



# Simplifying Integration of Sensor Data Using the NFC Enabled Multi-Sensors node, **STEVAL-SMARTAG1**

## Hands-on Workshop

John Tran  
STMicroelectronics



**Technology Tour 2019**  
Minneapolis, MN | October 24



# What's NFC?



# Radio Frequency IDentification

RFID is a short range contactless communication technology

Employs an active reader/writer and a passive tag/transponder

The reader powers the tag and initiates the communication

Frequency Bands

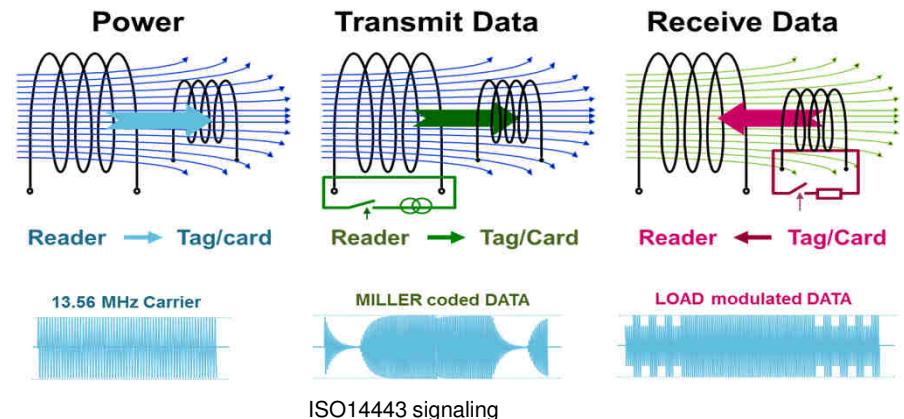
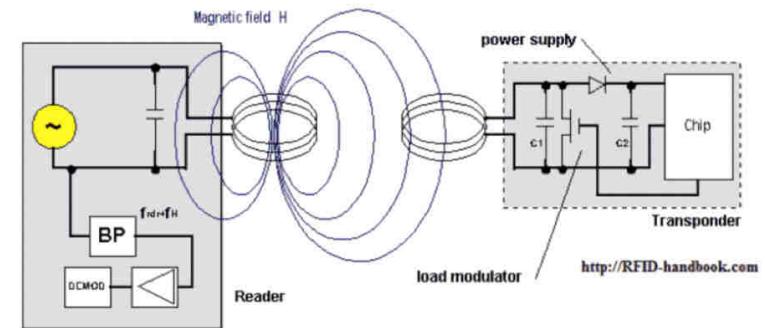
- LF (120-150 KHz)
- HF (13.56 MHz)
- UHF (433 to 960 MHz)

Operating ranges

- Proximity (few cm)
- Vicinity (up to 1m)
- Long Range (up to 10m)

Applications

- Transit, payment, inventory tracking, building and car access, etc.





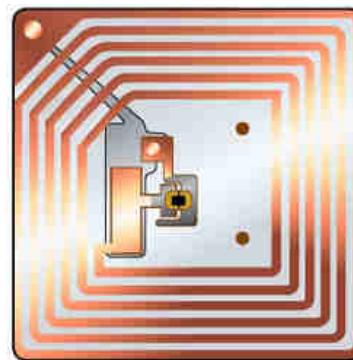
# RFID Technologies at a Glance

RFID	LF	HF	UHF
Coupling mode	Inductive	Inductive	Electro-magnetic backscatter
Operating frequency	125kHz – 134kHz	13.56MHz	860MHz – 960MHz
Antenna	Coil	Coil	Dipole
Max operating distance	up to 1m	Vicinity: up to 1m Proximity: up to 10cm	~10m
Regulation	Worldwide harmonized	Worldwide harmonized	Different regulations per country
Standards	ISO14223 ISO18000-2	ISO14443 A/B ISO15693 ISO18092 ISO18000-3  NFC Forum	ISO18000-6 B/C EPC Class 1 Gen 2  RAIN RFID
Environmental influences	Small influence on operating distance Works in metal and industrial environment	Small influence on operating distance Works in metal and industrial environment	Influence on operating distance by reflection and absorption (metal and liquids)
Applications	Animal tagging	Product identification Public transport / Libraries Access control / Payment	Pallets and container ID Retail / Logistics Authentication
ST solutions		X	X



# Comparison of RFID vs. Barcode

- Works in harsh and contaminated environments
- Options to implement data security / encryption
- Protocol supports “anti-collision” which allows reading/writing of an individual tag when multiple tags are in the reader field
- Unlike barcodes, line of sight is not necessary
- Offers data storage with options to interface to local processing (i.e. MCUs, FPGAs, etc.)





# NFC Technology at a Glance

An interactive technology enabling engagement with IoT devices



- Near Field Communication, a **short range wireless technology**
  - Operating at **13.56MHz**
  - Based on the **RFID HF standard (ISO14443 & ISO15693)**
- **Interactive** and **zero power**, enabling convenient connection to the Internet of Things



- ➔ **NFC-enabled mobile phone can engage with items by a simple tap**



- NFC is maintained by the **NFC Forum**
  - Ensures **Interoperability** between devices
  - **Standardized** use cases (web link, Bluetooth handover,...)
- Fast growing deployment in **Mobile phone**
  - Since 2010 virtually all **Android phones support NFC**
  - Apple has use **NFC ApplePay** since 2014, and in 2017 Apple added support of NFC reader mode in **iOS11** onward





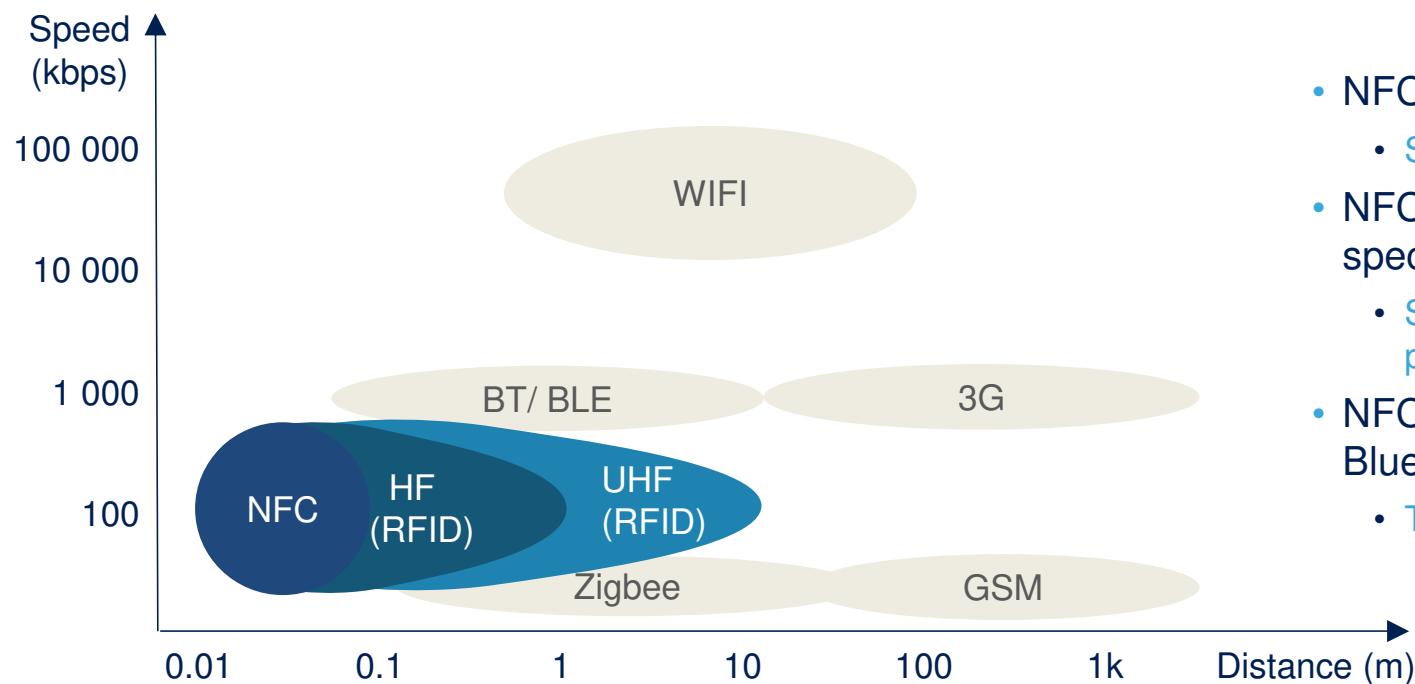
# NFC in Depth

- Requires an action such as bringing your card/phone near the reader in order to use
- Maximum data transfer rate is 424 kbps
  - Proprietary modes can go up to 6.8Mbps
- NFC operating modes
  - Read/Write (reader-to-passive tag/card)
  - **Card Emulation (e.g. Apple Pay, Android Pay)**
  - **Peer-to-Peer (e.g. reader-to-reader)**
- Applications include access control, payments, electronic passports, transportation ticketing, device pairing, data file exchange (e.g. Android Beam)
- Various combinations of memory and security
- Standards and specifications
  - ISO14443A, ISO14443B, Sony FeliCa, ISO15693





# NFC in the Wireless Spectrum



- NFC is a flavor of RFID
  - Subset of RFID HF standard: 13.56MHz
- NFC is complementary in the wireless spectrum
  - Short distance, Low data-rate & Zero power for applications
- NFC is complementary to Wi-Fi and Bluetooth technologies
  - Tap & Pair: convenient pairing use case



# History of NFC

- 1983 RFID Patented
- 2003 NFC approved as an ISO/IEC standard
- 2004: Philips, Sony and Nokia create the NFC Forum
- 2006: First specifications for NFC tags are released
- 2006 Nokia releases the Nokia 6131 NFC compatible phone
- 2007: Commercial roll-outs in US, Europe, and China
- 2008: NFC Forum membership increases to +150 members
- 2010: Samsung releases the first Android-based NFC phone, the Nexus S
- 2011: Google Wallet introduced
- 2012: NFC in Windows 8
- 2013: Apple Pay introduced. Rolled out to mass market in 2014.
- 2017: Apple Core NFC introduced in iOS11





# NFC Forum Tag Types

	Type 1	Type 2	Type 3	Type 4	Type 5
Products	BROADCOM “Topaz”	NXP NTAG MIFARE	SONY “FELICA”	ST25T(A&B) DESFire	ST25TV iCode
Specification	ISO 14443-A	ISO 14443-A	JIS X 6319-4	ISO 14443-A/B	ISO15693
Data Rate	106 kbit/s	106 kbit/s	212/424 kbit/s	106-424 kbit/s	26kbit/s
Protocol	Specific command Set	Specific command Set	FeliCa protocol	ISO 14443-4 ISO 7816-4	ISO/IEC 21481
Cost	Low	Low	Moderate	Moderate	Low
Use cases	Tags with small and fixed memory for single applications		Flexible tags with larger memory offering multi-application capabilities.		Long range tags with multiple applications
Memory type	Memory cards		CPU cards/Memory Cards0		Memory Cards

# NFC Standards

NFC specification  
→ **Upper layer SW**



RFID HF ISO standards  
→ **HW / SW protocol**



NDEF (NFC Data Exchange Format)

NFC Forum  
Type 2 and Type 4

NFC Forum  
Type 5 \*

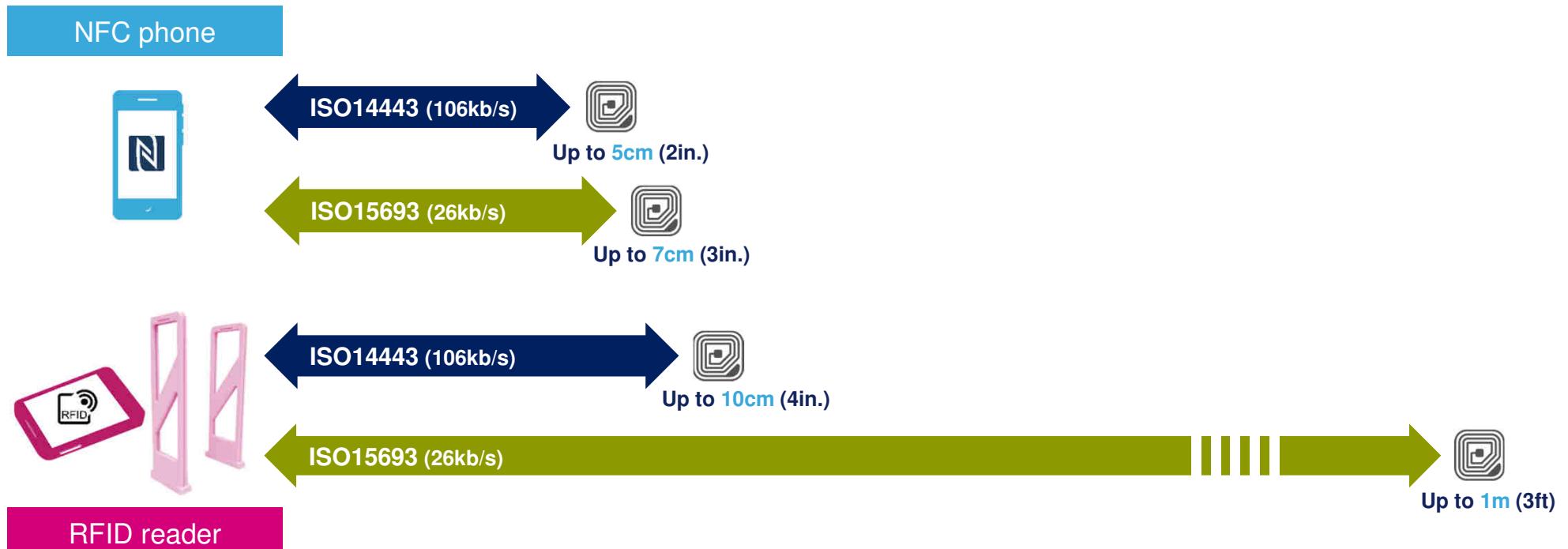
ISO14443  
Type A and Type B  
« Short Range »  
106kbps

ISO15693  
« Long Range »  
26kbps

(\*) ISO15693 integrated in NFC Forum specifications in October 2015 as NFC Forum type 5 (aka type V)

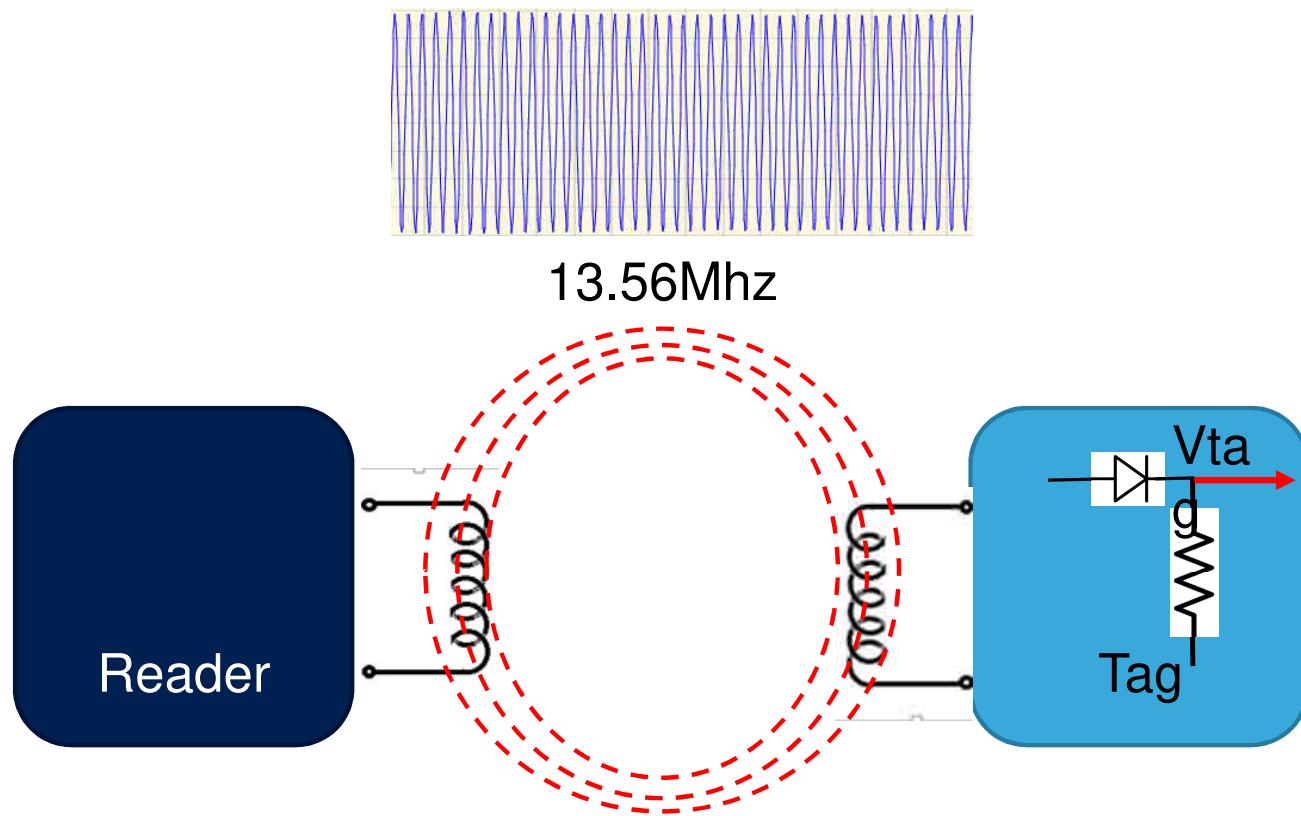


## Typical NFC / RFID range



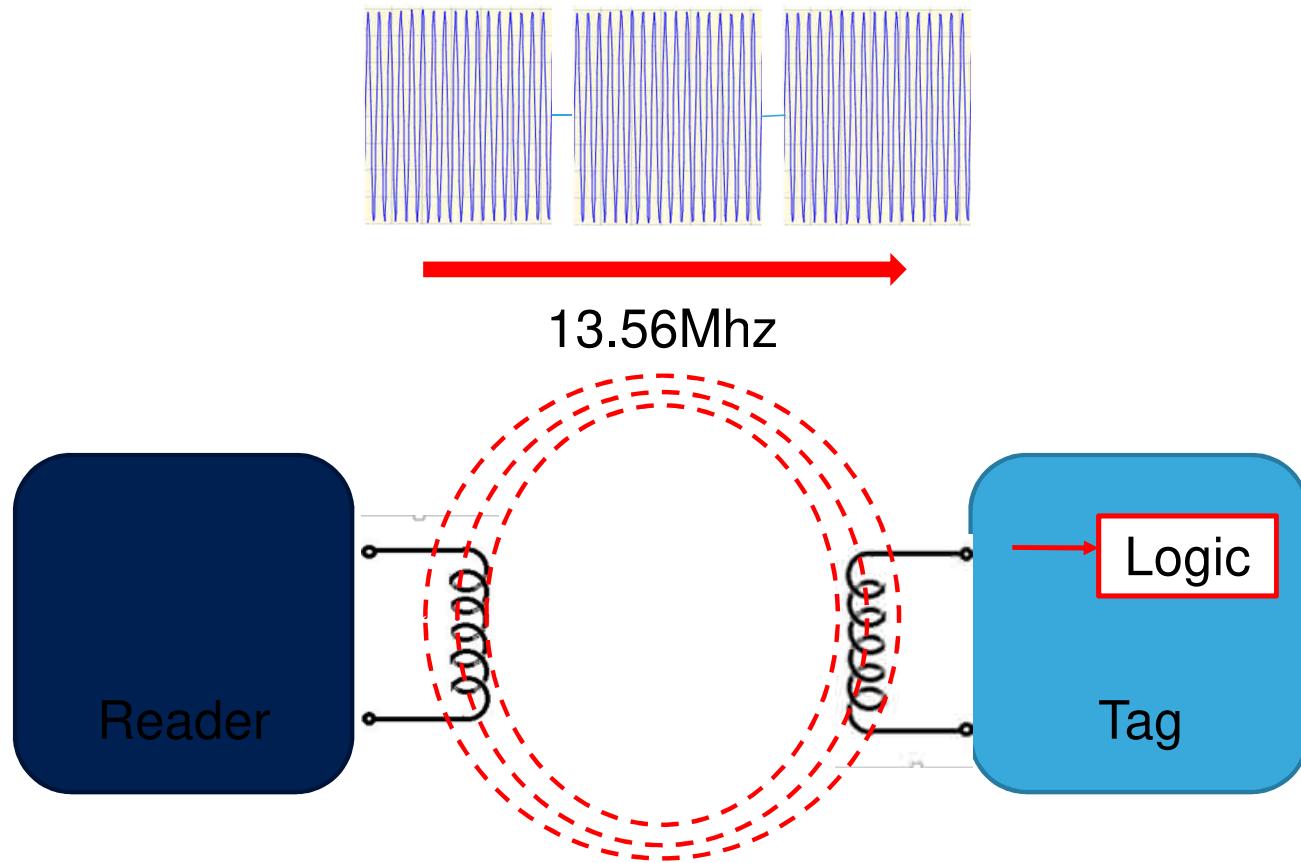
- ISO14443 is called « **short range** » standard while with higher RF speed
- ISO15693 is called « **long range** » standard

# HF Communication – The Basis



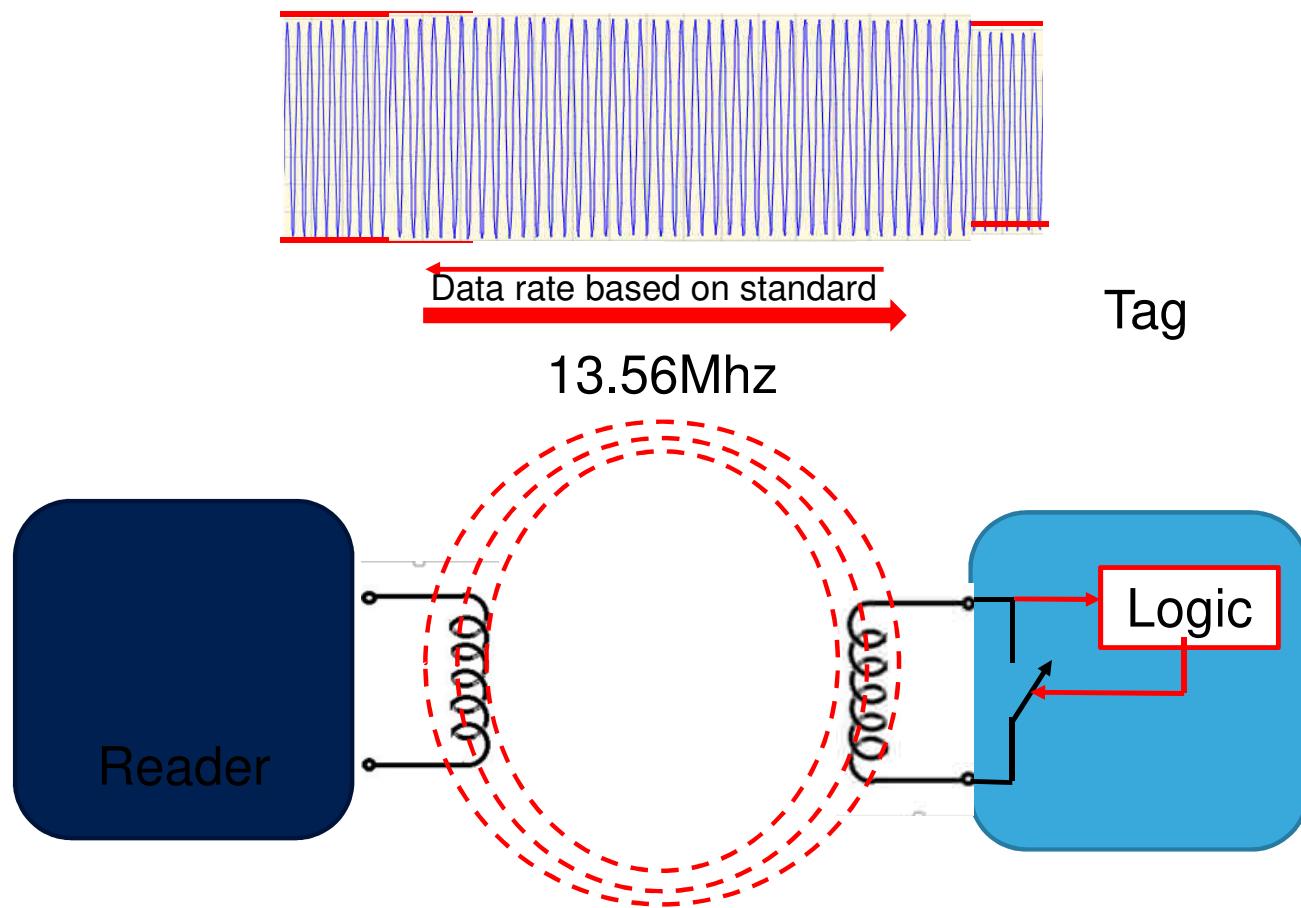
# HF Communication

Reader to Tag



# HF Communication

## Tag to reader





## HF Communication

Communication	ISO14443A	ISO14443B	ISO15693	Felica
Reader to Tag	100% ASK Miller modified coding	10% ASK NRZ Coding	10% or 100% ASK Manchester Coding	8 – 30% ASK Manchester Coding
Tag to Reader	Subcarrier fc/16 OOK Manchester	Subcarrier fc/16 BPSK NRZ-L	Single or Double Subcarrier fc/32 or fc/28 Manchester	>12% ASK load modulation Manchester Coding

# ST NFC Sensor TAG



# ST NFC Sensor TAG

NFCSensorTAG is a **NFC enabled sensor node** that can sense temperature, humidity, pressure, vibration, motion and **transmit the data when triggered by an NFC reader**. It is a **reference platform** that can be scaled down/up based on requirement of final applications and use cases.

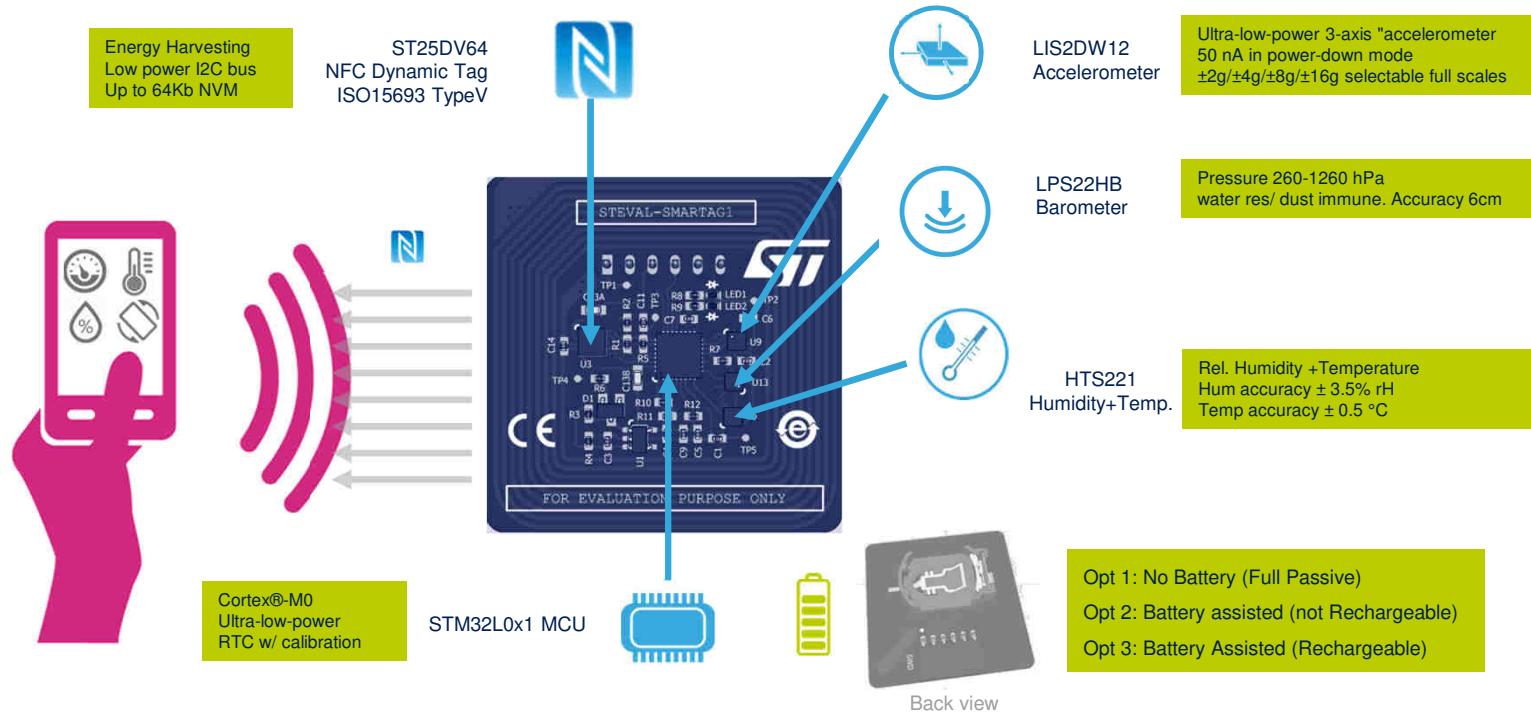
An **alternative way of connectivity** for applications that:

- Are extremely **low POWER** (also full passive) and **low COST**;
- Require small real estate (**reduced BOM**) and **fast implementation**;
- Do **NOT** require **Real-Time Remote** monitoring (Near Field Communication)

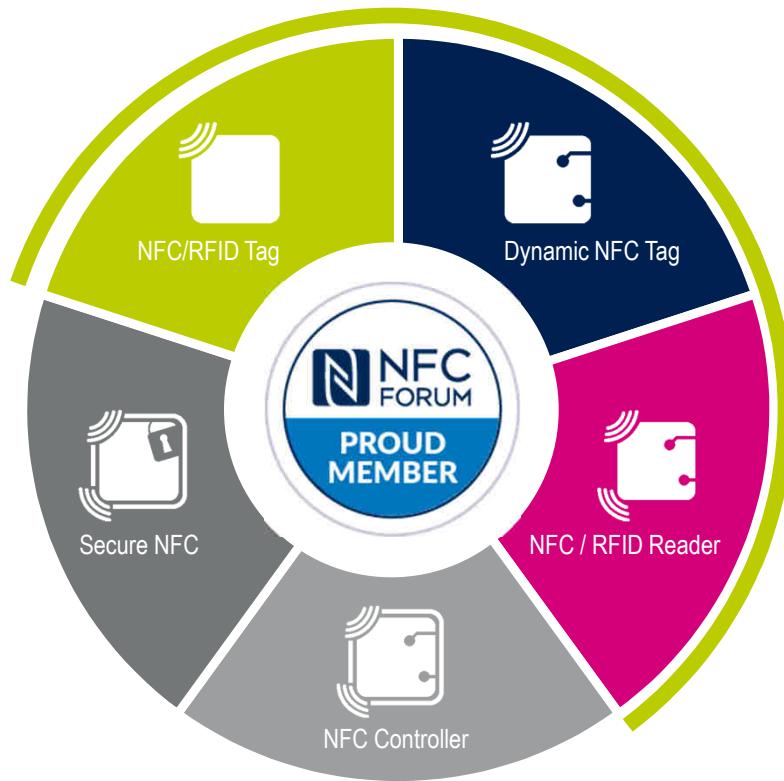
# Leveraging ST Technology



# What's on the Board

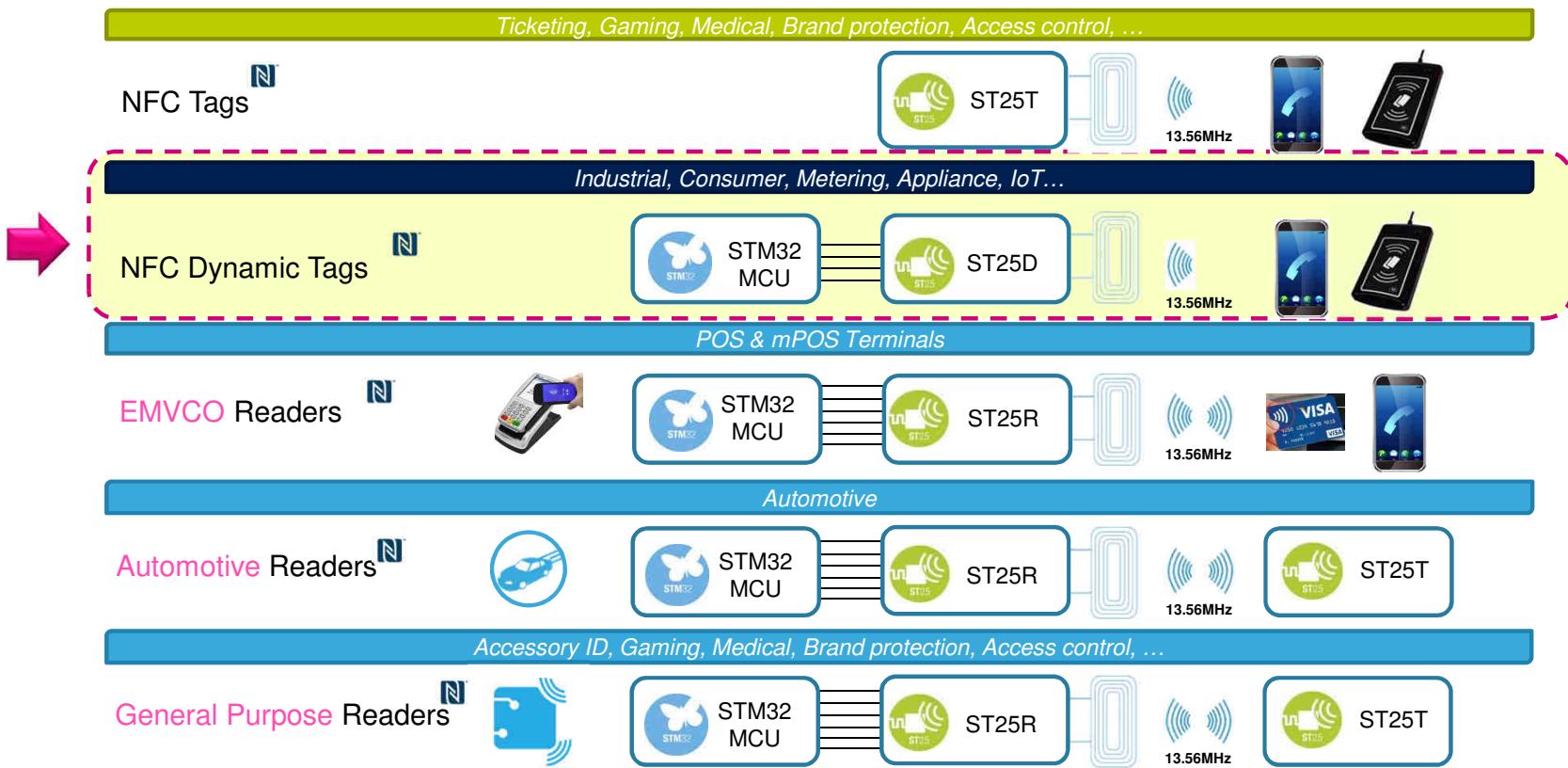


# ST25 NFC Value Proposition



**NFC / RFID Tags and Readers**  
Simply More Connected  
Covering all NFC application needs and  
leveraging a rich ecosystem

# ST25 Product Lines



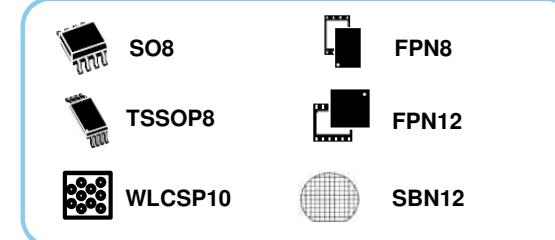
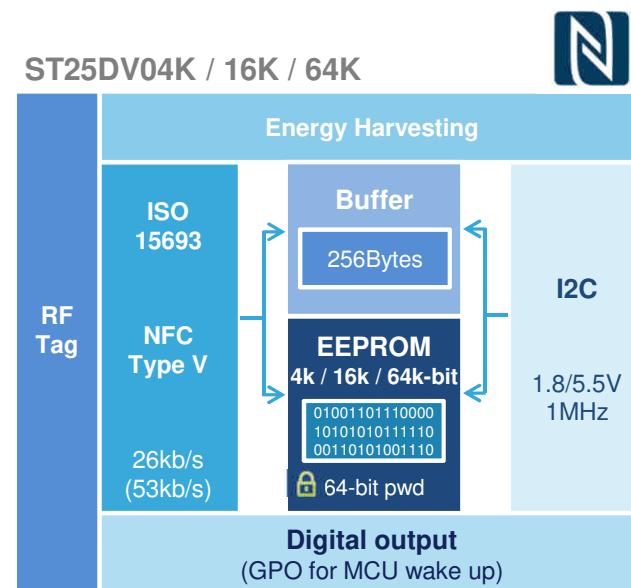
# ST25 Product Portfolio

Tags			Dynamic tags			HF Readers			UHF Readers
ST25TA	ST25TB	ST25TV	M24SR	M24LR	ST25DV	CR95HF ST95HF	ST25R3910	ST25R3911B - ST25R3915	ST25RU3993
ISO14443-A 106kb/s NFC type 4	ISO14443-B 106kb/s	ISO15693 up to 53kb/s NFC type 5	ISO14443-A 106kb/s NFC type 4	ISO15693 up to 53kb/s	ISO15693 up to 53kb/s NFC type 5	ISO14443-A/B ISO15693	ISO14443-A/B ISO15693 FeliCa	ISO14443-A/B FeliCa ISO15693 ISO18092	ISO18000 6c & b Gen2 Protocol
EEPROM 512b-64Kbit 200-year 1Mcycles	EEPROM 512b-4Kbit 40-year 1Mcycles	EEPROM 512b-64Kbit 60-year 100kcycles	EEPROM 2Kbit to 64Kbit 200-year 1Mcycles	EEPROM 4Kbit to 64Kbit 40-year 1Mcycles	256Bytes buffer EEPROM 4Kbit to 64Kbit 40-year 1Mcycles	Reader / Writer Card Emulation	Reader / Writer Limited P2P	Reader / Writer P2P EMVco & PBOC AECQ100	Reader / Writer -90dBm sensitivity Internal VCO
TruST25 128bit password 20bit counter UID Field Detect	32bit counters Lock OTP bits UID	TruST25 64bit password 16-bit counter UID Tamper detect	128bit password RF disable Field Detect	32bit password E-harvesting Field Detect	Fast transfer mode 64bit password E-harvesting Field Detect	-	AAT	VHBR AAT Multi Antenna Dynamic output power	Dense Reader Mode Linear RSSI Automatic PSRR Auto ACK
			I2C 1MHz 2.7V - 5.5V	I2C 400kHz 1.8V - 5.5V	I2C 1MHz 1.8V - 5.5V	SPI & UART 2Mbit/s 2.7V - 5.5V 230mW	SPI 6Mbit/s 2.4V - 3.6V 700mW max	SPI 6Mbit/s 2.4V - 5.5V 1 - 1.4W max	SPI 5Mbit/s 1.65V - 5.5V 0/20dBm Output



# Focus on ST25DV Type5 Dynamic Tag

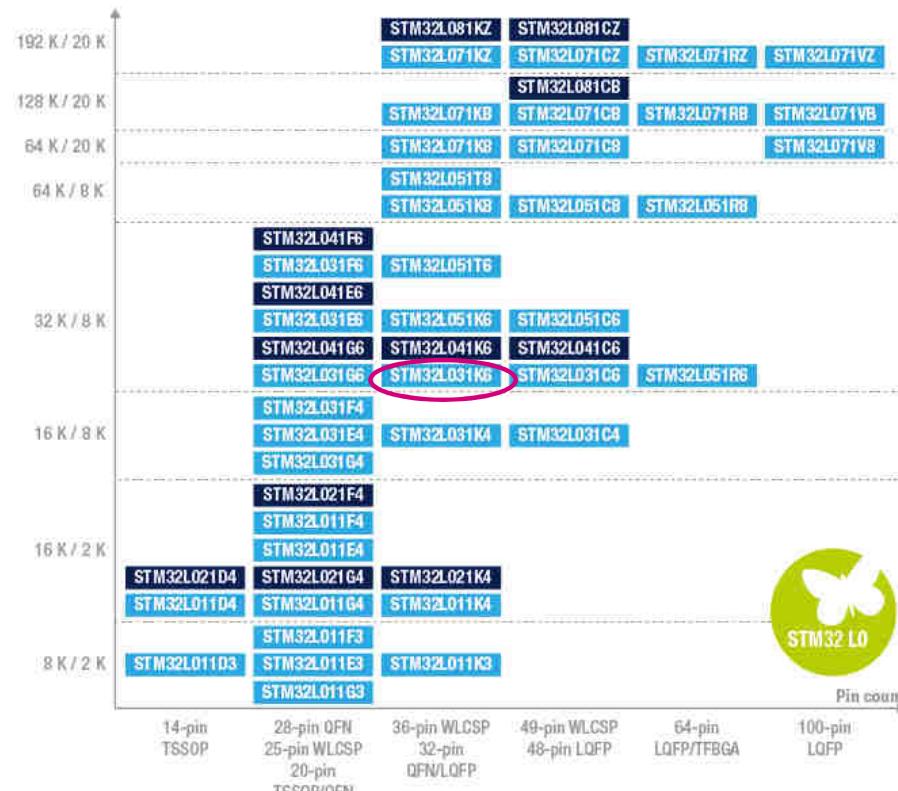
ST25DV series	
Contactless Interface	ISO15693 / NFC Forum Type 5
RF speed	up to 53kbps (26kbps std)
Single supply voltage	1.8V to 5.5V
Serial Interface	I2C @1MHz
Extra Features	GPO: 7 interrupts modes (OD or CMOS) Energy Harvesting Low Power Mode (<1uA stby)
Memory format & size	EEPROM data - 4 / 16 / 64-kbit
Data retention	40-year at +55°C
Fast Transfer Mode	256 Bytes memory buffer
Erase/Write cycles	1M cycles
Data protection	Password 64-bit
Temperature range	-40°C to +85°C
Package	SO8/TSSOP8/FPN8-12/WLCSP/SBN12





# STM32L0 ULP MCU Series

Flash memory / RAM size (bytes)



Pin count

Legend:

■ With 128-bit AES Hardware Encryption

■ Without 128-bit AES Hardware Encryption



# STM32L031 ULP MCU

All the ingredients for your application

32bits CPU  
8KB SRAM  
1KB EEPROM

Control  
Connectivity  
Analog

Battery friendly  
250nA standby

Goes everywhere  
Down to 2 x 2,5 mm package



TSSOP20  
169 mils



UFQFPN28 4x4 mm  
UFQFPN32 5x5 mm



LQFP32/48  
7x7 mm  
WLCSP25  
2.097x2.493 mm



## STM32L031x6

### System

Power supply  
1.8 V regulator  
POR/PDR/PVD/BOR  
Xtal oscillators  
32 kHz + 1 to 32 MHz  
Internal RC oscillators  
38 kHz + 16 MHz  
PLL  
Internal multispeed  
ULP RC oscillator  
64 kHz to 4 MHz  
Clock control  
RTC/AWU  
SysTick timer  
2x watchdogs  
(independent and  
window)  
15/20/27/38 I/Os  
Cyclic redundancy  
check (CRC)  
Voltage scaling  
3 modes

Available from 16K to 32KB FLASH

ARM® Cortex®-M0+ CPU  
32 MHz

Nested Vector  
Interrupt  
Controller (NVIC)

Memory Protection  
Unit (MPU)

SW debug

AHB-Lite+ bus matrix

AHB-bus - I/O port Bus

Up to 7-channel DMA

### Analog

2x ultra-low-power  
comparators  
Temperature sensor  
1x 12-bit ADC SAR  
10 channels / 1 µs

32-Kbyte  
Flash memory

8-Kbyte SRAM

1-Kbyte EEPROM

20-bytes backup data

BOOT ROM

### Connectivity

1x SPI, 1x I²C  
1x USART  
LIN, smartcard, IrDA,  
modem control  
1x ULP UART

### Control

1x ultra-low-power  
16-bit timers  
3x 16-bit timer



Smart Things

# ST Leading Sensors

## ULTRA LOW POWER ACCELEROMETER & GYROSCOPE

LIS2DW12 / LSM6DSL

Low power & noise for  
UI, IoT, wearable

[http://www.st.com/content/st\\_com/en/products/mems-and-sensors/accelerometers/lis2dw12.html](http://www.st.com/content/st_com/en/products/mems-and-sensors/accelerometers/lis2dw12.html)



<http://www.st.com/en/mems-and-sensors/lsm6dsl.html>

## ACCELEROMETER & MAGNETOMETER COMPASS

LIS2MDL / LSM303AH

Accuracy, with  
pedometer (LSM303AH)

[http://www.st.com/content/st\\_com/en/products/mems-and-sensors/e-compasses/lis2mdl.html](http://www.st.com/content/st_com/en/products/mems-and-sensors/e-compasses/lis2mdl.html)



[http://www.st.com/content/st\\_com/en/products/mems-and-sensors/e-compasses/lsm303ah.html](http://www.st.com/content/st_com/en/products/mems-and-sensors/e-compasses/lsm303ah.html)

## HIGH ACCURACY PRESSURE SENSOR

LPS22HB / LPS33HW

Compact, low power,  
water resistant

[http://www.st.com/content/st\\_com/en/products/mems-and-sensors/pressure-sensors/lps22hb.html](http://www.st.com/content/st_com/en/products/mems-and-sensors/pressure-sensors/lps22hb.html)



[http://www.st.com/content/st\\_com/en/products/mems-and-sensors/pressure-sensors/lps33hw.html](http://www.st.com/content/st_com/en/products/mems-and-sensors/pressure-sensors/lps33hw.html)

## COMBO HUMIDITY & TEMPERATURE

HTS221

High accuracy  
Humidity and Temp



<http://www.st.com/en/mems-and-sensors/hts221.html>

## ANALOG & DIGITAL MICROPHONES

MP23AB01DH /  
MP34DT05-A

Better sound quality

<http://www.st.com/en/audio-ics/mp23ab01dh.html>



<http://www.st.com/en/audio-ics/mp34dt05-a.html>

### ST Advantage:

- Flexibility Power Consumption vs. Noise
- Ecosystem (SW, libraries, ref design, Nucleo boards ...)

### ST Advantage:

- Power Consumption
- Thermal Stability
- Precision

### ST Advantage:

- LPS22HB: Stability over time
- LPS33HW: resistant to harsh environment (automotive gel, metal lid, ceramic substrate)

### ST Advantage:

- Low power consumption 2uA @ 1Hz
- -40C to +120C operation
- SPI/I2C host interface

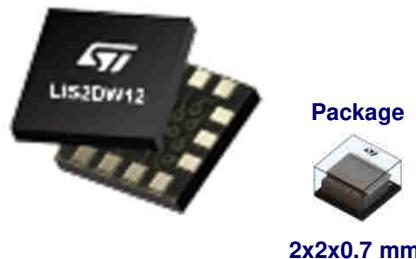
### ST Advantage:

- High performance 135dB AOP
- 65dB SNR

# ST Motion Sensors

## LIS2DW12

3-axis Accelerometer



Applications examples

- Wearables
- Gesture/Position detect
- IoT
- Vibration/Motion monitor
- Anti-theft

## Accelerometer

- $\pm 2/\pm 4/\pm 8/\pm 16$  g selectable Full Scales
- Noise/accuracy level flex
- 12 to 14bit res
- Low current consumption
- 380nA LP Mode and 40nA Standby

## Advantages

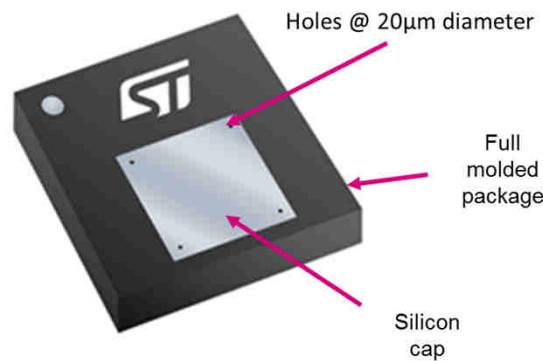
- High Accuracy for precise motion tracking
- Low Power Consumption for long battery life
- Embedding Digital Features, including Freefall, Wakeup, Orientation, Tap
- Easy integration, Software & Tools available



# Pressure/Altimeter Sensor

## LPS22HB

Pressure Barometer/Altimeter  
Sensor



### Applications examples

- Weather Stations
- Indoor and Outdoor Navigation

### Pressure

- Range 260-1260 hPa
- Relative accuracy of pressure measurement:
  - < 10 µbar
  - 6cm resolution

### Advantages

- World's Smallest pressure sensor in production
- Full-mold Package with silicon cap and four micro holes guaranteeing sensor moisture and dust resistance
- Very low Power consumption 3µA
- Embedded Temperature compensation



# Humidity & Temperature Sensor

## HTS221

Relative Humidity and Temperature  
combo



### Applications examples

- Environmental stations/monitoring
- HVAC Conditioning
- Medical equipment
- Home appliances
- White goods

### Humidity

- Range: 0%RH : 100%RH
- Accuracy:  $\pm 3.5\%$ RH

### Temperature

- Range: -40°C : +125°C
- Accuracy:  $\pm 0.5^\circ\text{C}$ , from 15°C to +40°C

### Advantages

- High Accuracy
- Low power consumption 2  $\mu\text{A}$
- Extended operative supply voltage

# Benefits

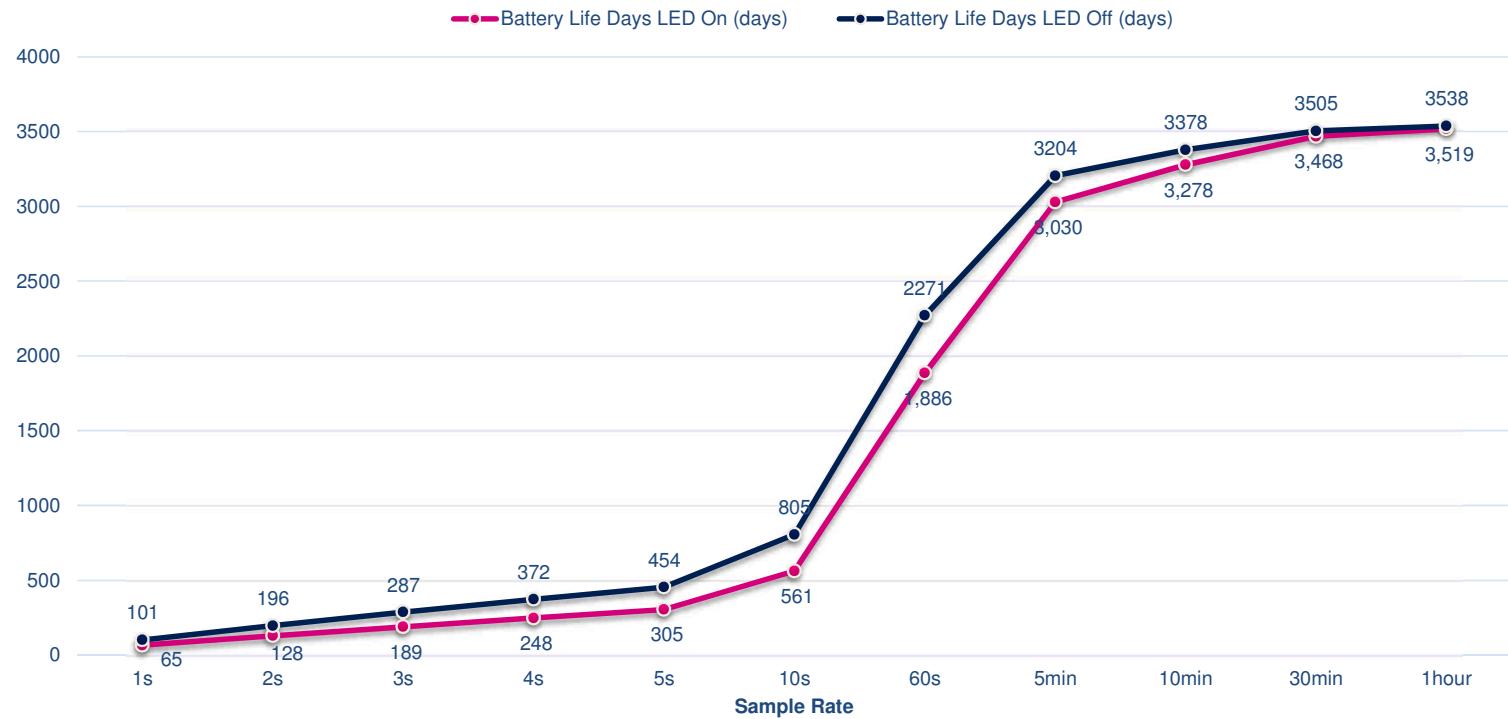
# ST NFC Sensor TAG Benefits

- **Low Cost** compared to other Wireless Technologies. Reduced BOM (doesn't require RF matching circuitry), Lower cost RF IC, simplified Layout (low cost PCB), low cost certification.
- **Low Power**: Ultra Low Power Connectivity, Computing and Sensing Technologies allow long battery life and battery less operations.
- **Easy to implement**: easy certifications (Passive RF)
- **Flexible**: scalable to the right configuration for many applications
- Enhanced **added values features** can be implemented through NFC TAG (ID, Error Logs, Security, FW upgrade...)

# Battery Life

# Battery Life Profile / Average

(Ref. CR2032 240mA/h batt.)



## Measuring Condition:

- All sensors are selected (active)
- All sensors values are stored written) to the NFC DTag each time (at selected sample rate)

# Addressing multiple Needs & Markets

# Asset Tracking

Monitoring goods from Manufacturing to the end User



**NFC** → Secure (Near Field). Interoperability with Smartphone and/or dedicated Reader → Cloud  
**Temperature, Humidity & Pressure** → Monitoring for Goods sensitive to environmental changes

**Motion** Sensors to detect **Vibrations**, Shocks, Freefall etc...which can damage/alter the goods

**Low Power** → Passive RF Tag (also battery less operation) / Long lasting battery life

**Low Cost** → Reduced BOM / Low cost connectivity.

**Scalable** → easy to add or remove sensors based on use case.

# Smart Buildings/Cities

- Environmental and Motion Data Monitoring (Smart Badge / Weather Stations...)
- Building/Constructions Structure Monitoring (Vibration alert, Pipe Leaks...)



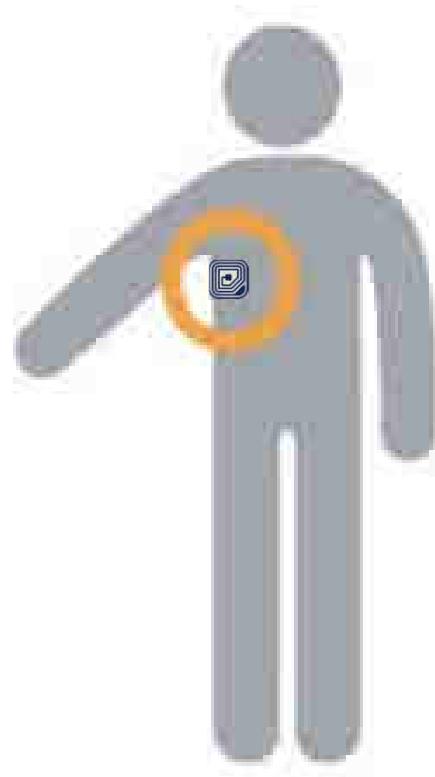
# Smart Packaging

- Connect the “Unconnected”
- Temperature, Humidity, Vibration
- Level of Content and Use information
- Good ID / info / Commerce / Experience



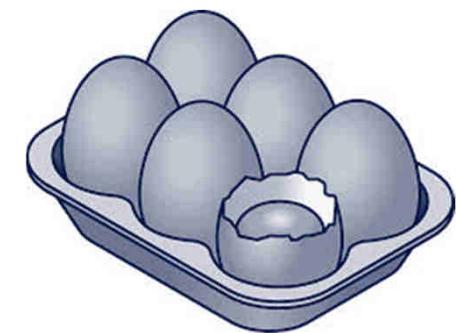
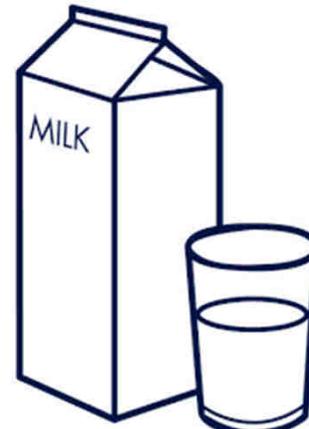
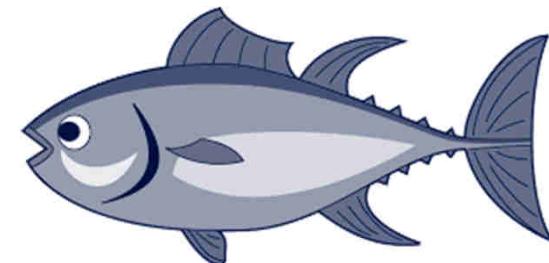
# Healthcare

- Temperature Patches (also disposable)
- Patient activity tracker (also disposable)



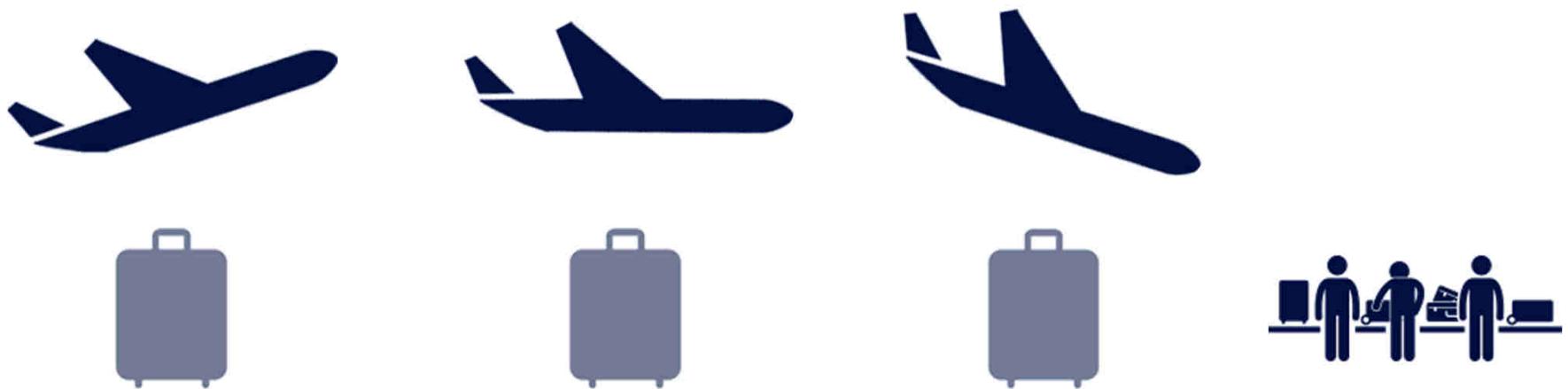
# Perishable Goods

- Condition of goods monitoring
- Shelf/Storage Life Calculation
- Exp. date alert
- Product ID/info/History



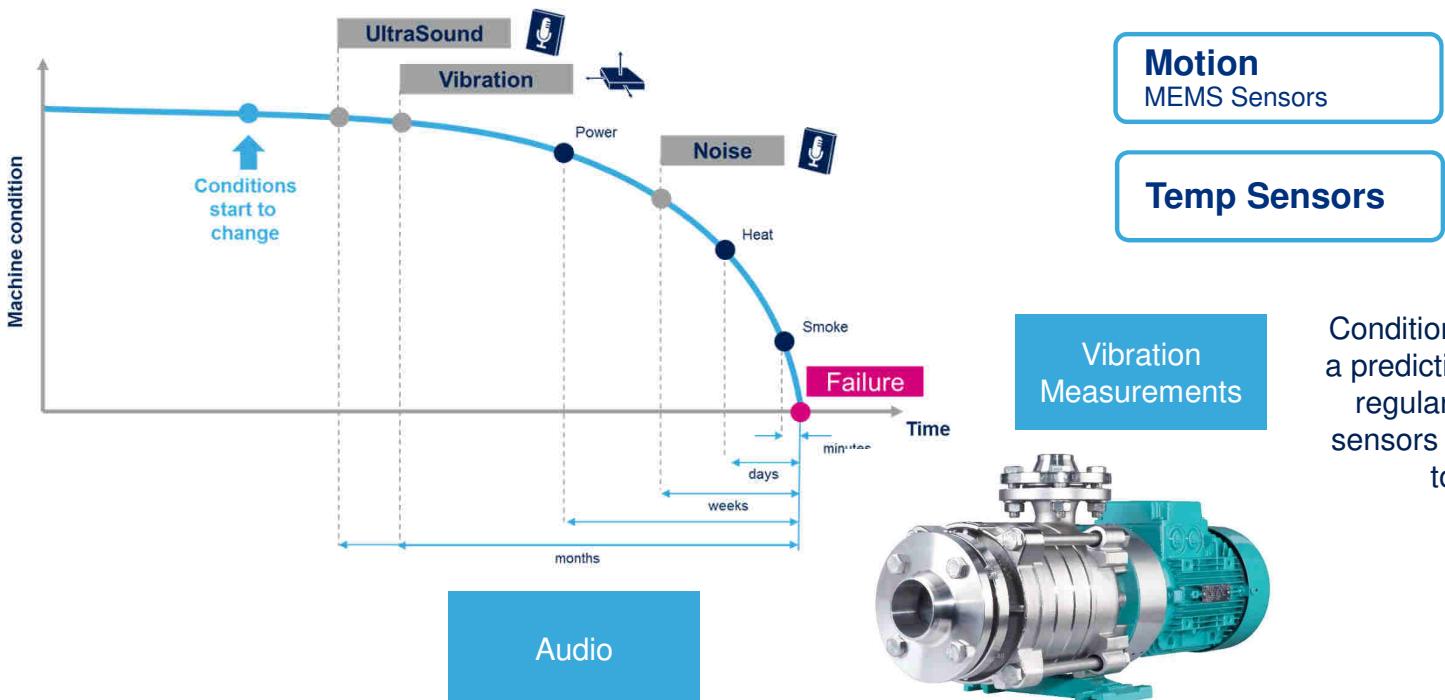
# Smart Baggage

- Baggage Check-in/out
- Baggage ID and Geo-Location
- In-flight condition monitoring





# Predictive Maintenance



Microphones to capture noise and for ultrasound emissions

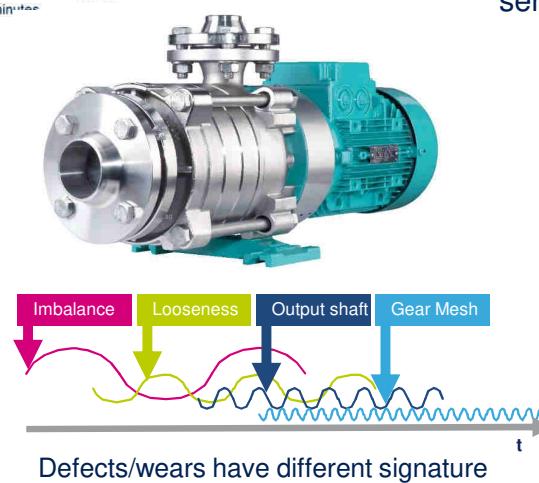
**Motion**  
MEMS Sensors

**Environmental**  
Humidity and Pressure

**Temp Sensors**

**Microphone**  
For Audio and Ultrasound

Condition monitoring enables to implement a predictive maintenance strategy by taking regular **vibration measurements** from sensors and comparing them to a baseline to detect health degradation



# Use Case Example - Summary

## Supply Chain & QC

- from manufacturing to end user.



## Smart building/home/city:

- Environmental and Motion Data
- Building/Constructions Structure



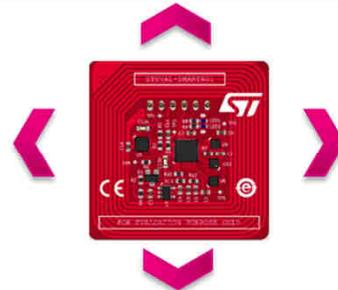
## Agriculture & Planting

Humidity, Temperature, light...



## Retail/ Fitness

Smart Clothing / Patches: Activity, Temperature, Sweat, mktg...



## Healthcare:

Medical patches & loggers: Patient temperature & activity monitoring



## Animal tracking

Activity, Temperature, big data monitoring production, health...



## Perishable Goods:

Exp. date alert, Shelf/Storage Life Calculation/Condition of goods



## Smart Packaging:

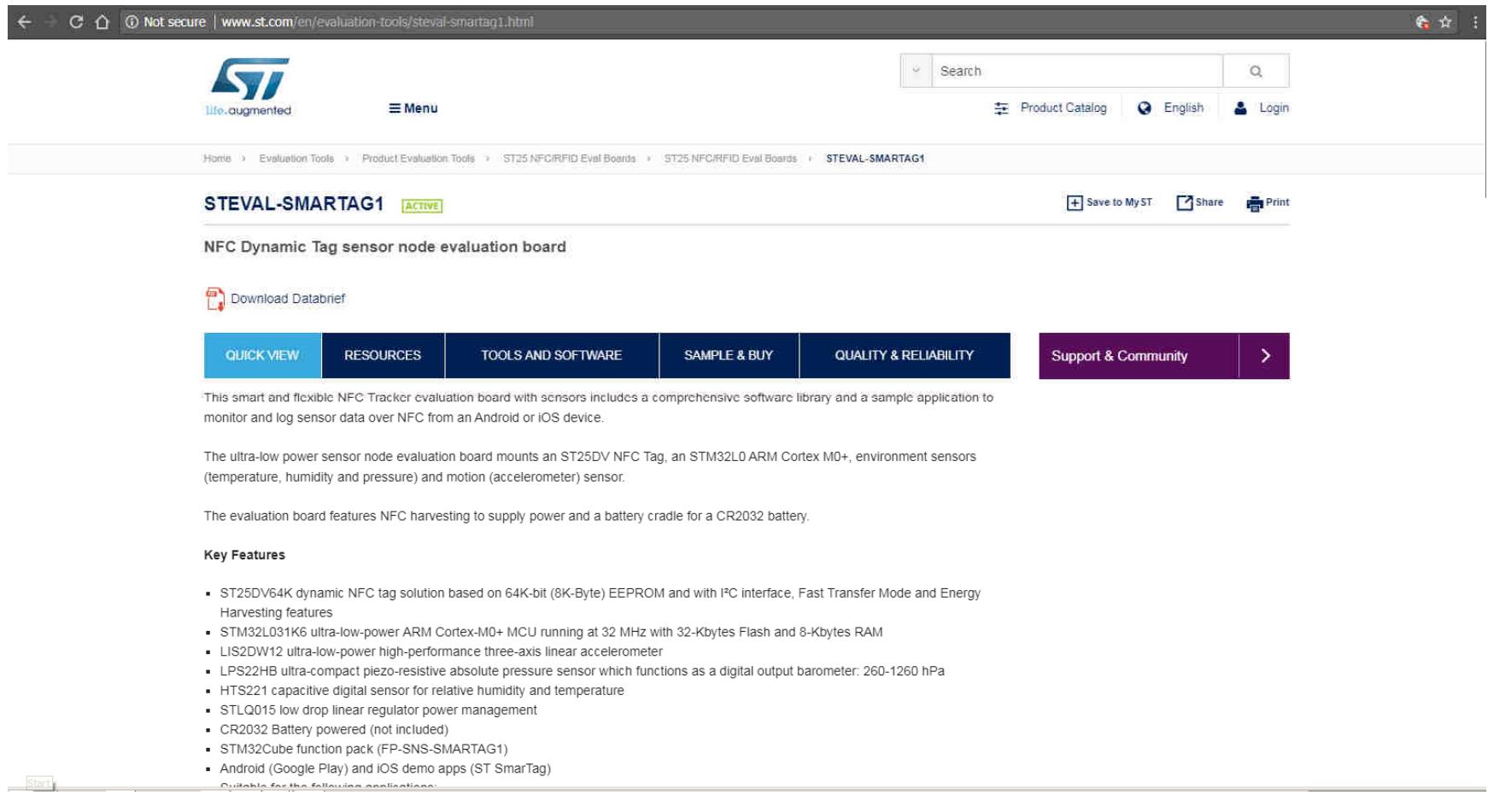
Temperature, Vibration Use, ID, . . .



And much more...



# www.st.com/nfcSENSORTAG



The screenshot shows a web browser displaying the STMicroelectronics website for the STEVAL-SMARTAG1 evaluation board. The page includes the ST logo, a search bar, and navigation links for Product Catalog, English, and Login. The main content area shows the product title 'STEVAL-SMARTAG1' (ACTIVE), a 'Download Databrief' button, and a menu bar with 'QUICK VIEW', 'RESOURCES', 'TOOLS AND SOFTWARE', 'SAMPLE & BUY', and 'QUALITY & RELIABILITY'. Below the menu, a brief description of the board's features is provided, followed by a 'Key Features' section listing various components and capabilities.

**STEVAL-SMARTAG1** ACTIVE

NFC Dynamic Tag sensor node evaluation board

[Download Databrief](#)

QUICK VIEW RESOURCES TOOLS AND SOFTWARE SAMPLE & BUY QUALITY & RELIABILITY

This smart and flexible NFC Tracker evaluation board with sensors includes a comprehensive software library and a sample application to monitor and log sensor data over NFC from an Android or iOS device.

The ultra-low power sensor node evaluation board mounts an ST25DV NFC Tag, an STM32L0 ARM Cortex M0+, environment sensors (temperature, humidity and pressure) and motion (accelerometer) sensor.

The evaluation board features NFC harvesting to supply power and a battery cradle for a CR2032 battery.

**Key Features**

- ST25DV64K dynamic NFC tag solution based on 64K-bit (8K-Byte) EEPROM and with I<sup>2</sup>C interface, Fast Transfer Mode and Energy Harvesting features
- STM32L031K6 ultra-low-power ARM Cortex-M0+ MCU running at 32 MHz with 32-Kbytes Flash and 8-Kbytes RAM
- LIS2DW12 ultra-low-power high-performance three-axis linear accelerometer
- LPS22HB ultra-compact piezo-resistive absolute pressure sensor which functions as a digital output barometer: 260-1260 hPa
- HTS221 capacitive digital sensor for relative humidity and temperature
- STLQ015 low drop linear regulator power management
- CR2032 Battery powered (not included)
- STM32Cube function pack (FP-SNS-SMARTAG1)
- Android (Google Play) and iOS demo apps (ST SmarTag)

Suitable for the following applications:

**Start**

# Technical Documentation

## Databrief

### Technical Documentation

#### Product Specifications

Description	Version	Size
 DB3533: NFC Dynamic Tag sensor node evaluation board	2.0	626 KB

# Hardware Resources

Gerbers



BOM



SCHEMATIC

## Hardware Resources

### Board Manufacturing Specifications

Description	Version	Size
STEVAL-SMARTAG1 gerber files	1.0	70 KB

### Bill of Materials

Description	Version	Size
STEVAL-SMARTAG1 BOM	1.0	66 KB

### Schematic Pack

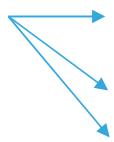
Description	Version	Size
STEVAL-SMARTAG1 schematic	1.0	90 KB

[www.st.com/nfcsensorstag](http://www.st.com/nfcsensorstag)

# Miscellaneous

47

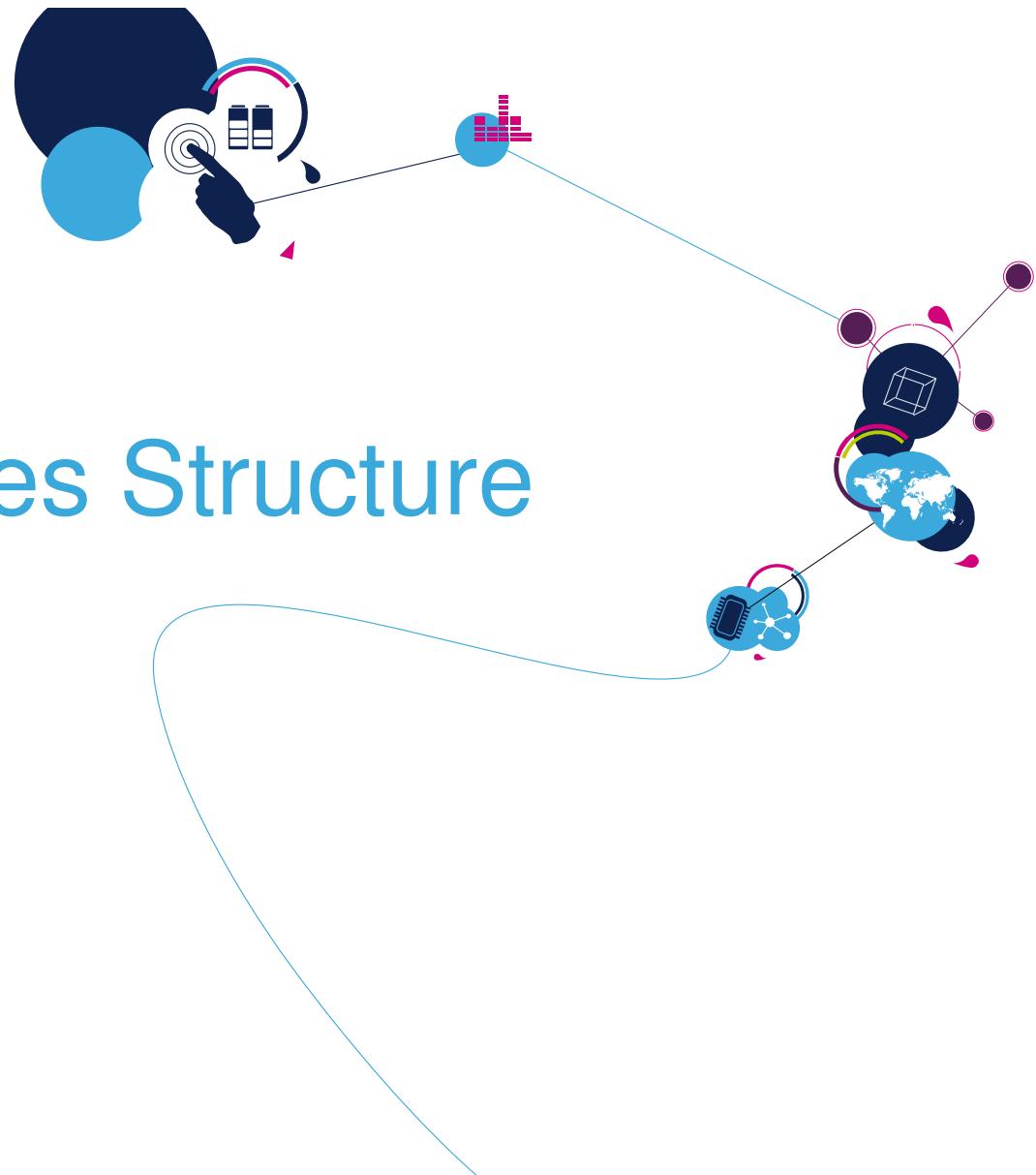
License agreements  
and certifications



Legal			
License Agreement			
Description	Version	Size	
Evaluation products license agreement	1.4	128 KB	
Open Reference Material License Agreement v5		42 KB	

[www.st.com/nfcsensorstag](http://www.st.com/nfcsensorstag)

# Firmware Packages Structure



# ST Ecosystem: STM32 ODE

## HARDWARE

**16**  
NUCLEO  
L0 to L4  
F0 to F7



**33**  
X-NUCLEO



STM32  
Nucleo  
Development  
Boards



STM32  
Nucleo  
Expansion  
Boards

[www.st.com/stm32ode](http://www.st.com/stm32ode)



## SOFTWARE

STM32Cube  
Development  
Software

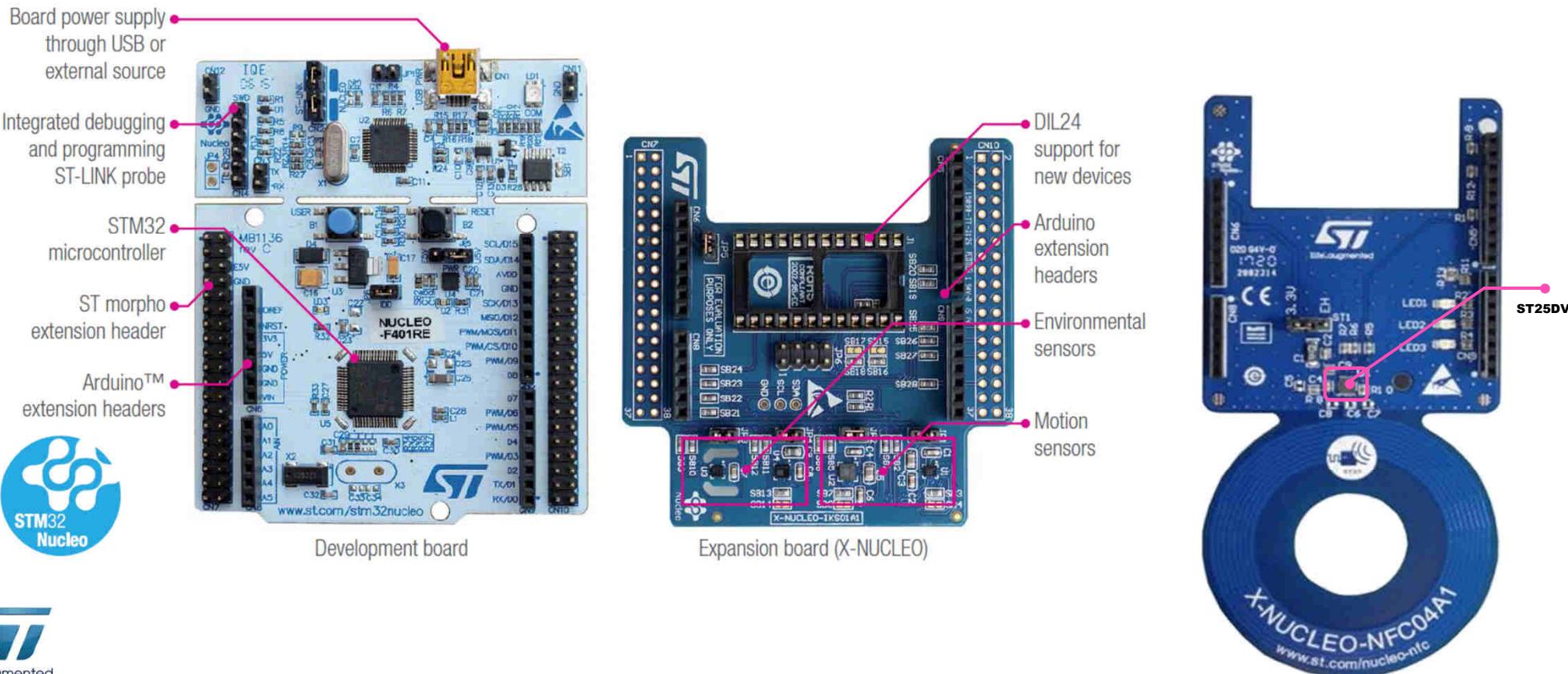


STM32Cube  
Expansion  
Software



Function  
Packs

# Nucleo / X-Nucleo



# Nucleo / X-Nucleo & NFCSensorTAG

- Modular development system
- Rich set of **firmware packages**

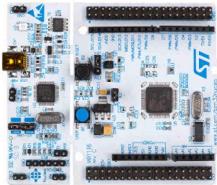
- Form-factor development system
- Same set of firmware packages & **more**



X-NUCLEO-NFC04A1

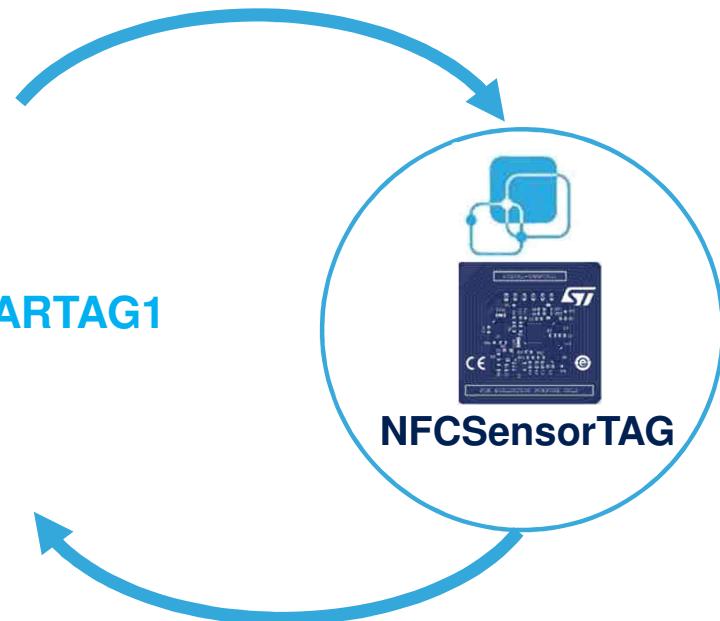


X-NUCLEO-IKS01A2

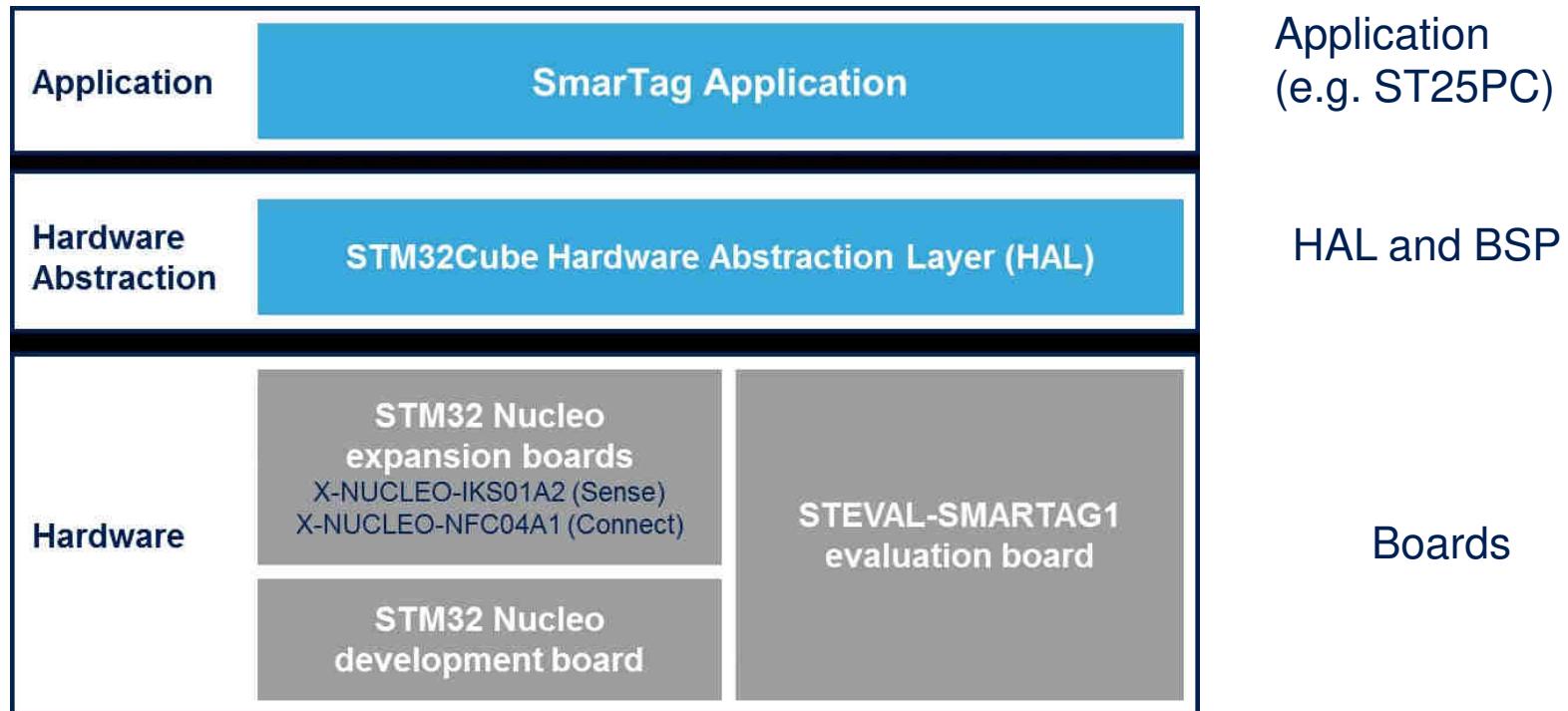


NUCLEO-L053RE

FP-SNS-SMARTAG1



# Hardware / Software Block Diagram



# Function Packages

## STM32ODE software package

Open Development Environment – src code

- **FP-SNS-SMARTAG1** Version 1.1.1

### Technical Documentation

Product Specifications			
Description	Version	Size	
 DB3553: STM32Cube function pack for IoT node with Dynamic NFC Tag, environmental and motion sensors	1.0	293 KB	

# Function Pack User Manual

User Manuals			
	Description	Version	Size
	UM2389: Getting started with the STM32Cube function pack for IoT node with Dynamic NFC Tag, environmental and motion sensors	1.0	2 MB

# Function Pack Presentations

## Presentations & Training Material

Presentations			
Description	Version	Size	
 FP-SNS-SMARTAG1 Quick Start Guide	1.1	2 MB	
 STM32 and STM8 embedded software solutions	5.0	3 MB	

# Miscellaneous

56

## Legal

License Agreement			
	Description	Version	Size
	SLA0055: SOFTWARE LICENSE AGREEMENT ("Agreement")	4.13	122 KB

# Function Pack and Nucleo Boards Stack

57

Name	Date modified	Type	Size
htmresc	6/12/2018 9:56 AM	File folder	
Documentation	6/12/2018 9:56 AM	File folder	
Drivers	6/12/2018 9:56 AM	File folder	
Projects	6/12/2018 9:56 AM	File folder	
package.xml	6/1/2018 2:06 AM	XML Document	1 KB
Release_Notes.html	6/1/2018 2:06 AM	Chrome HTML Docu...	61 KB

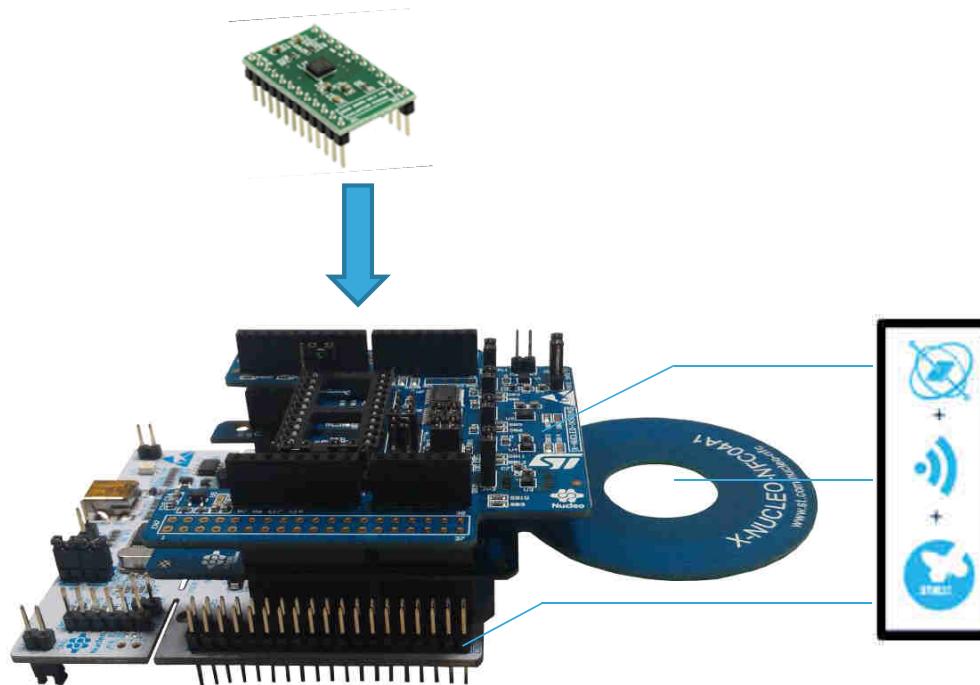
Name	Date modified	Type
STEVAL-SMARTAG1	6/12/2018 9:56 AM	File folder
STM32L053R8-Nucleo	6/12/2018 9:56 AM	File folder

Name	Date modified	Type	Size
Binary	6/12/2018 9:56 AM	File folder	
EWARM	6/12/2018 9:56 AM	File folder	
Inc	6/12/2018 9:56 AM	File folder	
MDK-ARM	6/12/2018 9:56 AM	File folder	
Src	6/12/2018 9:56 AM	File folder	
SW4STM32	6/12/2018 9:56 AM	File folder	
readme.txt	6/1/2018 6:13 AM	Text Document	5 KB

IAR project files  
Keil project files  
SystemWorkbench

# Benefits of using Nucleo Stack

58



- Flexibility and Scalability with STM32 Variants (L0,L1,L4,F4,F3 and more)
- Note: Functional pack example only include L053
- DIL24 socket on IKS01A2 for additional MEMs adapter and other sensors

# Function Pack and STEVAL-SMARTAG1

59

The screenshot shows the file structure of the STM32CubeFunctionPack\_SMArtAG1\_V1.1.1 package. It includes sub-folders for Documentation, Drivers, and Projects. The Projects folder contains two entries: STEVAL-SMARTAG1 (selected) and STM32L053R8-Nucleo. A pink oval highlights the STEVAL-SMARTAG1 project, which is expanded to show its contents: Examples, SmarTag1, Binary, EWARM, Inc, MDK-ARM, Src, SW4STM32, and readme.txt.

IAR project files

Keil project files

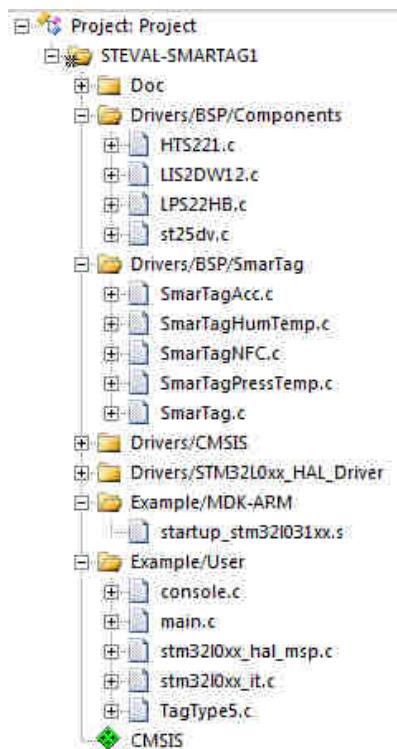
SystemWorkbench

Name	Date modified	Type
htmresc	6/12/2018 9:56 AM	File folder
Documentation	6/12/2018 9:56 AM	File folder
Drivers	6/12/2018 9:56 AM	File folder
Projects	6/12/2018 9:56 AM	File folder
package.xml	6/1/2018 2:06 AM	XML Document
Release_Notes.html	6/1/2018 2:06 AM	Chrome HTML Docu...

Name	Date modified	Type
STEVAL-SMARTAG1	6/12/2018 9:56 AM	File folder
STM32L053R8-Nucleo	6/12/2018 9:56 AM	File folder

Name	Date modified	Type
Binary	6/12/2018 9:56 AM	File folder
EWARM	6/12/2018 9:56 AM	File folder
Inc	6/12/2018 9:56 AM	File folder
MDK-ARM	6/12/2018 9:58 AM	File folder
Src	6/12/2018 9:56 AM	File folder
SW4STM32	6/12/2018 9:56 AM	File folder
readme.txt	6/1/2018 6:12 AM	Text Document

# Folder Structure (KEIL)



**BSP = Board Support Package**

- Components (typ. MEMS sensors)
- Boards (SensorTile, Nucleo, Nucleo-expansion)

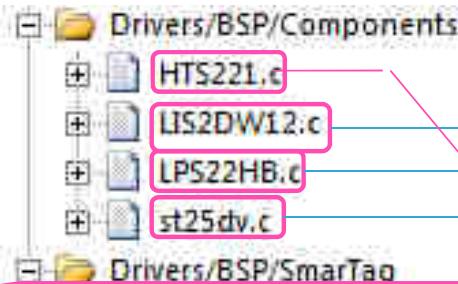
**CMSIS = Cortex Microcontroller Software Interface Standard**

- DSP library collection (fixed / float)

**HAL = Hardware Abstraction Layer**

- STM32 specific hardware drivers

**Main.c is in Example...\Src\**



Accelerometer

NFC Dynamic Tag

```

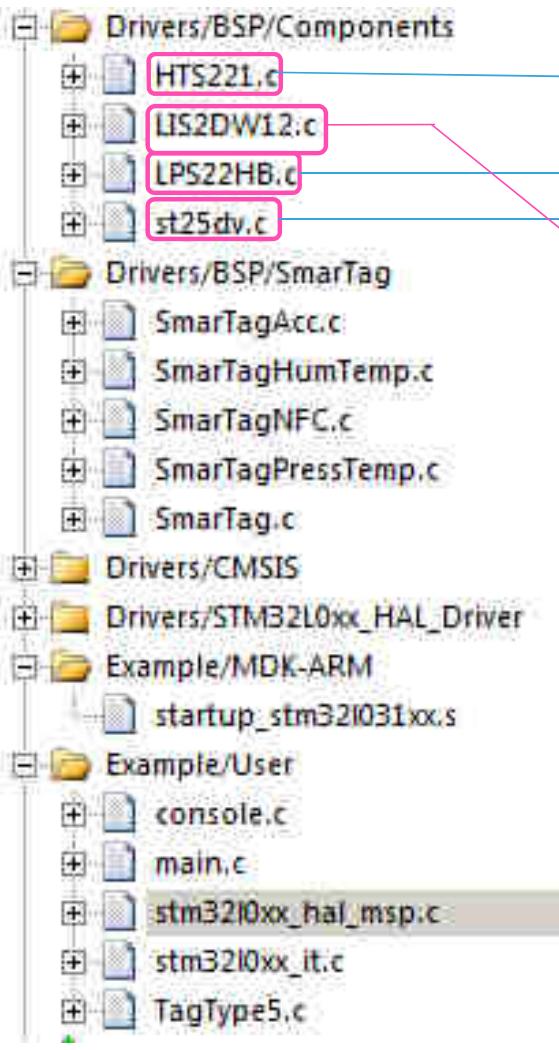
HTS221_Error_et HTS221_ReadReg( void *handle, uint8_t RegAddr, uint16_t NumByteToRead, uint8_t *Data );
HTS221_Error_et HTS221_WriteReg( void *handle, uint8_t RegAddr, uint16_t NumByteToWrite, uint8_t *Data );

HTS221_Error_et HTS221_Get_DeviceID(void *handle, uint8_t* deviceid);

HTS221_Error_et HTS221_Set_InitConfig(void *handle, HTS221_Init_st* pxInit);
HTS221_Error_et HTS221_Get_InitConfig(void *handle, HTS221_Init_st* pxInit);
HTS221_Error_et HTS221_DeInit(void *handle);
HTS221_Error_et HTS221_IsMeasurementCompleted(void *handle, HTS221_BitStatus_et* Is_Measurement_Completed);

HTS221_Error_et HTS221_Get_Measurement(void *handle, uint16_t* humidity, int16_t* temperature);
HTS221_Error_et HTS221_Get_RawMeasurement(void *handle, int16_t* humidity, int16_t* temperature);
HTS221_Error_et HTS221_Get_Humidity(void *handle, uint16_t* value);
HTS221_Error_et HTS221_Get_HumidityRaw(void *handle, int16_t* value);
HTS221_Error_et HTS221_Get_TemperatureRaw(void *handle, int16_t* value);
HTS221_Error_et HTS221_Get_Temperature(void *handle, int16_t* value);
HTS221_Error_et HTS221_Get_DataStatus(void *handle, HTS221_BitStatus_et* humidity, HTS221_BitStatus_et* temperature);
HTS221_Error_et HTS221_Activate(void *handle);
HTS221_Error_et HTS221_DeActivate(void *handle);

HTS221_Error_et HTS221_Set_AvgHT(void *handle, HTS221_Avgh_et avgh, HTS221_Avgt_et avg);
HTS221_Error_et HTS221_Set_AvgH(void *handle, HTS221_Avgh_et avgh);
HTS221_Error_et HTS221_Set_AvgT(void *handle, HTS221_Avgt_et avg);
HTS221_Error_et HTS221_Get_AvgHT(void *handle, HTS221_Avgh_et* avgh, HTS221_Avgt_et* avg);
HTS221_Error_et HTS221_Set_BduMode(void *handle, HTS221_State_et status);
HTS221_Error_et HTS221_Get_BduMode(void *handle, HTS221_State_et* status);
HTS221_Error_et HTS221_Set_PowerDownMode(void *handle, HTS221_BitStatus_et status);
HTS221_Error_et HTS221_Get_PowerDownMode(void *handle, HTS221_BitStatus_et* status);
HTS221_Error_et HTS221_Set_Odr(void *handle, HTS221_Odr_et odr);
HTS221_Error_et HTS221_Get_Odr(void *handle, HTS221_Odr_et* odr);
HTS221_Error_et HTS221_MemoryBoot(void *handle);
HTS221_Error_et HTS221_Set_HeaterState(void *handle, HTS221_State_et status);
HTS221_Error_et HTS221_Get_HeaterState(void *handle, HTS221_State_et* status);
HTS221_Error_et HTS221_StartOneShotMeasurement(void *handle);
HTS221_Error_et HTS221_Set_IrqActiveLevel(void *handle, HTS221_DrdyLevel_et status);
HTS221_Error_et HTS221_Get_IrqActiveLevel(void *handle, HTS221_DrdyLevel_et* status);
HTS221_Error_et HTS221_Set_IrqOutputType(void *handle, HTS221_OutputType_et value);
HTS221_Error_et HTS221_Get_IrqOutputType(void *handle, HTS221_OutputType_et* value);
HTS221_Error_et HTS221_Set_IrqEnable(void *handle, HTS221_State_et status);
HTS221_Error_et HTS221_Get_IrqEnable(void *handle, HTS221_State_et* status);
  
```



Humidity + Temperature

Pressure

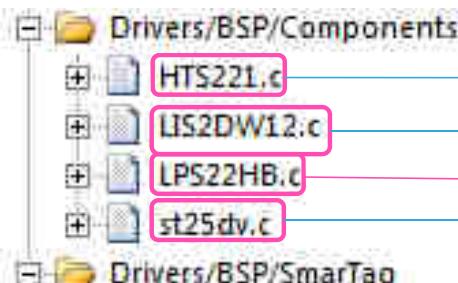
NFC Dynamic Tag

```
/*
 * Function Name  : status_t LIS2DW12_ACC_Get_Acceleration(u8_t *buff)
 * Description   : Read Acceleration output register
 * Input         : pointer to [u8_t]
 * Output        : Acceleration buffer u8_t
 * Return        : Status [MEMS_ERROR, MEMS_SUCCESS]
 */
status_t LIS2DW12_ACC_Get_Acceleration(void *handle, u8_t *buff)
{
    u8_t i, j, k;
    u8_t numberofByteForDimension;

    numberofByteForDimension=6/3;

    k=0;
    for (i=0; i<3;i++)
    {
        for (j=0; j<numberofByteForDimension;j++)
        {
            if( !LIS2DW12_ACC_ReadReg(handle, LIS2DW12_ACC_OUT_X_L+k, &buff[k], 1))
                return MEMS_ERROR;
            k++;
        }
    }

    return MEMS_SUCCESS;
}
```



Humidity + Temperature

Accelerometer

NFC Dynamic Tag

```

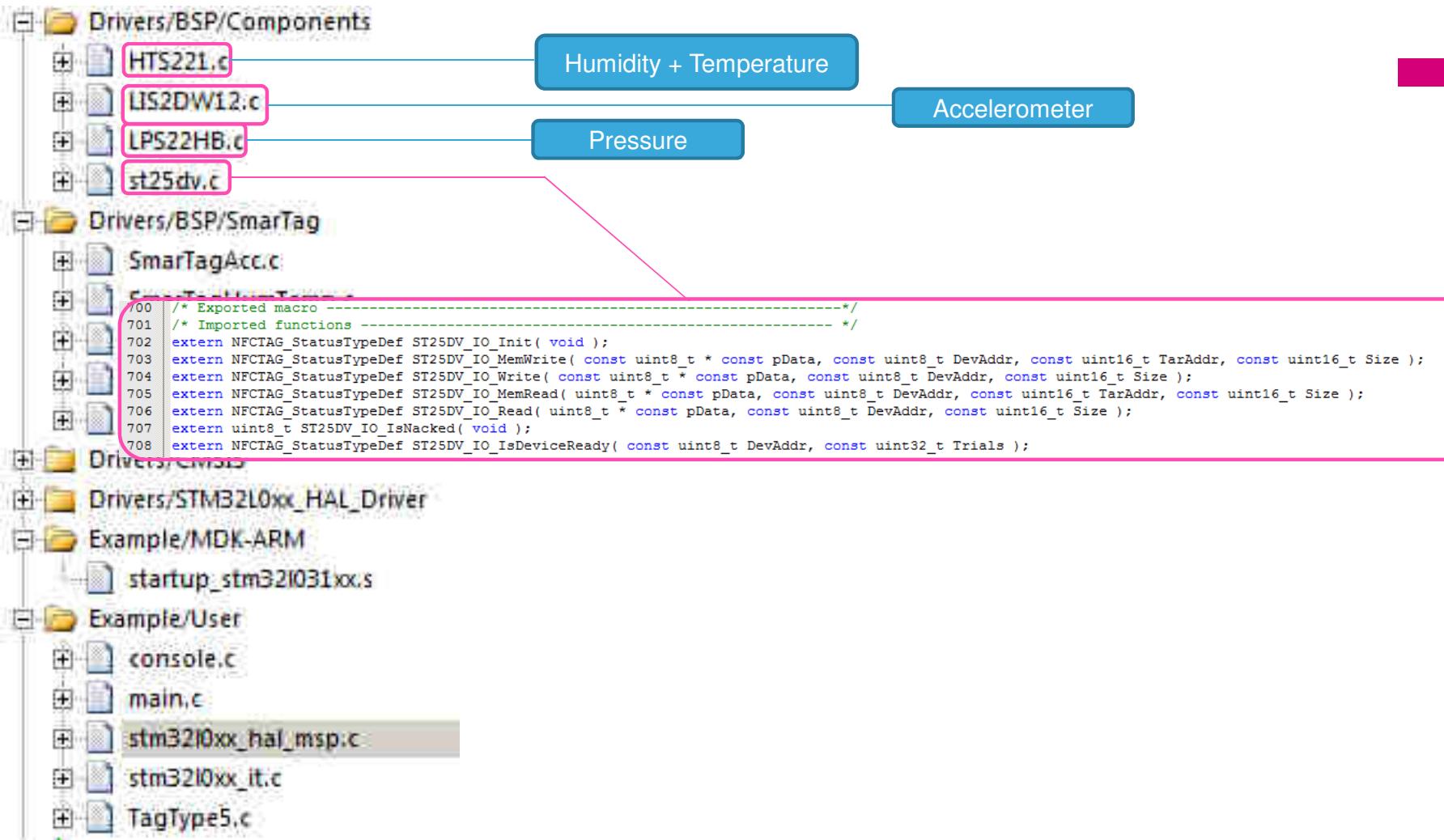
/**
 * @brief Get the LPS22HB raw pressure value
 * @param The buffer to empty with the pressure raw value
 * @retval Error Code [LPS22HB_ERROR, LPS22HB_OK]
 */
LPS22HB_Error_et LPS22HB_Get_RawPressure(void *handle, int32_t *raw_press);

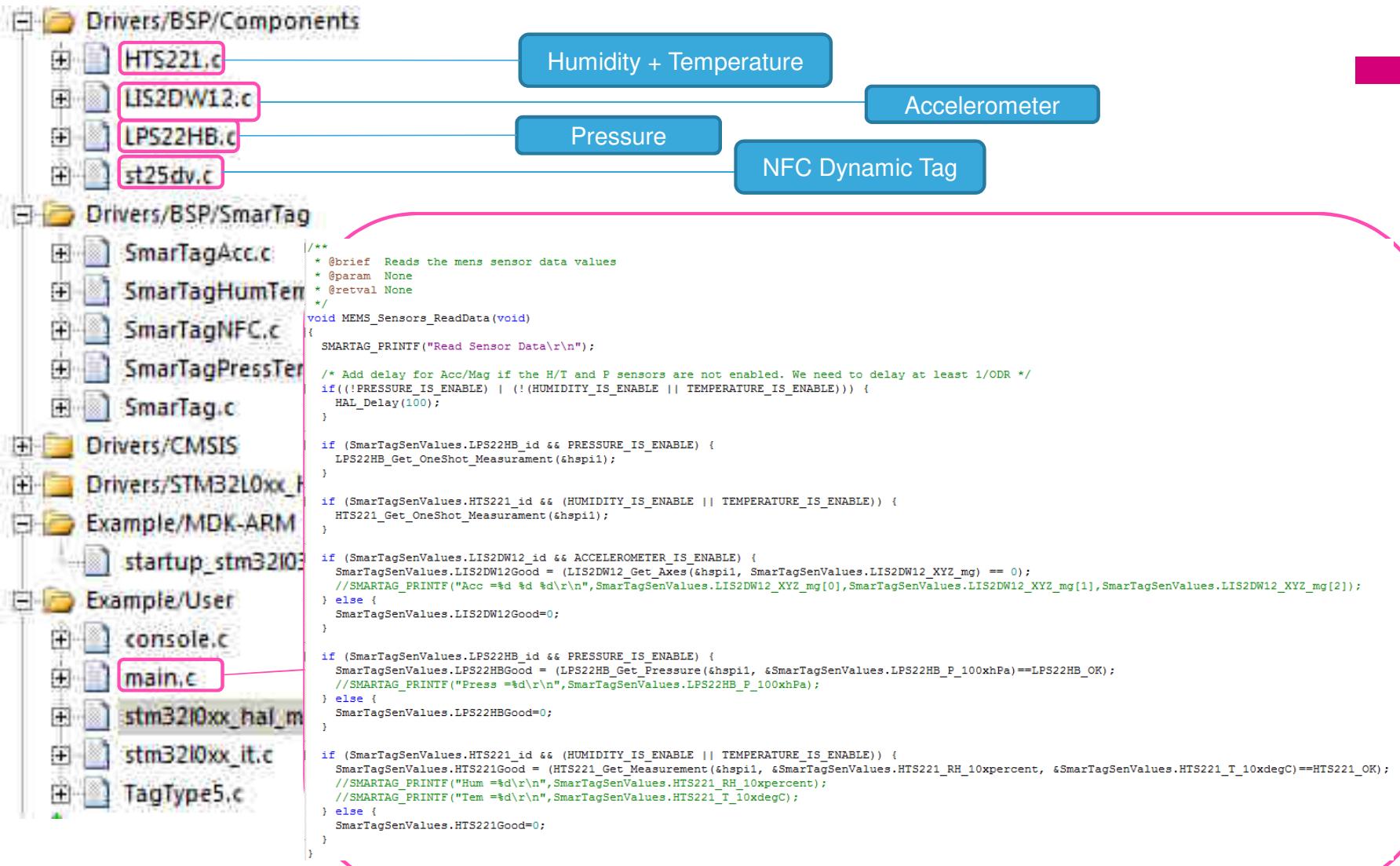
/**
 * @brief Get the LPS22HB Pressure value in hPa.
 * @param The buffer to empty with the pressure value that must be divided by 100 to get the value in hPa
 * @retval Error Code [LPS22HB_ERROR, LPS22HB_OK]
 */
LPS22HB_Error_et LPS22HB_Get_Pressure(void *handle, int32_t* Pout);

/**
 * @brief Read LPS22HB output register, and calculate the raw temperature.
 * @param The buffer to empty with the temperature raw value
 * @retval Error Code [LPS22HB_ERROR, LPS22HB_OK]
 */
LPS22HB_Error_et LPS22HB_Get_RawTemperature(void *handle, int16_t *raw_data);

/**
 * @brief Read the Temperature value in °C.
 * @param The buffer to empty with the temperature value that must be divided by 10 to get the value in ['C]
 * @retval Error Code [LPS22HB_ERROR, LPS22HB_OK]
 */
LPS22HB_Error_et LPS22HB_Get_Temperature(void *handle, int16_t* Tout);

/**
 * @brief Set One Shot bit to start a new conversion (ODR mode has to be 000)
 * @param void
 * @retval Error Code [LPS22HB_ERROR, LPS22HB_OK]
 */
LPS22HB_Error_et LPS22HB_StartOneShotMeasurement(void *handle);
  
```





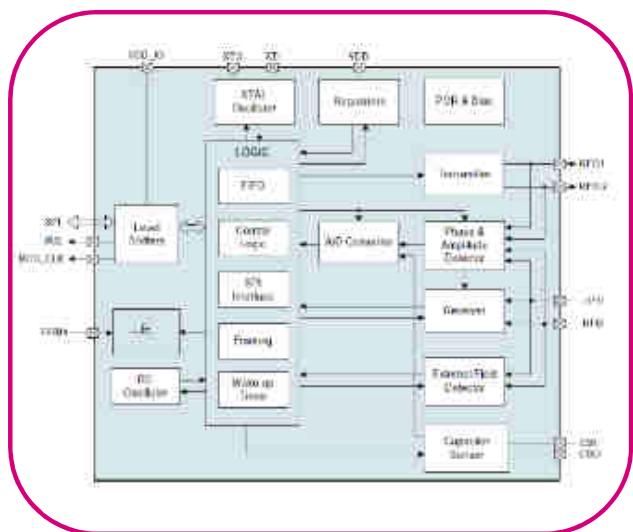
# NFC Reader Discovery Board



# ST25R3911B NFC / RFID Reader

*1.4W High Power reader solution*

ST25R3911B



- Use cases

- Ideal for Payment Applications
- Access Control, Gaming, eGovernment

- Key features

- NFC forum compatible (no passive target)
- **1.4W** output power at 5V
- Passes **EMVco & PBOC** certification without external power amplifier
- Automatic Antenna Tuning
- **VHBR** support up to 6.8Mb/s
- -40°C to 125°C temperature range

- Key benefits

- Low power operation & standby
- Works in challenging environment
- Enhanced fast transfer rate for Passport
- Easy-to-use evaluation / development kits
- Reference designs, application notes

# Output power & sensitivity

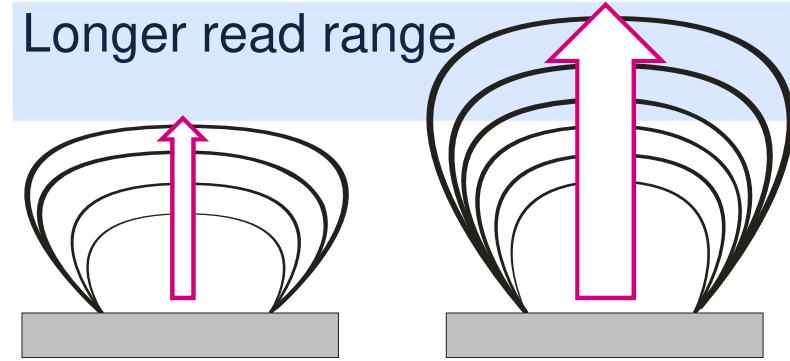
68

- Higher output power

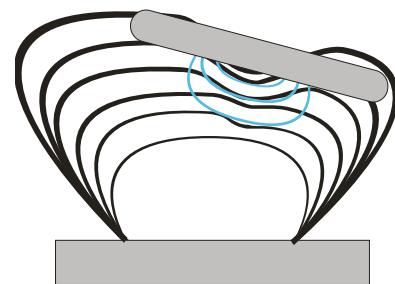
- The ST25R3911B includes low impedance drivers capable of generating  $>1$  W of output power
- EMVco certification easily possible without external boosters
- “Slave” devices like interface tags are able to harvest far more energy for batteryless devices
- The ST25R3911B is able to operate in metal encapsulation like doorlocks

- Higher sensitivity

- The ST25R3911B has a 10x higher sensitivity than any competitor.

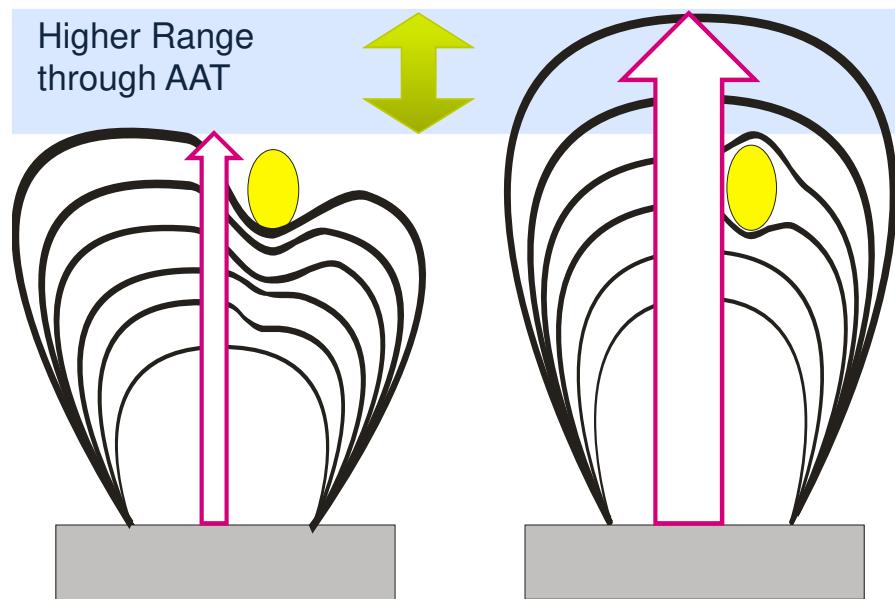


10x more sensitivity



# Automatic Antenna Tuning

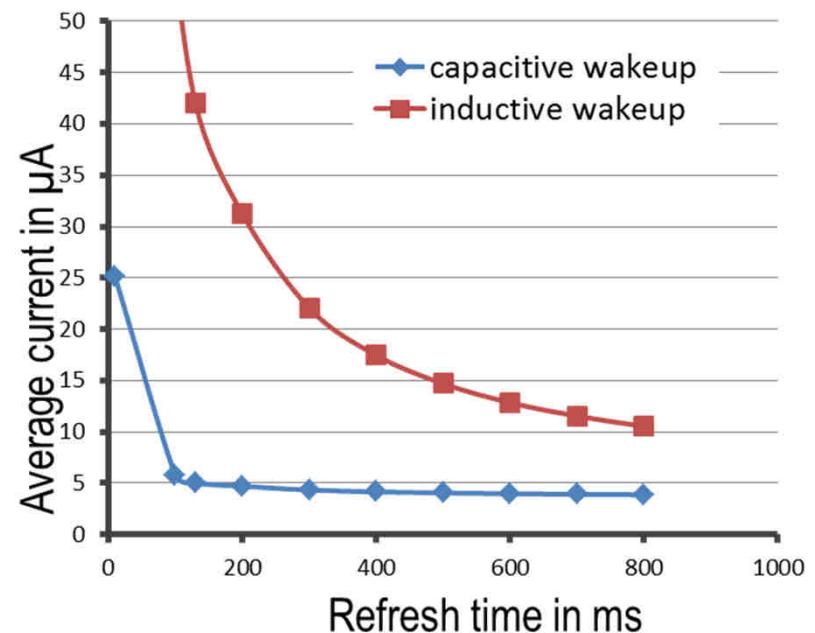
- Range & Field strength
  - AAT increases the range of an HF reader in bad environmental conditions and sustains maximum output power to the field with best efficiency
- Compensates for environment
  - Automatic antenna tuning analyses the phase shift of the antenna and retunes automatically
- Reduces production cost
  - The antenna can be tuned with an automatic procedure during production to fine adjust the design to different housings.

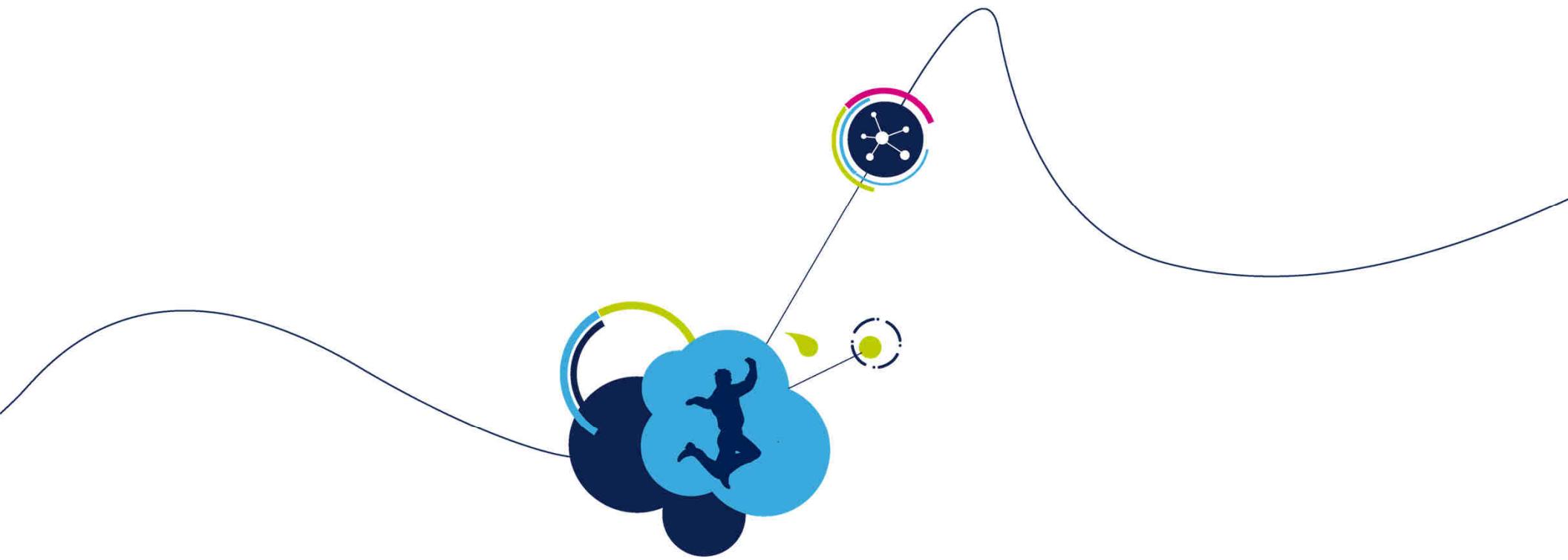


Multiple Tag placement  
Multiple tags in the field can be compensated to transfer a maximum of power for each.

# Low Power Wake-up

- Internal wakeup circuitry
  - The ST25R3911B includes a fully programmable wake-up scheme. All relevant parameters like cycle time & sensitivity can be programmed.
  - No MCU required to run the wake-up
  - Capacitive & Inductive wakeup can be combined for sophisticated wake-up scripts
- Capacitive wake-up
  - ST25R3911B can detect capacitive changes. E.g. the approach of a hand
- Inductive wake-up
  - The inductive wakeup is dedicated to detect approaching cards only

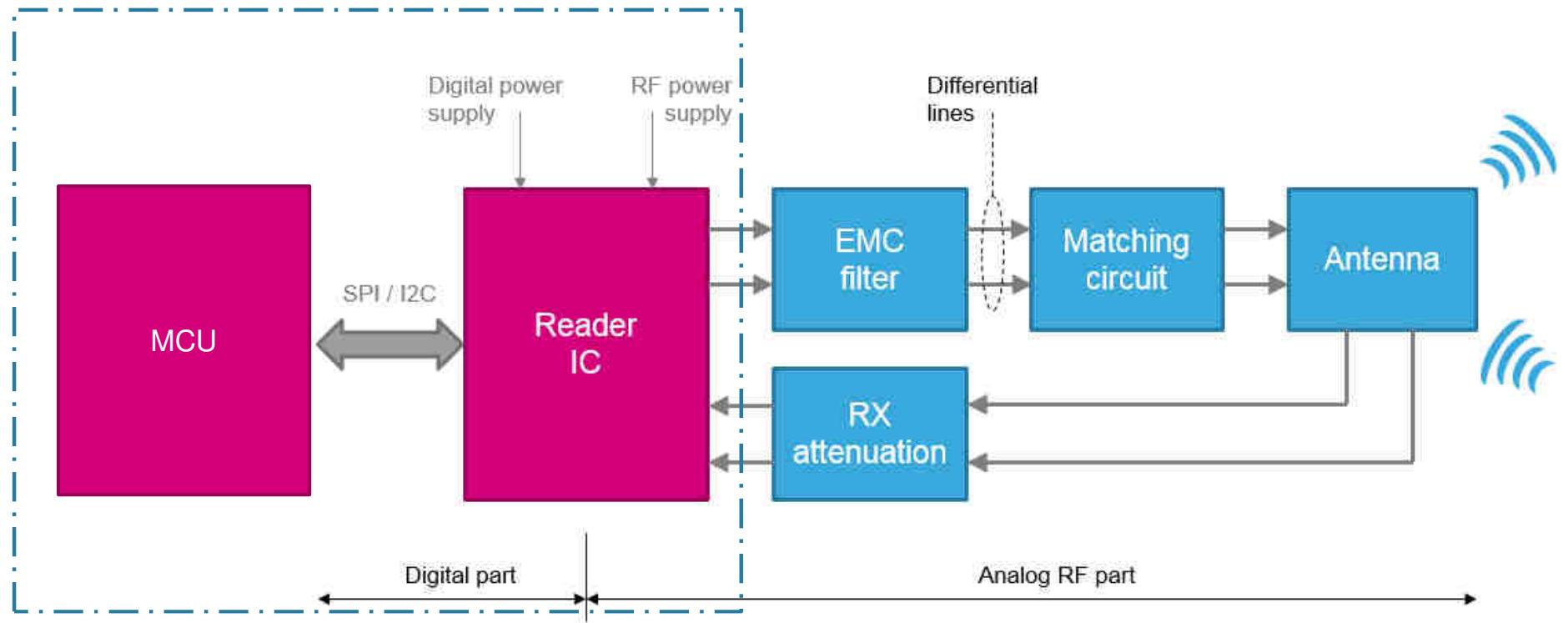




# Discovery System Block Diagram

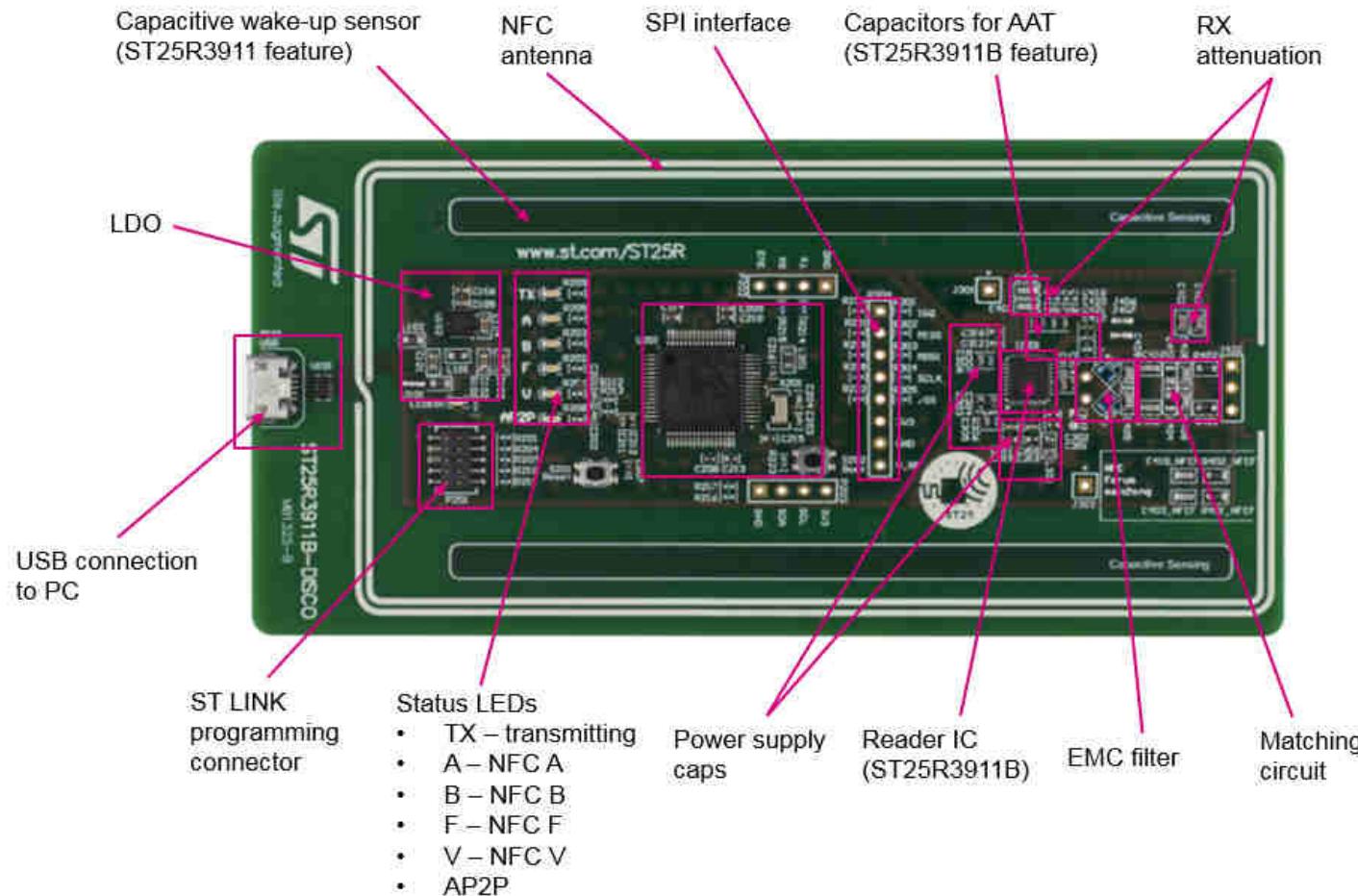
# Discovery System Block Diagram

72



# Discovery Board

73



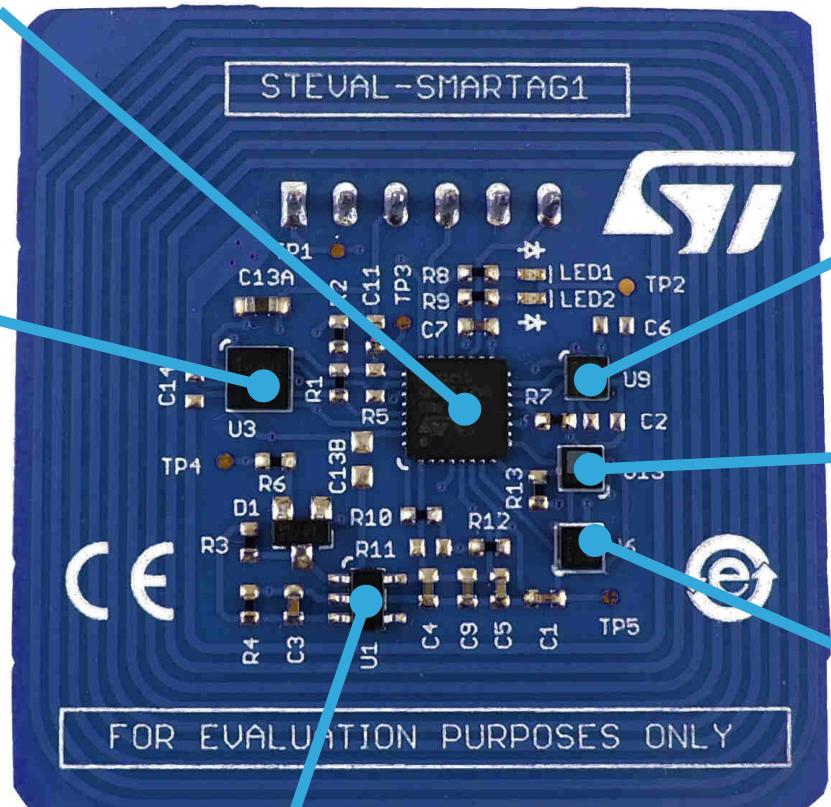
# Sensor Tag Hardware Features



# Take a good look at the sensor tag

**STM32L031K6U6**  
ARM Cortex-M0+ 32bit Microcontroller

**ST25DV64K-JFR6D3**  
64Kbit dynamic NFC/RFID tag



**STLQ015M18R**  
Low dropout linear regulator

**LIS2DW12**  
Ultra-low power 3-axis digital Accelerometer

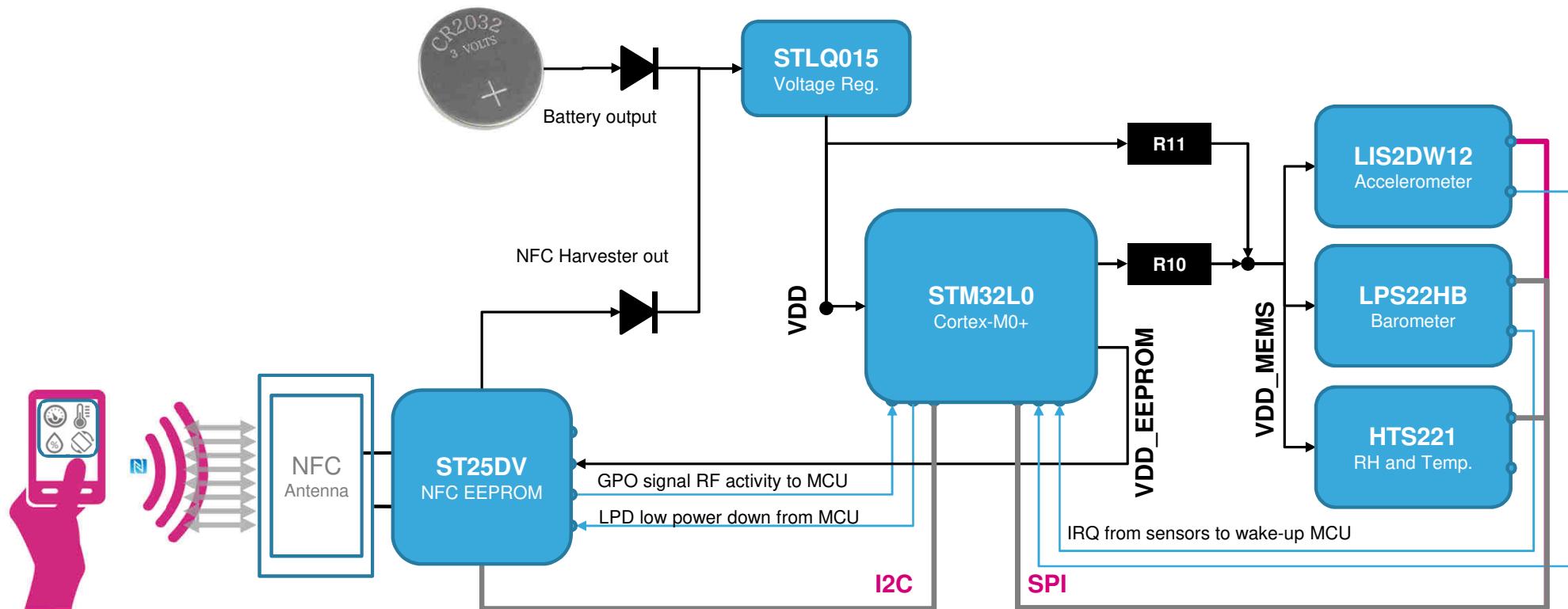
**LPS22HB**  
Low-power digital sensor for ambient Pressure

**HTS221**  
Capacitive digital sensor for Relative Humidity and Temperature



# SmartTag Block Diagram

76



# Power configuration

77

**R10 (MEMS power gating) and R11 (always-on MEMS)** are mutually exclusive.

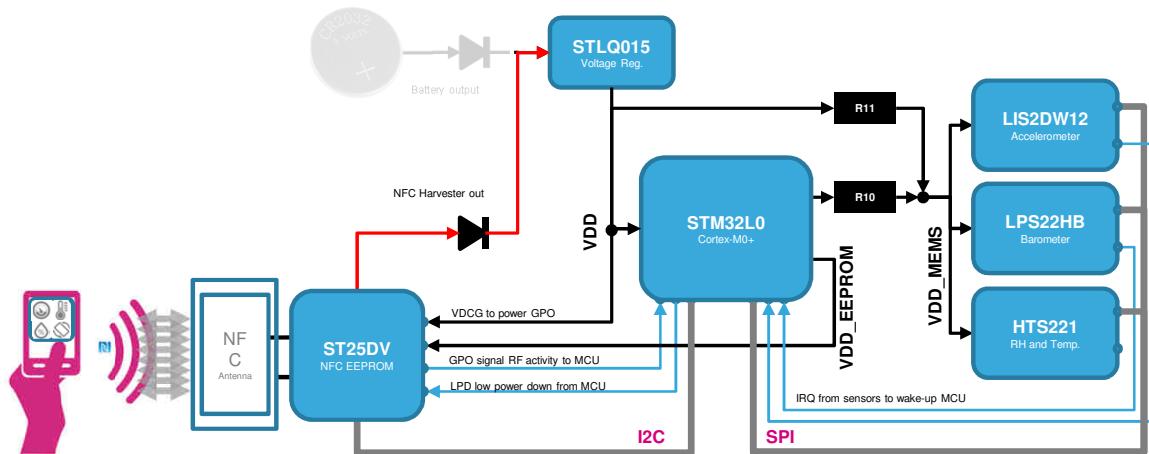
**Table 1. Solder bridge details for power Path configuration**

Solder bridge	Power source	Power sink
<b>R11</b> (enables always-on MEMS)	VDD (OUT of STLQ015)	VDD_MEMS
<b>R10</b> (enables MEMS power gating)	VDD_SENS (PB8 of STM32L0)	VDD_MEMS

# Battery-less Mode

78

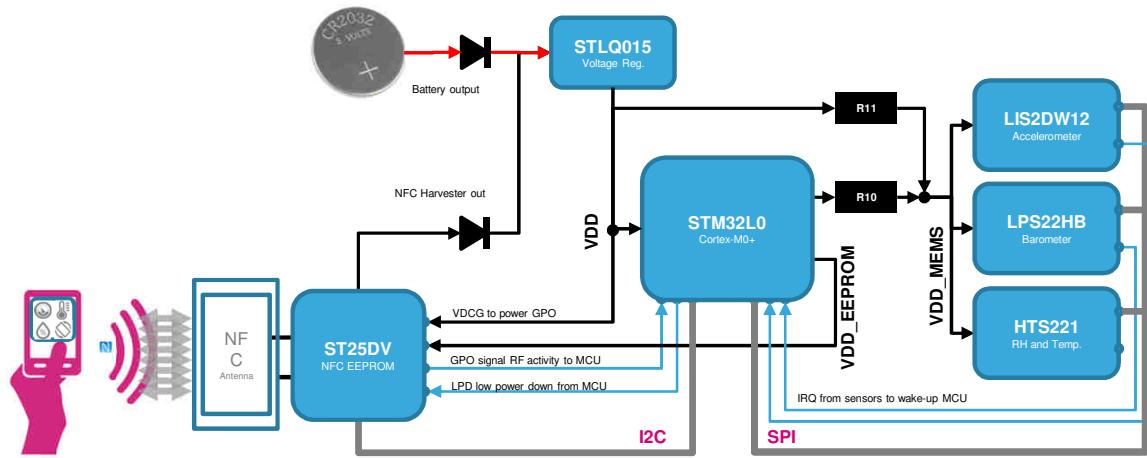
- No battery needed.
- The entire sensor tag is powered by NFC Reader RF field.
- One can perform ONE-SHOT mode



# Battery Mode

79

- Synchronous sampling and logging at interval that is controlled by the MCU Real Time Clock
- Asynchronous sampling and logging when an event detection logic embedded in MEMS sensors
- Only accelerometer and ambient pressure sensor can wake-up the microcontroller.



# Arbitration and Timing(I2C and RF)

80

- MCU communicates with ST25DV via I2C
- NFC Reader communicates with ST25DV via RF
- Communication is First Come First Serve
- When RF transaction is in progress, I2C commands are Nack (no acknowledgement)
- When I2C communication is in progress, any RF request receives no-response code of 0xFh

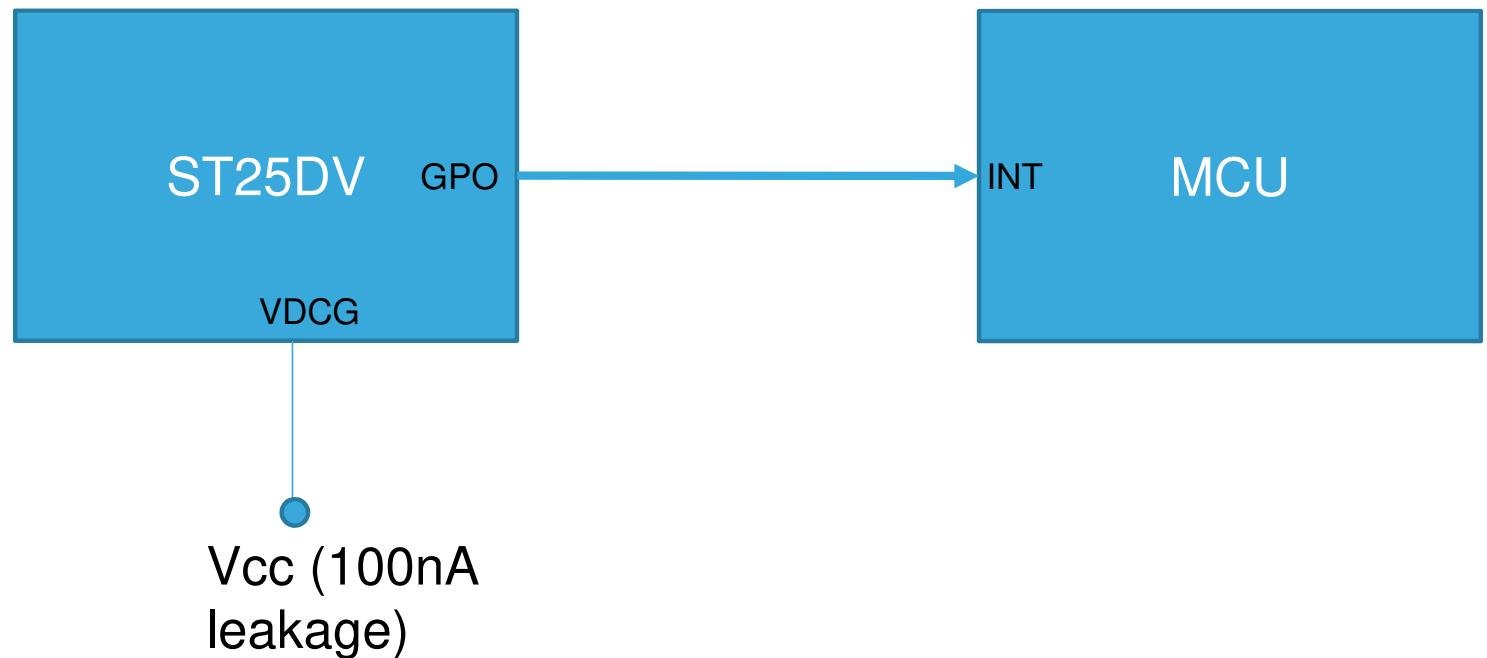
# Using ST25DV GPO for arbitration

RF_USER	<p>'ManageGPO' cmd must be sucessful</p>
RF_INTERRUPT	<p>'ManageGPO' cmd must be sucessful</p>
RF_ACTIVITY	<p>If ST25DV answers (not in quiet state), or addressed request not for this VICC</p>
FIELD_DETECT	<p>Only if ST25DV is powered through VCC</p>
RF_PUT_MSG RF_GET_MSG	<p>'Write MB Msg' and 'Read MB Msg' cmds must be sucessful</p>
RF_WRITE	<p>Write cmd must be sucessful Only for EEPROM writes</p>

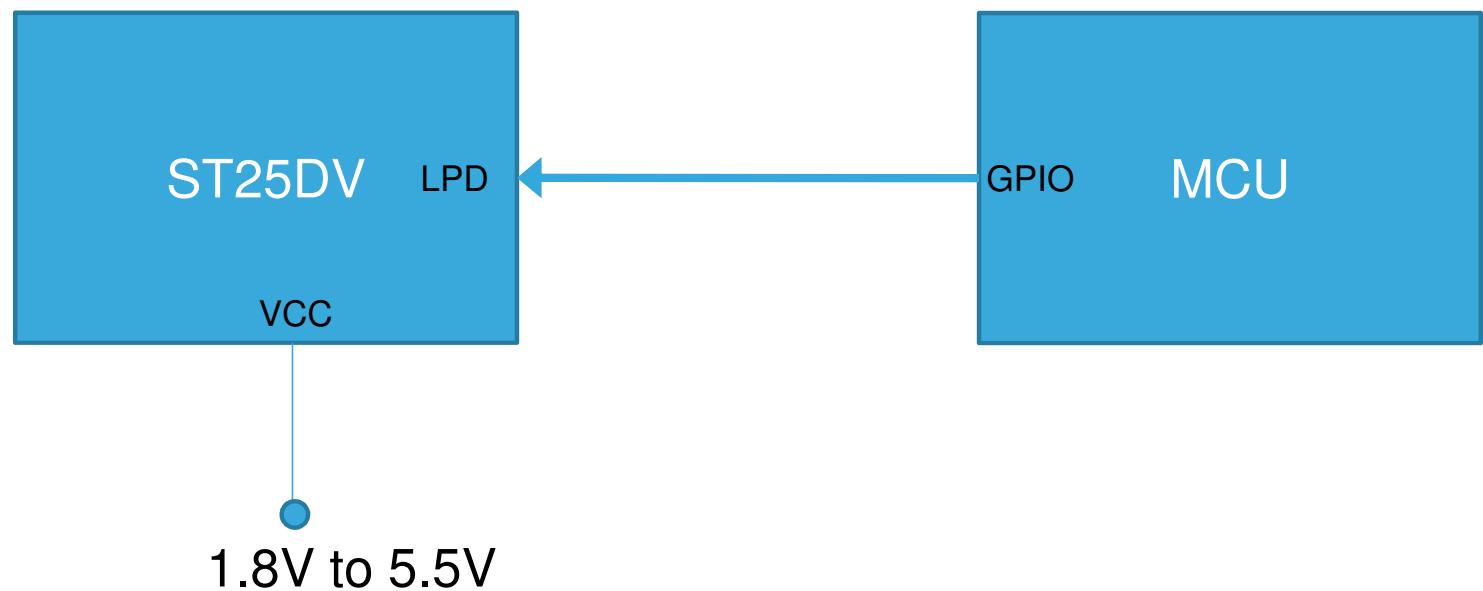
# ST25DV GPO Power Block

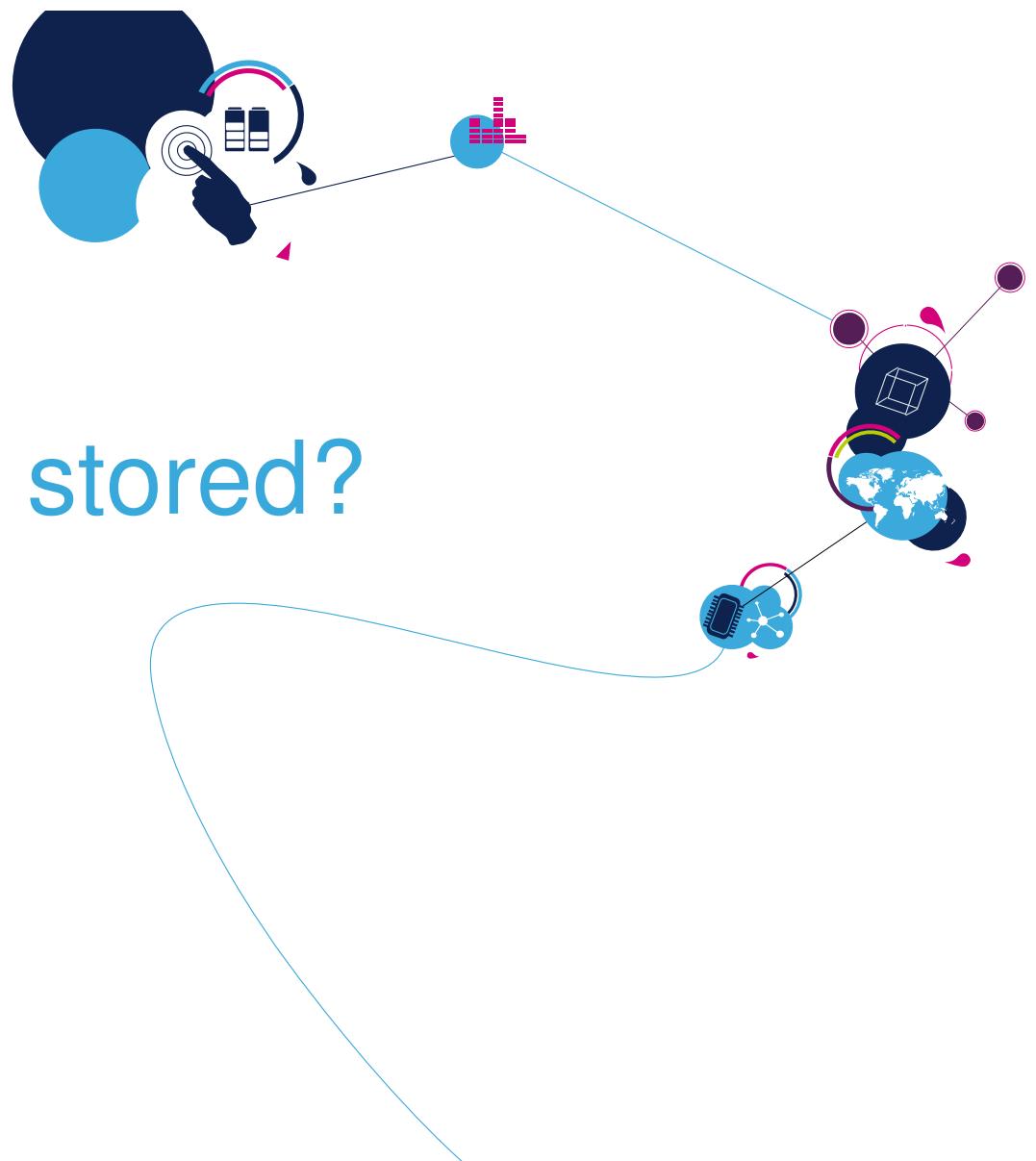
82

In this configuration, a RF field detection will wakeup the MCU.



In this configuration, MCU puts the ST25DV in low-power mode consuming less than 1uA by driving the LPD pin HIGH.





# How is data being stored?

# NFC and RFID

NFC specification  
→ Upper layer SW

NFC Forum  
Type 2 and Type 4

NFC Forum  
Type 5 \*

ISO standards  
→ HW/SW protocol  
→ 13.56 MHz

ISO14443  
Type A and Type B  
« Short Range »  
106kbps

ISO15693  
« Long Range »  
26kbps

(\*) ISO15693 integrated in NFC Forum specifications in October 2015 as NFC Forum type 5 (aka type V)

# What is a NDEF record?

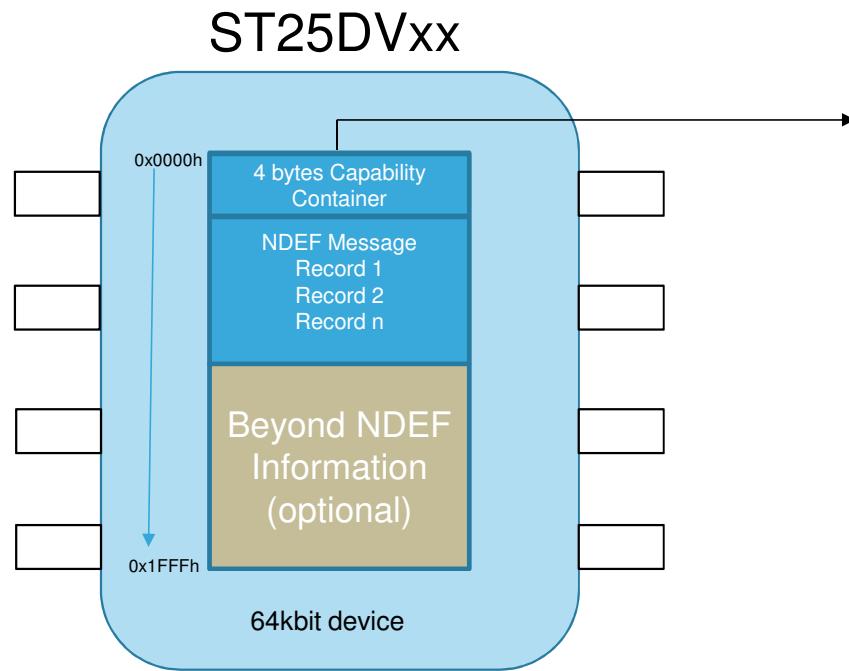
- Contain the User data.
- The NFC forum defines the format of the NDEF record.
- Different kinds of NDEF records :
  - URL
  - Text
  - SMS
  - Wi-Fi pairing
  - Bluetooth pairing
  - Smart poster....

→ for detailed Info for NDEF record structure, please refer to the [Technical Specifications of the NFC Forum!](#)



# Sensor data stored as NDEF message

87



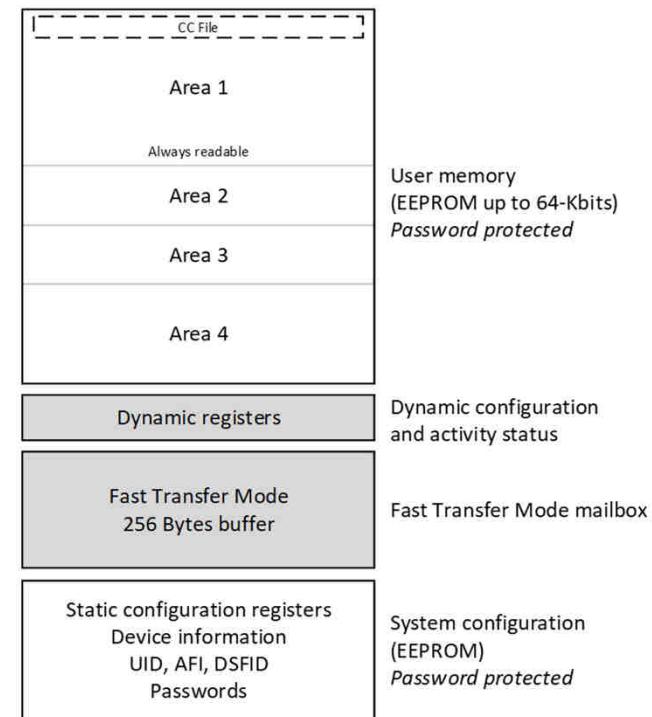
Area	Block	Data	ASCII
01	00	E2 40 00 01	â @ ..
01	01	00 00 03 FF	.. . y
01	02	03 FF 00 FF	. y . y
01	03	44 0E 00 00	D ...
01	04	00 EB 73 74	. è st
01	05	2E 63 6F 6D	. c o m
01	06	3A 73 6D 61	: s m a
01	07	72 74 61 67	r t a g
01	08	01 01 00 02	....
01	09	05 00 01 3F	... ?
01	0A	DC A7 CC 00	Ü s i .
01	0B	41 40 32 28	A @ 2 (
01	0C	54 4B 3B 04	T K 8 .
01	0D	00 01 00 00	....
01	0E	05 00 22 00	.. " .
01	0F	77 B9 CC 00	w i l .
01	10	05 00 22 00	.. " .
01	11	0A 00 22 00	.. " .
01	12	40 40 3B 3A	@ @ ; :
01	13	05 00 22 00	.. " .
01	14	12 A9 CC 00	. @ l .
01	15	F0 A8 CC 00	ð " l .
01	16	E9 86 6E 04	é † n .
01	17	05 00 88 00	.. " .
01	18	05 00 22 00	.. " .
01	19	C4 0E 98 6E	Ã . " n
01	1A	0A 00 22 00	.. " .
01	1B	84 0E 98 6E	„ " n
01	1C	0F 00 22 00	.. " .

- NDEF NFC Data Exchange Format

# ST25DV Memory

- Memory organization
  - User memory
    - 4Kbits/16Kbits/64Kbits EEPROM
    - 4 configurable areas
    - Areas protectable by passwords
  - System configuration
    - EEPROM
    - Static registers, define behavior at boot
    - Device identifiers and passwords
    - Protected by password
  - Dynamic configuration
    - Dynamic registers
    - Status and temporary configuration
  - Fast transfer mode buffer
    - Message exchange between I2C and RF
    - 256 Bytes mailbox buffer

ST25DV memory organization



# Sensor data structure

89

- Each event requires 8 bytes
  - 4 bytes are time stamp
  - 4 bytes are sensor data
- When orientation is also logged, 12 bytes are required vs. 8 bytes
- When memory is full, last sample pointer will be at the beginning of data

# Tag configuration data and addresses

90

Value	Byte Addr	Add	Add I2C	Add RF	Note				
SmarTag SW Versions	RecordVersion	0	0	0					
	Major FW Version	1	1	0					
	Minor	2	2	0					
	Patch	3	3	0					
SmarTag Conf	Sample rate (Low)	4	4	1					
	Sample rate (High)	5	5	1					
	Log Mode	6	6	1	0x0 Inactive	0x1 Sampling Log	0x3 Sampling with Ths	0x4 Save next Sample	
	Sensor Enable Flags	7	7	1	0x01 Temp	0x02 Hum	0x04 Press	0x08 Acc	0x10 6DOrientation
TimeStamp		8	8	2	Year = (((TimeStamp >> 26) & 0x1F);	Minutes = (((TimeStamp >> 6) & 0x3F);			
		9	9	2	Date = (((TimeStamp >> 21) & 0x1F);	Seconds = (((TimeStamp )) & 0x3F);			
		10	A	2	Month = (((TimeStamp >> 17) & 0x0F);				
		11	B	2	Hours = (((TimeStamp >> 12)) & 0x1F);				
Thresholds Used for Log Mode==0x3	T Max	12	C	3					
	T Min	13	D	3					
	H Max	14	E	3					
	H Min	15	F	3					
	Pmax(12)	16	10	4					
	Pmin(12)	17	11	4					
		18	12	4					
	AccMax(8bit)	19	13	4					

# Tag configuration data and addresses

91

Value	Byte Addr	Add	Add I2C	Add RF		
ADDR COMMAND REPLY	Read New Conf	20	14	5	Read New Configuration and new RTC value setted by App	
	Single Shot Ready	21	15	5	Written Sigle shot result by SmarTag	
	RFU	22	16	5		
	RFU	23	17	5		
MAX_T_32BITDATATIME_ADDR		24	18	6	TimeStamp = 0 <<31;	TimeStamp  = (((uint32_t) (Hours )) << 12);
		25	19	6	TimeStamp  = (((uint32_t) (Year &0x1F))	TimeStamp  = (((uint32_t) (Minutes )) << 6);
		26	1A	6	TimeStamp  = (((uint32_t) (Date )) << 6);	TimeStamp  = (((uint32_t) (Seconds )) ) );
		27	1B	6	TimeStamp  = (((uint32_t) (Month )) << 17);	

# Tag configuration data and addresses

Value	Byte Addr	Add	Add I2C	Add RF
MIN_T_32BITDATATIME_ADDR	28	1C	7	
	29	1D	7	
	30	1E	7	
	31	1F	7	
MAX_H_32BITDATATIME_ADDR	32	20	8	
	33	21	8	
	34	22	8	
	35	23	8	
MIN_H_32BITDATATIME_ADDR	36	24	9	
	37	25	9	
	38	26	9	
	39	27	9	
Max&Min for T and H	Max T	40	28	A
	MinT	41	29	A
	MaxH	42	2A	A
	MinH	43	2B	A
MAX_P_32BITDATATIME_ADDR	44	2C	B	
	45	2D	B	
	46	2E	B	
	47	2F	B	
MIN_P_32BITDATATIME_ADDR	48	30	C	
	49	31	C	
	50	32	C	
	51	33	C	

MIN_P_32BITDATATIME_ADDR	48	30	C
	49	31	C
	50	32	C
	51	33	C
MAX_Acc_32BITDATATIME_ADDR	52	34	D
	53	35	D
	54	36	D
	55	37	D
Max&Min for P and Max Acc	56	38	E
	Pmax(12)	57	39
	Pmin(12)	58	3A
	AccMax(8bit)	59	3B

# Tag configuration data and addresses

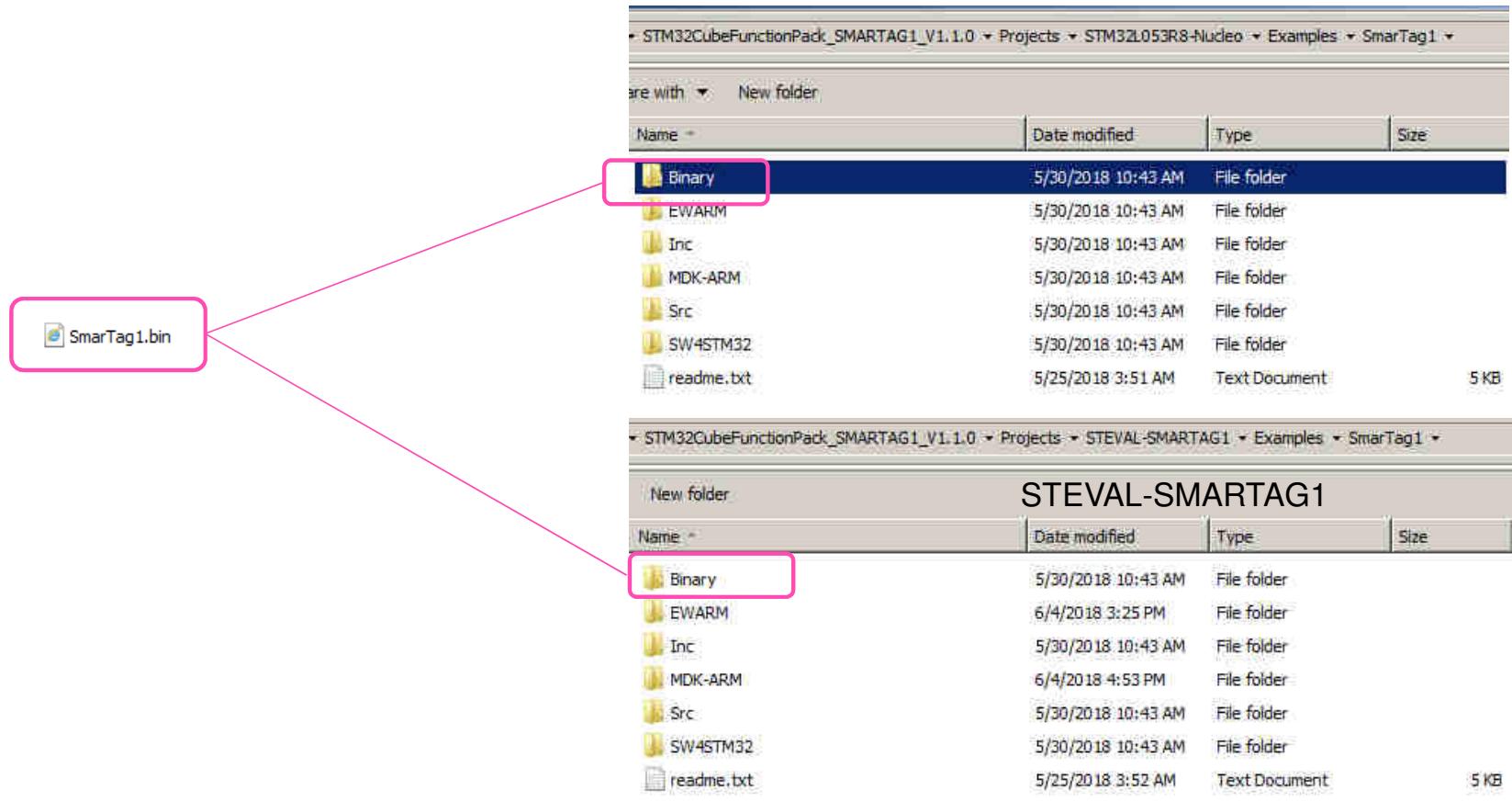
93

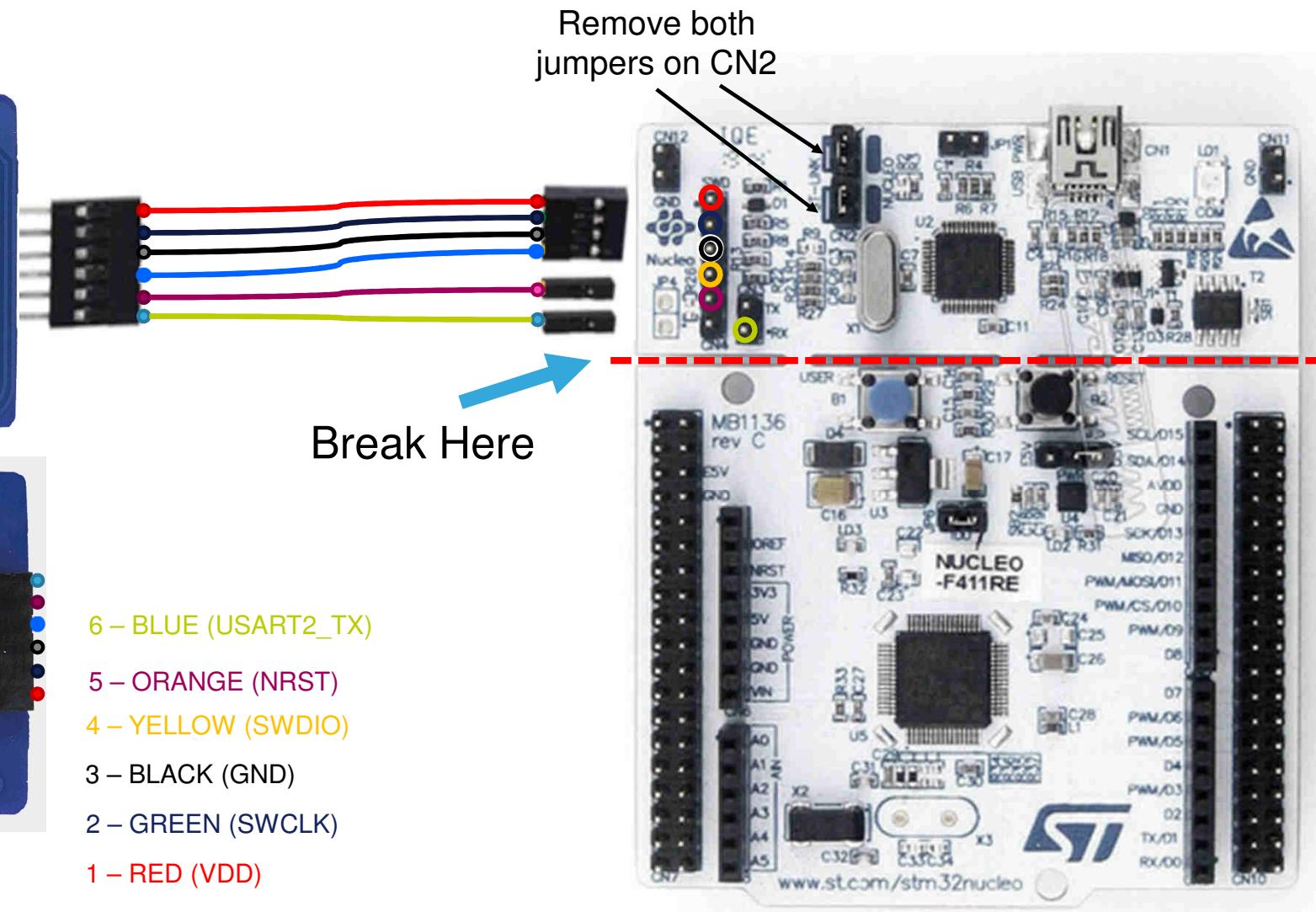
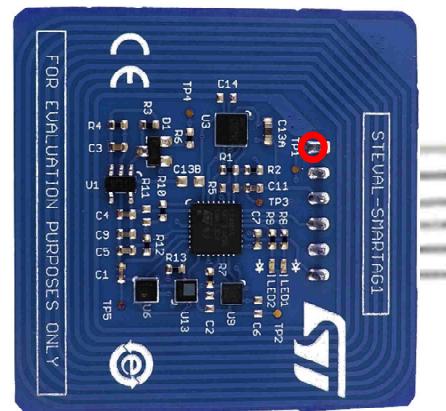
Value	Byte Addr	Add	Add I2C	Add RF				
DataLog (TimesTamp 4 bytes + Values 4 bytes) (1015 Possible values without Header NDEF)	TimeStamp1	64	40	10	TimeStamp = Sync/Async <<31;	TimeStamp  = (((uint32_t) (Hours )) << 12);		
		65	41	10	TimeStamp  = (((uint32_t) (Year &0x1F))	TimeStamp  = (((uint32_t) (Minutes )) << 6);		
		66	42	10	TimeStamp  = (((uint32_t) (Date )) << 14);	TimeStamp  = (((uint32_t) (Seconds )) );		
		67	43	10	TimeStamp  = (((uint32_t) (Month )) << 17);			
	Values1	68	44	11	For Sync Event:			
		69	45	11	For Values from MSB to LSB			
		70	46	11	Press 12 bits, Temp 7Bits, Hum 7Bits, Acc 6Bits			
		71	47	11				
	TimeStamp2	72	48	12				
		73	49	12				
		74	4A	12				
		75	4B	12				
	Values2	76	4C	13	For Async Event:			
					First 3 bits: Orientation type: #define ORIENTATION_UNDEF 0x00 #define ORIENTATION_RIGHT 0x01 #define ORIENTATION_TOP 0x02 #define ORIENTATION_LEFT 0x03 #define ORIENTATION_BOTTOM 0x04 #define ORIENTATION_UP 0x05 #define ORIENTATION_DOWN 0x06	Next 6 bits: Type of event: #define ACC_WAKEUP 0x01 #define ACC_6D_ORIENTATION 0x02 #define ACC_SINGLE_TAP 0x04 #define ACC_DOUBLE_TAP 0x08 #define ACC_FREE_FALL 0x10 #define ACC_TILT 0x20		Next 6 bits: Acc max for WakeUp event
		77	4D	13				

# Downloading Firmware



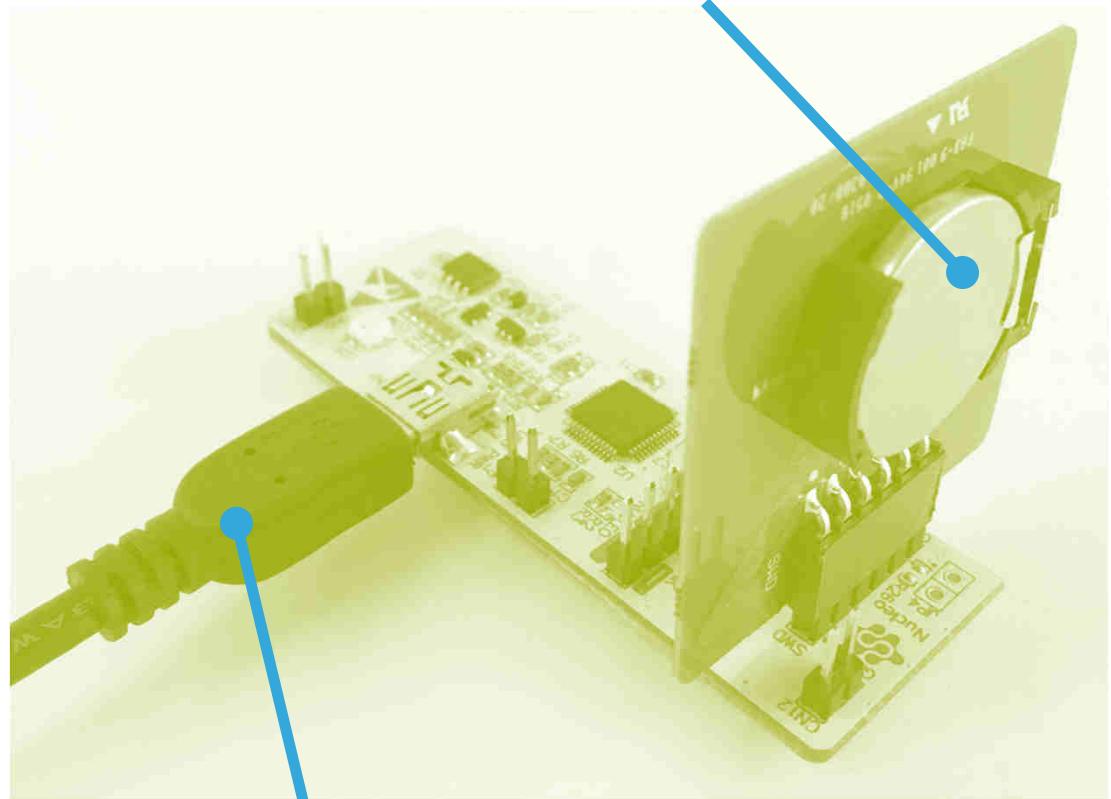
# Downloading Binary





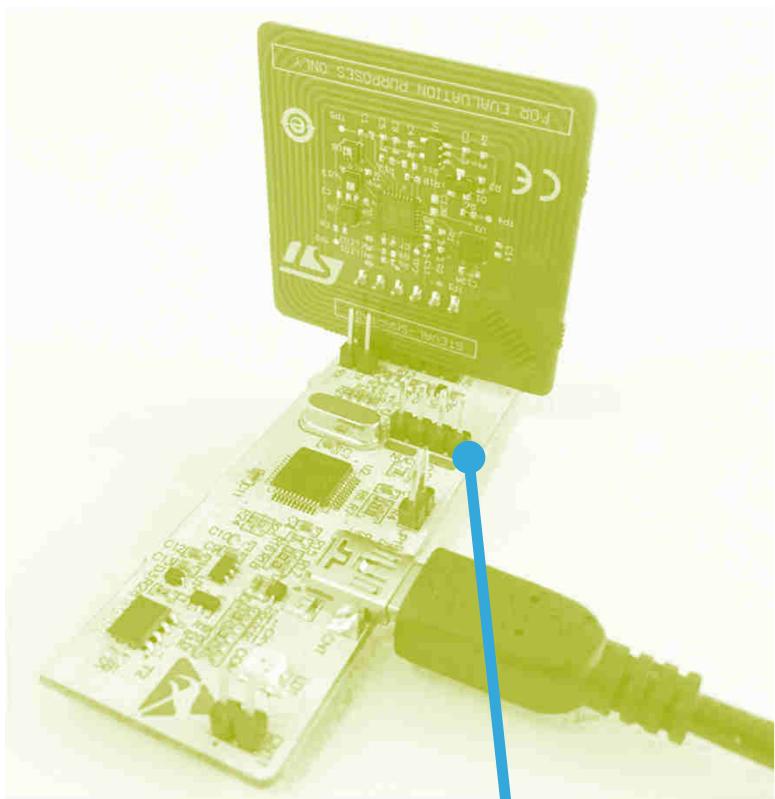
## CR2032 battery

NFCSensorTAG must be powered  
to be able to flash and debug



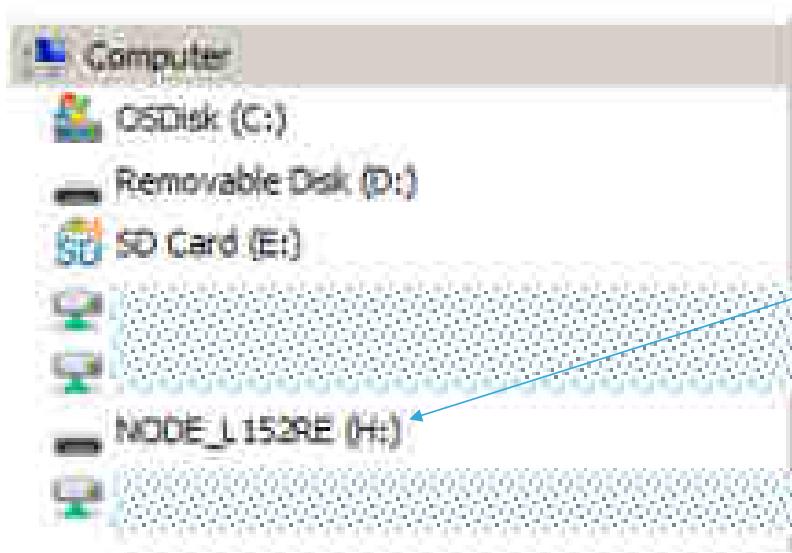
## CN2

Remove jumpers to be able  
to flash and debug



# Binary Drag and Drop

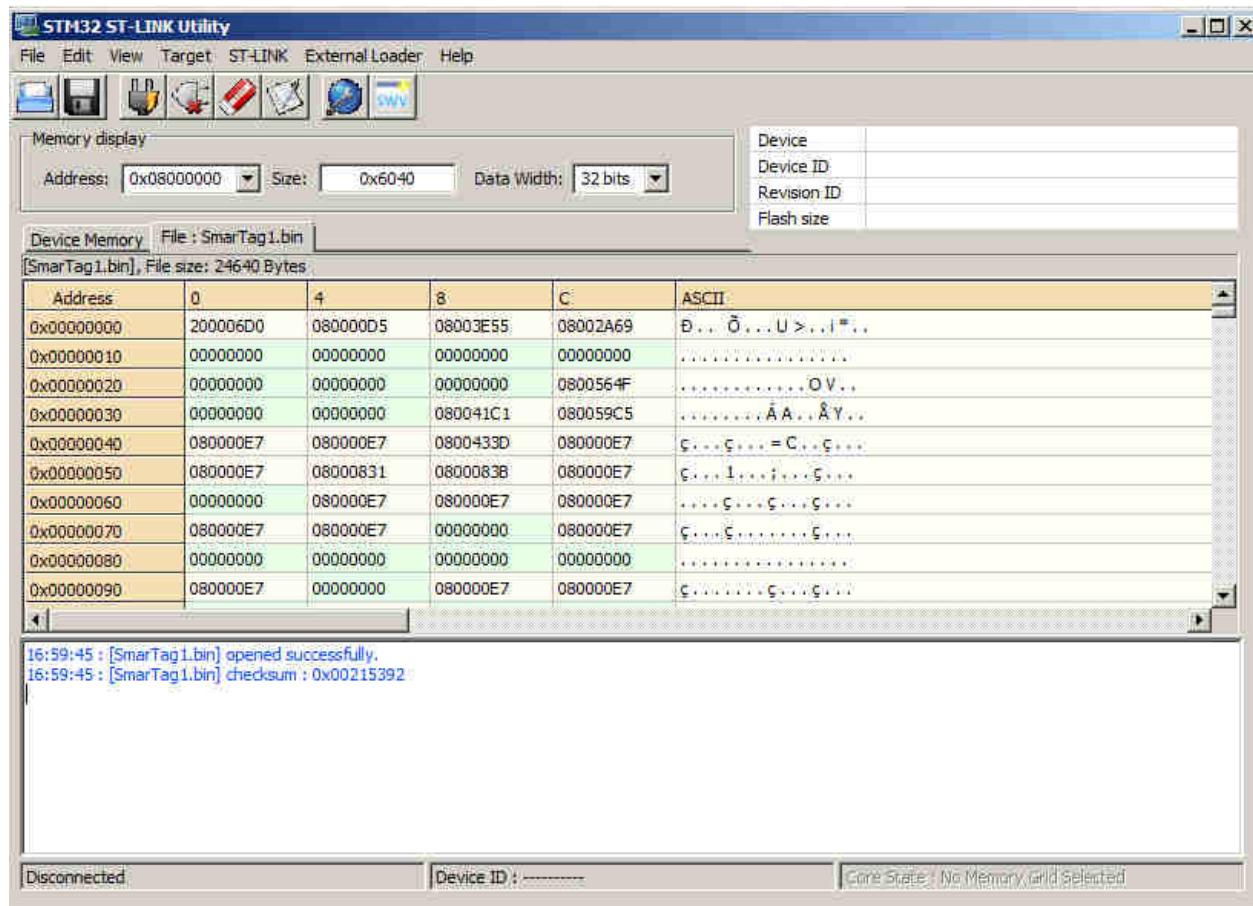
98



STM32CubeFunctionPack_SMARTAG1_V1.1.0 ▾ Projects ▾ STEVAL-SMARTAG1 ▾ Examples ▾ SmarTag1 ▾ Binary			
New folder			
Name	Date modified	Type	Size
SmarTag1.bin	5/25/2018 3:45 AM	BIN File	25 KB

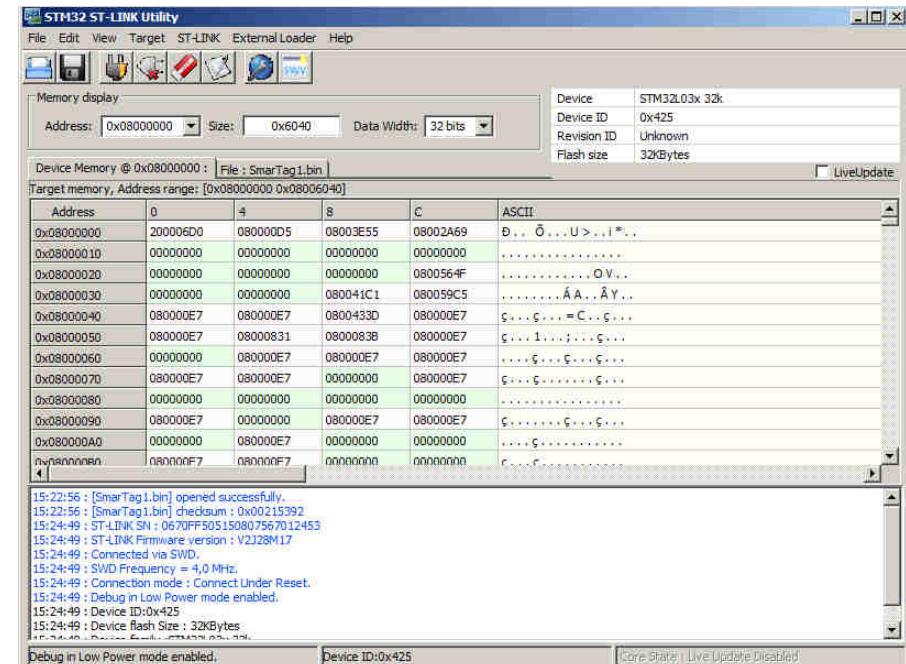
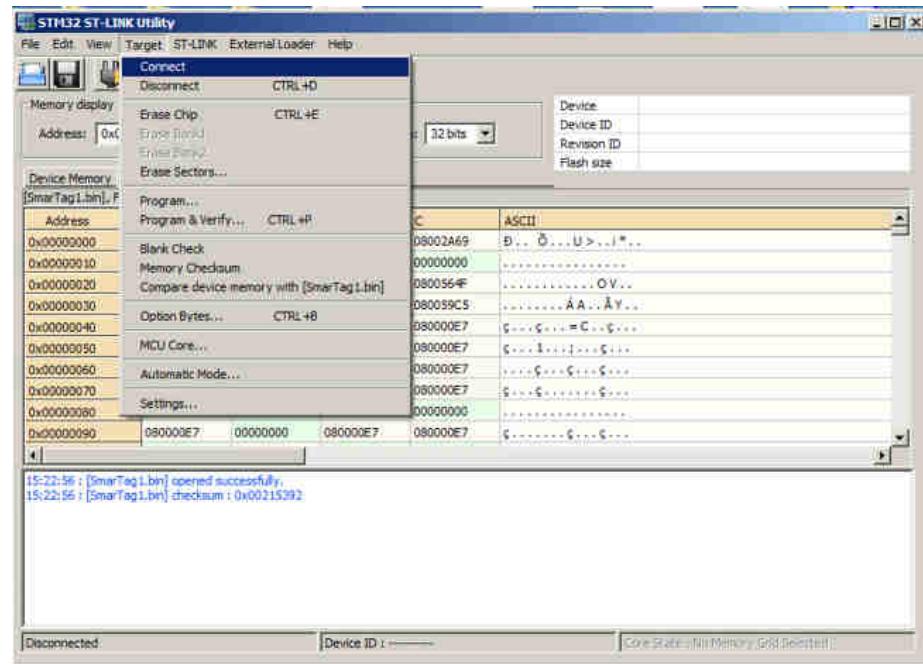
Note: The name of the drive node does not matter. You can use the STLINK part from any STM32 Nucleo board

# Using STM32 ST-LINK Utility



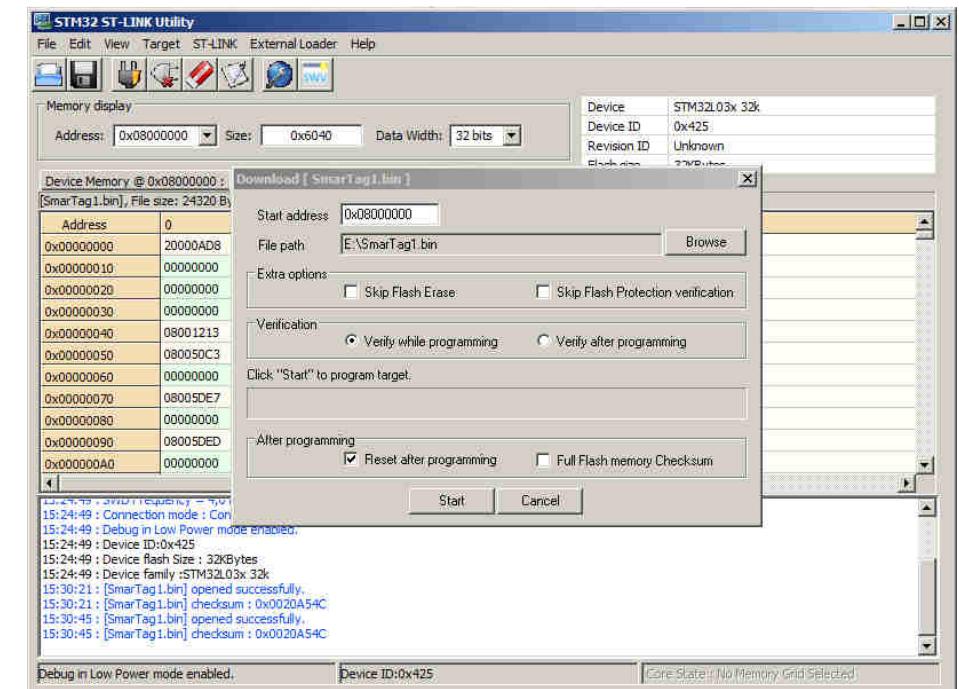
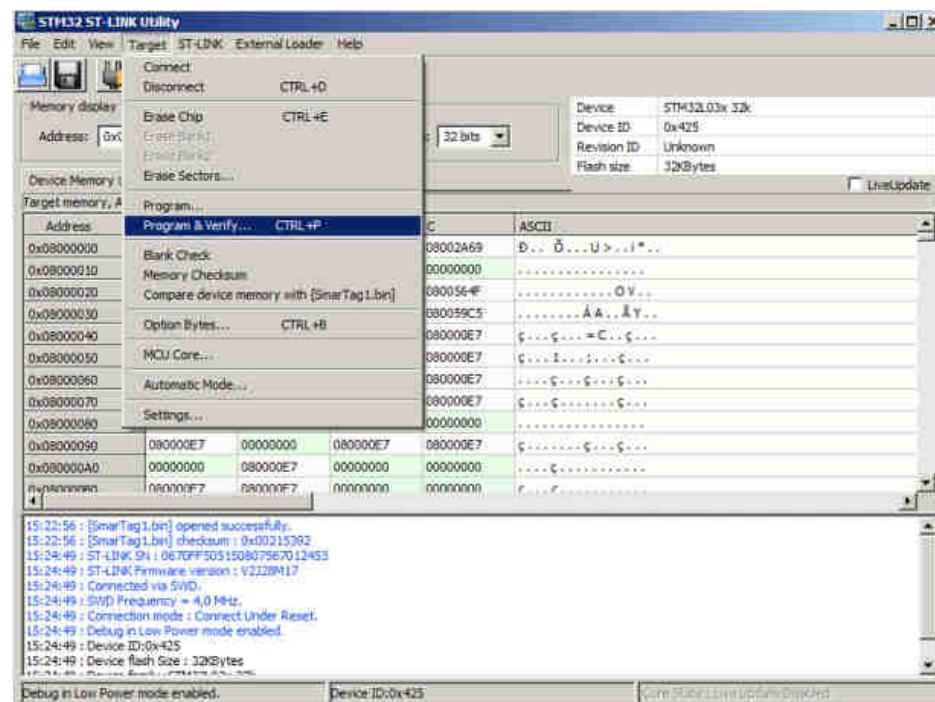
# Connecting to ST-LINK

100



# Start Programming

101



# Let's get to the Hands-On Section

# LAB Preparation

- For the workshop ST will provide



**ST USB Key**  
with presentation of the  
workshop



**ST25R3911B Discovery Kit**



**Micro USB Cable**



**CR2032 Battery**



**NFC Sensor Tag Evaluation Board**

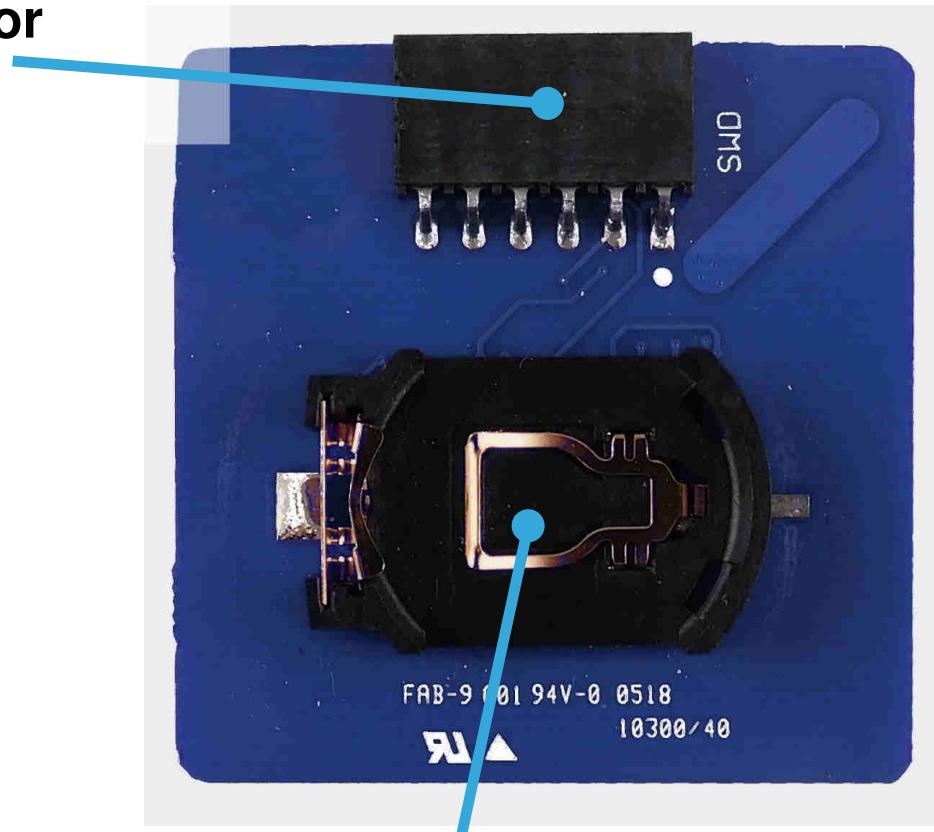
# Installation of ST25PC Software

- Locate the executable en.ST25PC-NFC.exe in the flash drive.
- Click and install it. In some computers, the installation might need internet access to download certain Microsoft Visual C++ Redistributable (x86)
- Say “yes” to driver install.

# Install the battery

## SWD connector

To ST-Link/V2

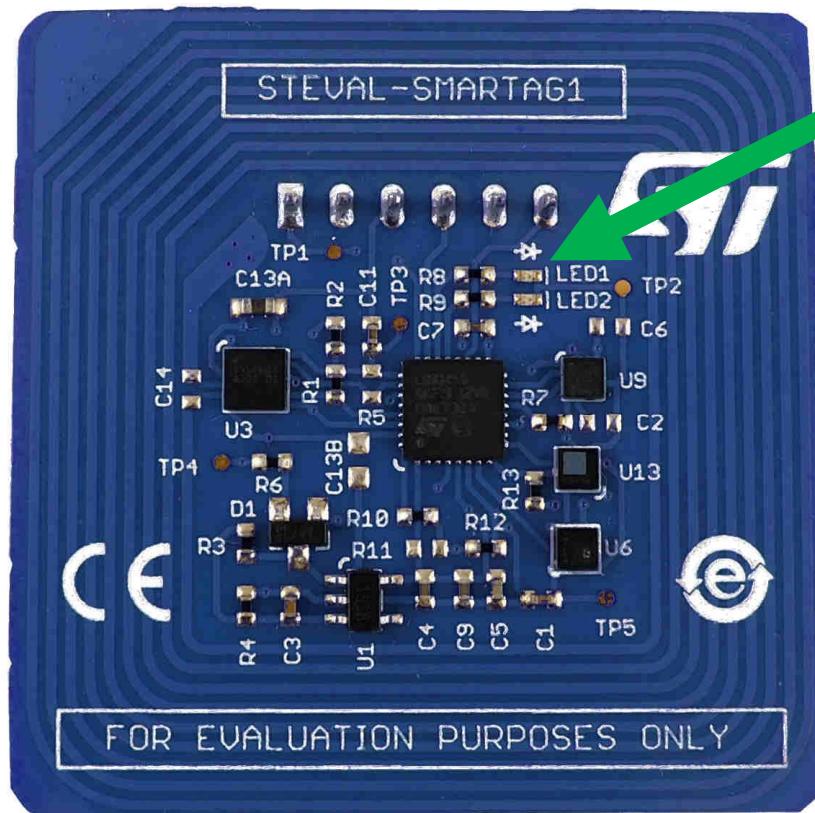


**Important Note:**  
Load your coin  
cell battery with  
+ facing out.



# Check if battery and tag are good!

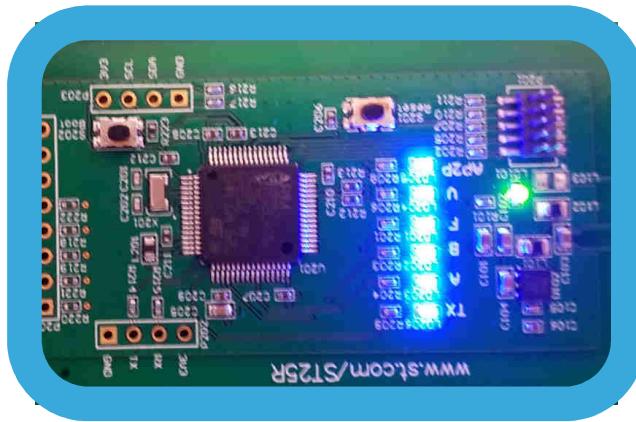
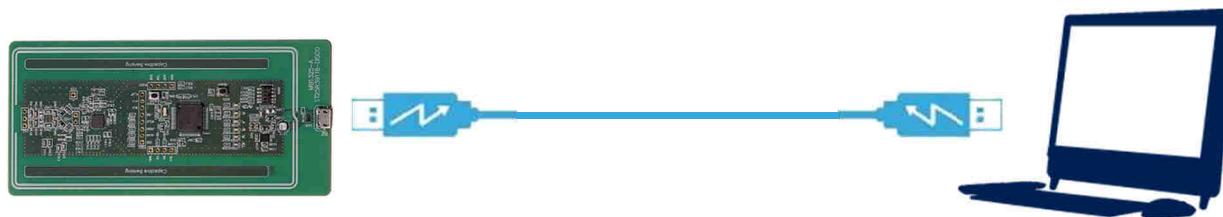
106



After 8 seconds, LED1 should blink  
every 5 seconds

# Connect your reader

107

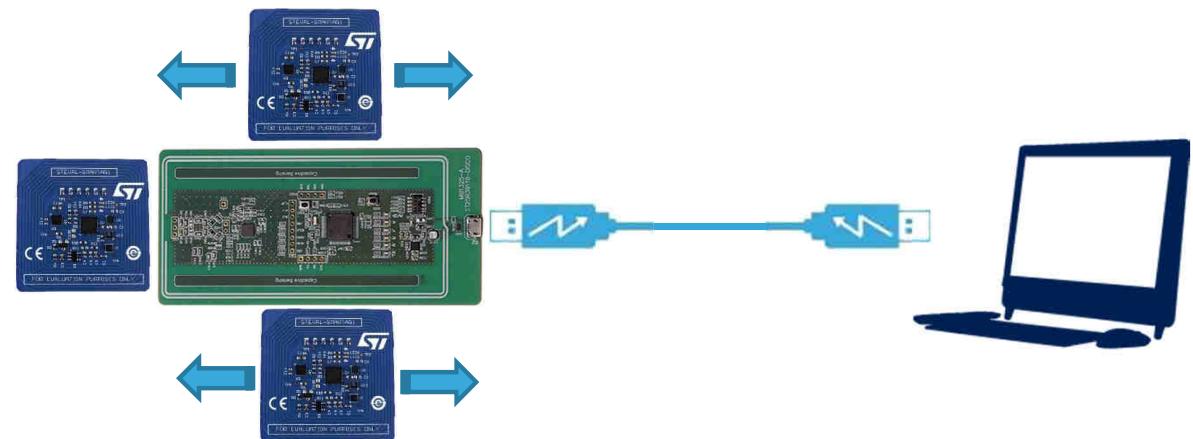


# Tag Placement while reading

## Important Note:

You can place the tag on top or under the ST25R3911B ONLY if there is insulation between them.

Or you will risk shorting out the components!



# Now run your ST25PC-NFC Application

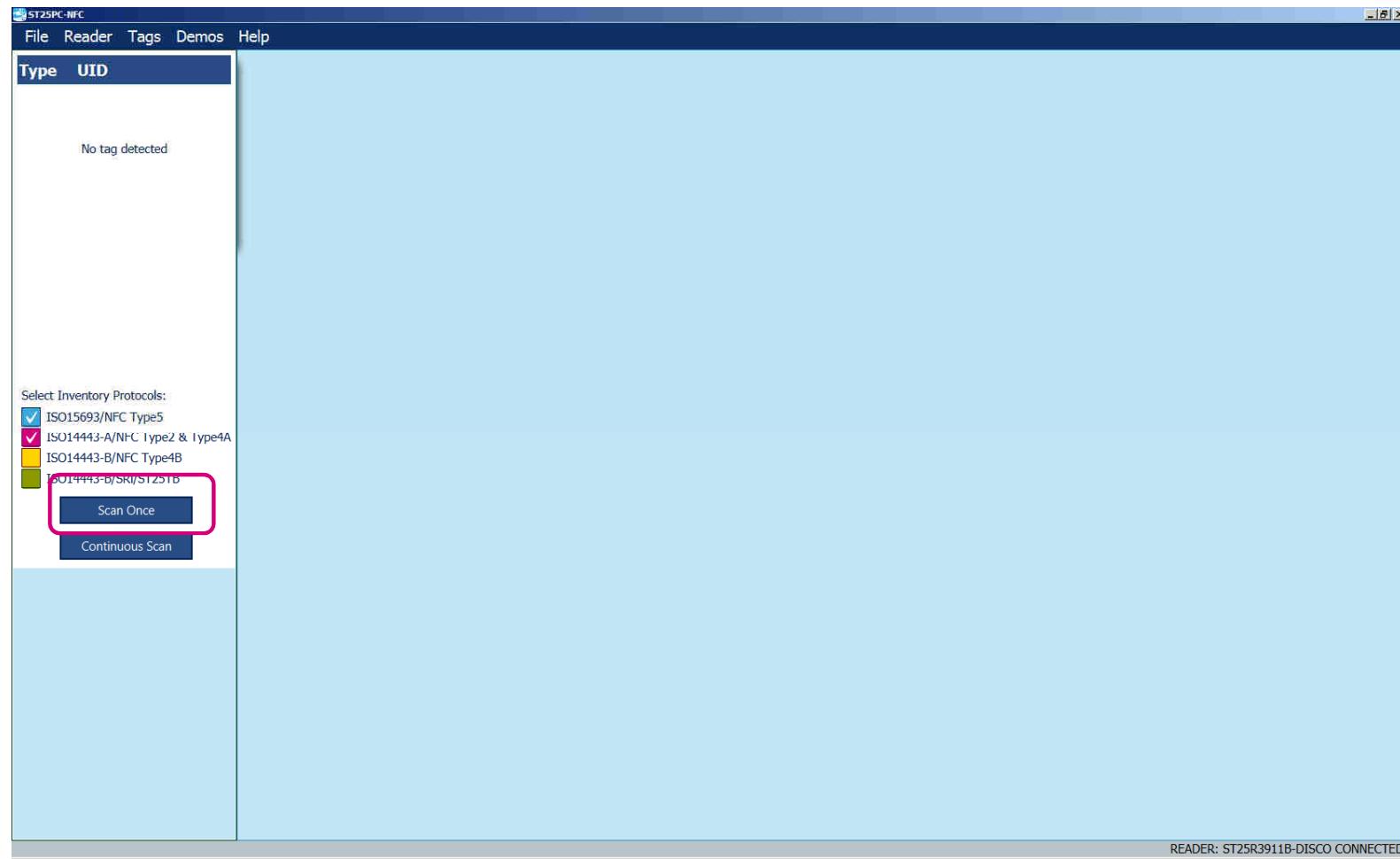
109



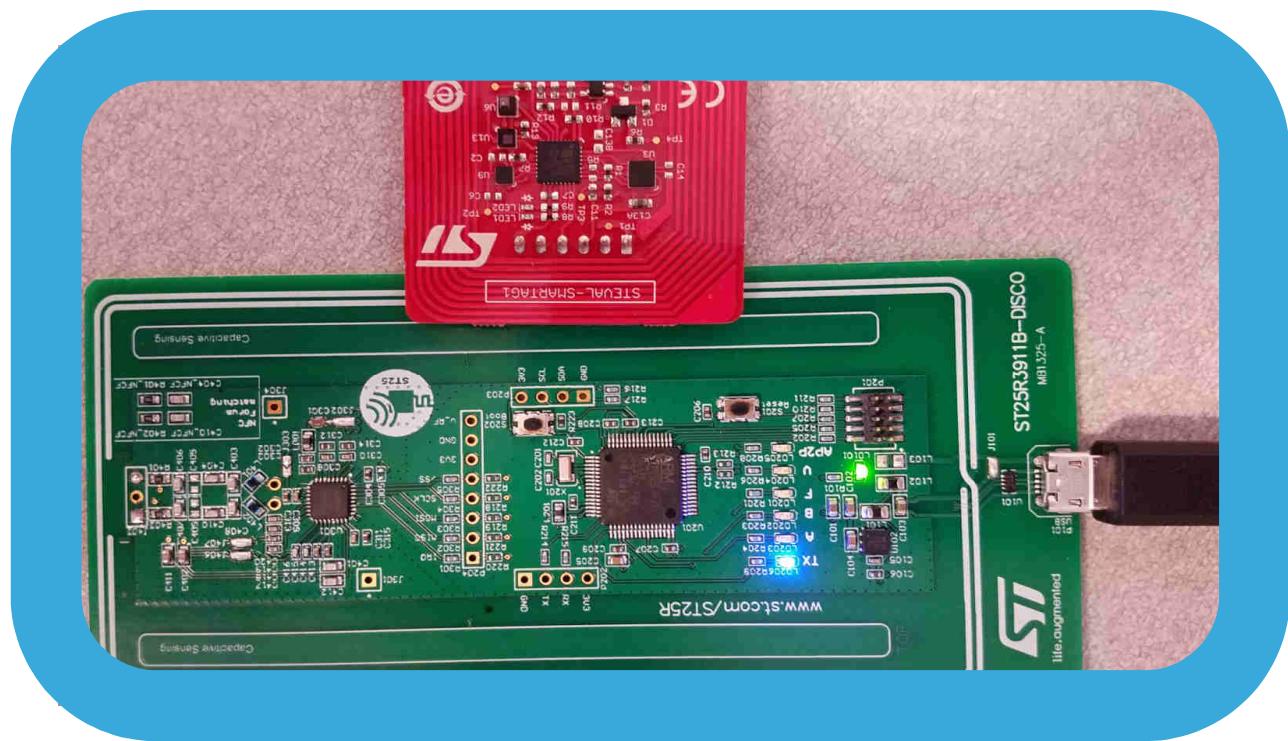
Make sure that the  
software reports that  
the reader is  
connected as seen  
here

# First, scan for the tag

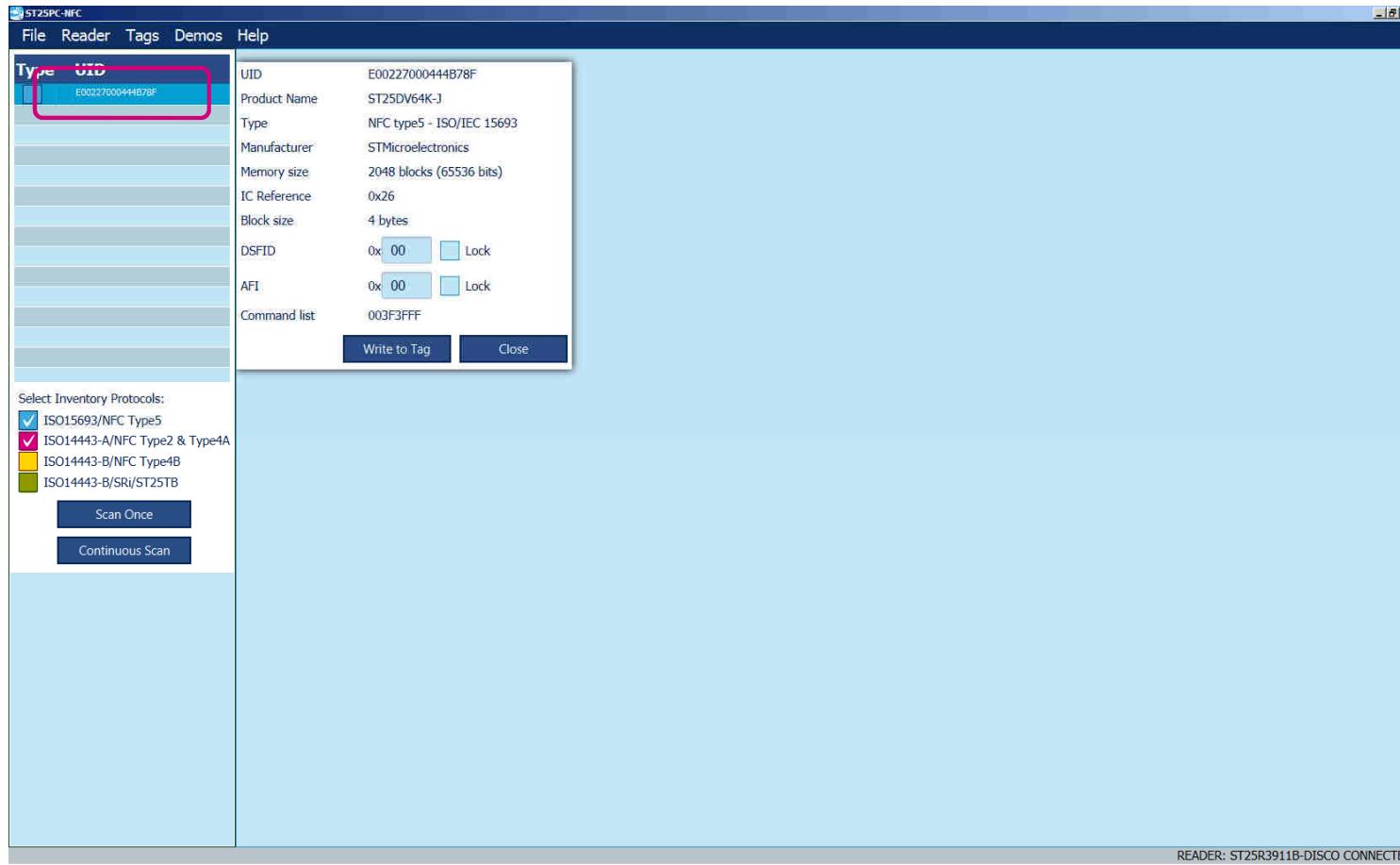
110



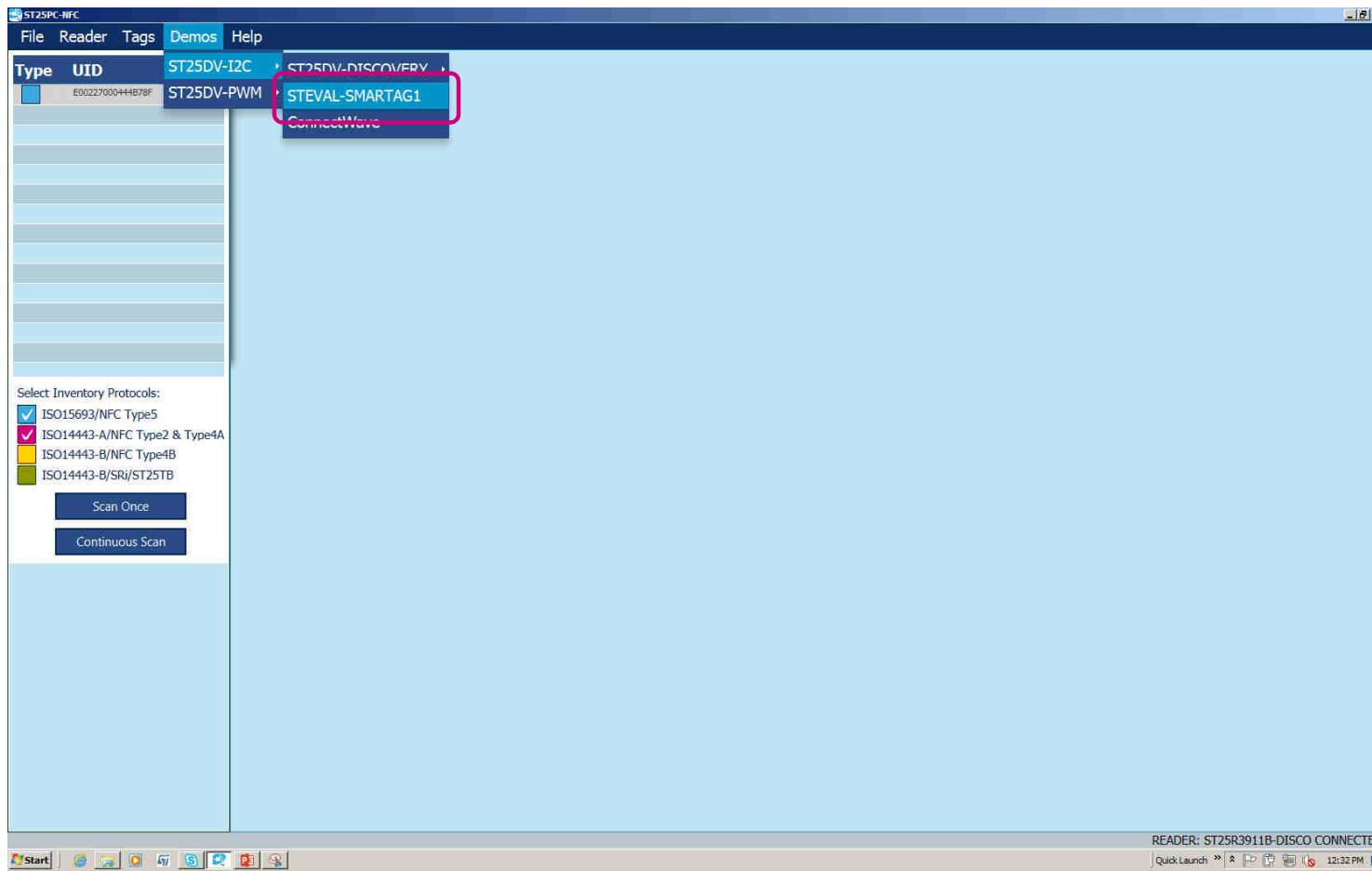
# Correct LED patterns on the ST25R3911B



# View tag UID, memory size and more

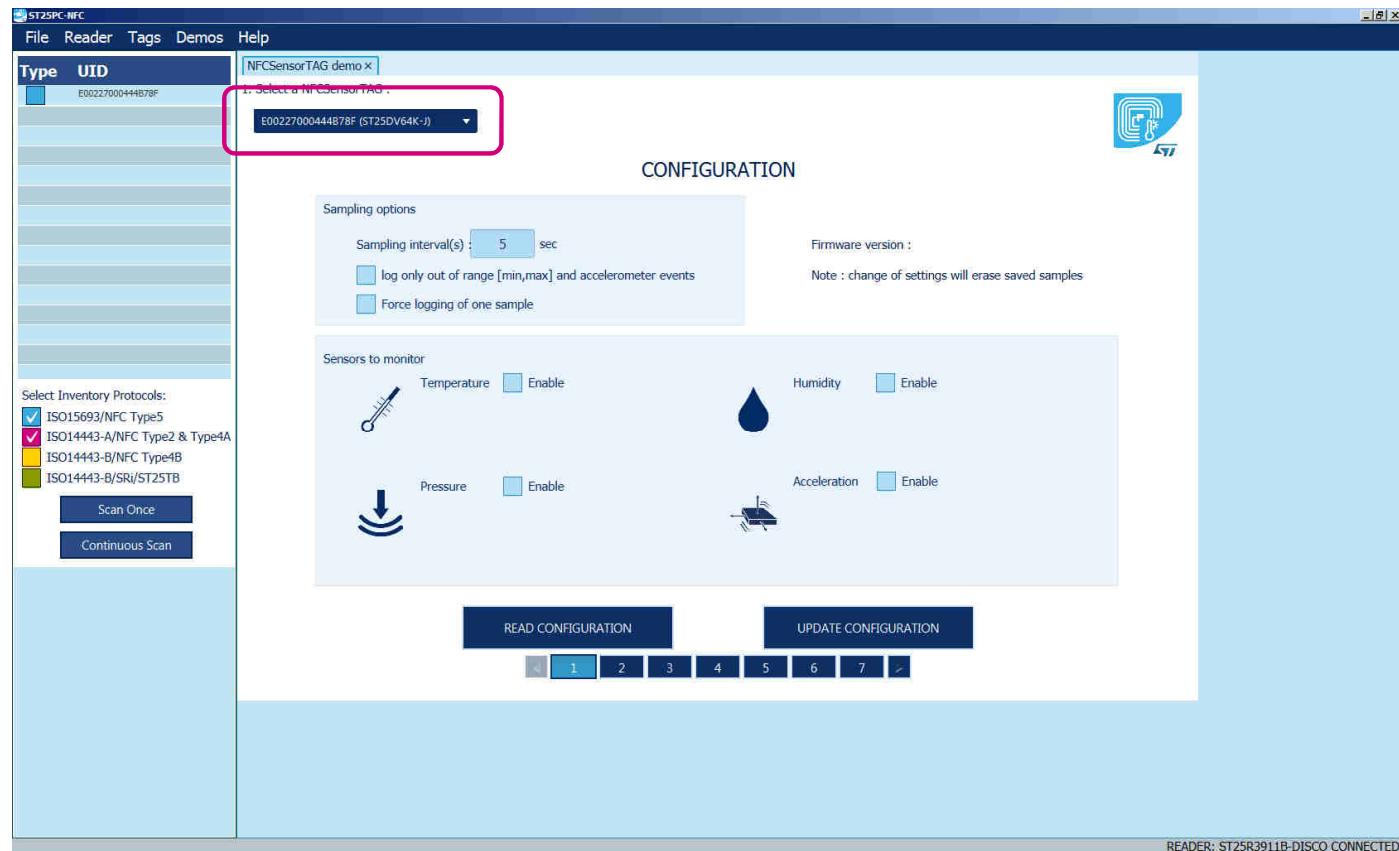


# Running the sensor tag demo



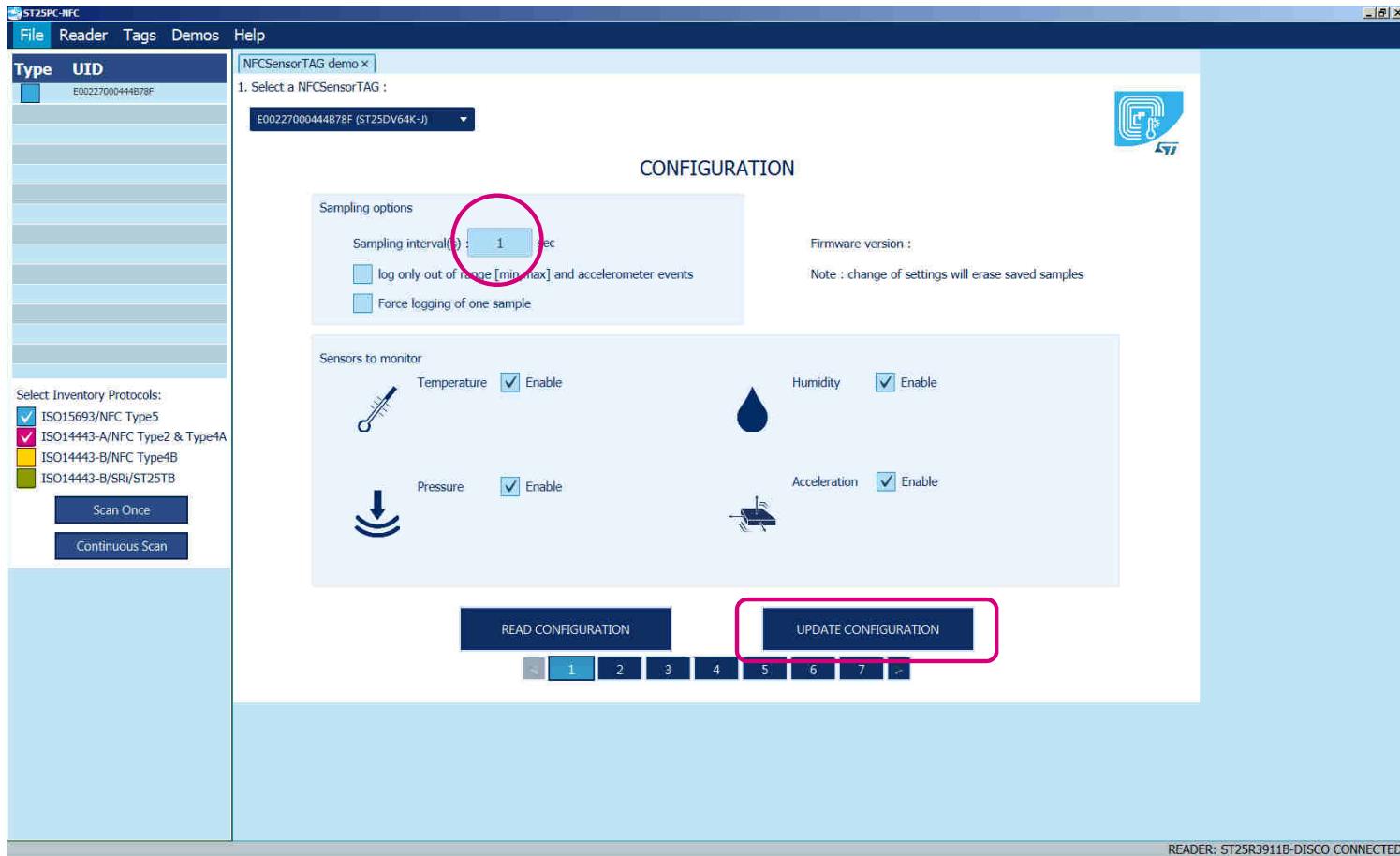
# Sensor default configuration

114

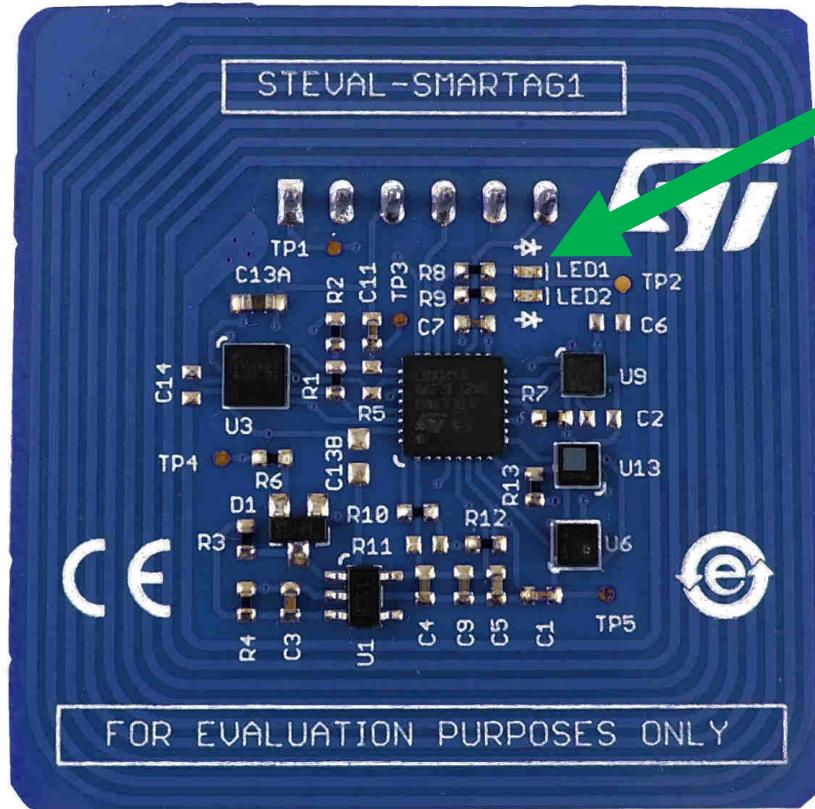


# Setting new configuration

115



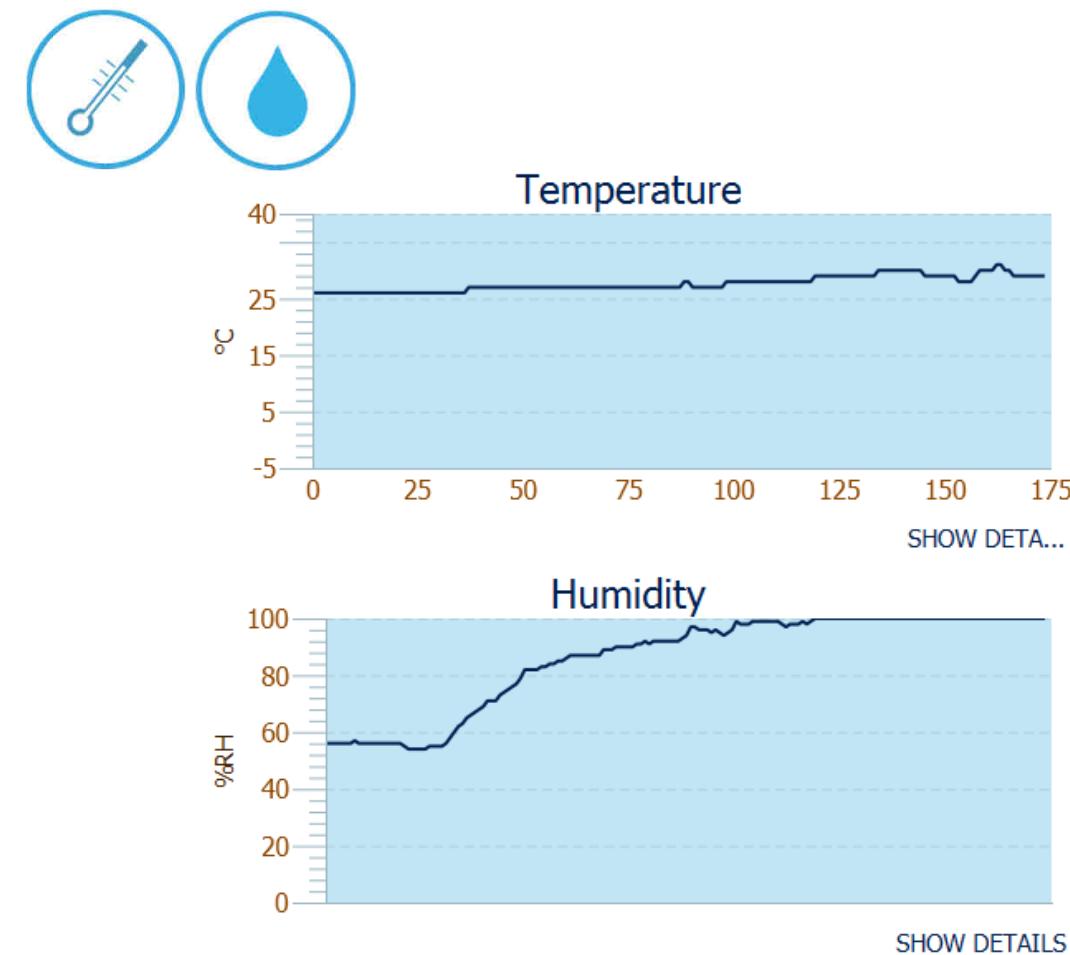
# Check if your successful!



- Remove the tag from the reader RF field
- LED1 now should blink every 1 seconds



# Time to exercise your sensor tag



## HTS221

Relative Humidity and Temperature combo



### Humidity

- Range: 0%RH : 100%RH
- Accuracy:  $\pm 3.5\%$ RH

### Temperature

- Range: -40°C : +125°C
- Accuracy:  $\pm 0.5^\circ\text{C}$ , from 15°C to +40°C

### Advantages

- High Accuracy
- Low power consumption 2  $\mu\text{A}$
- Extended operative supply voltage

# Changing the humidity and temperature

119

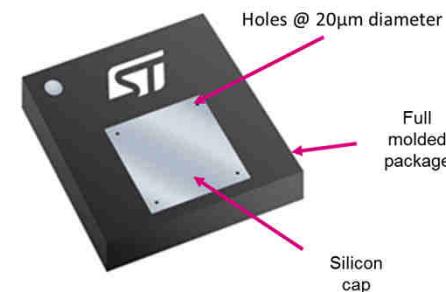
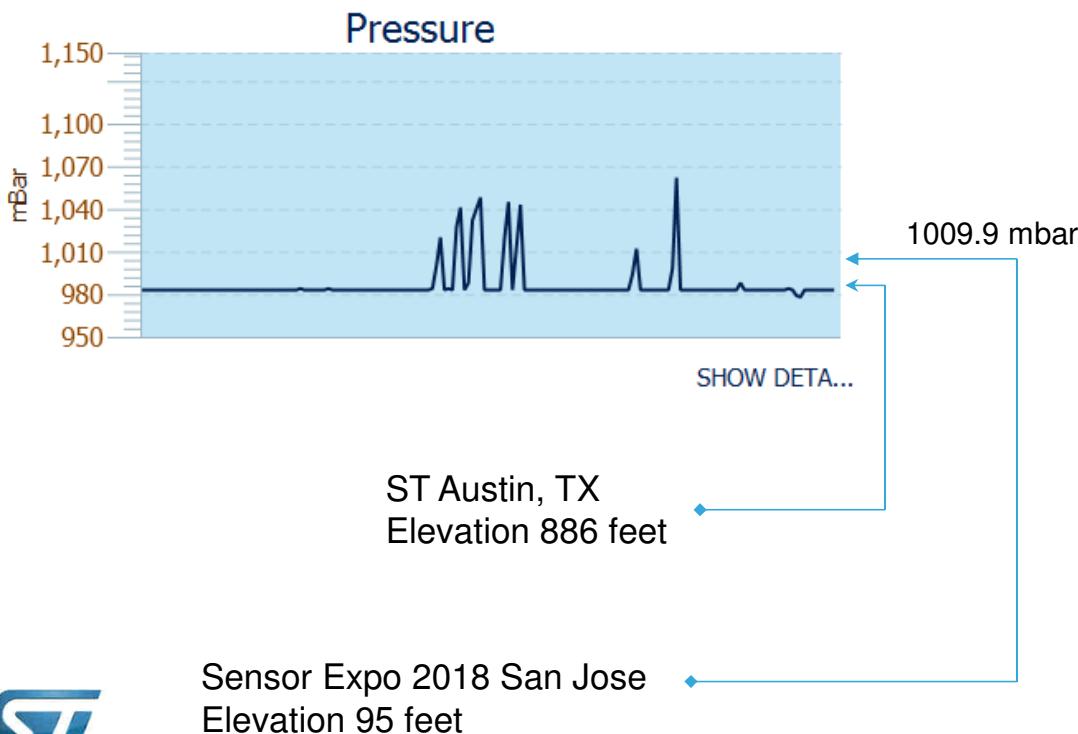
- Blowing on the sensor will increase the humidity at point of measurement.
- Put the sensor tag into the bag and inflating with your breath will put it in a high humidity environment.
- Please don't put water on the tag because it will short out the circuit.
- Warm your hand up against a cup of coffee and keep the tag in your hand to log higher temperature



## LPS22HB

120

Pressure  
Barometer/Altimeter  
Sensor



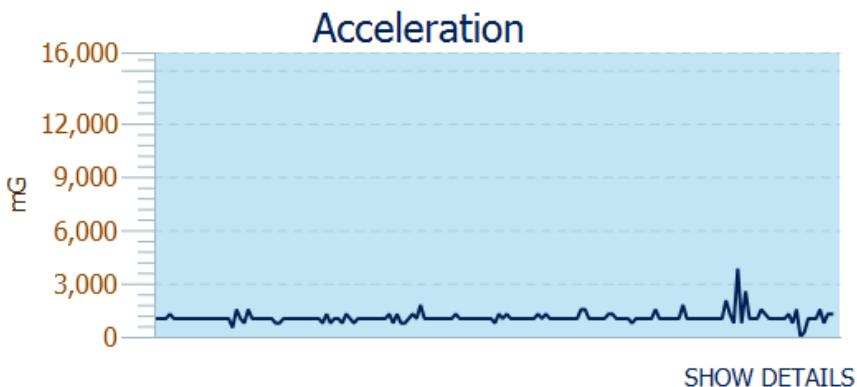
### Pressure

- Range 260-1260 hPa
- Relative accuracy of pressure measurement:  
 $< 10 \mu\text{bar}$   
6cm resolution

# Changing the pressure

121

- Change pressure by inflating the Ziploc bag
- Close the Ziploc bag and lightly press on the “air bag”. Try not to rupture the bag. Each time you do this you will increase the pressure.



## LIS2DW12

### 3-axis Accelerometer



Package

2x2x0.7 mm

#### Accelerometer

- $\pm 2/\pm 4/\pm 8/\pm 16$  g selectable Full Scales
- Noise/accuracy level from 12 to 14bit resolution
- Low current consumption  
380nA LP Mode and 40nA Standby

# Shock and Impact

123

- Register the following action and make note of it.
- Drop onto a carpeted floor
- A tap of your finger against the body of the tag
- Please don't throw the sensor tag against a hard surface as components might come loose.

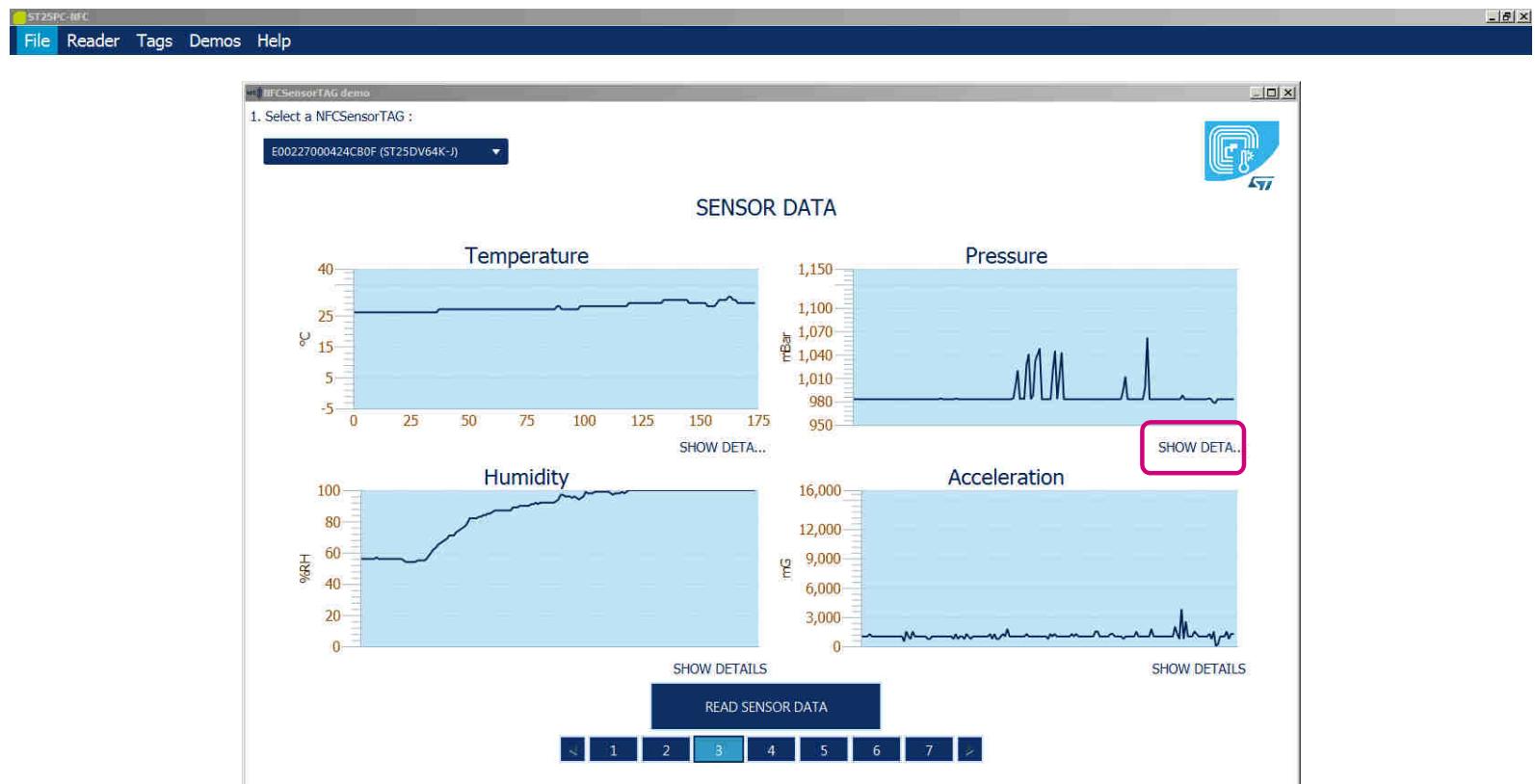


# Let's view your data!



# Show details

125



# Show Details Panel

126

ST25PC-NFC

File Reader Tags Demos Help

NFCSensorTAG demo

1. Select a NFCSensorTAG :

E00227000424C80F (ST25DV64K-J) ▾

SENSOR DATA

Temperature

SHOW DETAILS...

Humidity

SHOW DETAILS...

Acceleration

SHOW DETAILS...

Index Pressure Date

Index	Pressure	Date
45	1000	19/Jun/2018 11:01:53
46	984	19/Jun/2018 11:01:54
47	986	19/Jun/2018 11:01:55
48	1014	19/Jun/2018 11:01:56
49	985	19/Jun/2018 11:01:57
50	1023	19/Jun/2018 11:01:58
51	1027	19/Jun/2018 11:01:59

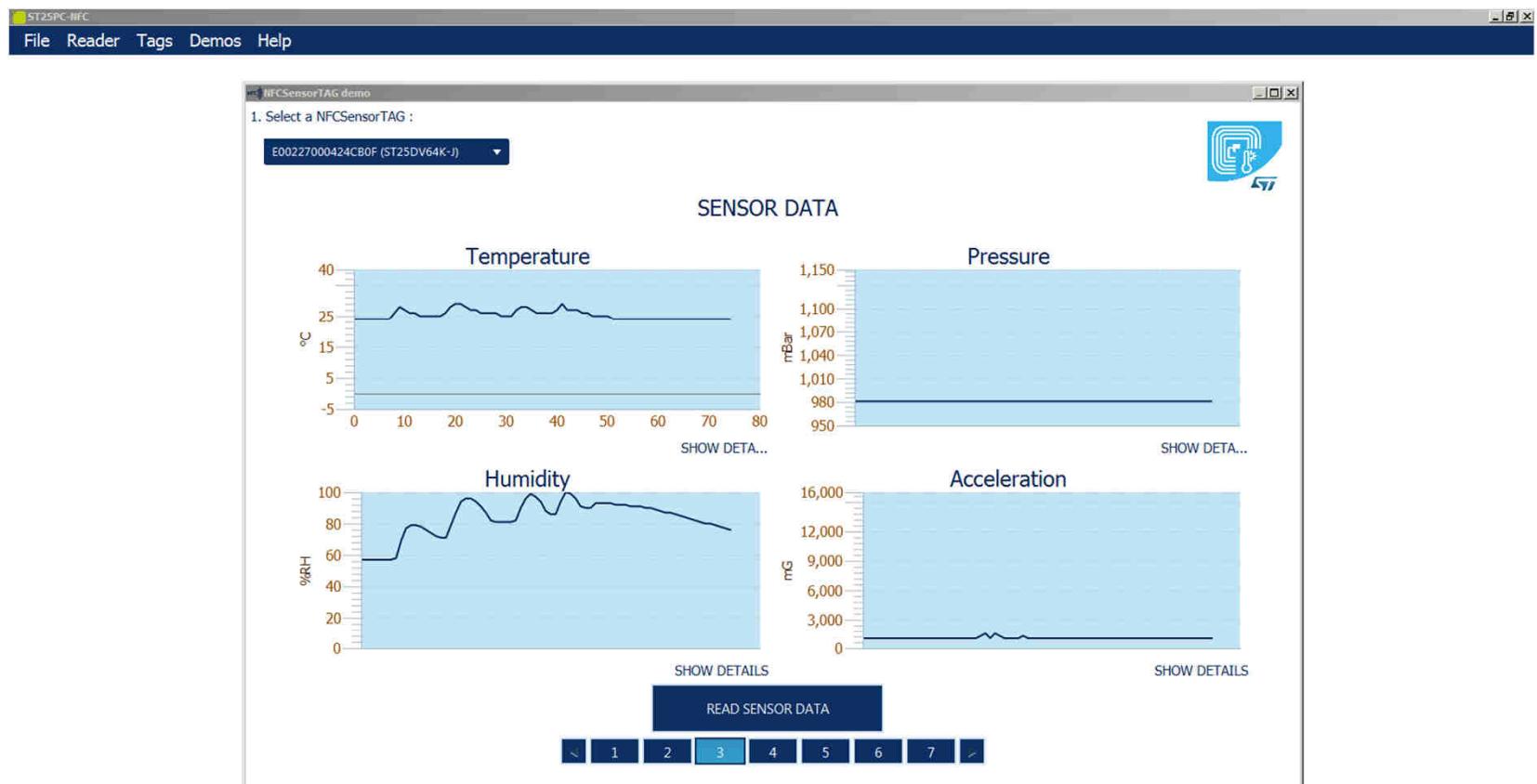
SHOW GRAPH

READ SENSOR DATA

◀ 1 2 3 4 5 6 7 ▶

# My sample data

127



# Process of data logging on the tag in Battery Assisted Mode

```

/* Receive one interrupt from Timer */
if((RFActivityStatus==FIELD_FALLING) | (ForceStart==1)) {
    if(ForceStart) {
        RFActivityStatus=FIELD_FALLING;
        ForceStart=0;
    }

    if( (ReadSensorAndLog & SYNC_EVENT ) ||
        (ReadSensorAndLog & ASYNC_EVENT) ) {
        if(NFCStatus == NFC_STATUS_OFF) {
            PowerOnNFC();
            /* rise time required by VDD_EEPROM for NFC */
            HAL_Delay(200);
            NFCStatus = NFC_STATUS_ON;
        }
    }

    SmarTag_LED_GREEN_On();

    if (NFC_EEPROM_Data.LogMode == SMARTAG_LOGMODE_INACTIVE) {
        /* Do Nothing */
        goto SMARTAG_SLEEP;
    }

    if (!NFC_EEPROM_Data.EnableFlags) {
        /* Do Nothing */
        goto SMARTAG_SLEEP;
    }

    /* beginning of Active log */
    if(ReadSensorAndLog & SYNC_EVENT) {
        /* Init SmarTag sensor */
        if(NFC_EEPROM_Data.LogMode != SMARTAG_LOGMODE_ACTIVE_THS) {
            InitSmarTagSensor();
        }
    }
}

```

If RF field is detected, don't data log. Read in Progress

Turn on LED1

Begin the logging process and subsequently writing to the memory

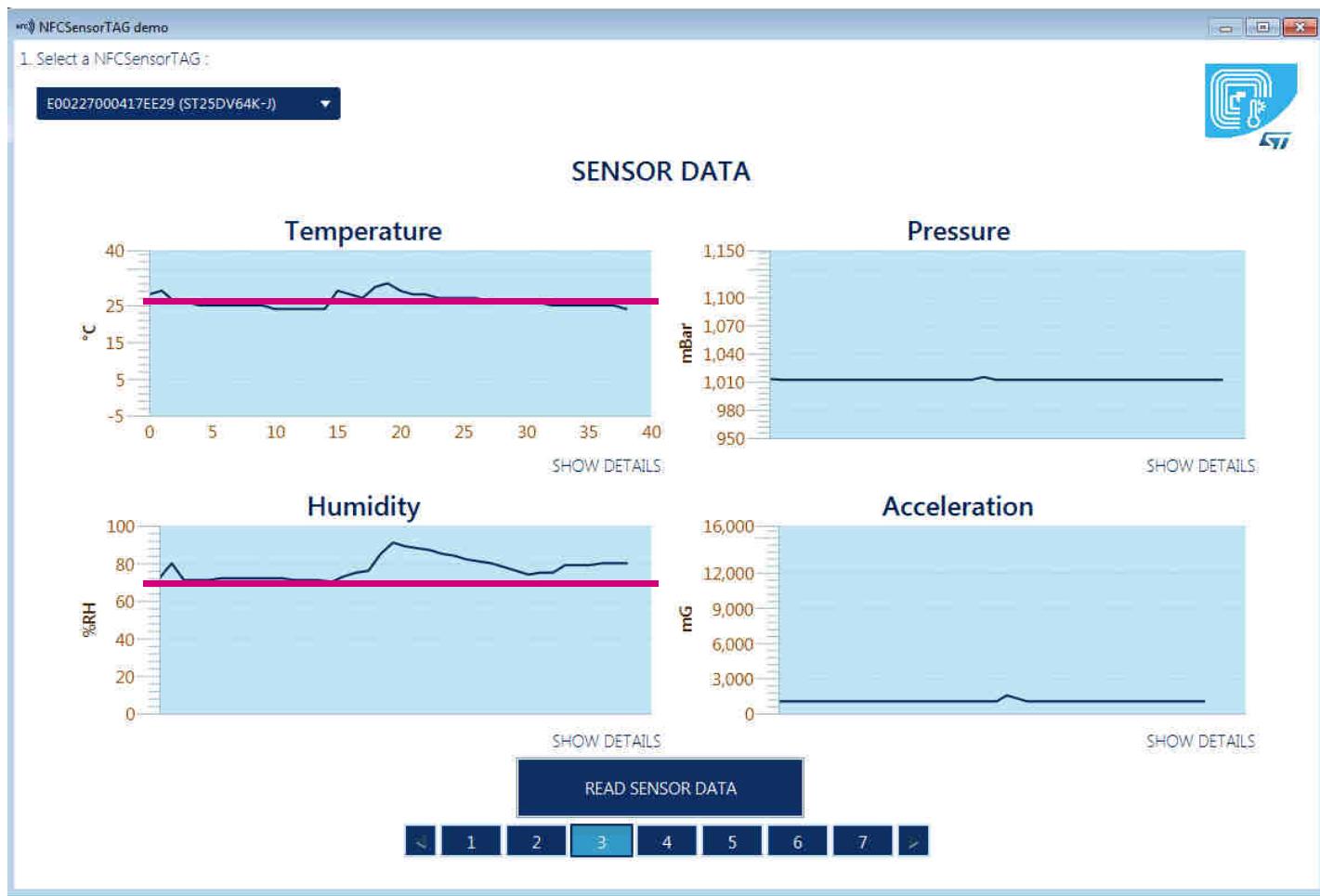
# Setting min-max in Temp and Humidity

129



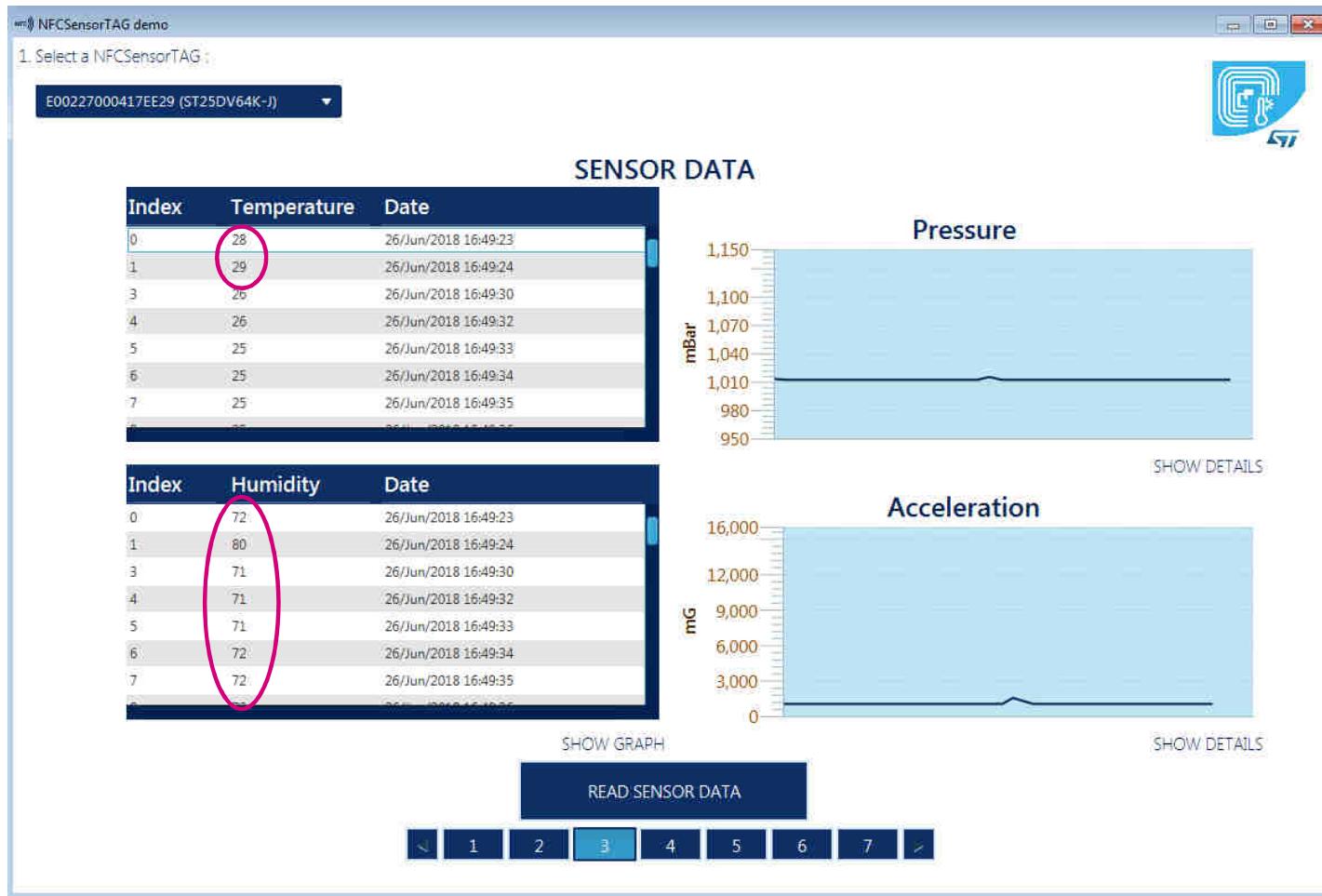
# Data Log only when min-max values

130

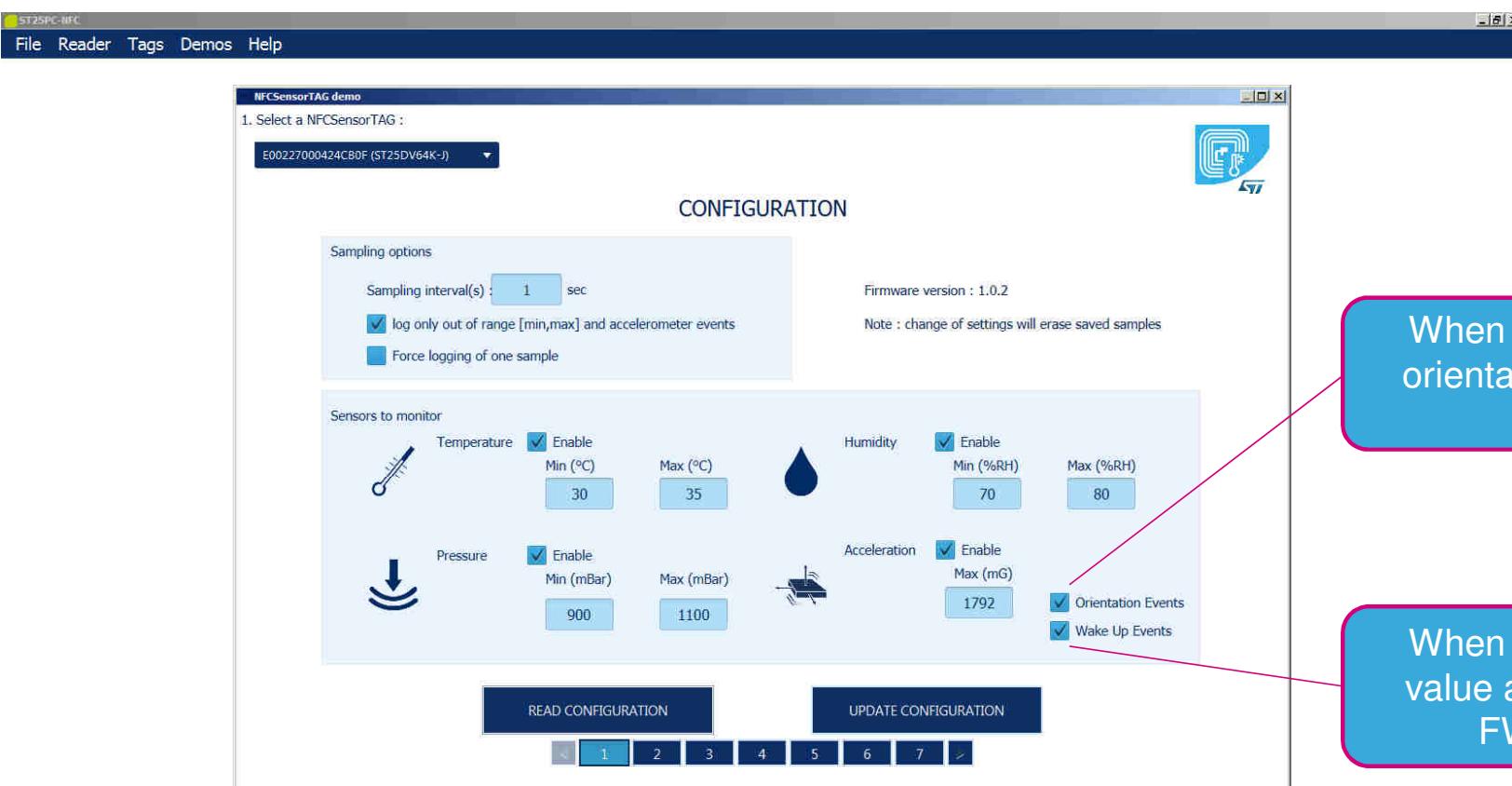


# Data Log only when min-max values

131



# Event Driven Logging



When accelerometer detects orientation change, it logs the data.

When accelerometer detects value above 4G (value set by FW), it logs the data

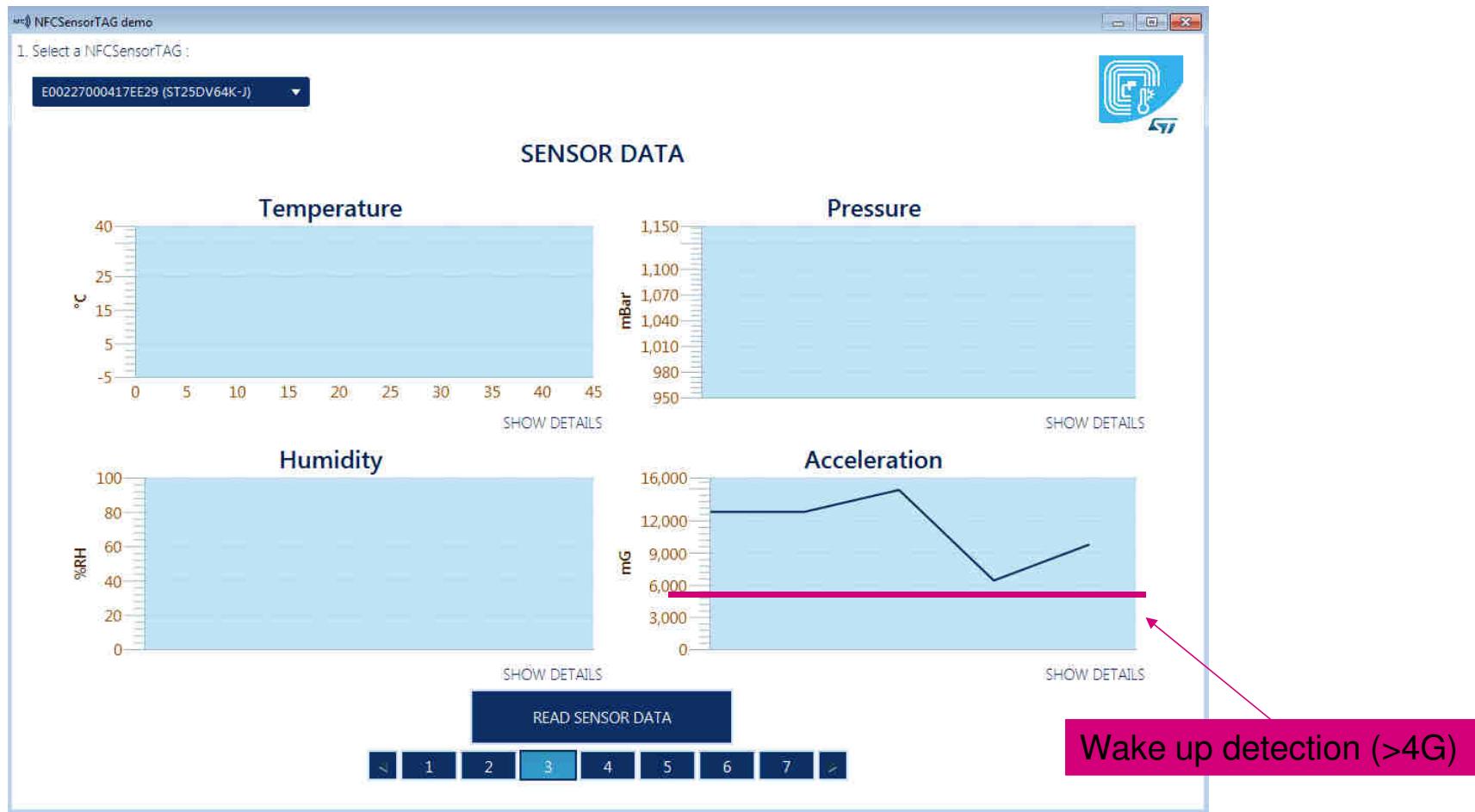
# Setting threshold event for single sensor

133



# Setting threshold event for single sensor

134



# Setting threshold event for single sensor

135

NFCSensorTAG demo

1. Select a NFCSensorTAG :

E00227000417EE29 (ST25DV64K-J)

SENSOR EVENT DATA

Index	Vibration	Event	Details	Date
0	12800	WAKEUP	UP_RIGHT	26/Jun/2018 17:12:07
1	12800	WAKEUP	TOP	26/Jun/2018 17:13:11
2	14848	WAKEUP	TOP	26/Jun/2018 17:14:54
3	6400	WAKEUP	TOP	26/Jun/2018 17:14:55
4	9728	WAKEUP	TOP	26/Jun/2018 17:14:56
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				
35				
36				
37				
38				
39				
40				
41				
42				
43				
44				
45				
46				
47				
48				
49				
50				
51				
52				
53				
54				
55				
56				
57				
58				
59				
60				
61				
62				
63				
64				
65				
66				
67				
68				
69				
70				
71				
72				
73				
74				
75				
76				
77				
78				
79				
80				
81				
82				
83				
84				
85				
86				
87				
88				
89				
90				
91				
92				
93				
94				
95				
96				
97				
98				
99				

READ SENSOR EVENTS

1 2 3 4 5 6 7 < >

# Setting threshold event for single sensor

136



# Setting threshold event for single sensor

137

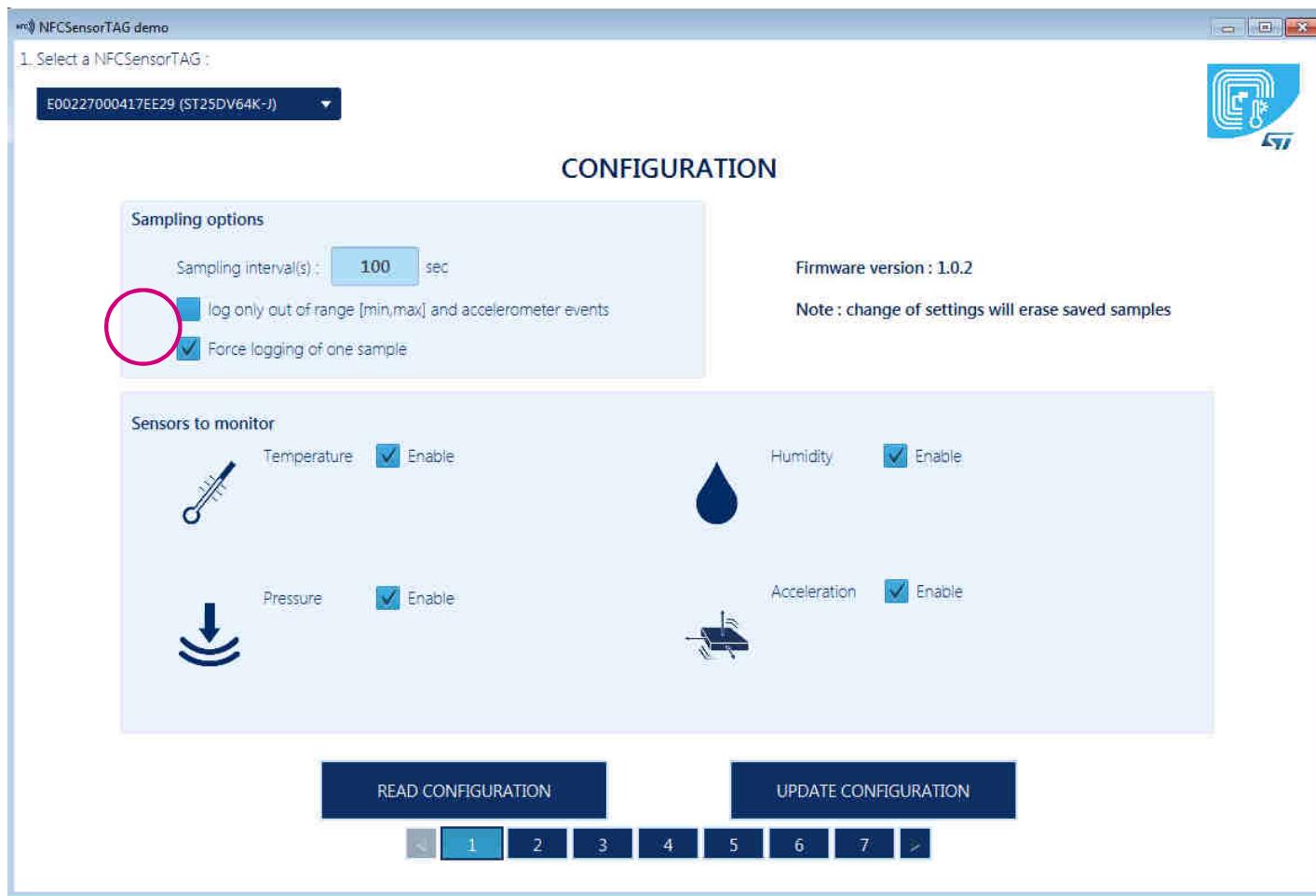
The screenshot shows a Windows application window titled "NFCSensorTAG demo". At the top left, it says "1. Select a NFCSensorTAG:" followed by a dropdown menu showing "E00227000417EE29 (ST25DV64K-J)". On the right side, there is a blue logo featuring a stylized "ST" and a gear-like pattern. Below the title bar, the main area is titled "SENSOR EVENT DATA" and contains a table with the following data:

Index	Vibration	Event	Details	Date
0	1280	ORIENTATION	TOP	26/Jun/2018 17:20:46
1	1280	ORIENTATION	TOP	26/Jun/2018 17:20:47
2	1024	ORIENTATION	TOP	26/Jun/2018 17:20:48
3	1024	ORIENTATION	BOTTOM	26/Jun/2018 17:20:49
4	1024	ORIENTATION	TOP	26/Jun/2018 17:20:51
5	1024	ORIENTATION	TOP	26/Jun/2018 17:20:52

Below the table, there is a button labeled "READ SENSOR EVENTS" and a page navigation bar with numbers 1 through 7.

# Setting threshold event for single sensor

138



# Viewing sensor extreme data

The screenshot shows the ST2SPC-NFC software interface. At the top, there's a menu bar with File, Reader, Tags, Demos, and Help. A sub-menu under Reader is open, showing "NFCsensorTAG demo" and a dropdown menu with the option "E00227000424CB0F (ST25DV64K-J)". On the right side of the screen is a logo for "ST" featuring a stylized "S" and "T".

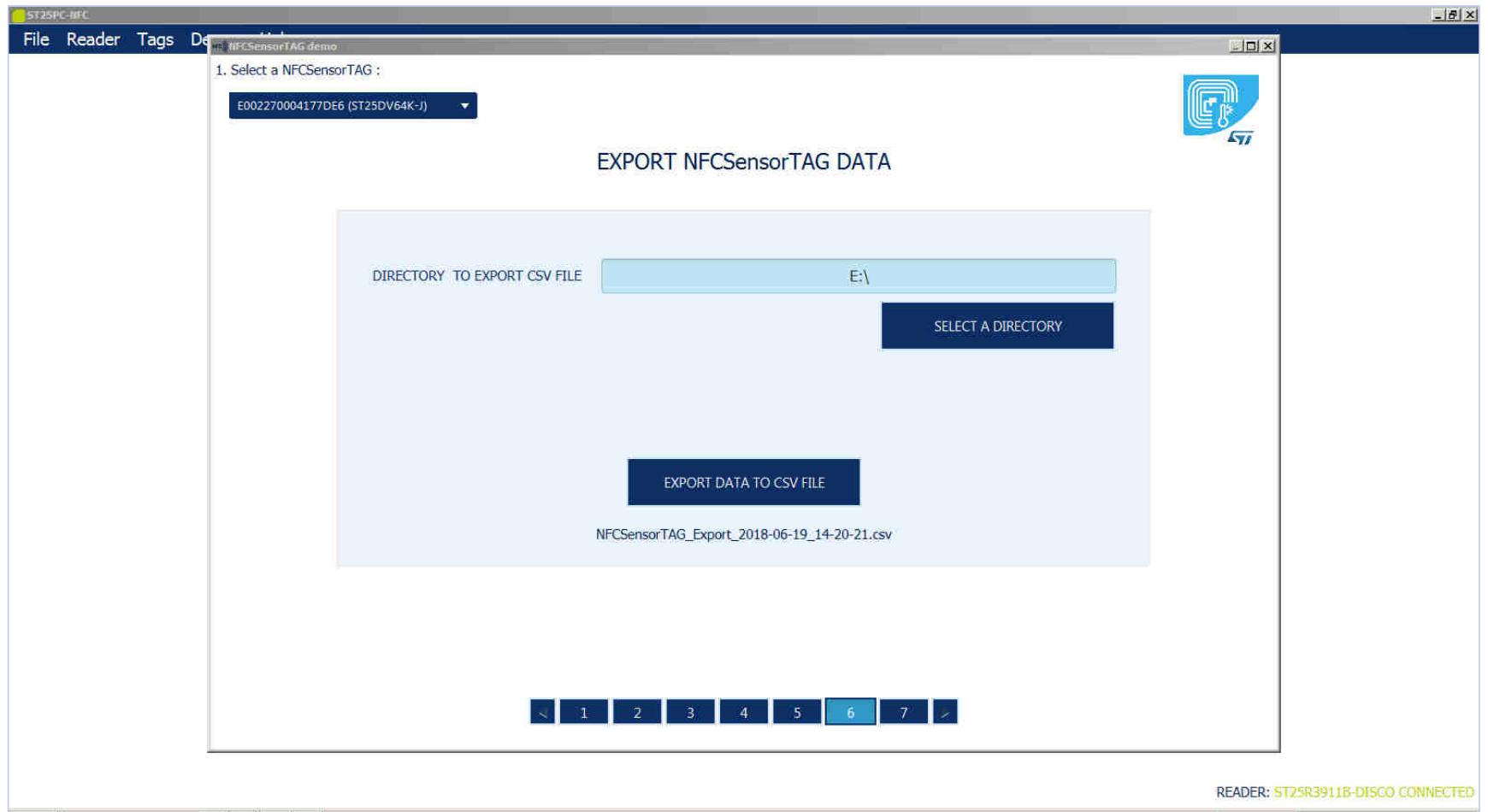
The main window title is "NFCsensorTAG demo" and it displays "1. Select a NFCSensorTAG : E00227000424CB0F (ST25DV64K-J)". Below this, the title "SENSOR EXTREME DATA" is centered.

The data is presented in four sections:

- Temperature (°C)**: Shows Max value 40 at 20/Jun/2018 10:18:00 and Min value 25 at 20/Jun/2018 10:16:39.
- Pressure (mBar)**: Shows Max value 984 at 20/Jun/2018 10:22:22 and Min value 983 at 20/Jun/2018 10:16:54.
- Humidity (%RH)**: Shows Max value 96 at 20/Jun/2018 10:17:08 and Min value 49 at 20/Jun/2018 10:18:03.
- Acceleration (mG)**: Shows Max value 16128 at 20/Jun/2018 10:16:44.

At the bottom center is a blue button labeled "READ SENSOR EXTREME DATA". Below the button is a navigation bar with page numbers 1 through 7, where page 2 is highlighted in blue. To the right of the navigation bar is the text "READER: ST25R3911B-DISCO CONNECTED".

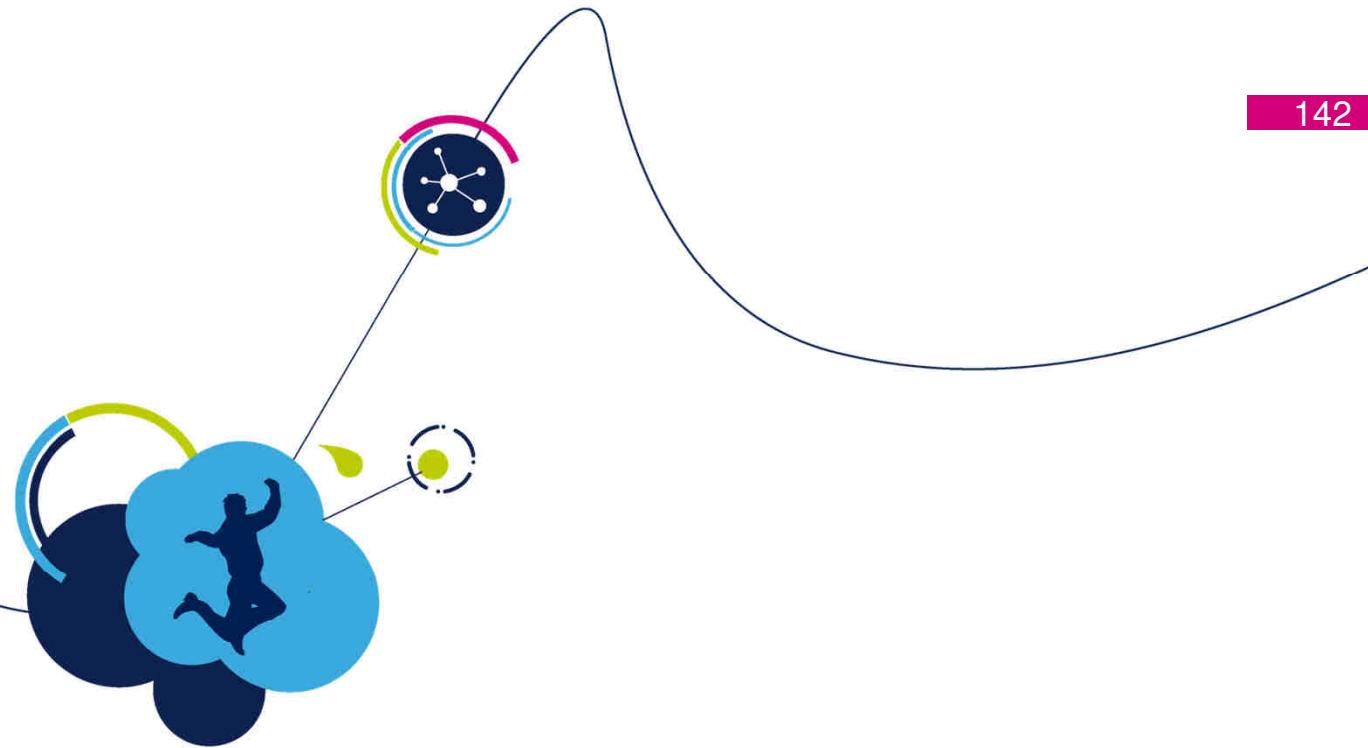
# Exporting Sensor Data



# Using exported data

141

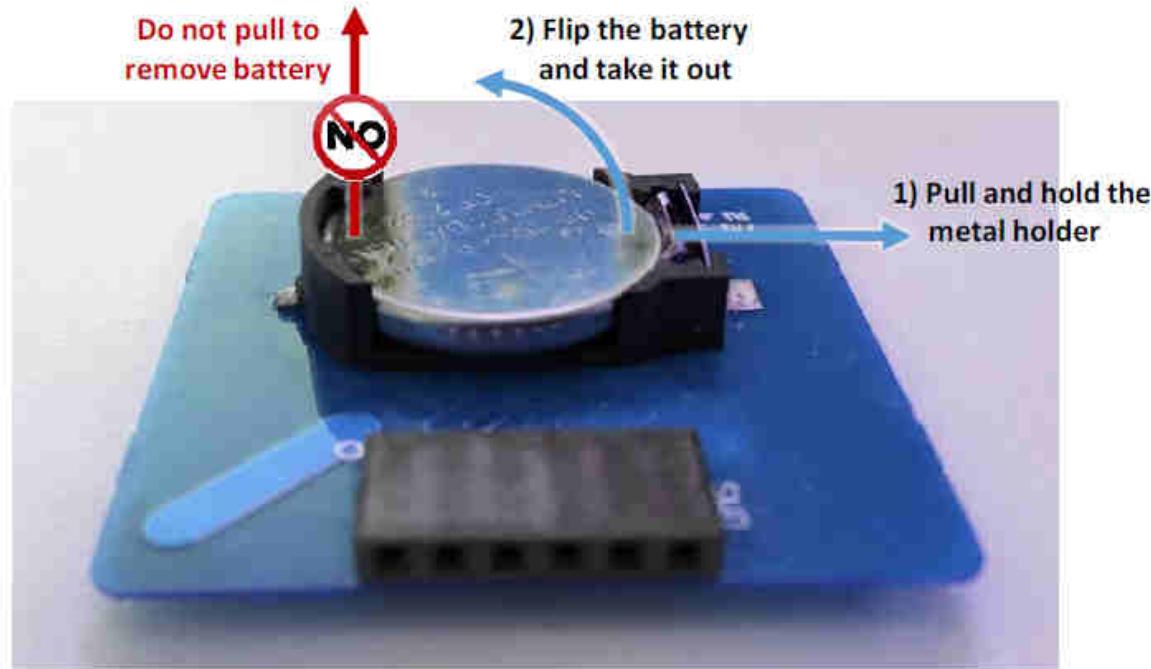
A	B
1 Sampling Interval;1;seconds	
2 Temperature Enabled;Yes	
3 Humidity Enabled;Yes	
4 Pressure Enabled;Yes	
5 Acceleration Enabled;Yes	
6	
7	
8 Threshold;Min;Max	
9 Temperature;24;25	
10 Humidity;40;50	
11 Pressure;900;1100	
12 Acceleration;NaN;1024	
13	
14	
15 Data Log	
16 Date;Temperature (°C);Humidity (%RH);Pressure (mBar);Acceleration (mG)	
17 19/Jun/2018 14:17:01;30;52;974;768	
18 19/Jun/2018 14:17:02;30;53;974;1024	
19 19/Jun/2018 14:17:03;30;53;974;1024	
20 19/Jun/2018 14:17:04;30;53;974;1024	
21 19/Jun/2018 14:17:05;30;53;974;1024	
22 19/Jun/2018 14:17:06;30;53;974;1024	
23 19/Jun/2018 14:17:07;30;53;974;1024	



## Battery-less operation

# Correct way to remove coin cell battery

143



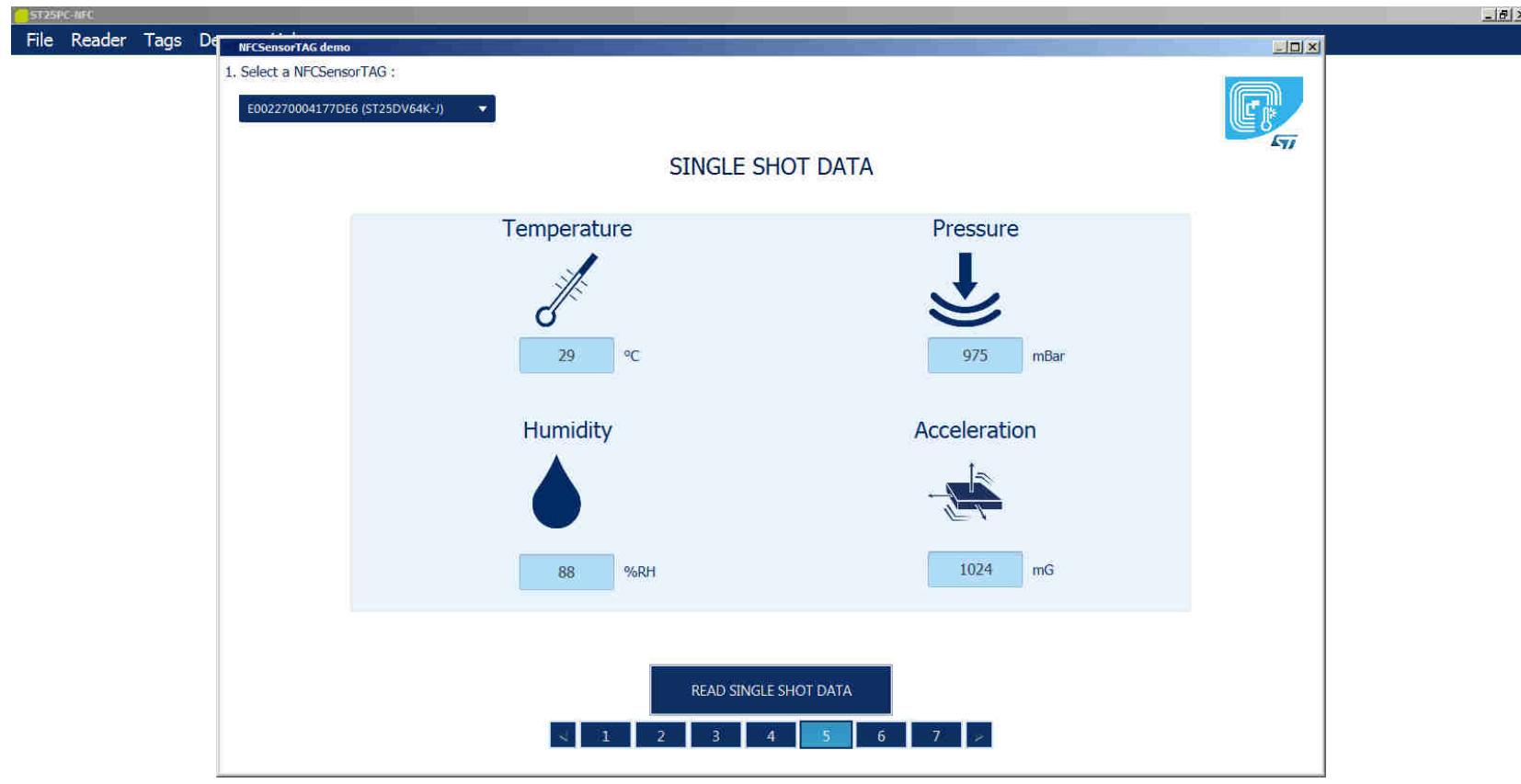
# Exercise

144

- Remove the battery from your sensor tag
- Set for single shot mode from the PC Application Tab #5
- Put the tag in the plastic bag and put it right on top of the reader
- If the placement is good, you will be able to read sensor data.

# Single Shot Reading

145



# Storing 1-shot data on the tag

```
/* Normal Start */  
NfcType5_NDEFInitHeader();  
  
/* Init Environment Variables */  
InitEnvVariables();  
  
/*********************  
/* Like default we will make the single shot */  
/*******************/  
  
/* Init SmarTag sensor */  
InitSmarTagSensor();  
  
MEMS_Sensors_ReadData();  
  
/* De-Init SmarTag sensor */  
DeInitSmarTagSensor();  
  
SensorDataToCompactData();  
  
OneShotWrite();
```

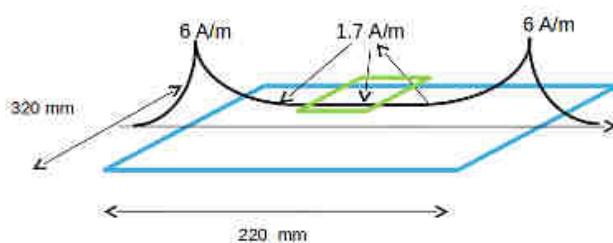
We initialized the ST25DV with an NDEF message header

We compact the sensor data

We store the data in the tag

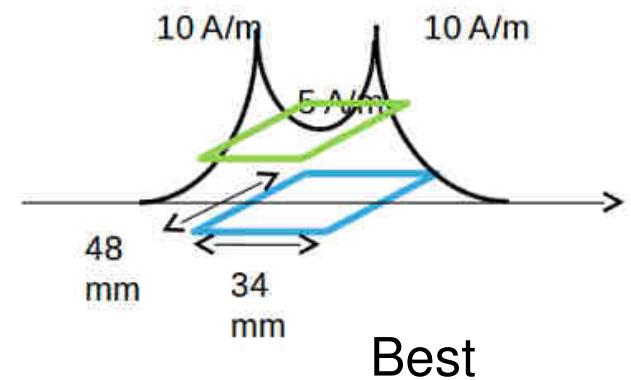
# Tag Placement for Optimal EH

147

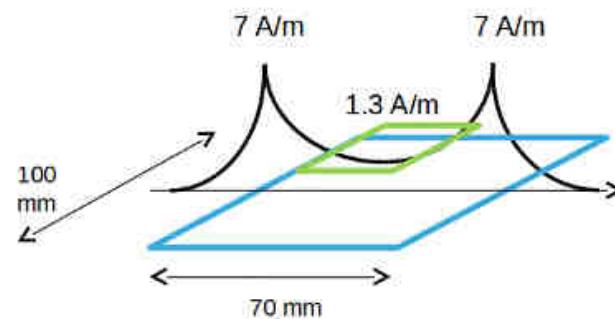


Good

Better



Best



# Optimized Architecture for Battery-less Application

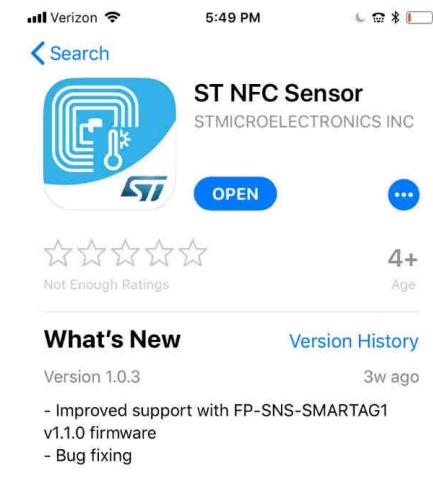
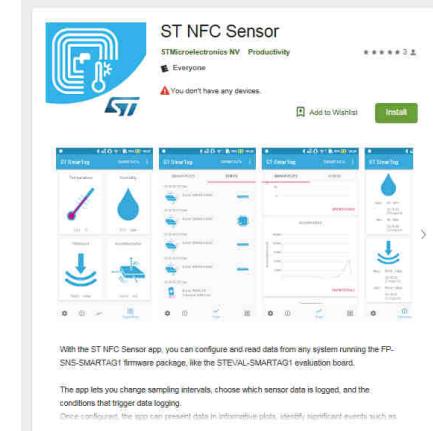
148

- Don't need large EEPROM memory
- Use ST25DV 256bytes buffer (mail box)
- No issue with EEPROM endurance

# Smartphone APK

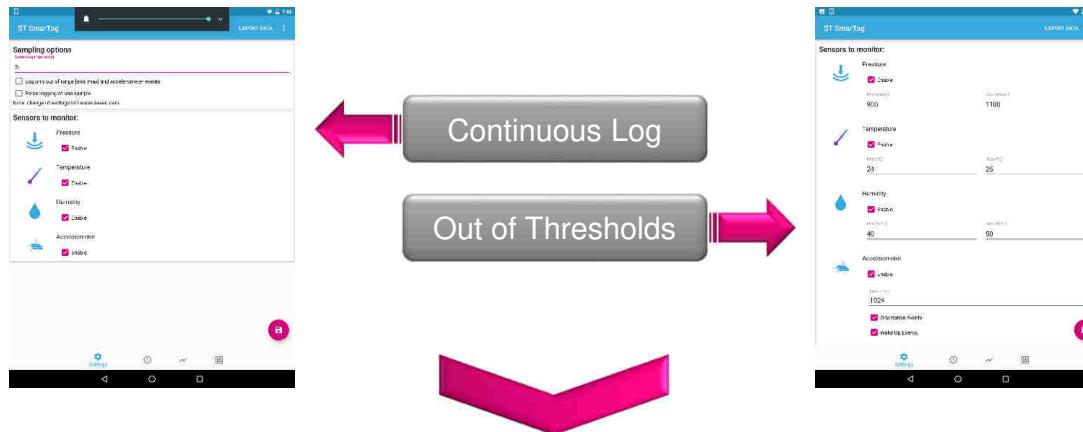


iOS

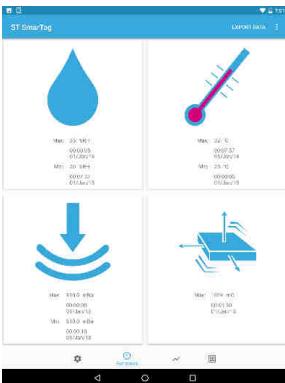


# Android App

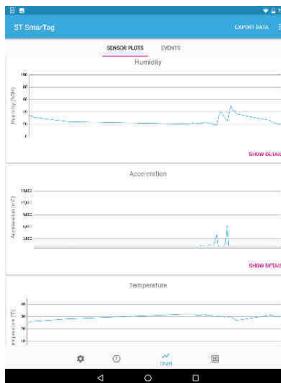
150



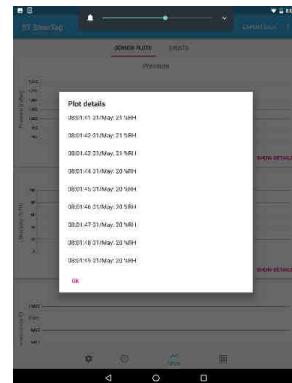
## Min Max Rec



## Data Plot

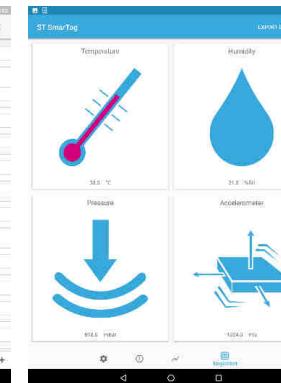


## Plot Details

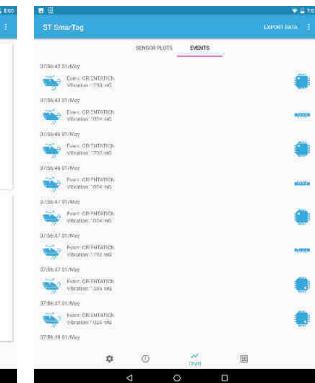


## Data Export

# One Shot EH



## Event Logging



# iOS support of NFC tag reader mode

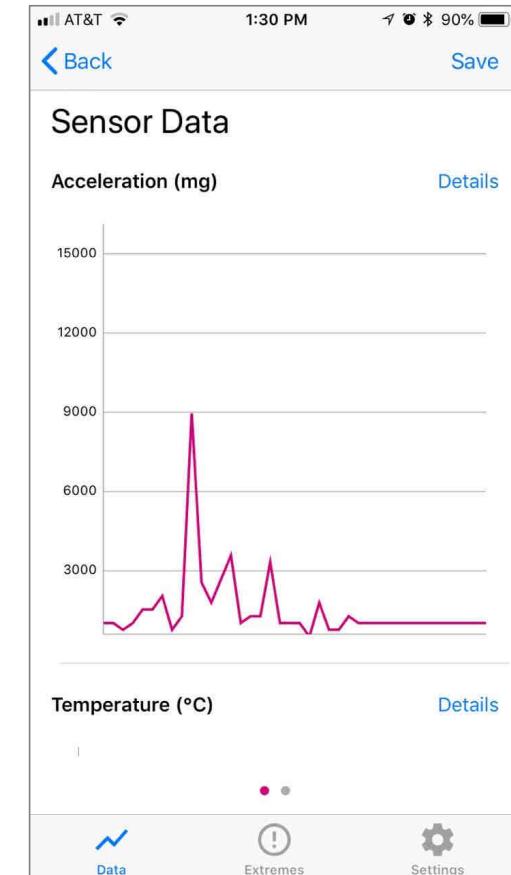
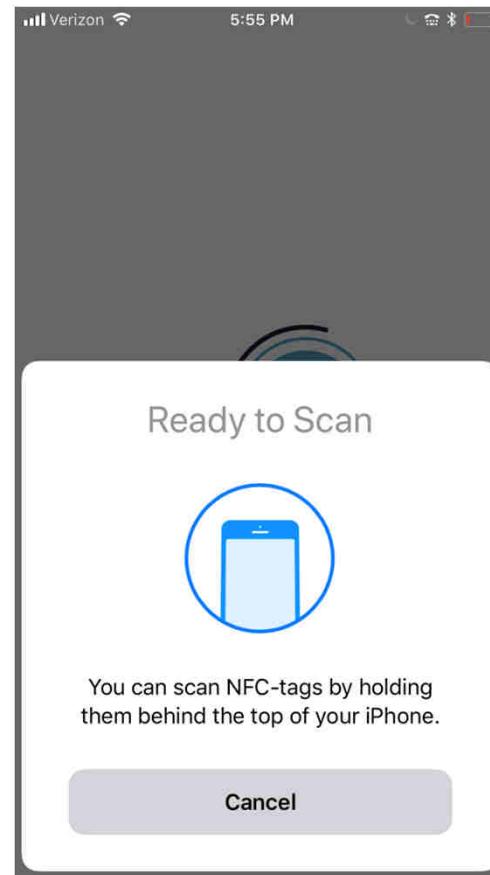
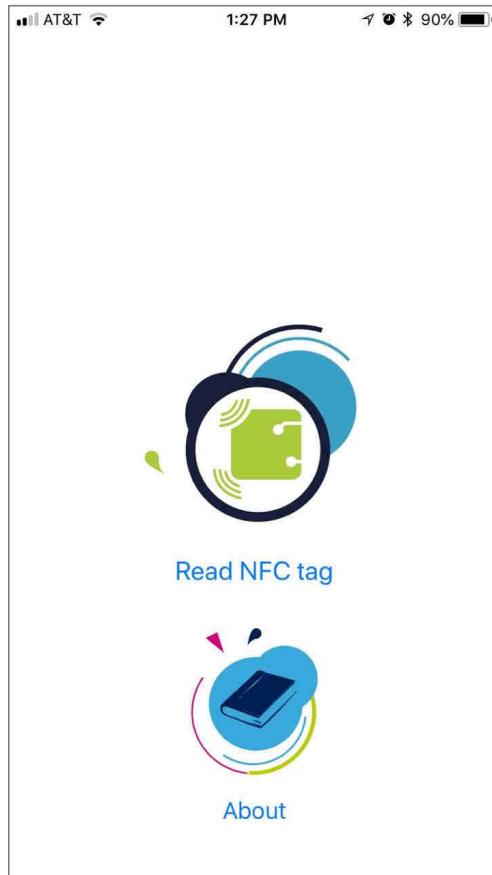
151

- A new core NFC function of Apple iOS11 adds support for NFC tag reading to iPhone7 and iPhone7 Plus as well as the new iPhone8 and 8 Plus and iPhoneX
- iOS11 use cases
  - Read tags of **types 1 through 5** with NDEF (\*)
  - Need iOS application (not «native» as Android)
- Download the NFC Sensor Tag App on iTunes



# iPhone App

152



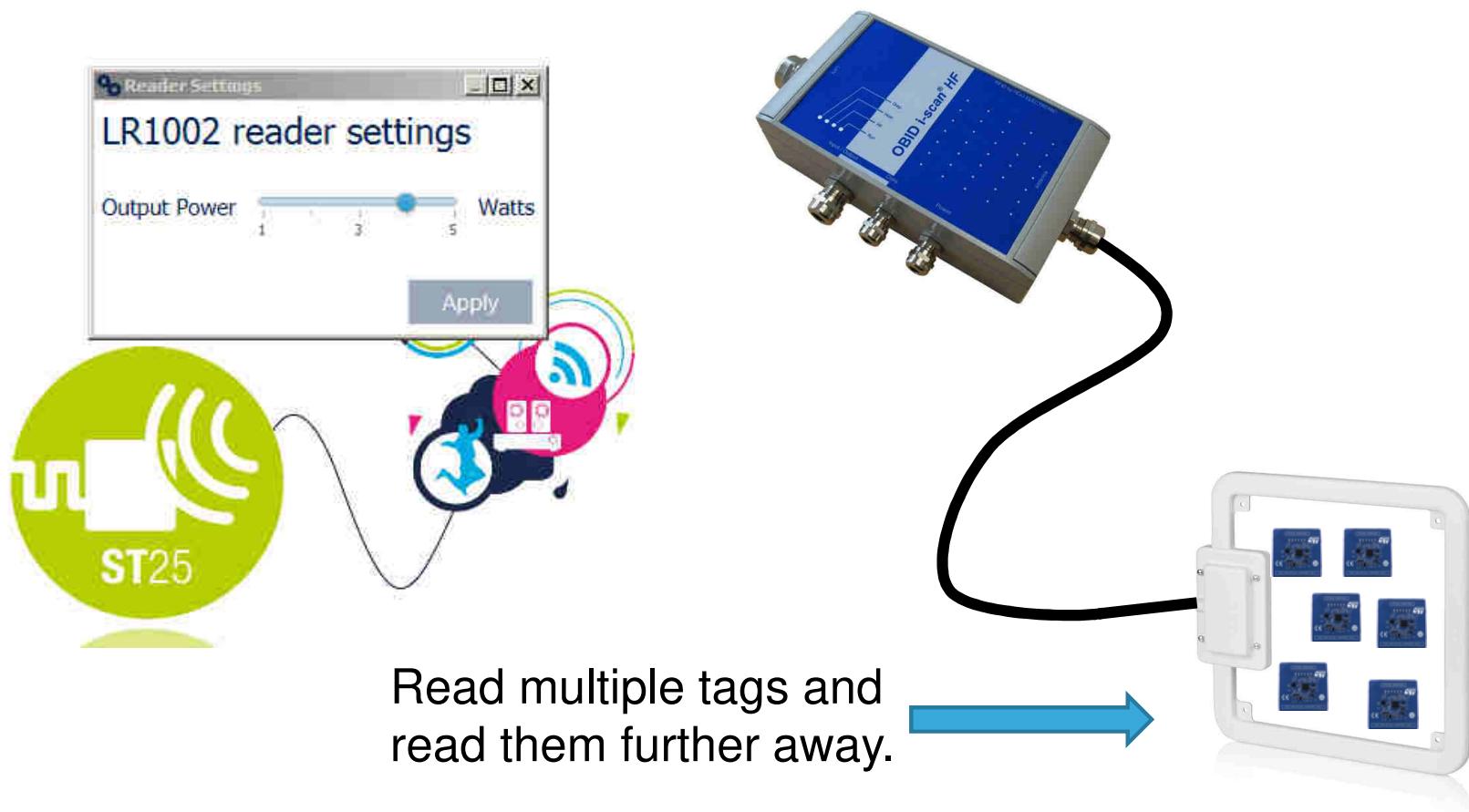
# Iphone Capability

153

- iOS 11 is currently does not allow writing NDEF message. Only reading
- Cannot change default configuration of the sensor tag (e.g. logging time interval and thresholds)

# Using more powerful NFC readers

154



# ST NFC Sensor to Cloud



# What do you need



**UM2427**

User manual

How to use the ST NFC Sensor TAG evaluation board

**Introduction**

The STEVAL-SMARTAG1 NFC sensor tag platform is an NFC-enabled sensor node with inertial and environmental digital MEMS sensors, an STM32 microcontroller and a dynamic NFC tag for communication with NFC readers such as tablets and smartphones.

You can use the NFC-enabled sensor node as the basis for your own designs and as a platform to test and develop NFC-enabled applications with the STM32 Open Development Environment ([STM32 ODE](#)).

The ST NFC Sensor TAG platform is also an evaluation tool to help you assess the performance of the sensors and the capabilities of the NFC dynamic tag embedded on the STEVAL-SMARTAG1 evaluation board.

The board has a small and thin form factor, which makes it particularly useful for deployment in field research and data collection activities that help refine application-specific algorithms.

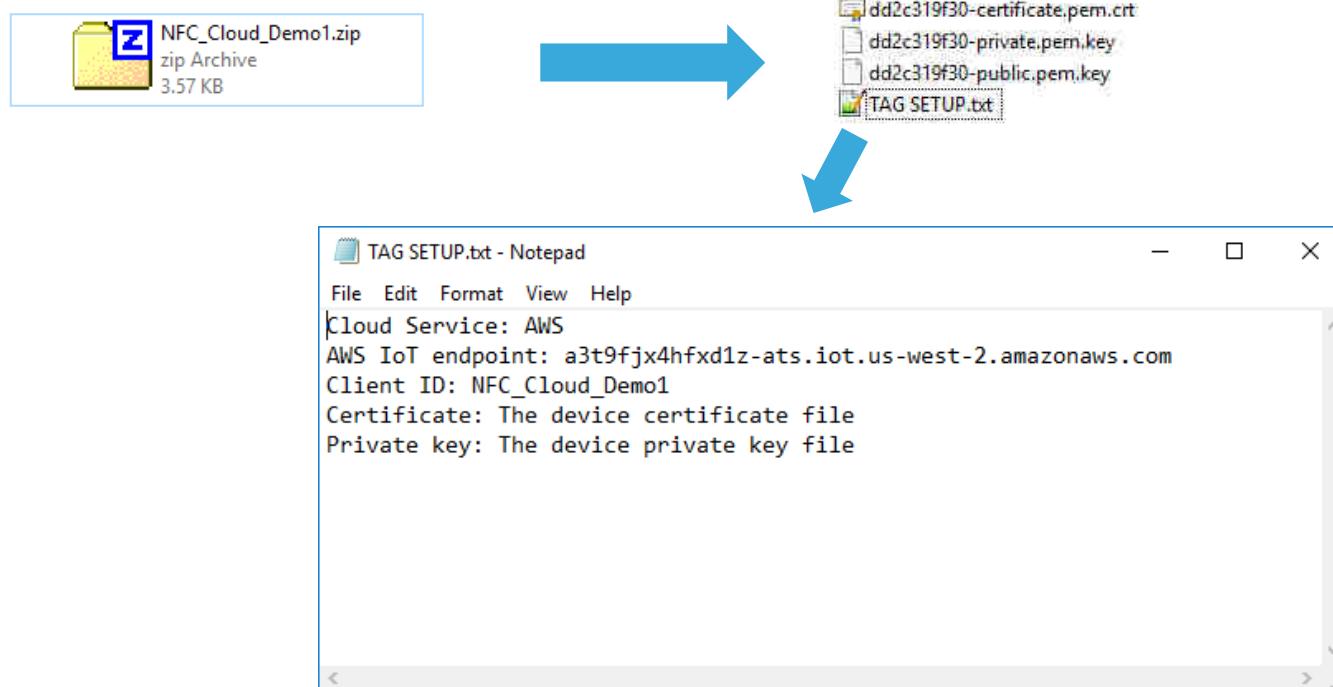
**Figure 1. STEVAL-SMARTAG1 NFC sensor tag**

UM2427 - Rev 1 - June 2016  
For further information contact your local STMicroelectronics sales office.  
[www.st.com](#)

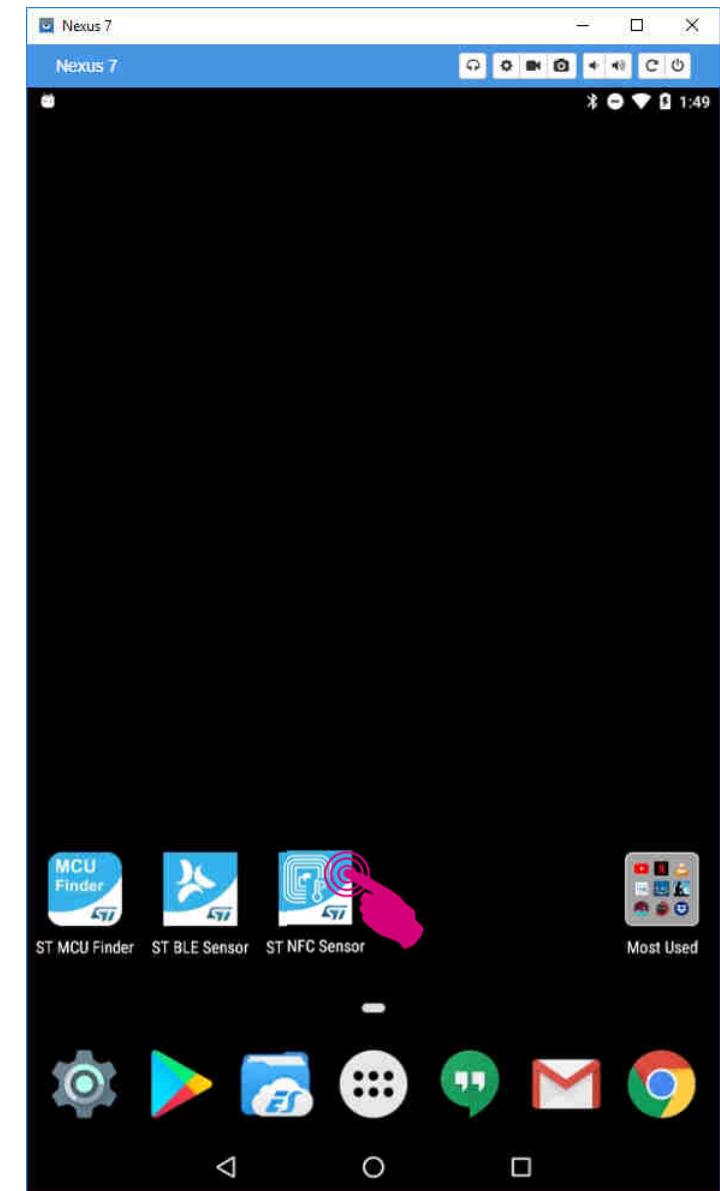
- Android 6.0.1 or greater Tablet/Phone
- ST NFC Sensor V1.1.0
- **STEVAL-SMARTAG1**
- CR2032 Battery
- **UM2427: How to use the ST NFC Sensor TAG evaluation board**
- [NFC\\_Cloud\\_Demo1.zip](#)

# NFC\_Cloud\_Demo1.zip

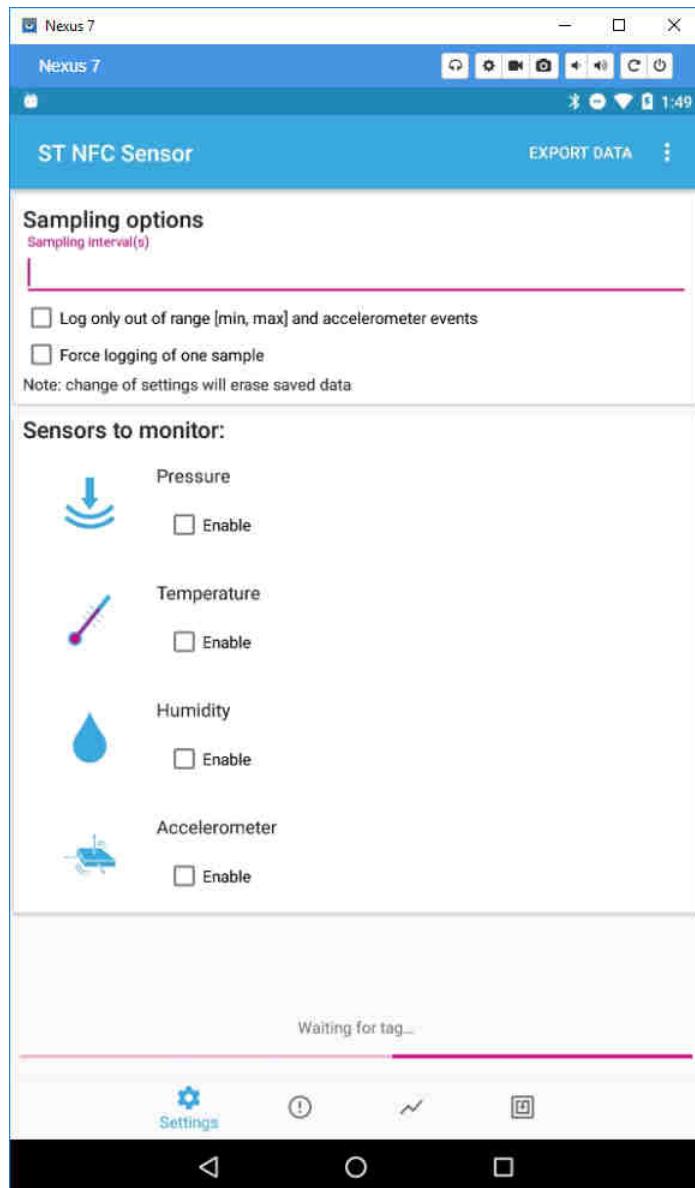
157



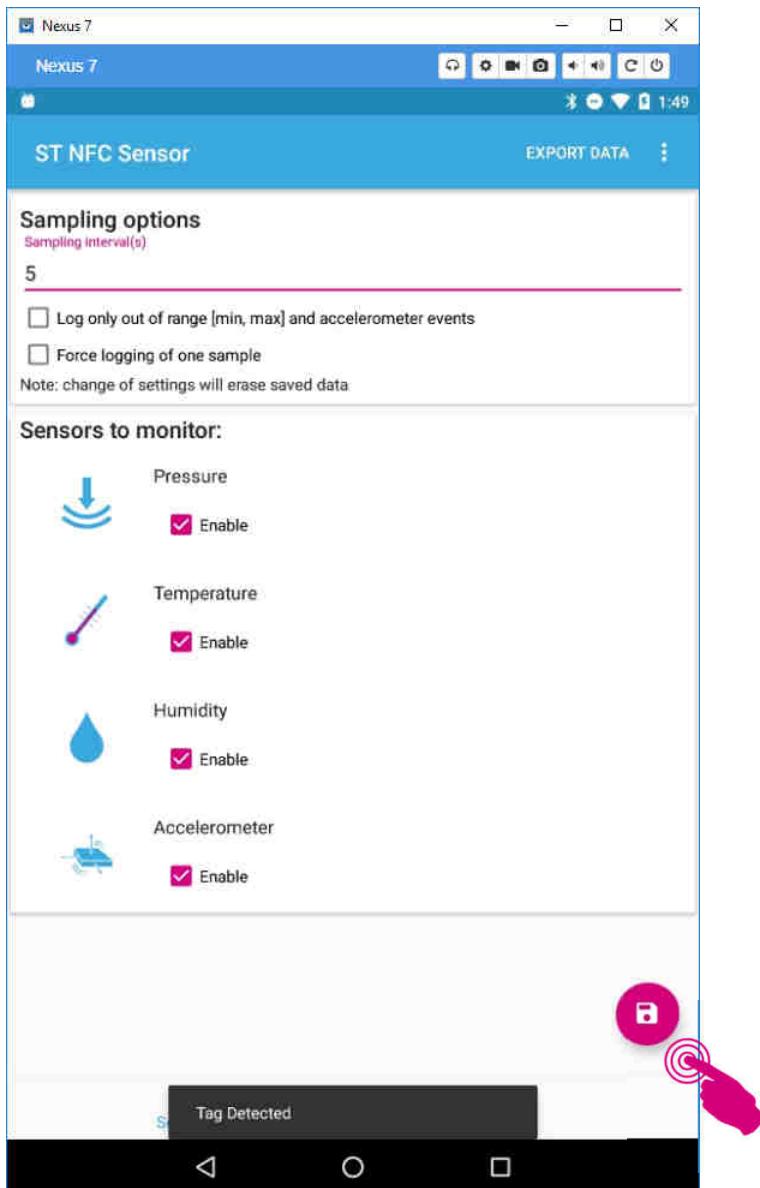
Copy the extracted folder in a known location on the Android Tablet (you'll need this later)



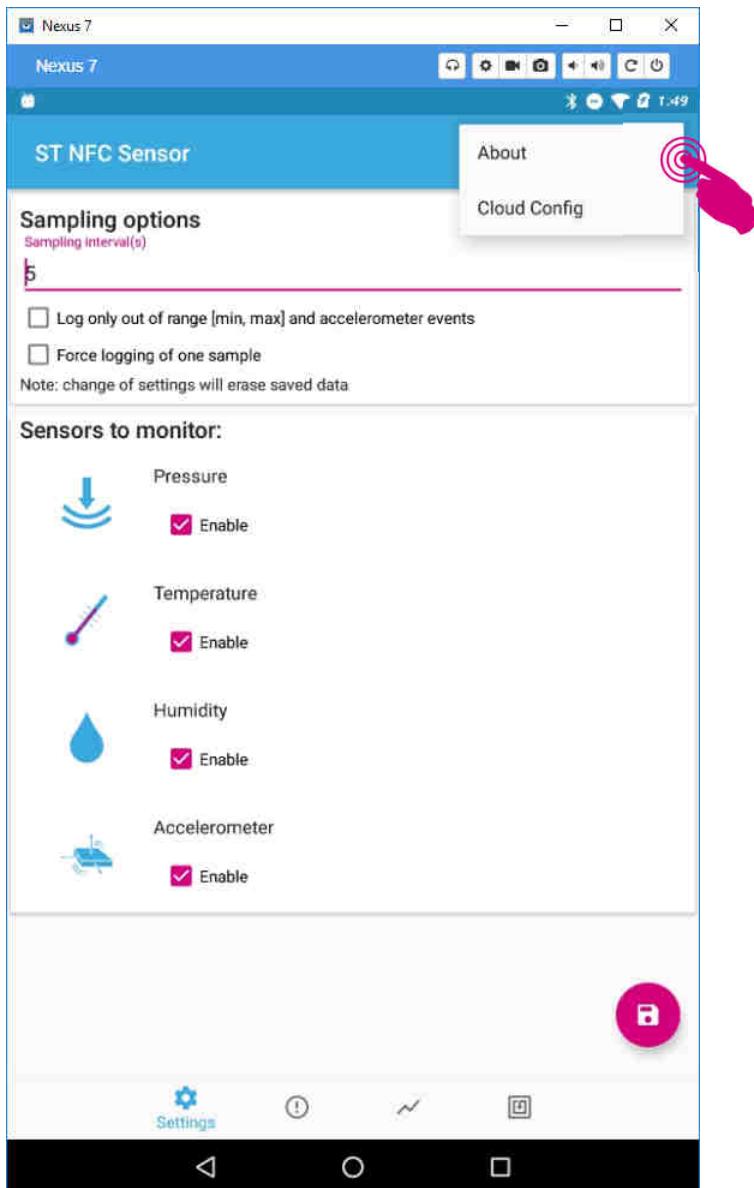
- Open ST NFC Sensor



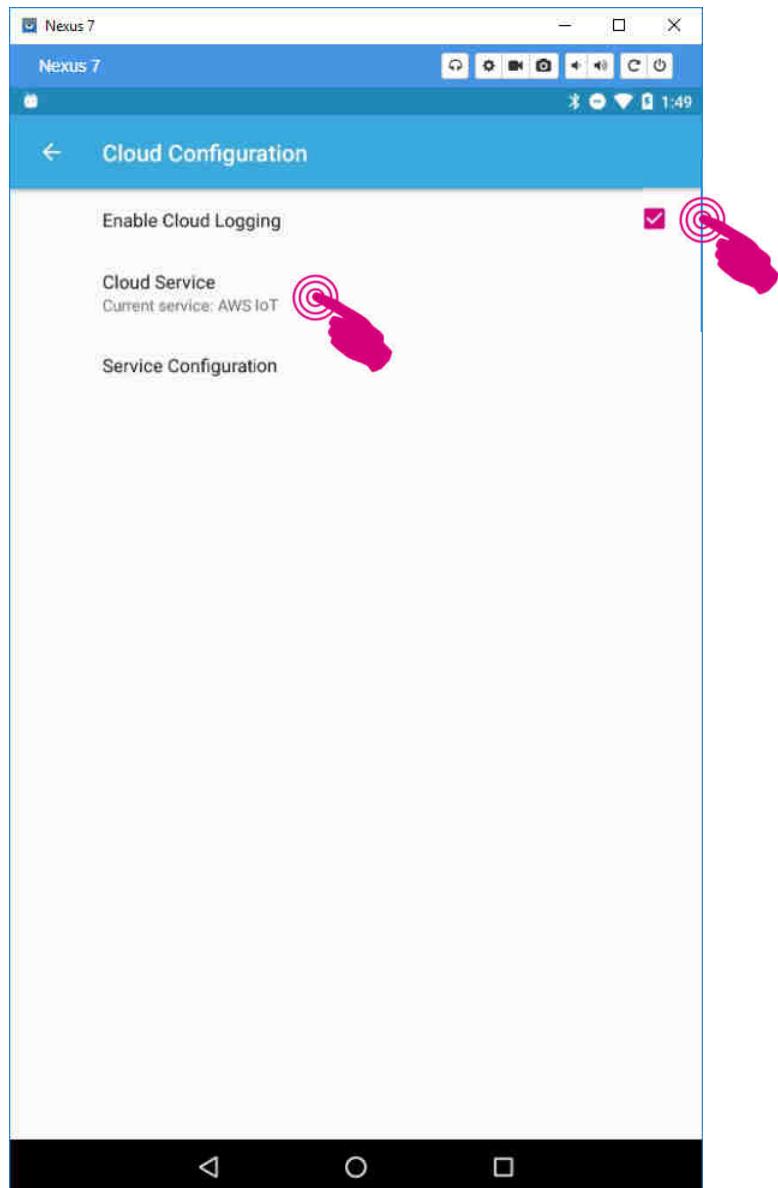
- While on Settings TAB, tap the ST NFC Sensor



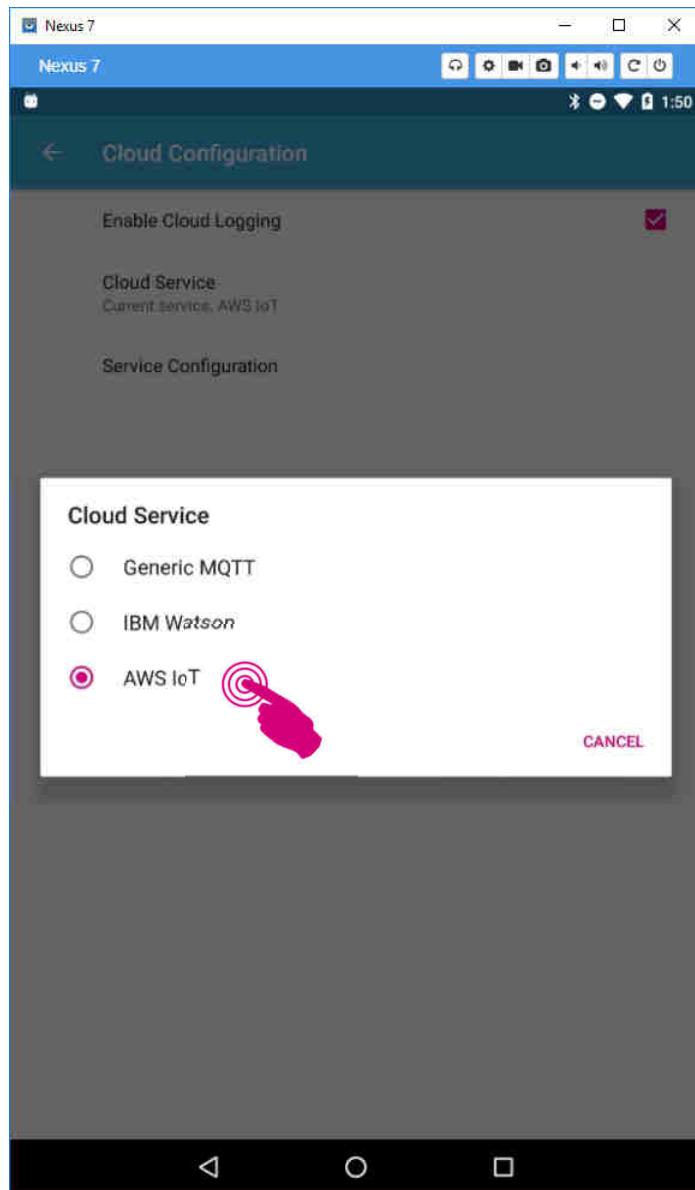
- Previous configuration will be loaded
- You can change to:
  - Sampling option 5s
  - All sensor enabled
- Write tapping on Disc pink icon with ST NFC Sensor in range



- Tap on the 3 dots on top right corner
- Tap on **Cloud Config**



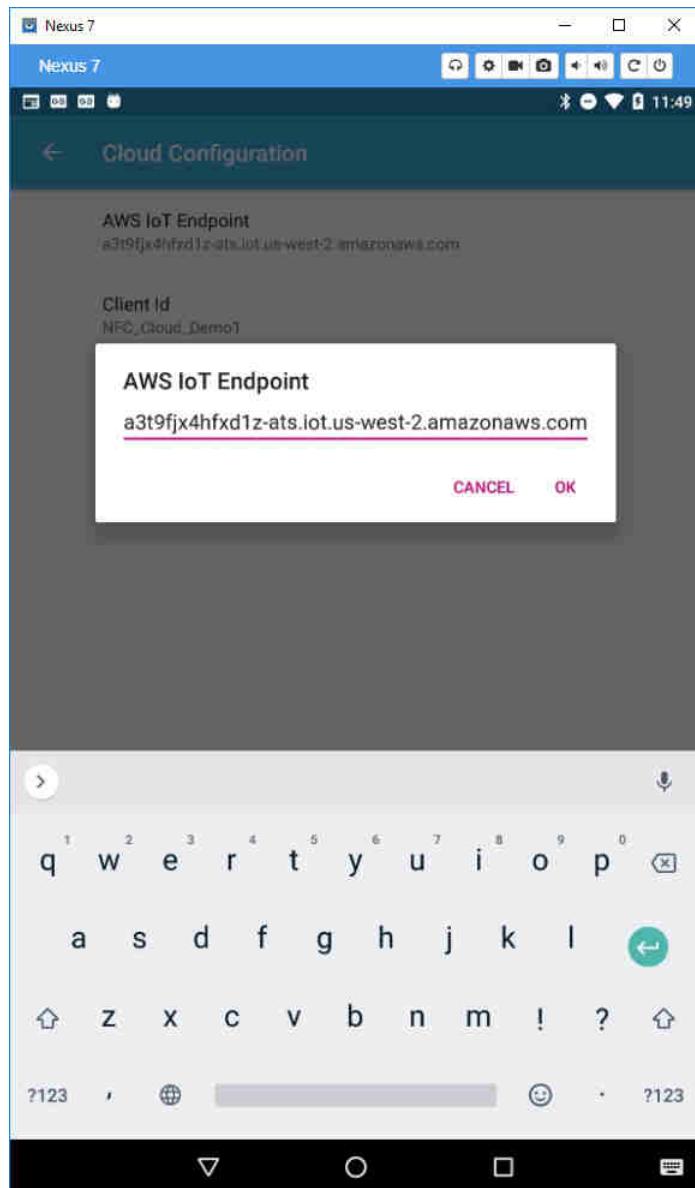
- Tap on **Enable Cloud Logging**
- Tap on **Cloud Service**



- Tap on **AWS IOT**

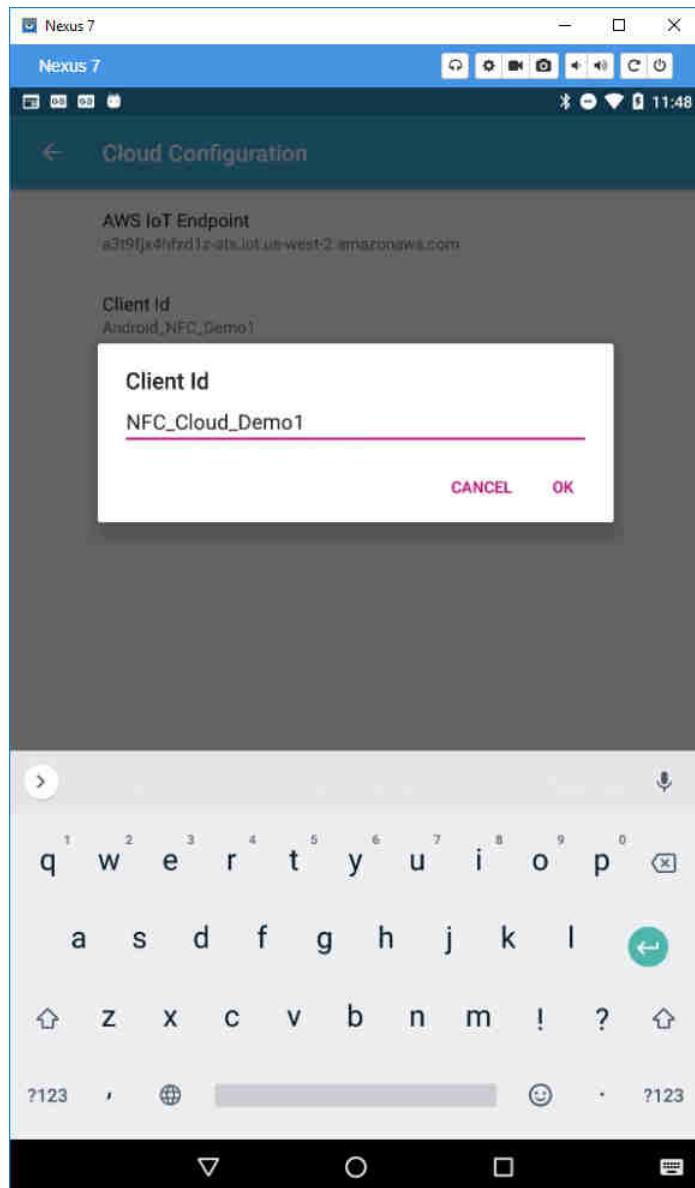


- On the Cloud configuration tool we'll now need to insert
  - **AWS IoT Endpoint**
  - **Cliend ID**
  - **Certificate**
  - **Private Key**
- Recommendation is to have a txt file with AWS IoT Endpoint and client ID to paste in the field (TAG SETUP.txt)
- Certificate and Private Key in separate files (downloaded from AWS).



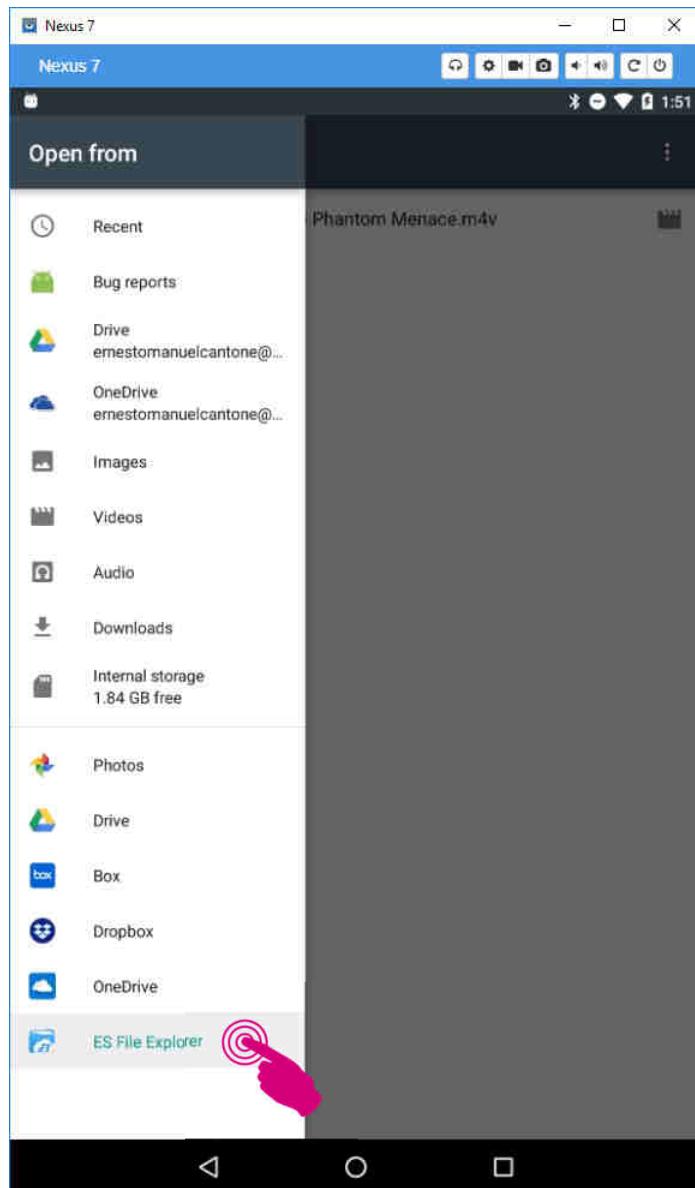
- Cut and paste AWS IoT Endpoint

a3t9fjx4hfxd1z-ats.iot.us-west-2.amazonaws.com

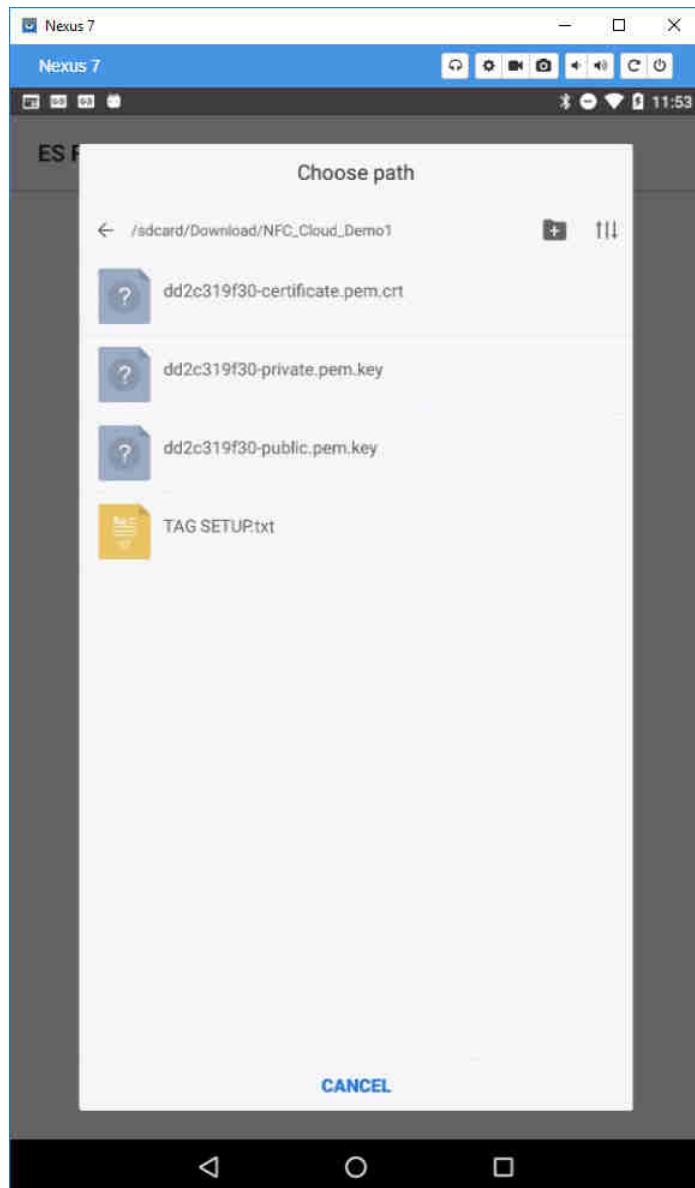


- Cut and paste Client Id

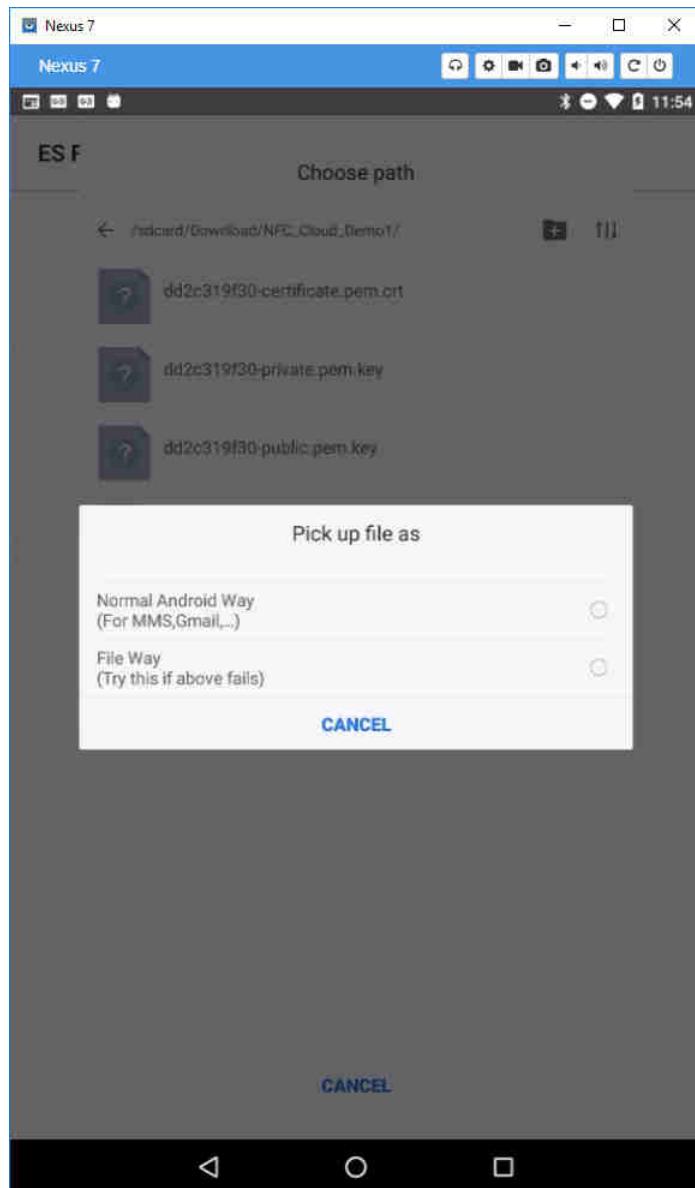
Android\_NFC\_Demo1



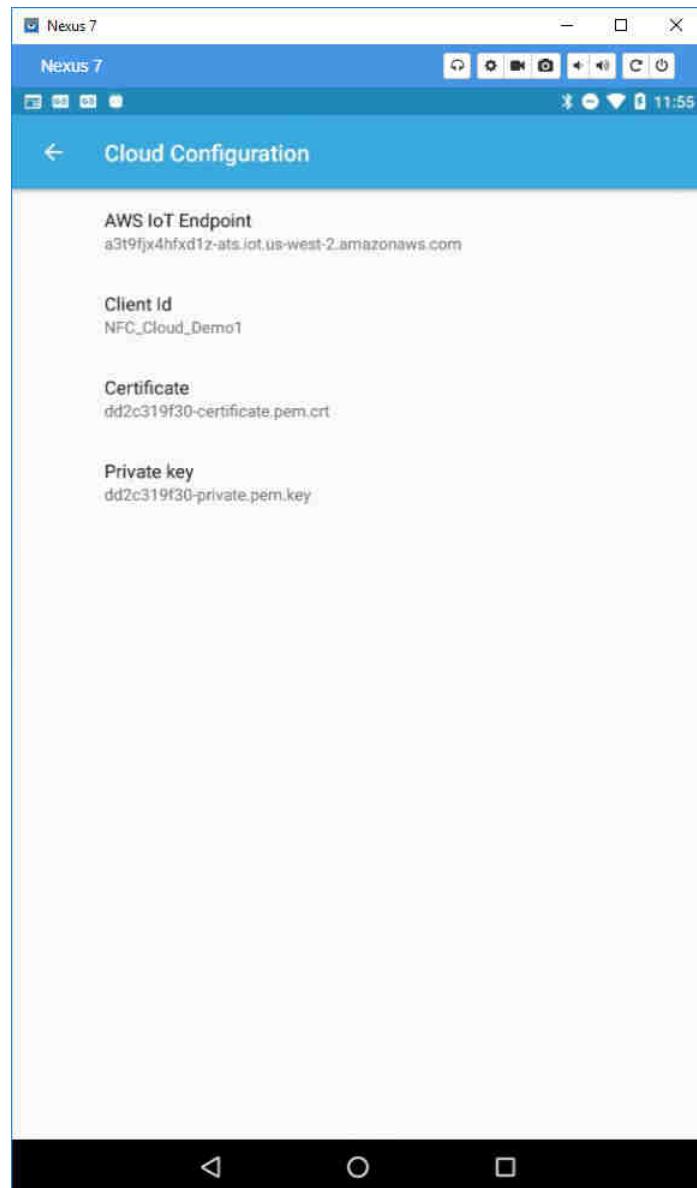
- Browse your tablet with a File Explorer app (ES file Explorer in this example) to locate the correct certificate/key
  - NOTE: a cloud location can also be used (Box, Dropbox, Google Drive, OneDrive, ...)



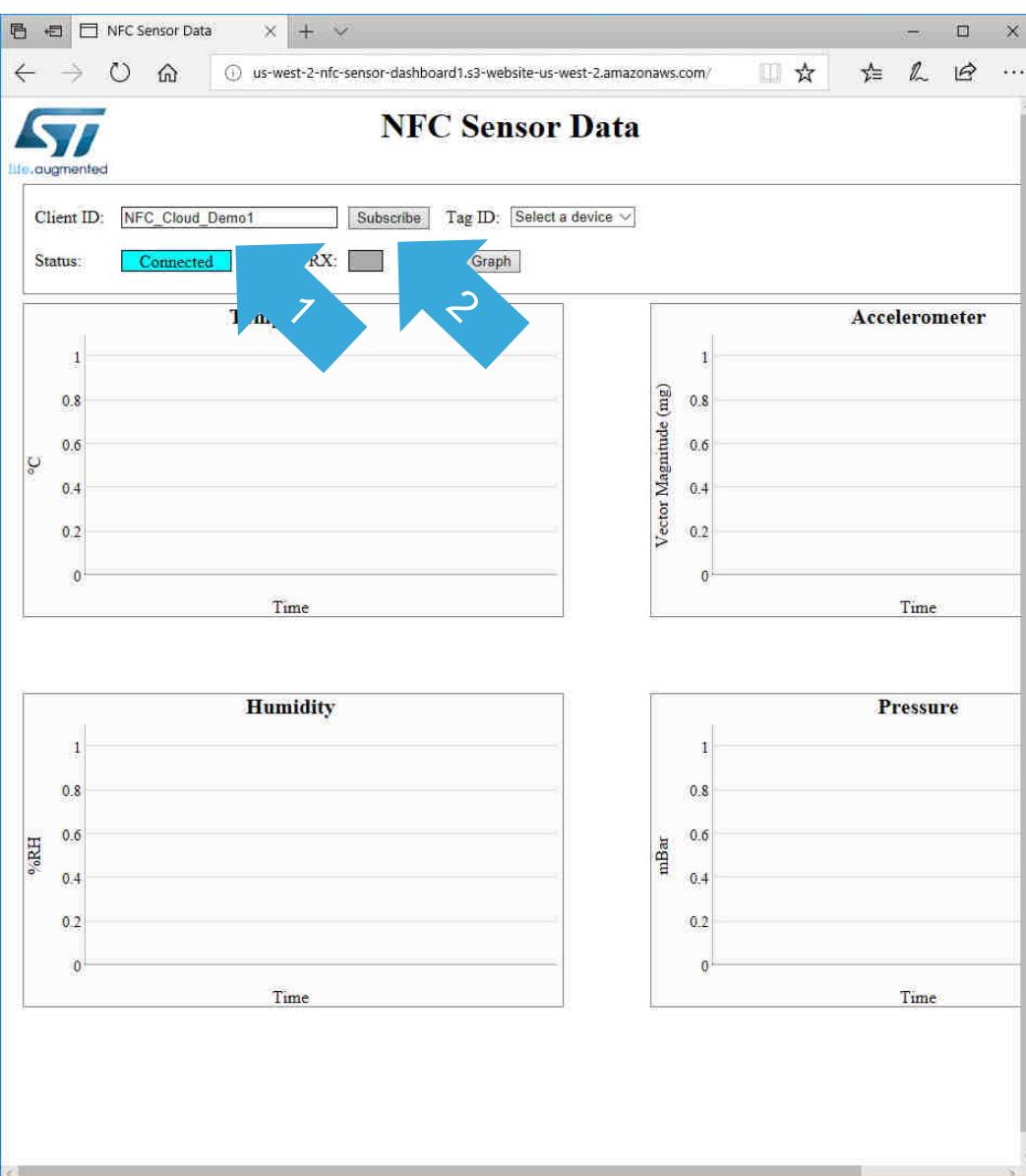
- In this example we stored the files in the folder `..\Download\NFC_Cloud_Demo1`
- In this location we have the following files:
  - `dd2c319f30-certificate.pem.crt`
  - `dd2c319f30-private.pem.key`
  - `dd2c319f30-public.pem.key`



- Link the Certificate and Private Key with the provided methods
  - In the example “File Way” was used



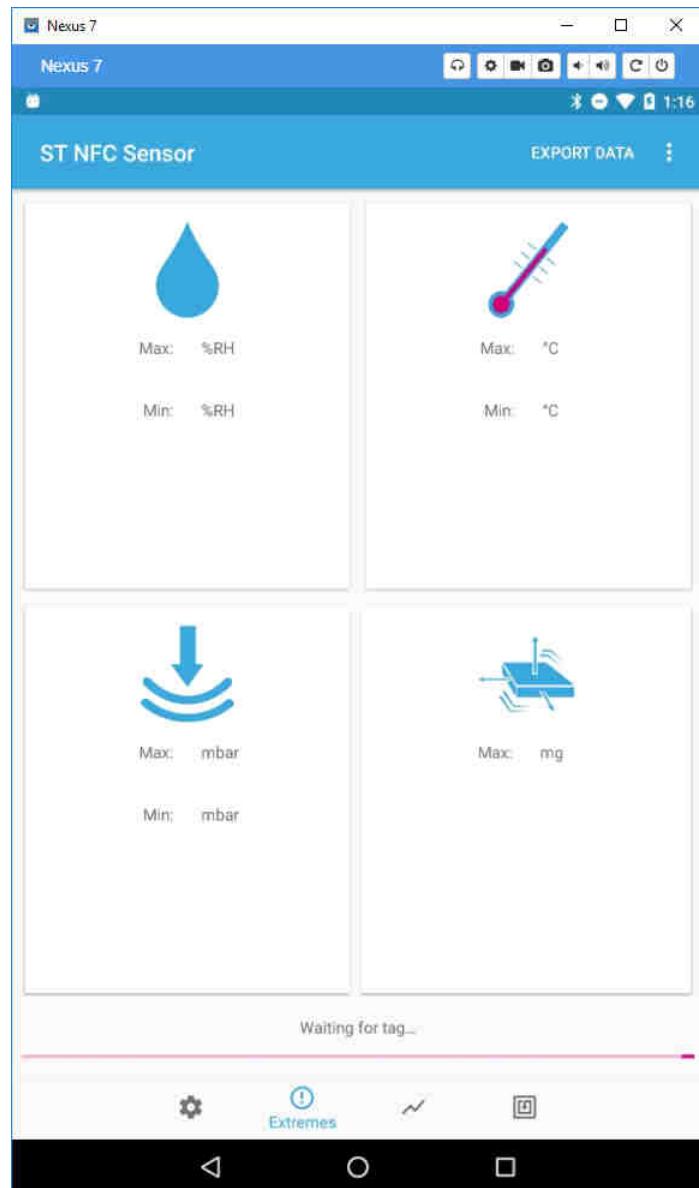
- Once every field is filled you can go back and they will be stored in the app for next access



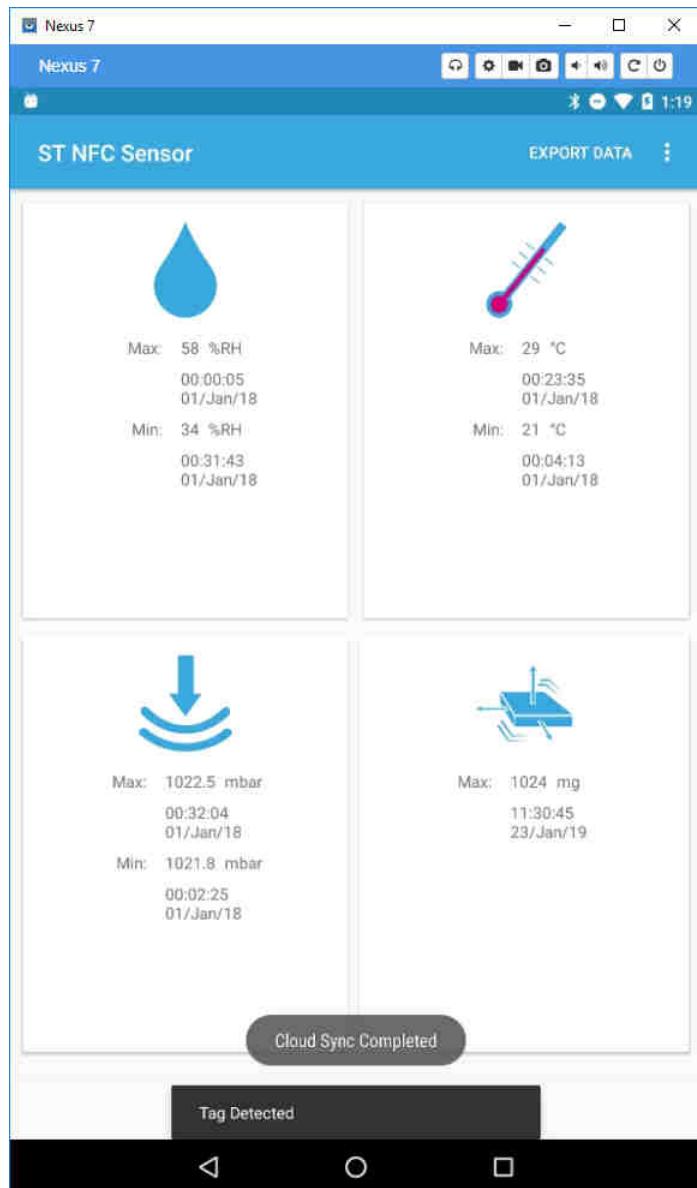
- On a browser of your choice (Chrome is visualized) go to

<http://us-west-2-nfc-sensor-dashboard1.s3-website-us-west-2.amazonaws.com/>

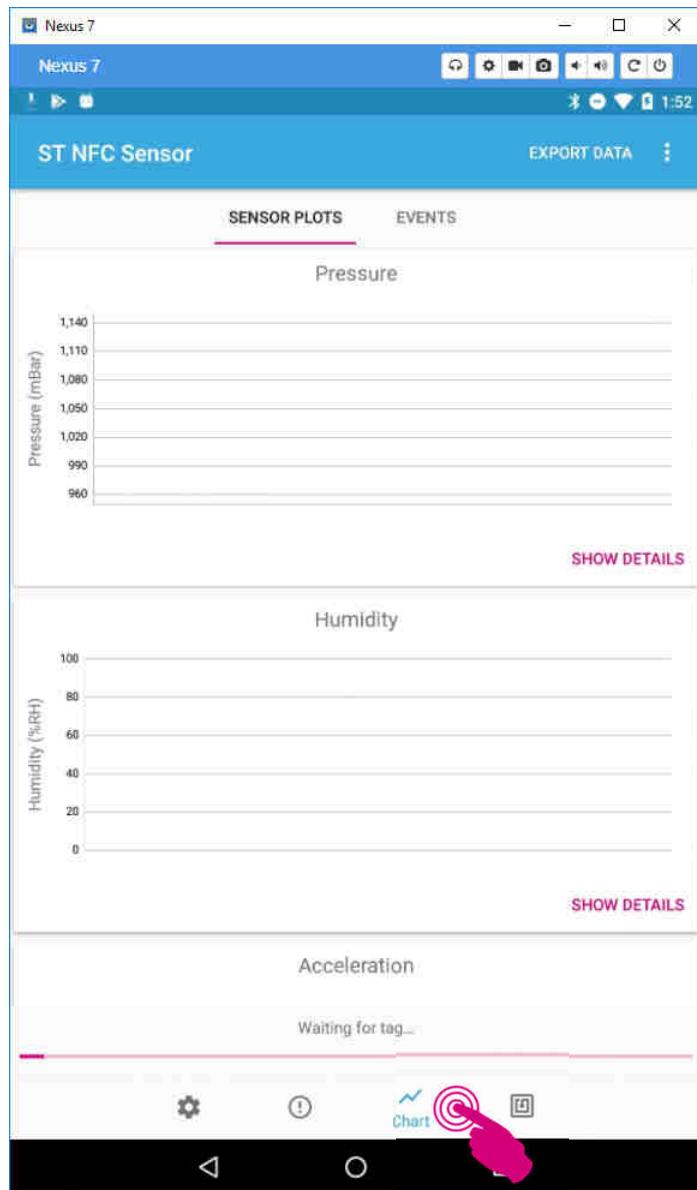
- On the Client ID field insert  
**NFC\_Cloud\_Demo1**
- Click Subscribe



- Select **Extremes** Tab



- Tap Tag to set Extremes



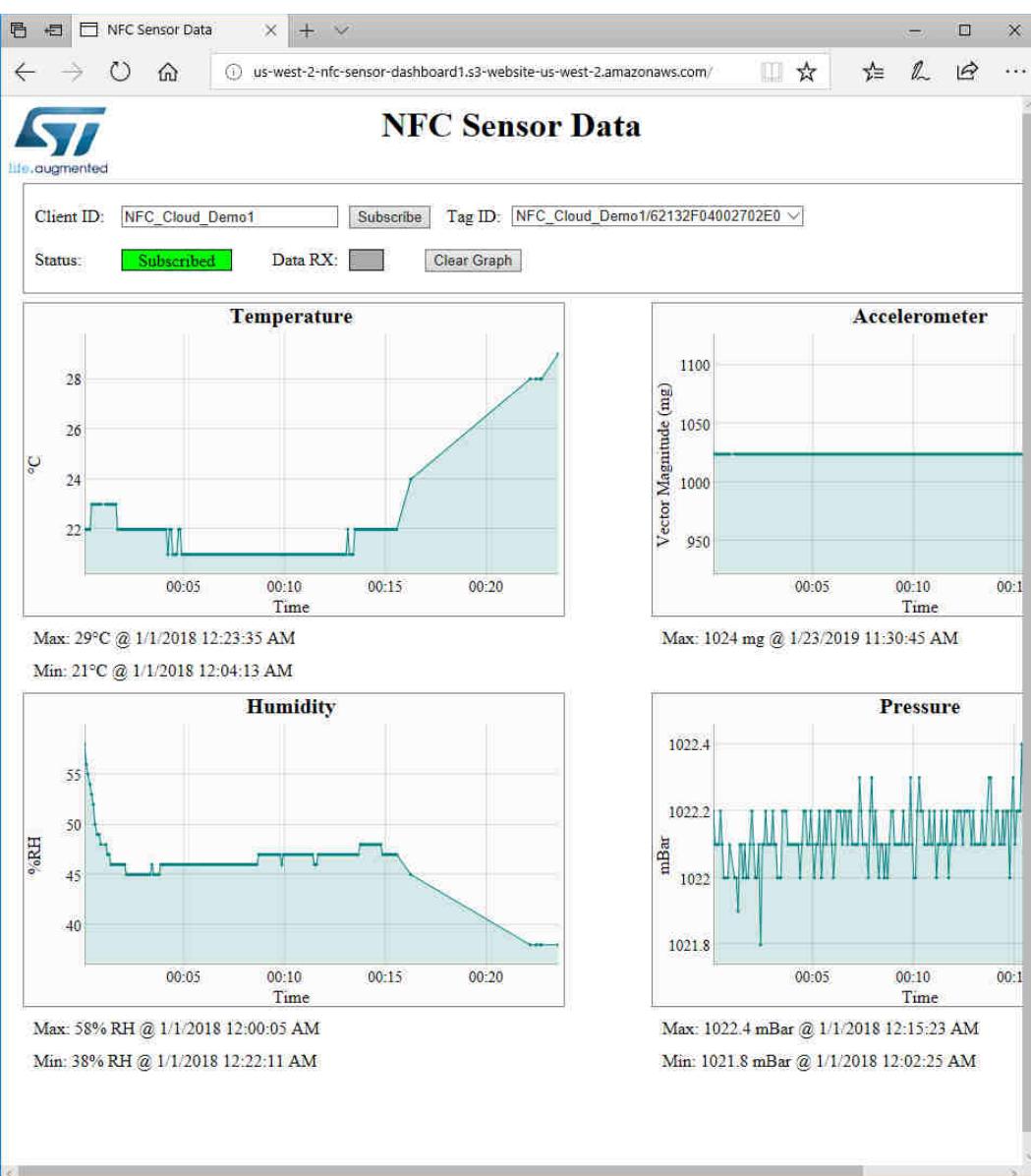
- Select **Chart** Tab



- Tap the NFC Sensor

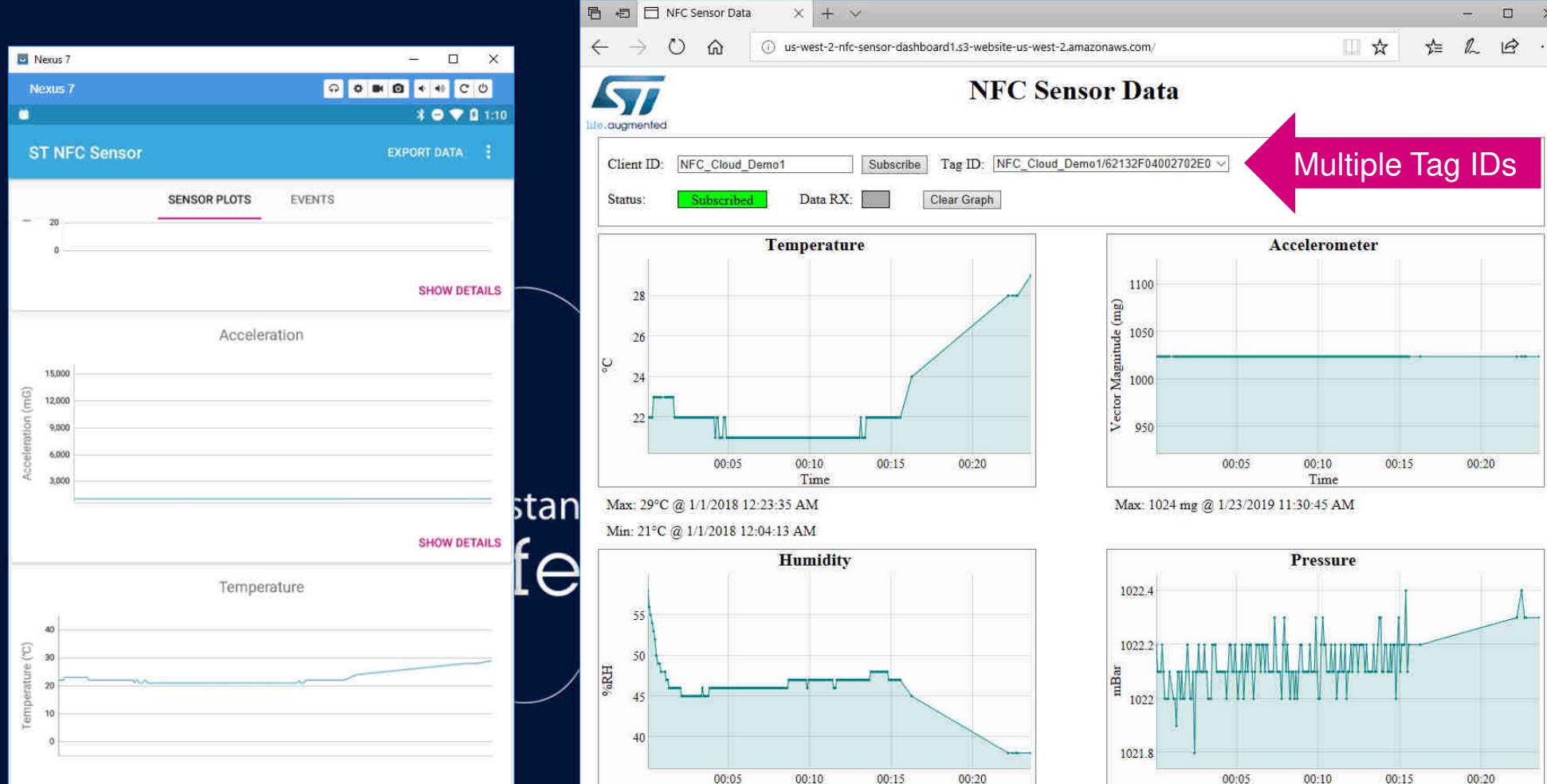


- Data will be automatically sync'd with the Cloud

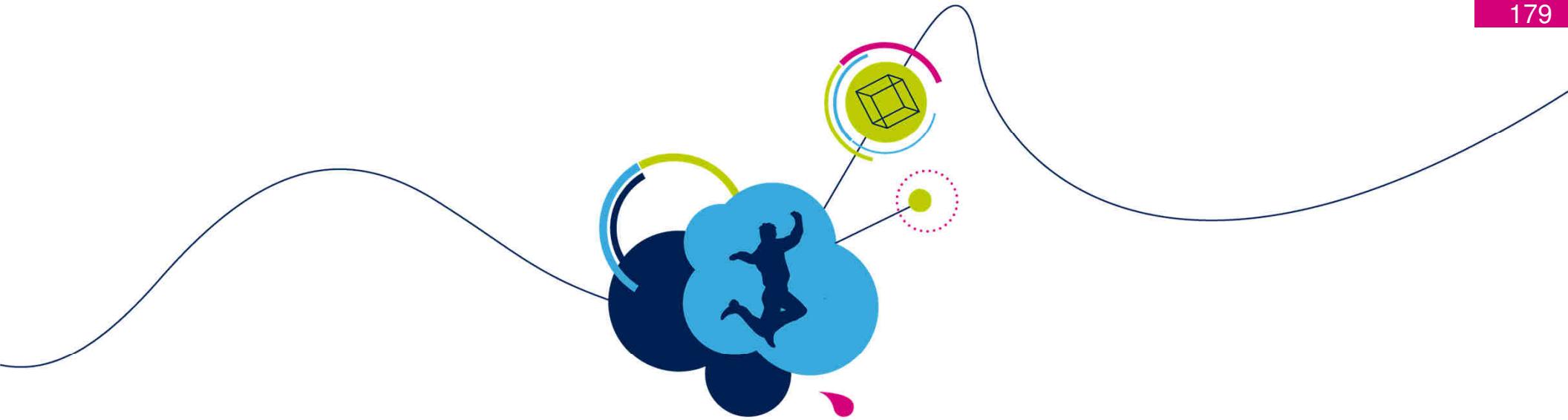


- When a NFC Sensor is now read from a tablet data will be displayed





At every tap from the NFC Sensor data will now be visualized both on the tablet and on the cloud dashboard.  
EXAMPLE: Multiple users in different moments and with different handheld devices can now tap the same NFC Sensor and post on the same Cloud dashboard to rebuild a shipment history



What's happening in the background?

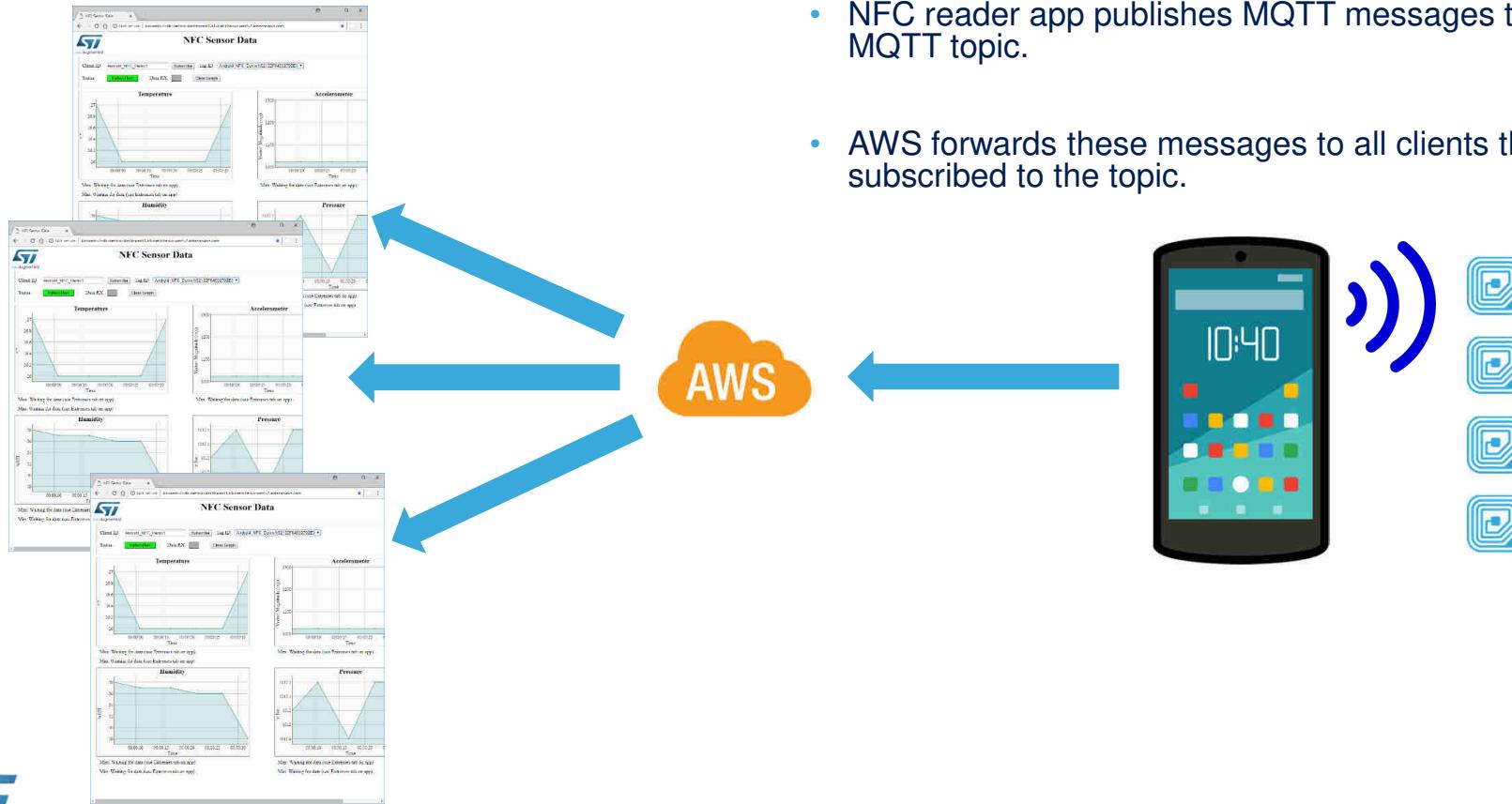
# Web Dashboard

180

- Web dashboard is a browser-hosted JavaScript application based around the open-source Paho MQTT client.
  - MQTT is a simple one-to-many messaging protocol
  - Clients publish and subscribe to one or many MQTT “topics”
- Web dashboard HTML is hosted in an AWS S3 bucket
  - Provides data storage, public web hosting, and DNS
- NFC reader app connects to the AWS cloud and publishes sensor data to an MQTT topic of the form:  
`<Client ID>/<Tag Serial Number>/<Data Type>`  
Where:
  - Client ID is the AWS Thing name used by the NFC reader app
  - Tag Serial Number is the S/N exposed by the NFC tag itself
  - Data Type is “extremes” (for min/max data) or “sensorData” (for charts)
- Web dashboard subscribes to MQTT messages sent from <Client ID> and parses messages based on Tag S/N and data type.

# MQTT Message Publication

- NFC reader app publishes MQTT messages to a given MQTT topic.
- AWS forwards these messages to all clients that are subscribed to the topic.



# AWS Thing Creation



- NFC reader app is an AWS IoT “Thing”
- Thing must be created and registered with AWS prior to use.

AWS Management Console

AWS services

Find services  
You can enter names, keyword or acronyms:

Q iot core

IoT Core  
Connect Devices to the Cloud

▶ Recently visited services

▼ All services

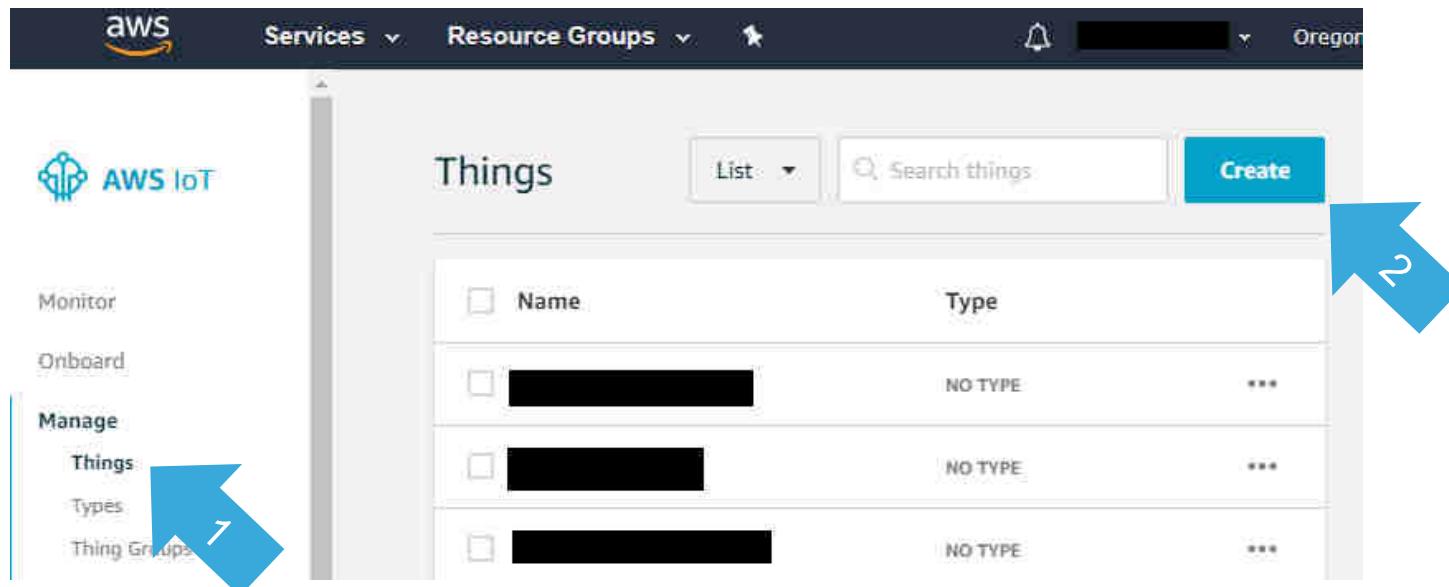
- Compute
  - EC2
  - Lightsail
  - ECR
  - ECS
  - EKS
  - Lambda
  - Batch
  - Elastic Beanstalk
- Developer Tools
  - CodeStar
  - CodeCommit
  - CodeBuild
  - CodeDeploy
  - CodePipeline
  - Cloud9
  - X-Ray
- Machine Learn
  - Amazon SageM
  - Amazon Compr
  - AWS DeepLens
  - Amazon Lex
  - Machine Learni
  - Amazon Polly
  - Rekognition
  - Amazon Transc
  - Amazon Transla
  - Amazon Person
- Robotics

# AWS Thing Creation

- Login to AWS console
- Go to the AWS IoT Core control panel

# AWS Thing Creation

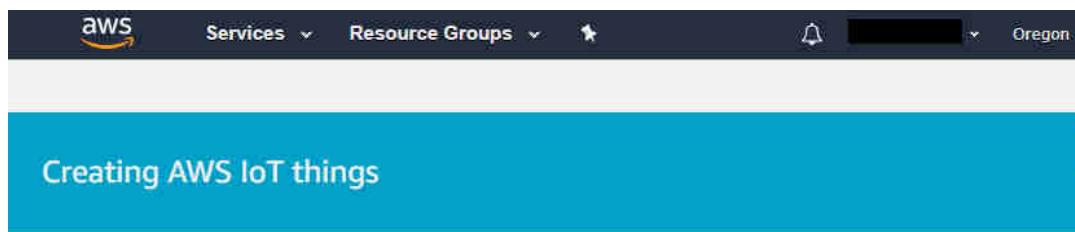
184



- 1) Go to Manage -> Things
- 2) Go to the IoT Core control panel

# AWS Thing Creation

185



1) Click “Create a single thing”

An IoT thing is a representation and record of your physical device in the cloud. Any physical device needs a thing record in order to work with AWS IoT. [Learn more](#).

Register a single AWS IoT thing  
Create a thing in your registry

Create a single thing

Bulk register many AWS IoT things  
Create things in your registry for a large number of devices already using AWS IoT, or register devices so they are ready to connect to AWS IoT.

Create many things

Cancel

Create a single thing

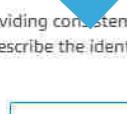
# AWS Thing Creation

**CREATE A THING**  
Add your device to the thing registry

STEP  
1/3

This step creates an entry in the thing registry and a thing shadow for your device.

Name  

Apply a type to this thing 

Using a thing type simplifies device management by providing consistent registry data for things that share a type. Types provide things with a common set of attributes, which describe the identity and capabilities of your device, and a description.

Thing Type

Add this thing to a group 

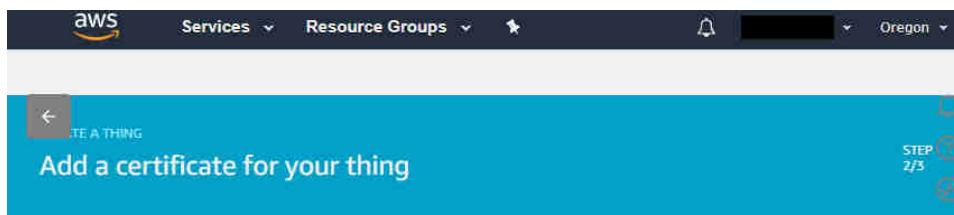
Adding your thing to a group allows you to manage devices remotely using jobs.

Thing Group

Set searchable thing attribute (optional)

# AWS Thing Creation

187



- 1) Create a device certificate using the One-click certificate creation method.

A screenshot of the AWS IoT 'Create a Thing' wizard. The title bar shows 'AWS Services Resource Groups Oregon'. The main heading is 'CREATE A THING' with a sub-instruction 'Add a certificate for your thing'. Below it, a note says 'A certificate is used to authenticate your device's connection to AWS IoT.' There are four options: 'One-click certificate creation (recommended)' (selected), 'Create with CSR', 'Use my certificate', and 'Skip certificate and create thing'. Each option has a corresponding 'Get started' or 'Create' button. A blue arrow points to the 'Create certificate' button.

# AWS Thing Creation

The screenshot shows the AWS IoT Thing Creation interface. At the top, there's a green banner saying "Certificate created!". Below it, instructions say to download files for connecting a device. A table lists three files: "A certificate for this thing" (dd2c319f30.cert.pem), "A public key" (dd2c319f30.public.key), and "A private key" (dd2c319f30.private.key), each with a "Download" button. A large blue arrow labeled "1" points to the "Download" button for the private key. Another blue arrow labeled "2" points to the "Activate" button. A third blue arrow labeled "3" points to the "Attach a policy" button.

A certificate for this thing	dd2c319f30.cert.pem	<a href="#">Download</a>
A public key	dd2c319f30.public.key	<a href="#">Download</a>
A private key	dd2c319f30.private.key	<a href="#">Download</a>

You also need to download a root CA for AWS IoT:  
A root CA for AWS IoT [Download](#)

[Activate](#)

[Cancel](#)

[Done](#) [Attach a policy](#)

- 1) The device certificate and key pair have been created. Download these now and store them in a safe location. **This is your only chance to download the private key.**
- 2) Click Activate
- 3) Click Done (we'll attach a policy later)

# AWS Thing Creation

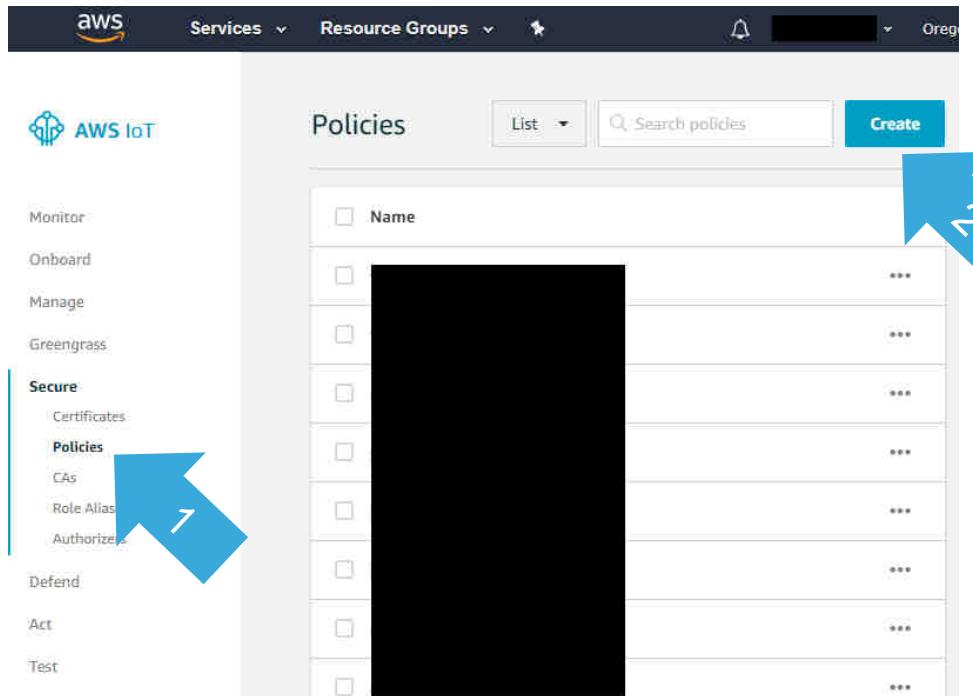
- 1) The new AWS Thing should now appear in the list of registered Things.

The screenshot shows the AWS IoT Things list. On the left is a sidebar with navigation links: AWS IoT, Monitor, Onboard, Manage (selected), Things, Types, Thing Groups, Billing Groups, Jobs, Greengrass, Secure, Defend, Act, and Test. The main area lists 14 things, each with a checkbox and the name followed by 'NO TYPE'. A blue arrow points to the 15th row, which contains the text 'NFC\_Cloud\_Demo1' in blue, indicating it is the newly created thing. The entire row is highlighted with a light gray background.

<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	[REDACTED]	NO TYPE
<input type="checkbox"/>	NFC_Cloud_Demo1	NO TYPE

# AWS Thing Creation

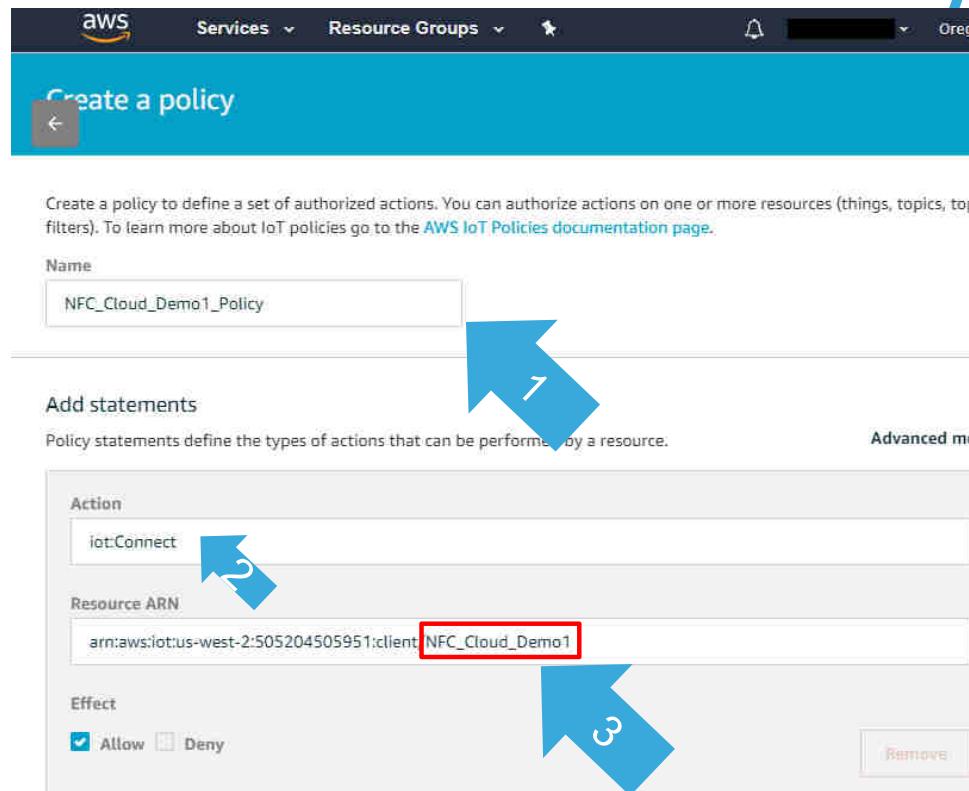
190



- 1) Go to Secure → Policies
- 2) Click on Create

# AWS Thing Creation

191



- 1) Give the new policy a unique name
- 2) Add a new statement as follows:
  - a) Action: "iot:Connect"
  - b) Resource ARN: Change client name at end of string to match the AWS thing name you've created in the previous steps
  - c) Effect: Allow
- 3) Click Add statement

# AWS Thing Creation

The screenshot shows the AWS IoT Thing Creation interface. It displays two policy statements:

- Statement 1:** Action: iot:Connect, Resource ARN: arn:aws:iot:us-west-2:505204505951:client/NFC\_Cloud\_Demo1, Effect: Allow.
- Statement 2:** Action: iot:Publish, Resource ARN: arn:aws:iot:us-west-2:505204505951:topic/\*, Effect: Allow.

A large blue arrow labeled '1' points to the second statement.

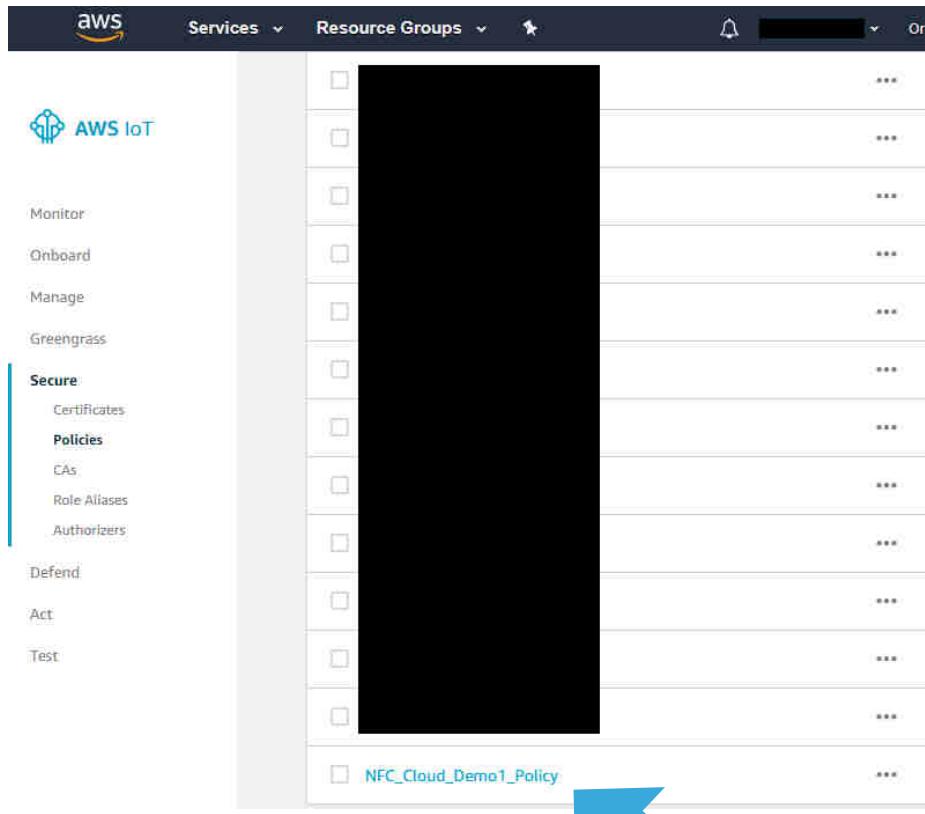
- 1) Add a new statement as follows:
  - a) Action: "iot:Publish"
  - b) Resource ARN: Change topic name at end of string to "\*" to represent all topics
  - c) Effect: Allow

These two statements combined allow the NFC\_Cloud\_Demo1 thing to connect to AWS IoT Core services and publish MQTT messages to any topic.

- 2) Click "Create"

# AWS Thing Creation

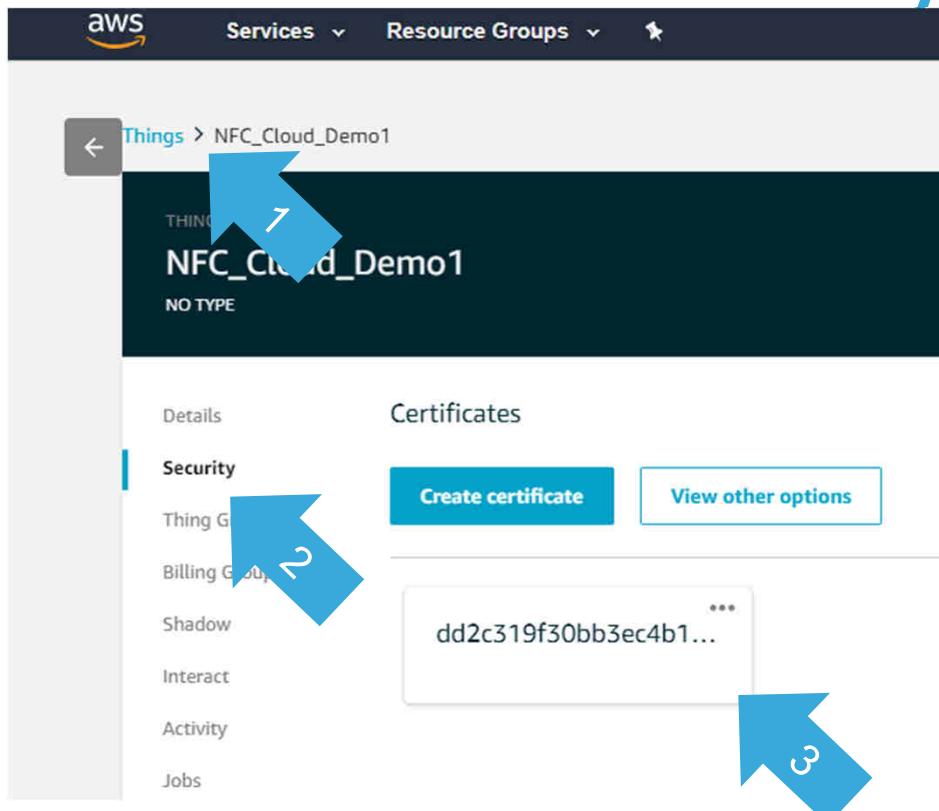
193



- 1) The policy list should now include the newly-created policy.

# AWS Thing Creation

194



- 1) Go back to Manage → Things and click on your newly-created thing.
- 2) Go to the Security tab
- 3) Click on the certificate

# AWS Thing Creation

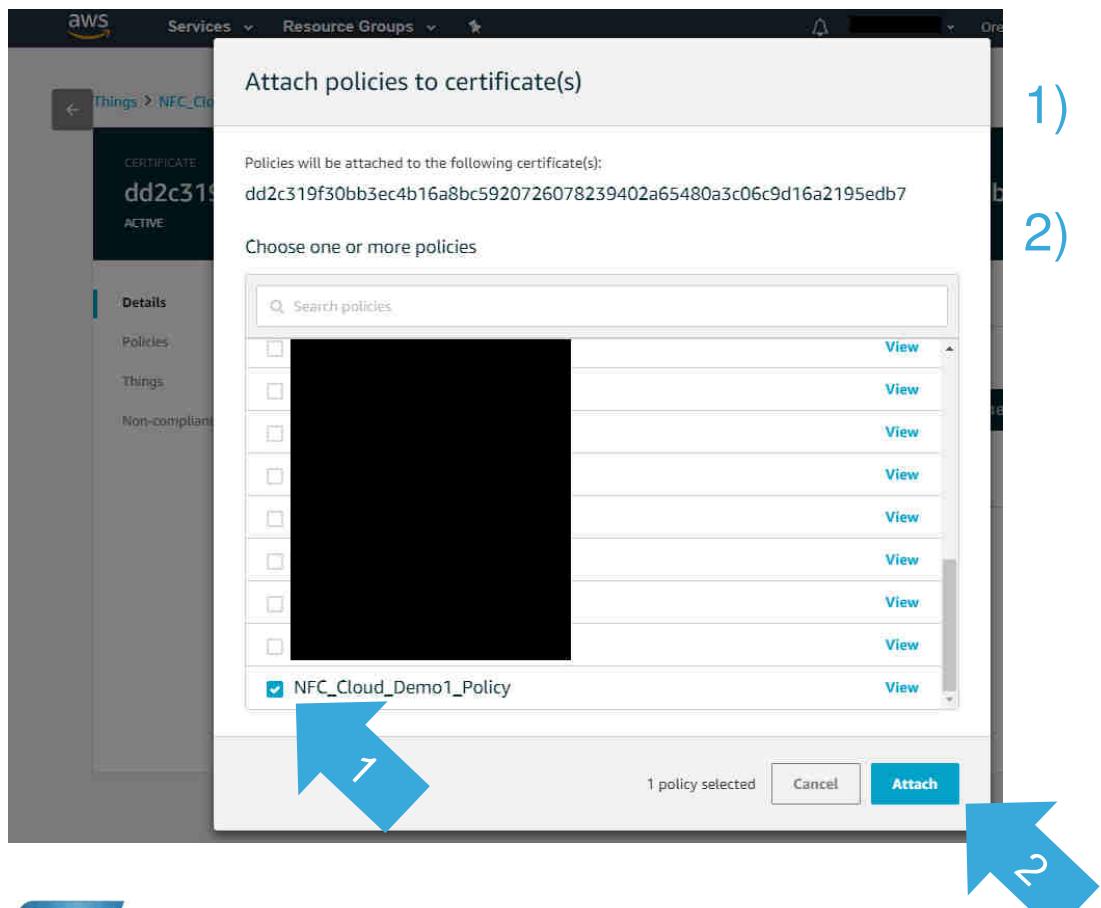
195

The screenshot shows the AWS IoT Certificate creation interface. At the top, there's a navigation bar with the AWS logo, Services dropdown, Resource Groups dropdown, a bell icon, Oregon location, and Support link. Below the navigation, the path is shown as Things > NFC\_Cloud\_Demo1 > dd2c319f30bb3ec4b16a... . The main area displays a certificate with the ID dd2c319f30bb3ec4b16a8bc5920726078239402a65480a3c06c9d16a2195edb7 and status ACTIVE. On the right, an 'Actions' dropdown menu is open, listing options like Activate, Deactivate, Revoke, Accept transfer, Reject transfer, Revoke transfer, Start transfer, Attach policy, Attach thing, Download, and Delete. A blue arrow points from the text '1) Click on Actions → Attach policy' to the 'Attach policy' option in the dropdown. The 'Details' section below the certificate shows Issuer information (OU=Amazon Web Services O=Amazon.com Inc. L=Seattle ST=Washington C=US), Subject (CN=AWS IoT Certificate), Create date (Jan 21, 2019 10:25:50 AM -0800), Effective date (Jan 21, 2019 10:23:50 AM -0800), and Expiration date (Dec 31, 2049 3:59:59 PM -0800).

- 1) Click on Actions → Attach policy

# AWS Thing Creation

196



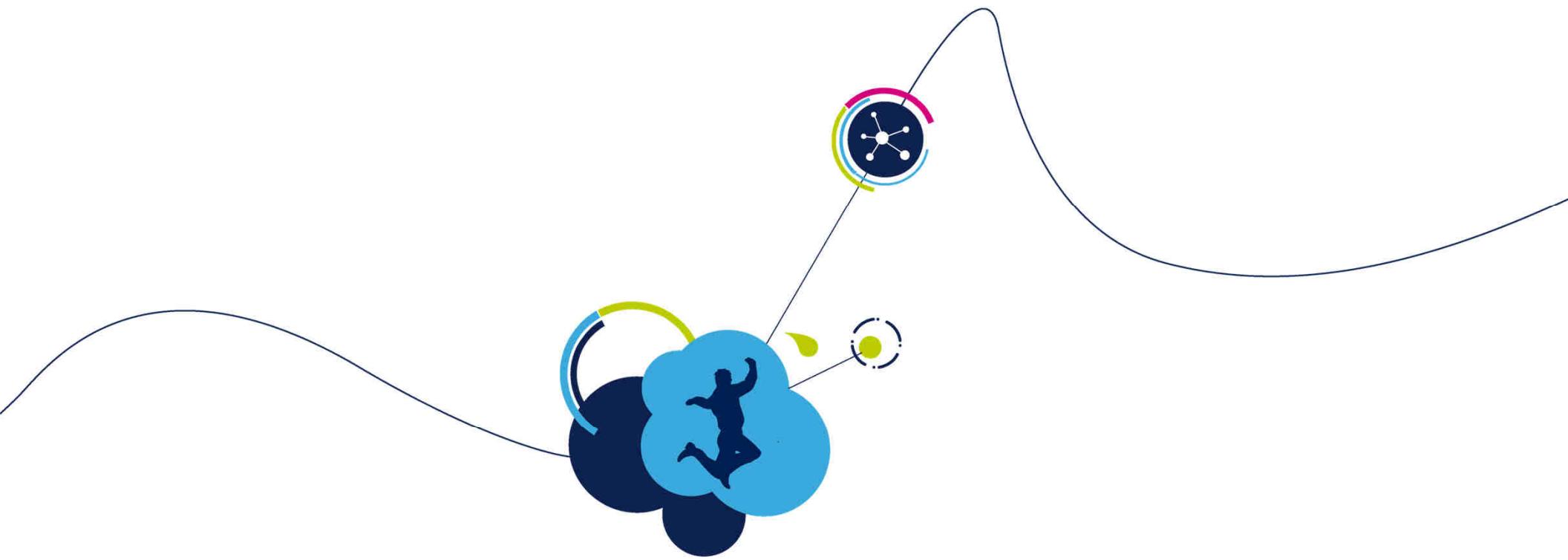
- 1) Select the newly-created policy
- 2) Click Attach

# AWS Thing Creation

197

The screenshot shows the AWS IoT Things console interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, 'Resource Groups' dropdown, a bell icon, 'Oregon' region selector, and 'Support' link. Below the navigation is a breadcrumb trail: 'things > NFC\_Cloud\_Demo1'. The main area has a dark header with 'THING' and the name 'NFC\_Cloud\_Demo1' in white, followed by 'NO TYPE' and an 'Actions' dropdown menu. The left sidebar lists several tabs: 'Details' (selected), 'Security', 'Thing Groups', 'Billing Groups', 'Shadow', 'Interact' (selected), 'Activity', 'Jobs', and 'Violations'. The 'Interact' tab is highlighted with a blue arrow labeled '1'. In the main content area, under the 'Interact' tab, there's a section for 'HTTPS' and another for 'MQTT'. The 'MQTT' section contains the text: 'Use topics to enable applications and things to get, update, or delete the shadow information for a Thing (Thing Shadow)' and a 'Learn more' link. A red box highlights the MQTT endpoint URL 'a3t9fjx4hfxd1z-ats.iot.us-west-2.amazonaws.com' with a blue arrow labeled '2' pointing to it.

- 1) Go to the Interact tab
- 2) Note the endpoint URL
- 3) Use the endpoint URL along with the device certificate and private key to configure NFC application.



Thank You!