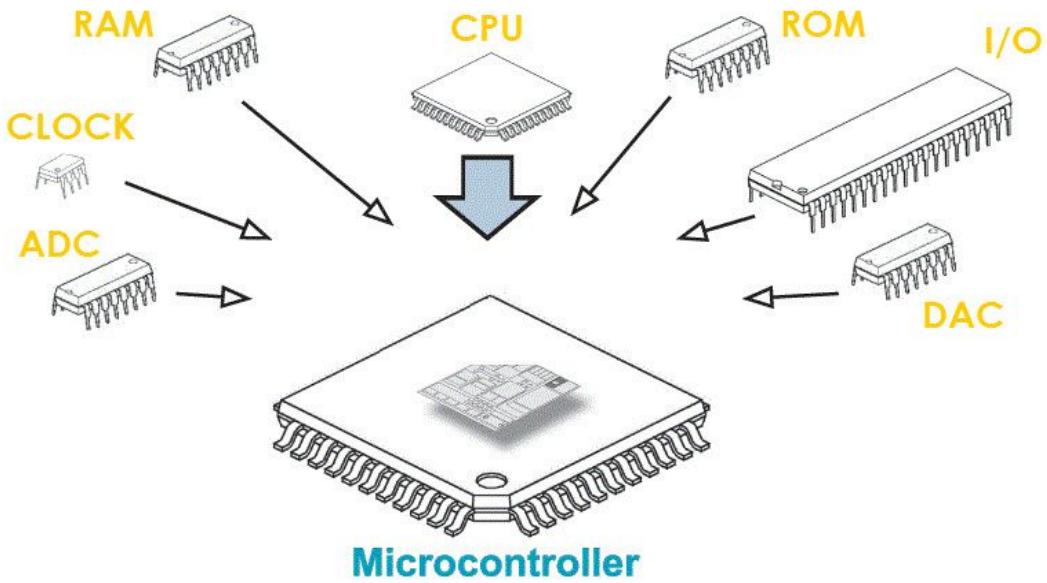
CLASE 11
SISTEMAS EMBEBIDOS

SISTEMAS EMBEBIDOS

1. Configuración mínima de un microcontrolador
2. Grabación de la memoria de programa
3. Bootloader
4. Fuses
5. Módulos de comunicación
6. Programación en alto nivel
7. RTOS

https://es.wikipedia.org/wiki/Sistema_embebido
Aunque no parezca a primera vista, todos estos temas están relacionados entre sí.
En la cátedra utilizamos un kit, pero eso no es un embedded, es un kit, que es un tipo de embedded pero con fines educativos.

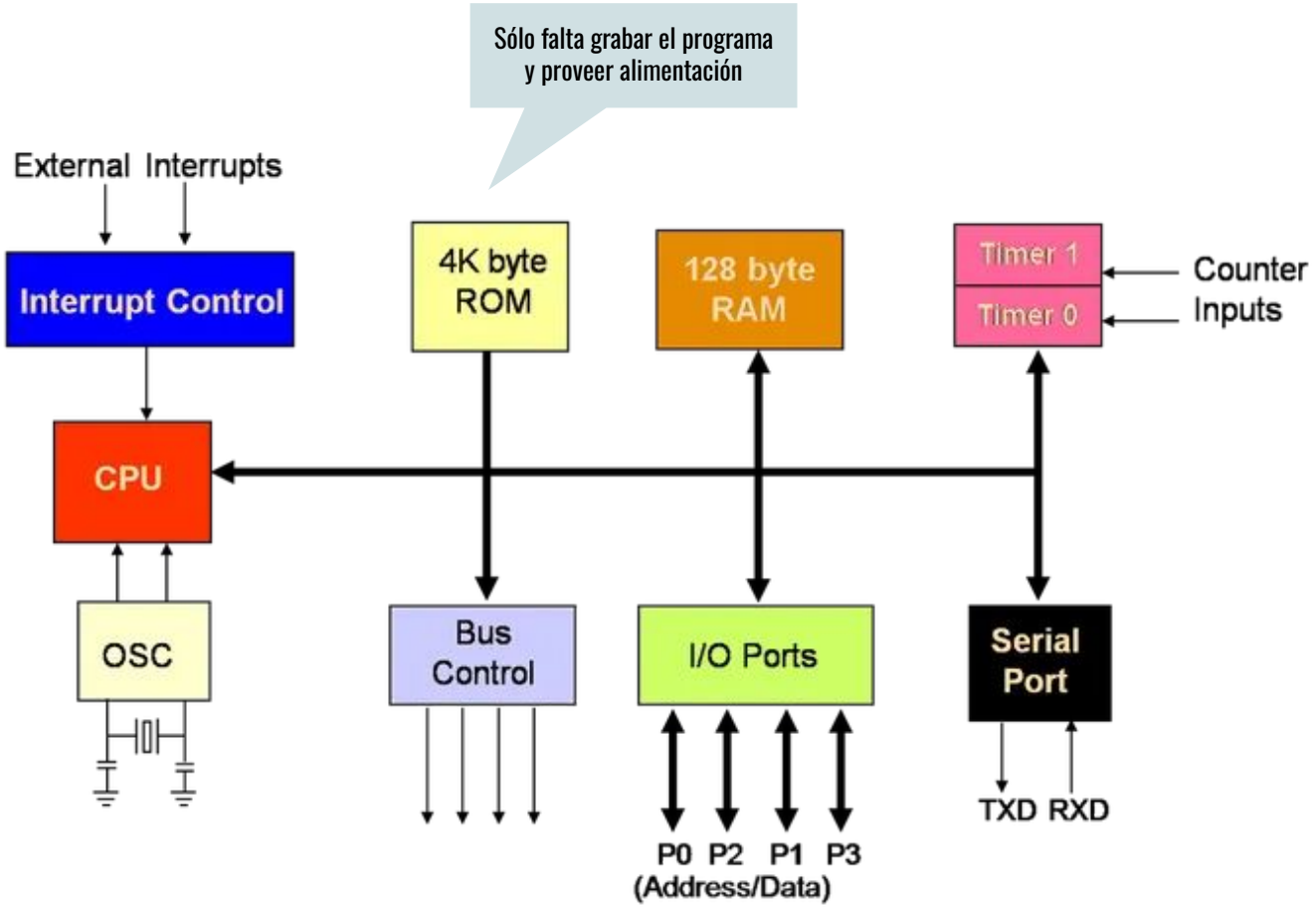


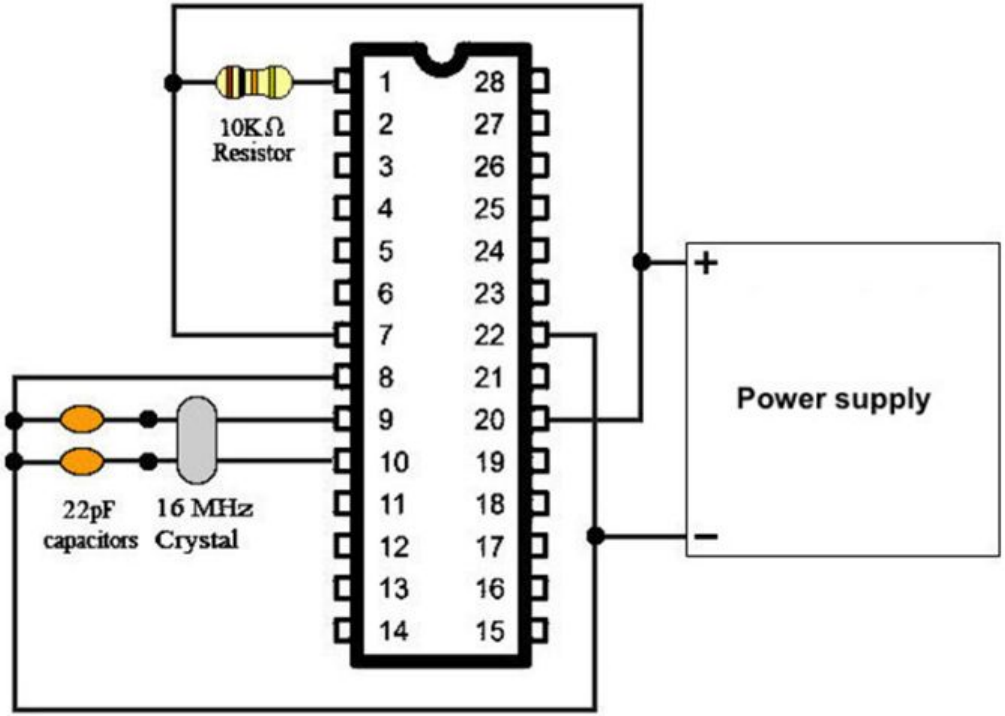
Single-chip (VLSI), sistemas embebidos (embedded systems): aplicaciones de medida, control y display.

Diseño en función del costo, el consumo de potencia y el espacio, single supply.

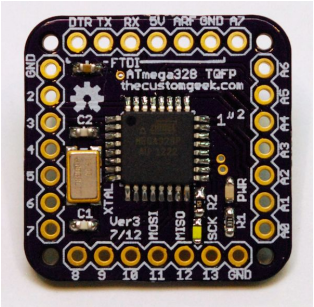
No son necesariamente diseños “simples”.

“Mixed signal” manejan también señales analógicas (Amplificadores, comparadores, ADC, DAC)


Sólo falta grabar el programa y proveer alimentación
External Interrupts
Interrupt Control
4K byte ROM
128 byte RAM
Timer 1
Timer 0
Counter Inputs
CPU
OSC
Bus Control
I/O Ports
Serial Port
TXD RXD
P0 P2 P1 P3
(Address/Data)


10KΩ
Resistor
22pF
capacitors
16 MHz
Crystal
Power supply
+
−
1
2
3
4
5
6
7
8
9
10
11
12
13
14
28
27
26
25
24
23
22
21
20
19
18
17
16
15


UN PESO
PROVINCIAS DEL RIO DE LA PLATA
1993


DTR TX RX 5V ARF GND A7
FTDI
ATmega328 TQFP
thecustomgeek.com
C2
C1
XTAL
Ver3
7/12
MOSI
MISO
SCK
RST
PWR
LED
GND


CR2032


20 mm

¿Cómo se graba la memoria de programa?

La memoria de programa es tipo FLASH.
Se proveen mecanismos de grabación en paralelo y en serie
ICSP (In Chip Serial Programmer)

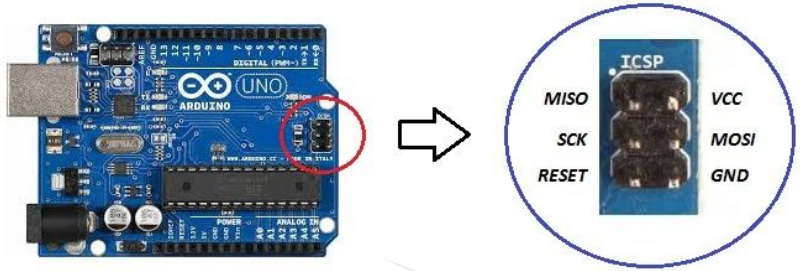
AVR Ver manual
ATmega328, Sección 28. Memory Programming. Pag. 289
Ver Lock-bits y Fuse-bits

No son protocolos estándar. Se necesita un grabador que reciba el .HEX desde la PC y realice la secuencia de grabación.

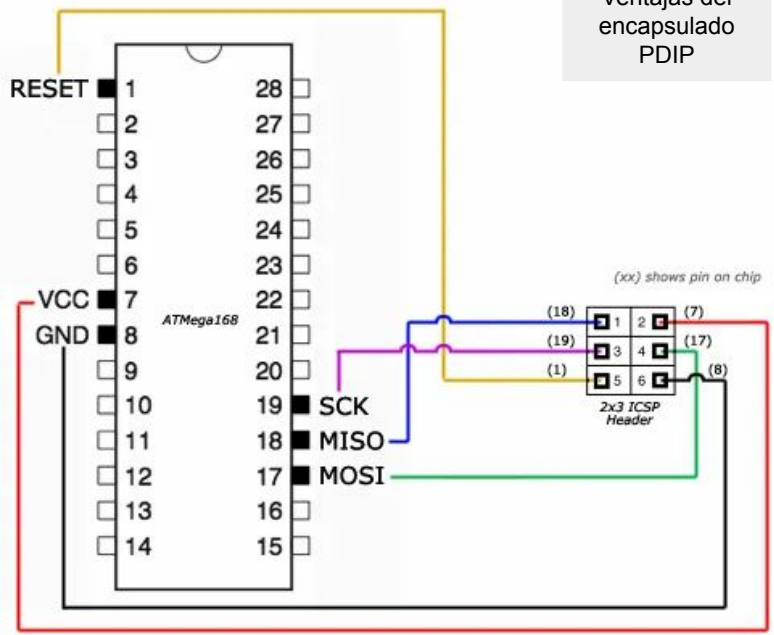
Si el microcontrolador recién llega de la fábrica no tiene grabado el bootloader. Hay que grabarlo por esta vía.

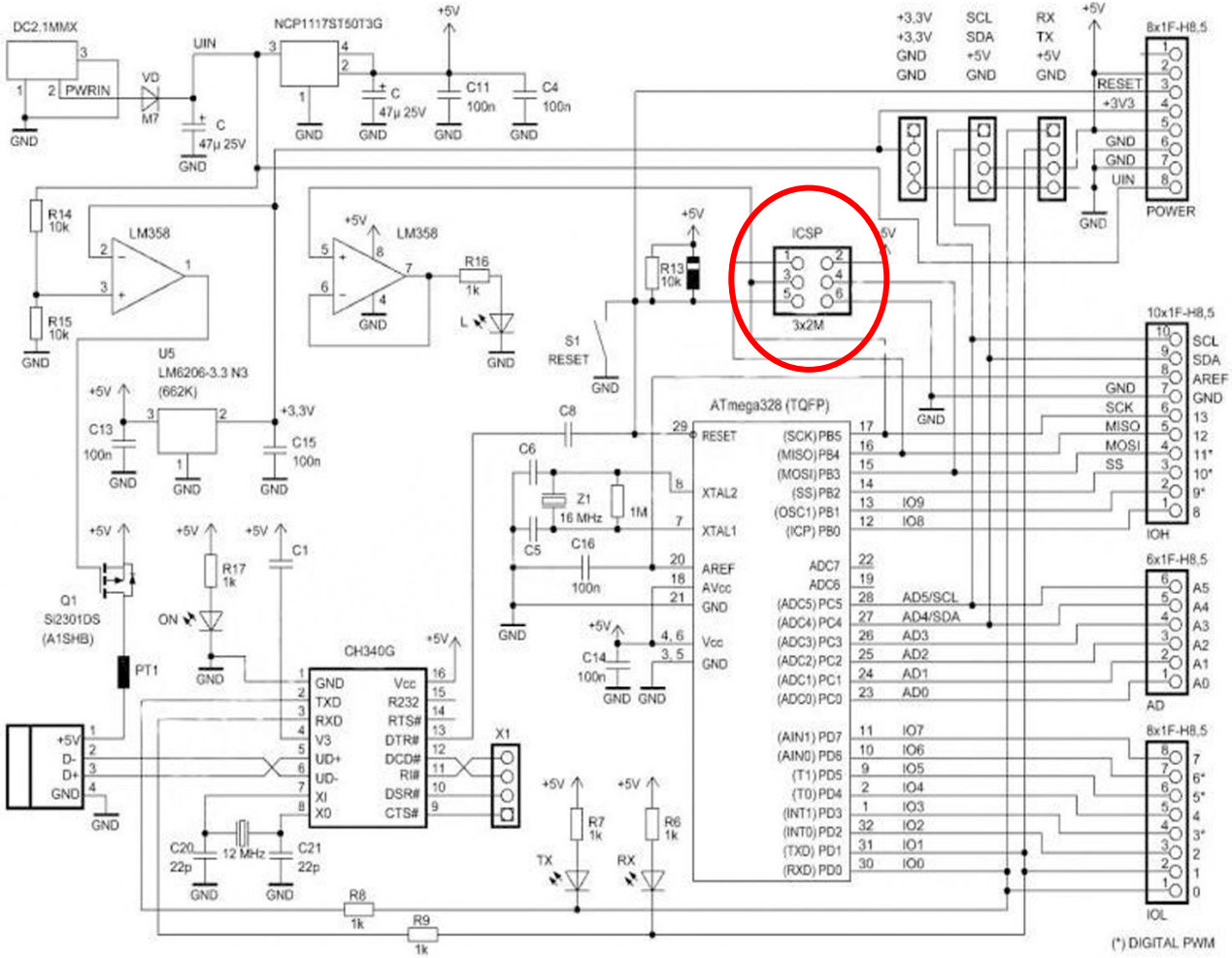
Interesante: se puede utilizar una placa Arduino para fabricar un ICSP.

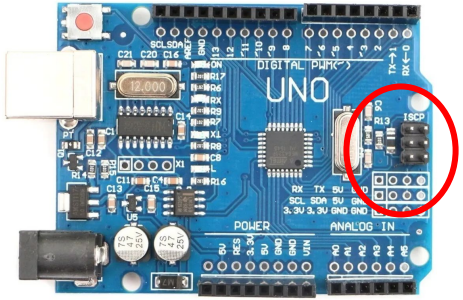
Serial programmer

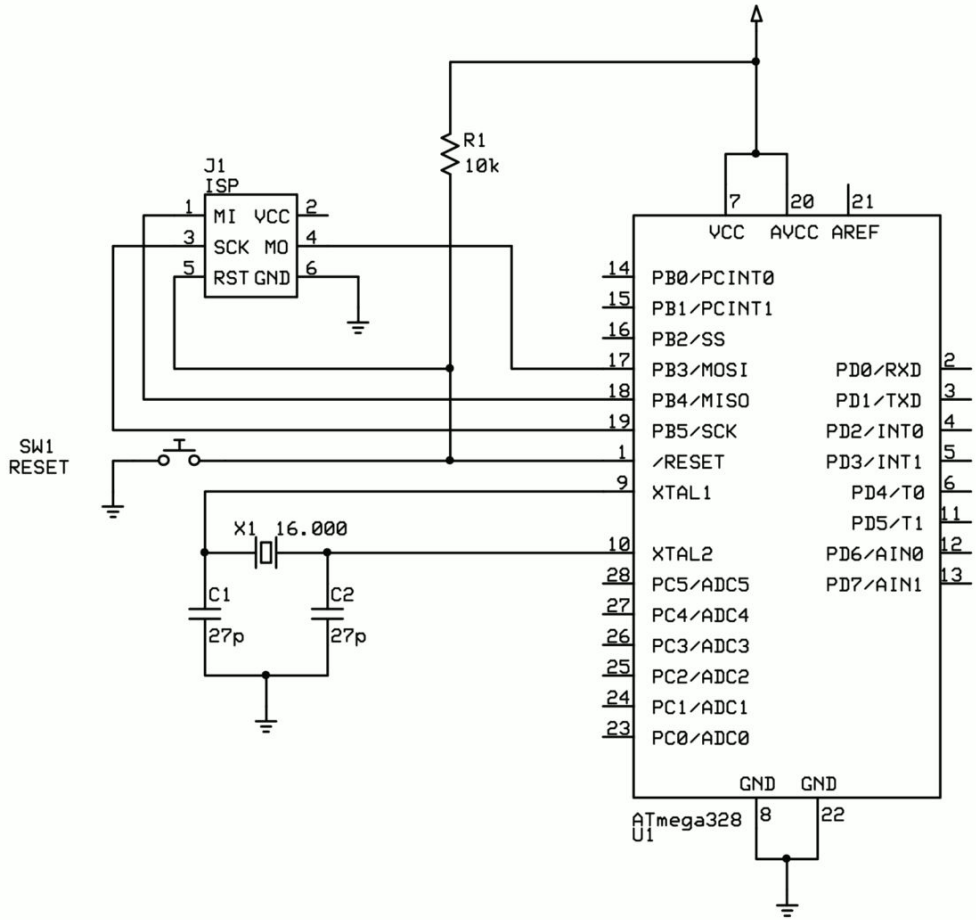


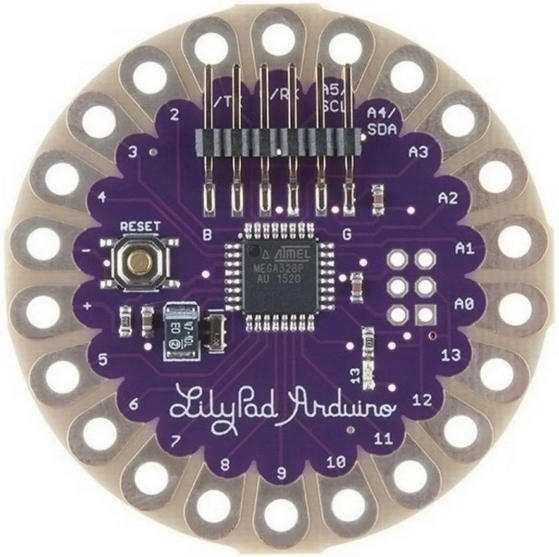
Ventajas del encapsulado PDIP

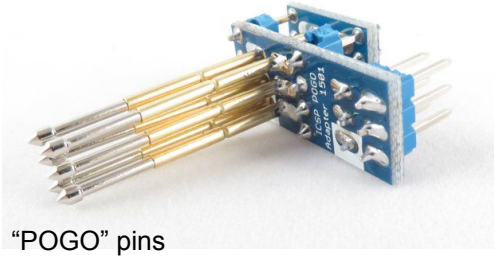



DC2.1MMX
PWRIN
VD
M7
UIN
C
47µ 25V
GND
NCP1117ST50T3G
+5V
C
47µ 25V
C11
100n
C4
100n
+3,3V
+3,3V
GND
GND
SCL
SDA
GND
RX
TX
+5V
+5V
GND
8x1F-H8,5
RESET
+3V3
GND
GND
UIN
POWER
R14
10k
LM358
R15
10k
GND
+5V
LM358
R16
1k
L
GND
R13
10k
+5V
ICSP
3x2M
5V
S1
RESET
GND
U5
LM6206-3.3 N3
(662K)
+5V
C13
100n
+3,3V
C15
100n
GND
GND
GND
10x1F-H8,5
SCL
SDA
AREF
GND
13
12
11*
10*
9*
8
IOH
ATmega328 (TQFP)
C8
C6
Z1
16 MHz
1M
C5
C16
100n
C14
100n
GND
GND
RESET
XTAL2
XTAL1
AREF
AVcc
GND
Vcc
GND
(SCK) PB5
(MISO) PB4
(MOSI) PB3
(SS) PB2
(OSC1) PB1
(ICP) PB0
ADC7
ADC6
(ADC5) PC5
(ADC4) PC4
(ADC3) PC3
(ADC2) PC2
(ADC1) PC1
(ADC0) PC0
(AIN1) PD7
(AIN0) PD6
(T1) PD5
(T0) PD4
(INT1) PD3
(INT0) PD2
(TXD) PD1
(RXD) PD0
IO9
IO8
AD5/SCL
AD4/SDA
AD3
AD2
AD1
AD0
IO7
IO6
IO5
IO4
IO3
IO2
IO1
IO0
GND
SCK
MISO
MOSI
SS
6x1F-H8,5
A5
A4
A3
A2
A1
A0
AD
8x1F-H8,5
7
6*
5*
4
3*
2
1
0
IOL
Q1
Si2301DS
(A1SHB)
PT1
+5V
R17
1k
ON
C1
GND
CH340G
GND
TXD
RXD
V3
UD+
UD-
XI
XO
Vcc
R232
RTS#
DTR#
DCD#
RI#
DSR#
CTS#
X1
+5V
D-
D+
GND
GND
C20
22p
12 MHz
C21
22p
GND
GND
R8
1k
R9
1k
+5V
R7
1k
+5V
R6
1k
TX
RX
(*) DIGITAL PWM

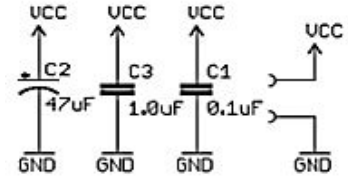

UNO
DIGITAL PWM~
POWER
ANALOG IN
ICSP

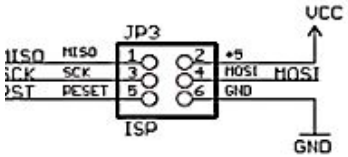

R1
10k
J1
ISP
MI VCC
SCK MO
RST GND
SW1
RESET
X1 16.000
C1
27p
C2
27p
7 20 21
VCC AVCC AREF
PB0/PCINT0
PB1/PCINT1
PB2/SS
PB3/MOSI
PB4/MISO
PB5/SCK
/RESET
XTAL1
XTAL2
PC5/ADC5
PC4/ADC4
PC3/ADC3
PC2/ADC2
PC1/ADC1
PC0/ADC0
PD0/RXD
PD1/TXD
PD2/INT0
PD3/INT1
PD4/T0
PD5/T1
PD6/AIN0
PD7/AIN1
GND GND
ATmega328
U1
8 22

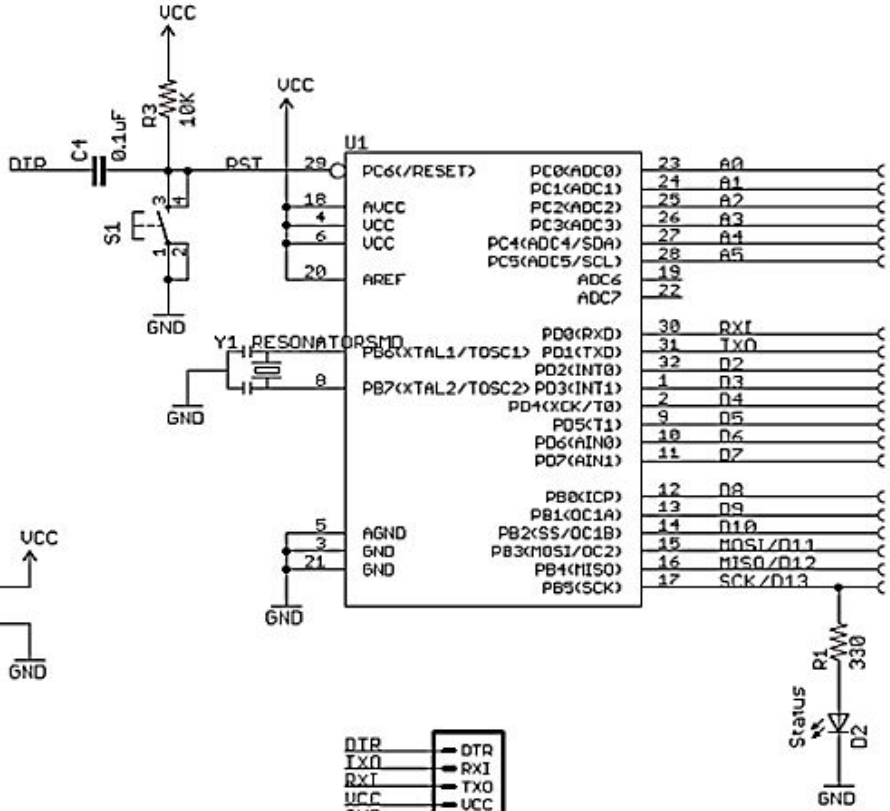

RESET
A4/
SDA
A3
A2
A1
A0
13
12
11
10
9
8
7
6
5
4
3
2
LilyPad Arduino



"POGO" pins


VCC
VCC
VCC
VCC
C2
47uF
C3
1.0uF
C1
0.1uF
GND
GND
GND
GND


JP3
VCC
MISO
SCK
RESET
+5
MOSI
GND
ISP
GND


VCC
VCC
R3
10K
0.1uF
C1
DTR
RST
S1
GND
U1
Y1 RESONATORSMD
GND
PC6(/RESET)
AVCC
VCC
VCC
AREF
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
ADC6
ADC7
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
AGND
GND
GND
GND
A0
A1
A2
A3
A4
A5
RXI
TXO
D2
D3
D4
D5
D6
D7
D8
D9
D10
MOSI/D11
MISO/D12
SCK/D13
R1
330
Status
D2
GND
DTR
TXO
RXI
VCC
GND
GND
DTR
RXI
TXO
VCC
CTS
GND
JP2

Intel HEX

https://es.wikipedia.org/wiki/HEX_(Intel)

El **Hexadecimal Object File Format** es un formato de archivo para la programación de microcontroladores, EPROMs y otros circuitos integrados. Es uno de los formatos más antiguos con esta finalidad.

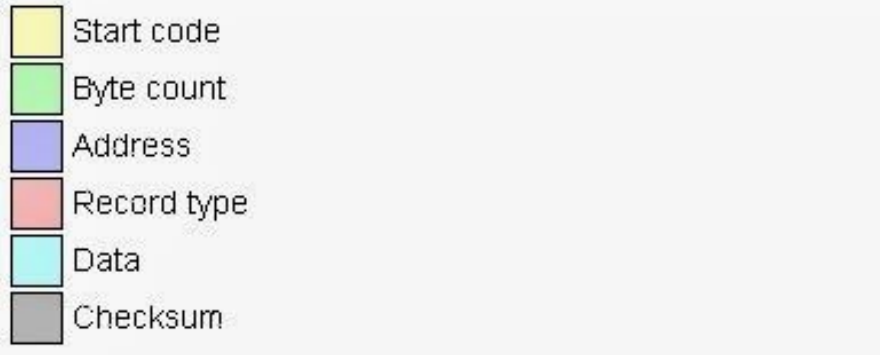
Consiste en un archivo de texto cuyas líneas contienen valores hexadecimales en ASCII que codifican los datos y su offset o dirección de memoria.

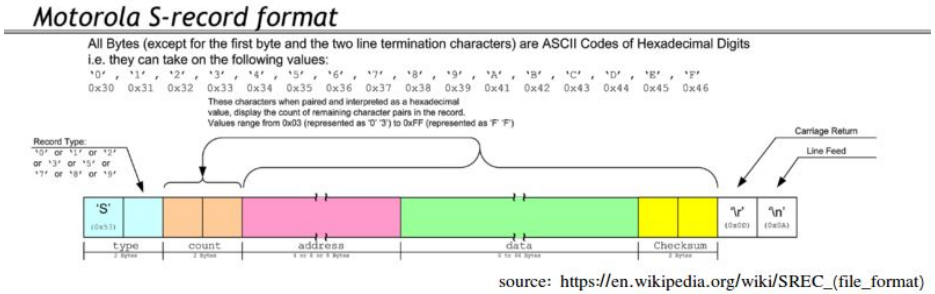
Existen otros formatos con el mismo objetivo, por ejemplo **S-record** de Motorola:

https://es.wikipedia.org/wiki/SREC

¿Cuál es la ventaja de que sea ASCII?

```
:10010000214601360121470136007EFE09D2190140
:100110002146017EB7C20001FF5F16002148011988
:10012000194E79234623965778239EDA3F01B2CAA7
:100130003F0156702B5E712B722B732146013421C7
:00000001FF
```





source: https://en.wikipedia.org/wiki/SREC_(file_format)

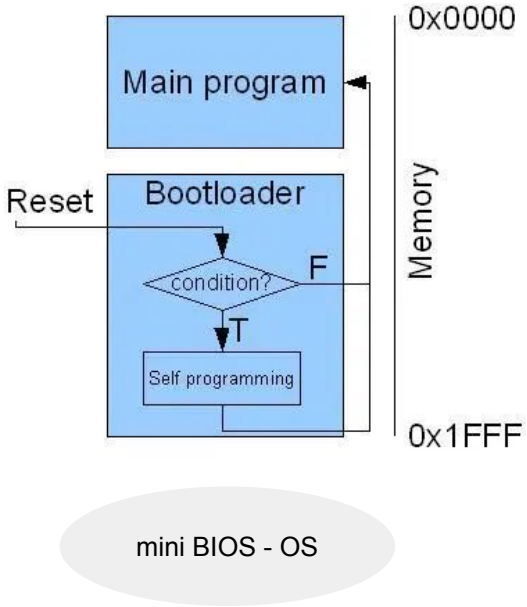
AVR Bootloader

Una solución ágil y de bajo costo para reprogramar un kit de desarrollo es grabar (una única vez) una sección de programa en la FLASH (bootloader), que reciba el programa principal por una vía estándar (por ejemplo un puerto serie) y lo grabe en la memoria FLASH de programa. Pero para eso se necesita disponer en el procesador de instrucciones que escriban la memoria de programa: en von Neumann no es problema, pero en Harvard es imposible. Entonces **Harvard modificado. IMPORTANTE.**

Las posteriores grabaciones se realizan enviando el programa en binario por el puerto serie al bootloader, quien se encarga de grabar la FLASH. **Esto es lo que hace nuestro kit.** Para ello tiene configurado un puerto serie con adaptador a USB, porque las PCs actuales no disponen de RS-232. Una ventaja de ésto es que, para uso posterior, queda implementada una vía de comunicación estándar a PC o a otro embedded.

El mecanismo para invocar el bootloader (no es más que una zona de memoria de programa, hay que modificar el PC para saltar) suele ser mantener en alto un pin determinado durante el Reset o enviar un determinado código por la puerta serie.

El bootloader podría recibir el programa por otra vía, por ejemplo una red inalámbrica (lo que suele llamarse *over-the-air programming*). El diseño del bootloader es importante si se desean implementar **actualizaciones de firmware** en un equipo.



https://www.digikey.com/en/maker/projects/build-your-own-arduino-breadboard/f243b09293ae4e3189bda47a821bb97a
https://scienceprog.com/usb-bootloaders-for-avr-microcontrollers/

AVR Bootloader

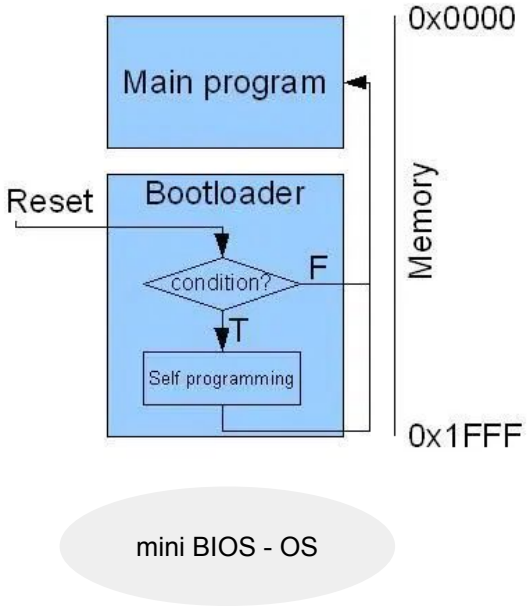
All bootloaders are simply pieces of software, just like the application code they will help you load on to the processor. Generally speaking, bootloaders are designed to be as small and simple as possible so they do not take up space that otherwise could be used for your application.

On reset, a processor will always start executing the code located at a particular memory location. Normally in an embedded system, if you don't have a bootloader you would just put your application at this startup memory location. With a bootloader, instead of the application starting at the memory location that the processor first executes after a reset, the bootloader is put in that spot and the application code in another part of memory. This means that the bootloader is the program that the processor will run every time it resets.

The job of the bootloader then, is to do one of two things: 1. Replace the existing application code with a new application, or 2. Start the existing application. How it determines which of these two things to do varies by implementation, but generally some external signal (such as a command on the serial port or a particular I/O line being pulled low) will tell it to load a new application from the communication port (or some other location) and write it to the application memory.

If the signal to load a new application is not present, the bootloader simply runs the existing application by executing a jump instruction to tell the processor to run the code at the memory location where the application starts.

https://aprendiendoarduino.wordpress.com/2016/11/06/icsp/



AVR Bootloader

1. Cambiar Reset Vector (manual pag. 277) y elegir el tamaño - Fuses

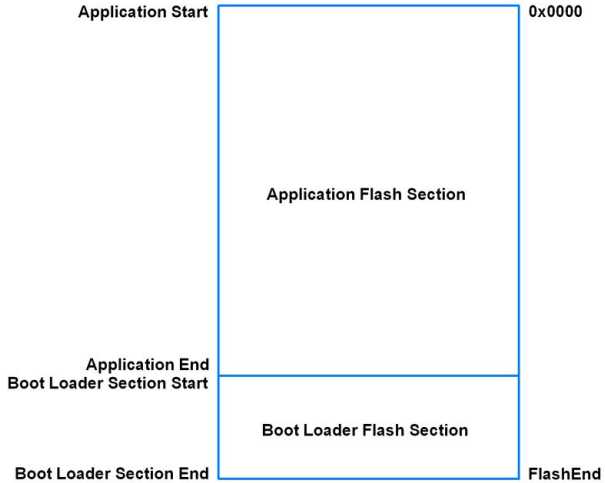
Table 27-4. Boot Reset Fuse(1)

BOOTRST	Reset Address
1	Reset Vector = Application Reset (address 0x0000)
0	Reset Vector = Boot Loader Reset (see Table 27-7 on page 275)

Note: 1. "1" means unprogrammed, "0" means programmed

Table 27-13. Boot Size Configuration, ATmega328/328P

BOOTSZ1	BOOTSZ0	Boot Size	Pages	Application Flash Section	Boot Loader Flash Section	End Application Section	Boot Reset Address (Start Boot Loader Section)
1	1	256 words	4	0x0000 - 0x3EFF	0x3F00 - 0x3FFF	0x3EFF	0x3F00
1	0	512 words	8	0x0000 - 0x3DFF	0x3E00 - 0x3FFF	0x3DFF	0x3E00
0	1	1024 words	16	0x0000 - 0x3BFF	0x3C00 - 0x3FFF	0x3BFF	0x3C00
0	0	2048 words	32	0x0000 - 0x37FF	0x3800 - 0x3FFF	0x37FF	0x3800



1. Grabar el código del bootloader via In-Circuit Serial Programming (ICSP)

Detalles en la Nota de Aplicación ATMEL - AVR109: Self Programming.

Se pueden comprar "already-bootloaded" ATmega328P

AVR Fuses

Se trata de 3 bytes de almacenamiento FLASH permanente (se preservan al remover la alimentación, pueden grabarse cuantas veces sea necesario). No son “fusibles” en el sentido eléctrico. Son “externos”, no pueden accederse desde el ISA. Determinan algunos aspectos básicos del comportamiento del chip. Se pueden grabar con avrdude. Ver en el reporte.

https://www.engbedded.com/fusecalc/

- **Clock selection:** External Clock, Internal 8MHz clock, Internal 4MHz clock, Internal 128KHz clock, External Crystal (0.4-0.9 MHz), External Crystal (0.9MHz - 3.0MHz), External Crystal (3.0MHz - 8.0MHz) or **External Crystal (8.0MHz +)**
- **Clock Out on PortD2 (no)**
- **Clock Divide by 8 (no)**
- **Boot FLASH section (start 0x3800)**
- **Reset Disable (no)**
- **Preserve EEPROM** memory through the Chip Erase cycle **(no)**
- **Watch-dog Timer always on (no)**
- **Serial program downloading (SPI) enabled (yes)**
- **Brown-out Detect** (BOD) **(no)** A brownout for a chip means that the power voltage is too low for it to run reliably at the speed of the clock. if the voltage dips, the chip will turn off until the voltage returns. It will then reset and start over. Ejemplo: a 16 MHz, por debajo de 4.3V el procesador falla. Ver speed and voltage grades de la hoja de datos.

TAREA: comprobar los que estamos usando.

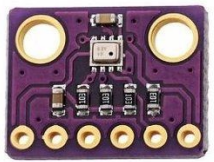
MÓDULOS DE COMUNICACIÓN

UART: Universal Asynchronous Receiver-Transmitter
Comunicación serie **asincrónica, bytes.**
Formato y velocidad configurable (por ejemplo 9600-8-N-1).
Práctica 6 de Técnicas Digitales: Registros de desplazamiento.
Señales eléctricas, Niveles lógicos 0-5V o 0-3.3V, RS-232 y RS-485 (bus).
Serial console (ASCII), Standard output -> printf!
UART a RS-232 y RS-485, instrumentos, protocolos industriales. **MAX232** y **MAX485**.

USB: Universal Serial Bus
Comunicación **sincrónica, paquetes,** modelos que incluyen el hardware. AVR modelos U.
Si no, UART a USB, **FT232** de FTDI, drivers para Windows, COM virtual, terminal.
Nuestro kit tiene el circuito integrado chino **CH340G**.

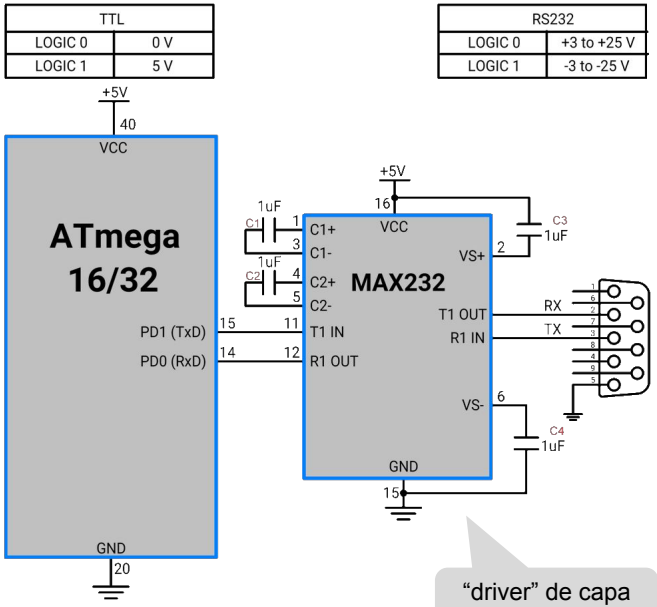
Otros protocolos serie sincrónicos (interchip 0-5V)
I2C: Inter-Integrated Circuit (Phillips 1982), bus
https://en.wikipedia.org/wiki/I%C2%B2C
SPI: Serial Peripheral Interface (Motorola 1985), bus
https://en.wikipedia.org/wiki/Serial_Peripheral_Interface
1-wire protocol: DS18B20

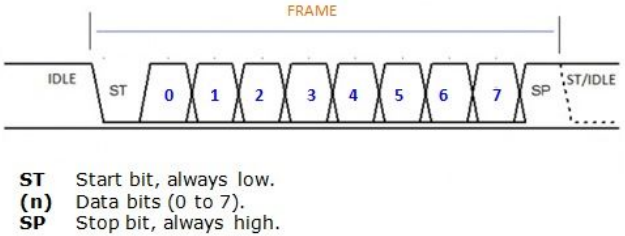
USB-UART - FT232



“Breadboard” BME280, I2C
temperatura y presión atmosférica

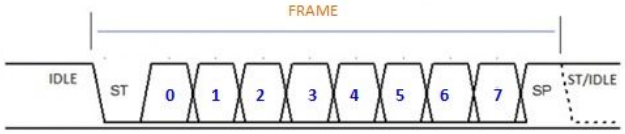
DS18b20



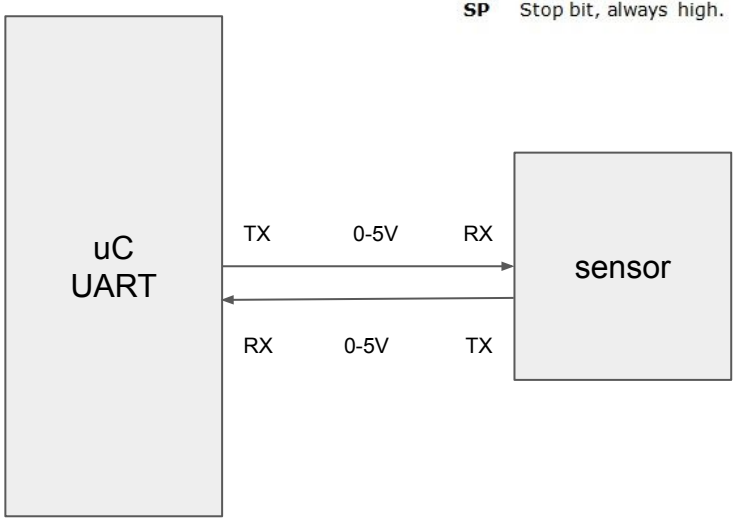


ST Start bit, always low.
(n) Data bits (0 to 7).
SP Stop bit, always high.

UART

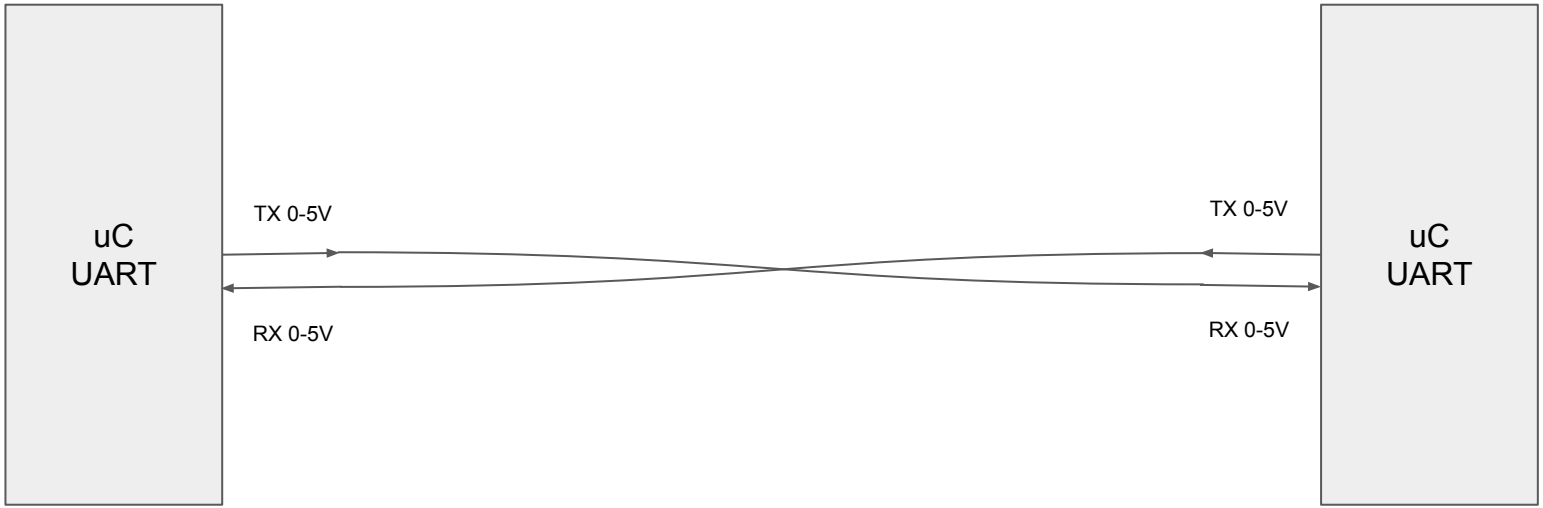


ST Start bit, always low.
(n) Data bits (0 to 7).
SP Stop bit, always high.

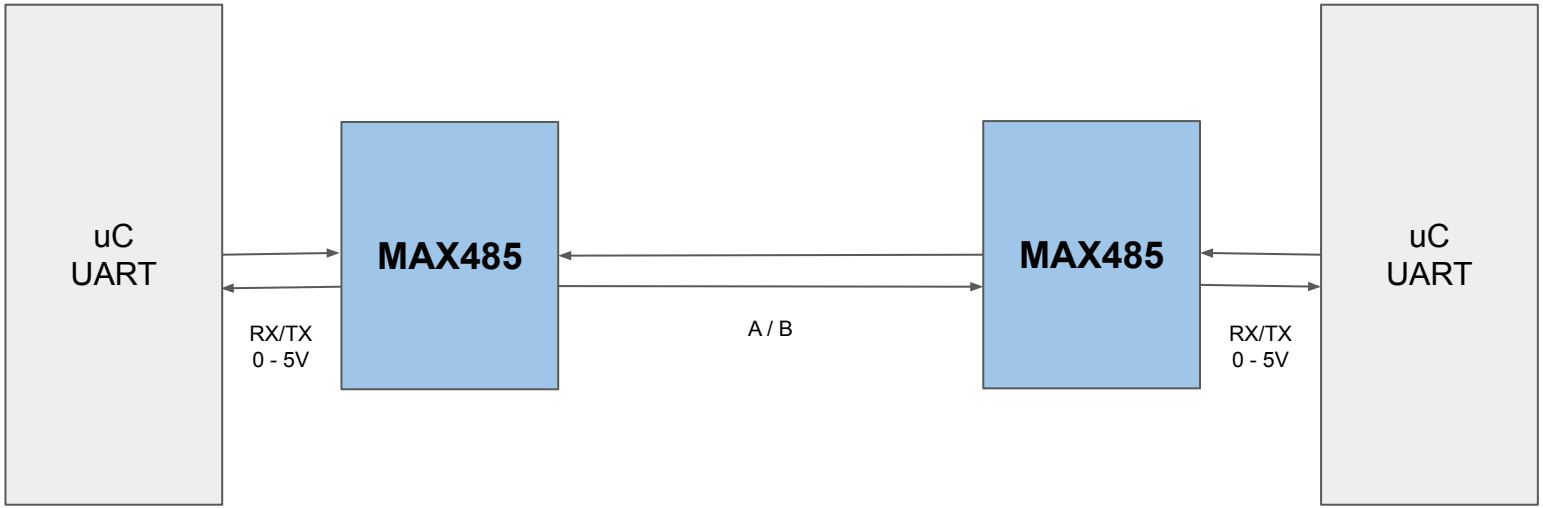


UART - Null modem cable

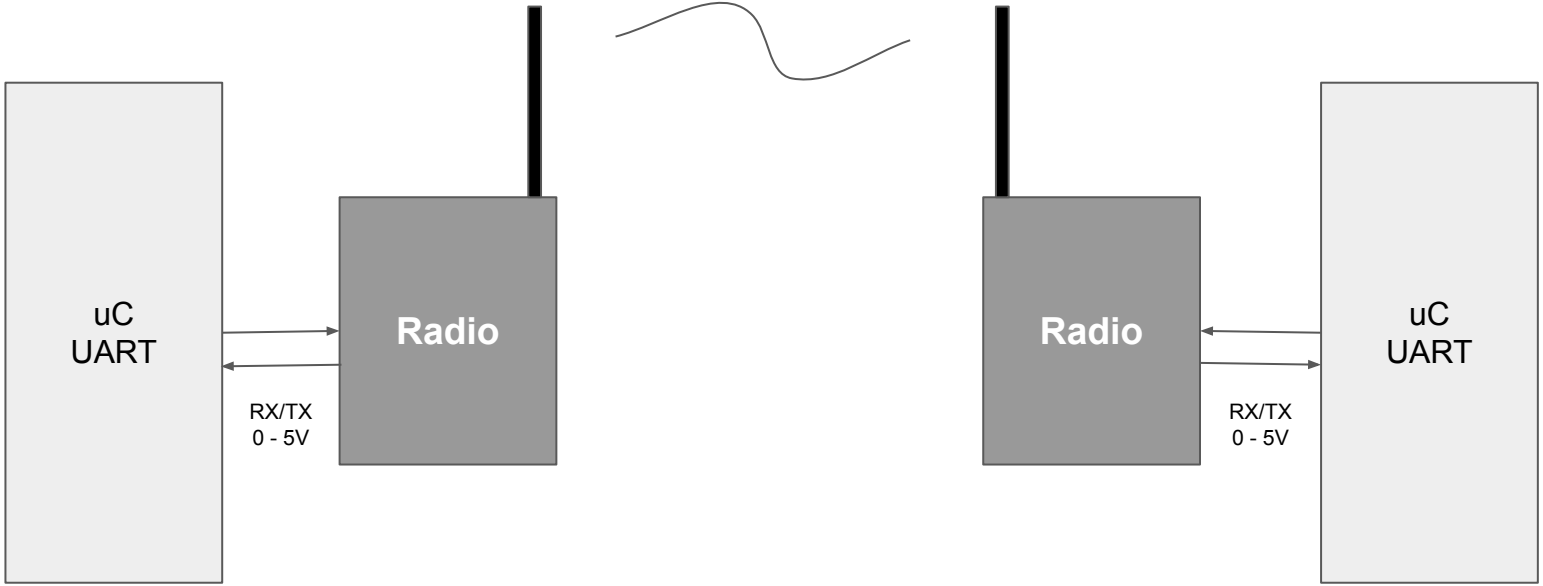
Asincrónico:
recepción basada en
interrupciones



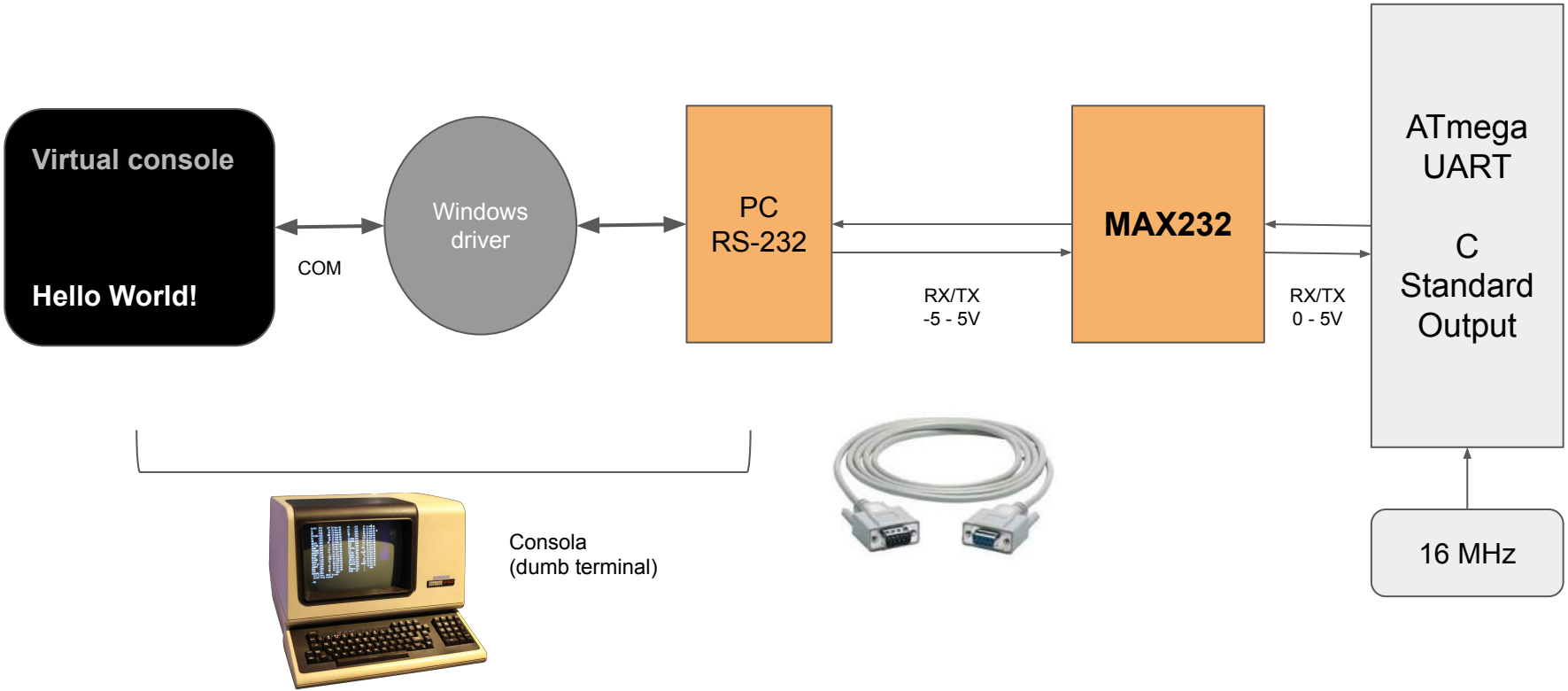
UART - RS-485 / RS-232



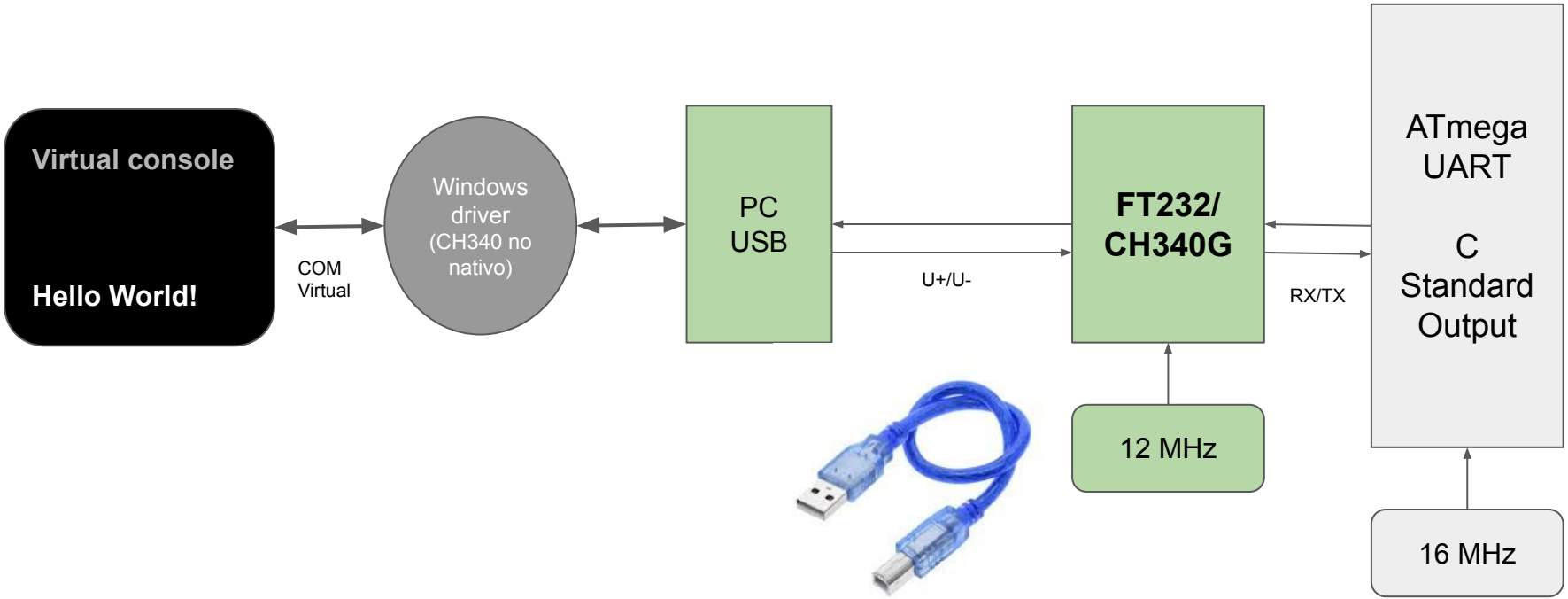
UART - Radio link “transparente”

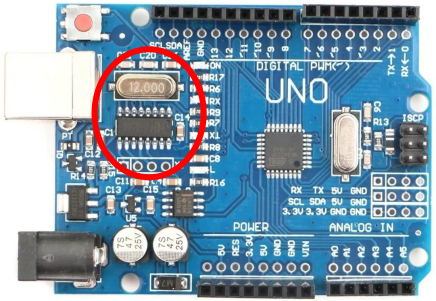


RS-232


Virtual console
Hello World!
COM
Windows driver
PC RS-232
RX/TX
-5 - 5V
MAX232
RX/TX
0 - 5V
ATmega UART
C Standard Output
16 MHz
Consola
(dumb terminal)

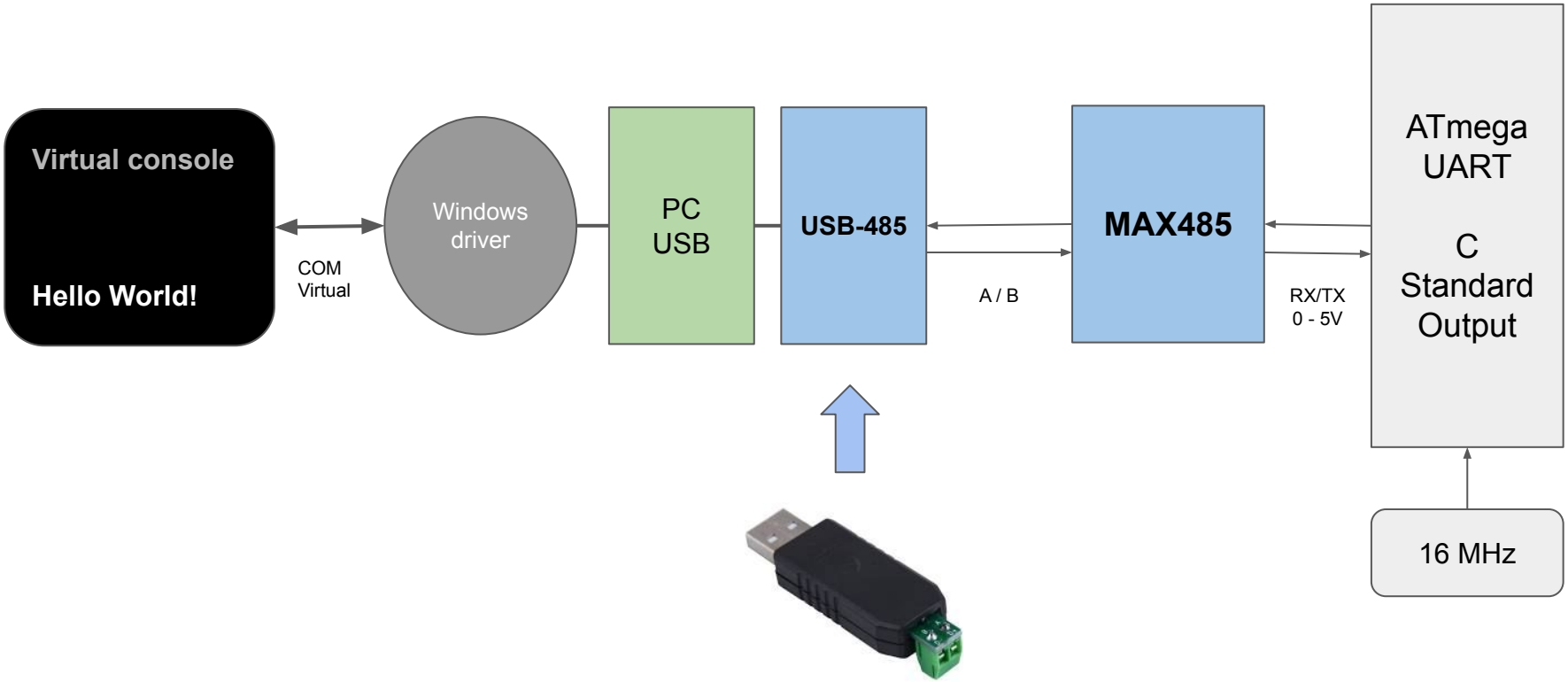
USB


Virtual console
Hello World!
COM
Virtual
Windows
driver
(CH340 no
nativo)
PC
USB
U+/U-
FT232/
CH340G
RX/TX
ATmega
UART
C
Standard
Output
12 MHz
16 MHz



CH340G
Alimentación 5 V
Reloj de 12 MHz

RS-485


Virtual console
Hello World!
COM
Virtual
Windows
driver
PC
USB
USB-485
A / B
MAX485
RX/TX
0 - 5V
ATmega
UART
C
Standard
Output
16 MHz

UART - RS-485 - Modbus

ATmega
UART

C
Standard
Output

getc()
putc()

RX/TX
0 - 5V

IE/OE