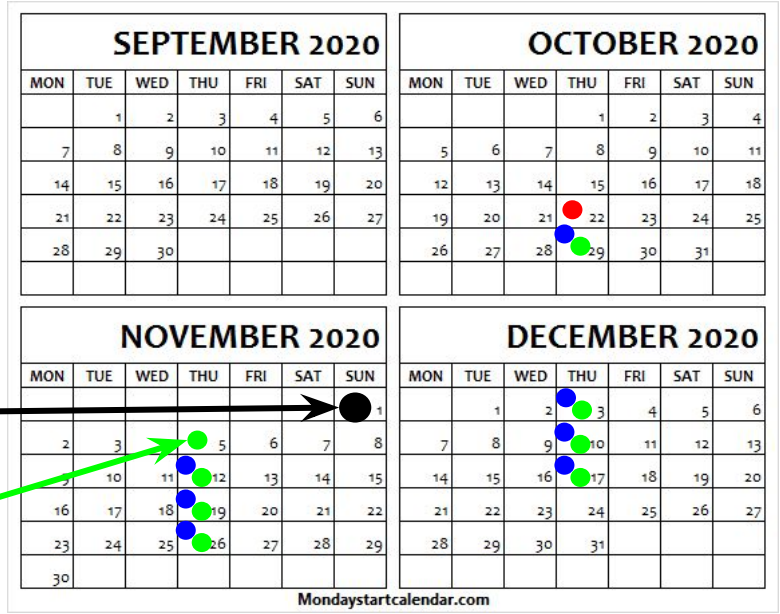
Your projects

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

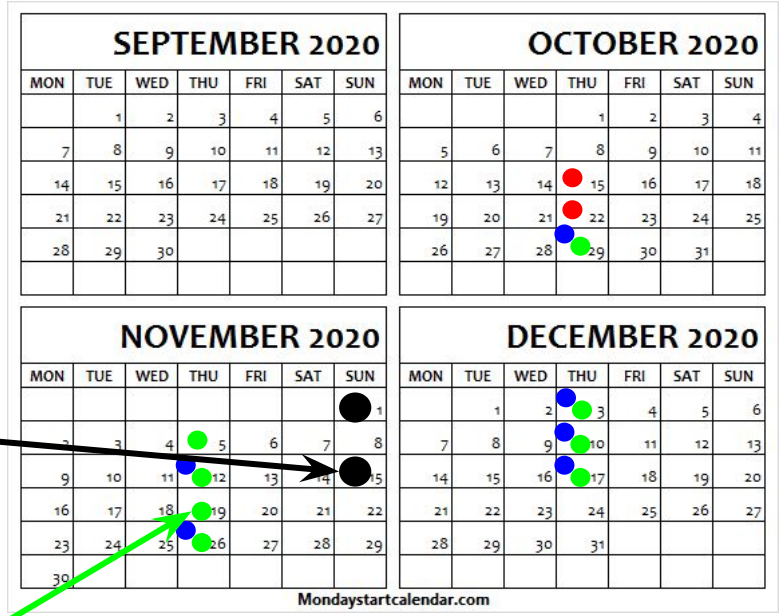


Back to organization (Cont'd)

- 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

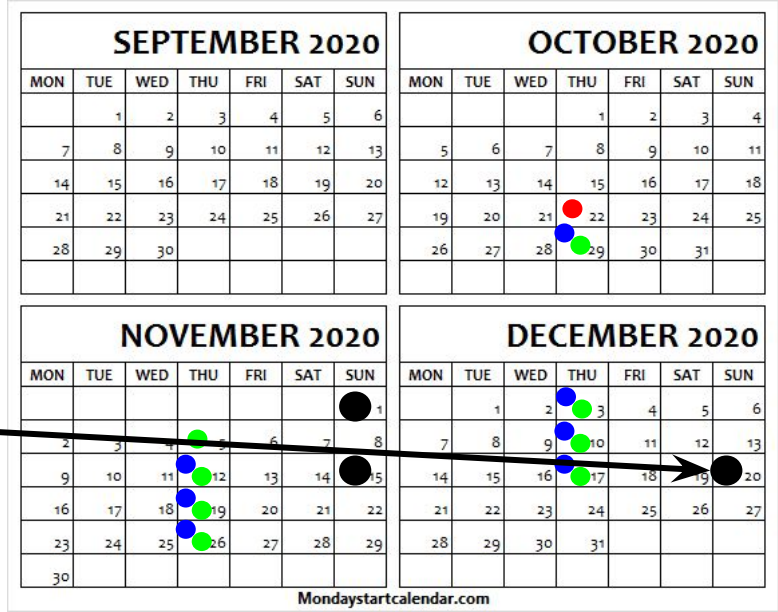


Back to organization (Cont'd)

- 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)



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)

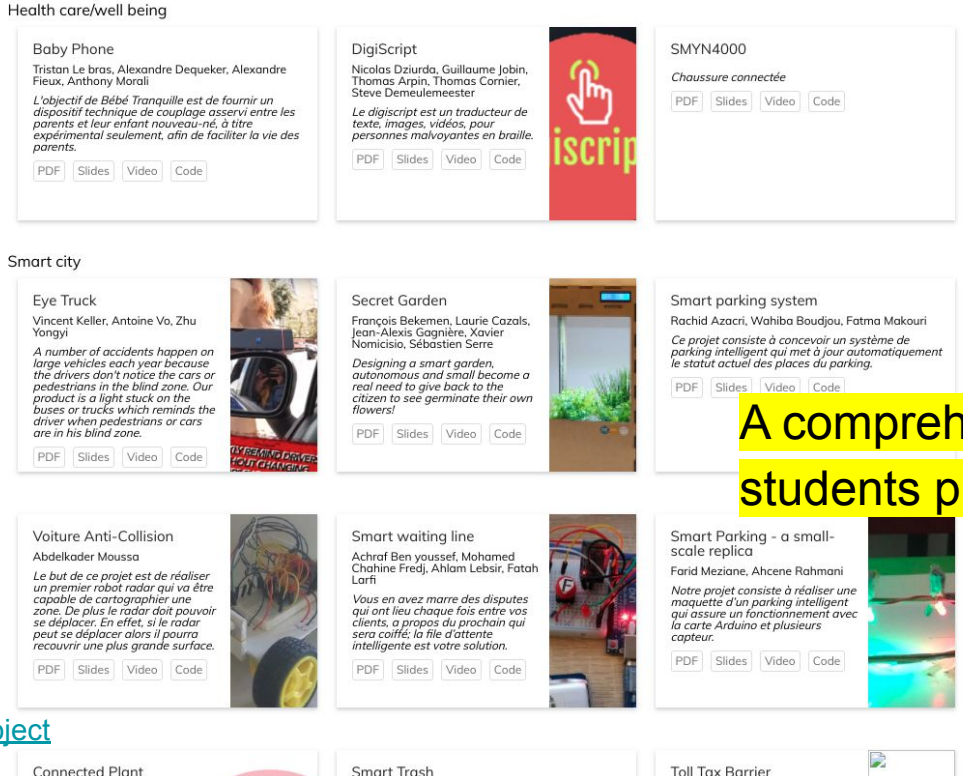
Project proposal presentation

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

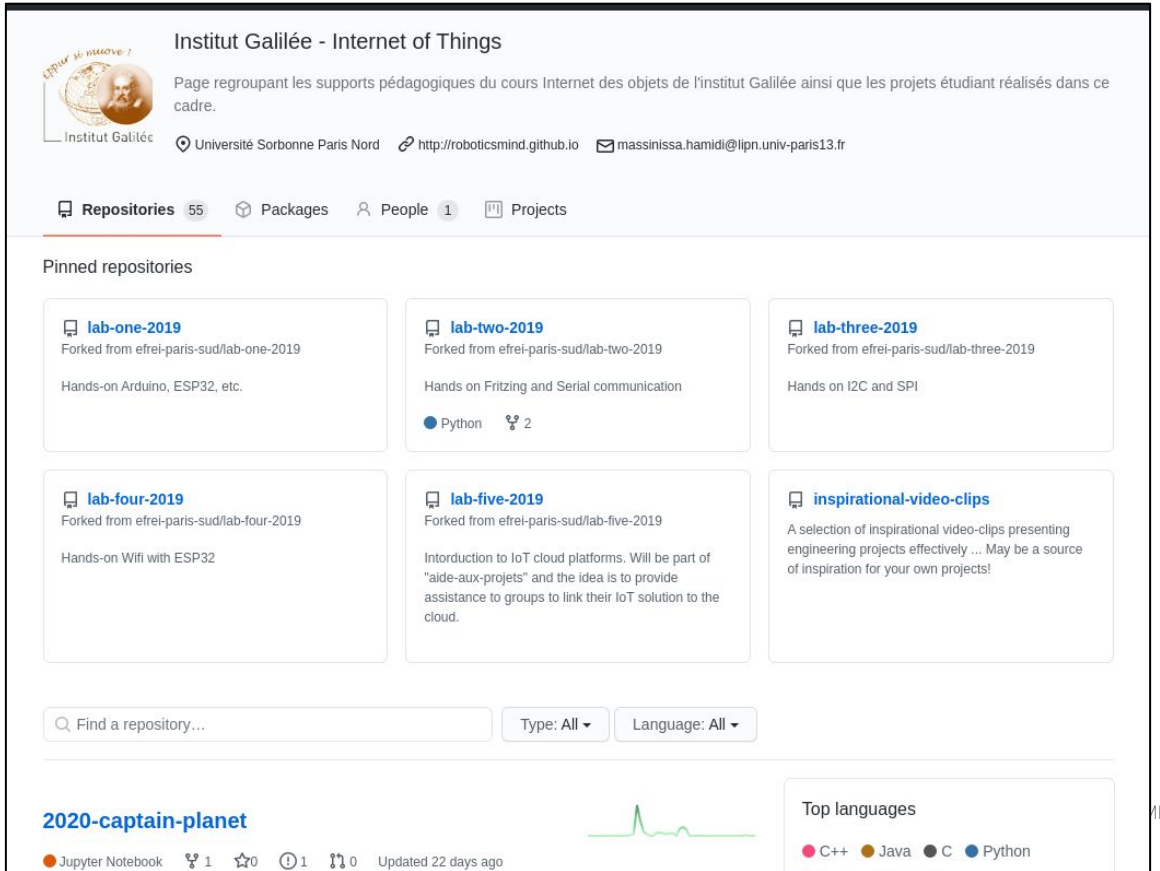
Objectifs (le site web? pourquoi faire?)



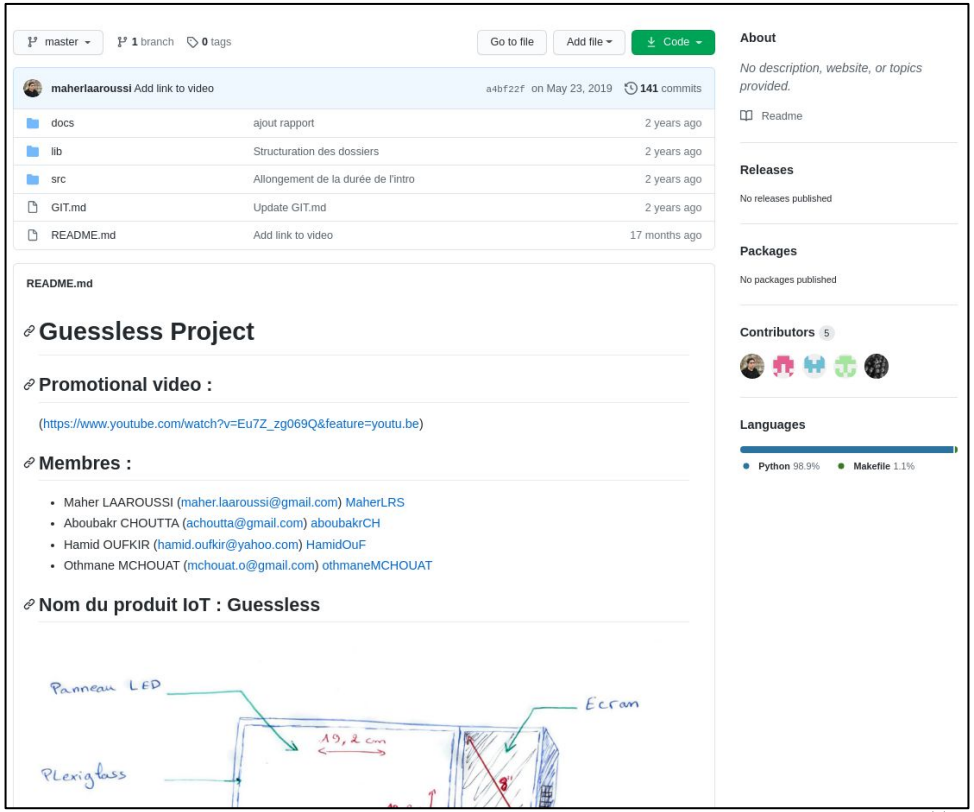
A comprehensive resource for students projects, eventually ...

https://roboticsmind.github.io/project

GitHub Organization

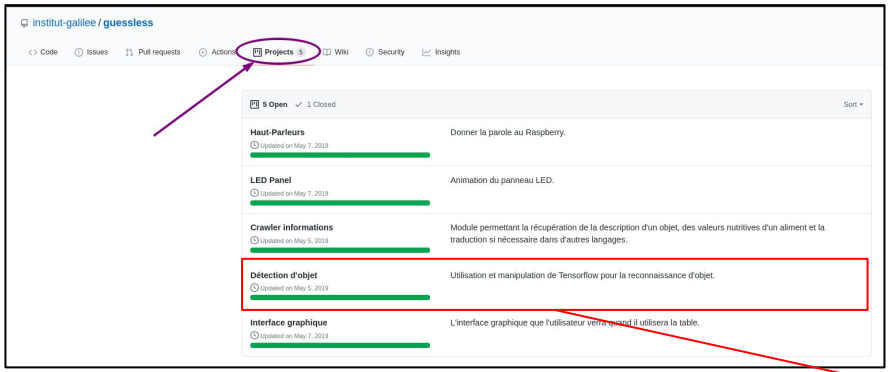


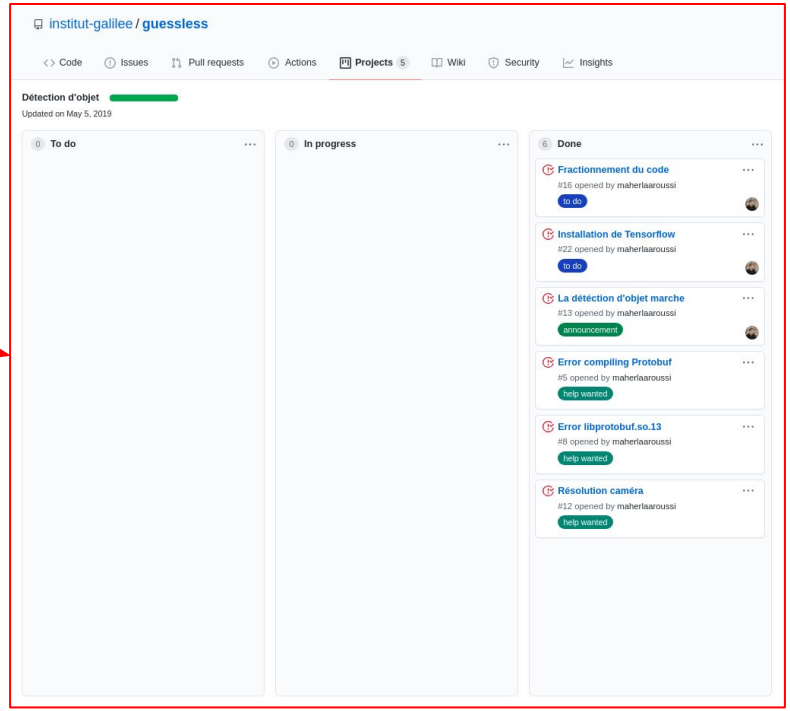
GitHub Repositories



https://github.com/institut-galilee/guessless

Project management inside GitHub





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

https://github.com/institut-galilee/guessless

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!);

https://github.com

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!

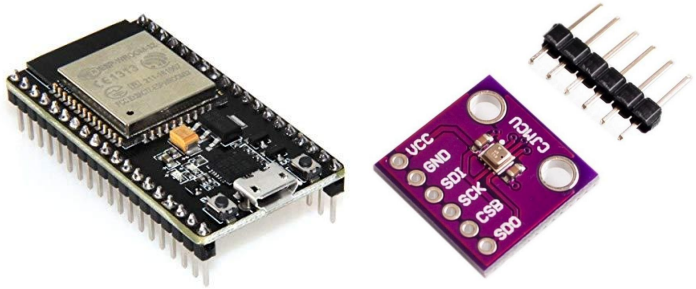
Hardware & Software

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;

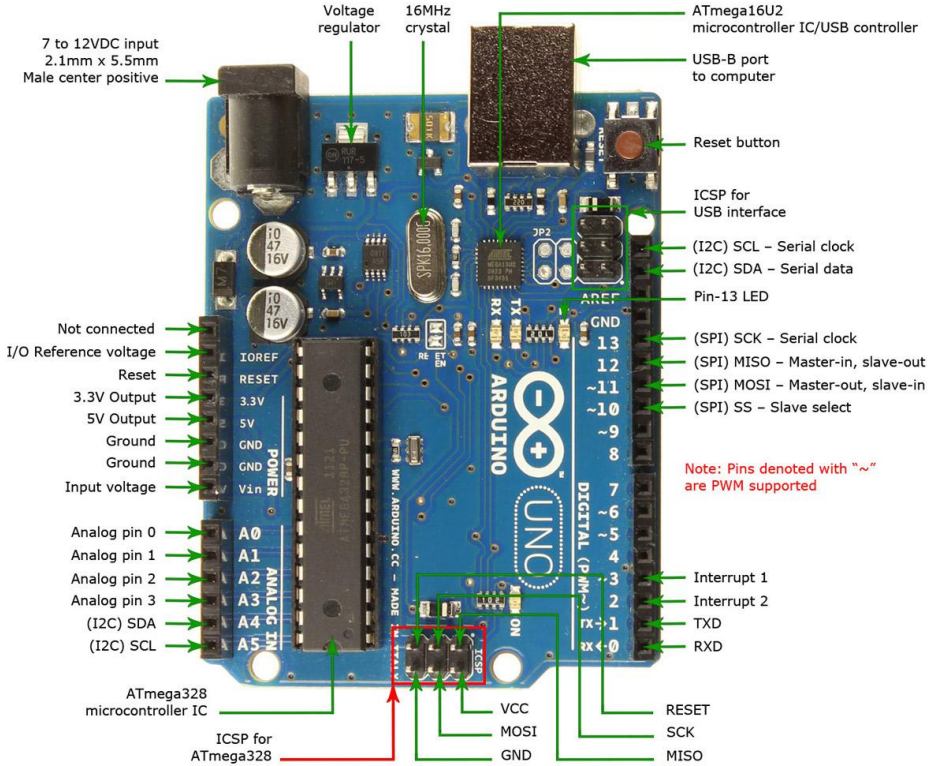
Arduino

Arduino Uno SMD R3

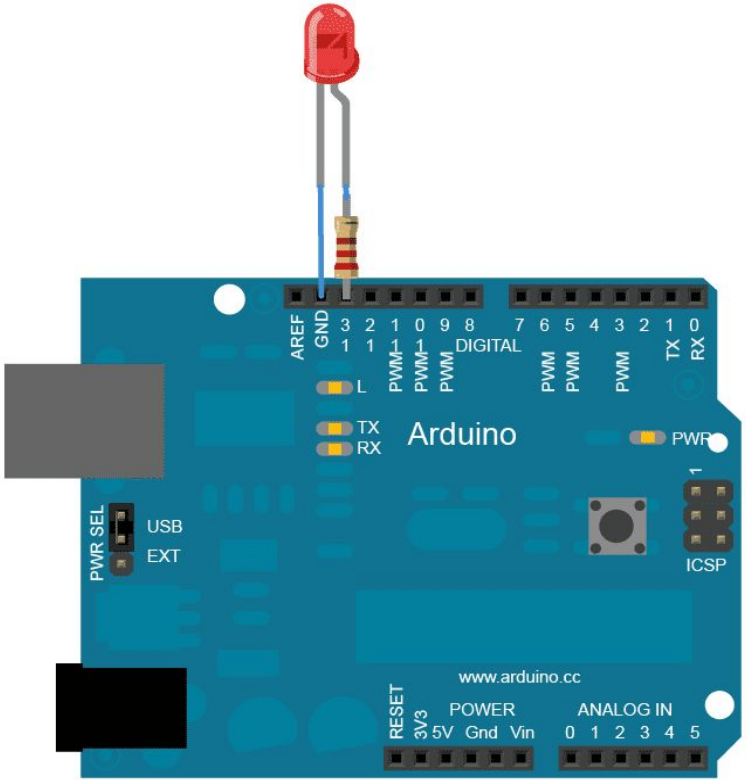
Developer	arduino.cc
Manufacturer	Many
Type	Single-board microcontroller
Operating system	None
CPU	Atmel AVR (8-bit), ARM Cortex-M0+ (32-bit), ARM Cortex-M3 (32-bit), Intel Quark (x86) (32-bit)
Memory	SRAM
Storage	Flash, EEPROM
Website	www.arduino.cc

https://en.wikipedia.org/wiki/Arduino

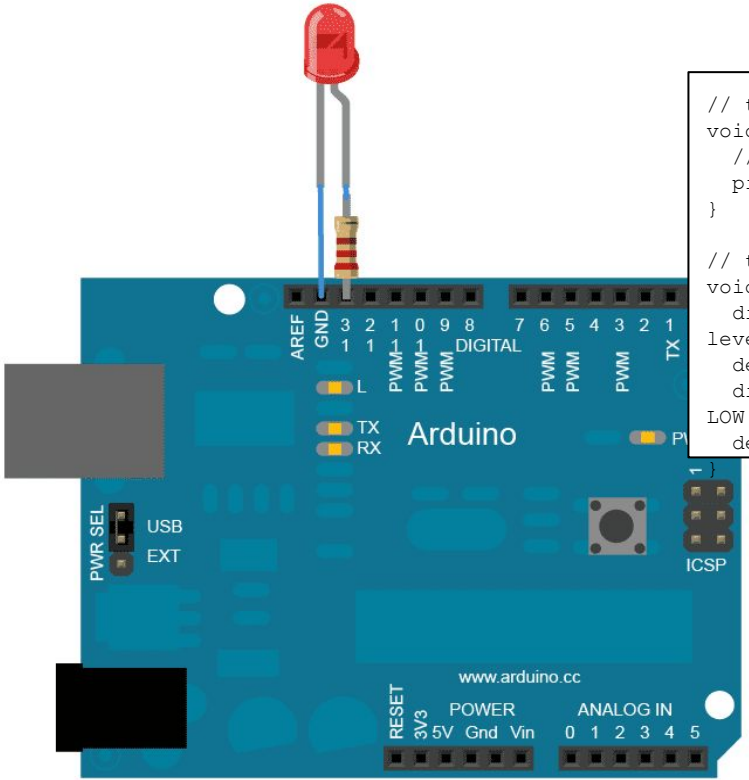
Arduino UNO (pinout)



Arduino UNO (blink LED)



Arduino UNO (blink LED)



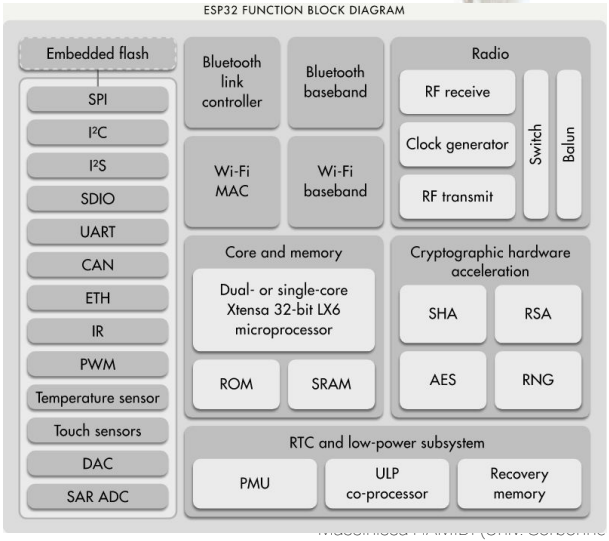

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// 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
  delay(1000);                       // 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
- **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 Kibit
      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-D0WDQ6, ESP32-D0WD, 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.

    ESP32 chips with embedded flash do not support the address mapping between external flash and peripherals.
- **Peripheral input/output:** Rich peripheral interface with DMA that includes capacitive touch, ADCs (analog-to-digital converter), DACs (digital-to-analog converter), I²C (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!



Source http://esp32.net/

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 (B
- **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 d
    - **RTC slow SRAM:** 8 KiB
      For co-processor accessing during deep-sleep mode.
    - **eFuse:** 1 Kibit
      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-mapp
      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 mapp
- **Peripheral input/output:** Rich peripheral interface with DMA
  analog converter), I²C (Inter-Integrated Circuit), UART (unive
  Peripheral Interface), I²S (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

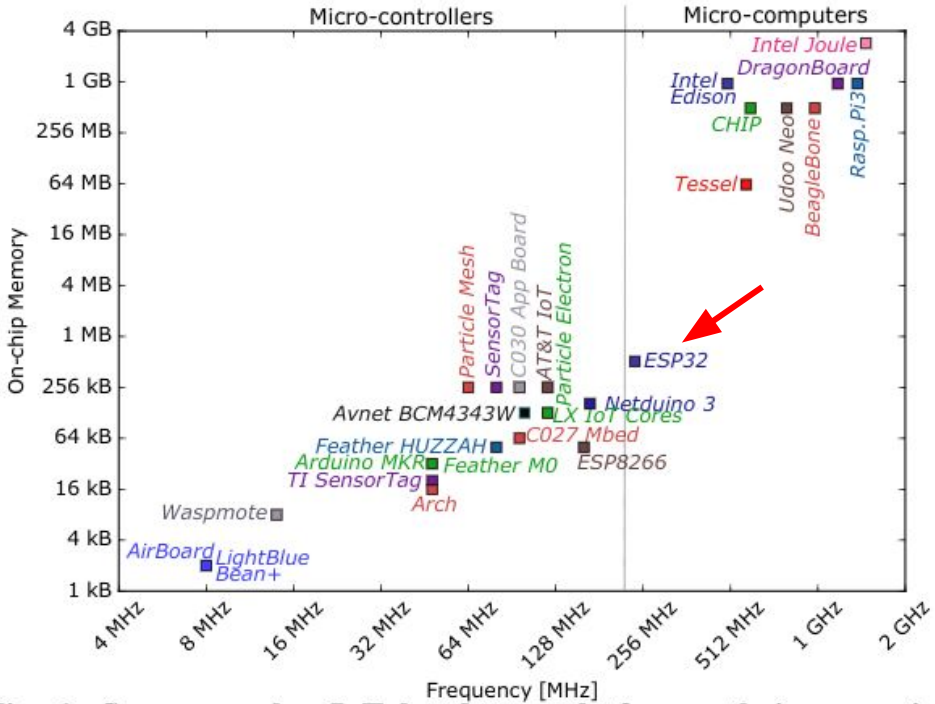
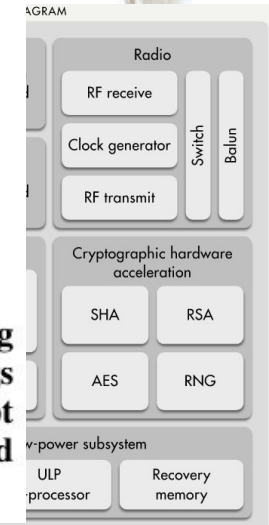
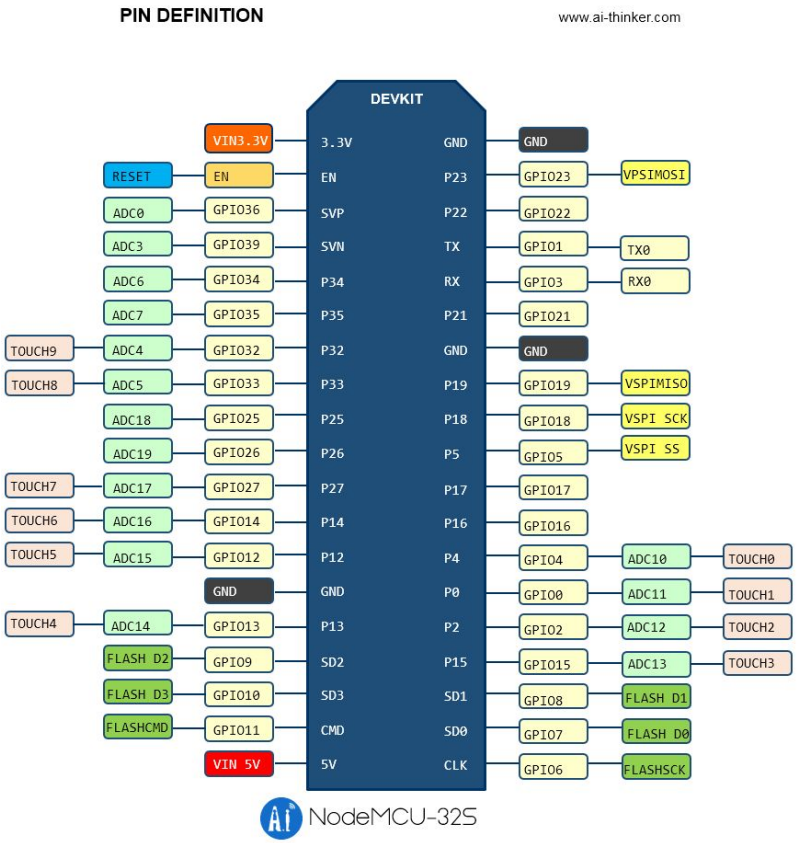


Fig. 1: Some popular IoT hardware platforms, their operating frequencies, and on-chip internal memories (each square belongs to the platform annotated with the same color next to it). Except the micro-computers –which are suitable for gateways–, the end devices have limited memory and processing capabilities.

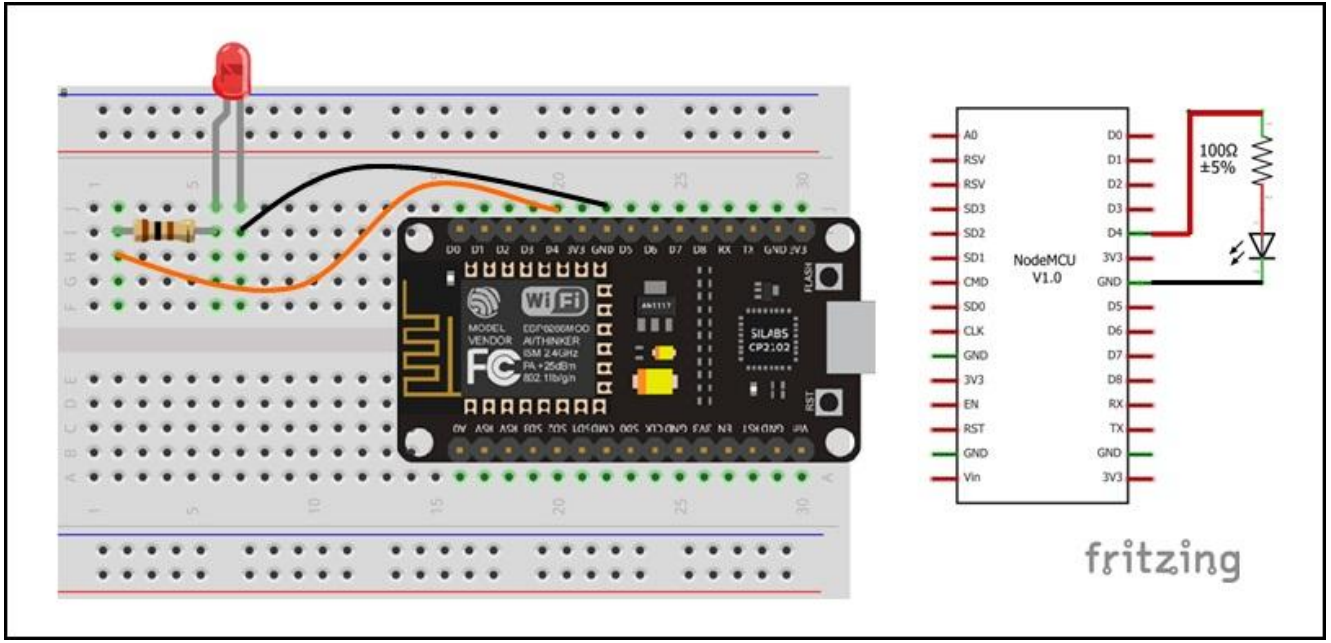


Samie F, Bauer L, Henkel J. - In IEEE Internet of Things Journal. 2020.

ESP32 (pinout)



ESP32 (Blink LED)



ZigBee Modules

Module sans fil CC2530 pour Module sans fil ZIGBEE

TEXAS INSTRUMENTS

CC2530F32, CC2530F64
CC2530F128, CC2530F256

www.ti.com

SWRS081B – APRIL 2009 – REVISED FEBRUARY 2011

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

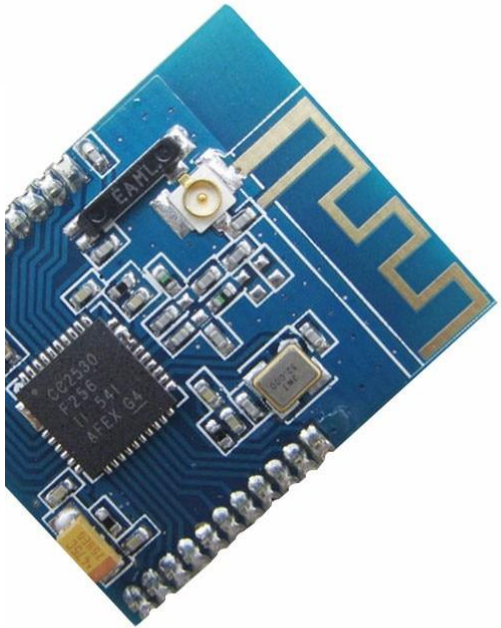
Check for Samples: CC2530F32, CC2530F64, CC2530F128, CC2530F256

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/

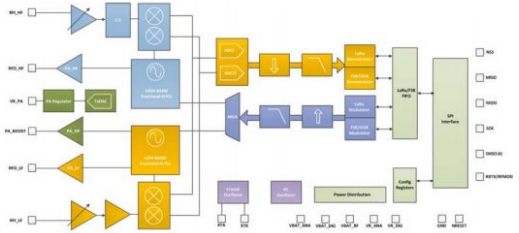
LoRa Modules

LoRa 915MHz SX1276 rf émetteur-récepteur

SEMTECH LoRa 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
SX1276IMLTRT	T&R	3000 pieces
SX1277IMLTRT	T&R	3000 pieces
SX1278IMLTRT	T&R	3000 pieces
SX1279IMLTRT	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

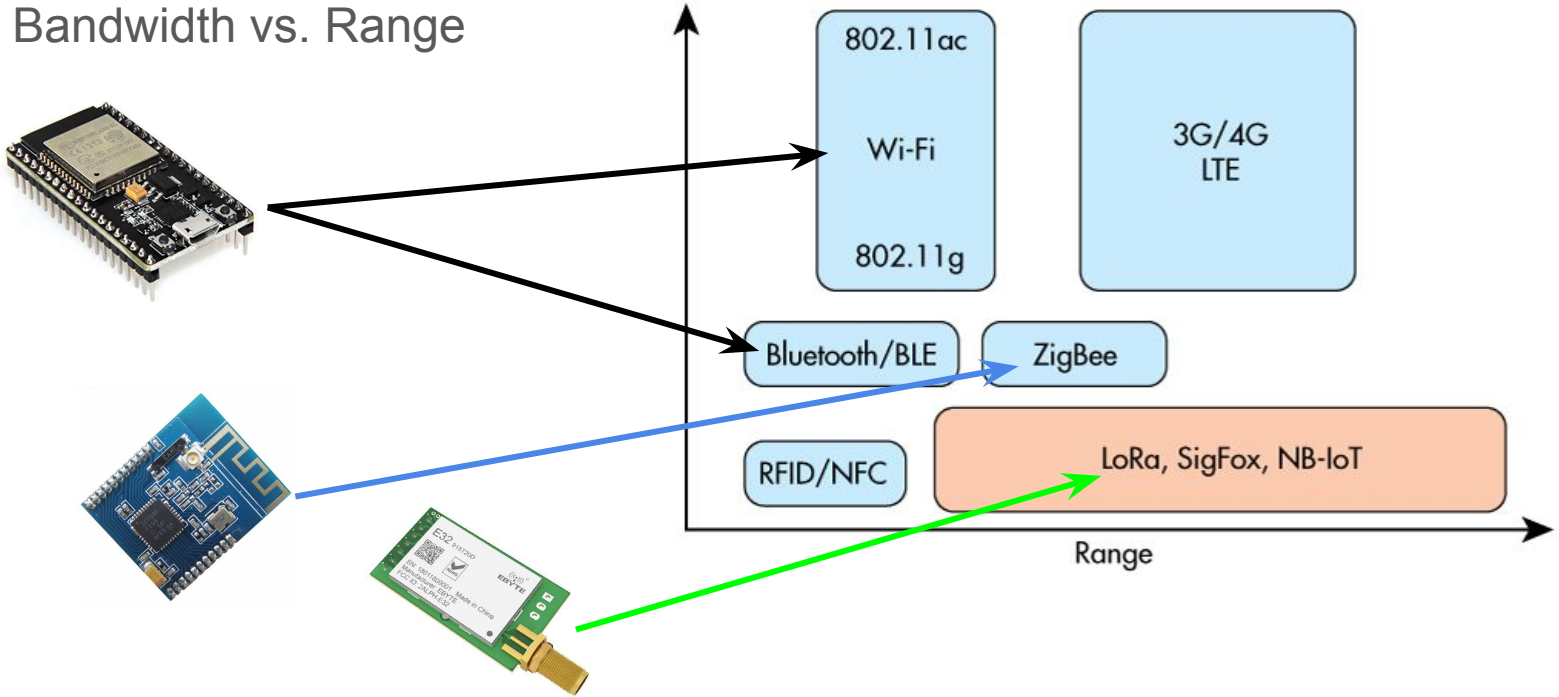
https://www.semtech.com/uploads/documents/DS_SX1276-7-8-9_W_APP_V6.pdf

Some projects tagged ‘lorawan’ https://hackaday.io/projects?tag=lorawan

https://www.thethingsnetwork.org/

From a connectivity perspective

Bandwidth vs. Range



https://www.electronicdesign.com/industrial-automation/choices-abound-long-range-wireless-iot

From a connectivity perspective (Cont'd)

Network types

Figure from the lectures of Aomar Osmani on Internet of things

Connectivity: examples

From a TCP/IP-model perspective

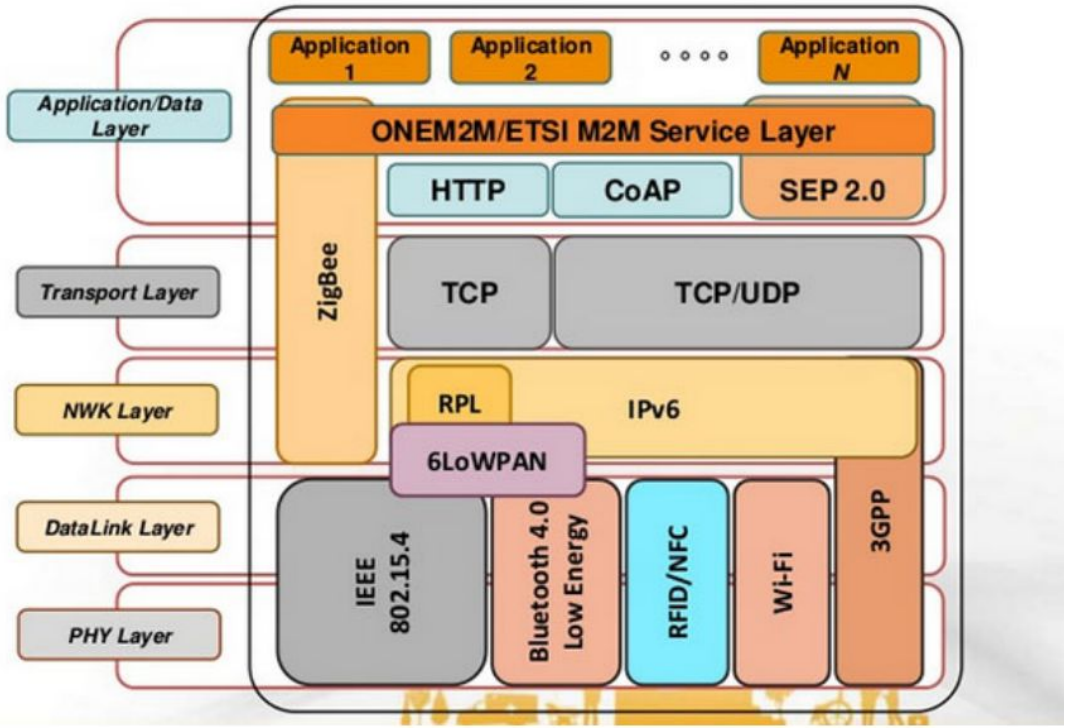


Figure from the lectures of Aomar Osmani on Internet of things

Connectivity: examples (cont'd)

Industrial context

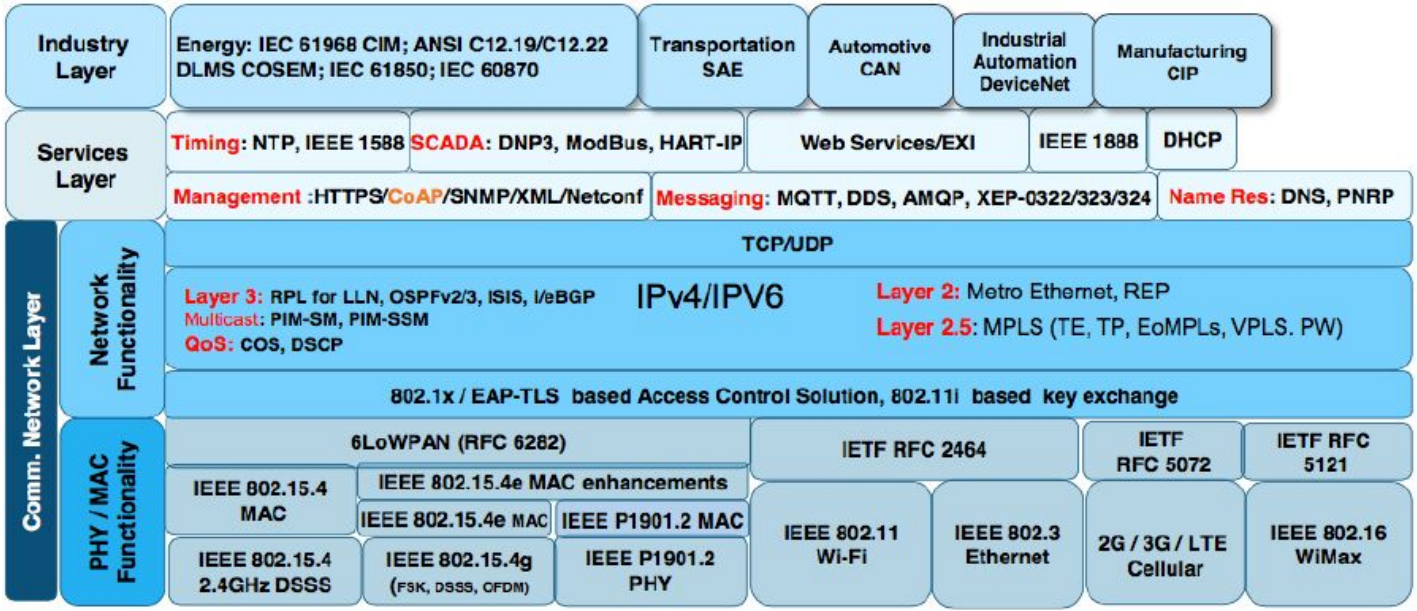
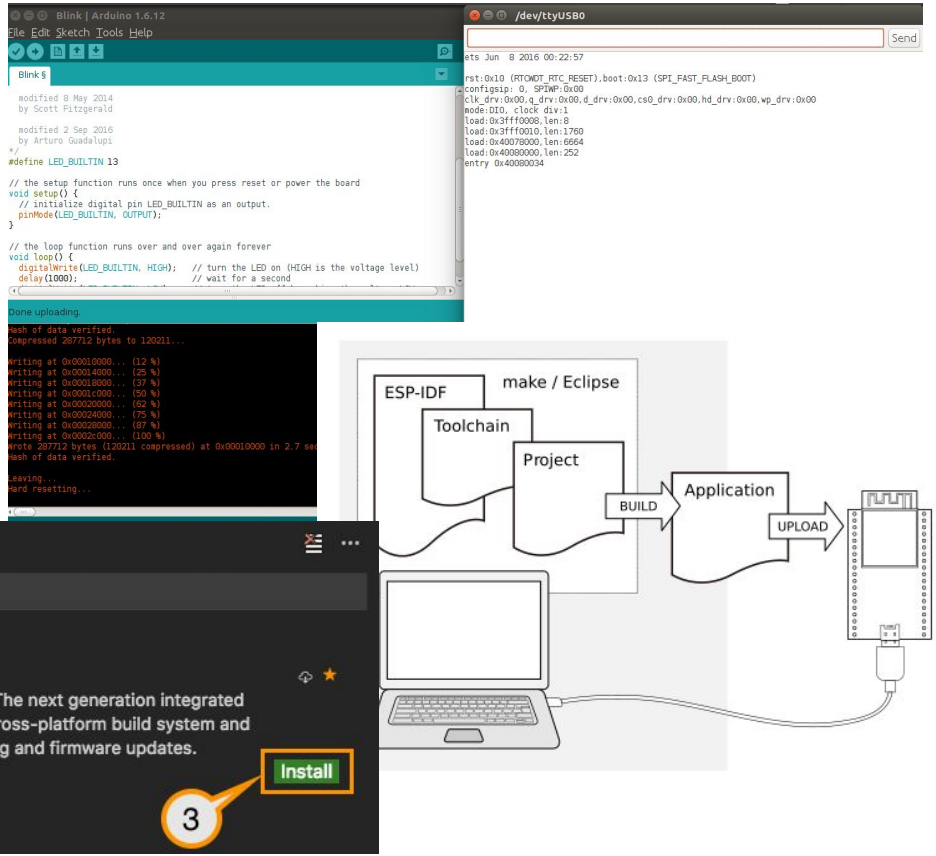


Figure from the lectures of Aomar Osmani on Internet of things

Software

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

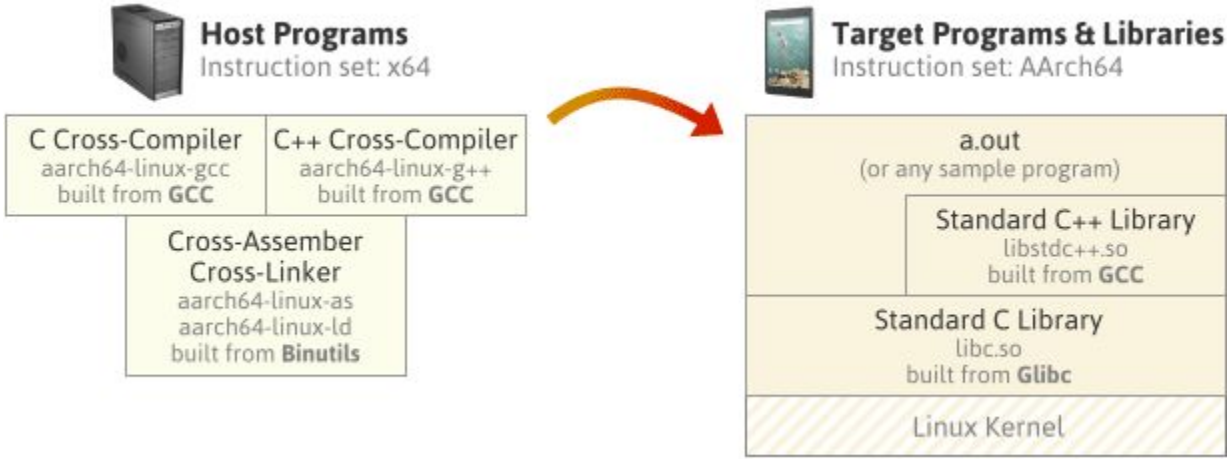


Compilation croisée

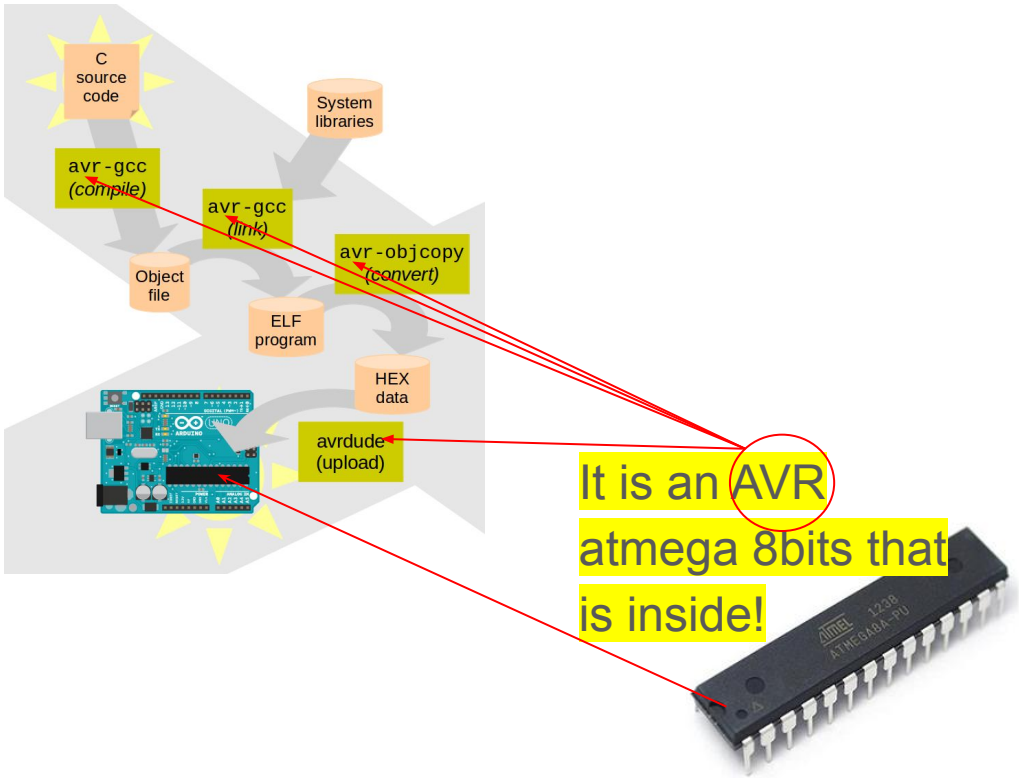
La compilation croisée permet de créer des exécutables depuis une certaine architecture pour une autre. Cela permet de créer des paquet pour les systèmes pour lesquels on n'a pas la main.
En fonction de l'architecture cible (ARM, Atmel AVR, x86, Xtensa, etc.), on doit se doter d'un environnement de compilation 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 :

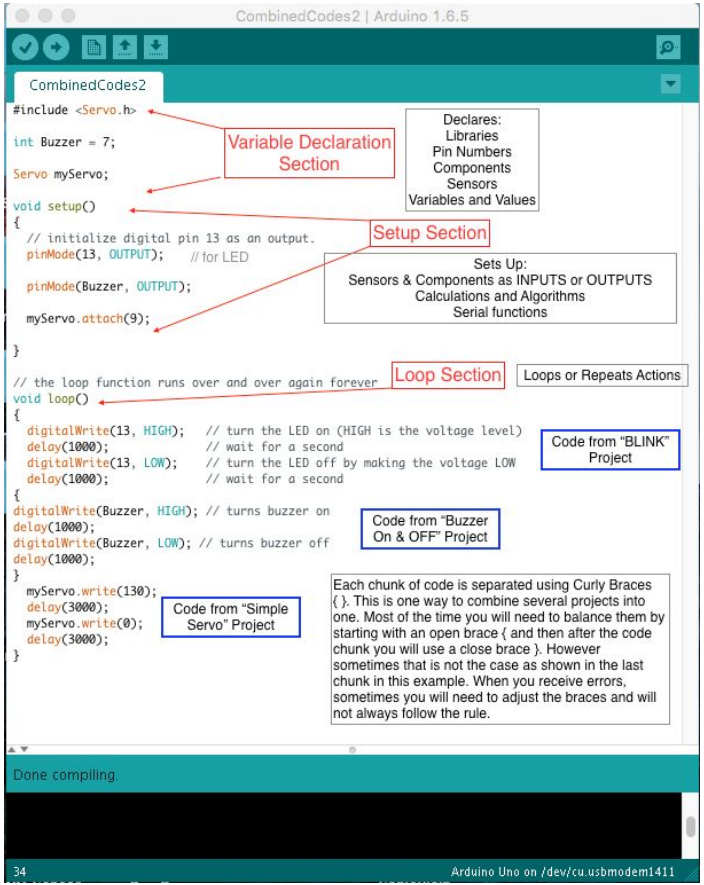
- GNU Compiler Collection (GNU) ;
- binutils ;
- debugger ;
- C library (glibc, ulibc, etc.) ;

Cross-Compiling environment



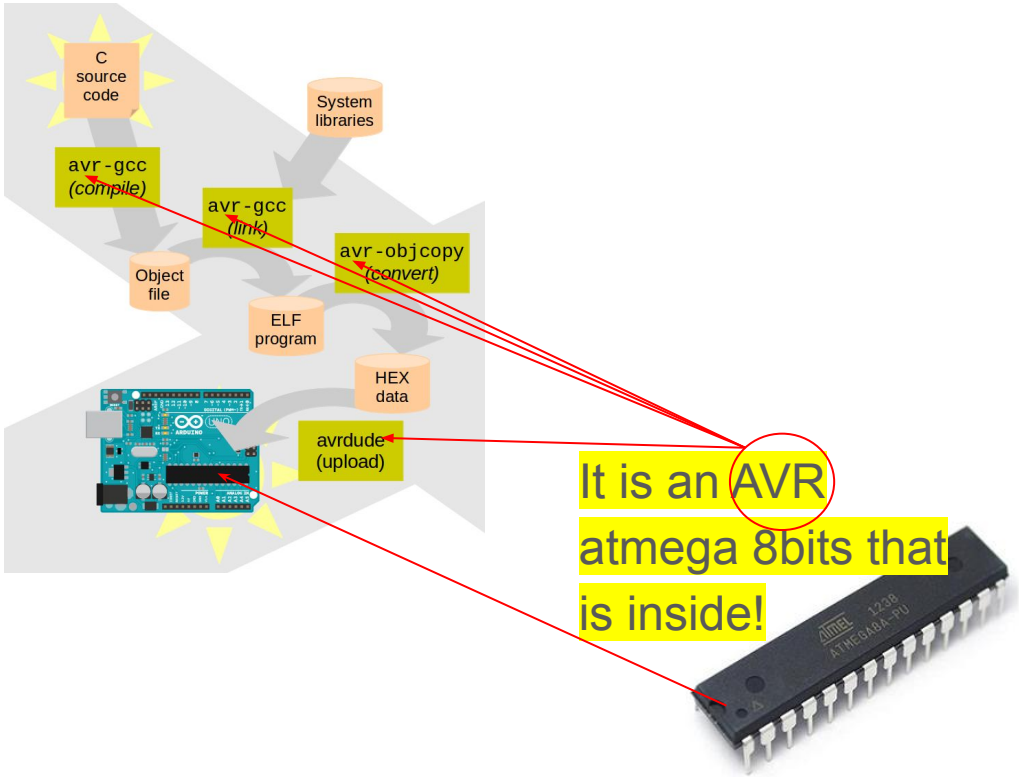
The case of Arduino UNO

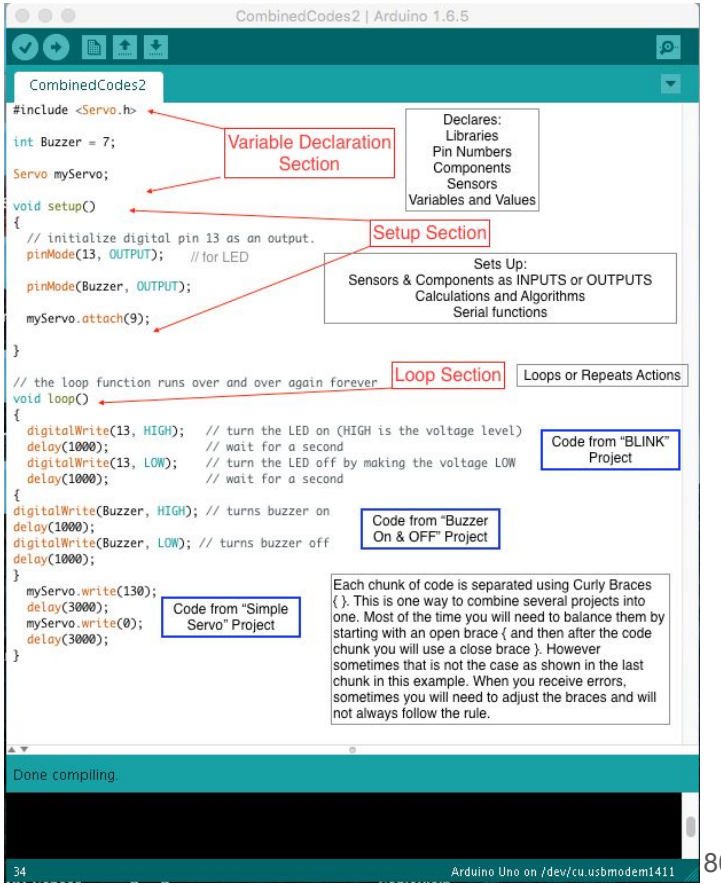




https://www.microchip.com/wwwproducts/en/ATmega328

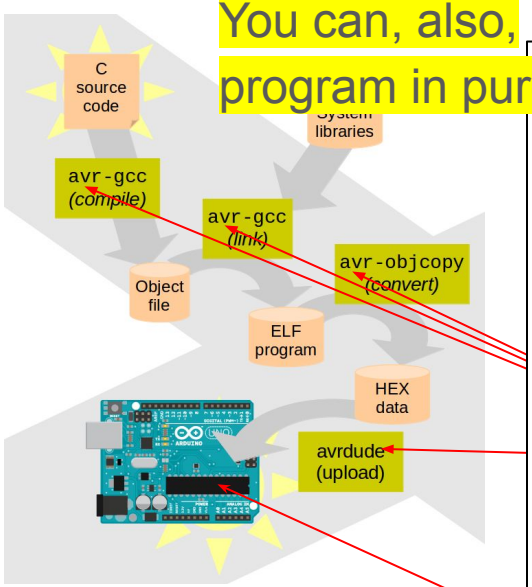
The case of Arduino UNO



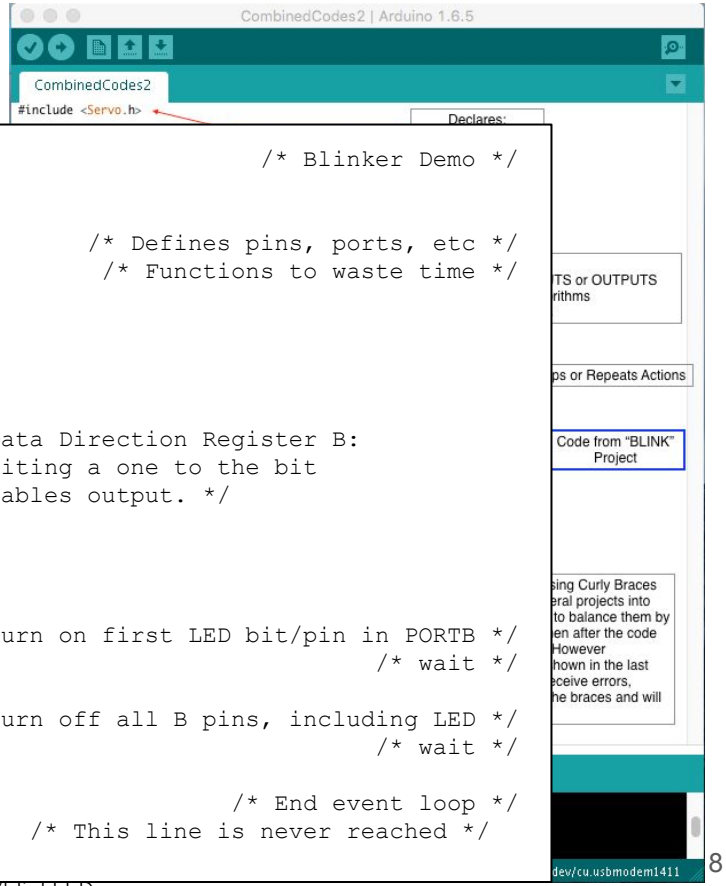


The case of Arduino UNO

You can, also, program in pure C



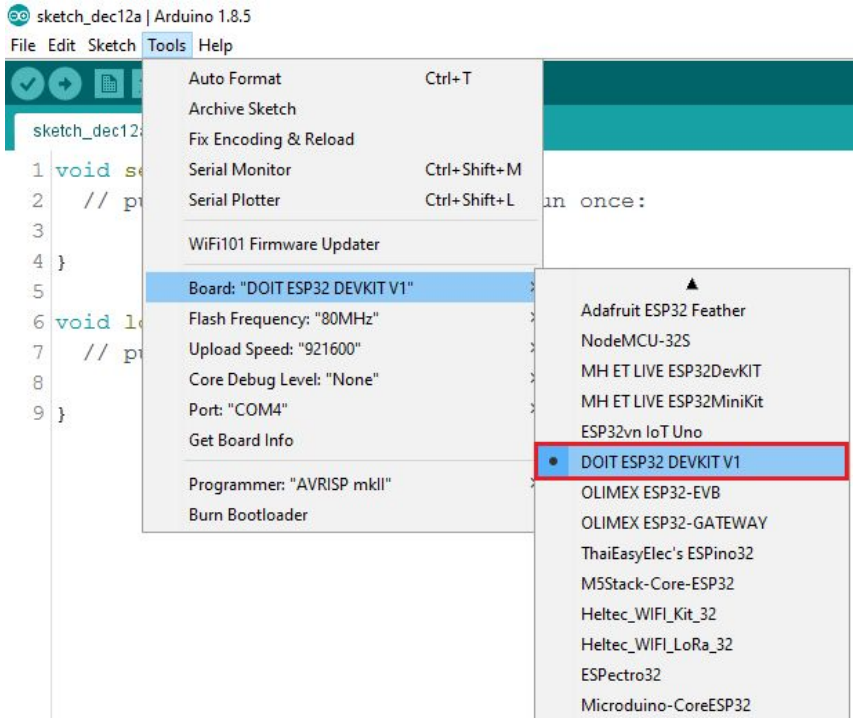

```
                                                        /* Blinker Demo */

// ------- Preamble -------- //
#include <avr/io.h>                         /* Defines pins, ports, etc */
#include <util/delay.h>                      /* Functions to waste time */


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 */

  }                                                  /* End event loop */
  return 0;                                 /* This line is never reached */
}
```



https://github.com/hexagon5un/AVR-Programming/blob/master/Chapter02_Programming-AVRs/blinkLED/blinkLED.c

Arduino on ESP32

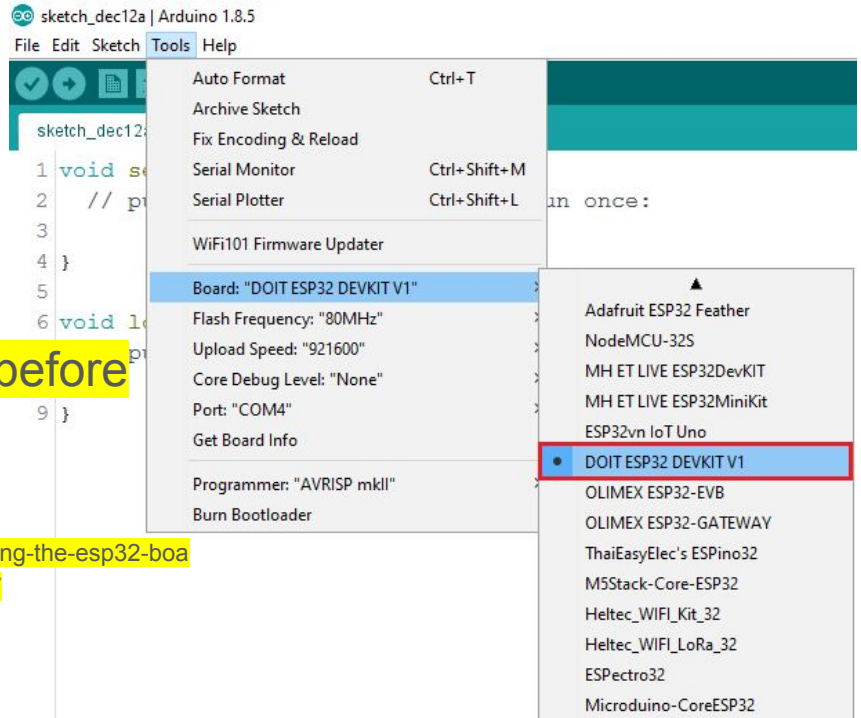
Espressif developed arduino-esp32, an abstraction layer to write Arduino code for the ESP32



https://github.com/espressif/arduino-esp32

Arduino on ESP32

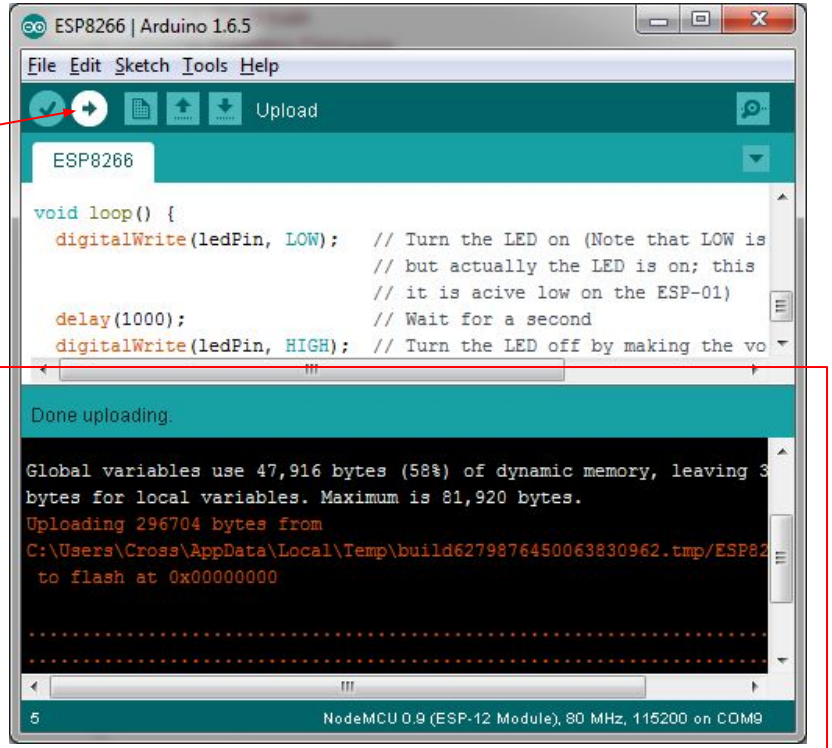
Espressif developed `arduino-esp32`, an abstraction layer to write Arduino code for the ESP32



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/

https://github.com/espressif/arduino-esp32

Flashing device: Arduino IDE



Anatomy of an ESP32 program

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

```
</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
```

```
#
# Security features
#
CONFIG_SECURE_BOOT_ENABLED=
CONFIG_FLASH_ENCRYPTION_ENABLED=

#
# Serial flasher config
#
CONFIG_ESPTOOLPY_PORT="/dev/ttyUSB0"
CONFIG_ESPTOOLPY_BAUD_115200B=y
CONFIG_ESPTOOLPY_BAUD_230400B=
CONFIG_ESPTOOLPY_BAUD_921600B=
CONFIG_ESPTOOLPY_BAUD_2MB=
CONFIG_ESPTOOLPY_BAUD_OTHER=
CONFIG_ESPTOOLPY_BAUD_OTHER_VAL=115200
CONFIG_ESPTOOLPY_BAUD=115200
CONFIG_ESPTOOLPY_COMPRESSED=y
```

```c
/* Blink Example

   This example code is in the Public Domain (or CC0 licensed, at your option.)

   Unless required by applicable law or agreed to in writing, this
   software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
   CONDITIONS OF ANY KIND, either express or implied.
*/
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "sdkconfig.h"

/* Can run 'make menuconfig' to choose the GPIO to blink,
   or you can edit the following line and set a number here.
*/
#define BLINK_GPIO CONFIG_BLINK_GPIO

void blink_task(void *pvParameter)
{
    /* Configure the IOMUX register for pad BLINK_GPIO (some pads are
       muxed to GPIO on reset already, but some default to other
       functions and need to be switched to GPIO. Consult the
       Technical Reference for a list of pads and their default
       functions.)
    */
    gpio_pad_select_gpio(BLINK_GPIO);
    /* Set the GPIO as a push/pull output */
    gpio_set_direction(BLINK_GPIO, GPIO_MODE_OUTPUT);
    while(1) {
        /* Blink off (output low) */
        gpio_set_level(BLINK_GPIO, 0);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
        /* Blink on (output high) */
        gpio_set_level(BLINK_GPIO, 1);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
}

void app_main()
{
    xTaskCreate(&blink_task, "blink_task", configMINIMAL_STACK_SIZE, NULL, 5, NULL);
}
```

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.

Slide from the lectures of Aomar Osmani on Internet of things
esp-idf components https://github.com/efrei-paris-sud/esp-idf/tree/master/components

ESP32 : FreeRTOS

C'est le système d 'exploitation pour microcontroleurs 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

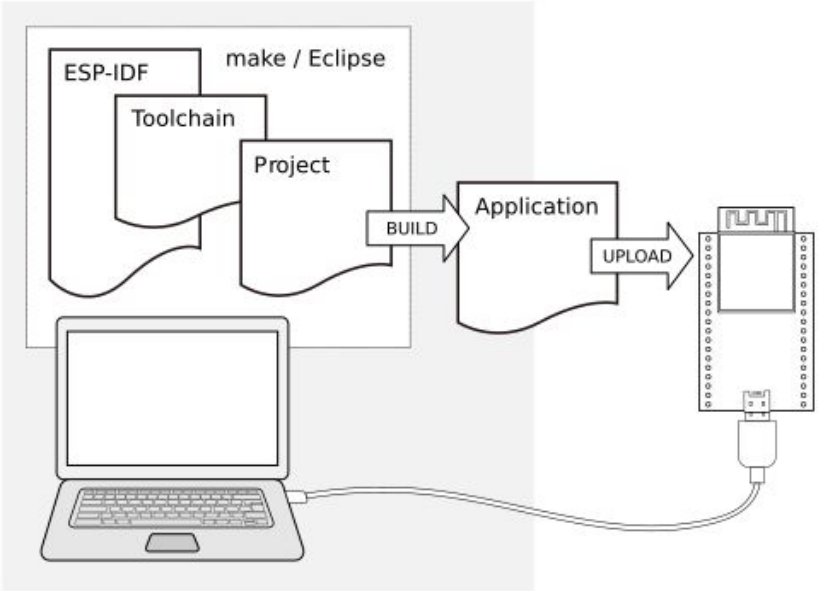
```
</efrei/lab-four/
▸ build/
▸ doc/
▾ lib/BME280_driver/
  ▸ examples/
  ▸ include/
  ▸ selftest/
    bme280.c
    component.mk
    LICENSE
    README.md
▸ server/
▾ src/
  ▸ include/
    app_layer.c
    component.mk
    connectivity.c
    formatting.c
    main.c
    net_layer.c
    phy_layer.c
  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

```
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

Flashing device: ESP32

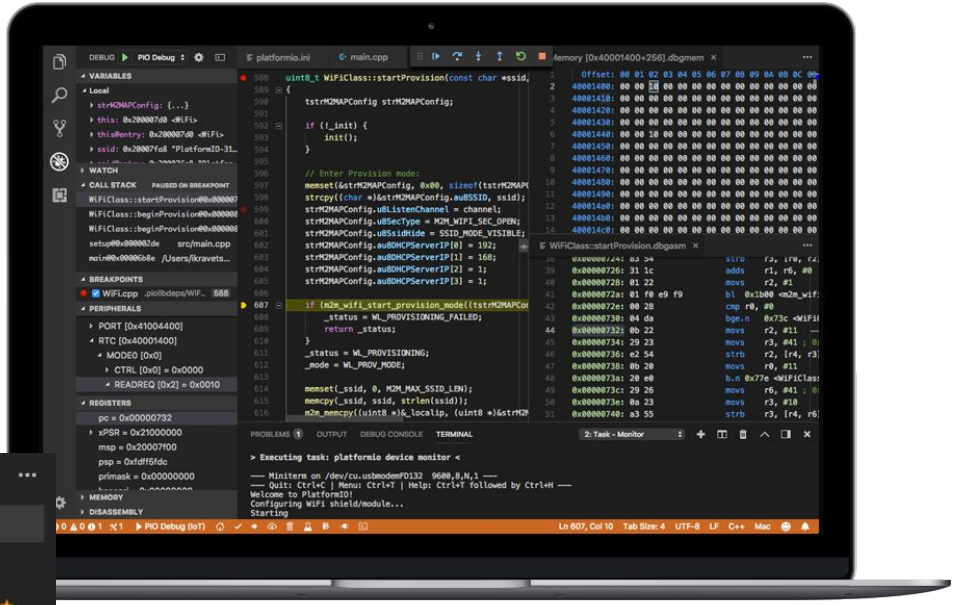
- $ make
- $ make flash
- $ make app
- $ make monitor

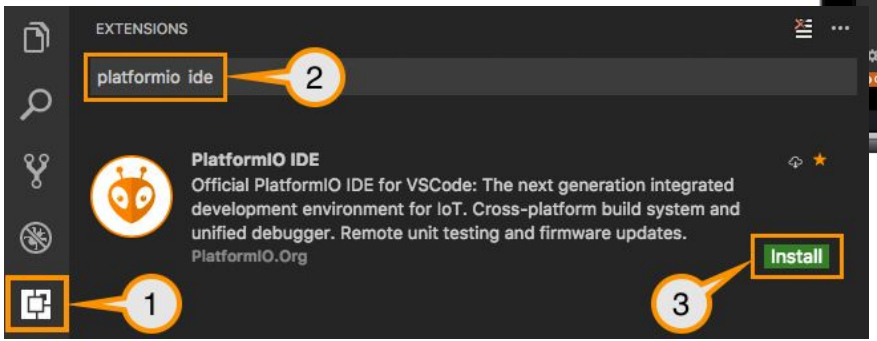


https://github.com/efrei-paris-sud/esp-idf/blob/master/README.md

PlatformIO

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





Quick start https://docs.platformio.org/en/latest/ide/vscode.html#quick-start

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