Arduino y el IoT

Taller Teórico-Práctico

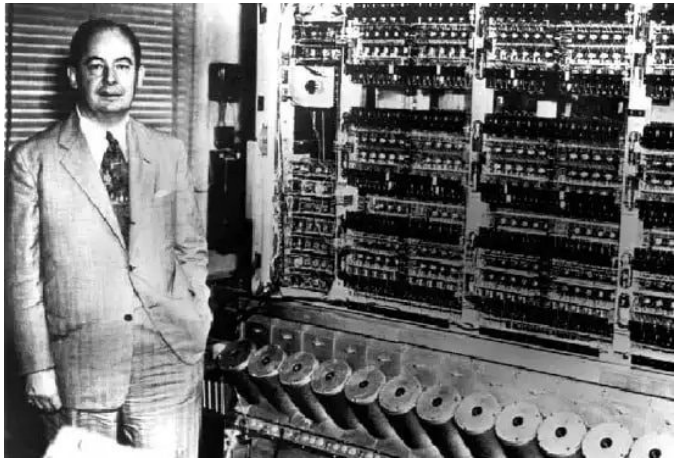
Ing. Antonio Russoniello

Arduino y el IoT

"El mundo ha llegado a la edad de dispositivos baratos y complejos de gran confiabilidad; algo importante saldrá de esto"

Vannevar Bush, "*Cómo podríamos pensar*", 1945

Memex: Vannevar Bush fue el primero en describir el funcionamiento del Memex en su artículo **As we may think**, de 1945, capaz de crear y seguir enlaces entre distintos documentos almacenados en microficha, un sistema muy parecido a lo que hoy conocemos como un hipertexto.



Arduino y el IoT

“tenemos que convertir a nuestras computadoras elefantes en saltamontes, porque con la reducción de escala dramática, las propiedades básicas de la tecnología cambiarán y abrirán la puerta para la creación de componentes mucho más pequeños con una capacidad computacional nunca imaginada antes”

Douglas Carl Engelbart, 1959

1968

Arduino y el IoT

“Las tecnologías más importantes son las que desaparecen. Son las que se entrelazan con el tejido de nuestra vida cotidiana hasta que ya no son distinguibles de ella.”

Mark Weiser

(padre de la computación ubícua), 1988

Acuñó el término "computación ubicua" en 1988 para describir un futuro donde las computadoras están integradas sin problemas en los objetos cotidianos y nuestro entorno.

Abogó por computadoras que sean invisibles, se desvanezcan en el fondo y desaparezcan de nuestra conciencia.

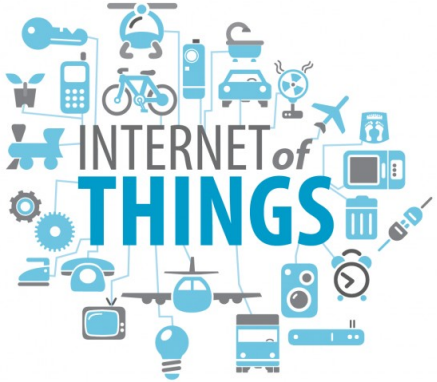
Arduino y el IoT

¿Qué es IoT ?

The IoT contains an enormous variety of connected objects

Tiny Stuff: Smart Dust

Computers smaller than a grain of sand can be sprayed or injected almost anywhere to measure chemicals in the soil or to diagnose problems in the human body.



Arduino y el IoT

IoT Aplicaciones

- **Domótica:** viene de la unión de las palabras domus (casa en latín) y tica (automática, palabra en griego que significa que algo funciona por si mismo)
- **Inmótica:** automatización de edificios de gran tamaño (hoteles, centros comerciales, hospitales, universidades, plantas industriales)
- **Urbótica**: automatización y gestión de ciudades de manera inteligente

Arduino y el IoT

IoT Aplicaciones

- Video vigilancia en calles
- Generación de energía: conversión de energía cinética en energía renovable
- Gestión energética: climatización, gestión eléctrica, energías renovables
- Seguridad: alarmas de intrusión, cierre de persianas, detección de incendio, fugas de gas, alerta médica, tele asistencia.
- Calidad de vida: iluminación, gestión de dispositivos multimedia, control remoto por internet
- Accesibilidad: facilidad de acceso a instalaciones, autonomía de personas que presenten alguna discapacidad.
- Comunicación: informes de consumo, transmisión de alarmas y otra información, integración de dispositivos con servicios de voz IP

Arduino y el IoT

Factores aceleradores del IoT

- Miniaturización (nano tecnología)
- Adopción masiva de IPv6 y nuevos protocolos de red ()
- Protocolos M2M optimizados
- Avances recientes en redes neuronales (autonomía)
- Conectividad y ancho de banda (Fibra, 5G)
- Reducción de costos de sensores
- Infraestructura para computación en la nube (esalabilidad y volumen de datos)

Arduino y el IoT

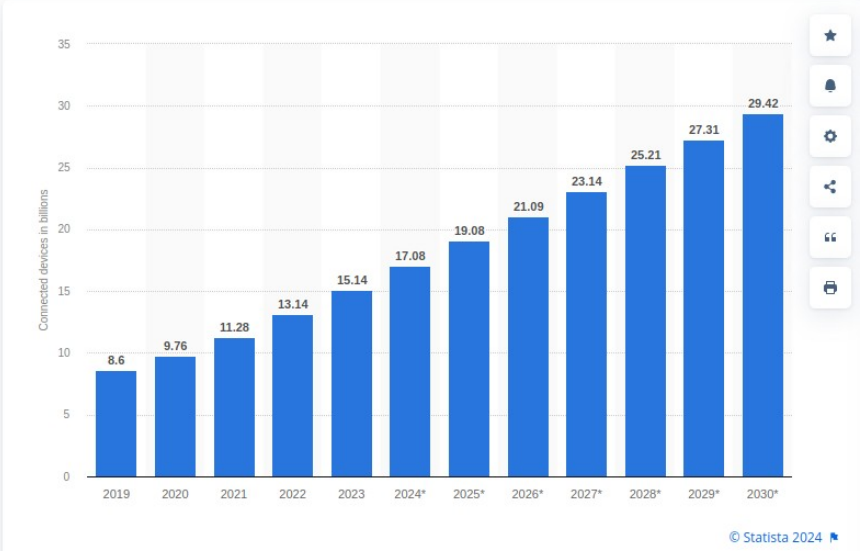
Retos

- Optimización de uso de baterías
- Necesidad de estandarización (fragmentación del mercado)
- Debe mejorarse la eficiencia de algoritmos y protocolos
- Mejorar la seguridad de los dispositivos
- Privacidad (recopilación y uso de los datos)

Arduino y el IoT

Proyecciones para IoT

Statista: Pronostica que habrá 17.100 millones de dispositivos IoT conectados en todo el mundo para el 2025, y 30.900 millones para el 2030



McKinsey & Company: Prevé que la inversión global en IoT alcanzará los 1.5 mil millones de dólares para el 2030

https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

IOT ANALYTICS

April 2023

Your Global IoT Market Research Partner

The Leading IoT Software Companies 2023

Based on feedback from IoT adopters[1]

Applications[2] (selection only)

Remote Asset Access

Microsoft, ProSoft, intel, ZOHO, Cisco, ORACLE, N-ABLE, TeamViewer, evatec, jumpcloud, ninjaOne, Pulseway, AVEVA, CONNECTWISE

AR/VR

YORD, Meta, iTechArt, ptc, WIR, CXR, REPLY, Microsoft, SAMSUNG, HQ SOFTWARE

Digital Twin

SIEMENS, Microsoft, aws, AUTODESK, anylogic, celonis, GE, LANNER, cadline, CUPIX BUILD SMART

Security

Cisco, IBM, aws, paloalto, Microsoft, FORTINET, praetorian, AhnLab, Symantec, McAfee, Bitdefender, SUBEX

Middleware/Platforms

IoT Platforms

Microsoft, Cisco, aws, Honeywell, Capgemini, SIEMENS, Google, ptc, Schneider Electric, dynatrace, ABB, Blynk, Rockwell Automation, ANDERSEN, ITAC SOFTWARE, GE, ADVANTECH, software AG, AVEVA, SAP, Telkom IoT, TOPCON, ORACLE, EMERSON, IBM, tcs, beanTech, WISE, openremote, Trimble, UnifyTwin, SCIENCESOFT, infor, ZEBRA, salesforce, BOSCH, PARTICLE, hakuna matata, 亚控科技, WellinTech

Edge Mgmt. Platforms

SIEMENS, aws, IBM, Microsoft, Cisco, AVEVA, DELL, aruba, ALIZENT, Schneider Electric, Hewlett Packard Enterprise, software AG, EPICOR, EQUINIX

Databases

mongoDB, aws, SIEMENS, Microsoft, Cisco, ORACLE, APACHE, OSIsoft, CrateDB, influxdb, McObject, kinetica, salesforce, IBM

Other IoT Middleware

Microsoft, ORACLE, Cisco, aws, SAP, NUCLEUS SOFTWARE, IBM, Informatica, tuya, MuleSoft, DELL

Data ingestion tools

Microsoft, aws, APACHE, tamr, TIGERGRAPH, AUTODESK, SIEMENS, Qlik, Adobe, IBM, snowflake, inductive automation, salesforce, SAP

Msg. Brokers/Protocol Converters

SIEMENS, Microsoft, ptc, IBM, aws, RabbitMQ, APACHE, Honeywell, HIVEMQ, mosquitto, MuleSoft

Devices/Edge

Operating system for IoT devices

aws, Microsoft, IBM, WINDRIVER, Red Hat, ORACLE, CANONICAL, Envision digital

Notes: 1. The logos shown here present a non-exhaustive list of companies that were highlighted by respondents of an extensive global IoT adopter survey of 100 organizations, conducted in October 2022; 2. Only 3 selected IoT applications shown - there are many more.

Source: IoT Analytics Research 2023 – IoT Software Adoption Report 2023. We welcome republishing of images but ask for source citation with a link to the original post or company website.

Arduino y el IoT

¿Que son y para qué sirven los nanómetros?

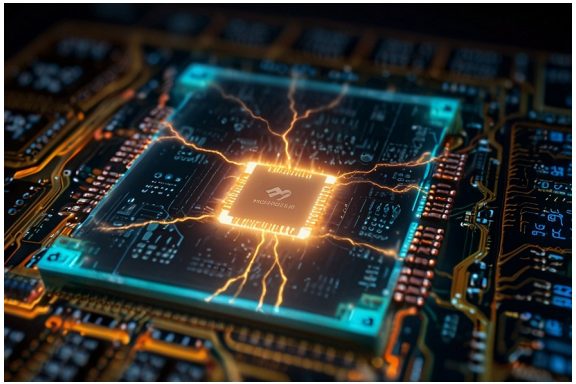
Medida de longitud del sistema métrico.

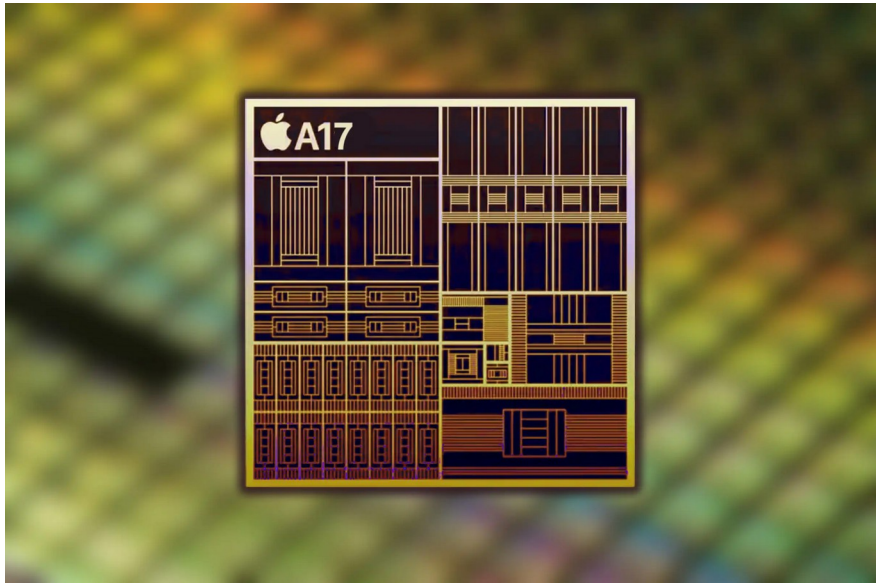
Un nanómetro es la **milmillonésima parte de un metro**. Un cabello humano promedio tiene aproximadamente 60.000 nanómetros de espesor. Los nanómetros se usan para medir las longitudes de onda de la luz y las distancias entre los átomos de las moléculas

- En general, se estima que se pueden integrar entre **30 y 70 millones de MOSFETs por milímetro cuadrado en tecnologías de 7 nm**.
- Procesadores TSMC N7 (7 nanómetros): Alcanzan entre 40 y 50 millones de MOSFETs por milímetro cuadrado.
- Procesadores Samsung N7 (7 nanómetros): Prometen entre 50 y 60 millones de MOSFETs por milímetro cuadrado

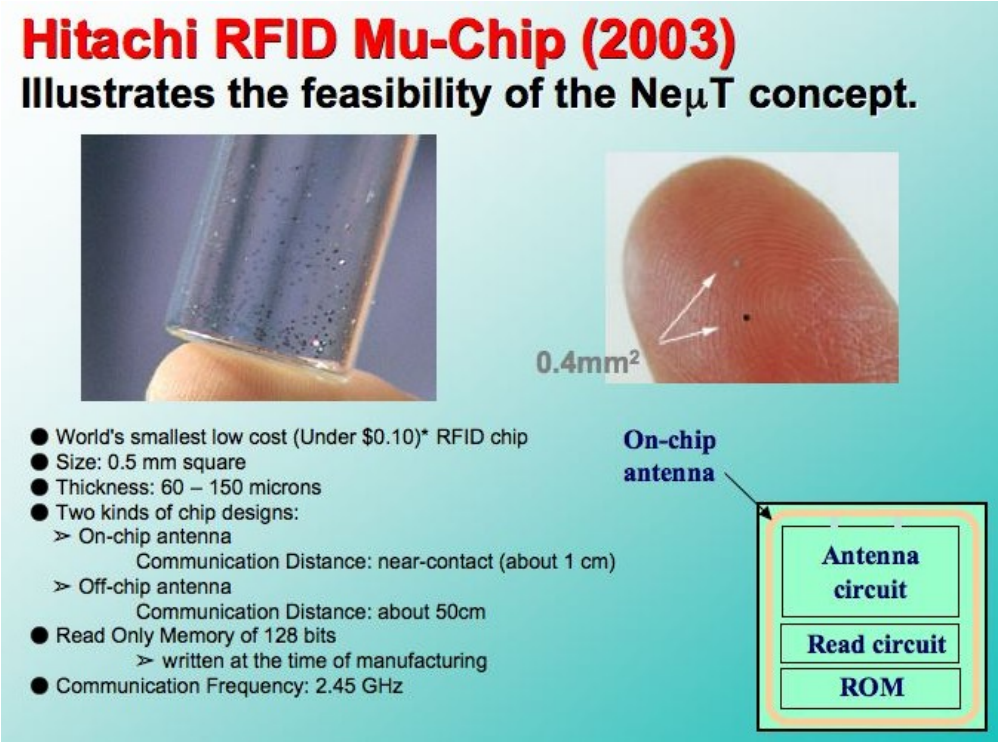
Arduino y el IoT

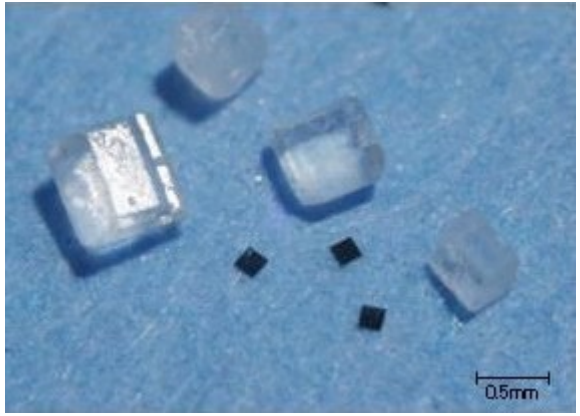
En general, se estima que la tecnología de 3nm puede permitir la integración de entre 300 y 400 transistores por nanómetro. Esto significa que **un chip de 3nm podría albergar entre 90 y 120 mil millones de MOSFETs**





Arduino y el IoT





El último chip, que aún no tiene nombre, es 60 veces más pequeño que el Mu-chip pero puede manejar la misma cantidad de información que se almacena como un número de 38 dígitos, según Hitachi. (2007)

Arduino y el IoT

¿Cómo los conectamos a la Internet?

Necesitamos un mecanismo de direccionamiento global

¿lo tenemos?

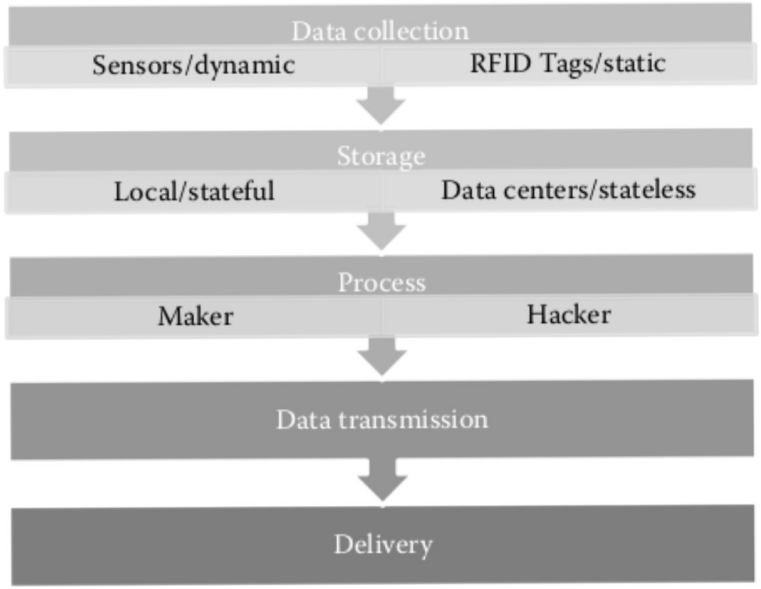
Arduino y el IoT

IPv6 puede asignar aprox. 670 mil millones de direcciones por cada milímetro cuadrado de la superficie de la tierra

En términos matemáticos, IPv6 puede asignar 2^128 direcciones, lo que equivale a aproximadamente 340 trillones de trillones de direcciones. Esto es un número inimaginablemente grande, y es mucho más que la cantidad de átomos en el universo observable.

Arduino y el IoT

Capas



- Recolección de datos: adquisición, percepcción desde sensores, chips, RFID
- Almacenamiento: generalmente los dispositivos poseen poco almacenamiento y baja capacidad de procesamiento. La “nube” sería la solución
- Procesamiento Inteligente: analizar los datos almacenados y proveer servicios inteligentes. Monitoreo y Control
- Transmisión: desde sensores, etiquetas RFID o chips a dentros de datos, de centros de datos a unidades de procesamiento, de procesadores a controladores, equipos o usuarios finales
- Entrega: entrega de datos sin errores sin alteraciones.

Arduino y el IoT

Microcontroladores

- Sistema integrado: Son circuitos integrados (IC) que contienen todos los componentes esenciales de una computadora en un solo chip, incluyendo CPU, memoria, periféricos y puertos de entrada/salida.
- Propósito específico: Diseñados para realizar tareas específicas y repetitivas de manera eficiente, generalmente dentro de sistemas embebidos.
- Bajo costo y consumo de energía: Optimizados para el uso eficiente de recursos, lo que los hace ideales para aplicaciones con limitaciones de presupuesto y energía.
- Menor potencia de procesamiento: En comparación con los microcomputadores, tienen una capacidad de procesamiento limitada, suficiente para las tareas para las que están diseñados.
- Ejemplos: Controladores de motores, sensores, electrodomésticos inteligentes, juguetes electrónicos.

Microcomputadores

- Propósito general: Diseñados para una amplia gama de tareas y aplicaciones, ofreciendo mayor flexibilidad y capacidad de procesamiento.
- Arquitectura más compleja: Integran componentes más sofisticados, como unidades de punto flotante y coprocesadores, para un mayor rendimiento.
- Mayor potencia de procesamiento: Capaces de manejar tareas más complejas y procesar grandes cantidades de datos.
- Más costosos y con mayor consumo de energía: Su mayor complejidad se traduce en un mayor costo y consumo de energía.
- Ejemplos: Computadoras personales, tabletas, teléfonos inteligentes.

Arduino y el IoT

Fabricantes de Microcontroladores

Atmel

Texas Instruments

ARM

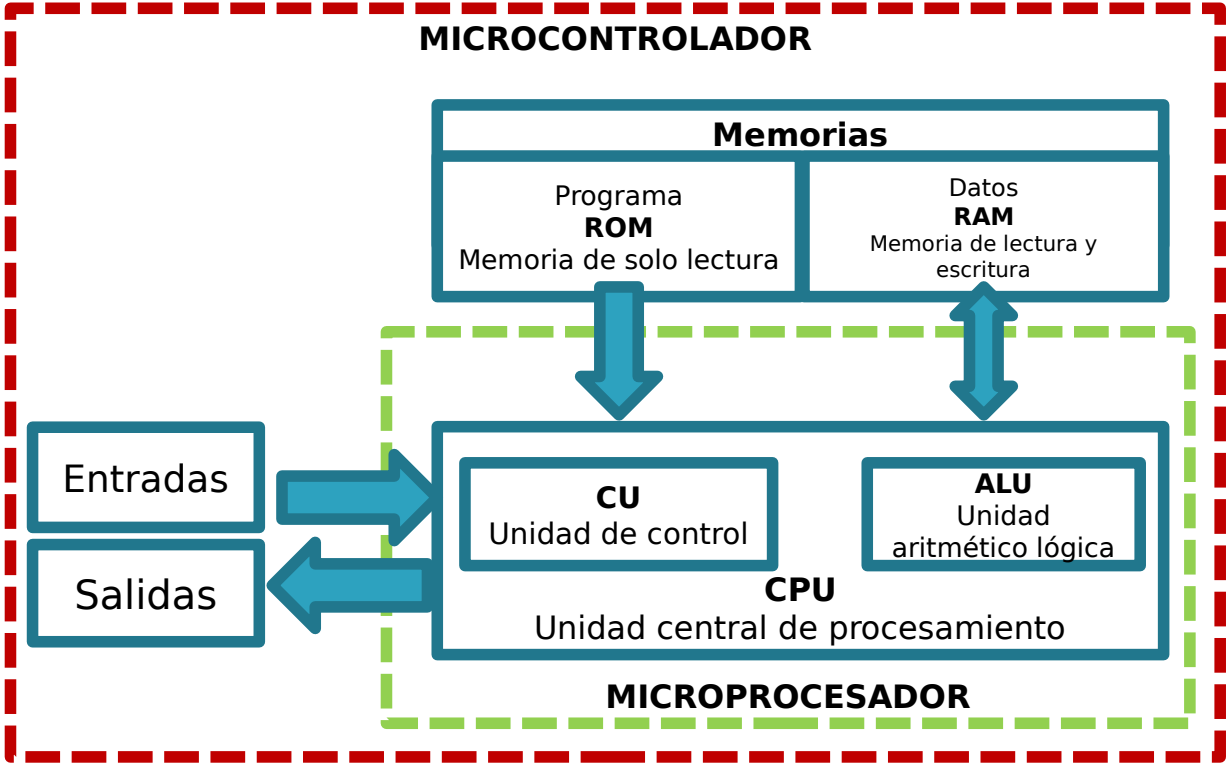
freescale

zilog

SIEMENS

Arduino y el IoT

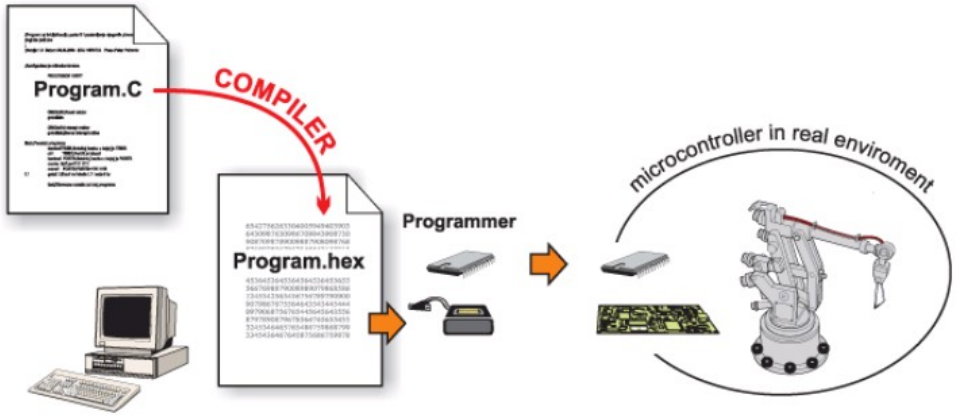
Característica	Microcontroladores	Microprocesadores
Tipo de sistema	Integrado	Modular
Propósito	Específico	General
Arquitectura	Más simple	Más compleja
CPU	Menos potente	Más potente
Memoria	Menos memoria integrada	Más memoria integrada
Periféricos	Periféricos integrados específicos para la aplicación	Periféricos más generales y configurables
E/S	Puertos de E/S limitados	Puertos de E/S más numerosos y flexibles
Costo	Bajo	Alto
Consumo de energía	Bajo	Alto
Ejemplos	Controladores de motores, electrodomésticos inteligentes, dispositivos embebidos	Computadoras personales, servidores, estaciones de trabajo

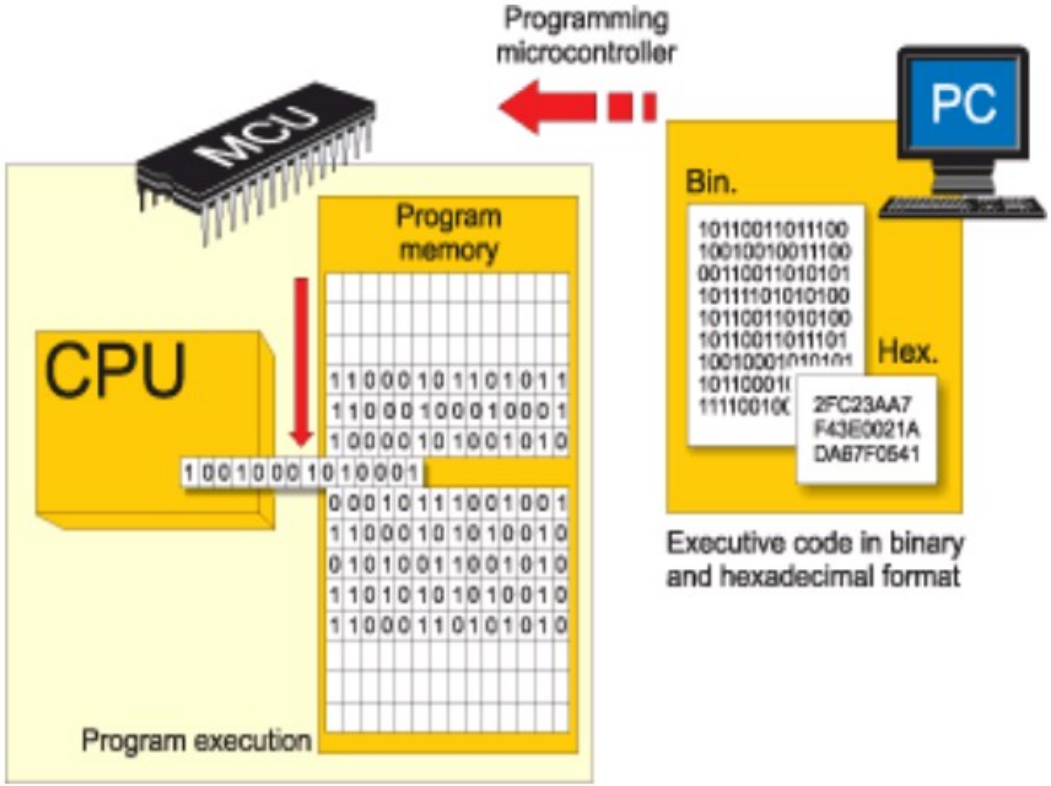


Arduino y el IoT

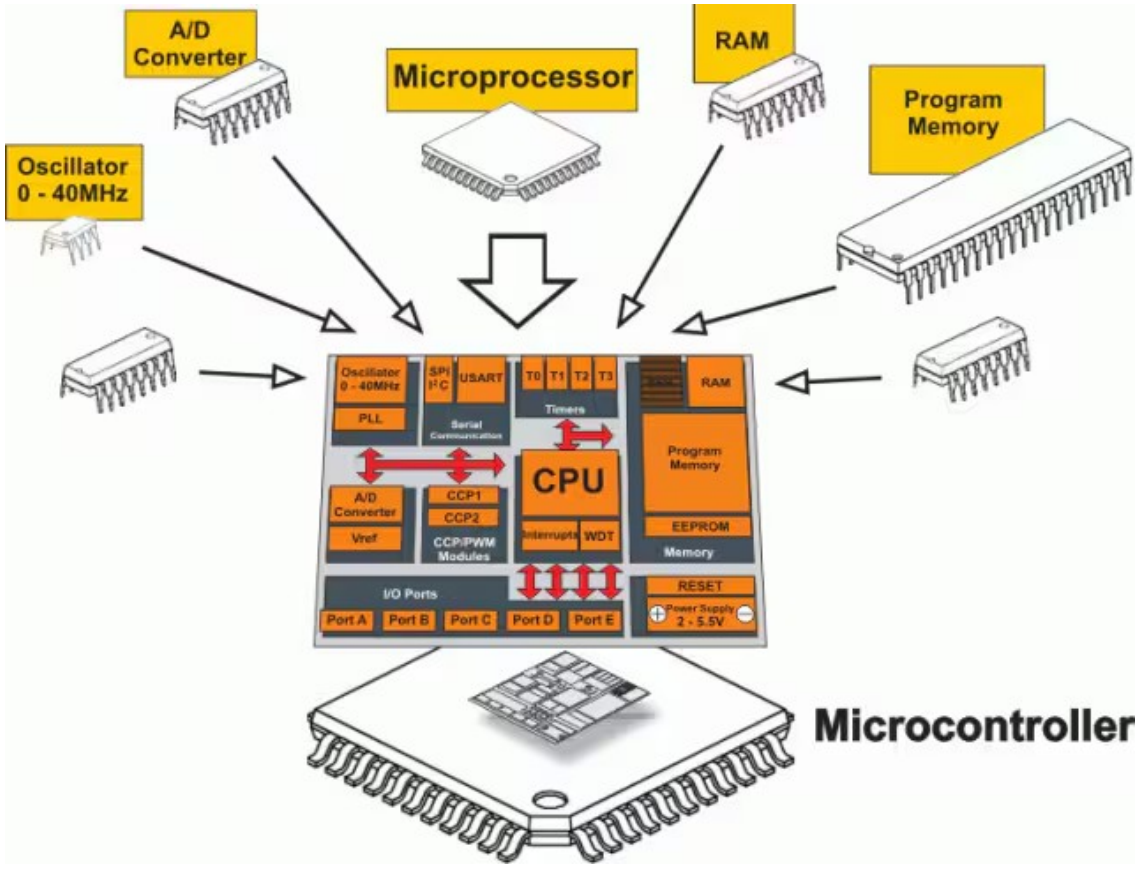
Programación de Microcontroladores

- Computador
- Programa de grabación
- Dispositivo de grabación





Arduino y el IoT

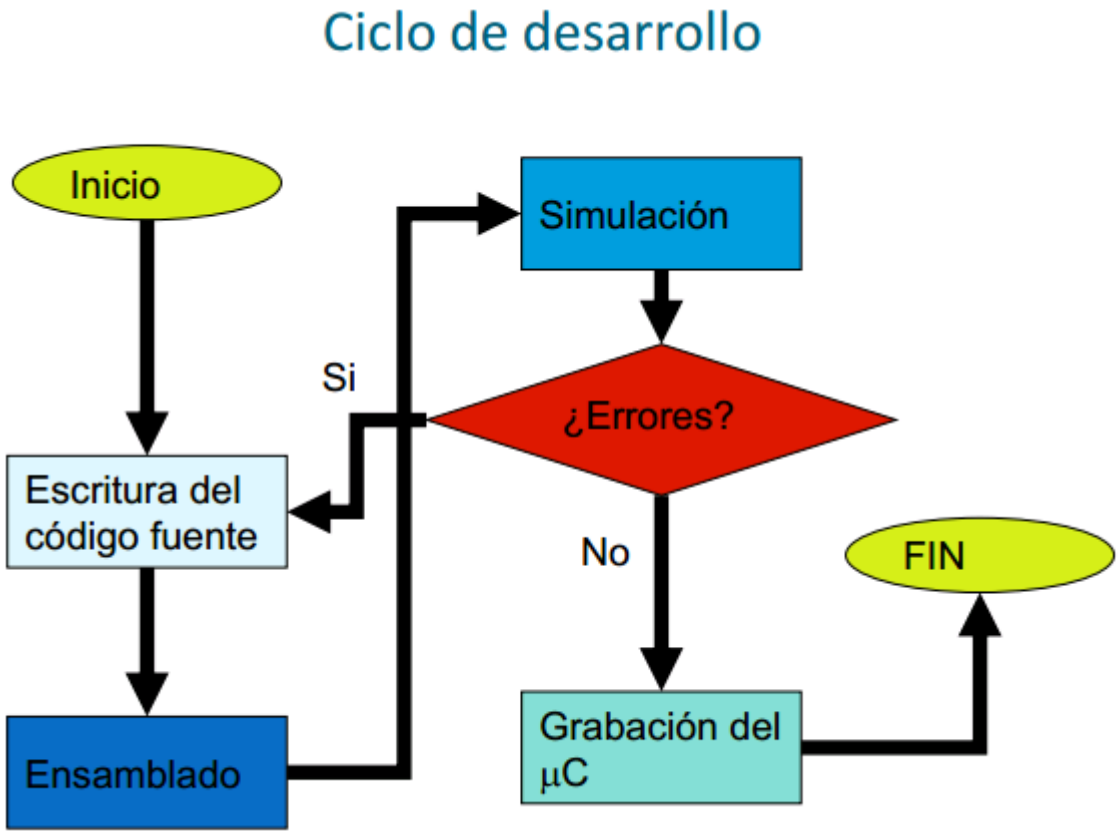


Arduino y el IoT

Dividir el problema en **micro-tareas**.

Por ejemplo para medir temperatura y mostrar valor en pantalla:

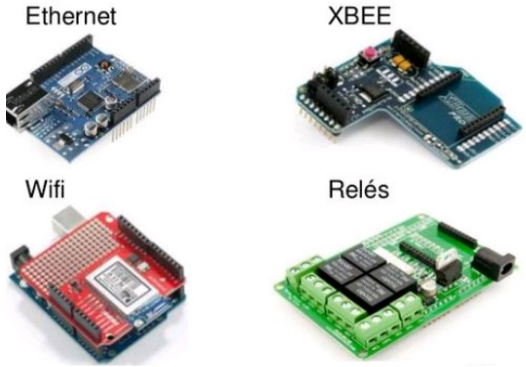
- Seleccionar el convertidor A/D
- Medir el valor analógico
- Calcular temperatura
- Enviar datos ajustados al formato LCD



Arduino y el IoT

Arduino

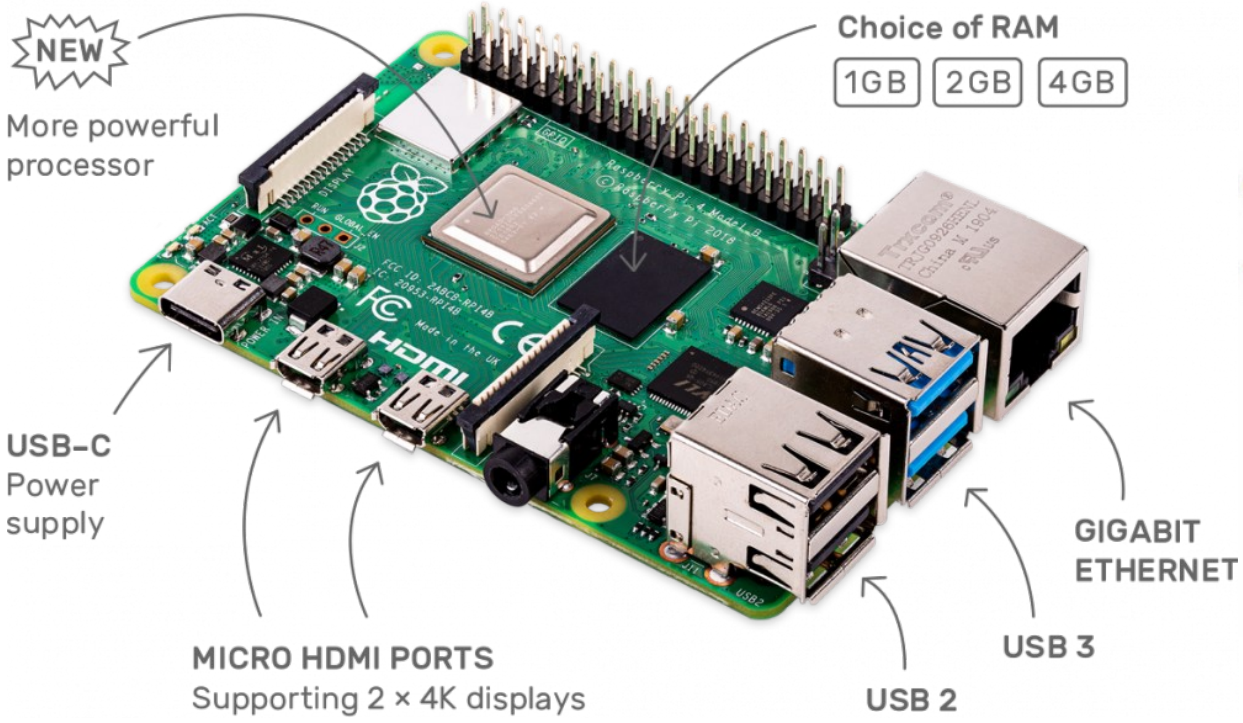
- Placa de desarrollo de hardware abierto
- Basada en un microcontrolador
- Cuenta con su propio IDE
- Tiene a disposición una amplia librería de software
- Permite la interconexión del mundo virtual con el mundo físico (mundo digital con mundo analógico)

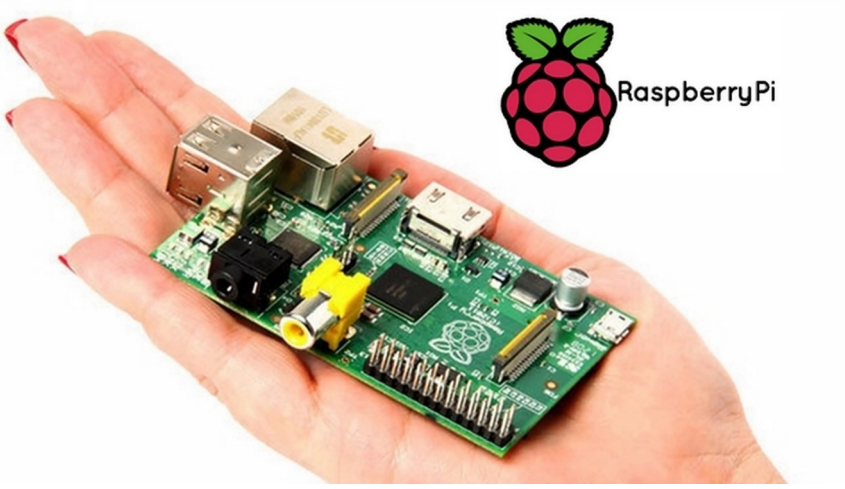


Arduino UNO Device

Hardware libre

Arduino y el IoT



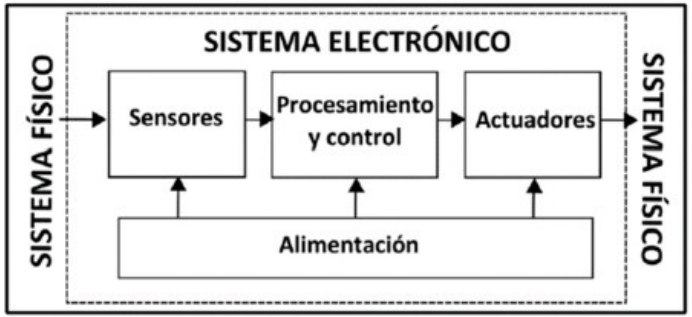


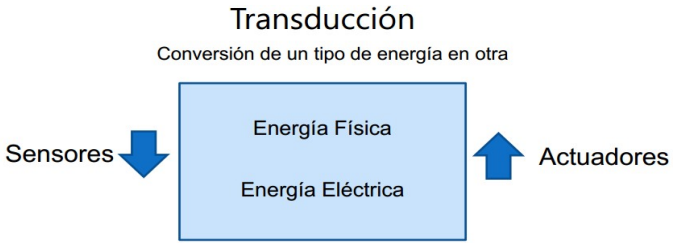
Arduino y el IoT

Sensores y Actuadores

Entre los dispositivos electrónicos utilizados en IoT tenemos aquellos que pueden "sentir" el entorno y son conocidos como **Sensores.** Un Sensor convierte las señales del medio ambiente en señales eléctricas.

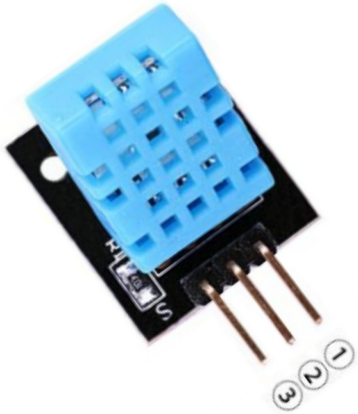
También tenemos dispositivos que pueden "actuar" con el mundo exterior, estos son conocidos como **Actuadores**. Un actuador convierte una señal eléctrica en otro tipo de energía asociado a una acción física.





Arduino y el IoT

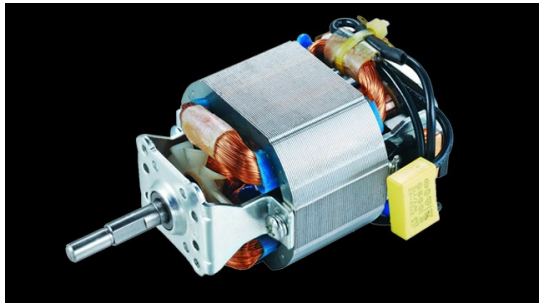
Sensores

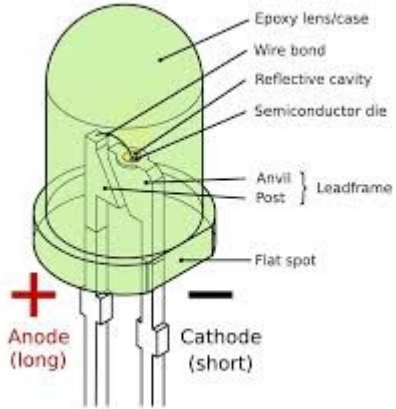


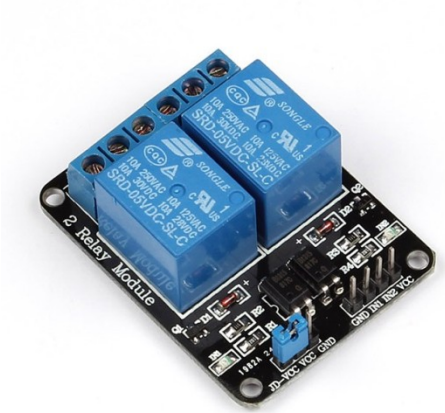
1.GND:ground
2.VCC: 3.3V-5V DC
3.OUTPUT

Arduino y el IoT

Actuadores







Comunicaciones

GPS

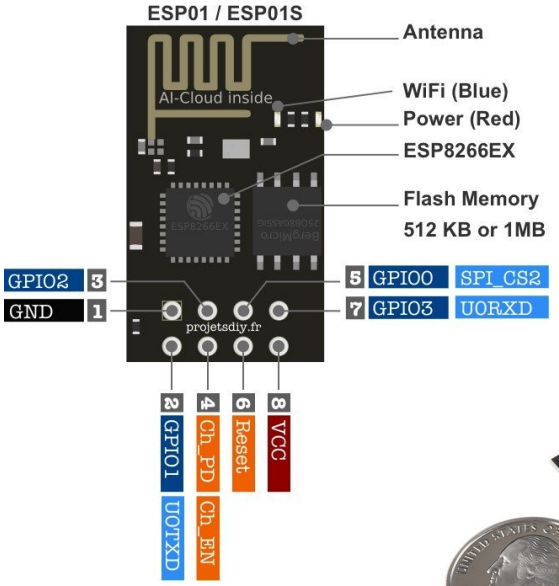
<u>Lectura de la sentencia GPRMC:</u>

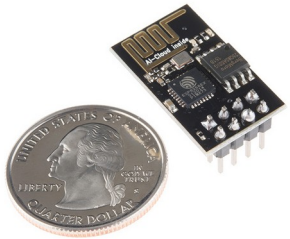
$GPRMC,194509.000,A,4042.6142,N,07400.4168,W,2.03,221.11,160412,,,A*77

RMC: recommended minimun
194509.00: tiemp actual GMT
A: status code, V (datos vacios o inválidos) A (activo)
4042.6142: Latitud 40º 42.6142 minutos DDMM.MMMM
N: latitud North
07400.4168: Longitud 074º 0,4168 minutos DDDMM.MMMM
W: longitud West
2.03: velocidad terrestre en nudos
221.11: tracking angulo
160412: fecha actual 16-04-2012
A*77: checksum

Arduino y el IoT

Comunicaciones





ESP01

Modulo ESP01
Implementación de un módulo de comunicaciones wifi usando el chip esp8266. Desarrollado por AI-Thinker.

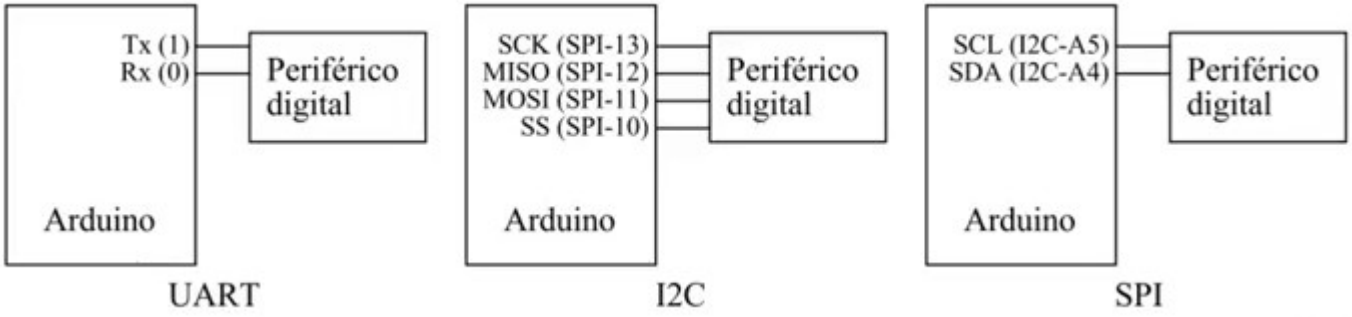
Imax de salida por pin 12 mA
Imax de entrada por pin 20 mA
Es importante recordar que estos equipos trabajan con 3,3 V.

Pines:

- Vcc debe estar a 3,3V
- GND a GND (conexión a tierra)
- RX (pin de recepción serial)
- Tx (pin de transmisión serial)
- CH_PD (chip enable)
- RST (pin de reset)
- GPIO0, GPIO2 (pines de entrada y salida, el pin **GPIO0** controla el modo de operación del esp, es decir modo programación o modo operación, cuando **se coloca 0V se está en modo programación**)

Arduino y el IoT

Comunicación Digital

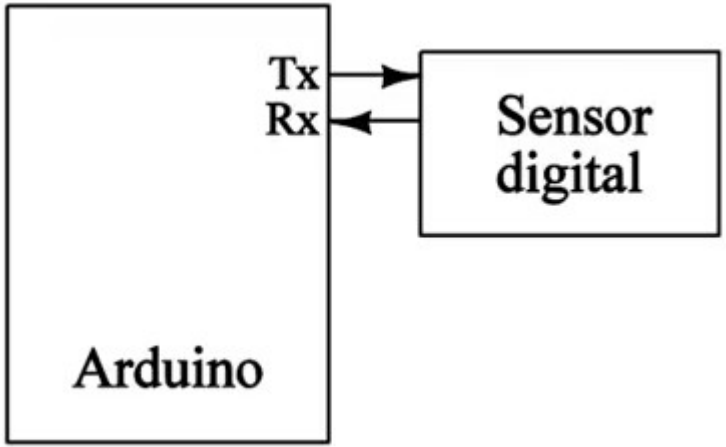


Arduino y el IoT

- **UART (***Universal Asynchronous Receiver-Transmitter***)**

* Tasa y formato configurable
* Trabajan en modo full-dúplex
* Como es comunicación asíncrona no requiere reloj

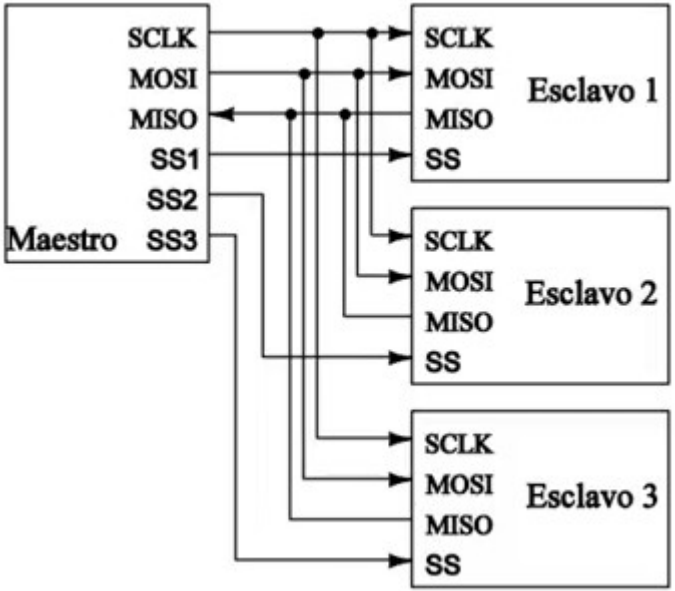
En Arduino al usar la consola serial se usa comunicación UART



Arduino y el IoT

- **SPI (***Serial Peripheral Interface***)**

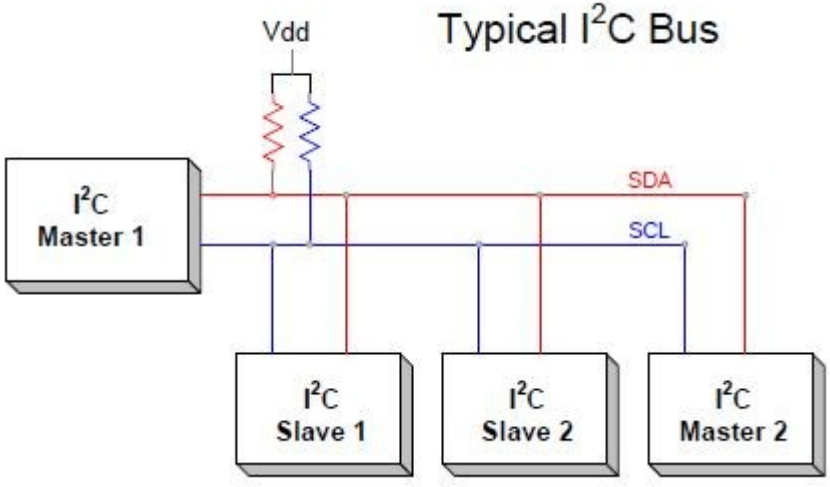
* Trabaja en paradigma maestro-esclavo
* Trabajan en modo full-dúplex
* Usa tres pines (SCLK, MOSI, MISO) y pines adicionales dependiendo de la cantidad de periféricos esclavos



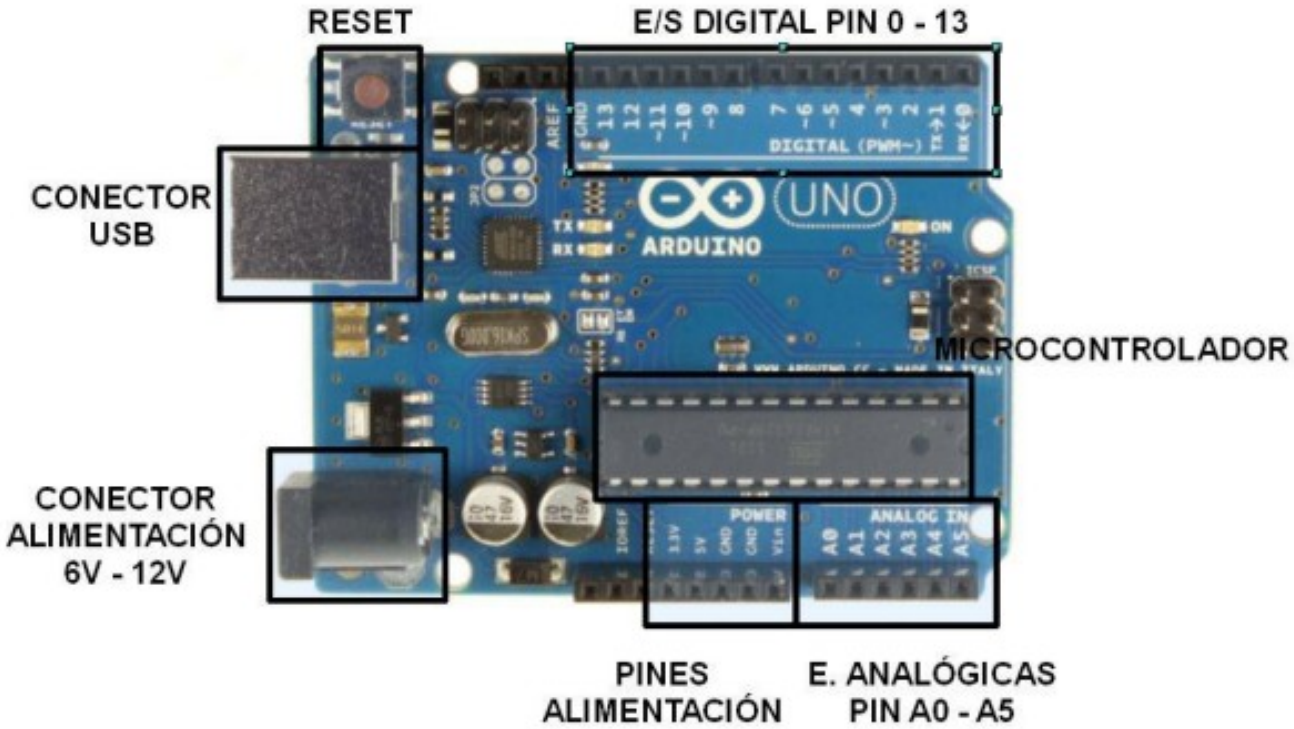
Arduino y el IoT

- **I2C** *(Inter-Integrated Circuit)*

* utiliza dos líneas de datos SDA (datos en serie) y SCL (reloj en serie)
* cada dispositivo en el bus tiene una dirección única de 7 bits que utiliza el master para identificarlo
* se pueden conectar varios dispositivos en el mismo bus



Arduino y el IoT



ESPECIFICACIONES TECNICAS

- Microcontrolador ATmega328
- Voltaje de operación 5V
- Voltaje de entrada (recomendado) 7-12 V
- Voltaje de entrada (limite) 6-20 V
- **Digital I/O Pins 14**
- **Canales PWM 6**
- **Entradas analógicas (6 pines)**
- DC corriente I/O Pin 40 mA
- DC corriente 3.3V Pin 50 mA
- Memoria Flash 32 KB (ATmega328) con 0.5KB para el bootloader
- SRAM 2KB (ATmega328)
- EEPROM 1KB (ATmega328)
- Velocidad de reloj 16MHz
- ATmega 16U2 para comunicación USB

Arduino y el IoT

Memoria Flash:

Tamaño: La placa Arduino Uno tiene 32 KB de memoria Flash. El sketch de Arduino se carga en la memoria Flash mediante el IDE de Arduino. Una vez cargado, el sketch se ejecuta repetidamente por el microcontrolador.

SRAM (Memoria Estática de Acceso Aleatorio):

Tamaño: La placa Arduino Uno tiene 2 KB de SRAM. El sketch de Arduino puede declarar variables y utilizarlas para almacenar datos. Las variables se almacenan en la SRAM y están disponibles para el sketch durante su ejecución.

EEPROM (Memoria de Solo Lectura Borrable Electrically Programable):

Tamaño: La placa Arduino Uno tiene 1 KB de EEPROM. El sketch de Arduino puede leer y escribir datos en la EEPROM utilizando funciones específicas. La EEPROM es útil para almacenar datos que no cambian con frecuencia.

ARDUINO CHEAT SHEET

Content for this Cheat Sheet provided by Gavin from Robots and Dinosaurs.
For more information visit: http://arduino.cc/en/Reference/Extended

sparkfun ELECTRONICS

Structure
void **setup**() void **loop**()

Control Structures
if (x<5){ } **else** { }
switch (myvar) {
  case 1:
  break;
  case 2:
  break;
  default:
}
for (int i=0; i <= 255; i++){ }
while (x<5){ }
do { } **while** (x<5);
continue; //Go to next in do/for/while loop
return x; // Or 'return;' for voids.
goto // considered harmful :-)

Further Syntax
// (single line comment)
/* (multi-line comment) */
#define DOZEN 12 //Not baker's!
#include <avr/pgmspace.h>

General Operators
= (assignment operator)
+ (addition) - (subtraction)
* (multiplication) / (division)
% (modulo)
== (equal to) != (not equal to)
< (less than) > (greater than)
<= (less than or equal to)
>= (greater than or equal to)
&& (and) || (or) ! (not)

Pointer Access
& reference operator
* dereference operator

Bitwise Operators
& (bitwise and) | (bitwise or)
^ (bitwise xor) ~ (bitwise not)
<< (bitshift left) >> (bitshift right)

Compound Operators
++ (increment) -- (decrement)
+= (compound addition)
-= (compound subtraction)
*= (compound multiplication)
/= (compound division)
&= (compound bitwise and)
|= (compound bitwise or)

Constants
HIGH | LOW
INPUT | OUTPUT
true | false
143 // **Decimal** number
0173 // **Octal** number
0b11011111 //**Binary**
0x7B // **Hex** number
7**U** // Force unsigned
10**L** // Force long
15**UL** // Force long unsigned
10.0 // Forces floating point
2.4e5 // 240000

Data Types
void
boolean (0, 1, false, true)
char (e.g. 'a' -128 to 127)
unsigned char (0 to 255)
byte (0 to 255)
int (-32,768 to 32,767)
unsigned int (0 to 65535)
word (0 to 655word (0 to 65535)
long (-2,147,483,648 to 2,147,483,647)
unsigned long (0 to 4,294,967,295)
float (-3.4028235E+38 to 3.4028235E+38)
double (currently same as float)
sizeof(myint) // returns 2 bytes

Strings
char S1[15];
char S2[8]={'a','r','d','u','i','n','o'};
char S3[8]={'a','r','d','u','i','n','o','\0'};
//Included \0 null termination
char S4[ ] = "arduino";
char S5[8] = "arduino";
char S6[15] = "arduino";

Arrays
int my**I**nts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};

Conversion
char() **byte**()
int() **word**()
long() **float**()

Qualifiers
static // persists between calls
volatile // use RAM (nice for ISR)
const // make read-only
PROGMEM // use flash

Digital I/O
pinMode(pin, [INPUT,OUTPUT])
digitalWrite(pin, value)
int **digitalRead**(pin)
//Write High to inputs to use pull-up res

Analog I/O
analogReference([DEFAULT, INTERNAL,EXTERNAL])
int **analogRead**(pin) //Call twice if switching pins from high Z source.
analogWrite(pin, value) // PWM

Advanced I/O
tone(pin, freqhz)
tone(pin, freqhz ,duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin, [MSBFIRST,LSBFIRST], value)
unsigned long **pulseIn**(pin,[HIGH,LOW])

Time
unsigned long **millis**() // 50 days overflow.
unsigned long **micros**() // 70 min overflow
delay(ms)
delayMicroseconds(us)

Math
min(x, y) **max**(x, y) **abs**(x)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
pow(base, exponent) **sqrt**(x)
sin(rad) **cos**(rad) **tan**(rad)

Random Numbers
randomSeed(seed) // Long or int
long **random**(max)
long **random**(min, max)

Bits and Bytes
lowByte()
highByte()
bitRead(x,bitn)
bitWrite(x,bitn,bit)
bitSet(x,bitn)
bitClear(x,bitn)
bit(bitn) //bitn: 0-LSB 7-MSB

External Interrupts
attachInterrupt(interrupt, function, [LOW,CHANGE,RISING,FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()

Libraries:

Serial.
  begin([300, 1200, 2400, 4800, 9600,14400, 19200, 28800, 38400, 57600,115200])
  end()
  int **available**()
  int **read**()
  flush()
  print()
  println()
  write()

EEPROM (#include <EEPROM.h>)
byte **read**(intAddr)
  write(intAddr,myByte)

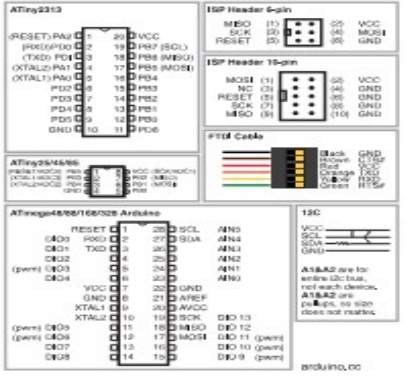
Servo (#include <Servo.h>)
  attach(pin , [min_uS, max_uS])
  write(angle) // 0-180
  writeMicroseconds(uS) //1000-2000,1500 is midpoint
  read() // 0-180
  attached() //Returns boolean
  detach()

SoftwareSerial(RxPin,TxPin)
  // #include<**SoftwareSerial.**h>
  begin(longSpeed) // up to 9600
  char **read**() // blocks till data
  print(myData) or **println**(myData)

Wire (#include <Wire.h>) // For **I2C**
  begin() // Join as master
  begin(addr) // Join as slave @ addr
  requestFrom(address, count)
  beginTransmission(addr) // Step 1
  send(mybyte) // Step 2
  send(char * mystring)
  send(byte * data, size)
  endTransmission() // Step 3
  byte **available**() // Num of bytes
  byte **receive**() //Return next byte
  onReceive(handler)
  onRequest(handler)

	ATMega168	ATMega328	ATMega1280
Flash (2k for bootloader)	16kB	32kB	128kB
SRAM	1kB	2kB	8kB
EEPROM	512B	1kB	4kB

	Duemilanove/ Nano/ Pro/ ProMini	Mega
# of IO	14 + 6 analog (Nano has 14 + 8)	54 + 16 analog
Serial Pins	0 - RX 1 - TX	0 - RX1 1 - TX1 19 - RX2 18 - TX2 17 - RX3 16 - TX3 15 - RX4 14 - TX4
Ext Interrupts	2 - (Int 0) 1 - (Int 1)	2,3,21,20,19,18 (IRQ0 - IRQ5)
PWM Pins	5,6 - Timer 0 9,10 - Timer 1 3,11 - Timer 2	0 - 13
SPI	10 - SS 11 - MOSI 12 - MISO 13 - SCK	53 - SS 51 - MOSI 50 - MISO 52 - SCK
I2C	Analog4 - SDA Analog5 - SCK	20 - SDA 21 - SCL



Arduino Programming Cheat Sheet

Primary source: Arduino Language Reference
http://arduino.cc/en/Reference/

Structure & Flow

Basic Program Structure

```
void setup() {
  // Runs once when sketch starts
}
void loop() {
  // Runs repeatedly
}
```

Control Structures

```
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
for (int i = 0; i < 10; i++) { ... }
break;     // Exit a loop immediately
continue;  // Go to next iteration
switch (var) {
  case 1:
    ...
    break;
  case 2:
    ...
    break;
  default:
    ...
}
return x;  // x must match return type
return;    // For void return type
```

Function Definitions

<ret. type> <name>(<params>) { ... }
e.g. `int double(int x) {return x*2;}`

Operators

General Operators

=	assignment		
+	add	-	subtract
*	multiply	/	divide
%	modulo		
==	equal to	!=	not equal to
<	less than	>	greater than
<=	less than or equal to		
>=	greater than or equal to		
&&	and	||	or
!	not		

Compound Operators

++	increment
--	decrement
+=	compound addition
-=	compound subtraction
*=	compound multiplication
/=	compound division
&=	compound bitwise and
|=	compound bitwise or

Bitwise Operators

&	bitwise and	|	bitwise or
^	bitwise xor	~	bitwise not
<<	shift left	>>	shift right

Pointer Access

& reference: get a pointer
* dereference: follow a pointer

Built-in Functions

Pin Input/Output

Digital I/O - pins 0-13 A0-A5

```
pinMode(pin,
  [INPUT, OUTPUT, INPUT_PULLUP])
int digitalread(pin)
digitalWrite(pin, [HIGH, LOW])
```

Analog In - pins A0-A5

```
int analogRead(pin)
analogReference(
  [DEFAULT, INTERNAL, EXTERNAL])
```

PWM Out - pins 3 5 6 9 10 11

```
analogWrite(pin, value)
```

Advanced I/O

```
tone(pin, freq_Hz)
tone(pin, freq_Hz, duration_ms)
noTone(pin)
shiftOut(dataPin, clockPin,
  [MSBFIRST, LSBFIRST], value)
unsigned long pulseIn(pin,
  [HIGH, LOW])
```

Time

```
unsigned long millis()
  // Overflows at 50 days
unsigned long micros()
  // Overflows at 70 minutes
delay(msec)
delayMicroseconds(usec)
```

Math

```
min(x, y)   max(x, y)   abs(x)
sin(rad)    cos(rad)    tan(rad)
sqrt(x)     pow(base, exponent)
constrain(x, minval, maxval)
map(val, fromL, fromH, toL, toH)
```

Random Numbers

```
randomSeed(seed) // long or int
long random(max) // 0 to max-1
long random(min, max)
```

Bits and Bytes

```
lowByte(x)    highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn)  // bitn: 0=LSB 7=MSB
```

Type Conversions

```
char(val)       byte(val)
int(val)        word(val)
long(val)       float(val)
```

External Interrupts

```
attachInterrupt(interrupt, func,
 [LOW, CHANGE, RISING, FALLING])
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Libraries

Serial - comm. with PC or via RX/TX

```
begin(long speed)  // Up to 115200
end()
int available()  // #bytes available
int read()   // -1 if none available
int peek()   // Read w/o removing
flush()
print(data)    println(data)
write(byte)    write(char * string)
write(byte * data, size)
SerialEvent()  // Called if data rdy
```

SoftwareSerial.h - comm. on any pin

```
SoftwareSerial(rxPin, txPin)
begin(long speed)  // Up to 115200
listen()       // Only 1 can listen
isListening()  // at a time.
read, peek, print, println, write
  // Equivalent to Serial library
```

EEPROM.h - access non-volatile memory

```
byte read(addr)
write(addr, byte)
EEPROM[index]  // Access as array
```

Servo.h - control servo motors

```
attach(pin, [min_uS, max_uS])
write(angle)  // 0 to 180
writeMicroseconds(uS)
   // 1000-2000; 1500 is midpoint
int read()    // 0 to 180
bool attached()
detach()
```

Wire.h - I²C communication

```
begin()     // Join a master
begin(addr) // Join a slave @ addr
requestFrom(address, count)
beginTransmission(addr) // Step 1
send(byte)              // Step 2
send(char * string)
send(byte * data, size)
endTransmission()       // Step 3
int available() // #bytes available
byte receive()  // Get next byte
onReceive(handler)
onRequest(handler)
```

Variables, Arrays, and Data

Data Types

boolean	true | false
char	-128 - 127, 'a' '$' etc.
unsigned char	0 - 255
byte	0 - 255
int	-32768 - 32767
unsigned int	0 - 65535
word	0 - 65535
long	-2147483648 - 2147483647
unsigned long	0 - 4294967295
float	-3.4028e+38 - 3.4028e+38
double	currently same as float
void	i.e., no return value

Strings

```
char str1[8] =
  {'A','r','d','u','i','n','o','\0'};
  // Includes \0 null termination
char str2[8] =
  {'A','r','d','u','i','n','o'};
  // Compiler adds null termination
char str3[] = "Arduino";
char str4[8] = "Arduino";
```

Numeric Constants

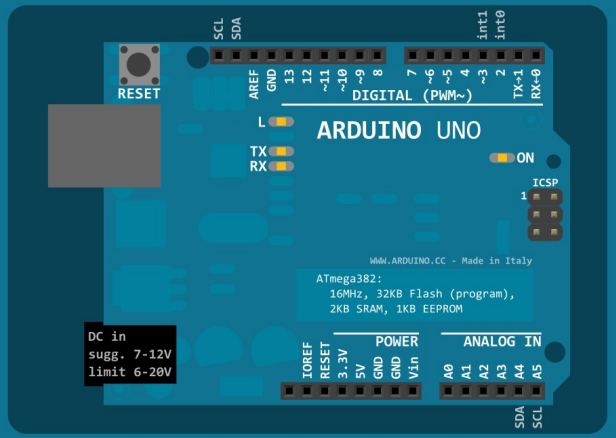
123	decimal
0b01111011	binary
0173	octal - base 8
0x7B	hexadecimal - base 16
123U	force unsigned
123L	force long
123UL	force unsigned long
123.0	force floating point
1.23e6	1.23*10^6 = 1230000

Qualifiers

static	persists between calls
volatile	in RAM (nice for ISR)
const	read-only
PROGMEM	in flash

Arrays

```
int myPins[] = {2, 4, 8, 3, 6};
int myInts[6];    // Array of 6 ints
myInts[0] = 42;   // Assigning first
                  // index of myInts
myInts[6] = 12;   // ERROR! Indexes
                  // are 0 though 5
```



by Mark Liffiton

Adapted from:
- Original: Gavin Smith
- SVG version: Frederic Dufourg
- Arduino board drawing: Fritzing.org

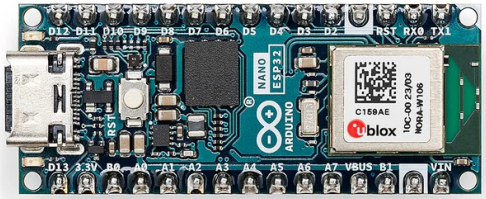
Arduino y el IoT

Razones para el uso de Arduino en proyectos IoT

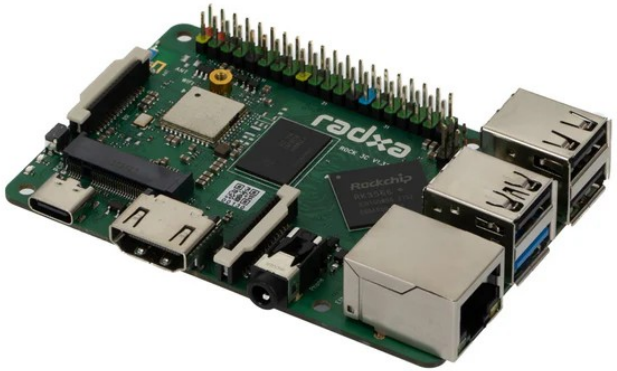
- Facilidad de uso
- Versatilidad
- Bajo costo
- Comunidad de desarrolladores extensa
- Disponibilidad de herramientas y dispositivos
- Código abierto
- Propósito general

Arduino y el IoT

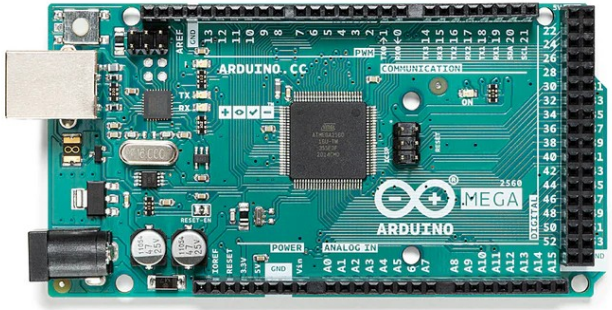
Arduino Nano ESP32



Rock 3 Model C 2GB



Arduino Mega 2560 Rev3



Arduino y el IoT

Portenta H7 Lite Connected está diseñado para proporcionar la potencia computacional del Portenta H7 para **aplicaciones de inteligencia artificial y soluciones de control de baja latencia**, con un módulo que se vuelve más rentable al eliminar la función de interfaz de video de alta resolución.

Portenta H7 Lite Connected ejecuta simultáneamente código de alto nivel y tareas en tiempo real gracias a sus dos procesadores. Por ejemplo, **puede ejecutar código compilado por Arduino y MicroPython al mismo tiempo**, y hacer que los dos núcleos se comuniquen a través de un mecanismo de llamada a procedimiento remoto que permite llamar funciones en el otro procesador sin problemas para realizar múltiples tareas reales. **Sus algoritmos de aprendizaje automático ahora pueden ejecutarse simultáneamente junto con la interacción sensor/actuador de baja latencia.**

Portenta H7 Lite Connected

SKU ABX00046 - **Barcode** 7630049202511 - Show more

$106.80

Arduino y el IoT

Objetivos del Taller:

1) Explicar el funcionamiento de la placa Arduino UNO
2) Mostrar el uso del IDE Arduino y presentar los pasos a seguir para la creación y la compilación del código
3) Conectar un sensor de temperatura y humedad DHT22 al Arduino
4) Programar una alerta ante subidas de temperatura
5) Conectar Arduino UNO al módulo de comunicación inalámbrico ESP01 para envío de datos a un servidor (Raspberry pi)
6) Configurar un servidor API Restful en el Raspberry pi
7) Configurar un script de envio de alertas en el servidor utilizando el bot de telegram