IOT Online Course

Fundamentals of IoT

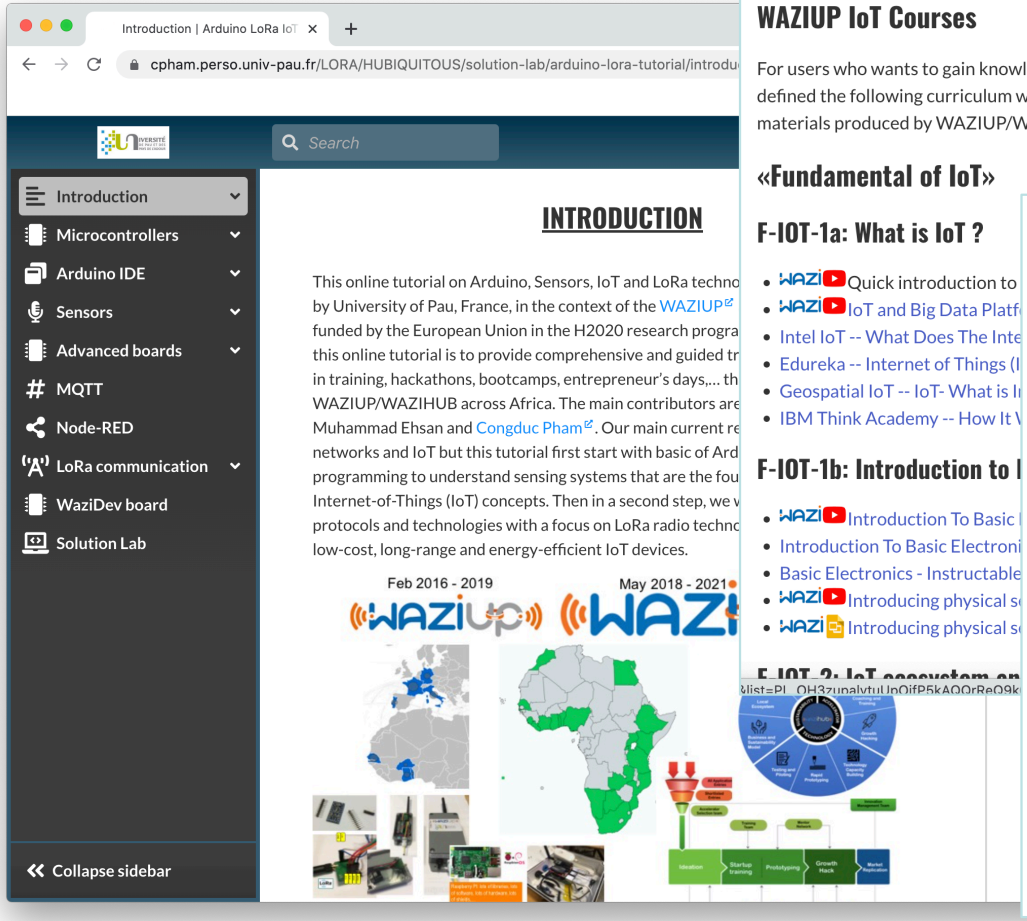
F-IOT-2c: Introduction to IoT hardware

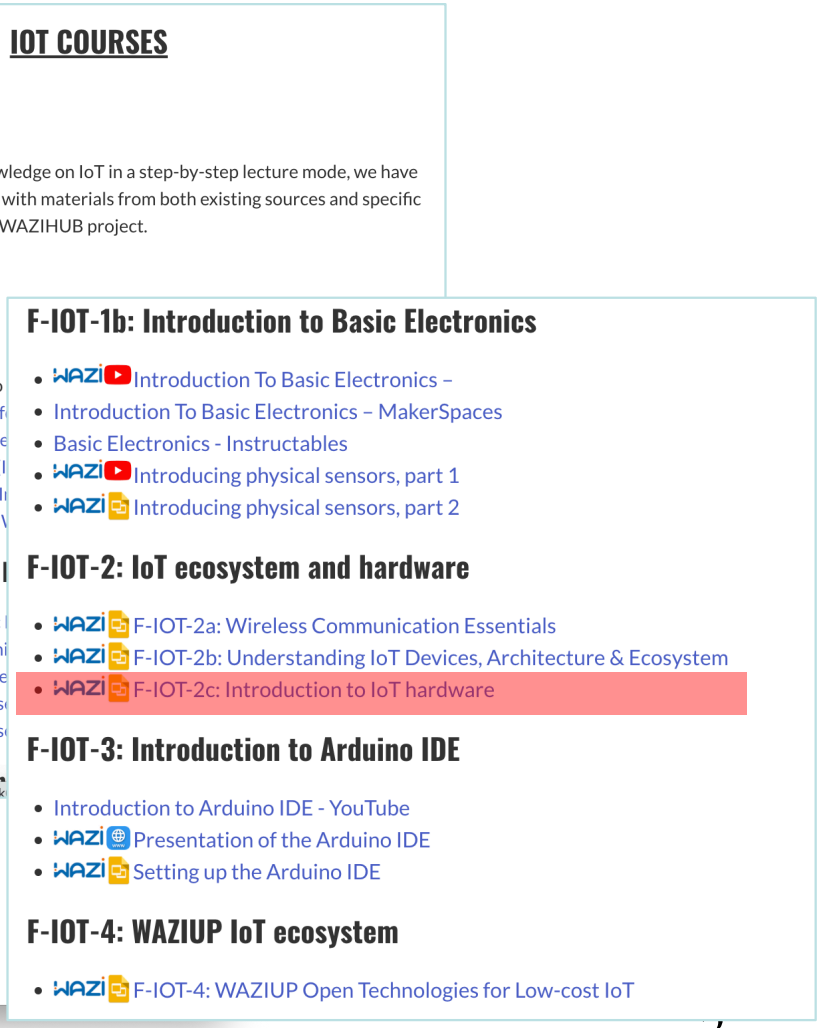
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

IoT – from idea to reality

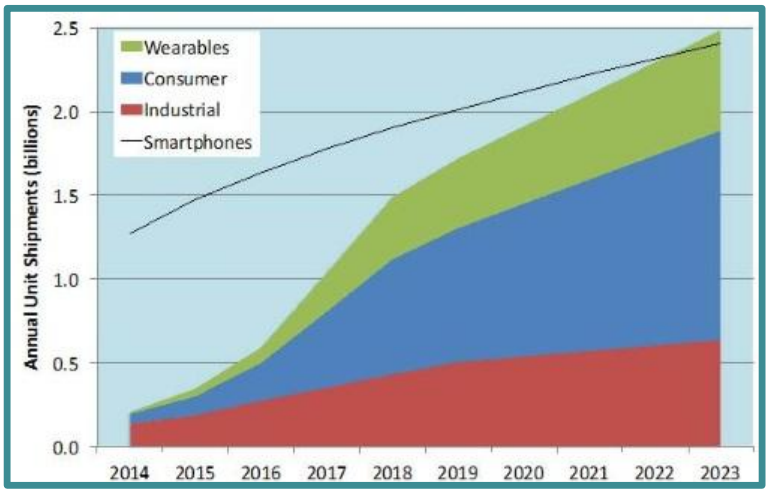
IoT Online Course

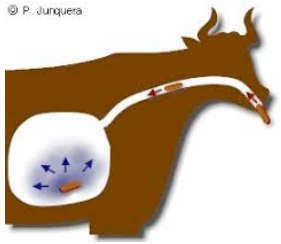
- http://diy.waziup.io

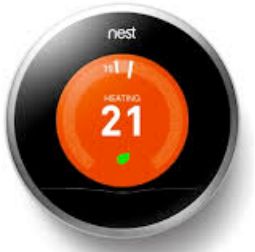




IoT devices



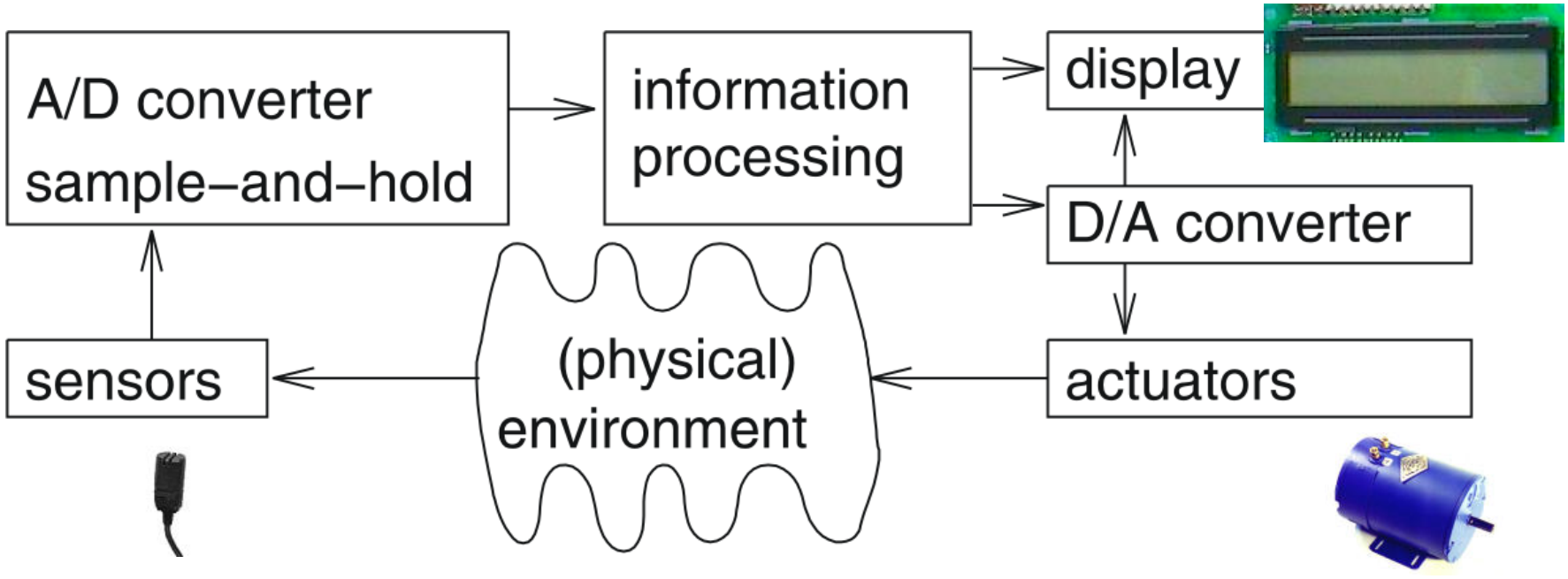




Prof. Congduc Pham
http://www.univ-pau.fr/~cpham

At the beginning…

- IoT device can be viewed as a simple Embedded System Hardware which is frequently used in a loop ("hardware in a loop"):



From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

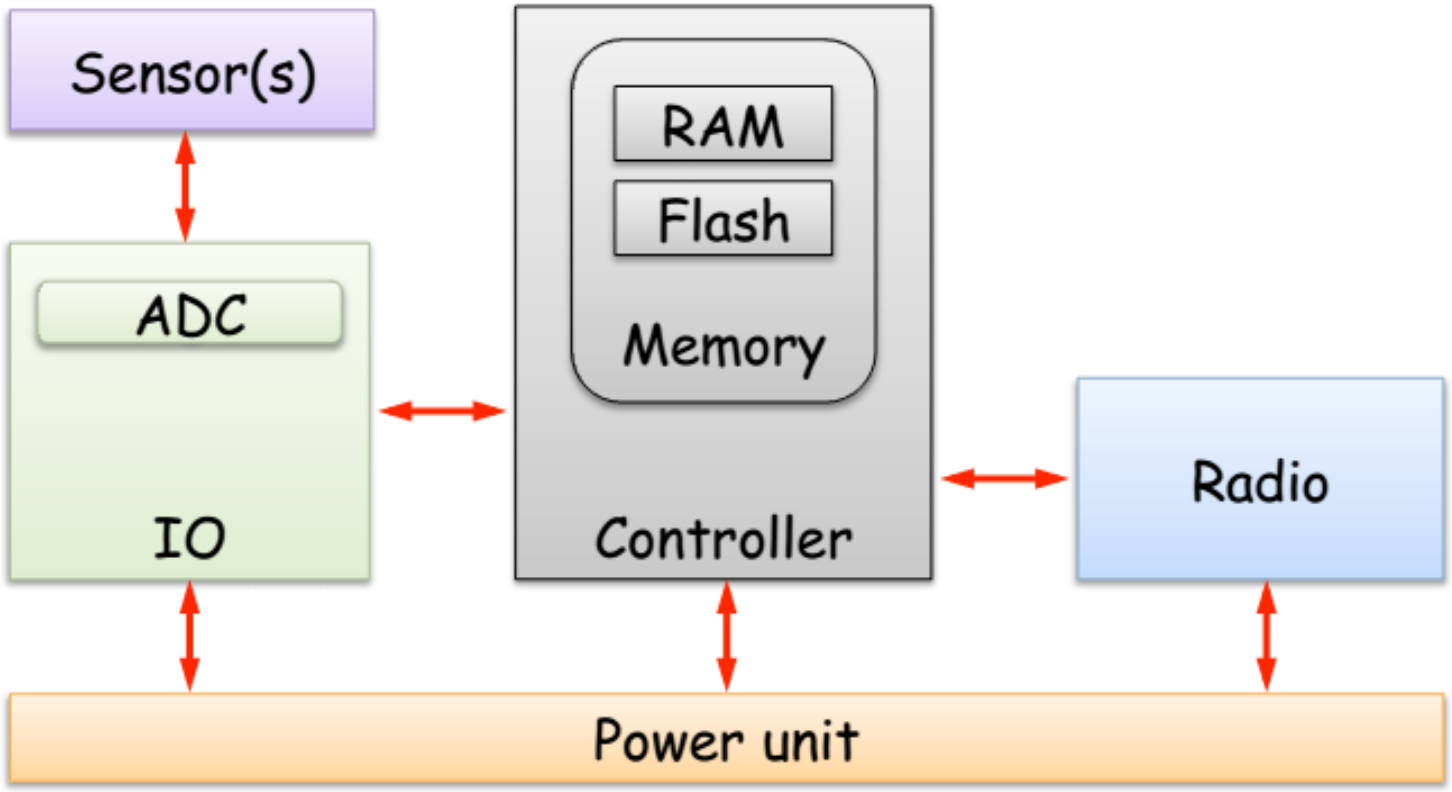
Main characteristics of ES

- *Dependability*
- Single-functioned (dedicated System)
  - Executes a single program, repeatedly
- Tightly-constrained (Efficient)
  - Low cost, low power, small, fast, etc.
- *Reactive and real-time*
  - *Continually reacts to changes in the system's environment*
  - *Must compute certain results in real-time without delay*

- Dependability and real-time are not mandatory for IoT

From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

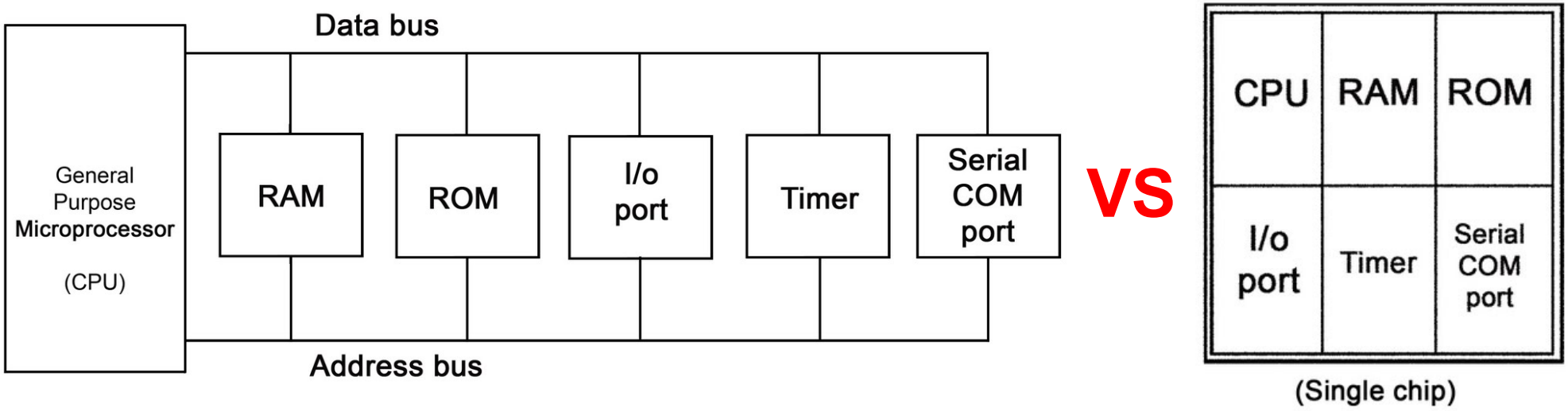
Generic Hardware Node



From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

Microprocessors & Microcontrollers

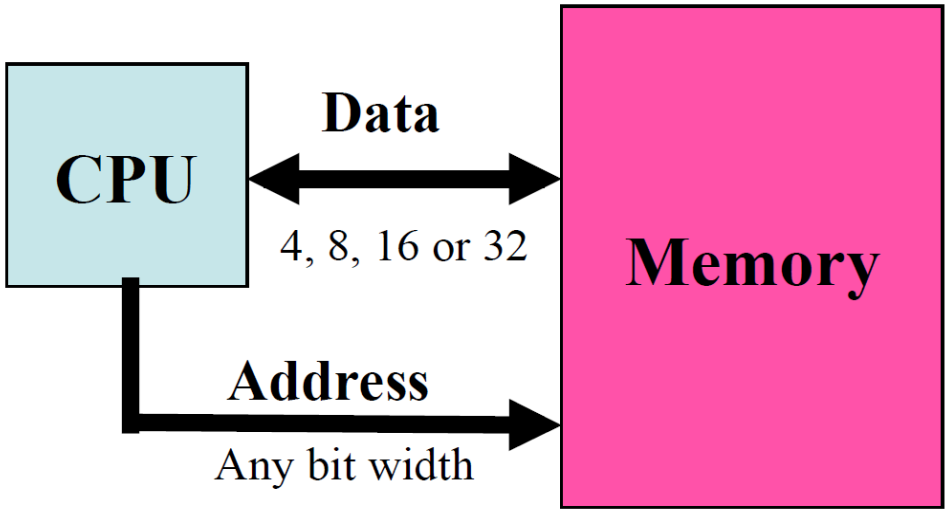
- A microprocessor unit (MPU) is a processor on one silicon chip
- A microcontroller unit (MCU) is a microprocessor with some added circuitry on one silicon chip
- Microcontrollers are used in embedded computing and most IoT devices are based on microcontrollers



From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

MCU Bit Definition

- The number of bits describing the data path defines the Microcontroller Bit Definition



From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

Common Memory Types in MCUs

- RAM
  - Most microcontrollers have little amount of internal RAM. Generally, 1 Kbytes (“embedded” in MCUs) is a common amount, although some more powerfull microcontrollers can have more
  - RAM is typically used to store variables (global, local, stack,…) and data of your program
  - So use the small amount of RAM wisely!
- EEPROM - electrically-erasable-and-programmable ROM
  - Internally, they are similar to EPROMs, but the erase operation is accomplished electrically, rather than by exposure to ultraviolet light
  - higher cost and write cycles are also significantly longer than RAM
  - EEPROM are typically used to store permanent data
- Flash memory

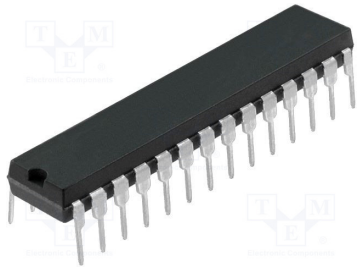
Adapted from "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

Flash memory

- Flash memory devices are
  - high density,
  - low cost,
  - nonvolatile,
  - fast (to read, but not to write), and
  - electrically reprogrammable.
  - Write/Erase large blocks of bytes
- EEPROM is similar to flash memory but the principal difference is that
  - EEPROM requires data to be written or erased one byte at a time whereas flash memory allows data to be written or erased in blocks.
- Flash memory is typically used in MCUs for storing program code

Ex: ATmega328 specs

The high-performance Microchip picoPower 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1024B EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, a 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts.



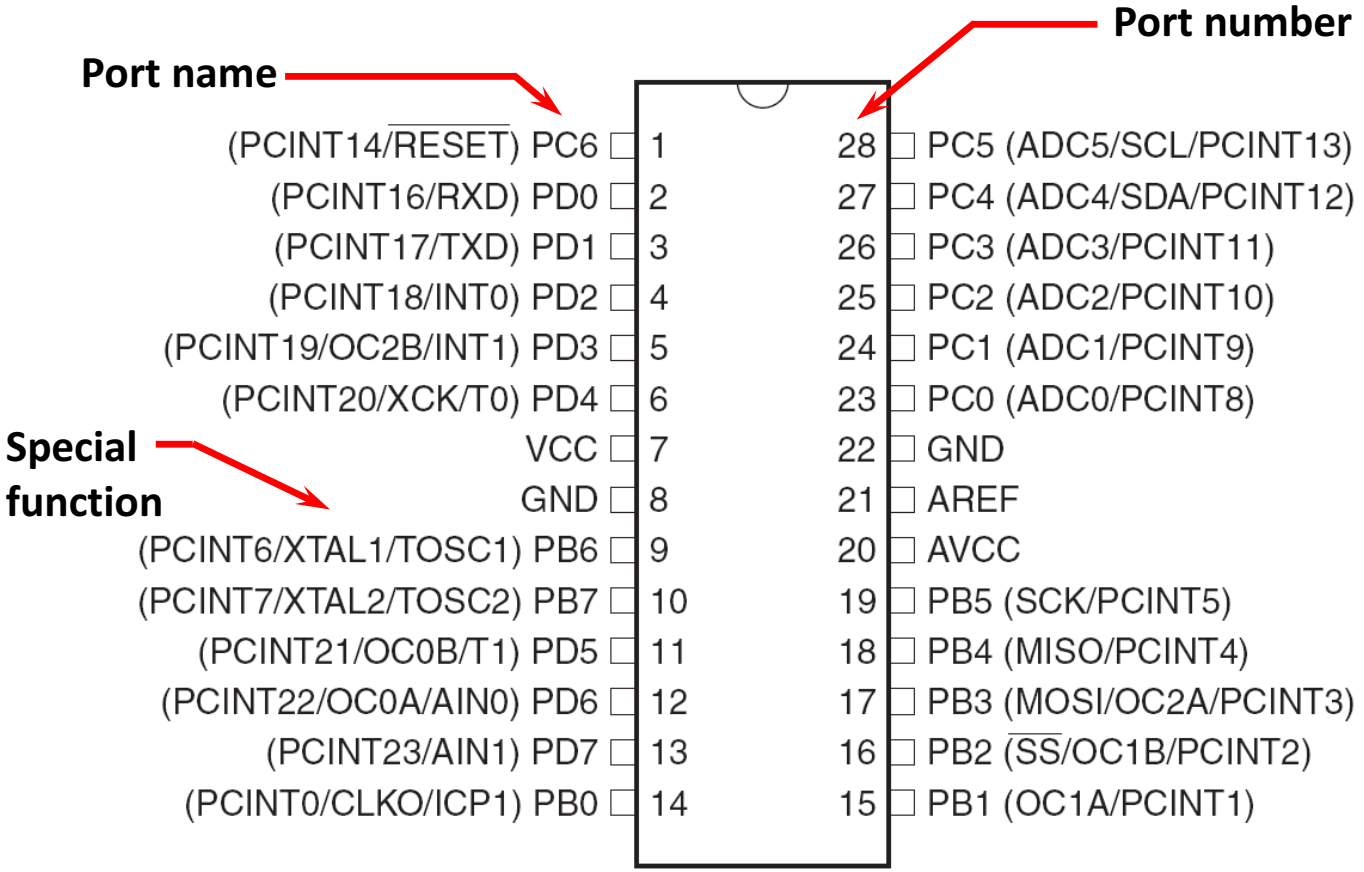
Program Memory Type	Flash
Program Memory Size (KB)	32
CPU Speed (MIPS/DMIPS)	20
SRAM Bytes	2,048
Data EEPROM/HEF (bytes)	1024
Digital Communication Periphe...	1-UART, 2-SPI, 1-I2C
Capture/Compare/PWM Periph...	1 Input Capture, 1 CCP, 6PWM

Timers	2 x 8-bit, 1 x 16-bit
Number of Comparators	1
Temperature Range (C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
Pin Count	32
Low Power	Yes

https://www.microchip.com/wwwproducts/en/ATmega328p

ATmega328 ports

- Ports/pins are communication channels through which information flows into or out of the microcontroller



Port Pin Data Directionality

- Input
  - When you want to take information from the external world (sensors) into the MCU
- Output
  - When you want to change the state of something outside the MCU (turn a led/motor on or off, etc.)
- Pins default to input direction on power-up or reset
- Your program can set or change the directionality of a pin at any time

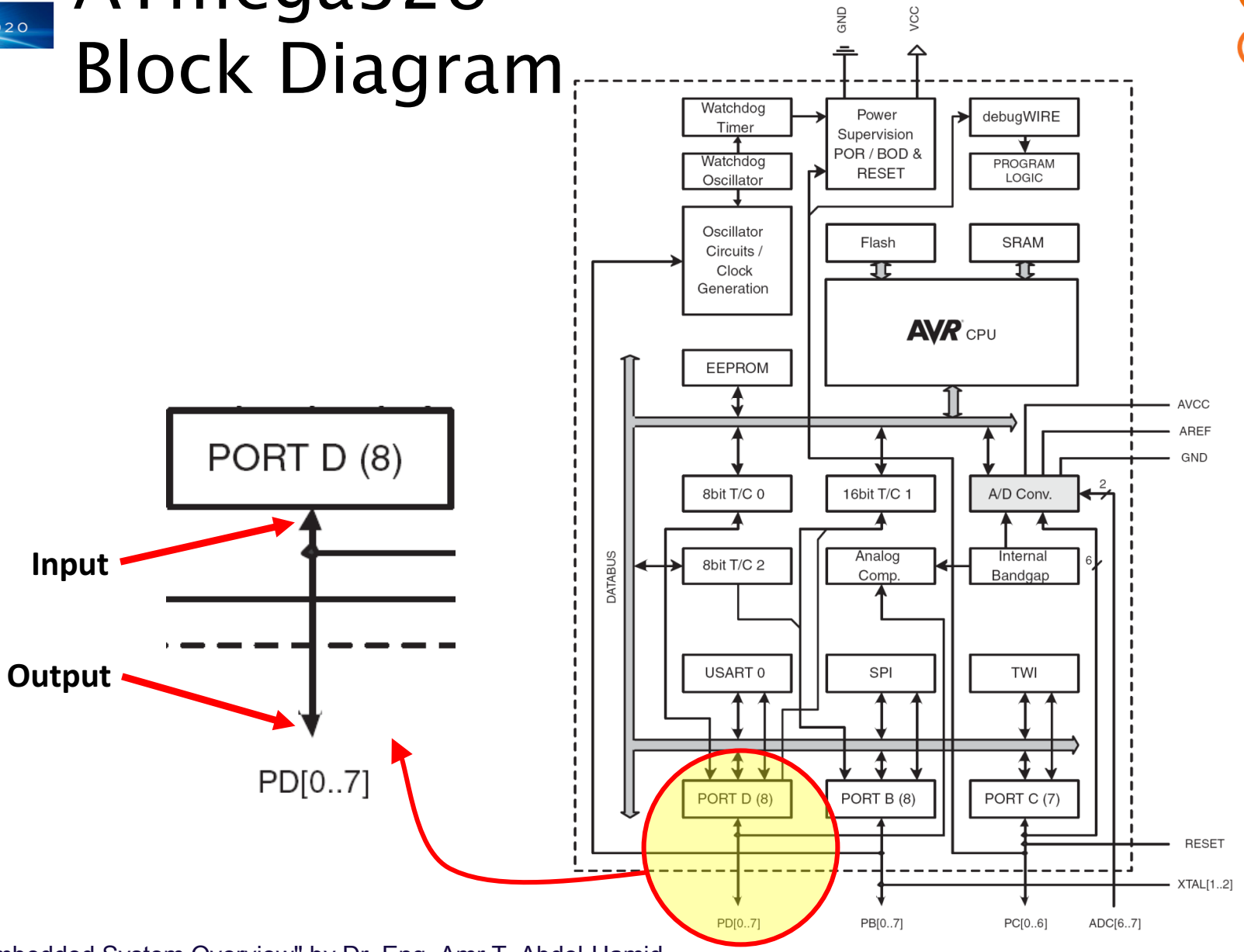
From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

Pin Voltages

- Microcontrollers are fundamentally ***digital*** devices.
- For digital IO pins, information is 'coded' in two discrete states:
  - HIGH or LOW (logic: 1 or 0)
  - Voltages
    - TTL
      - 5 V (for HIGH)
      - 0 V (for LOW)
    - 3.3 V CMOS
      - 3.3 V (for HIGH)
      - 0 V (for LOW)

From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

ATmega328 Block Diagram

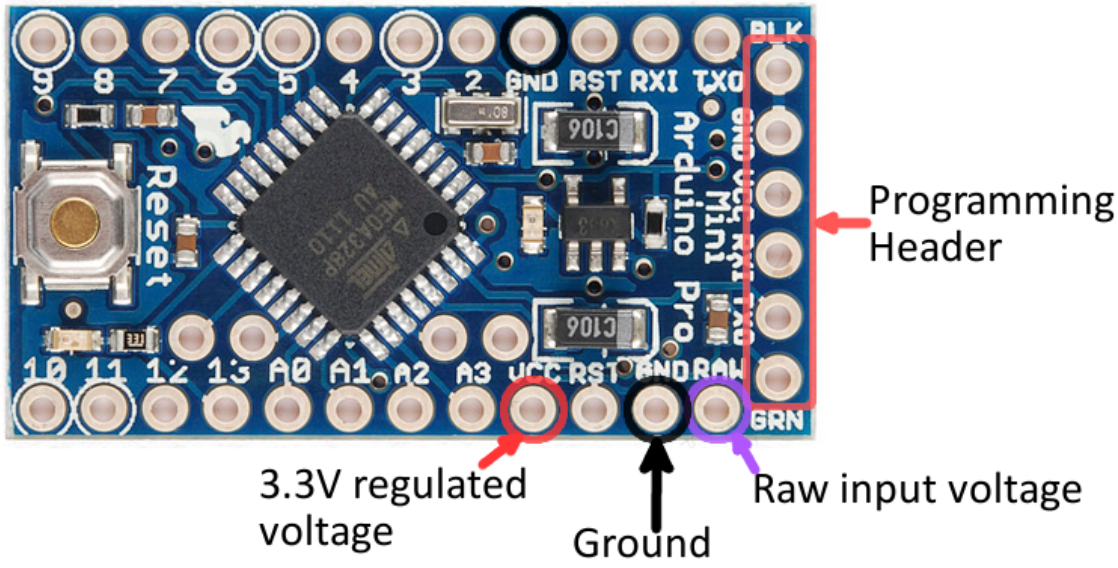


From "An Embedded System Overview" by Dr. Eng. Amr T. Abdel-Hamid

From µcontroller to µcontroller board

- Although a µcontroller can be standalone, it is usually mounted on a board with additional electronics parts
  - Leds
  - Voltage regulators
  - Reset button
  - Serial-USB

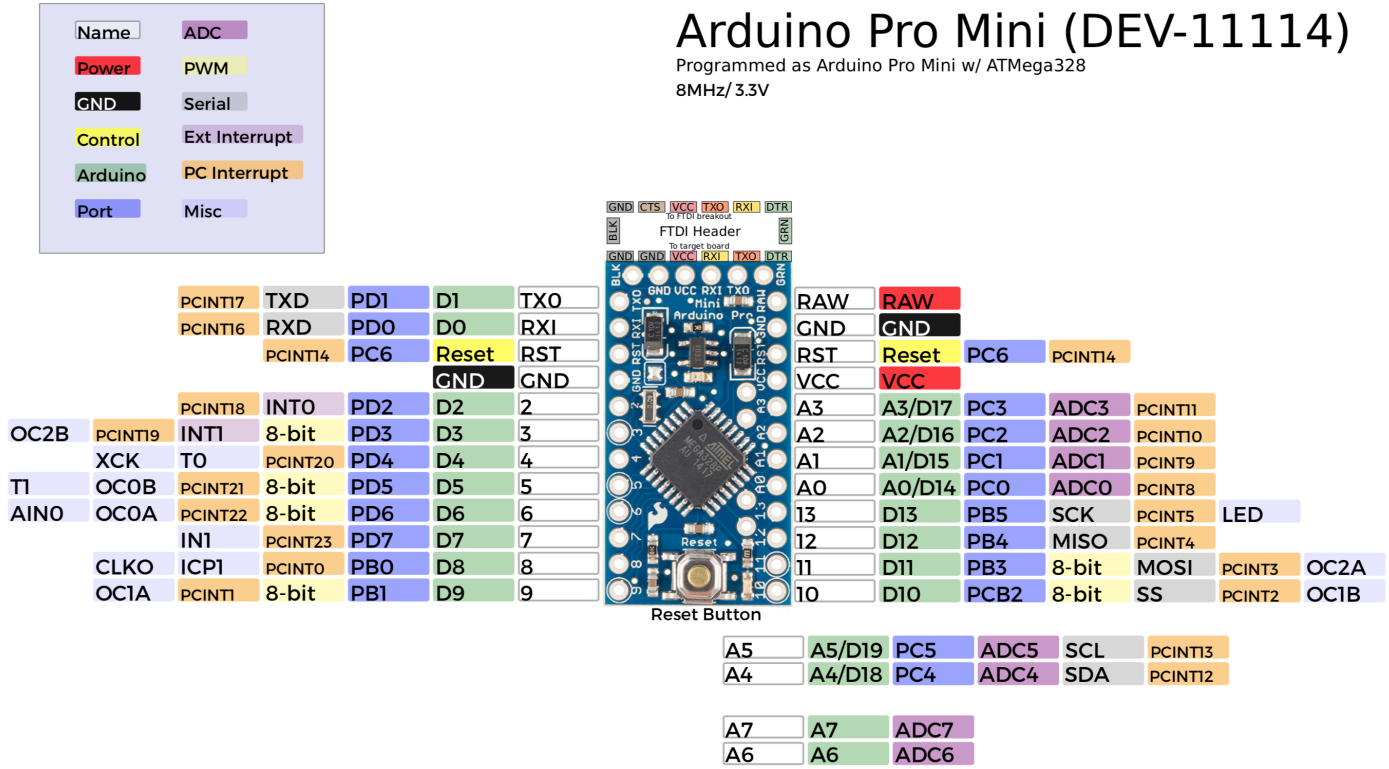
- Ex: Arduino Pro Mini
  - See next slide on Arduino boards



- You can use an existing microcontroller board to build your IoT device or make your own board by integrating yourself the microcontroller

µcontroller pins to µcontroller board pins

- Mapping of µcontroller pins to µcontroller board pins is not always obvious, nor standardized
- So need to look at the pin-out schematic



Arduino boards

WHAT IS ARDUINO?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

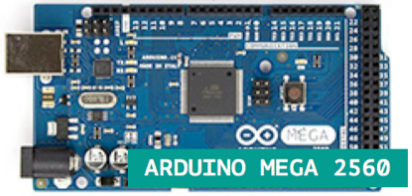
ARDUINO BOARD

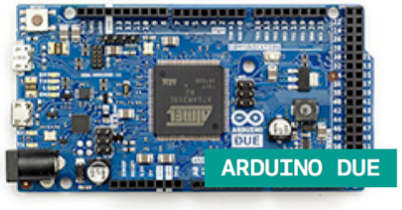
Arduino senses the environment by receiving inputs from many sensors, and affects its surroundings by controlling lights, motors, and other actuators.

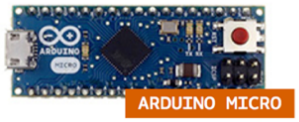
```
void setup() {
 }

void loop() {
 }
```

ARDUINO SOFTWARE

You can tell your Arduino what to do by writing code in the Arduino programming language and using the Arduino development environment.

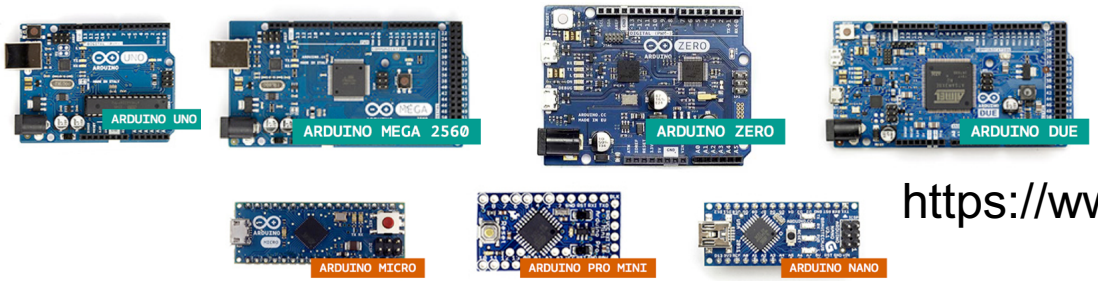






Arduino boards

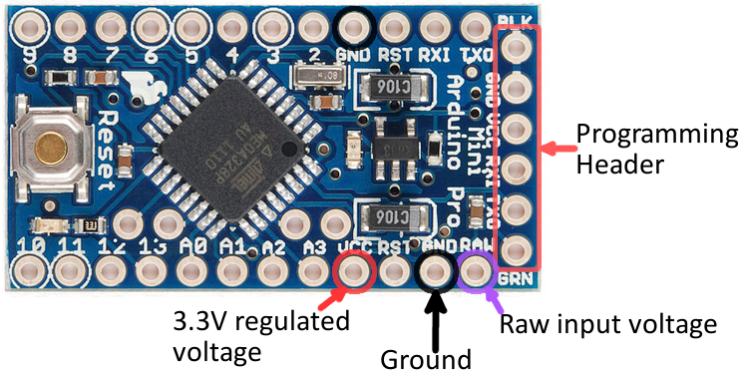
- The Arduino project was started at the *Ivrea Interaction Design Institute* in Italy and targets students, tech enthousiasts and hobbyists
- It is now one of the de facto standard and other boards usually offers Arduino compatibility if they want to attract users
- There are also a lot of Arduino clones from mostly China manufacturers and they are working just fine
- The basic Arduino boards (Uno, MEGA, Due, Mini, Nano) are now completed with a lot of other boards



https://www.arduino.cc/en/Main/Products

Which Arduino board for IoT?

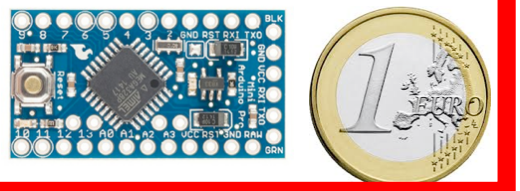
- Depending on the application constraint, any board can be considered suitable for IoT provided that it can have radio capabilities (natively or with additional hardware)
- However, in many IoT applications, size and energy consumption are very important issues so large boards and boards with energy-hungry radio such as WiFi are not really suitable

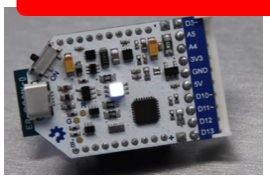


- The previous Arduino Pro Mini which exists in 3.3V and 8MHz version is a great board for early integration phase

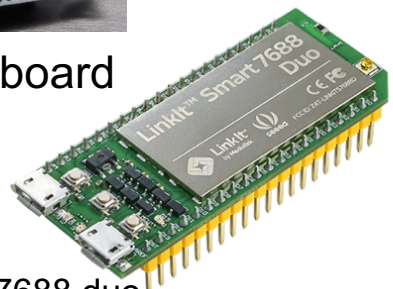
Large ecosystem, still growing…

Arduino Pro Mini





Theairboard

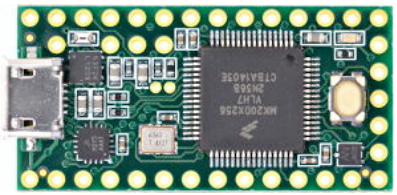


LinkIt Smart7688 duo

LoPy

http://blog.atmel.com/2015/12/16/rewind-50-of-the-best-boards-from-2015/

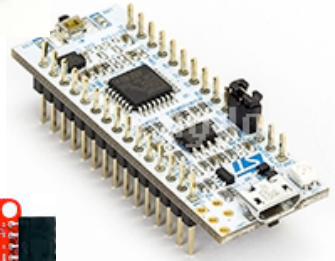
http://blog.atmel.com/2015/04/09/25-dev-boards-to-help-you-get-started-on-your-next-iot-project/

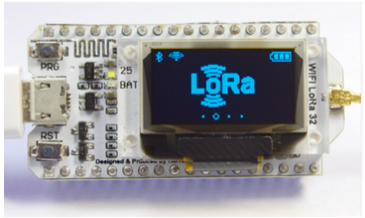


Teensy 3.2

Expressif ESP32

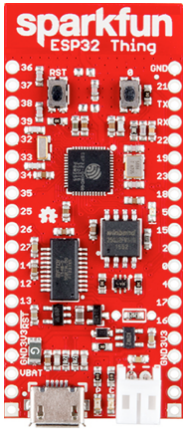
STM32 Nucleo-32





Heltec ESP32 + OLED

Adafruit Feather



Sparkfun ESP32 Thing

Tessel

SodaqOnev2

Tinyduino

The Arduino IDE

- Open source IDE to program Arduino boards
- Arduino boards can be programmed in C++ **-like** language
- Additional functions are introduced to manage I/O and other features such as interrupts, delays ...

Digital I/O

digitalRead()

digitalWrite()

pinMode()

Analog I/O

analogRead()

analogReference()

analogWrite()

Time

delay()

delayMicroseconds()

micros()

millis()

External Interrupts

attachInterrupt()

detachInterrupt()

Interrupts

interrupts()

noInterrupts()

Communication

Serial

Stream

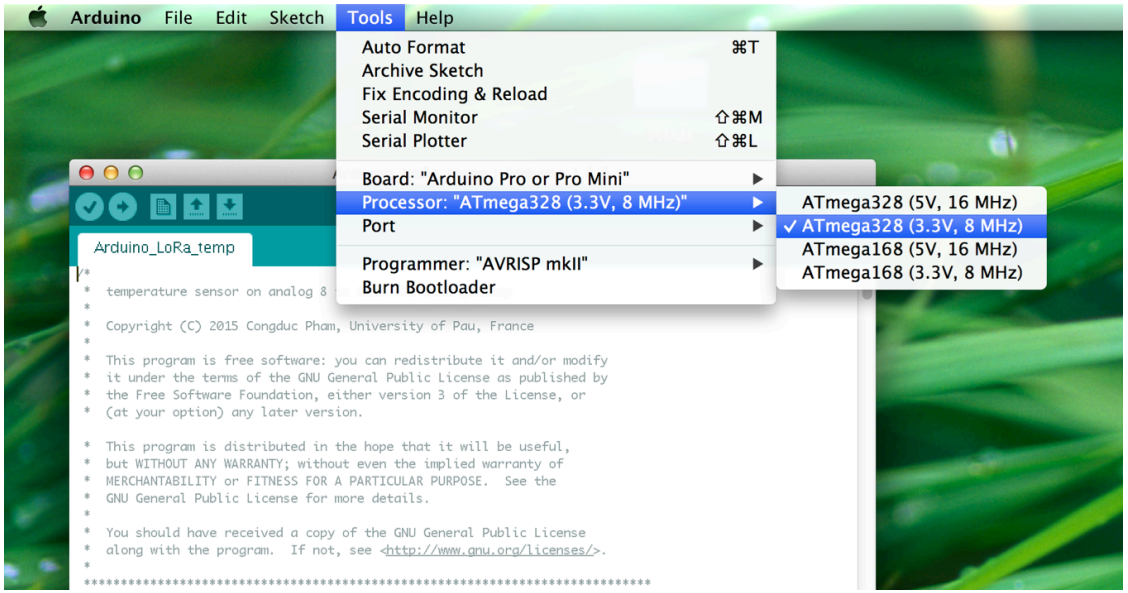
https://www.arduino.cc/reference/en/

Structure of an Arduino Program

- An arduino program == 'sketch'
  - Must have:
    - setup()
    - loop()
  - setup()
    - configures pin modes and registers
  - loop()
    - runs the main body of the program forever
      - like while(1) {...}
  - Where is main() ?
    - Arduino simplifies things
    - Does things for you

```
/* Blink  -  turns on an LED for DELAY_ON msec, then
off for DELAY_OFF msec, and repeats
BJ Furman rev. 1.1   Last rev: 22JAN2011
*/
#define LED_PIN  13    // LED on digital pin 13
#define DELAY_ON  1000
#define DELAY_OFF 1000

void setup()
{
  // initialize the digital pin as an output:
  pinMode(LED_PIN, OUTPUT);
}

// loop() method runs forever,
// as long as the Arduino has power

void loop()
{
  digitalWrite(LED_PIN, HIGH);    // set the LED on
  delay(DELAY_ON);  // wait for DELAY_ON msec
  digitalWrite(LED_PIN, LOW);   // set the LED off
  delay(DELAY_OFF);  // wait for DELAY_OFF msec
}
```

Compiling

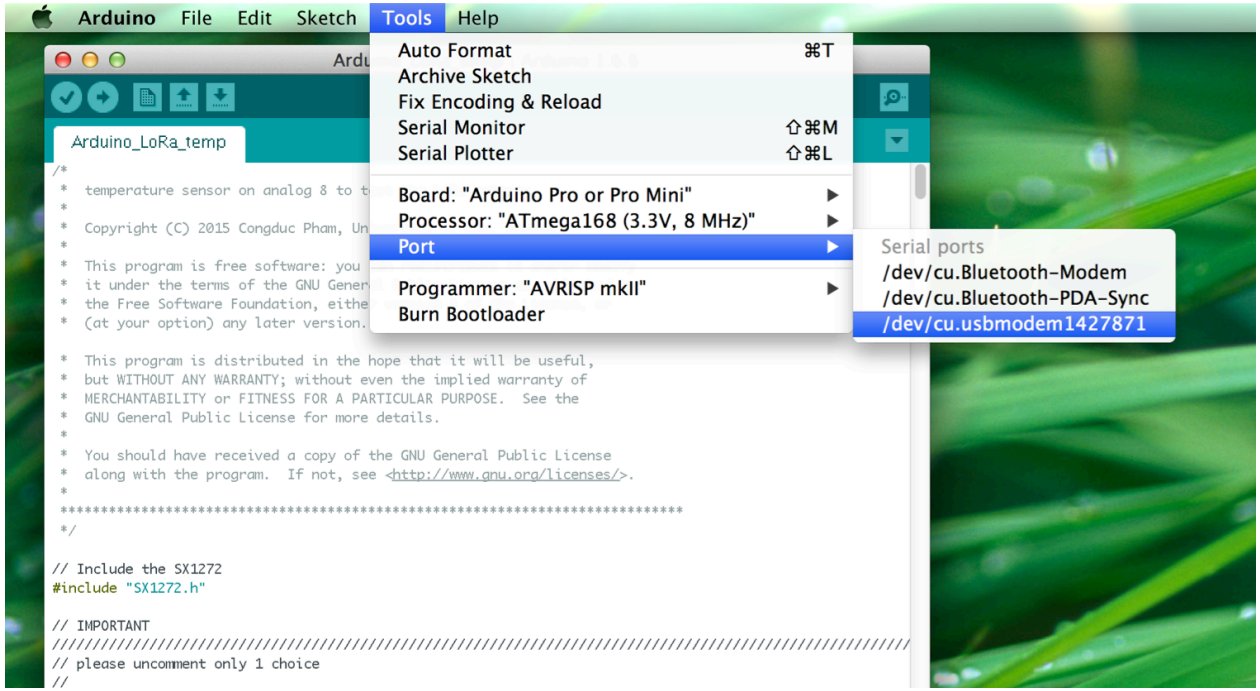


Select the Arduino board you have, here Arduino Pro Mini 3.3V 8MHz version

Then, click on the « verify » button 

Prof. Congduc Pham
http://www.univ-pau.fr/~cpham

Uploading

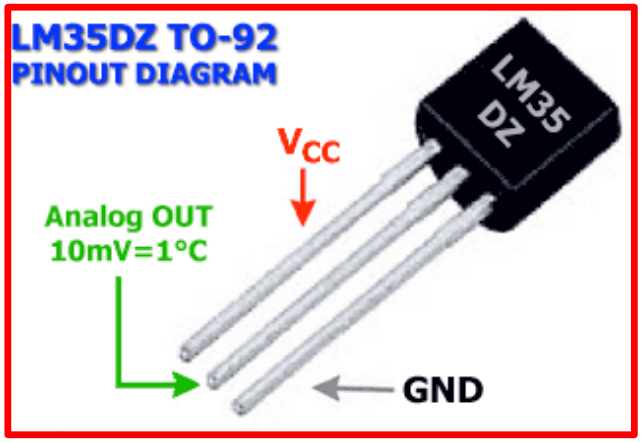


Connect the USB end to your computer and the USB port should be detected in the Arduino IDE. Select the serial port for your device. It may have another name than what is shown in the example. Then click on the « upload » button

Connecting a physical sensor

- Main feature of IoT!

Basic: analog output

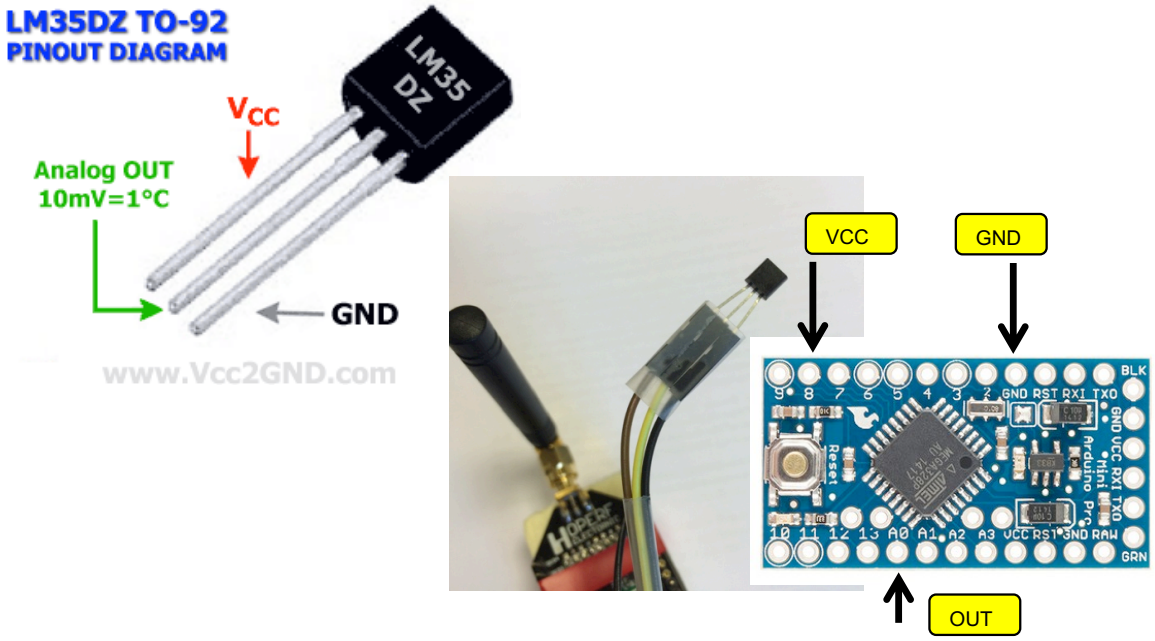


Vcc is typically 5V or 3.3V, assuming 3.3V here

If 0 means 0V and 1024 means 3300mV (10-bit resolution) then 3300mV/1024=3.22mV is the granularity of the measure

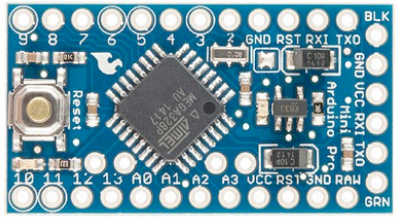
A digital value of 100 means 100*3.22mV=322mV
If the sensor output is 10mV/1°C then the physical temperature is 322mV/10mV=32.2°C

Connecting to board



GND should be connected to one of the board's GND, VCC can be connected to the output of digital pin 8 for instance (to get power) and the OUT pin can be connected to the analog A0 pin.

Reading analog pin value




```
// sensor output connected to A0 analog pin

value = analogRead(A0);

// now need to convert to Celcius degree
```

And converting into Celcius

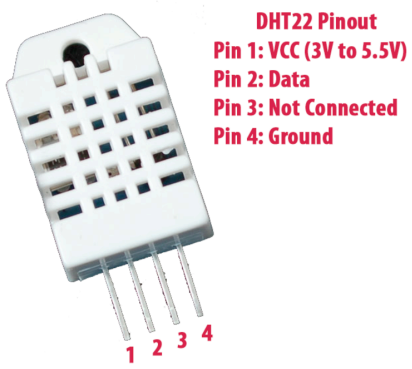
```
Temp = value * 3300.0/1024.0;        // 3300/1024=3.22

Temp = Temp / 10;                    // 10mV means 1°C

// now process and transmit the data
```

Advanced: digital output

- Sensors with a digital output usually provide the sensed data in some advanced digital format, not only an analog value.
- In that case, additional libraries are often needed to read the sensor's data depending on the interface/protocol/bus
  - Serial UART (traditional serial line)
  - 1-Wire
  - I2C: Inter-Integrated Circuit, or TWI: Two-Wire Interface
  - SPI: Serial Peripheral Interface
- In many cases, there are both library and example provided with the sensor to read the data, so do not start from scratch!
- Example with the DHT22 sensor follows. You can search for "DHT22 Arduino" on the web

DHT22 and digital output

- The DHT22 is a temperature and humidity sensor with a digital output on pin 2



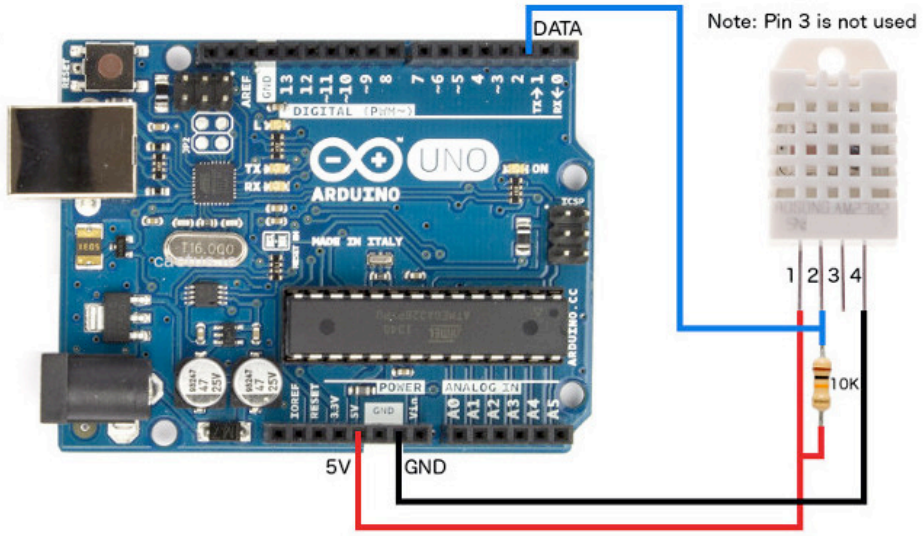
- There are various libraries available (search Arduino+DHT22 on the web)
- We use the library developed by Ben Adams: https://github.com/nethoncho/Arduino-DHT22

Sensor preparation

- Many sensors need a little preparation before you can safely connect them to a microcontroller board
- Usually, you may need to add some resistor and/or condensator to some pins or wires
- Consult the sensor datasheet or search the web on the required schematic for a safe and proper operation of the sensor

Ex: reading the DHT22

```
// include Ben Adams's DHT22 library
#include "DHT22.h"

DHT22* dht = NULL;

// provide the digital pin to which the
// DHT22 data pin is connected to. Here
// digital pin 2 on the Arduino
dht = new DHT22(2);

// start reading
dht->readData();

// then get both temperature and humidity
double temp=(double)dht->getTemperatureC();
double hum=(double)dht->getHumidity();
```



Ex: DS18B20

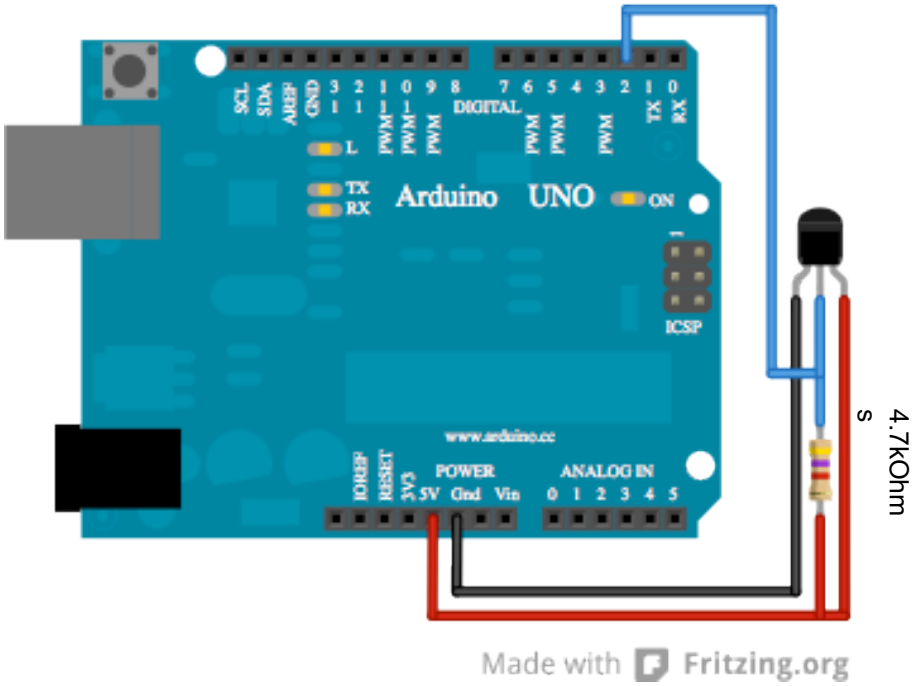
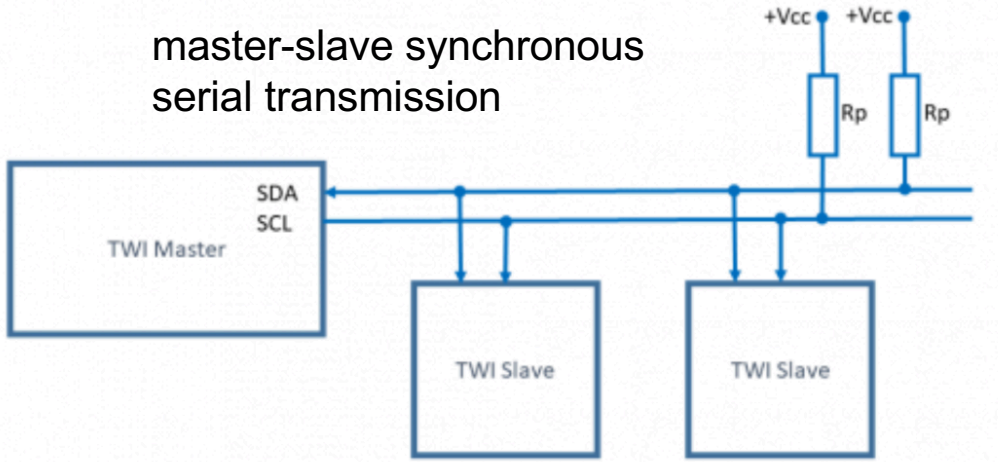


Image from http://fun-tech.se/FunTechHouse/RoomTemperature/

I2C bus

- I2C (Inter-Integrated Circuit) is one of the most popular communication protocol used in embedded systems. It has been designed by Philips for simple audio-video appliances controlled by the microprocessor
- There are many chips that can be connected to the processor with this interface which uses SDA (data) and SCL (clock)
  - EEPROM memory chips
  - RAM memory chips
  - AD/DA converters
  - Real-time clocks
  - Sensors (temperature, pressure, gas, air pollution)

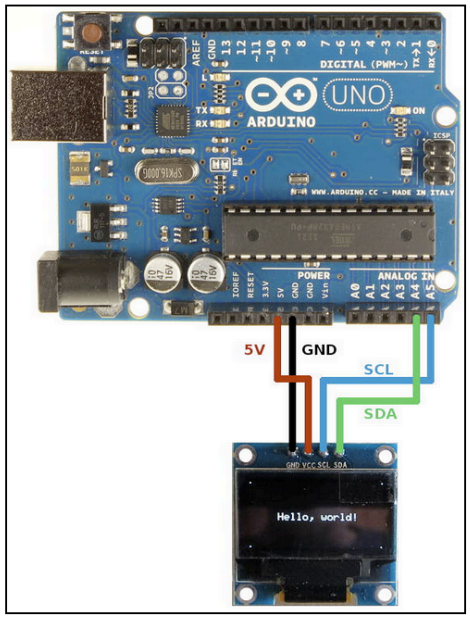


A slave is assigned an 8-bit address

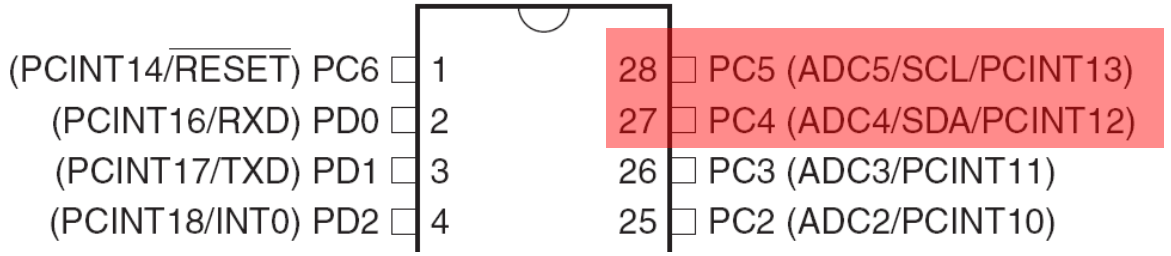
From iot-open.eu IoT course book

I2C example: connecting OLED

- Small OLED screens usually use I2C communication interface with SDA and SCL wire
- Just connect the corresponding wires to their matching pins on the Arduino. Usually SDA=A4 and SCL=A5 (100kHz-400kHz)



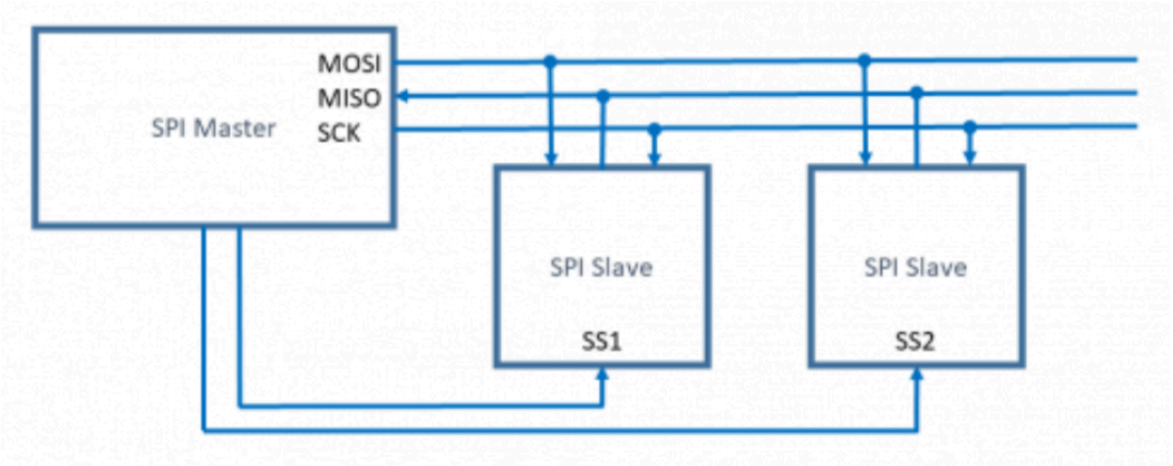
SHT Sensor Pins	Arduino Pins
VCC	5V or 3.3V
GND	GND
SDA	A4
SCL	A5



SPI bus

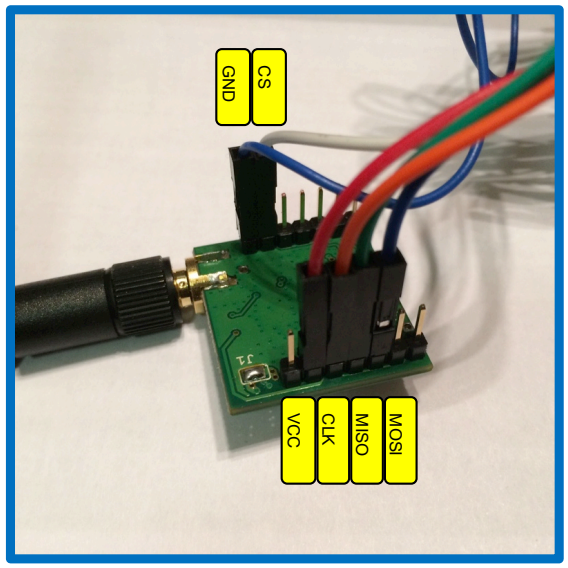
- SPI (Serial Peripheral Interface) is a synchronous serial interface and protocol that can transmit data with speed up to 20Mbps
- To communicate SPI uses three lines common to all of the connected devices, and one enabling line for every slave element

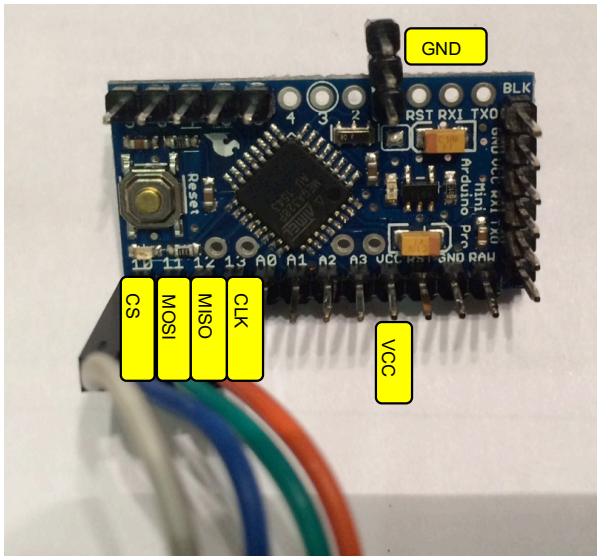
Line	Description	Direction
MISO	Master In Slave Out	peripheral → uC
MOSI	Master Out Slave In	uC → peripheral
SCK	Serial Clock	uC → peripheral
SS	Slave Select	uC → peripheral

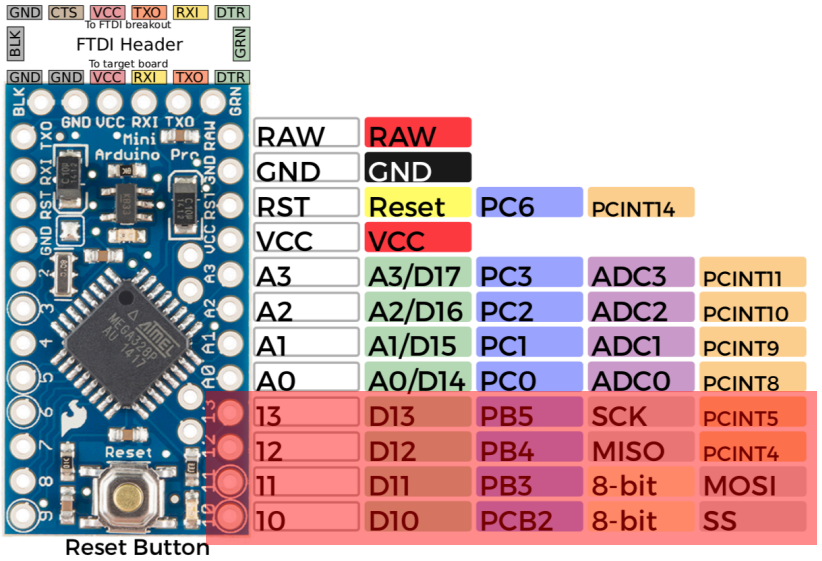


SPI example: connecting radio module

- Many radio modules use SPI interface to communicate with the host microcontroller
- Example with a LoRa radio module





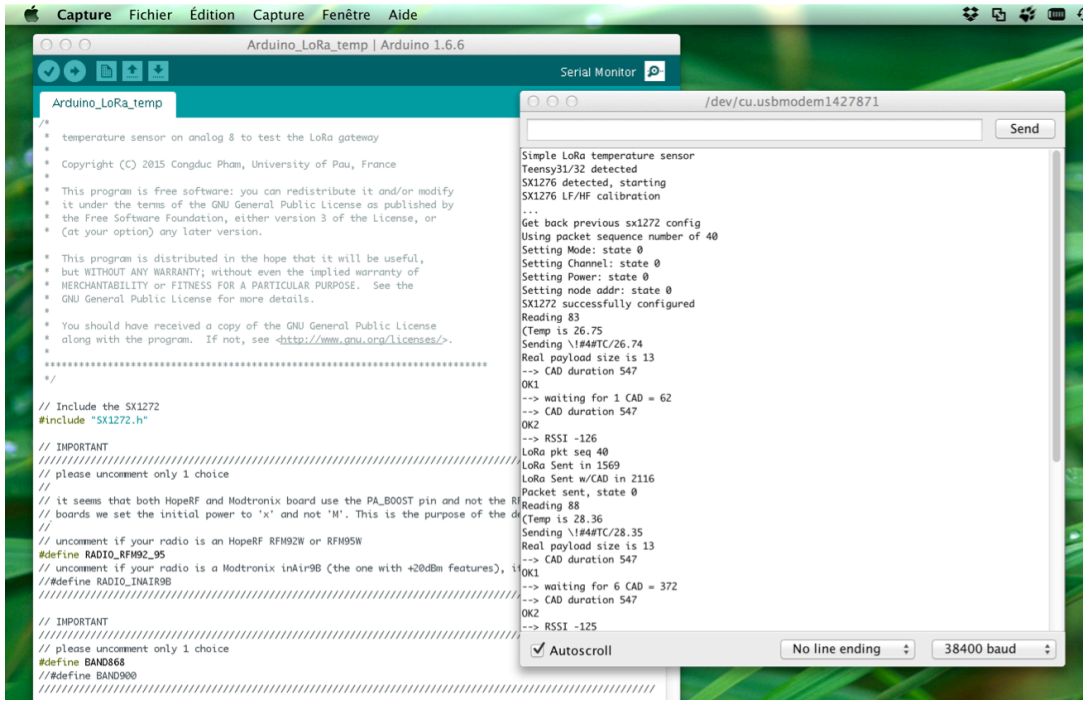


RAW	RAW			
GND	GND			
RST	Reset	PC6	PCINT14	
VCC	VCC			
A3	A3/D17	PC3	ADC3	PCINT11
A2	A2/D16	PC2	ADC2	PCINT10
A1	A1/D15	PC1	ADC1	PCINT9
A0	A0/D14	PC0	ADC0	PCINT8
13	D13	PB5	SCK	PCINT5
12	D12	PB4	MISO	PCINT4
11	D11	PB3	8-bit	MOSI
10	D10	PCB2	8-bit	SS

Printing text output

- Most microcontrollers do not have display
- When connected to a computer, output can be displayed by the computer, mostly for debugging purposes
- In most cases, the connection is realized using serial interface (Universal Asynchronous Receiver Transmitter, UART)
- With the Arduino IDE, use
  - `Serial.begin(38400);`
  - `Serial.print("The value is: ");`
  - `Serial.println(my_value);`
- There are libraries or turnaround to have C-like `printf` function
- Or use `sprintf` then `Serial.print`

Serial monitor



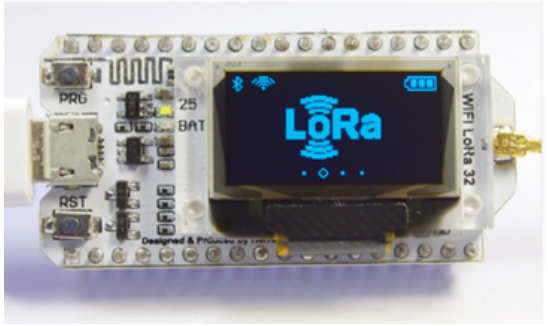
You can see the output from the sensor if it is connected to your computer. Use the Arduino IDE « serial monitor » to get such output, just to verify that the sensor is running fine, or to debug new code. Use same baud rate (e.g. 38400 baud) that the one fixed by the program.

Additional shields

- Most of microcontroller boards have very basic capabilities
- Additional hardware implementing advanced features can be added by using shields
  - Ethernet, Bluetooth, 2G/3G, WiFi,...
  - Real-Time Clock
  - SD card
  - Audio, Display
  - High power relay
  - GPS
  - ...
- Most of the addons use I2C or SPI buses

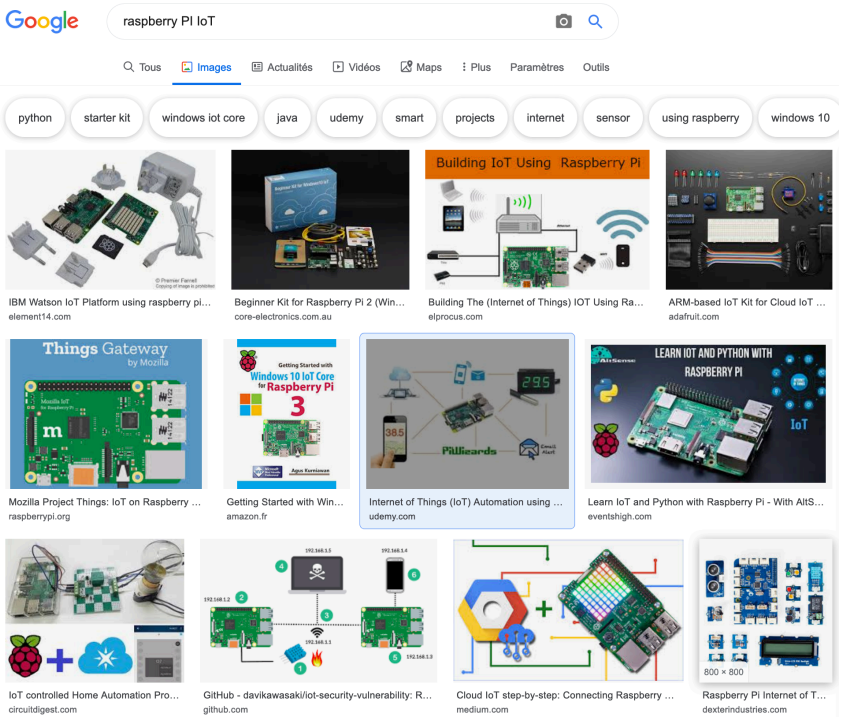
Other MCU boards

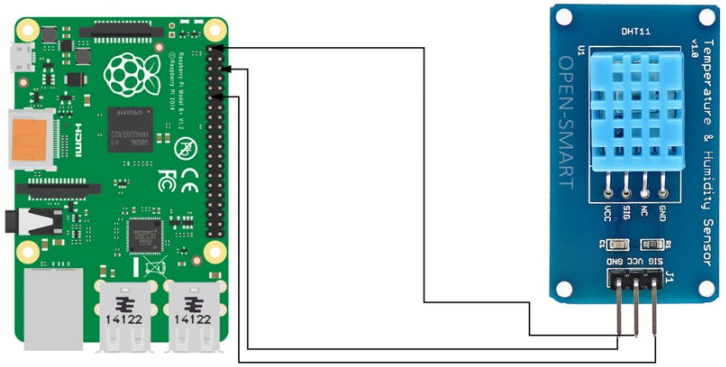
- ESP32
  - 32 bit
  - WiFi
  - Heltec ESP32 has embedded OLED screen
- Teensy3X
  - 32 bit, ARM Cortex
  - Lots of RAM (e.g. 96KB of RAM)
  - https://www.pjrc.com/teensy/
- Many boards are compatible with the Arduino IDE by installing dedicated plug-ins
- It is always possible to use another IDE (Eclipse) but sometime it is very difficult to make the setup



What about single-board computers?

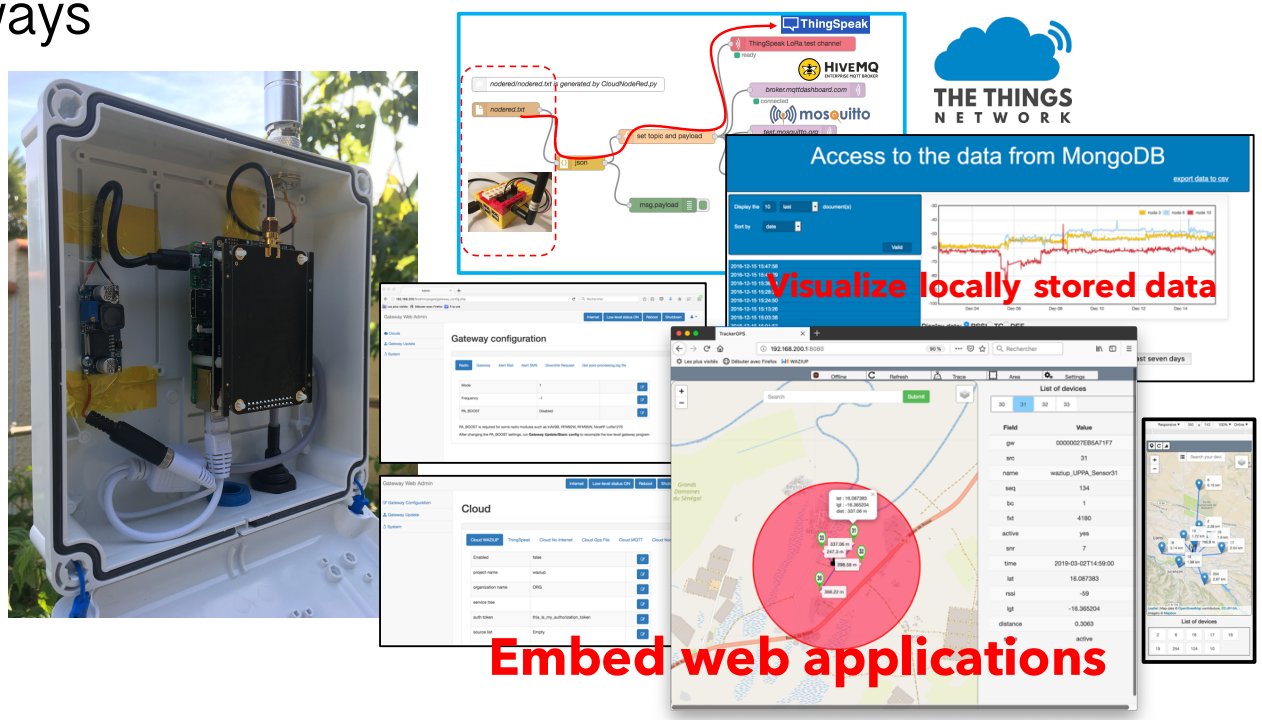
- Single-board computers such as the well-known Raspberry PI are oftenly advertised as IoT devices
- However, as they are very powefull, their high energy consumption make them not suitable in many IoT applications





Where are they useful?

- Most of these single-board computers can run a complete Linux-based OS with various connectivity features
- They are definitely suitable as low-cost, flexible, versatile IoT gateways



Additional materials/tutorials (1)

- IoT course book from iot-open.eu
  - http://iot-open.eu/download/io1-introduction-to-the-iot/
- Slides
  - https://github.com/CongducPham/tutorials/blob/master/Low-cost-LoRa-IoT-step-by-step.pdf
  - https://github.com/CongducPham/tutorials/blob/master/Low-cost-LoRa-IoT-outdoor-step-by-step.pdf
- Videos
  - Build your low-cost, long-range IoT device
    - https://www.youtube.com/watch?v=YsKbJeeav_M
  - Extreme low-cost & low-power LoRa IoT for real-world deployment
    - https://www.youtube.com/watch?v=2_VQpcCwdd8
- Online tutorial/course
  - http://cpham.perso.univ-pau.fr/LORA/WAZIUP/tuto/index.html

Additional materials/tutorials (2)

- Resources from Arduino Community
  - https://www.arduino.cc/en/Tutorial/HomePage
- Resources from Adafruit
  - https://learn.adafruit.com/
- Resources from Instructables.com
  - https://www.instructables.com/circuits/arduino/projects/
- Resources from hackster.io
  - https://www.hackster.io/arduino/projects
- Resources from makerspaces.com
  - https://www.makerspaces.com/we-love-maker-educators/
- Resources from circuit digest
  - https://circuitdigest.com/arduino-projects
- Resources from Electronics Hub
  - https://www.electronicshub.org/arduino-project-ideas/
- And much more, just crawl the web!

IOT ONLINE COURSE

Fundamentals of IoT

Continue with
F-IOT-3: Introduction to Arduino IDE
F-IOT-4: Low-cost & Open-source Technologies for Low-Cost IoT

WAZIUP

IoT – from idea to reality

WAZIHUB