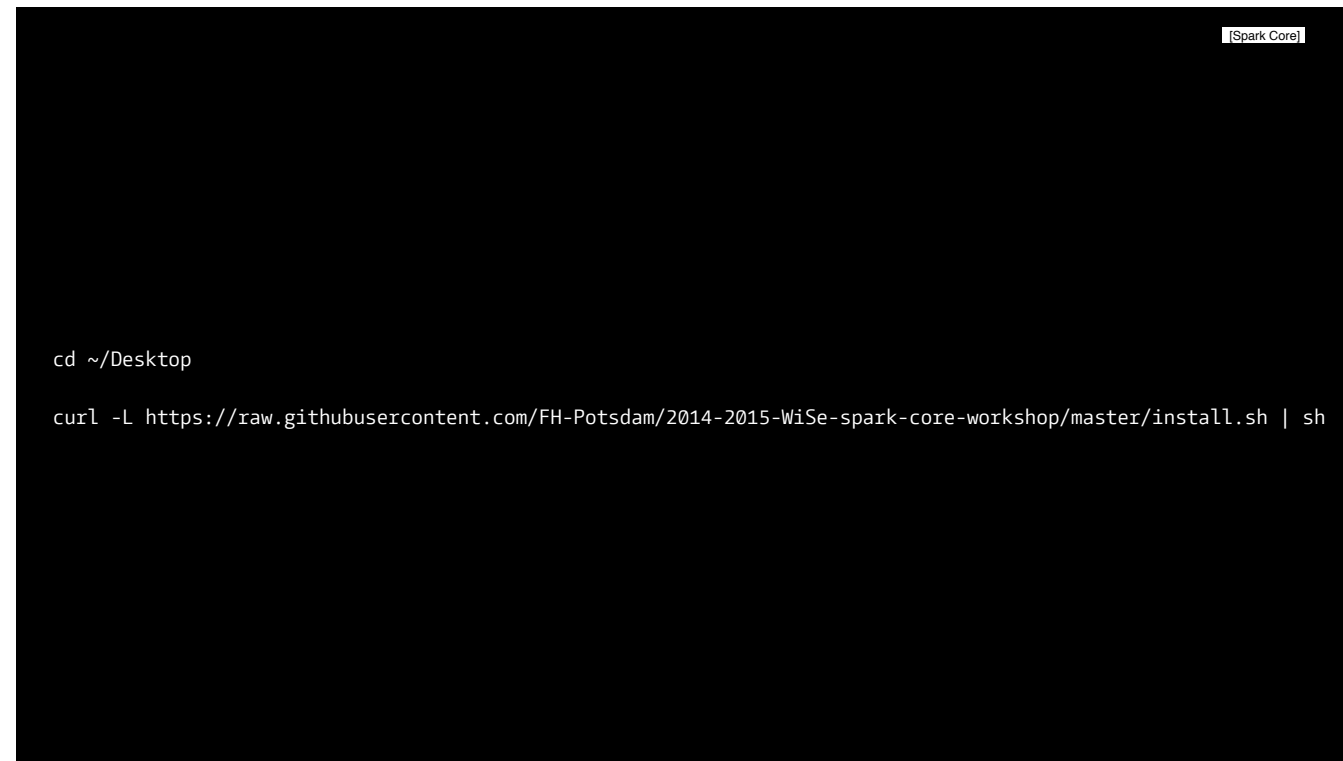
spark CLI and an account

<https://www.spark.io/signup>

Account erstellen <https://www.spark.io/signup>

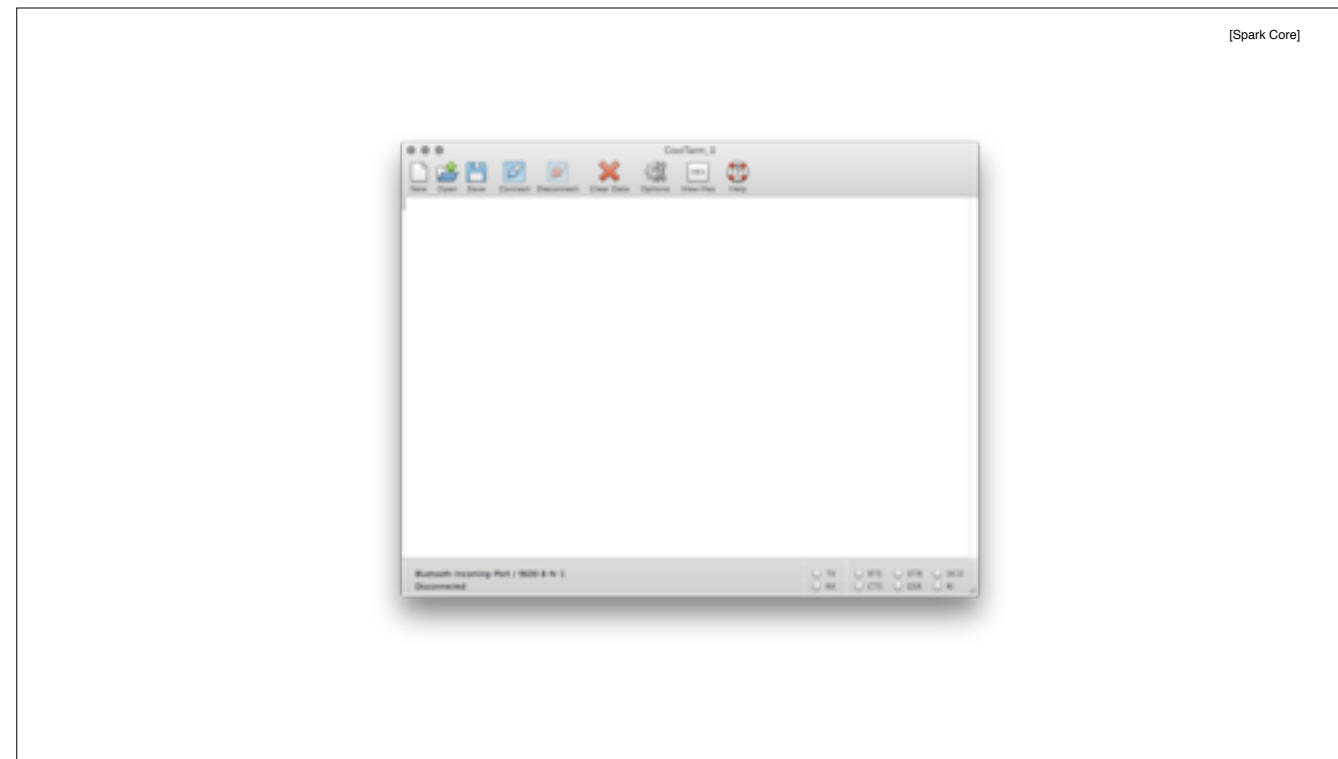
[Spark Core]



A terminal window with a black background and white text. In the top right corner, there is a small white box containing the text "[Spark Core]". The terminal displays two lines of code: "cd ~/Desktop" followed by a blank line, and then "curl -L https://raw.githubusercontent.com/FH-Potsdam/2014-2015-WiSe-spark-core-workshop/master/install.sh | sh".

```
[Spark Core]  
  
cd ~/Desktop  
  
curl -L https://raw.githubusercontent.com/FH-Potsdam/2014-2015-WiSe-spark-core-workshop/master/install.sh | sh
```

Easiest way to install



CoolTerm for serial communication / wifi setup (used cores)

```
brew install caskroom/cask/brew-cask  
brew cask install coolterm  
# will be linked to ~/Applications/CoolTerm.app
```

This is the easy and nerdy way to install CoolTerm



# SPARK-CLI

[command line interface]

```
spark

Welcome to the Spark Command line utility!
https://github.com/spark/spark-cli

Usage: spark <command_name> <arguments>
Common Commands:

    setup, list, call, get, core, identify, flash, subscribe
    compile, monitor, login, logout, help

Less Common Commands:
    cloud, config, function, keys, serial, udp, variable
    webhook

For more information Run: spark help <command_name>

+ ~ █
```

## SETUP, CLAIM & BLINK

## SETUP & CLAIM

**FRESH SPARK?**

## SETUP VIA USB

LED flashing blue on connection  
(listening mode)

Listening mode flashing blue

```
spark login
```

```
spark setup
```

Wifi Credentials?

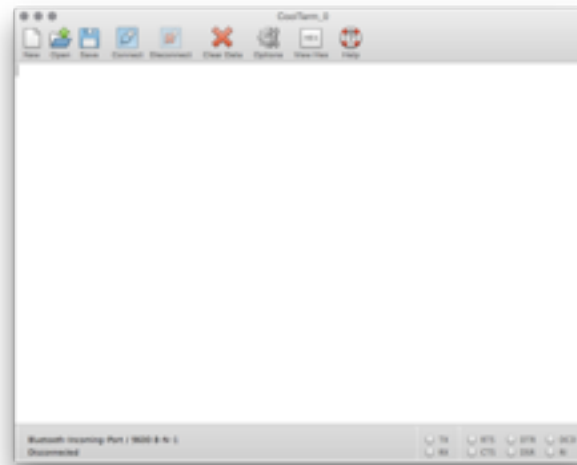
**USED SPARK?**



## LISTENING MODE

(Hold the "Mode" button for 3 seconds, then the LED should be flashing blue. To remove all WIFI settings hold the "Mode" button 10 seconds

```
spark setup wifi
```



# COOLTERM

## Settings & Usage

- Baudrate: 9600
- Data Bits: 8
- Parity: none
- Stop Bits: 1
- w: Set up your Wi-Fi SSID and password
- i: ("i" as in identify) Read out the Spark Core ID

# MODES

- **Blinking blue:** Listening for Wi-Fi credentials
- **Solid blue:** Getting Wi-Fi info from app
- **Blinking green:** Connecting to the Wi-Fi network
- **Blinking cyan:** Connecting to the Spark Cloud
- **Blinking magenta:** Updating to the newest firmware
- **Breathing cyan:** Connected!



[See an animation](#)

# BLINK

[Out of the Box]

```
spark function call YOUR_CORES_NAME_OR_ID digitalWrite "D7/HIGH"
```

tinker



# TINKER

[docs.spark.io/tinker](http://docs.spark.io/tinker/)

<http://docs.spark.io/tinker/>

[Spark Core]



[Spark Core]

**BLINK**

[used core]

factory reset

```
spark flash [CORE ID OR NAME] tinker
```

## FLASH TINKER

- Try a factory reset. Hold down both buttons, then release the RST button, while holding down the MODE button. The LED should begin flashing yellow. Continue holding down the MODE button until you see the Core change from flashing yellow to flashing white. Then release the button. The Core should begin after the factory reset is complete. [link](#)

To reflash Tinker from within the app:

- iOS Users: Tap the list button at the top left. Then tap the arrow next to your desired Core and tap the "Re-flash Tinker" button in the pop out menu.
- Android Users: With your desired Core selected, tap the options button in the upper right and tap the "Reflash Tinker" option in the drop down menu. re should begin flashing blue after the factory reset is complete.

<http://docs.spark.io/connect/#appendix-factory-reset>

```
spark function call YOUR_CORES_NAME_OR_ID digitalWrite "D7/HIGH"
```

tinker

# CODE EXAMPLES

[firmware & web]

**SRC/\*.\***

My examples (hook up my cores)



## SPARK-HELPER/\*.\*

<https://github.com/jflasher/spark-helper>

<https://github.com/jflasher/spark-helper>

[interface.fh-potsdam.de/spark-core/helper](http://interface.fh-potsdam.de/spark-core/helper)

Credentials von mir

**SPARK-AJAX/\*.\***

<http://fmz.pictor.uberspace.de/2014-2015-WiSe-spark-core-workshop/spark-ajax/>

[http://fmz.pictor.uberspace.de/2014-2015-  
WiSe-spark-core-workshop/spark-ajax/](http://fmz.pictor.uberspace.de/2014-2015-WiSe-spark-core-workshop/spark-ajax/)

<http://fmz.pictor.uberspace.de/2014-2015-WiSe-spark-core-workshop/spark-ajax/>

## WEB-INTERFACE/\*.\*

<https://github.com/suda/spark-web-interface/>

<http://suda.github.io/spark-web-interface/>

[suda.github.io/spark-web-interface/](http://suda.github.io/spark-web-interface/)

[fh-potsdam.github.io/spark-web-interface/](http://fh-potsdam.github.io/spark-web-interface/)

<http://suda.github.io/spark-web-interface/>

<http://fh-potsdam.github.io/spark-web-interface/>

**SPARK-CLIENT-SIDE/\*.\***

error!



## TRIX & HINTS

## **SPLIT DEVELOPMENT**

hardware prototyping (with Arduino)  
cloud connection (with Spark Core)

**DON'T WRITE BLOCKING CODE**

```
delay(5000); // bad idea
```

```
// good idea
int update = 5000;
void setup(){}
void loop() {
    if(millis() > update){
        update+= millis();
        // do something
    }
}
```

```
spark help [COMMAND]
```

spark list

```
spark monitor [VARIABLE NAME]
```



```
spark function call [CORE NAME] [FUNCTION NAME] ["DATA"]
```

spark subscribe mine

```
spark flash [YOUR_CORES_NAME] [SOURCECODE_FOLDER]
```

via cloud

```
spark compile [SOURCECODE_FOLDER or FILE]
```

cloud compile to local file

## DFU MODE

(DEVICE FIRMWARE UPGRADE)

If you wish to program a Core with a custom firmware via USB, you'll need to use this mode. This mode triggers the on-board bootloader that accepts firmware binary files via the dfu-utility. Procedure:

- Hold down BOTH buttons
- Release only the RST button, while holding down the MODE button.
- Wait for the LED to start flashing yellow
- Release the MODE button
- The Core now is in the DFU mode.

<http://docs.spark.io/cli/#installing-advanced-install>

```
brew install dfu-util
```

<http://docs.spark.io/cli/#installing-advanced-install>

```
spark flash --usb firmware_XXX.bin [CORE ID or CORE NAME]
```

## UPS & DOWNS & TODOS



## UP:OUT OF THE BOX

**DOWN:KNOWLEDGE NEEDED**

Server, Command Line, HTML, CSS, JS

**DOWN:SPARK.SUBSCRIBE()**

Wo lag das Problem?

# TODO:SPARK-SERVER

<https://github.com/spark/spark-server>

# TODO:SPARKJS (CLIENT SIDE)

<https://github.com/spark/sparkjs>

## TODO:SPARKJS (SERVER SIDE)

<https://github.com/spark/sparkjs>

**Q&A**

**HANDS ON**