Capítulo 4. Microcontroladores

4.1 Introducción

El microcontrolador es el cerebro del robot. Es el encargado de controlar y ejecutar todo las tareas que deba realizar el robot, enviando órdenes a los actuadores y recibiendo datos de los sensores.

En este capítulo estudiaremos en detalle que hay dentro de los microcontrolado mo funcionan. Realizaremos actividades y ejemplos de cómo se utilizan los mic tr s en l robots y entenderemos, por ejemplo, los conceptos de arquitectura ARM, A y e emos que significa que sean de 8,16,32 o 64 bits

Algunos ejemplos de microcontroladores utilizados en robots educativ son los siguientes (ver figura X):

- *Robots lego NXT: Atmel AT91SAM7S256 (contro lor a bits uitectura ARM)*
- *Robots basado en Arduino UNO: ATMega328 ado bits n arquitectura AVR)*
- *Robot Robobuilder: Atmel ATMega128 (or de bit on arquitectura AVR)*

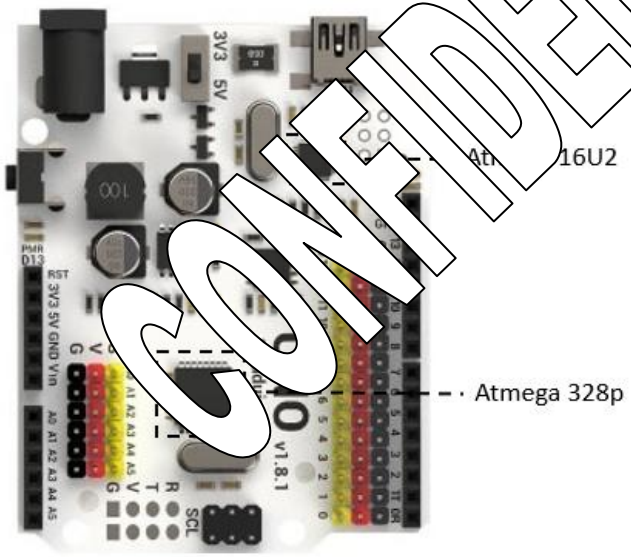
Figura X. Algunos controladores utilizados en robots educativos

En un robot, además del cerebro o microcontrolador principal, puede haber otros microcontroladores secundarios que realizan tareas complementarias. Por ejemplo un display LCD dispone de su propio controlador que descodifica mensajes y los convierte en caracteres o dibujos

en una pantalla[ASVFP1]*. Otro ejemplo son los controladores que se encargan de las comunicaciones (USB, bluetooth, etc).*

Si miramos por dentro la placa base de los brick NXT de lego (ver figura X2.1) podremos encontrar 3 microcontroladores: El principal, el encargado para las comunicaciones y otro encargado para el control de los motores. En un Ardunio UNO (ver figura X2.2) además del controlador principal, podemos encontrarnos un segundo controlador (Atmega 16U2) encargado de las comunicaciones USB.

[ASVFP2]



Definición de microcontrolador

Una definición más exacta de un microcontrolador podría ser: Integrado que incluye un microprocesador, memoria (de programa y datos) y unidades de entrada/salida (puertos paralelo, temporizadores, comparadores, conversores A/D, puertos serie, etc). Por lo tanto el siguiente cuadro resume lo que es un microcontrolador.

MICROCONTROLADOR

=

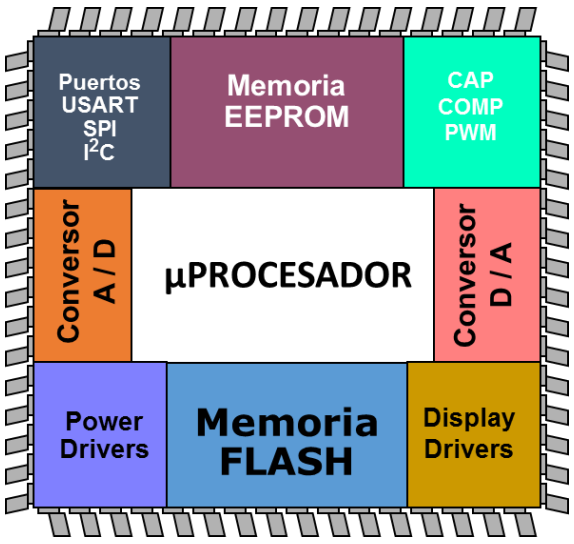
MICROPROCESADOR + MEMORIA + PERIFÉRICOS

- CPU
- Memoria RAM y EEPROM de datos.
- Memoria ROM de programa
- Puertos de entrada-salida.
- Temporizadores/contadores.
- Sistemas de interrupción.
- Convertidores A/D y D/A.
- Etc

robots y sus cerebros

- *Los computadores: microprocesador + memoria + periféricos*
- *El microcontrolador= un computador en un micro*

La figura X3 muestra esquemáticamente los distintos elementos que podemos encontrar dentro de un microcontrolador y que estudiaremos en los siguientes subapartados. Viendo todos estos componentes, podríamos decir que un microcontrolador es como un ordenador pero de reducidas dimensiones



4.2 Microprocesadores

Es el componente electrónico (circuito integrado) encargado de la ejecución de programas, operando con los datos que recibe de la memoria, los dispositivos de entrada/salida y periféricos.

Está compuesto de cuatro elementos principales:

- *La unidad aritmético lógica: realiza todas las operaciones matemáticas*
- *La unidad de control: dirige y controla todos los elementos del procesador*
- *Los registros: áreas de almacenamiento temporal que contienen datos o instrucciones*
- *Buses internos: vías de datos por donde circulan los datos e instrucciones.*

Clasificación según la estructura de memoria

Existen dos arquitecturas típicas de microprocesadores que se refieren a la manera que tienen de acceder a la memoria para leer instrucciones de un programa y obtener datos:

- *Arquitectura Von Neumann: Estos microprocesadores disponen de un único bus de datos para instrucciones y datos. Las instrucciones del programa y los datos se guardan conjuntamente en una memoria común. Cuando la CPU se dirige a la memoria principal, primero accede a la instrucción y después a los datos necesarios para ejecutarla, esto retarda el funcionamiento. (Máquina secuencial).*

- *Arquitectura Harvard: El bus de datos y el bus de instrucción están separados, haciéndolos totalmente independientes Esto permite una ejecución en paralelo más rápida que la Von-Neumann: La instrucción leída a través del bus de instrucciones puede al mismo tiempo utilizar el bus de datos.*

La figura x5 muestra las diferencias entre ambas arquitecturas, que como se puede observar radica en que la Von Neumann solo tiene una memoria para datos e instrucciones (con un solo bus a esa memoria) mientras que la Harvard contiene dos memorias separadas para datos e instrucciones (con buses independientes)

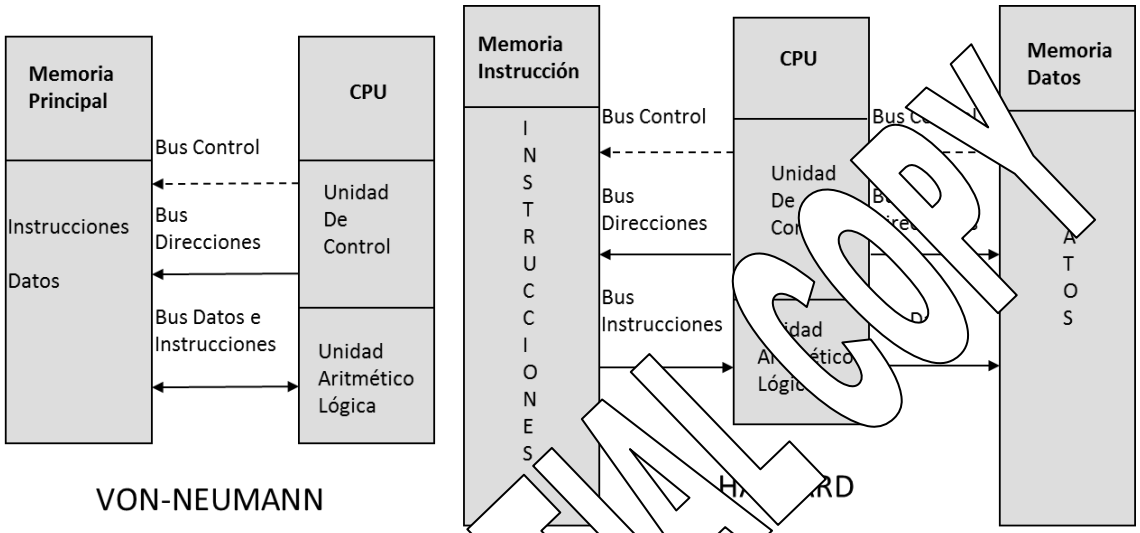


Figura X. Comparativa entre la arquitectura Von Neumann y la arquitectura Harvard

Por cuestiones históricas [illegible] en de microprocesadores con arquitectura Von Neumann: los primeros ordenadores personales se diseñaron con esta arquitectura para abaratar costes [illegible] y compatibilidad todos los ordenadores han seguido con la misma arquitectura. Sin embargo, la mayoría de los microcontroladores que nos encontramos fuera de un ordenador utilizan arquitectura Harvard ya que no necesitan mantener compatibilidades (una lavadora no tiene por qué ser compatible con un microondas) y son más eficientes que los Von Neumann.

[ACTIVIDAD] En el apartado X veremos un ejemplo, utilizando un simulador web, que nos permitirá entender de manera muy sencilla como funciona un microprocesador von-Neumann

Clasificación según el juego de instrucciones

Existe otra clasificación de microprocesadores que los define según su juego de instrucciones. El juego de instrucciones son todas las instrucciones que tiene el procesador. Para entenderlo, al igual que cada idioma tiene su diccionario de palabras con las que construir frases, cada familia de procesadores tiene su conjunto de instrucciones con las que construir programas. La clasificación básica según el juego de instrucciones es la siguiente:

- *Arquitecturas RISC (Reduced Instruction Set Computer). Tienen un diccionario con pocas instrucciones simples. El controlador es sencillo y el chip pequeño. Tienen una alta*

velocidad de ejecución de instrucciones (al ser instrucciones simples). Tienen un consumo reducido. Los programas suelen ser grandes (al no disponer de instrucciones "para todo", cualquier operación que se haga deberá ser una combinación de instrucciones simples). Las arquitecturas ARM que llevan la mayoría de los teléfonos móviles y la arquitectura AVR que llevan los micros de Arduino son familias de aquitecturas evolucionadas de la arquitectura RISC

- *Arquitecturas CISC (Complex Instruction Set Computer). Tienen un diccionario de instrucciones grande con instrucciones muy variadas (en algunos casos equivalen a muchas instrucciones simples). Esto hace que el procesador sea complejo, necesite un consumo alto, pero reduce el código del programa.*

En los robots es importante que un microprocesador ocupe poco espacio (debido al tamaño reducido del robot) y consuma poca energía (para que el robot tenga una autonomía larga), por ello se utilizan micros con arquitectura RISC. Esto ocurre también en los teléfonos móviles y en la mayoría de los dispositivos electrónicos que podemos encontrarnos que tengan un micro integrado (por ejemplo mandos a distancia, televisores, etc). Sin embargo, los ordenadores personales utilizan arquitecturas CISC ya que resulta más importante tener una gran potencia que un reducido consumo y espacio.

Ahora somos capaces de responder estas preguntas:

¿Cuál es la arquitectura de un microprocesador de un ordenador personal? Arquitectura Von Neumann con un juego de instrucciones CISC

¿Cuál es la arquitectura del microcontrolador que hay en un mando a distancia, en un teléfono móvil o en un robot? Arquitectura Harvard con un juego de instrucciones RISC

Ancho de palabra.

Cuando decimos que un procesador es de 8, 16, 32 o 64 bits nos estamos refiriendo al concepto de ancho de palabra del microprocesador. Este concepto se refiere al número de bits que pueden ser tratados "como un conjunto" por el procesador. Por ejemplo, en un procesador de 16 bits los registros serán de 16 bits y las operaciones serán entre números de 16 bits.

4.2.1 ACTIVIDAD: Simulación de un procesador Von-Neumann

Para entender o explicar mejor el funcionamiento de un microprocesador lo mejor es la utilización de simuladores. Existen multitud de simuladores en Internet que son offline (se pueden descargar) y online (para ser utilizados directamente desde un navegador). Por ejemplo, en este libro

proponemos la utilización del simulador online VNSIMULATOR, desarrollado por ***Luciano Ganni*** *para uso personal y educativo y disponible en* http://vnsimulator.altervista.org

Al escribir dicha dirección en un navegador nos aparece el esquema de un procesador Von Neumann simplificado (ver figura X). En él podemos identificar la ***Unidad aritmético lógica (ALU)*** *que será la encargada de realizar las operaciones, La unidad de control (CU) que será la encargada de controlar el funcionamiento del procesador decodificando las instrucciones. Podemos identificar también el registro* ***IR*** *donde se guarda la instrucción procedente de memoria, el registro* ***PC*** *que mantiene la cuenta del programa (línea que se está ejecutando en el instante) y el registro* ***ACC*** *(acumulador) que guarda temporalmente el resultado del contenido del acumulador. La Memoria (****RAM****), al ser de un procesador Von-Neumann almacena tanto instrucciones (parte superior derecha) como (parte inferior derecha).*

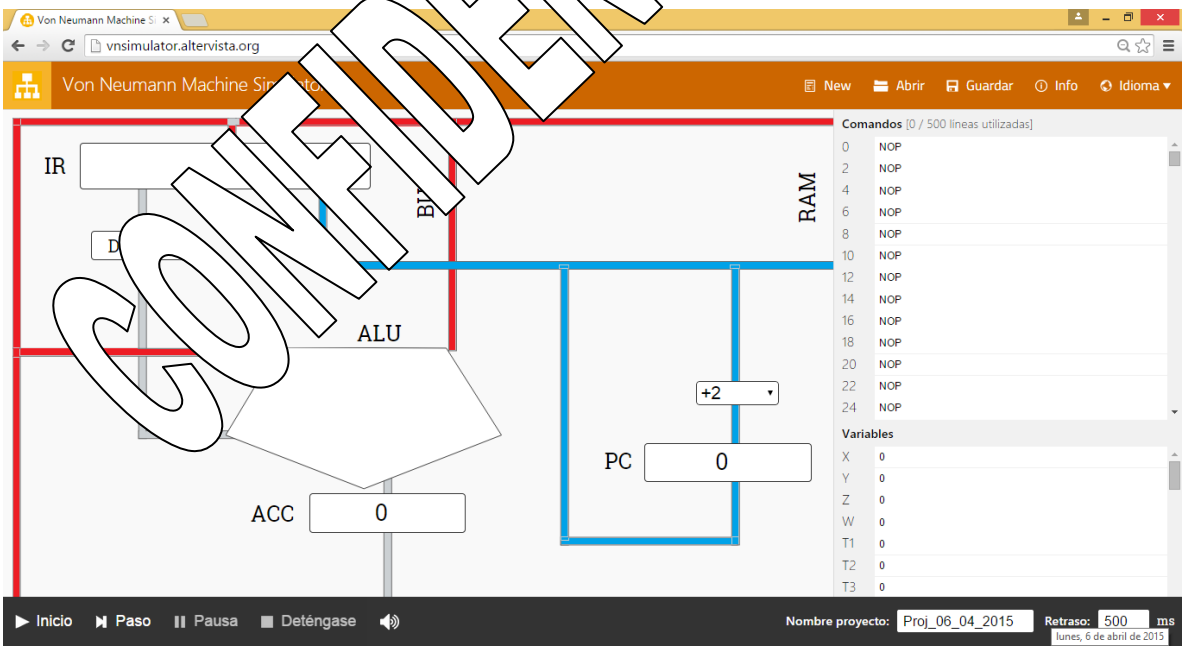


Fig X. Simulador VNSimulator de http://vnsimulator.altervista.org/ *En rojo se muestra el bus de datos y en azul el bus de direcciones.*

El juego de instrucciones de este procesador es muy reducido, por lo que podríamos decir que es un procesador RISC. En el encontramos las siguientes instrucciones:

LOD var	*Carga el contenido de una variable almacenada en memoria en el registro acumulador*

ADD var	*Suma una variable contenida en memoria con lo que haya en el registro acumulador y lo guarda en el acumulador*
MUL var	*Multiplica una variable contenida en memoria con lo que haya en el acumulador y lo guarda en el acumulador*
LOD #num	*Carga en el acumulador el valor de #num*
ADD #num	*Suma el valor de #num con el contenido que haya en acumulador y lo guarda en el acumulador*
MUL #num	*Multiplica el valor #num con el contenido que haya en el acumulador y lo guarda en el acumulador*
STO var	*Almacena lo que hay en el acumulador en una variable en memoria*
SUB var	*Resta una variable contenida en memoria con lo que haya en el registro acumulador y lo guarda en el acumulador*
SUB #num	*Resta el valor #num con el contenido que haya en el acumulador y lo guarda en el acumulador*
DIV var	*Divide una variable contenida en memoria con lo que haya en el registro acumulador y lo guarda en el acumulador*
DIV #num	*Divide el valor #num con el contenido que haya en el acumulador y lo guarda en el acumulador*
JMZ var	*Salta a la dirección de memoria almacenada en una variable de memoria si la operación que se haya realizado antes ha dado de resto 0.*
JMP var	*Salta a la dirección de memoria almacenada en una variable de memoria*
NOP	*No realiza ninguna operación*

HLT	*Para lo que estuviera haciendo en ese momento*

Cuando se escribe un programa utilizando directamente el juego de instrucciones se dice que se está programando a bajo nivel o en lenguaje ensamblador. En el Capitulo X veremos como programar a alto nivel utilizando lenguajes como C o visuales (bloques).

La realización y ejecución de un programa sencillo en ensamblados nos va ayudar a comprender el funcionamiento general de los procesadores. Por ejemplo vamos a escribir un programa que reste dos números cualquiera: ***Z=X-Y***

Escritura del programa

En general, para que la ALU puede operar con dos números (en este caso restar) hace falta que uno de ellos esté en el registro acumulador y el otro provenga de memoria. Por tanto el programa deberá realizar los siguientes pasos:

1º Cargar en el registro acumulador el primer número (que estará en una zona de memoria que llamamos X)

LOD X

2º Restar lo que hay en el acumulador con el segundo número (que estará en otra zona de memoria que llamamos Y)

SUB Y

3º Guardar el contenido del acumulador (que es el resultado de la resta) en una variable Z que almacenaremos en memoria.

STO Z

Como estamos simulando un procesador Von Neumann deberemos tener en memoria principal tanto las instrucciones como los datos. Las instrucciones las colocaremos en direcciones de memoria consecutivas. Como vemos, en el recuadro de la memoria RAM las direcciones vienen dadas por incrementos de 2 bytes (0, 2, 4, 6, …) esto nos indica que el procesador tiene un tamaño de palabra de 2 bytes (16 bits). Por lo tanto cada instrucción tendrá un tamaño de 2 bytes. Escribiremos el código o programa en estas direcciones de memoria consecutivas tal como aparece en la figura X. Vemos también que las variables están almacenadas en la misma memoria RAM, y que tenemos un mismo BUS para datos y para instrucciones lo que nos confirma que estamos con un procesador Von Neumann.

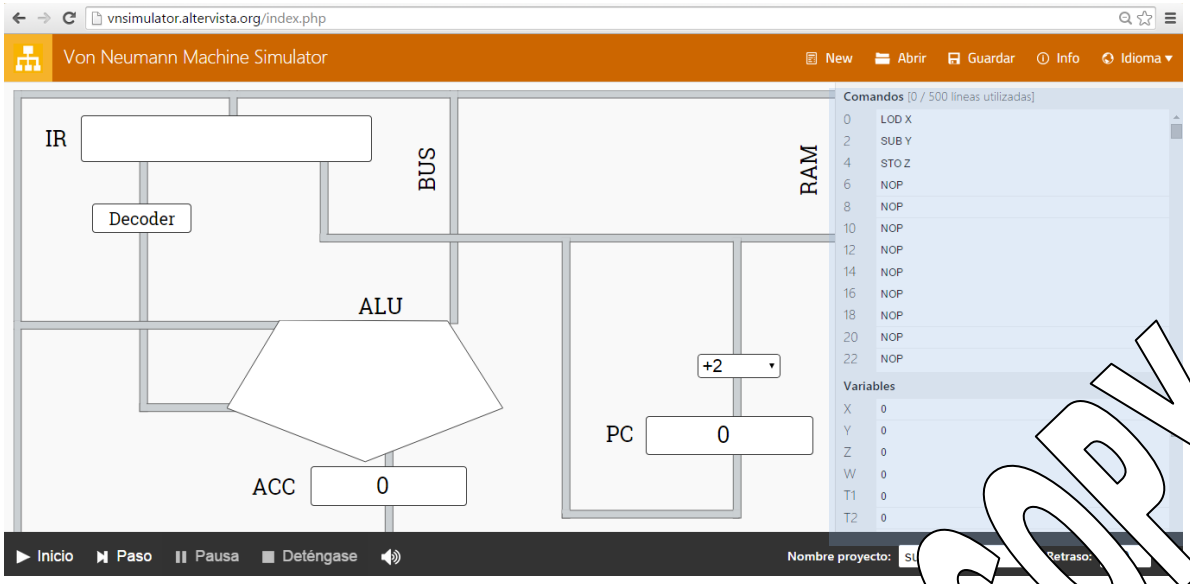


Figura X Zona de memoria (marcado en azul) donde se almacenan los programa (inst cc) y los datos (variables)

Una vez colocado nuestro programa en memoria, podemos introd n la ri le X por ejemplo un 4 y en la variable Y un 2.

Ejecución del programa

*Ahora pulsamos al botón de **inicio** para ver s en estro procesador el programa que hemos escrito. En pantalla se mostrará la ui te au n:*

1º Lo primero que hace el proces or le de cción de memoria que apunta el PC(contador de programa). Como Σ ía ce ro ce en IR el contenido de la dirección de memoria de dirección 0 (ver u X)

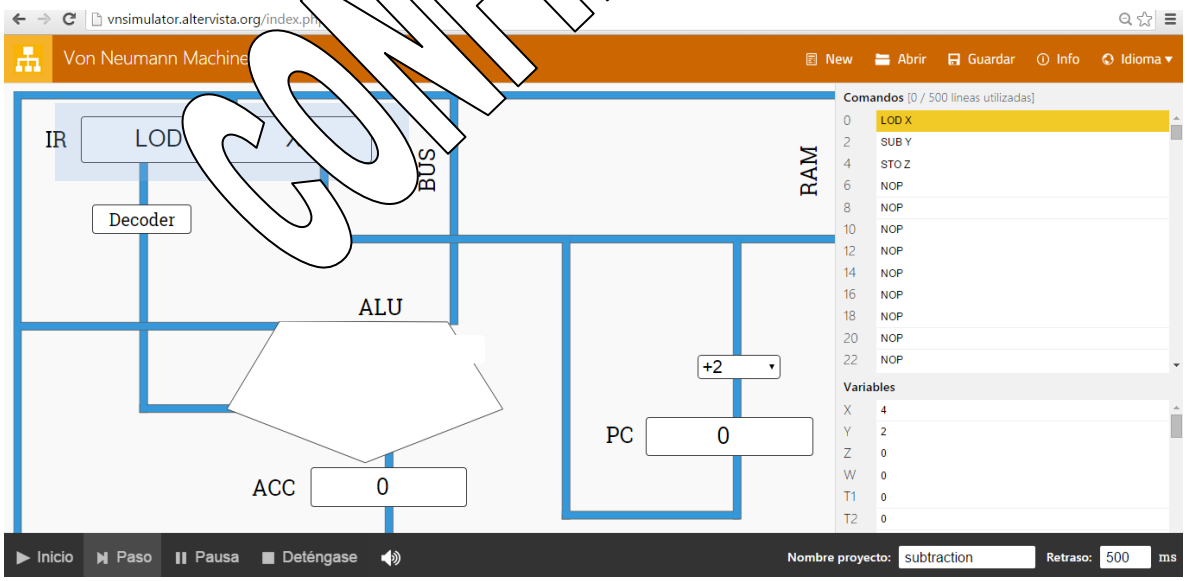


Figura X. Carga en el registro IR de la primera instrucción del programa

*Una vez que esta la instrucción en el registro **IR**, el **DECODER** (unidad de control) deco[illegible] el significado de esa instrucción. En este caso como la instrucción es LOD X la unidad de [illegible] realiza los pasos para meter el contenido de la dirección de memoria X en el acumul[illegible] acumulador valdrá 4. Una vez que termina con esta instrucción, la unidad de co[illegible]ta PC en 2 unidades para que en el siguiente ciclo se lea la siguiente instrucción del pro[illegible]ma.*

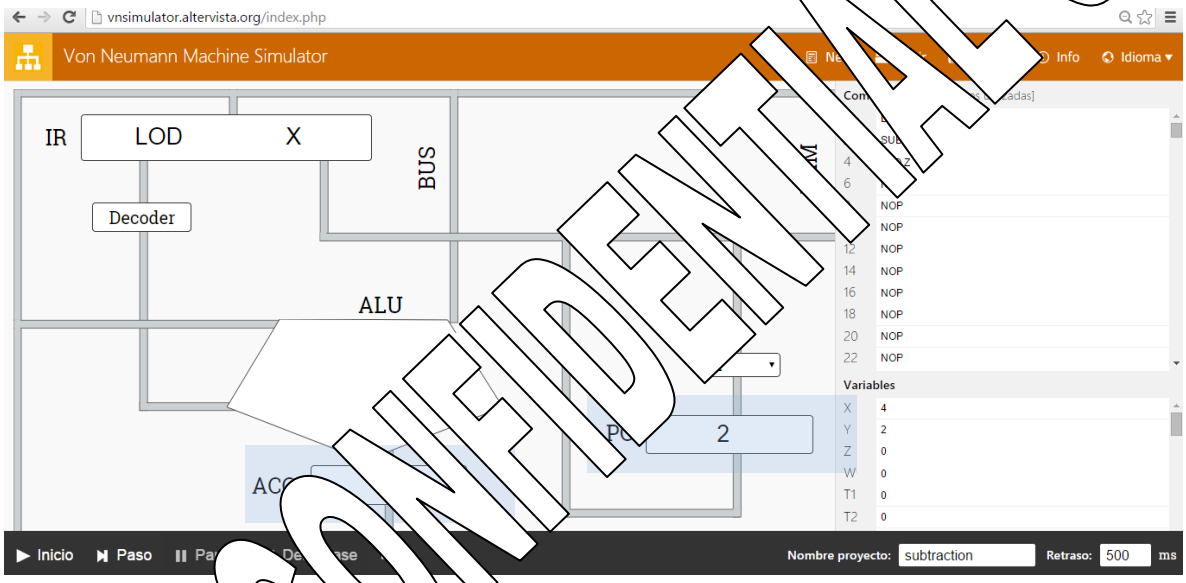


Figura X. Actualizac[illegible]de los r[illegible]s ACC (acumulador) y PC (contador de programa)

2º El procesador mira el [illegible]dor de programa (que ahora vale 2) y carga en el registro IR el contenido de la memoria a la que apunta el contador de programa.

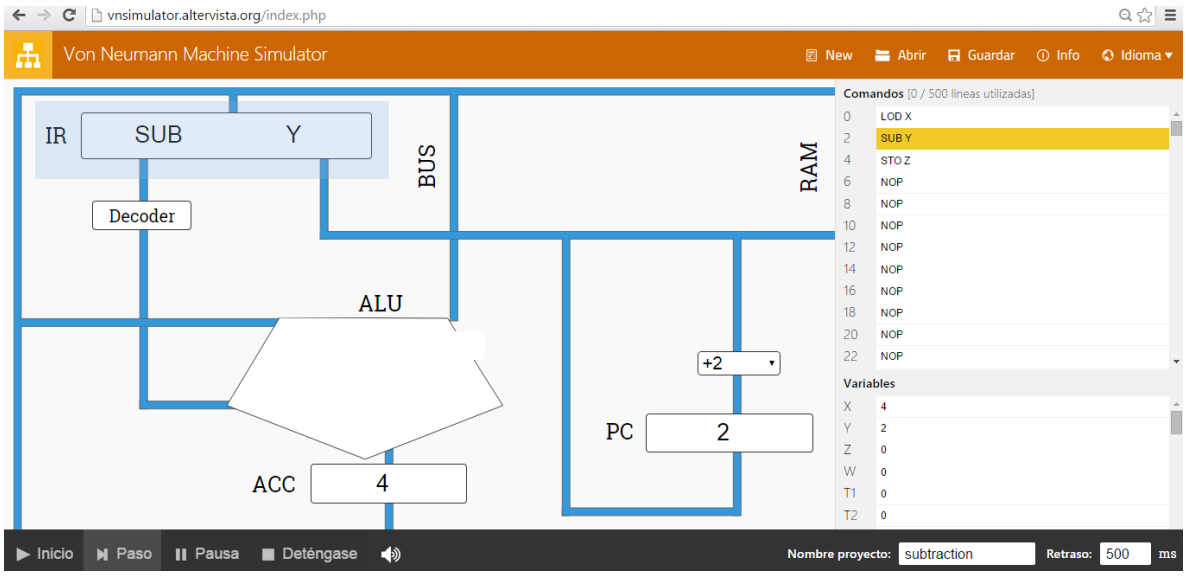


Figura X. Carga en el registro IR de la segunda instrucción del programa

En esta caso la unidad de control al decodificar la instrucción identifica que tiene que restar el acumulador con lo que haya en la dirección de memoria de la variable Y. Por ello lo siguiente que hace es cargar en la ALU por un lado el contenido del acumulador, por otro lado el contenido de la variable Y, y por último los resta guardando el resultado de nuevo en el acumulador. Una vez terminado aumenta el PC en dos unidades

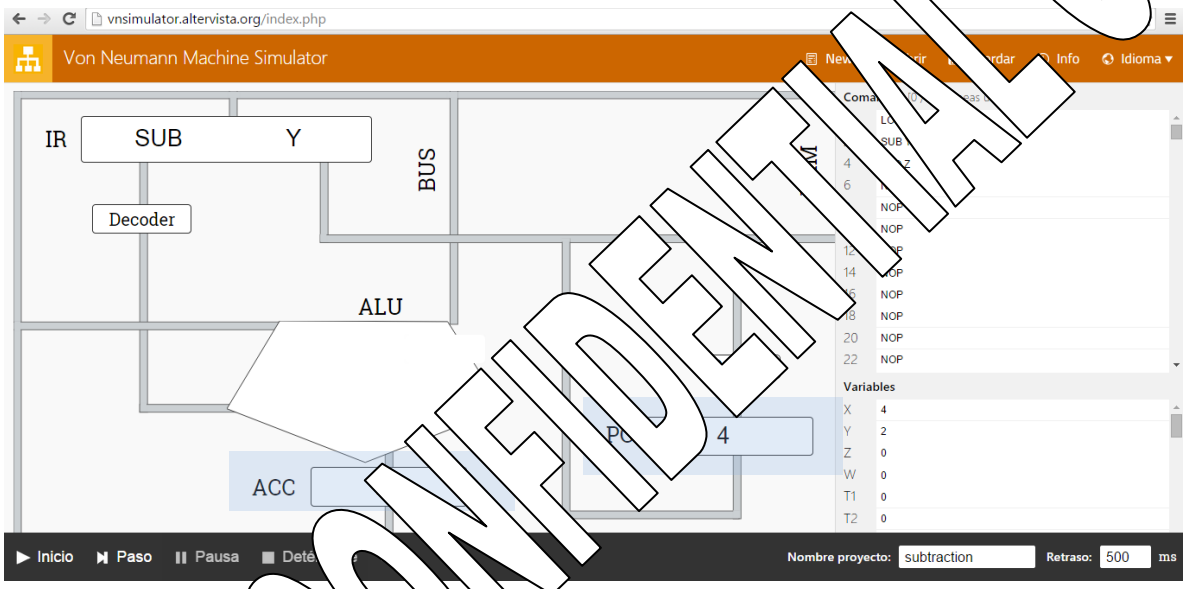


Figura X. Resta en la ALU y actualización del PC

3º El procesador lee del contenido de memoria apuntado por el PC (que ahora es 4) y lo introduce en el registro IR. En este caso la instrucción STO Z está diciendo a la unidad de control (decoder) que debe meter el contenido del acumulador en la variable Z.

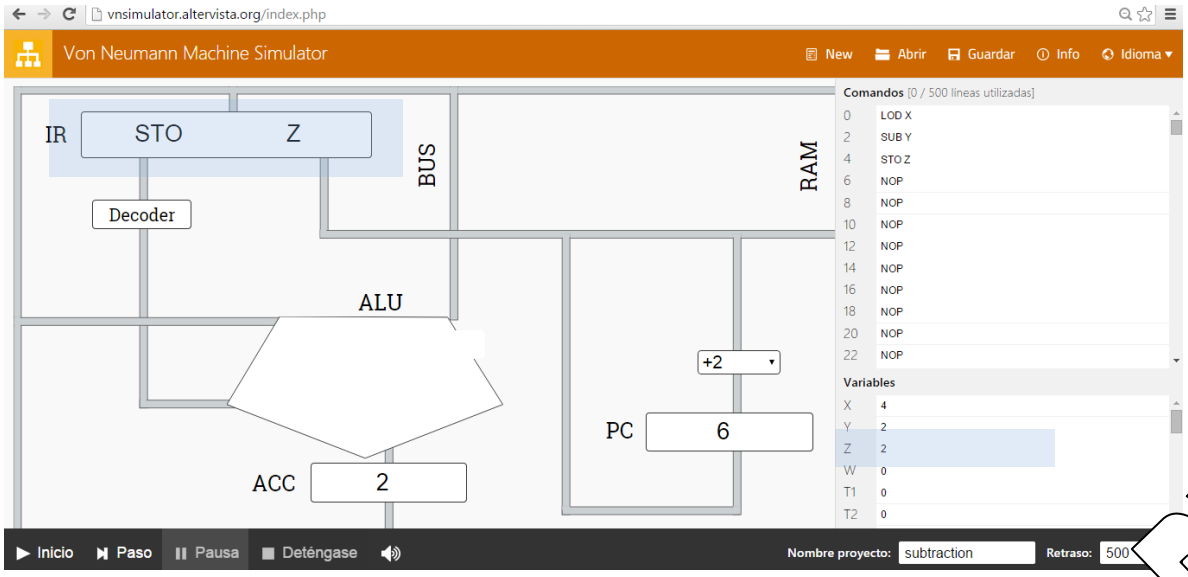


Figura X. Almenamiento del contenido del acumulador en la variable Z de memoria

Como vemos en la dirección de memoria apuntada por Z tendríamos almacenado el resultado de la resta de 4 menos 2.

Este era un ejemplo muy sencillo que nos ha servido para descubrir el funcionamiento básico de los microprocesadores. En la web de VNSimulator podremos encontrar ejemplos en donde aplicar otras instrucciones como multiplicaciones, divisiones, saltos, etc.

4.3. Memorias

Hemos visto que las CPUs o microprocesadores necesitan memorias extras para almacenar instrucciones y datos. En sí, una memoria es un dispositivo capaz de guardar el estado de un bit durante cierto tiempo. Está agrupada en celdas o palabras cada una con la capacidad de almacenar un dato generalmente de tamaño byte (8 bits).

Principios físicos de funcionamiento

- *CAPACITIVO: Basado en condensadores eléctricos. Si los condensadores están cargadas representan un 1 lógico y si se descargan represente un 0 lógico.*
- *FUSIBLES: Un filamento delgado de semiconductor que se quema o se deja intacto para representar un 1 o un 0*
- *ORIENTACIÓN MAGNÉTICA: La orientación determinada de un dispositivo magnético representa un 1 o un 0 lógico*

Tipos de memoria

Podemos clasificar las memorias según su utilidad o según su tecnología. Ya hemos visto los tipos de memoria que tienen utilidad en un procesador, que podemos resumir en:

- *Registros. Como vimos, son elementos de almacenamiento temporal en la CPU (memoria de corto plazo) utilizados para guardar por ejemplo instrucciones (Registro IR), almacenar resultados (Acumulador), etc. Estas memorias tienen un tamaño del ancho de palabra de un procesador (8, 16, etc)*

- *Memoria de instrucción/datos. Son relativamente grandes. Utilizadas en arquitecturas von-Neumann. Solo guardan datos mientras el CPU funciona.*

- *Memoria de programa. Relativamente Grande. Utilizadas en arquitecturas Harvard. Mantienen los datos incluso con el CPU apagada.*

- *Memoria de datos. Relativamente Grande. Utilizadas en arquitecturas Harvard. Almacenan datos mientras el CPU funciona.*

Si las clasificamos según su tecnología podemos encontrarnos

- *Memorias RAM (Random Access Memory). Son rápidas y que permiten acceder directamente a cualquier casilla o celda. Sirven para almacenamiento temporal (pierden la información cuando se le desconecta alimentación) por lo que son útiles como memorias de datos.*

- *Memorias ROM (Read Only Memory). Son más lentas que las RAM ya que para acceder a un elemento de la memoria se debe acceder a todos los anteriores. Sin embargo no pierden la información cuando se desconecta la alimentación por lo que son útiles como memorias de programa. Imaginemos por ejemplo un mando a distancia, el programa que se ejecuta en su microcontrolador debe estar ahí incluso cuando desconectamos las pilas.*

Las memorias ROM pueden clasificarse a su vez en:

  - *EPROM (Erasable-Programable Read Only Memory). Funcionan con el principio de fusibles. Se programa eléctricamente. Una vez programada puede borrarse mediante luz ultravioleta, por eso estas memorias tienen una especie de ventana donde aplicar la luz ultravioleta (ver imagen X)*

«ST Microelectronics M27C256B (2006)». Publicado bajo la licencia CC BY-SA 2.5 vía Wikimedia Commons - http://commons.wikimedia.org/wiki/File:ST_Microelectronics_M27C256B_(2006).jpg#/media/File:ST_Microelectronics_M27C256B_(2006).jpg

- *EEPROM (Electrically Erasable-Programable Read Only Memory) funcionan también con el principio de fusibles. Pueden borrarse con impulsos eléctricos programados (no hace falta luz ultravioleta), por ello son las más utilizadas hoy en día en microcontroladores utilizados en robots.*

- *Flash. Son una evolución de las EEPROM que permiten la lectura y escritura de múltiples* posiciones de memoria en la misma operación. Gracias a la tecnología *flash*, permite velocidades de funcionamiento muy superiores a la tecnología EEPROM, que sólo permite actuar sobre una única celda de memoria en cada operación de programación. Se trata de la tecnología muy empleada en las memorias USB.

[ACTIVIDAD escribir X, Ejemplo X, se muestran varios ejemplos donde los programas que se cargan en los microcontrolador son almacenados en la memoria Flash o memoria EEPROM

Buses de memoria

Como vimos en la figura X (comparativa). Los procesadores tienen un bus de direcciones para identificar las celdas de memoria sobre las que se quiere leer o escribir, y un bus (o buses, si es una arquitectura Havard) de datos e instrucciones por donde entran y salen datos e instrucciones de

cada una de las casillas o celdas de la memoria. La figura X mostraba en color azul el bus de direcciones y en color rojo el bus de datos e instrucciones en el simulador VNSimulator

Tamaño de memoria direccionable

A veces el concepto de ancho de palabra visto en los microprocesadores se aplica también al tamaño de memoria que se puede direccionar o acceder desde el procesador. Por ejemplo, en la mayoría de los microprocesadores actuales el bus de dirección es de 32 bits lo que permite especificar a la CPU 2^{32} = 4.294.967.296 direcciones de memoria distintas. Las direcciones de memoria se expresan a menudo en hexadecimal. Por ejemplo, para no tener que escribir 11111101010000000000010101100 podemos escribir 3F5000AC en hexadecimal

4.4 Periféricos

Son los dispositivos que intercambian datos con el procesador.

En un microcontrolador podemos encontrar diferente tipos de periféricos. Lo comunes es que estos periféricos correspondan a puertos de entrada/salida que pueden ser analógicos (pueden tomar cualquier valor) o digitales (valen 0 o 1).

Los puertos además podemos clasificarlos en paralelos o serie. Basicamente diremos que un puerto es paralelo cuando sus entradas, salidas y control son para leer directamente valores (señales de voltaje) de sensores digitales o analógicos. Diremos que un puerto es serie cuando lo utilicemos las entradas y salidas para comunicarnos mediante un protocolo serie, con otros microcontroladores o dispositivos. A continuación estudiamos los periféricos más comunes según sean paralelos o serie.

Puertos paralelos

- *Salidas digitales Se suelen utilizar para controlar relés, leds, etc. Se caracterizan por una corriente máxima individual y una máxima común. Existe un tipo especial que son salidas de Potencia que permiten activar elementos que requieran una corriente o tensión elevada. Para ello se utilizan circuitos especiales como son:*
    - *Montajes Darligton (basados en transistores)*
    - *Control de relé*

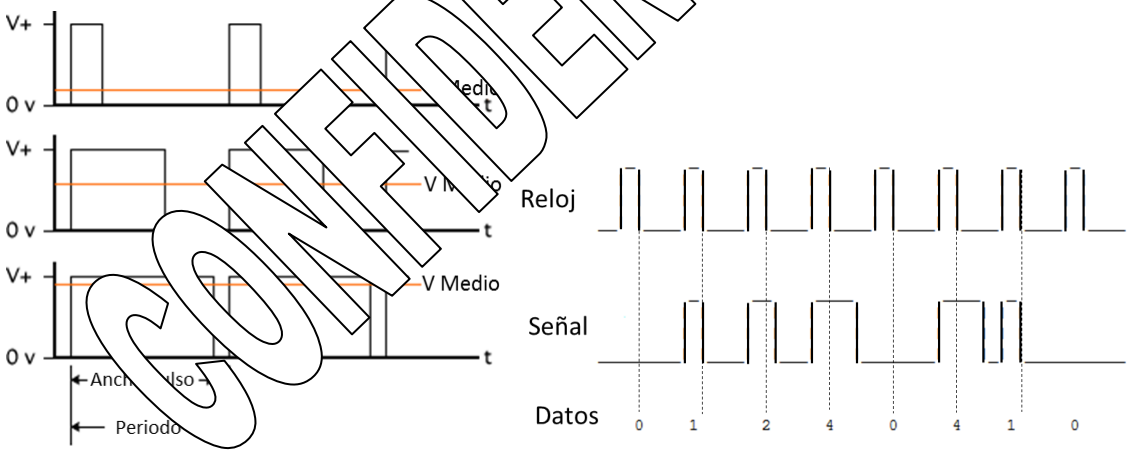
- *Control de triacs (basados en transistores con la función de interrumpir o dejar pasar corriente alterna)*

[ACTIVIDADES] En el libro X podemos encontrar las siguientes actividades relacionadas:
- Apartado X. Activación de un LED
- Apatado X. ….

Salidas digitales con PWM

Los microcontroladores no suelen disponer de salidas analógicas porque s[illegible]uy c[illegible]sas y difíciles de implementar. A cambio utilizan salidas digitales con PWM ([illegible]ción p[illegible] ancho de pulso). Básicamente un PWM consiste en sacar a través d[illegible]lid[illegible]ital una onda cuadrada, cuya amplitud puede ser variada (ver imágenes[illegible]

En la práctica una salida en PWM puede servir para emular un v[illegible]or analóg[illegible]yo valor medio viene definido por el ancho de pulso respecto [illegible]periodo (v[illegible]ma[illegible]n X). También puede servir para enviar información codificada [illegible]viendo el puer[illegible]n un canal de comunicaciones serie. Por ejemplo, en la figu[illegible]X.b[illegible]de [illegible]er[illegible]rse que según sea de grande la anchura de pulso respecto a la se[illegible]de[illegible]loj [illegible]ta[illegible]in[illegible]valor decimal: ningún pulso en la señal equivale al dato 0, un [illegible]de la [illegible]sm[illegible]amplitud que la señal de reloj significa un 1, un pulso el doble que la se[illegible]l de [illegible]i sig[illegible]ca[illegible]n 2, etc.



(a) (b)

Figura X. a PWM utilizado para emular una salida analogica. b) PWM utilizado para trasmitir datos conviertiendo una entrada/salida en un puerto de comunicaciones serie

La simplicidad de las salidas PWM y sus aplicaciones hacen que muchos microcontroladores incorporen este tipo de salidas. Por ejemplo todos los microcontroladores Atmel que incorporan las placas Arduino tienen este tipo de salidas, que vienen indicadas en la numeración de los pines con el símbolo ~ (ver Imagen X)

Figura X. Pines en Arduino UNO que pueden actuar como salidas analógicas

Las aplicaciones en robots de las salidas pwm [illegible] pueden utilizarse para comunicaciones por infrarrojo, para controlar la [illegible] de los servomotores, etc.

[ACTIVIDADES] En el [illegible] encontrar las siguientes actividades relacionadas:

- Apartado X. Transmisión [illegible] información a través de un emisor/receptor infrarrojo.

- Apartado X. Control de la posición de un servo motor.

- Apartado X. Control de luminosidad de un LED.

- Apartado X. Generación de tonos con un zumbador

- Apartado X. Control de la velocidad de giro de un motor de corriente contínua.

- *Entradas en paralelo digitales. Se denominan pines de entrada digitales y se utilizan para lectura de pulsadores, teclados, interruptores o en general para leer cualquier cualquier dispositivo todo/nada. Pueden estas optoacopladas lo que permite* el aislamiento eléctrico entre los circuitos de entrada y salida[ASVFP3]**.**

  En los robots las entradas en paralelo digitales son muy útiles y pueden servir para multitud de aplicaciones como encender leds, detectar colisiones utilizando pulsadores, seguir líneas con sensores de luz. etc.

> *[ACTIVIDADES] En el libro X podemos encontrar las siguientes actividades relacionadas:*
> *- Apartado X. Activación de un LED mediante un pulsador.*
> *-Apartado X. Detección de obstáculos con pulsadores.*
> *- Apartado X. Seguimiento de líneas con sensores infrarrojos digitales.*

Entradas de alta impedancia y resistencias pull-up o pull-do

Es común que las entradas en los microcontrola res ...de ...mpedancia" Esto significa que el circuito de lectura de la ent... de ...na ...a corriente, lo que permite leer sensores cuya señal es muy dé... ...mbarg... es... tipo de entradas tienen el problema que son muy sensibles al r... p... ejem...lo el ...ducido por la electricidad estática. Por ello, a este tipo de ... du... suele ...coplar un circuito muy simple llamado resistencia pull-up o pull-d...n ...e ...ite ...problemas producidos por el ruido eléctrico. De hecho, muchos ...n...adores, ...mo los Atmel de Arduino incorporan estas resistencias que pue... ac...rse ...le...mente.

Básicamente una ...sis...cia pu...up ... muestra la figura X.b lo que hace es tener siempre una en...da ...1 (...ctada ...+V) hasta que el sensor externo, como puede ser un interruptor, ...ue... ...a (la pone a 0). Una resistencia pull-down (figura x.a) lo que hace es ...ner...en...re ... entrada a 0 (masa) hasta que un sensor externo la pone a 1 (lo pu... ...a ...

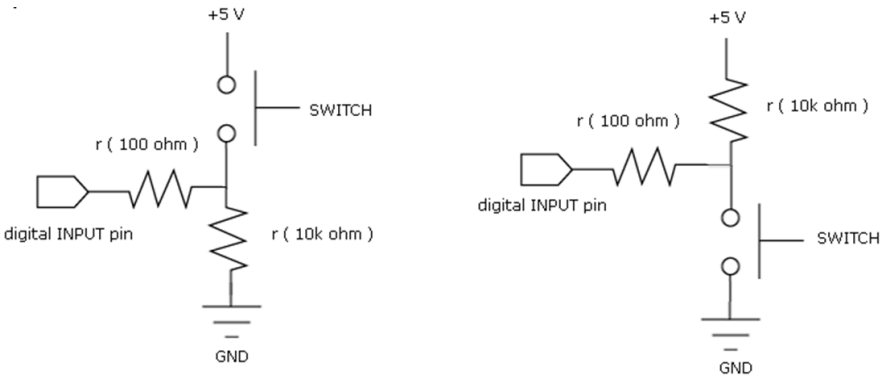


Figura X. a) Interruptor conectado a un entrada con resistencia pull-down. b) Interruptor conectado a una resistencia pull-up

[ACTIVIDAD] En el capítulo X del libro X se muestra un ejemplo práctico de lo que sin estos circuitos las entradas de un microntrolador arduino sufren los efectos indeseados de los ruidos eléctricos y cuando les conectamos las resistencias pull-up los ruidos desaparecen

Puertos serie

Los puertos serie son entradas y salidas digitales que se utilizan para transmitir datos de manera serie (un dato tras otro). En el apartado anterior vimos como un dispositivo PWM podría servir para realizar una comunicación serie con un protocolo muy sencillo a través de una salida digital (ejemplo de la figura X). Por lo tanto para comunicarnos necesitamos dispositivos que adapten las señales y protocolos que utilicemos en las comunicaciones. A continuación presentamos los dispositivos más comunes que permiten a los microcontroladores utilizar puertos para comunicación serie y los protocolos más utilizados.

Dispositivos

- *UART (Universal Asynchronous Receiver-Transmitter o Transmisión Serie Asíncrona). Es un dispositivo hardware que llevan casi todos los microcontroladores y la mayoría de ordenadores para traducir datos de paralelo a serie. Para ello utiliza un registro llamado de "desplazamiento" que va mentiendo*

secuencialmente todos los datos que se quieren utilizar. Al ser "asíncrona" significa que no necesita una señal de reloj comun para sincronizar el emisor y el receptor. Para que el receptor reciba bien los datos estos se envían en tramas (ver Figura X). Estas tramas están compuestas por los siguientes campos.

- *Un bit de comienzo que indica que se ha comenzado a enviar una trama.*
- *Entre 5 y 8 bits de datos. Estos datos corresponden a la información que el emisor quiere enviar y pueden ser caracteres, lecturas de sensores, etc.*
- *Uno o dos bit de paridad. Sirve para que el receptor compruebe si los datos recibidos son correctos. El emisor pone un 1 en este bit si el número de unos en los datos es par y un 0 si es impar. El receptor cuenta los unos en los datos y comprueba si coincide con el bit de paridad. Si no es así se considera que la trama es errónea (ha ocurrido un error durante la transmisión)*
- *El bit de parada, que es opcional. Sirve para indicar al receptor que se han dejado de enviar.*

Figura X. Estructura de una trama enviada por puerto serie

Se dice que la comunicación puede ser Full Duplex cuando se puede [illegible] mismo tiempo y Half Duplex cuando solo se puede hacer una de las dos [illegible] La velocidad típica de comunicación va de 9600 baudios (9600 bits en [illegible] 115200 baudios. Normalemente en la comunicación serie a través [illegible] protocolo rs232 que veremos a continuación.

- *USART (Unidad de Transmisión Recepción Síncron[illegible]) [illegible] el anterior pero que permite comunicaciones s[illegible] además de asíncronas. Los dispositivos que [illegible] veces) que un adaptador UART, por ello [illegible] USARTs hoy en día.*

[ACTIVIDADES] En el [illegible] se muestra un ejemplo práctico de utilización de puertos s[illegible]
- Comunicar un Robo[illegible] con un PC
- ..

. Protocolos

Blah blah blah

La comunicación entre el procesador y el periférico está regulada de acuerdo con dos métodos:

- *POLLING: El procesador revisa ordenadamente todos los periféricos para atender a cada uno de ellos secuencialmente.*
- *INTERRUPCIONES: El periférico que está listo para ser atendido por el procesador solicita una "interrupción" de la ejecución del programa para que el procesador lo atienda.*

+ Arduino