

# Microcontroller programming

1 - Introduction to microcontrollers



# Reading tips



- The keyboard symbol means it's practice time
- Colored text usually carries a meaning :
  - Blue text highlights a new concept or important take-away
  - Red text highlights a tricky part or risk of mistake
  - Green text highlights a skill
- At the end of each slide deck, you'll find a recap of technologies and concepts



# Agenda

- Goals of this course
- The limits of the PC architecture
- Characteristics of MCUs
- Introducing the STM32F429
- Setting up our development environment



# Goals of this course

Why am I here ?



# Goals of this course

- Working on embedded systems requires specific skills
  - Adopting a system-oriented approach
  - Understanding how hardware operates
  - Using specific tools and methods
- The ARM course aims to bring you from total beginner to someone able to implement a simple system
- After the ARM course:
  - Study wireless connectivity with IOT1 + IOTG
  - Study real-time operating systems with FRTOS



# Goals of this course

- **Let's take a look at a couple job offers**
  - Embedded Software Engineer @ Bose
  - Embedded Software Engineer @ Netatmo
- What does each of these job offers ask of us ?
- **One frame of reference** would be :
  - Concepts
  - Software technologies
  - Hardware technologies
  - Skills (know-how)



# Goals of this course

- **Embedded Software Engineer @ Bose** (as of April 2024)
- What you could be working on :



*Open the .pdf on Moodle and try applying the frame of reference*



# Goals of this course

- **Embedded Software Engineer @ Bose** (as of April 2024)
- What you could be working on :



- **Requirements :**
  - *Concepts* : RTOS, Active Noise Reduction, Audio processing, embedded security
  - *SW tech* : C, C++, Linux, Android, Git, Jenkins
  - *HW tech* : Bluetooth, Hexagon, STMicro, AMLogic, Airoha, BES, I2C, I2S, SPI





# Goals of this course

- **Embedded Software Engineer @ Netatmo** (as of April 2024)
- What you could be working on :



*Open the .pdf on Moodle and try applying the frame of reference*



# Goals of this course

- **Embedded Software Engineer @ Netatmo** (as of April 2024)
- What you could be working on :



- Requirements :
  - *Concepts* : Constrained memory, constrained energy, RTOS, event-driven, SoC
  - *SW tech* : C, TCP/IP, Git
  - *HW tech* : ARM Cortex-M, Wi-Fi, Bluetooth, Zigbee, Thread



# Goals of this course

- **What do these job offers have in common ?**
- They ask for complete embedded product design skills :
  - Requirements analysis
  - Hardware selection
  - Software design
  - Software drivers implementation (talking to hardware)
  - Software libraries implementation (talking to software)
  - Software validation
  - Documentation



# Goals of this course

- **ING1** taught you the basics
- Let's see what **ARM course** will teach you :
  - Requirements analysis
  - ⚙ ○ Hardware components selection
  - ⚙ ○ Software design
  - ⚙ ○ Software drivers implementation
  - ✓ ○ Software libraries implementation
  - ⚙ ○ Software validation (unit tests won't be enough !)
  - ✓ ○ Documentation (code)
  - ⚙ ○ Documentation (high-level)



# Goals of this course

The ARM course's **hardware stack** :

Technology	What's that ?	Rationale
ARMv7	Instruction Set Architecture	Very very common What's on the STM32 devkit
ARM Cortex-M4	Processor architecture (ARM subset)	Middle-range MCU (means extensible project !)
AMBA	Bus architecture (for ARM)	Knowing how a bus works is useful for system design
STM32F429	Microcontroller (MCU)	Available in lab
JTAG	Program/debug protocol	Very common
I2C, U(S)ART	Serial wire protocols	Very common, respectively for peripherals and for communication with another system
GPIO	Digital I/O	The most basic I/O ever in embedded systems !
AZ-Delivery logic analyzer	Digital I/O real-time monitoring on HW	Useful to learn and debug the above protocols

You may contrast this with this quite complete "[embedded systems engineering roadmap](#)" found on Github



# Goals of this course

The ARM course's **software stack** :

Technology	What's that ?	Rationale
STM32CubeIDE	Integrated Development Environment	Used by the majority of developers for ARM systems Easy pin configuration Easy study of the Hardware Abstraction Layer
C	Programming language	Every vendor HAL seems written with it
TinyFrame	Very simple protocol library ( <a href="#">link</a> )	Simple Nice asynchronous API Nobody flips bits manually on UART anymore
nanopb	Compact C frontend for Google's Protocol Buffers library ( <a href="#">link</a> )	Well-known protocol-building library Easy implementation on PC side
Saleae Logic 2	Logic analyzer software for PC	Allows using logic analyzer



# The limits of the PC architecture

No AMD in your fridge



# The limits of the PC architecture

- “*Why don't these people use good ol' PCs to run their software ?*”
  - 64-bits
  - At least 1 core clocked @ 2GHz+
  - At least 2GB of RAM
  - At least 32GB of storage
  - USB slots with plug and play
- After all, there are "rugged PC boxes" ready for hostile environment
  - See these pictures of Nodka and ASrock products
  - Smaller than servers, dust-resistant, plenty of ports
- Why couldn't this thermostat run Intel+Linux ?





# The limits of the PC architecture

- Multiple reasons !
  - Economics
  - Power consumption
  - Features
  - Lifetime
- An embedded system is a specialized system
- The vast world of embedded systems owes its variety to this
- As an embedded system engineer, your software stack is not the project's priority
- Hardware comes first !



# The limits of the PC architecture



- Economics
- An embedded system has production costs
- If the product should be sold in the thousands, picking HW that is 20 cents cheaper is a serious choice
- Predominance of discrete 8-bit architectures for simple devices
- ...until cost drop of 32-bit, very integrated MCUs made them more interesting in 2010s
- For many products of everyday life, computing power is not the problem ; the problem is I/O



# The limits of the PC architecture



- Power consumption
- An embedded system needs to be power efficient
- Multicore CPUs are the norm in PCs & servers
  - Designed for speed and concurrent application scheduling
  - Always plugged to a power supply
  - Order of magnitude : 10W
- Single-core CPUs are the norm in embedded systems
  - Designed for one precise application
  - May be on battery, not easily accessible
  - Order of magnitude : 10mW (without sleep modes !)



# The limits of the PC architecture



- Features
- An embedded system interacts with the physical world
- The PC architecture was largely driven by user desktop I/O
  - Keyboard, mouse, screen, sound system, webcam...
- Each “embedded” architecture uniquely combines industrial I/O :
  - **Sensors** : magnetic field, relative humidity, CO2 density, acceleration...
  - **Actuators** : stepper motor, DC motor, pump, LED, anti-zombie railgun...
  - Wireless com : 802.15.4, BLE, NFC, LoRA, SigFox, RFID, LTE-M...
  - Wired com : CAN, Flexray, USART, I2C, SPI, OneWire, MIPI...



# The limits of the PC architecture



- Lifetime
- Embedded systems may function for extended periods of time
- General-purpose systems are complex, but also easily accessed
  - How many USB gadgets actually exist ?
  - Swappable disk, swappable RAM, even swappable CPU !
  - Saying “modularity” makes SW engineers’ eyes shiny 😊
- But complex is our enemy !
  - HR problem: finding skilled engineers
  - R&D problem: fighting challenging problems
  - QA problem : testing every case
  - Maintenance problem: making it work 7 years after people changed jobs



# The limits of the PC architecture

- Quizz : do these products use 64-bit, 32-bit or 8-bit ?

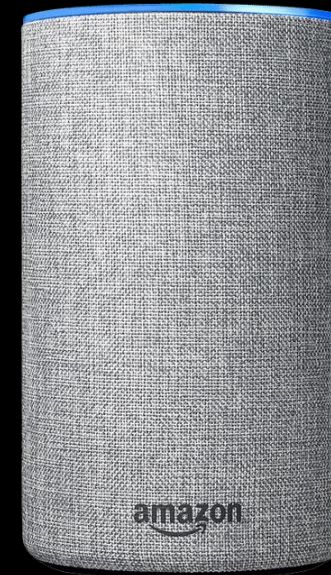
Rowenta SO9276



Palm Zire 31



Amazon Echo





# The limits of the PC architecture

- Let's search for technical specifications online ! (links below)

Rowenta SO9276



**8-bit** ([SN8P2722SG](#))

Palm Zire 31



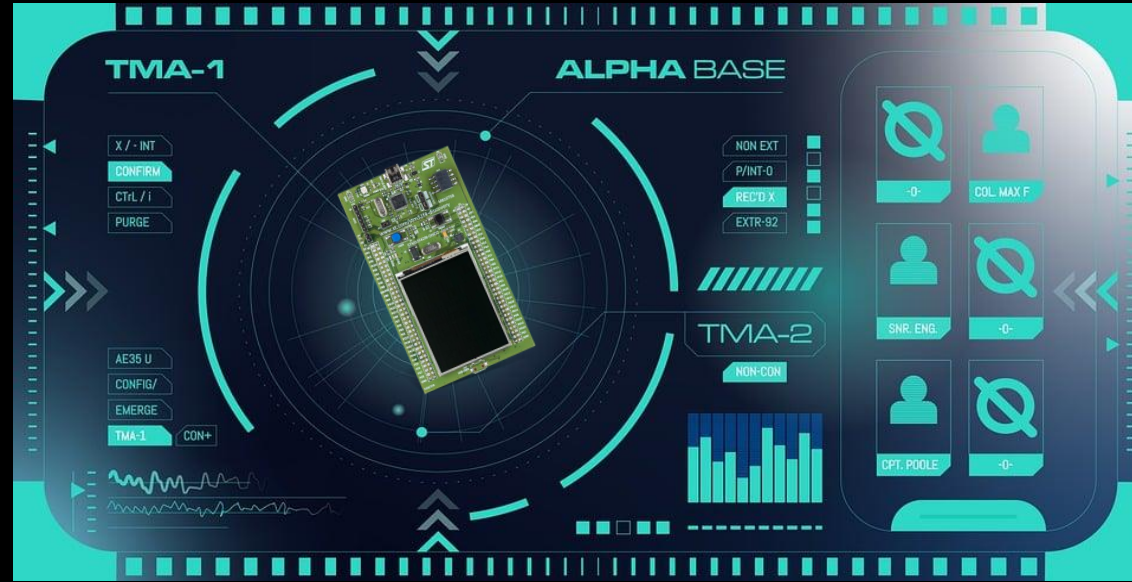
**32-bit** ([PXA255](#))

Amazon Echo



**32-bit** ([DM3725](#))





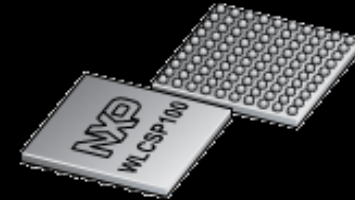
# Characteristics of MCUs

Welcome to the zoo



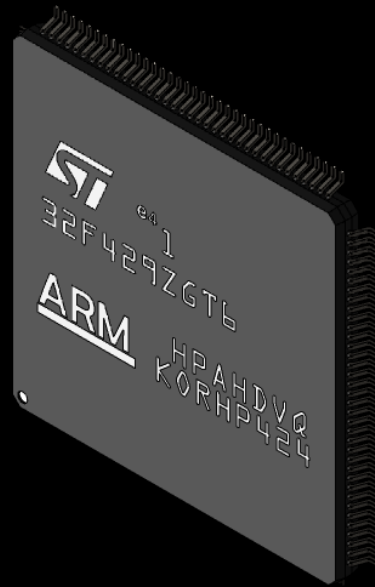
# Characteristics of MCUs

- **MicroController Units** (MCU) are what you will find on pretty much any embedded system
- Let's start by simply looking at one
- *Looks like a chip...*
- But visual inspection already shows some differences with the **MicroProcessors Units** (MPUs) you're used to see :
  - Small : biggest are 1cm large
  - Various types of **mounting** : BGA, PGA, ZIF socket, QFN...
  - Come in weird boards with pins all around (**development kits**)
- Those board are never used as-is in products
- Real products are laid out on dedicated PCBs

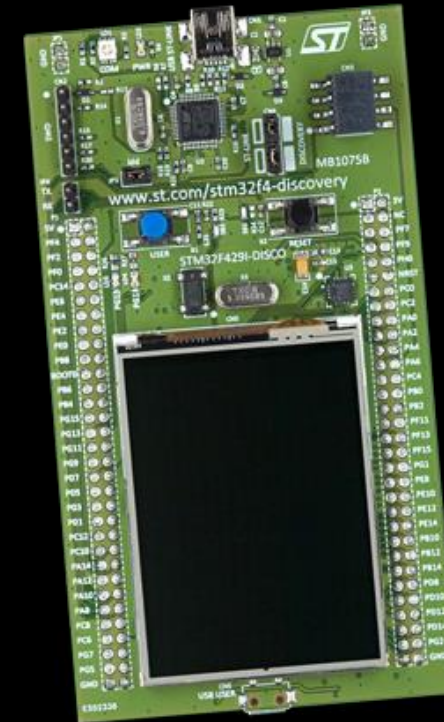


# Characteristics of MCUs

MCU (TQFP mount)

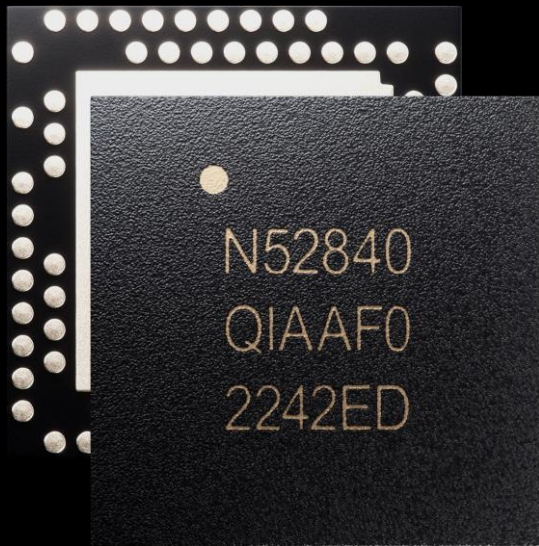


Devkit

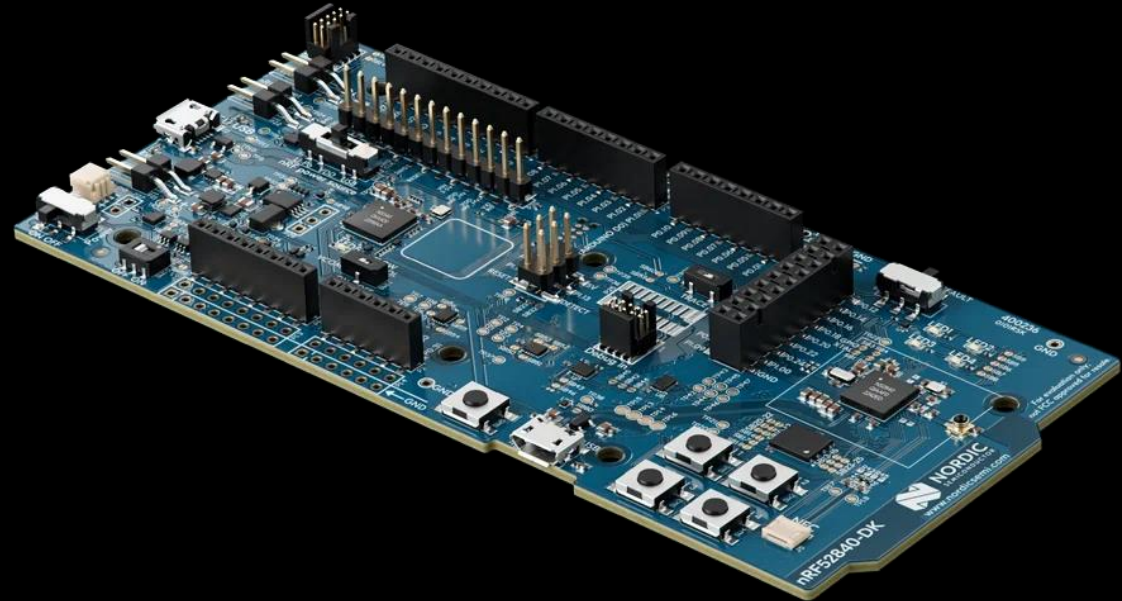


# Characteristics of MCUs

MCU (BGA mount)



Devkit



# Characteristics of MCUs

MCU (QFN mount)



Devkit

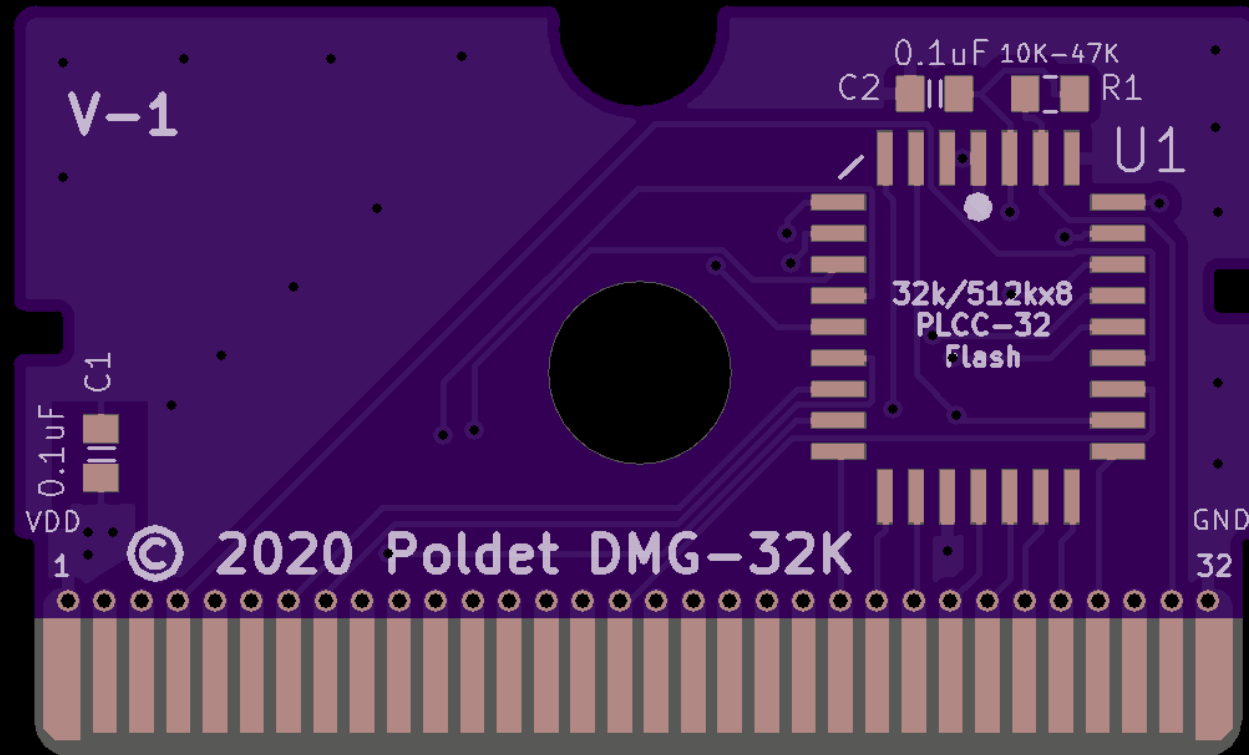


# Characteristics of MCUs

- The untrained eye only sees black plastic squares
- Check for **parts numbers** written on the chips
  - Needed to find the **datasheets** useful for programming them
  - Some chips are super small : use a **lab lens**, or ask your EE
  - Each manufacturer uses its own labelling logic
- Some components are easy to identify and can tell you what the board was made for
- Another piece of information on the board is the **silk** : the (usually white) text around parts
- Let's practice finding datasheets from parts numbers !



# Characteristics of MCUs



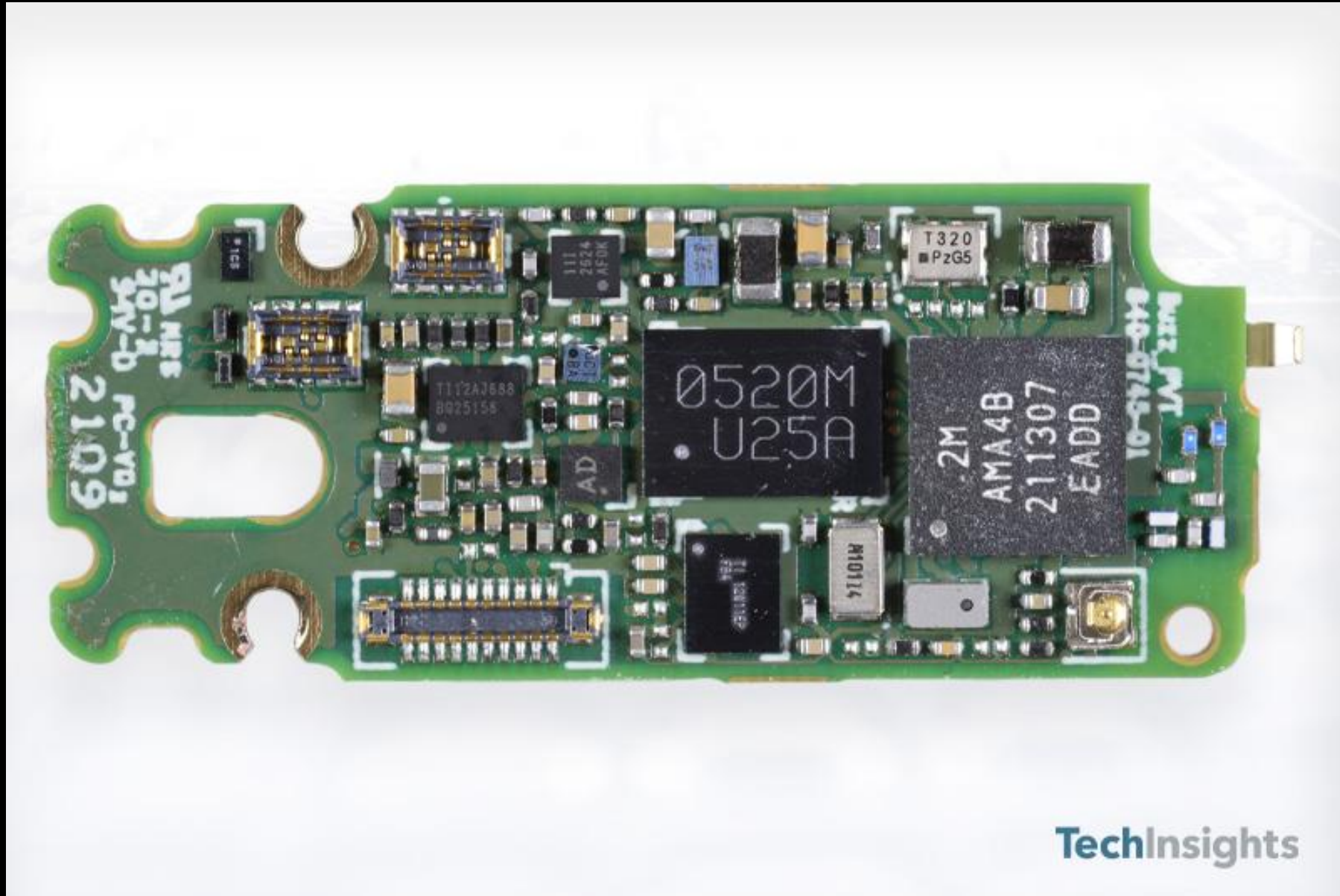
*Silk text (here in white) is inputted during the PCB design phase.*

*It is often used to identify signals, and give general information like design revision and author.*





# Characteristics of MCUs

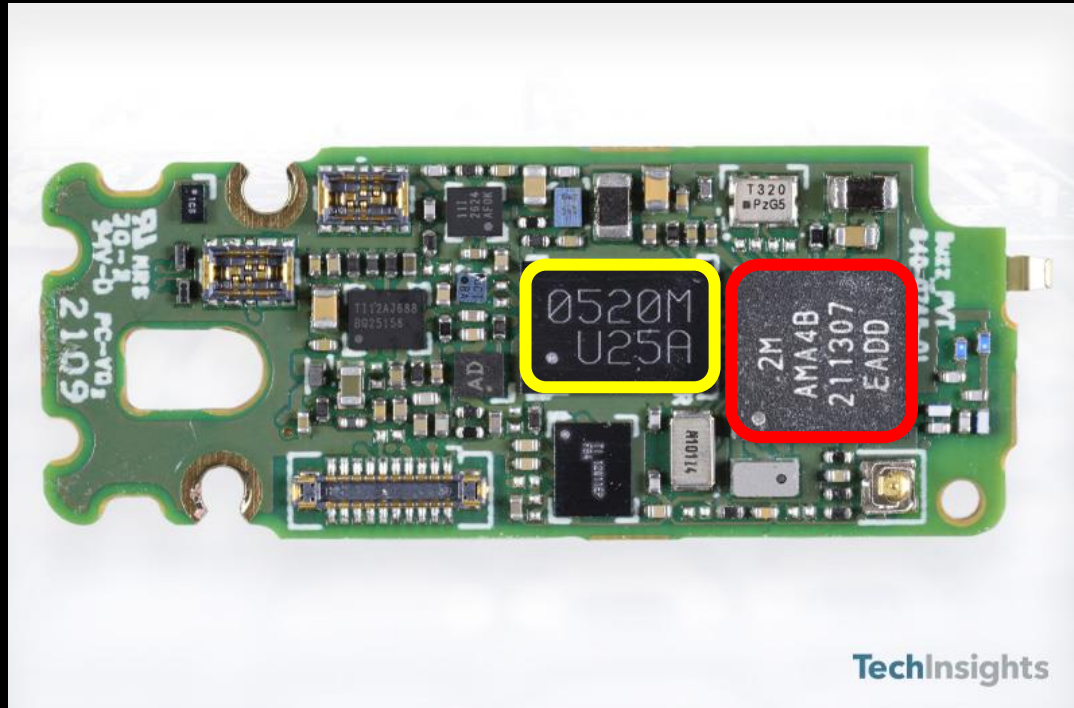


# Characteristics of MCUs



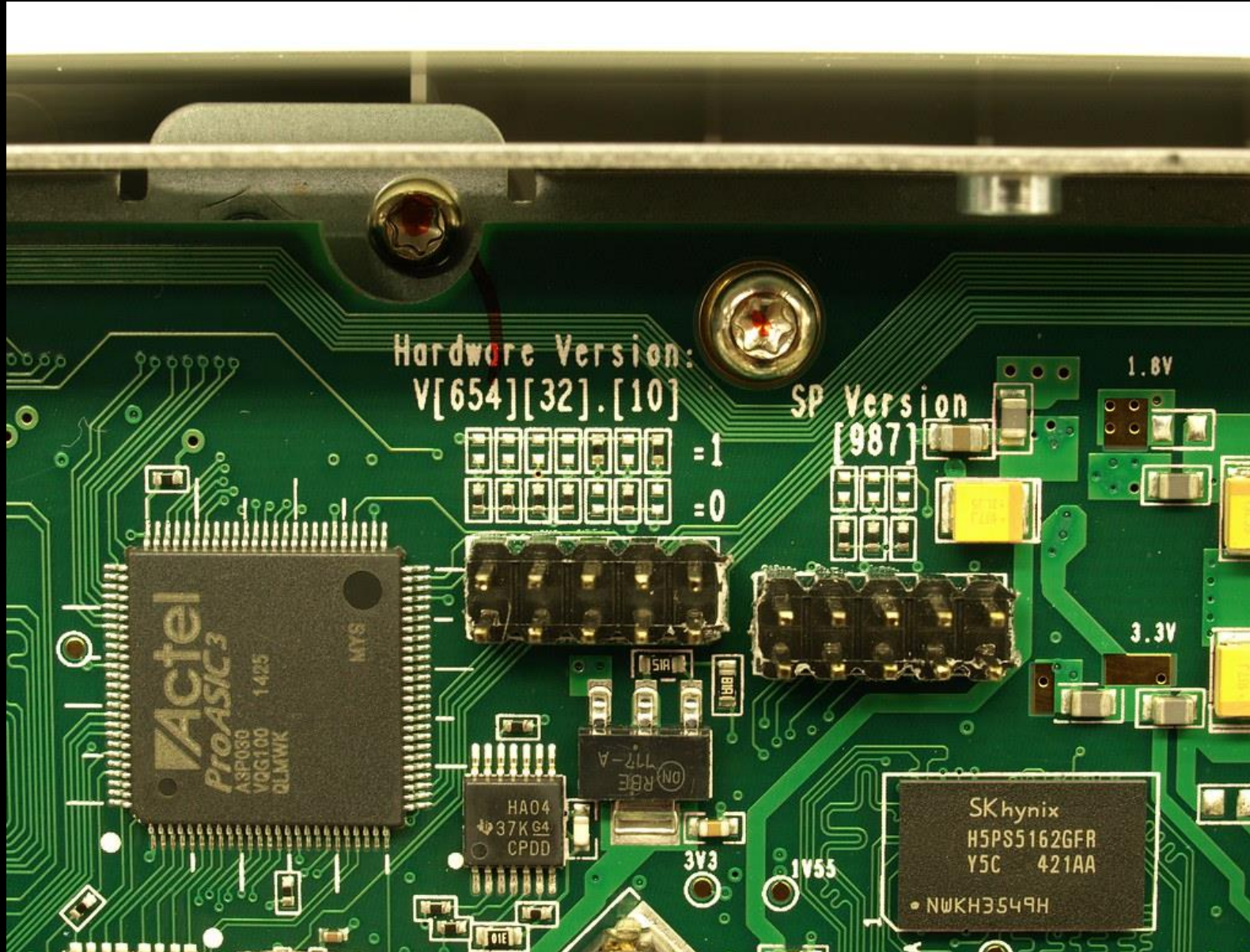
## Fitbit Luxe smartband

- 0520MU25A – ? – No luck !
- Ambiq Appolo4 Blue – MCU – [Link](#)
- Why are the other components so hard to find ?
  - Passive components tend to get discontinued quickly
  - The VAST majority come from China/Taiwan/Korea/... which your Google search engine might not know that well :-)
  - When it's hard, try using a search engine like [octopart.com](#)...
  - ...but here it won't help you !
- I was able to find the Appolo4 by typing "fitbit luxe teardown" ; sometimes "teardown" or "reverse engineering" might help
  - This chip is designed, manufactured and sold as an Ambiq <=> Fitbit private business



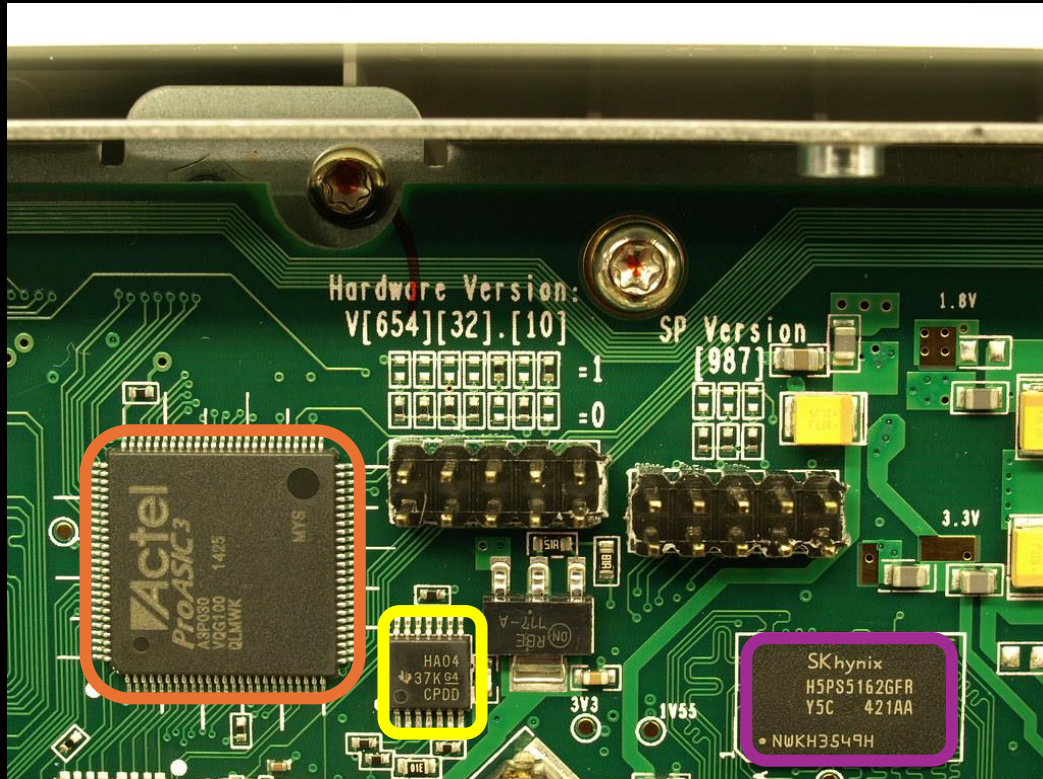


# Characteristics of MCUs





# Characteristics of MCUs



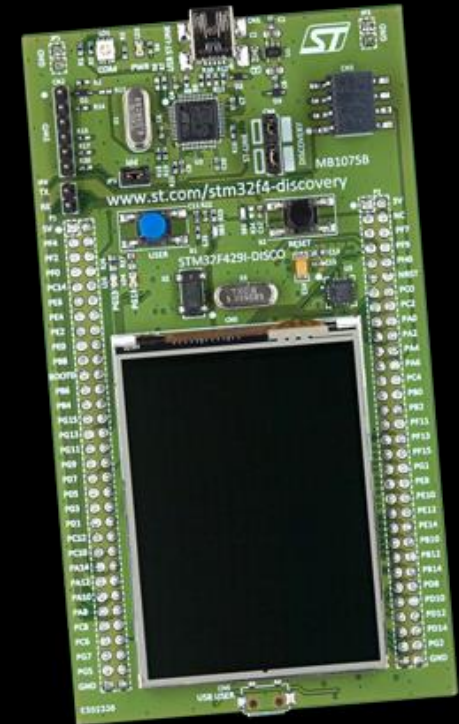
Rigol DS1054Z oscilloscope

- **Actel ProASIC3 A3P030 VQG100** – FPGA – [Link](#)
- **TI HA04-37K-CPDD** – TI – Mistral “Le Chat” says it could be close to [SN74AHC04](#). Packaging matches but this is not a proof at all !
- **SK Hynix H5PS5162GFR-Y5C** – DRAM – [Link](#)
  - The search bar on SK Hynix's own website does not find it !
- What we're doing is rebuilding the **Bill Of Materials...**
  - [Read here for an introduction to BoMs](#)



# Characteristics of MCUs

- Do the same for the STM32F429
- Your 1<sup>st</sup> job before programming this board !
- Search on [st.com](http://st.com)



# Characteristics of MCUs

- Now let's see more software oriented aspects of MCUs
- First, **using an OS is not always possible**
- 1st reason : hardware support
  - An OS has a generic part, but also a HW dependent part
  - In Linux, that would be whatever file is located in [arch/](#)
  - If your OS has a filesystem but does not know where/how to write files, how can it work ?
- 2nd reason : size
  - Linux is hardly smaller than 2MB...
  - ...my STM32F429I has 2MB exactly. Where do I put my programs ?



# Characteristics of MCUs

- The majority of MCUs are programmed using the C language
- C is the most portable language, even today
  - Open-source compilers (gcc, clang) have good ARM support
  - Proprietary compilers made by vendors (IAR, Microchip...) all target the C language
- Rust is starting to get notoriety in the embedded space too. But this world moves slower than the purely software industry
- Other, exotic languages may be available. Some are platform-specific



# Characteristics of MCUs

- As a software developer, you'll have to get new habits
- Memory handling is more delicate
  - You used to think RAM, or HDD/SSD
  - Now you'll have to think Flash, EEPROM, PROM, RAM, fuses...
- Knowledge of hardware is nearly mandatory
  - Each vendor (like ST, Espressif...) does provide a set of software libraries, linker scripts, code generators...named the **Hardware Abstraction Layer** (HAL) that eases the developers' lives
  - But as soon as you need to write a driver, you have to know the capabilities of your hardware



# Characteristics of MCUs

- But the biggest difference with "classic" multiprocessors is I/O
- MCUs are I/O driven beasts, not compute driven
- Often this means handling several devices in parallel
- So expect to :
  - Design multi-entry, event-based applications
  - Eat hardware interrupts for breakfast
  - Forget threads : use timers to launch tasks
  - Handle various sleep modes to save power
- Now that we've gathered some documentation, let's dive in





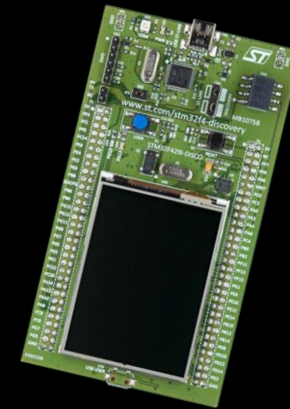
# Introducing the STM32F429

Your new friend !



# Introducing the STM32F429

- Assuming you did the parts study before, you should have found a few PDF files
- Let's do a small presentation still :
  - 1 Cortex-M4 core clocked @ 180MHz
  - Up to 2MB of Flash memory
  - Up to 256KB of SRAM
  - 4 power modes : Run, Sleep, Stop and Standby
  - Up to 168 I/O ports
  - Up to 21 communication interfaces



# Introducing the STM32F429

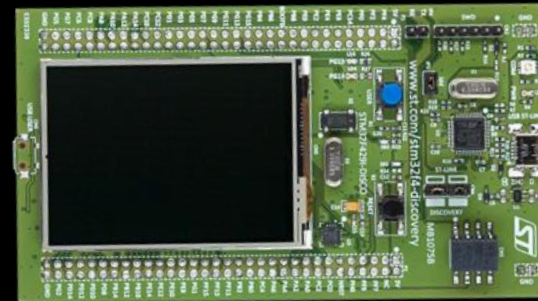
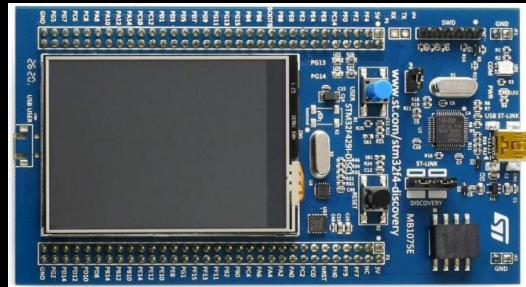
- *What do you mean "up to 2MB" ? Is it 2, or is it not ?!*
- MCUs, just like the processor in your laptop, come in variants
- Identifying what your board contains means **identifying the different levels of documentation**
  - There is no single document to explain how your devkit works
  - We will see the datasheet, the user manual, and the reference manual



# Introducing the STM32F429




- For example, my devkit looks like the blue one. But there is a green one. Both are STM32F429. What the heck ?!



- Let's learn about **MCU part numbering** and **chip revisions**
- Please search the STM32F429xx datasheet online

# Introducing the STM32F429

- My copy of the datasheet (official name DS9405) is revision 10
  - Latest as of 19th of January 2018 (check version history at the end)
  - If you got an older one, search again using ST website, and not Google !

 life.augmented

STM32F437xx STM32F439xx

32b Arm® Cortex®-M4 MCU+FPU, 225DMIPS, up to 2MB Flash/256+4KB RAM, crypto, USB OTG HS/FS, Ethernet, 17 TIMs, 3 ADCs, 20 com. interfaces, camera&LCD-TFT

Datasheet - production data




### Features

Includes ST state-of-the-art patented technology.

Core: Arm® 32-bit Cortex®-M4 CPU with FPU, Adaptive real-time accelerator (ART Accelerator™) allowing 0-wait state execution from Flash memory, frequency up to 180 MHz, MPU, 225 DMIPS/1.25 DMIPS/MHz (Dhrystone 2.1), and DSP instructions

**Memories**

- 512 bytes of OTP memory
- Up to 2 MB of Flash memory organized into two banks allowing read-while-write
- Up to 256+4 KB of SRAM including 64-KB of CCM (core coupled memory) data RAM



LQFP100 (14 × 14 mm)	UFBGA176 (10 × 10 mm)	WLCSP143
LQFP144 (20 × 20 mm)	UFBGA169 (7 × 7 mm)	
LQFP176 (24 × 24 mm)	TFBGA216 (13 × 13 mm)	
LQFP208 (28 × 28 mm)		

bit timers up to 180 MHz, each with up to 4 IC/OC/PWM or pulse counter and quadrature (incremental) encoder input

**Debug mode**

- SWD & JTAG interfaces
- Cortex-M4 Trace Macrocell™



# Introducing the STM32F429

- A datasheet is a document summing up technical characteristics and features of a hardware component
- It is **not written for software developers**, but for EE
- Does it mean it's no use ? No !
  - For MCUs, datasheets are the only place where to find part-specific (and I mean actual, physical chip) information
  - They also sum up features in a neat way
  - For sensors, motors...they explain how to program them
- In DS9405, take a look at § 8 "Part numbering"



# Introducing the STM32F429

- Deciphering the blue board's MCU first...
- It's an STM32F429ZIT6U

*Look up the datasheet and see what this name means.*



# Introducing the STM32F429

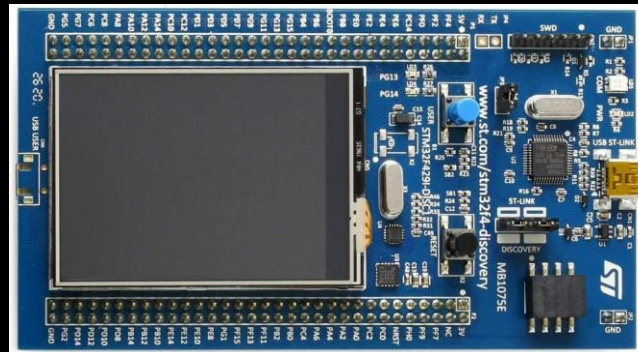


- Deciphering the blue board's MCU first...
- It's an STM32F429ZIT6U
  - *STM32* : ST's 32-bit ARM MCU portfolio
  - *F4* : High-performance MCU family, apparently
  - *429* : We got an LCD-TFT interface (devkit has a screen using that)
  - *Z* : 144 pins (on the MCU, not on the devkit ! The latter MAY be wired to the former)
  - *I* : 2MB of flash
  - *T* : LQFP packaging
  - *6* : package can operate between -40°C and 85°C
  - *U* : ?



# Introducing the STM32F429

- Now that we've identified the main part on the devkit, let's learn about the devkit itself
- We'll go back to the MCU datasheet later
- The datasheet tells us nothing of the header pins, buttons, LEDs, USB port, screen...
- We got to search for the [board user manual](#) now





# Introducing the STM32F429



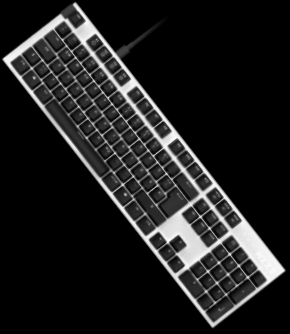
## Essential documentation you should obtain

- STM32F429 devkit
  - User manual – UM1670 : When you just got the board for the first time. Product naming, supported SW toolchains, required drivers, how to power and boot, and to what part of the MCU each peripheral/pin is linked to.
- STM32F429 MCU
  - Datasheet – DS9405 rev10 : When you are searching for the right board for your project, AND when you need very low-level / electrical information.
  - Reference manual – RM0090 rev20 : When you want to program the board.

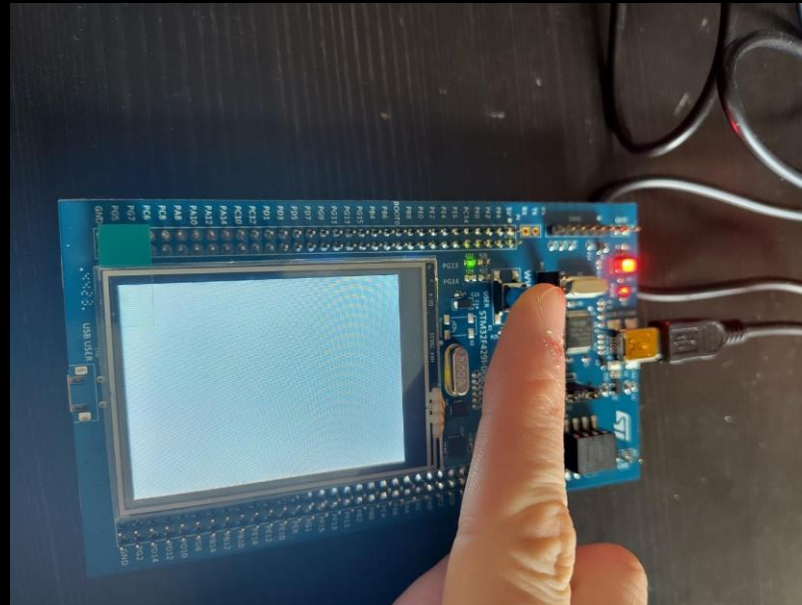
There's more doc', but we will expand our docbase later in the course.



# Setting up our development environment



- Please follow [gistre26-arm-setup.pdf](#) in Moodle
- Test your installation using the suggested example project
- Good to go when green LED blinks when blue button is pressed !



# Quick recap



## Technologies :

- ARMv7
- Cortex-M4
- STM32 family
- STM32 CubeIDE
- STM32F429
- STM32F429I-DISC1 development kit

## Concepts :

- Actuator
- Bill Of Materials
- Board reference manual
- Board user manual
- Chip revision
- Datasheet
- Development kit
- Hardware Abstraction Layer (HAL)



# Quick recap



## Concepts :

- Microcontroller (MCU)
- Microprocessor (MPU)
- Mounting [a chip on a PCB]
- Part number
- Revision [for a board design]
- Sensor
- Silk

