

Your projects

Back to organization

- Courses
- Labs
- Projects

1st deadline:

- Sunday 1st November, 2020
- Project proposal: state-of-the-art, market study, norms and regulations, implementation planning
- Presentations

The image shows a four-month calendar grid from September 2020 to December 2020. Each month is represented by a 6x7 grid of days. Colored dots (blue and green) are placed on specific dates to mark project milestones. In September, there are no dots. In October, a red dot is on the 21st. In November, blue dots are on the 5th, 11th, 18th, and 25th, and a green dot is on the 12th. In December, blue dots are on the 2nd, 9th, 16th, and 23rd, and green dots are on the 3rd, 10th, 17th, and 24th. A black arrow points from the text "Sunday 1st November, 2020" to the 1st November cell in the November row. A green diagonal line starts from the bottom-left and ends at the 25th November cell.

SEPTEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
	1	2	3	4	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				

OCTOBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
					1	2
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	

NOVEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
					1	
2	3	4	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						

DECEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
					1	2
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			

Mondaystartcalendar.com

Back to organization (Cont'd)

- Courses
- Labs
- Projects

2nd deadline:

- Sunday 15th November, 2020
- Technical details: components list, software design (uml diagrams), hardware design, etc.
- Presentations and validation of the project ... No major changes after that

SEPTEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
	1	2	3	4	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				

OCTOBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
					1	2
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	

NOVEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	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						

DECEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
			1	2	3	4
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			

Mondaystartcalendar.com

Back to organization (Cont'd)

- Courses
- Labs
- Projects

3rd deadline:

- Sunday 20th December, 2020
- What to submit: the final report, video, slides, and the code. Everything via your GitHub repository;
- Final Presentations (Jan 7th, 2021)

SEPTEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
	1	2	3	4	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				

OCTOBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
					1	2
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	

NOVEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
						1
2	3	4	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						

DECEMBER 2020						
MON	TUE	WED	THU	FRI	SAT	SUN
			1	2	3	4
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			

Mondaystartcalendar.com

Project-oriented course

At the end of the whole IoT-course, you have to submit:

1. a report;
2. a video;
3. mid-term presentations;
4. final presentation;

Report

Goals (a report? what to include?)

Your deliverables must include:

1. a clear description of the problem you tackle;
2. a comprehensive state-of-the-art around proposed solutions;
3. a description of your IoT-based solution and what makes it better than other solutions;
4. norms and regulations that apply in your specific case;
5. implementation planning in the form of Gantt diagram;
6. anything that would be of interest to your solution.

Video

Goals (a video? what to do? what not to do?)

Do's

- 3 minutes NO MORE, NO LESS;
- Concise description of the Hardware and software;
- Demo of your product;
- Take a look at inspirational works around YouTube, Vimeo, etc.

don'ts

- More than 3 minutes;
- Less than 3 minutes;
- Showing only a demo;

An exemple (<https://www.youtube.com/watch?v=U7y1wiYqHDc>)

Project proposal presentation

Presentation rules:

- Each group has 7 minutes to convince (followed by 3 min. questions);
- Each member of a group has to present, mainly, his part;

IoT website

Objectifs (le site web? pourquoi faire?)

Health care/well being

Baby Phone

Tristan Le bras, Alexandre Dequeker, Alexandre Fieux, Anthony Morali

L'objectif de Bébé Tranquille est de fournir un dispositif technique de couplage asservi entre les parents et leur enfant nouveau-né, à titre expérimental seulement, afin de faciliter la vie des parents.

[PDF](#) [Slides](#) [Video](#) [Code](#)

DigiScript

Nicolas Dzjurga, Guillaume Jobin, Thomas Arpin, Thomas Cornier, Steve Demuemeester

Le digiscript est un traducteur de texte, images, vidéos, pour personnes malvoyantes en braille.

[PDF](#) [Slides](#) [Video](#) [Code](#)



SMYN4000

Chaussure connectée

[PDF](#) [Slides](#) [Video](#) [Code](#)

Smart city

Eye Truck

Vincent Keller, Antoine Vo, Zhu Yongyi

A number of accidents happen on roads every year because the drivers don't notice the cars or pedestrians in the blind zone. Our product is a light stuck on the buses or trucks which reminds the driver when pedestrians or cars are in his blind zone.

[PDF](#) [Slides](#) [Video](#) [Code](#)



Secret Garden

François Bekerman, Laurie Cazals, Jean-Alexis Gagnière, Xavier Nomicosis, Sébastien Serre

Des jardins autonomes qui deviennent réellement interactifs et peuvent devenir une véritable source d'inspiration pour les citoyens.

[PDF](#) [Slides](#) [Video](#) [Code](#)



Smart parking system

Rachid Azaci, Wahiba Boudjou, Fatma Makouri

Ce projet consiste à concevoir un système de parking intelligent qui met à jour automatiquement le statut actuel des places du parking.

[PDF](#) [Slides](#) [Video](#) [Code](#)



A comprehensive resource for
students projects, eventually ...

Voiture Anti-Collision

Abdelkader Moussa

Le but de ce projet est de réaliser un premier robot radar qui va être capable de reconnaître une zone. De plus, le radar devra pouvoir se déplacer. En effet, si le radar peut se déplacer alors il pourra recouvrir une plus grande surface.

[PDF](#) [Slides](#) [Video](#) [Code](#)



Smart waiting line

Achraf Ben youssef, Mohamed Chahine Fredj, Ahlam Lebsir, Fatah Larti

Vous avez marre des disputes qui ont lieu chaque fois entre vos clients, à propos du prochain qui sera coiffé; la file d'attente intelligente est votre solution.

[PDF](#) [Slides](#) [Video](#) [Code](#)



Smart Parking - a small-scale replica

Farid Meziane, Ahcene Rahmani

Notre projet consiste à réaliser une maquette d'un parking intelligent qui assure un fonctionnement avec la carte Arduino et plusieurs capteur.

[PDF](#) [Slides](#) [Video](#) [Code](#)



GitHub Organization

Institut Galilée - Internet of Things



Institut Galilée

Page regroupant les supports pédagogiques du cours Internet des objets de l'institut Galilée ainsi que les projets étudiant réalisés dans ce cadre.

Université Sorbonne Paris Nord <http://roboticsmind.github.io> massinissa.hamidi@lipn.univ-paris13.fr

[Repositories 55](#) [Packages](#) [People 1](#) [Projects](#)

Pinned repositories

[lab-one-2019](#)
Forked from efrei-paris-sud/lab-one-2019
Hands-on Arduino, ESP32, etc.

[lab-two-2019](#)
Forked from efrei-paris-sud/lab-two-2019
Hands on Fritzing and Serial communication
Python 2

[lab-three-2019](#)
Forked from efrei-paris-sud/lab-three-2019
Hands on I2C and SPI

[lab-four-2019](#)
Forked from efrei-paris-sud/lab-four-2019
Hands-on Wifi with ESP32

[lab-five-2019](#)
Forked from efrei-paris-sud/lab-five-2019
Introduction to IoT cloud platforms. Will be part of "aide-aux-projets" and the idea is to provide assistance to groups to link their IoT solution to the cloud.

[inspirational-video-clips](#)
A selection of inspirational video-clips presenting engineering projects effectively ... May be a source of inspiration for your own projects!

Find a repository... [Type: All](#) [Language: All](#)

[2020-captain-planet](#)

Jupyter Notebook 1 ⚡ 0 ⚡ 1 ⚡ 0 Updated 22 days ago

Top languages

C++ Java C Python

MIDI (Univ. Sorbonne Paris Nord) 50

GitHub Repositories

Screenshot of a GitHub repository page for "guessless".

Repository statistics:

- master branch
- 1 branch
- 0 tags
- 141 commits
- 1 commit by maherlaroussi on May 23, 2019 (commit hash: a4bf22f)

File list:

- docs: ajout rapport (2 years ago)
- lib: Structuration des dossiers (2 years ago)
- src: Allongement de la duré de l'intro (2 years ago)
- GIT.md: Update GIT.md (2 years ago)
- README.md: Add link to video (17 months ago)

README.md content:

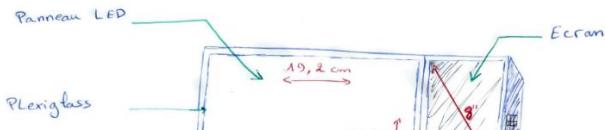
Guessless Project

Promotional video :
https://www.youtube.com/watch?v=Eu7Z_zg069Q&feature=youtu.be

Membres :

- Maher LAAROUSSI (maher.laaroussi@gmail.com) MaherLRS
- Aboubakr CHOUTTA (achoutta@gmail.com) aboubakrCH
- Hamid OUFKIR (hamid.oufkir@yahoo.com) HamidOuF
- Othmane MCHOUCAT (mchouat.o@gmail.com) othmaneMCHOUCAT

Nom du produit IoT : Guessless



The diagram illustrates the internal structure of the Guessless IoT device. It shows a central vertical component labeled "Ecran" (Screen) with a width of "19,2 cm". To the left is a blue rectangular panel labeled "Plexiglass" (Plexiglass). Above the central screen is a blue rectangular panel labeled "Panneau LED" (LED panel). A red arrow points from the bottom of the central screen towards the bottom edge of the device.

Project management inside GitHub

The screenshot shows the GitHub repository 'institut-galilee/guessless'. The 'Projects' tab is selected, displaying five projects:

- Haut-Parleurs: Donner la parole au Raspberry. (1 open, 1 closed)
- LED Panel: Animation du panneau LED. (1 open, 1 closed)
- Crawler informations: Module permettant la récupération de la description d'un objet, des valeurs nutritives d'un aliment et la traduction si nécessaire dans d'autres langages. (1 open, 1 closed)
- Détection d'objet: Utilisation et manipulation de Tensorflow pour la reconnaissance d'objet. (1 open, 1 closed)
- Interface graphique: L'interface graphique que l'utilisateur verra quand il utilisera la table. (1 open, 1 closed)

- Bug report (issues);
- Feature request (pull requests);
- Tasks assignment;
- Prioritize tasks;
- Versioning;

The screenshot shows the GitHub repository 'institut-galilee/guessless' with the 'Projects' board visible. The board is divided into three columns: 'To do', 'In progress', and 'Done'.

- To do:**
 - Fractionnement du code
 - Installation de Tensorflow
 - La détection d'objet marche
 - Error compiling Protobuf
 - Error libprotobuf.so.13
 - Résolution caméra
- In progress:**
 - (empty)
- Done:**
 - #16 opened by maherlaaroussi (to do)
 - #22 opened by maherlaaroussi (to do)
 - #13 opened by maherlaaroussi (announcement)
 - #5 opened by maherlaaroussi (Help wanted)
 - #8 opened by maherlaaroussi (Help wanted)
 - #12 opened by maherlaaroussi (Help wanted)

Evaluation

Sujet
[G] Difficulté
[G] Originalité
[P] Difficulté de la tâche

Présentation
[G] slides
[G] qualité générale
[P] Présentation personnelle
[P] Réponse aux questions
[G] Démo du travail effectué

Rapport
[G] Qualité présentation
[G] qualité état de l'art
[G] présentation de l'étude
[G] présentation de la solution
[G] aspects généraux
[P] Contribution personnelle

Vidéo
[P] Contribution personnelle
[G] qualité générale

Git [G]
activité en commits, issues, pull requests
pertinence des commits

Suivi lors des Tps
Présence
Engagement

Code
Aspects généraux
Fonctionnel?

Hardware
Aspects généraux

Time to create your GitHub accounts

Use usernames that correspond to
your real names (easy for us to find
out who is who!);

Time to team-up

- Based on personal affinities or shared interests;
- Flexible until the next course, no changes allowed after that;
- **3 members maximum**, no singletons;

**Even if you work in teams, final
marks are individual!**

Time to team-up (Cont'd)

- A GitHub repository will be created for each team;
- All your software developments will be done in GitHub (commits, issues, etc.) will be considered;
- All documents, reports, consulted resources, hardware designs, etc. have to be put in your respective project repository;
- A dedicated organization page, <https://github.com/institut-galilee>, will host your projects;
- You can find last years IoT projects in this organization page. You can also check the “sister” organization page, <https://github.com/efrei-paris-sud>.

Time to team-up (Cont'd)

- Team name;
- Team members:
 - GitHub username of each member;
- Description of your preliminary idea(s) (a small paragraph);

Send one email per group to
hamidi@lipn.univ-paris13.fr
cc: ao@lipn.univ-paris13.fr

**Deadline to transmit all these pieces of
information is today!**

Hardware & Software

Useful tools

- Git (small tutorial <https://rogerdudler.github.io/git-guide/>)
- Linux (small tutorial)
- GitHub projects (<https://github.com/institut-galilee/guessless>)
- Fritzing (<https://fritzing.org/home/>)
- Gantt diagrams (<https://www.teamgantt.com/>)

Hardware provided by Institut Galilée

- Arduino kits
- ESP32
- BME380

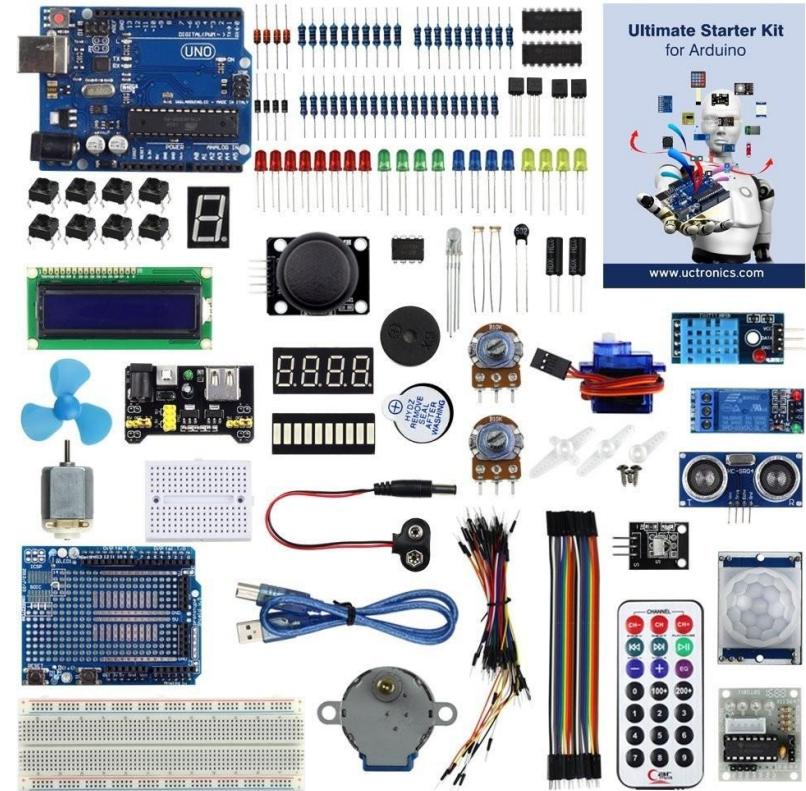


Depending on the respective needs of each team, you can acquire additional hardware, such as microphones, pulse sensors, etc. on your own.

You have to choose additional hardware
carefully and accordingly!

Arduino Kits

- Arduino UNO;
- Servo Motor;
- Movement sensor;
- Ultrasonic sensor;
- LCD;
- etc.

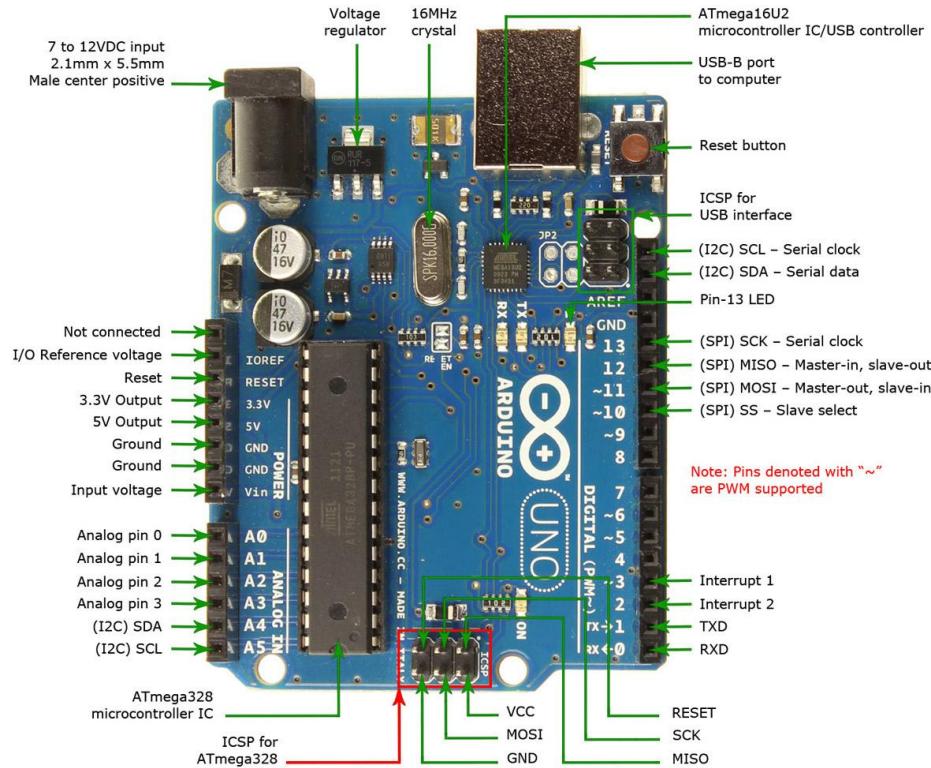


Arduino

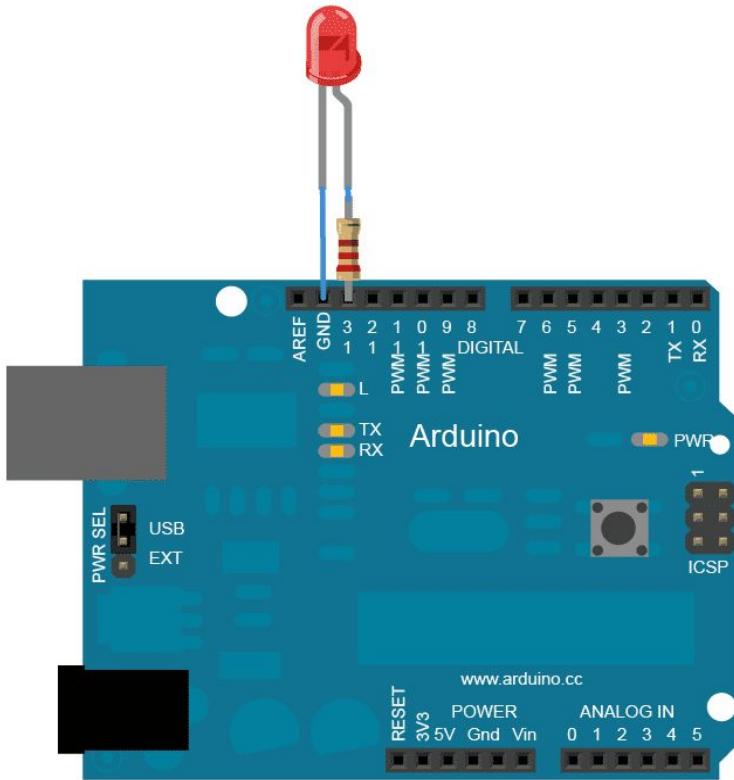
- No operating system;
- Many CPUs over time;

Arduino	
	ARDUINO Genuino
	Arduino Uno SMD R3
Developer	arduino.cc
Manufacturer	Many
Type	Single-board microcontroller
Operating system	None
CPU	Atmel AVR (8-bit), ARM Cortex-M0+ (32-bit), ARM Cortex-M3 (32-bit), Intel Quark (x86) (32-bit)
Memory	SRAM
Storage	Flash, EEPROM
Website	www.arduino.cc

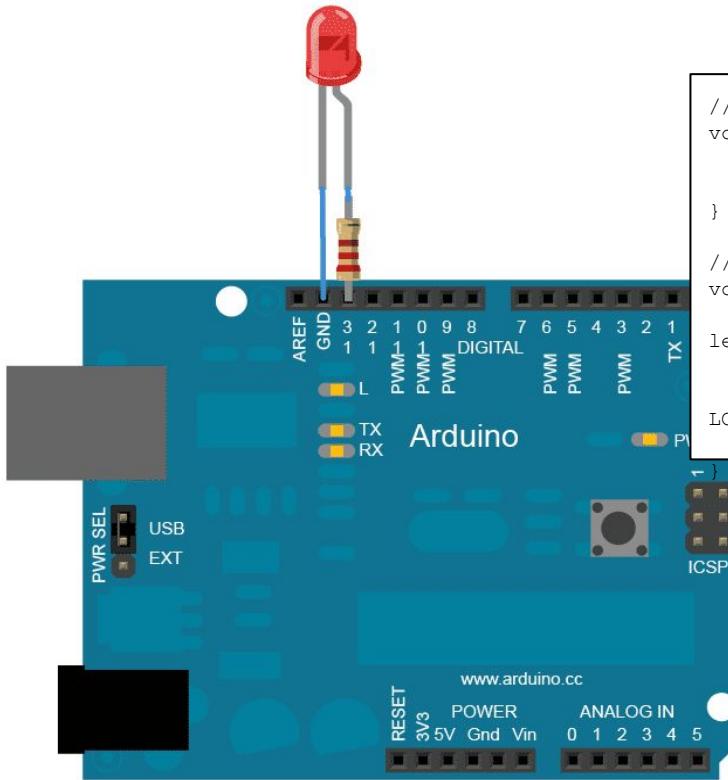
Arduino UNO (pinout)



Arduino UNO (blink LED)



Arduino UNO (blink LED)



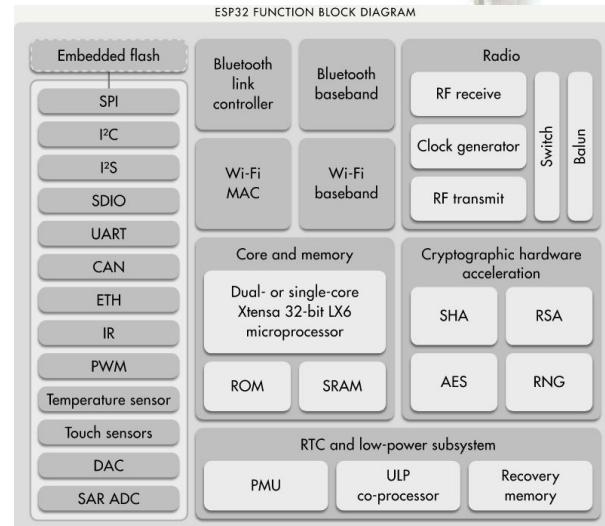
```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage
  LOW;                                // wait for a second
```

ESP32

- **Processors:**
 - **Main processor:** Tensilica Xtensa 32-bit LX6 microprocessor
 - **Cores:** 2 or 1 (depending on variation)
All chips in the ESP32 series are dual-core except for ESP32-S0WD, which is single-core.
 - **Clock frequency:** up to 240 MHz
 - **Performance:** up to 600 DMIPS
 - **Ultra low power co-processor:** allows you to do ADC conversions, computation, and level thresholds while in deep-sleep mode.
- **Wireless connectivity:**
 - **Wi-Fi:** 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 Mbit/s)
 - **Bluetooth:** v4.2 BR/EDR and Bluetooth Low Energy (BLE)
- **Memory:**
 - **Internal memory:**
 - **ROM:** 448 KIB
For booting and core functions.
 - **SRAM:** 520 KIB
For data and instruction.
 - **RTC fast SRAM:** 8 KIB
For data storage and main CPU during RTC Boot from the deep-sleep mode.
 - **RTC slow SRAM:** 8 KIB
For co-processor accessing during deep-sleep mode.
 - **eFuse:** 1 Kbit
Of which 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including Flash-Encryption and Chip-ID.
 - **Embedded flash:**
Flash connected internally via IO16, IO17, SD_CMD, SD_CLK, SD_DATA_0 and SD_DATA_1 on ESP32-D2WD and ESP32-PICO-D4.
 - 0 MiB (ESP32-D2WDQ6, ESP32-D2WD, and ESP32-S0WD chips)
 - 2 MiB (ESP32-D2WD chip)
 - 4 MiB (ESP32-PICO-D4 SIP module)
 - **External flash & SRAM:** ESP32 supports up to four 16 MiB external QSPI flashes and SRAMs with hardware encryption based on AES to protect developers' programs and data. ESP32 can access the external QSPI flash and SRAM through high-speed caches.
 - Up to 16 MiB of external flash are memory-mapped onto the CPU code space, supporting 8-bit, 16-bit and 32-bit access. Code execution is supported.
 - Up to 8 MiB of external flash/SRAM memory are mapped onto the CPU data space, supporting 8-bit, 16-bit and 32-bit access. Data-read is supported on the flash and SRAM. Data-write is supported on the SRAM.
- **Peripheral input/output:** Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), PC (Inter-Integrated Circuit), UART (universal asynchronous receiver/transmitter), CAN 2.0 (Controller Area Network), SPI (Serial Peripheral Interface), I²S (Integrated Inter-IC Sound), RMII (Reduced Media-Independent Interface), PWM (pulse width modulation), and more.
- **Security:**
 - IEEE 802.11 standard security features all supported, including WFA, WPA/WPA2 and WAPI
 - Secure boot
 - Flash encryption
 - 1024-bit OTP, up to 768-bit for customers
 - Cryptographic hardware acceleration: AES, SHA-2, RSA, elliptic curve cryptography (ECC), random number generator (RNG)

Improvement over
the ESP8266
which was, initially,
a WiFi module!



ESP32

- Processors:**
 - Main processor:** Tensilica Xtensa 32-bit LX6 micropro
 - Cores:** 2 or 1 (depending on variation)
 - All chips in the ESP32 series are dual-core except for ESP32
 - Clock frequency:** up to 240 MHz
 - Performance:** up to 600 DMIPS
 - Ultra low power co-processor:** allows you to do ADC
- Wireless connectivity:**
 - Wi-Fi:** 802.11 b/g/n/e/i (802.11n @ 2.4 GHz up to 150 M
 - Bluetooth:** v4.2 BR/EDR and Bluetooth Low Energy (E
- Memory:**
 - Internal memory:**
 - ROM:** 448 kB
 - For booting and core functions.
 - SRAM:** 520 kB
 - For data and instruction.
 - RTC fast SRAM:** 8 kB
 - For data storage and main CPU during RTC Boot from the d
 - RTC slow SRAM:** 8 kB
 - For co-processor accessing during deep-sleep mode.
 - eFuse:** 1 Kbit
 - Of which 256 bits are used for the system (MAC address an Encryption and Chip-ID).
 - Embedded flash:**
 - Flash connected internally via IO16, IO17, SD_CMD, SD_CLK,
 - 0 MiB (ESP32-D0WDQ6, ESP32-D0WD, a
 - 2 MiB (ESP32-D2WD chip)
 - 4 MiB (ESP32-PICO-D4 SIP module)
 - External flash & SRAM:** ESP32 supports up to four 16 protect developers' programs and data. ESP32 can ac
 - Up to 16 MiB of external flash are memory-map execution is supported.
 - Up to 8 MiB of external flash/SRAM memory are read is supported on the flash and SRAM. Data-
ESP32 chips with embedded flash do not support the address map
- Peripheral input/output:** Rich peripheral interface with DM/ analog converter), PC (Inter-Integrated Circuit), UART (unive Peripheral Interface), IIS (Integrated Inter-IC Sound), RMII (R
- Security:**
 - IEEE 802.11 standard security features all supported, i
 - Secure boot
 - Flash encryption
 - 1024-bit OTP, up to 768-bit for customers
 - Cryptographic hardware acceleration: AES, SHA-2, RS

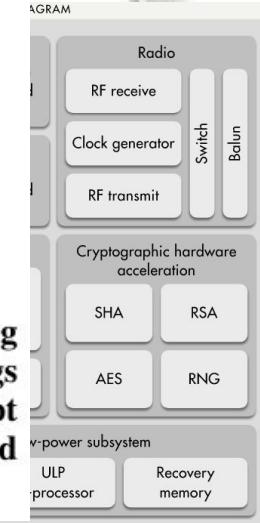
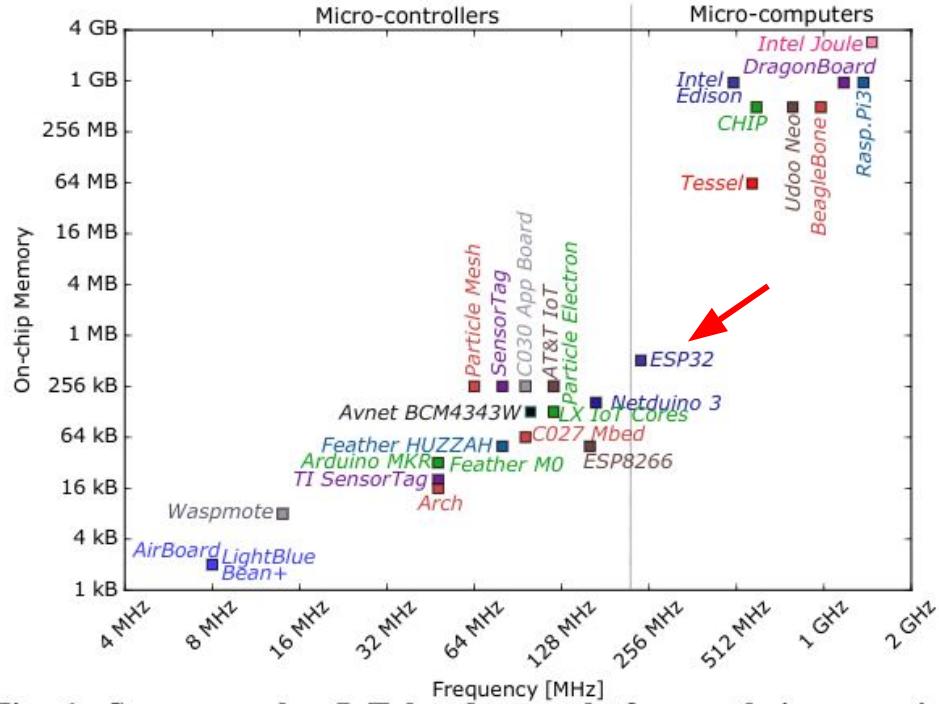
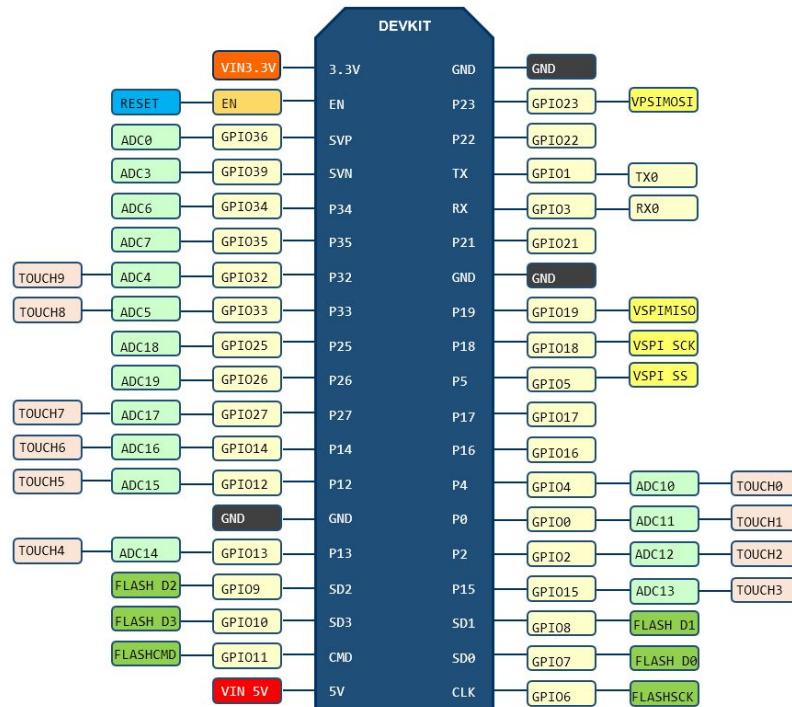


Fig. 1: Some popular IoT hardware platforms, their operating frequencies, and on-chip internal memories (each square belongs to the platform annotated with the same color next to it). Except the micro-computers –which are suitable for gateways–, the end devices have limited memory and processing capabilities.

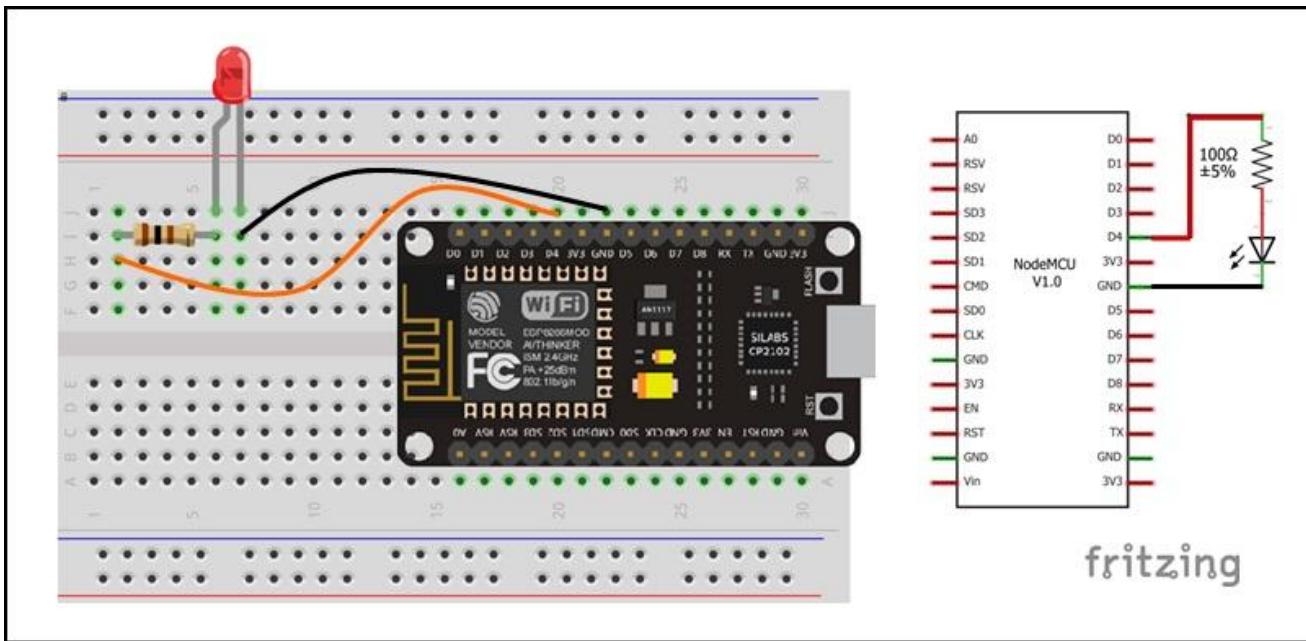
ESP32 (pinout)

PIN DEFINITION

www.ai-thinker.com



ESP32 (Blink LED)



ZigBee Modules

Module sans fil CC2530
pour Module sans fil
ZIGBEE



www.ti.com

CC2530F32, CC2530F64
CC2530F128, CC2530F256

SWRS081B—APRIL 2009—REVISED FEBRUARY 2011

A True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications

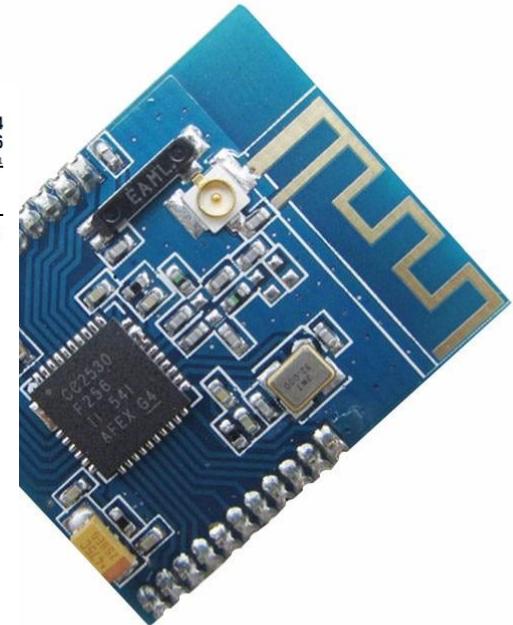
Check for Samples: [CC2530F32](#), [CC2530F64](#), [CC2530F128](#), [CC2530F256](#)

FEATURES

- RF/Layout
 - 2.4-GHz IEEE 802.15.4 Compliant RF Transceiver
 - Excellent Receiver Sensitivity and Robustness to Interference
 - Programmable Output Power Up to 4.5 dBm
 - Very Few External Components
 - Only a Single Crystal Needed for Asynchronous Networks
 - 6-mm × 6-mm QFN40 Package
 - Suitable for Systems Targeting Compliance With Worldwide Radio-Frequency Regulations: ETSI EN 300 328 and EN 300 440 (Europe), FCC CFR47 Part 15 (US) and ARIB STD-T-66 (Japan)
 - Low Power
 - Active-Mode RX (CPU Idle): 24 mA
 - Active Mode TX at 1 dBm (CPU Idle): 29 mA
 - Power Mode 1 (4 µs Wake-Up): 0.2 mA
 - Power Mode 2 (Sleep Timer Running): 1 µA
 - Power Mode 3 (External Interrupts): 0.4 µA
 - Wide Supply-Voltage Range (2 V–3.6 V)
 - Microcontroller
 - High-Performance and Low-Power 8051 Microcontroller Core With Code Prefetch
 - 32-, 64-, 128-, or 256-KB In-System-Programmable Flash
 - 8-KB RAM With Retention in All Power Modes
 - Hardware Debug Support
 - Peripherals
 - Powerful Five-Channel DMA
 - Integrated High-Performance Op-Amp and Ultralow-Power Comparator
 - IEEE 802.15.4 MAC Timer, General-Purpose Timers (One 16-Bit, Two 8-Bit)
 - IR Generation Circuitry
 - 32-kHz Sleep Timer With Capture
 - CSMA/CA Hardware Support
 - Accurate Digital RSSI/LQI Support
 - Battery Monitor and Temperature Sensor
 - 12-Bit ADC With Eight Channels and Configurable Resolution
 - AES Security Coprocessor
 - Two Powerful USARTs With Support for Several Serial Protocols
 - 21 General-Purpose I/O Pins (19 × 4 mA, 2 × 20 mA)
 - Watchdog Timer
- Development Tools
 - CC2530 Development Kit
 - CC2530 ZigBee® Development Kit
 - CC2530 RemoTI™ Development Kit for RF4CE
 - SmartRF™ Software
 - Packet Sniffer
 - IAR Embedded Workbench™ Available

APPLICATIONS

- 2.4-GHz IEEE 802.15.4 Systems
- RF4CE Remote Control Systems (64-KB Flash and Higher)
- ZigBee Systems (256-KB Flash)
- Home/Building Automation
- Lighting Systems
- Industrial Control and Monitoring
- Low-Power Wireless Sensor Networks
- Consumer Electronics
- Health Care



datasheet <http://www.ti.com/lit/ds/symlink/cc2530.pdf>

ZigBee-based smart-home products <https://xiaomi-mi.com/sockets-and-sensors/>

Massinissa HAMIDI (Univ. Sorbonne Paris Nord)

LoRa Modules

LoRa 915MHz SX1276 rf
émetteur-récepteur

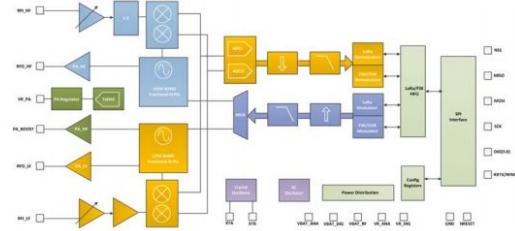


SX1276/77/78/79

WIRELESS & SENSING PRODUCTS

DATASHEET

SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver



GENERAL DESCRIPTION

The SX1276/77/78/79 transceivers feature the LoRa™ long range modem that provides ultra-long range spread spectrum communication and high interference immunity whilst minimising current consumption.

Using Semtech's patented LoRa™ modulation technique SX1276/77/78/79 can achieve a sensitivity of over -148dBm using a low cost crystal and bill of materials. The high sensitivity combined with the integrated +20 dBm power amplifier yields industry leading link budget making it optimal for any application requiring range or robustness. LoRa™ provides significant advantages in both blocking and selectivity over conventional modulation techniques, solving the traditional design compromise between range, interference immunity and energy consumption.

These devices also support high performance (G)FSK modes for systems including WMBus, IEEE802.15.4g. The SX1276/77/78/79 deliver exceptional phase noise, selectivity, receiver linearity and IIP3 for significantly lower current consumption than competing devices.

ORDERING INFORMATION

Part Number	Delivery	MOQ / Multiple
SX1276IMLRT	T&R	3000 pieces
SX1277IMLRT	T&R	3000 pieces
SX1278IMLRT	T&R	3000 pieces
SX1279IMLRT	T&R	3000 pieces
SX1276WS	Wafer Form	1 Wafer (2000 dies)

KEY PRODUCT FEATURES

- LoRa™ Modem
- 168 dB maximum link budget
- +20 dBm - 100 mW constant RF output vs. V supply
- +14 dBm high efficiency PA
- Programmable bit rate up to 300 kbps
- High sensitivity: down to -148 dBm
- Bullet-proof front end: IIP3 = -11 dBm
- Excellent blocking immunity
- Low RX current of 9.9 mA, 200 nA register retention
- Fully integrated synthesizer with a resolution of 61 Hz
- FSK, GFSK, MSK, GMSK, LoRa™ and OOK modulation
- Built-in bit synchronizer for clock recovery
- Preamble detection
- 127 dB Dynamic Range RSSI
- Automatic RF Sense and CAD with ultra-fast AFC
- Packet engine up to 256 bytes with CRC
- Built-in temperature sensor and low battery indicator

APPLICATIONS

- Automated Meter Reading.
- Home and Building Automation.
- Wireless Alarm and Security Systems.
- Industrial Monitoring and Control
- Long range Irrigation Systems

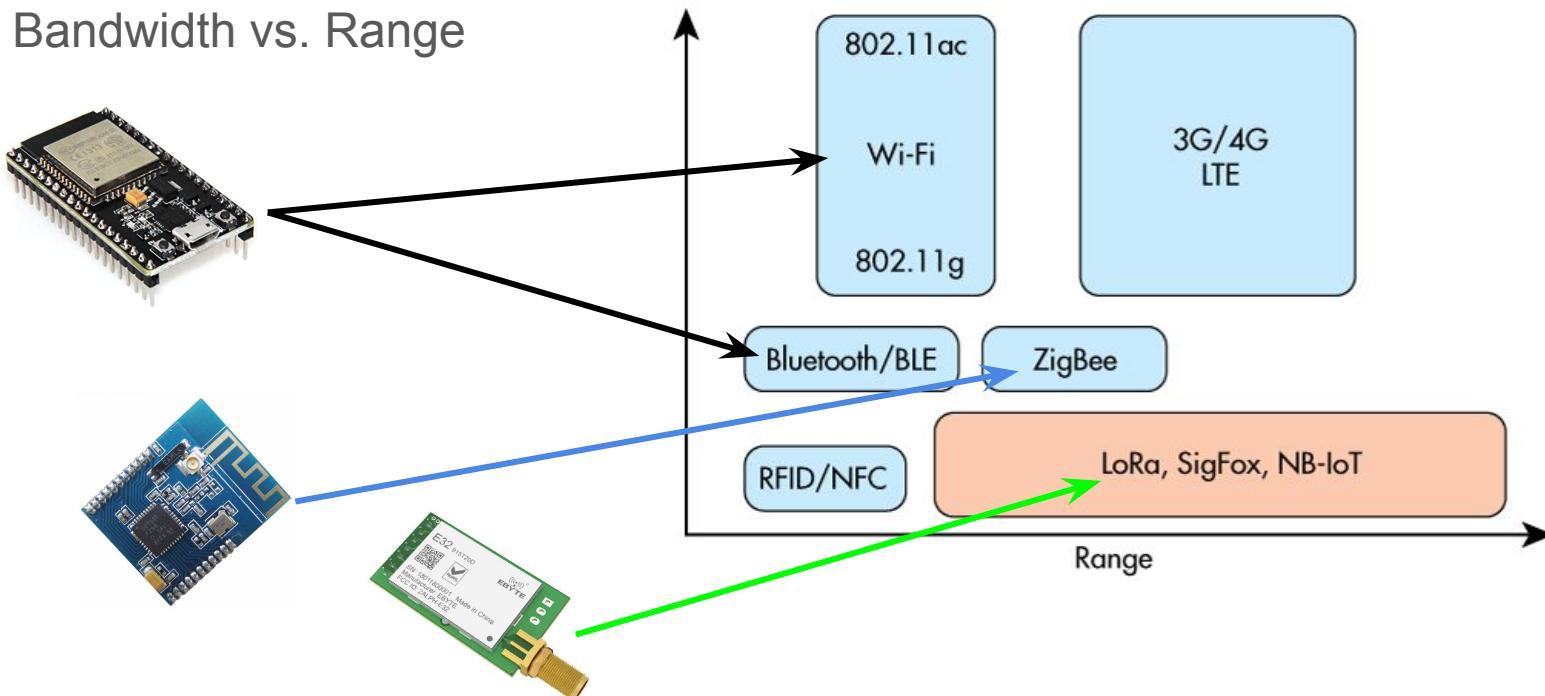
https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V6.pdf

Some projects tagged 'lorawan' <https://hackaday.io/projects?tag=lorawan>

<https://www.thethingsnetwork.org/>

From a connectivity perspective

Bandwidth vs. Range



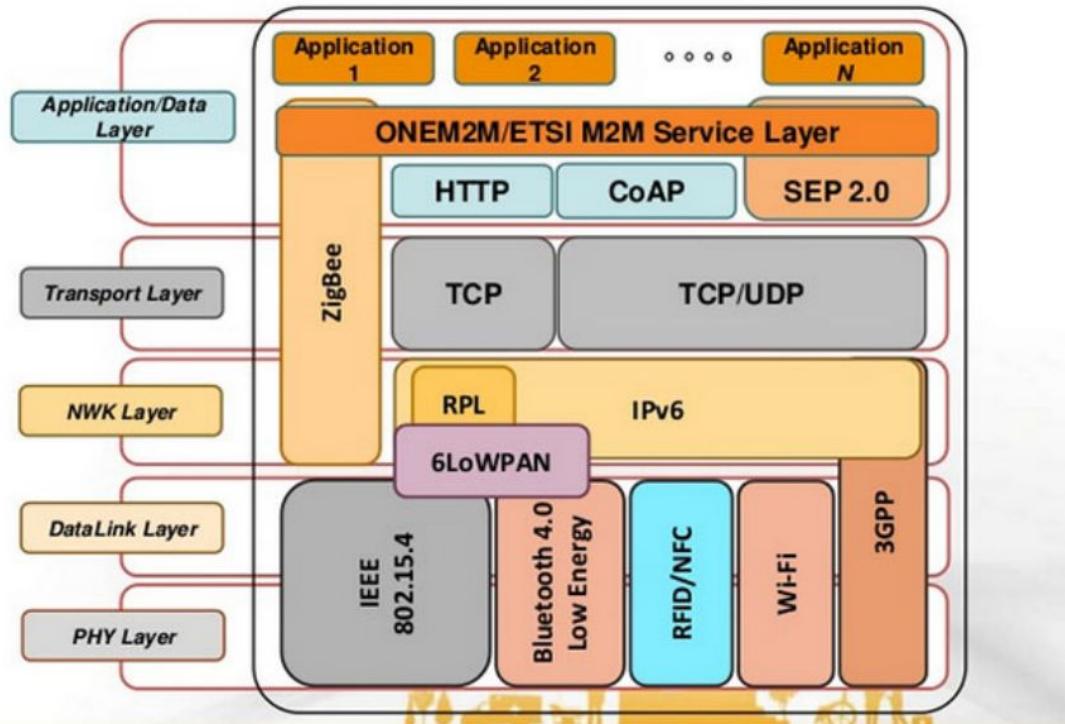
From a connectivity perspective (Cont'd)

Network types



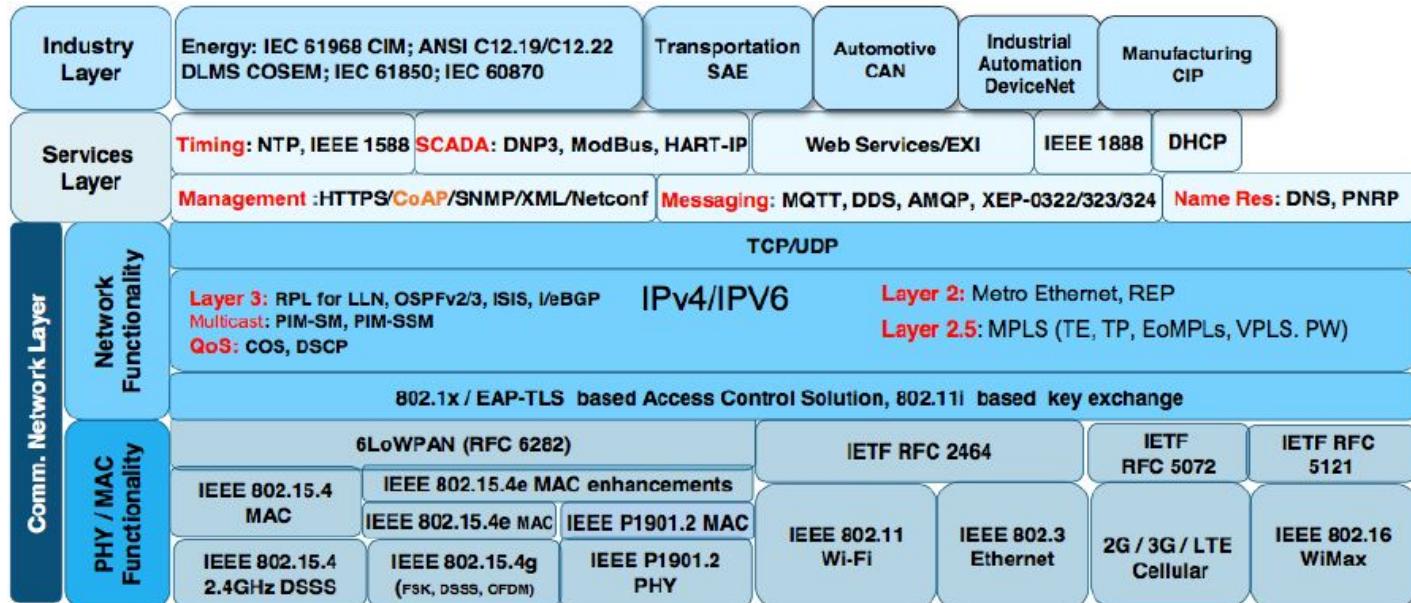
Connectivity: examples

From a TCP/IP-model perspective



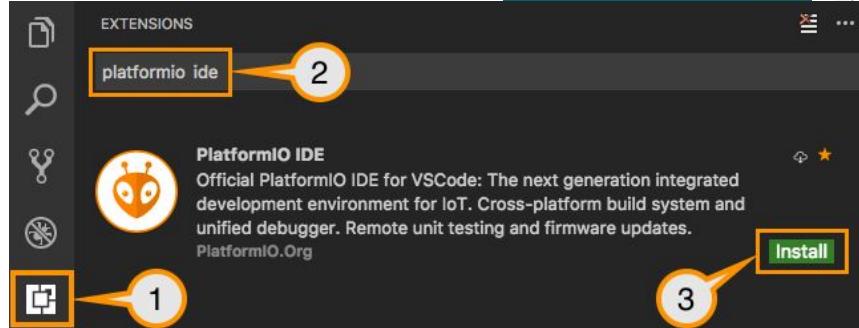
Connectivity: examples (cont'd)

Industrial context



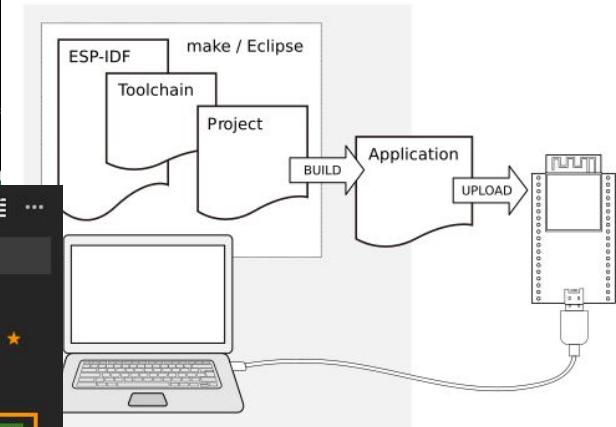
Software

- Arduino IDE
- ESP-IDF based on GNU-Make
 - make flash
 - make monitor
- PlatformIO



The screenshot shows the Arduino IDE interface. The main window displays the "Blink" sketch, which is a standard "Blink" example. The terminal window on the right shows the upload process to an ESP8266 board connected via USB. The output in the terminal includes:

```
ets Jun 8 2016 00:22:57
rst:0x0 (RSTOMD RTC RESET), boot:0x13 (SPI_FAST_FLASH_BOOT)
configSip: 0, SPMEn: 0x00
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x20000008, len:16
load:0x00000000, len:1760
load:0x40007800, len:6664
load:0x40008000, len:252
entry 0x40000034
```



Compilation croisée

La compilation croisée permet de créer des exécutables depuis une certaine architecture pour une autre. Cela permet de créer des paquet pour les systèmes pour lesquels on n'a pas la main.

En fonction de l'architecture cible (ARM, Atmel AVR, x86, Xtensa, etc.), on doit se doter d'un environnement de compilation croisée à l'aide duquel, nous allons générer un code exécutable spécifique à l'architecture cible.

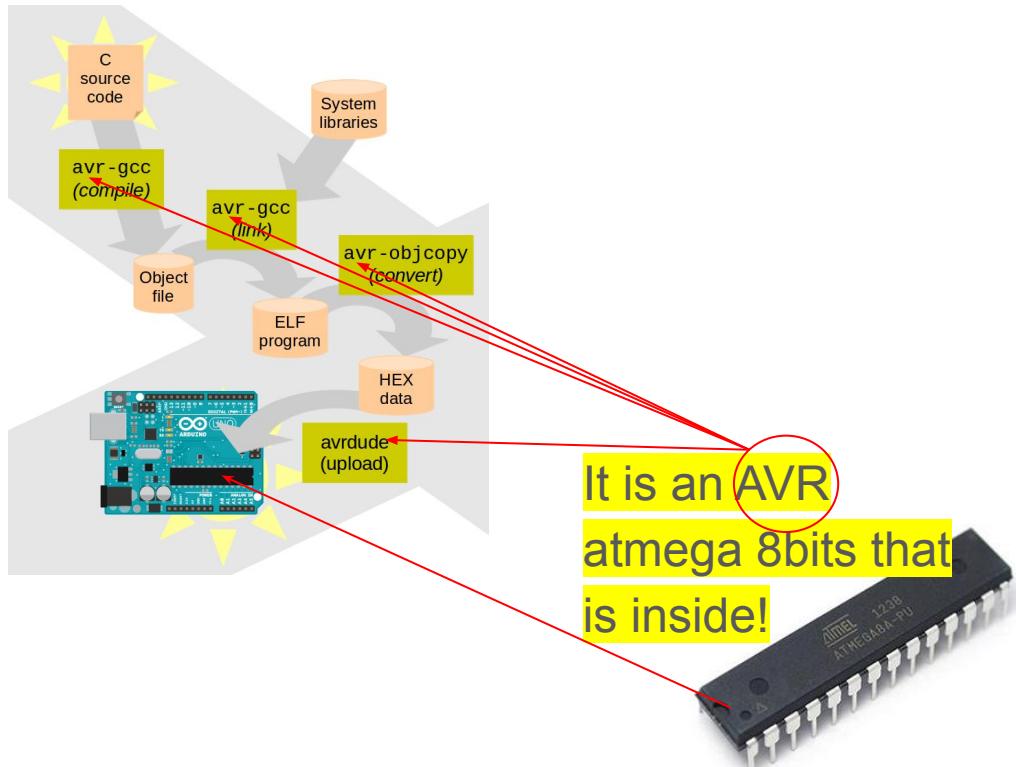
L'environnement de compilation croisée, appelé communément toolchain, se compose de :

- ▶ GNU Compiler Collection (GNU) ;
- ▶ binutils ;
- ▶ debugger ;
- ▶ C library (glibc, ulibc, etc.) ;

Cross-Compiling environment



The case of Arduino UNO



CombinedCodes2 | Arduino 1.6.5

```
#include <Servo.h>
int Buzzer = 7;
Servo myServo;
void setup()
{
    // initialize digital pin 13 as an output.
    pinMode(13, OUTPUT); // for LED
    pinMode(Buzzer, OUTPUT);
    myServo.attach(9);
}
// the loop function runs over and over again forever
void loop()
{
    digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
    delay(1000);
    digitalWrite(Buzzer, HIGH); // turns buzzer on
    delay(1000);
    digitalWrite(Buzzer, LOW); // turns buzzer off
    delay(1000);
    myServo.write(130);
    delay(3000);
    myServo.write(0);
    delay(3000);
}
```

Variable Declaration Section
Declares:
Libraries
Pin Numbers
Components
Sensors
Variables and Values

Setup Section
Sets Up:
Sensors & Components as INPUTS or OUTPUTS
Calculations and Algorithms
Serial functions

Loop Section
Loops or Repeats Actions

Code from "BLINK" Project

Code from "Buzzer On & OFF" Project

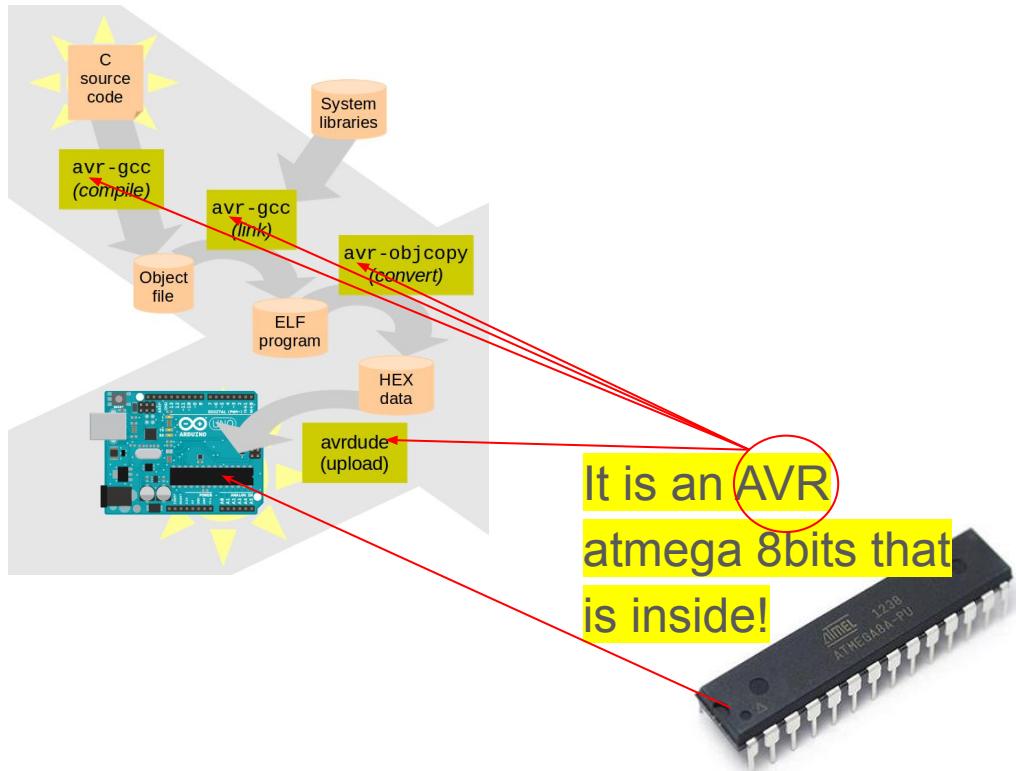
Code from "Simple Servo" Project

Each chunk of code is separated using Curly Braces {}. This is one way to combine several projects into one. Most of the time you will need to balance them by starting with an open brace { and then after the code chunk you will use a close brace }. However sometimes that is not the case as shown in the last chunk in this example. When you receive errors, sometimes you will need to adjust the braces and will not always follow the rule.

Done compiling.

34 Arduino Uno on /dev/cu.usbmodem1411

The case of Arduino UNO



CombinedCodes2 | Arduino 1.6.5

```
CombinedCodes2
#include <Servo.h>
int Buzzer = 7;
Servo myServo;
void setup()
{
    // initialize digital pin 13 as an output.
    pinMode(13, OUTPUT); // for LED
    pinMode(Buzzer, OUTPUT);
    myServo.attach(9);
}
// the loop function runs over and over again forever
void loop()
{
    digitalWrite(13, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(13, LOW); // turn the LED off by making the voltage LOW
    delay(1000);
    digitalWrite(Buzzer, HIGH); // turns buzzer on
    delay(1000);
    digitalWrite(Buzzer, LOW); // turns buzzer off
    delay(1000);
    myServo.write(130);
    delay(3000);
    myServo.write(0);
    delay(3000);
}
```

Variable Declaration Section

Setup Section

Sets Up:
Sensors & Components as INPUTS or OUTPUTS
Calculations and Algorithms
Serial functions

Loop Section

Loops or Repeats Actions

Code from "BLINK" Project

Code from "Buzzer On & OFF" Project

Code from "Simple Servo" Project

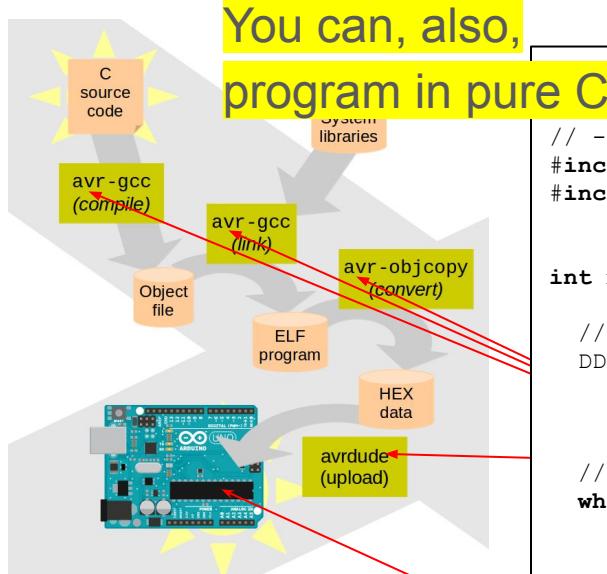
Each chunk of code is separated using Curly Braces {}. This is one way to combine several projects into one. Most of the time you will need to balance them by starting with an open brace { and then after the code chunk you will use a close brace }. However sometimes that is not the case as shown in the last chunk in this example. When you receive errors, sometimes you will need to adjust the braces and will not always follow the rule.

Done compiling.

34

Arduino Uno on /dev/cu.usbmodem1411

The case of Arduino UNO



```
// ----- Preamble ----- //
#include <avr/io.h>
#include <util/delay.h>

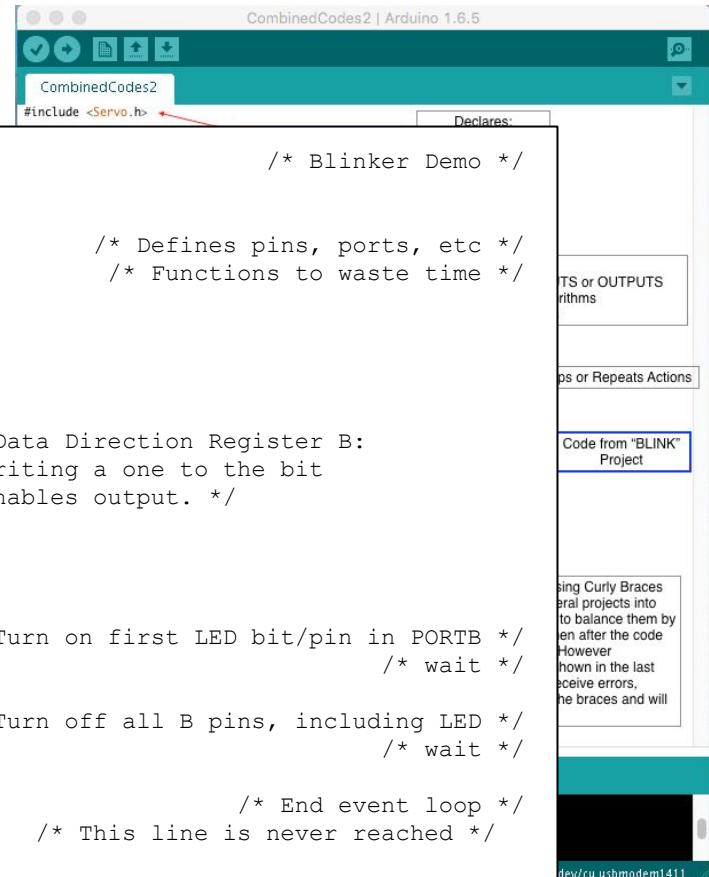
int main(void) {
    // ----- Inits ----- //
    DDRB |= 0b00000001;           /* Data Direction Register B:
                                    writing a one to the bit
                                    enables output. */

    // ----- Event loop ----- //
    while (1) {
        PORTB = 0b00000001;       /* Turn on first LED bit/pin in PORTB */
        _delay_ms(1000);          /* wait */

        PORTB = 0b00000000;       /* Turn off all B pins, including LED */
        _delay_ms(1000);          /* wait */

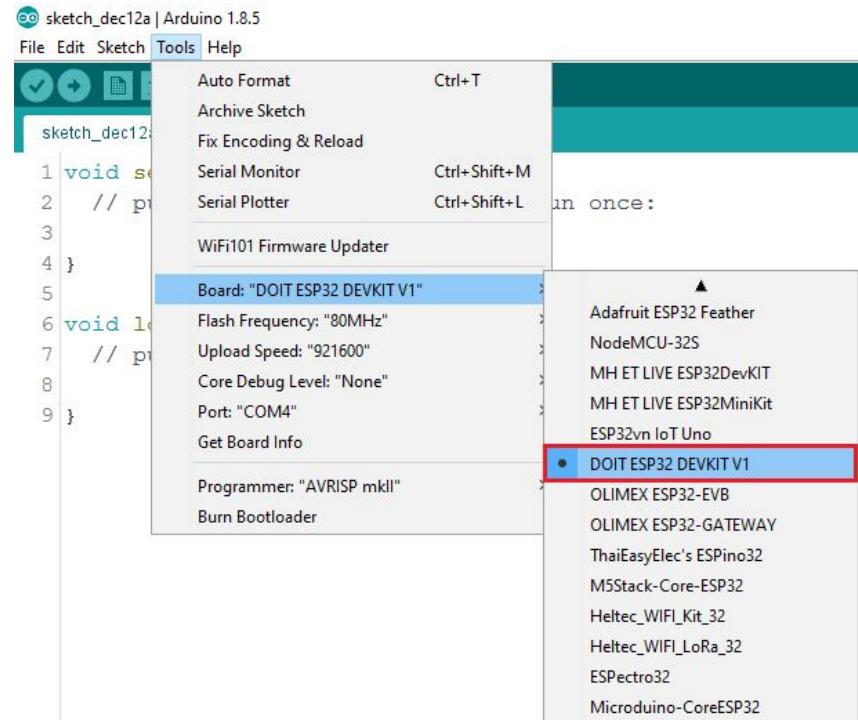
    }
    return 0;
}

/* End event loop */
/* This line is never reached */
```



Arduino on ESP32

Espressif developed arduino-esp32, an abstraction layer to write Arduino code for the ESP32

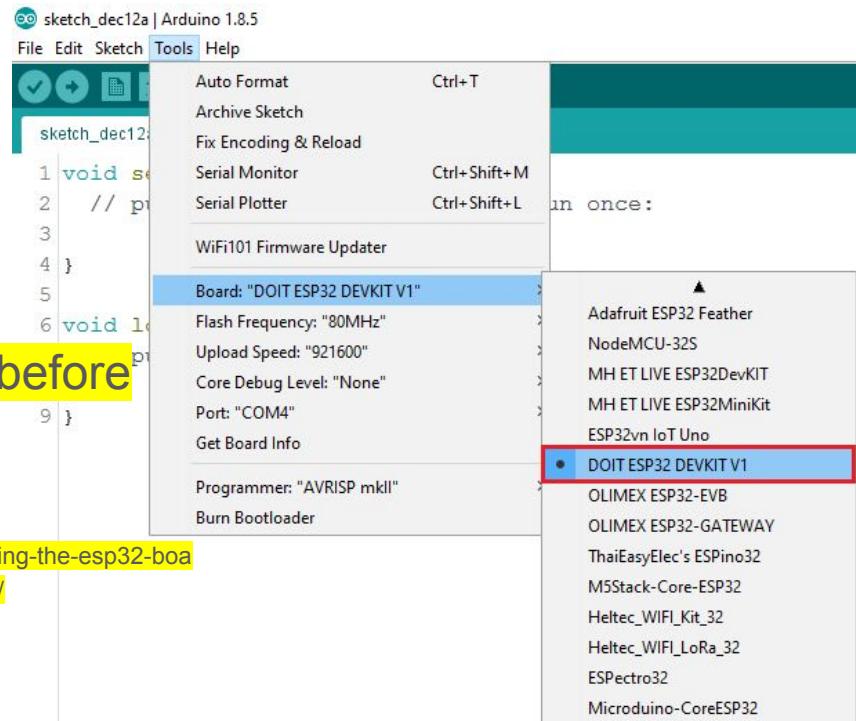


Arduino on ESP32

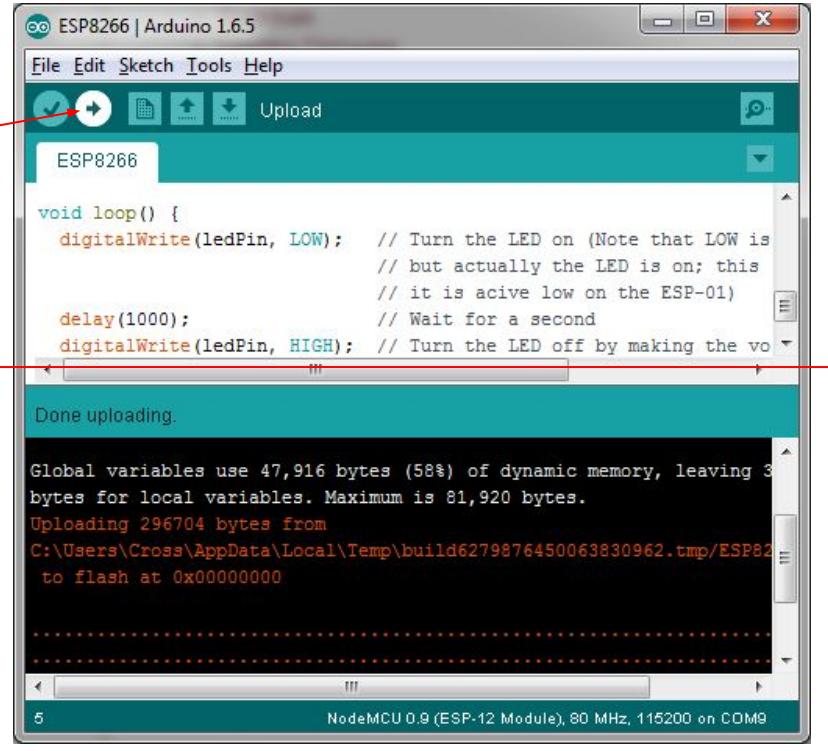
Espressif developed arduino-esp32, an abstraction layer to write Arduino code for the ESP32

You have to install
arduino-esp32 before
you can use it !
check here

<https://randomnerdtutorials.com/installing-the-esp32-board-in-arduino-ide-windows-instructions/>



Flashing device: Arduino IDE



The screenshot shows the Arduino IDE interface for an ESP8266 board. The code in the editor is:

```
void loop() {
    digitalWrite(ledPin, LOW); // Turn the LED on (Note that LOW is
                             // but actually the LED is on; this
                             // it is active low on the ESP-01)
    delay(1000); // Wait for a second
    digitalWrite(ledPin, HIGH); // Turn the LED off by making the vo
```

The serial monitor window displays the output of the upload process:

```
Done uploading.

Global variables use 47,916 bytes (58%) of dynamic memory, leaving 3
bytes for local variables. Maximum is 81,920 bytes.
Uploading 296704 bytes from
C:\Users\Cross\AppData\Local\Temp\build6279876450063830962.tmp\ESP82
to flash at 0x00000000
....
```

Anatomy of an ESP32 program

- C code;
- FreeRTOS primitives;
- GPIO driver primitives;

```
25 #
26 # Security features
27 #
28 CONFIG_SECURE_BOOT_ENABLED=
29 CONFIG_FLASH_ENCRYPTION_ENABLED=
30 #
31 #
32 # Serial flasher config
33 #
34 CONFIG_ESPTOOLPY_PORT="/dev/ttyUSB0"
35 CONFIG_ESPTOOLPY_BAUD_115200B=y
36 CONFIG_ESPTOOLPY_BAUD_230400B=
37 CONFIG_ESPTOOLPY_BAUD_921600B=
38 CONFIG_ESPTOOLPY_BAUD_2MB=
39 CONFIG_ESPTOOLPY_BAUD_OTHER=
40 CONFIG_ESPTOOLPY_BAUD_OTHER_VAL=115200
41 CONFIG_ESPTOOLPY_BAUD=115200
42 CONFIG_ESPTOOLPY_COMPRESSED=y
```

```
</efrei/lab-four/
► build/
► doc/
▼ lib/BME280_driver/
► examples/
► include/
► selftest/
bme280.c
component.mk
LICENSE
README.md
► server/
▼ src/
► include/
app_layer.c
component.mk
connectivity.c
formatting.c
main.c
net_layer.c
phy_layer.c
sensor_reading.c
trans_layer.c
transmission.c
Makefile
README.md
sdkconfig
```

```
1 /* Blink Example
2
3 This example code is in the Public Domain (or CC0 licensed, at your option.)
4
5 Unless required by applicable law or agreed to in writing, this
6 software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
7 CONDITIONS OF ANY KIND, either express or implied.
8 */
9 #include <stdio.h>
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "driver/gpio.h"
13 #include "sdkconfig.h"
14
15 /* Can run 'make menuconfig' to choose the GPIO to blink,
16 or you can edit the following line and set a number here.
17 */
18 #define BLINK_GPIO CONFIG_BLINK_GPIO
19
20 void blink_task(void *pvParameter)
21 {
22     /* Configure the IOMUX register for pad BLINK_GPIO (some pads are
23 mixed to GPIO on reset already, but some default to other
24 functions and need to be switched to GPIO. Consult the
25 Technical Reference for a list of pads and their default
26 functions.)
27 */
28     gpio_pad_select_gpio(BLINK_GPIO);
29     /* Set the GPIO as a push/pull output */
30     gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);
31     while(1) {
32         /* Blink off (output low) */
33         gpio_set_level(BLINK_GPIO, 0);
34         vTaskDelay(1000 / portTICK_PERIOD_MS);
35         /* Blink on (output high) */
36         gpio_set_level(BLINK_GPIO, 1);
37         vTaskDelay(1000 / portTICK_PERIOD_MS);
38     }
39 }
40
41 void app_main()
42 {
43     xTaskCreate(&blink_task, "blink_task", configMINIMAL_STACK_SIZE, NULL, 5, NULL);
44 }
```

ESP32 : ESP IDF

Framework de développement pour l'ESP est composé d'un ensemble de bibliothèques, pour la plupart implémentées par la société Espressif. Il permet d'exploiter les fonctionnalités fournies par cette SoC dans le but de faciliter le développement de l'IoT. Il

- ▶ Fournit une interface pour la communication série avec différents protocoles ;
- ▶ inclut une implémentation de la pile TCP/IP avec une faible empreinte mémoire ;
- ▶ Fournit une implémentation pour la communication par BLE et WiFi ;
- ▶ Impléme le protocole de communication CoAP ;
- ▶ Fournit une bibliothèque (libsodium) pour le cryptage, décryptage, signatures, etc.

ESP32 : FreeRTOS

©Internet des
Objets – Aomar
OSMANI

Composants
principaux de l'IoT

C'est le système d'exploitation pour microcontrôleurs utilisé dans l'ESP32 (<https://fr.wikipedia.org/wiki/FreeRTOS>). Il est :

- ▶ temps réel
- ▶ faible empreinte mémoire
- ▶ préemption et
- ▶ open source

Il n'implémente aucun pilote matériel et est composé de quelques fichiers en C. L'image binaire du noyau pèse entre 4ko et 9Ko. Le kit minimal comporte peu de fonctions pour la gestion des tâches et de la mémoire.

Wait ... ESP-IDF project structure

- You need to specify a specific structure in order to successfully compile your program

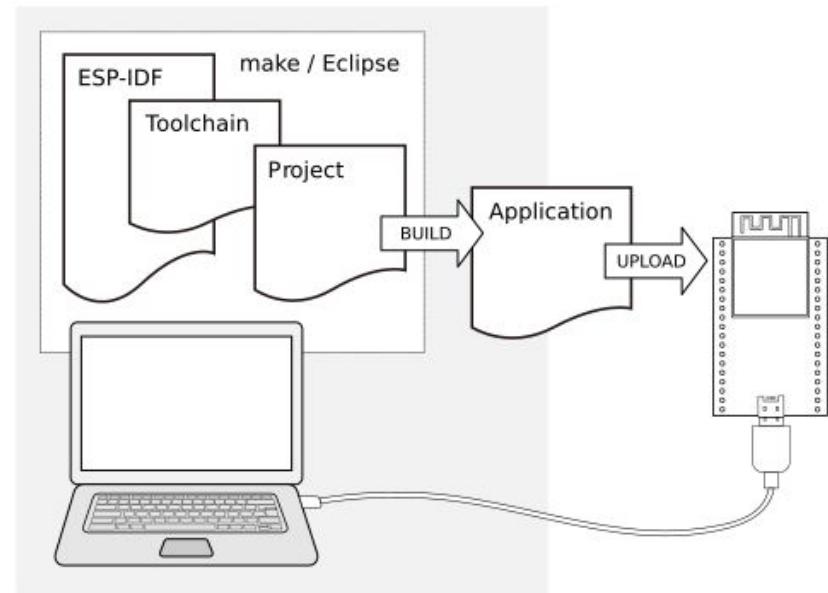
```
17  
18 PROJECT_NAME := iot-labs  
19 EXTRA_COMPONENT_DIRS := $(PWD)/lib $(PWD)/src/include  
20 SRCDIRS := src  
21 #SRCDIRS := src/  
22  
23 include $(IDF_PATH)/make/project.mk
```

Makefile

```
</efrei/lab-four/  
  build/  
  doc/  
  lib/BME280_driver/  
    examples/  
    include/  
    selftest/  
    bme280.c  
    component.mk  
    LICENSE  
    README.md  
  server/  
  src/  
    include/  
      app_layer.c  
      component.mk  
      connectivity.c  
      formatting.c  
      main.c  
      net_layer.c  
      phy_layer.c  
      sensor_reading.c  
      trans_layer.c  
      transmission.c  
    Makefile  
    README.md  
    sdkconfig
```

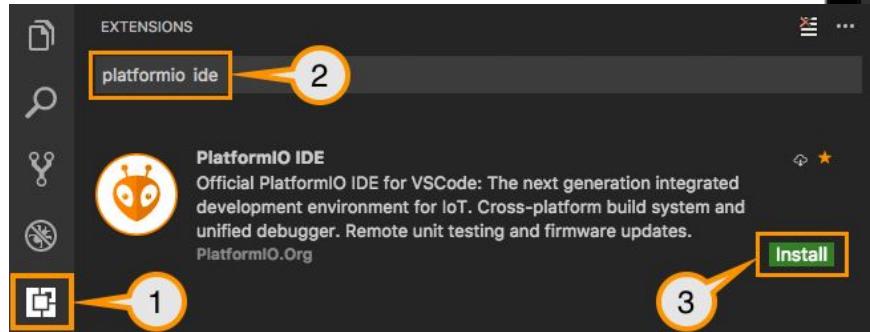
Flashing device: ESP32

- \$ make
- \$ make flash
- \$ make app
- \$ make monitor



PlatformIO

- Integration to VSCode/Atom;
- Cross-platform build system;



The screenshot shows the PlatformIO IDE interface. The main area is a code editor with tabs for "platformio.ino" and "main.cpp". The code is related to WiFi provisioning. To the right of the code editor are several panes: "MEMORY" showing memory dump, "WATCH" showing variable values, "CALL STACK" showing a paused breakpoint, "BREAKPOINTS" listing breakpoints, and "REGISTER" showing register values. At the bottom is a "TERMINAL" pane displaying the output of a "platformio device monitor" task. The terminal output includes the message "Minimtron on /dev/cu.usbmodemDF132 9600,N,8,1" and "Welcome to PlatformIO! Configuring WiFi shield/module... Starting".

Resources

- Do It Yourself (DIY) keyword;
- Teardown;
- Datasheets;
- esp32.net

Teardown: iFixIt



<https://www.ifixit.com/Teardown/Google+Home+Teardown/72684>

<https://www.ifixit.com/Teardown/Nest+Protect+Teardown/20057>

What's Inside a Google Home Mini - Teardown - Let's Find Google https://www.youtube.com/watch?v=UdFTvebhk_0