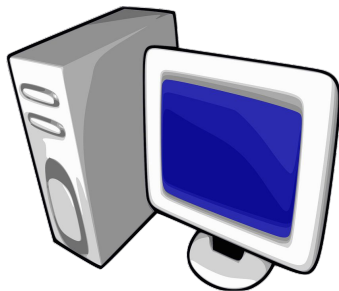


De la PC al LPC845

Curso R2004
UTN - FRBA

Un poco de perspectiva (I)

¿Qué hacíamos en Informática I?



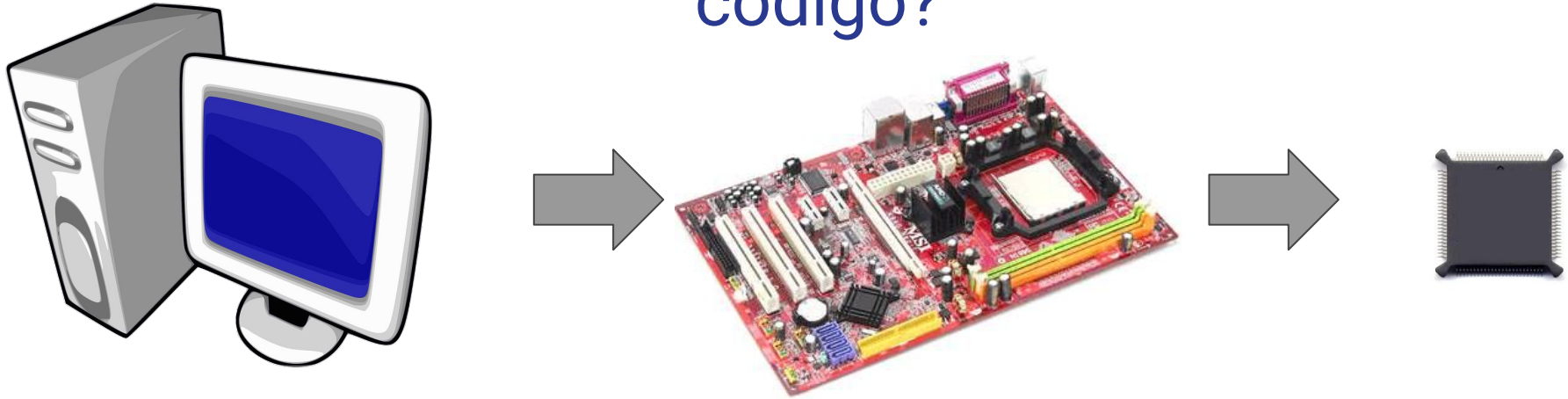
```
#include <stdio.h>
```

```
int main (void)
{
    printf("Hola Mundo\n");
    return 1;
}
```

```
chor@chor-ntbk: ~/Documentos/Facu/HolaMundo
chor@chor-ntbk:~/Documentos/Facu/HolaMundo$ gcc HolaMundo.c -o HolaMundo.o
chor@chor-ntbk:~/Documentos/Facu/HolaMundo$ ./HolaMundo.o
Hola Mundo
chor@chor-ntbk:~/Documentos/Facu/HolaMundo$
```

Hasta ahora veníamos programando, compilando, linkeando y ejecutando nuestros programas en la misma plataforma (en la PC). Esto implica que muchas veces nos olvidamos de “quien” ejecuta el programa, cuales son los recursos con los que contamos (periféricos, memoria, etc.)

Y entonces... ¿Quién es el que ejecuta nuestro código?

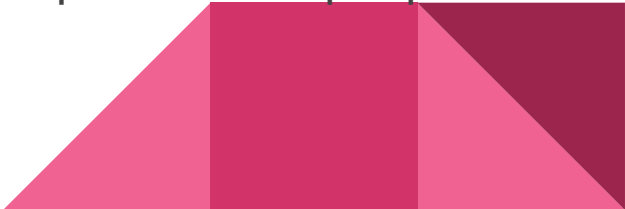


Dentro de la PC tenemos muchos dispositivos (periféricos) y sistemas embebidos. En particular, en la placa base (motherboard) se encuentran los periféricos principales, y ante todo, el procesador. Este último es un circuito electrónico inteligente cuya función es la de procesar todo el código (los programas) que se encuentren en ejecución en nuestra computadora. En última instancia, si queremos quedarnos con la pieza indispensable, o el corazón de nuestra PC, es el microcontrolador (con algunos periféricos asociados) quien se encarga de la ejecución de todos nuestros programas

¿Qué es un circuito electrónico inteligente?

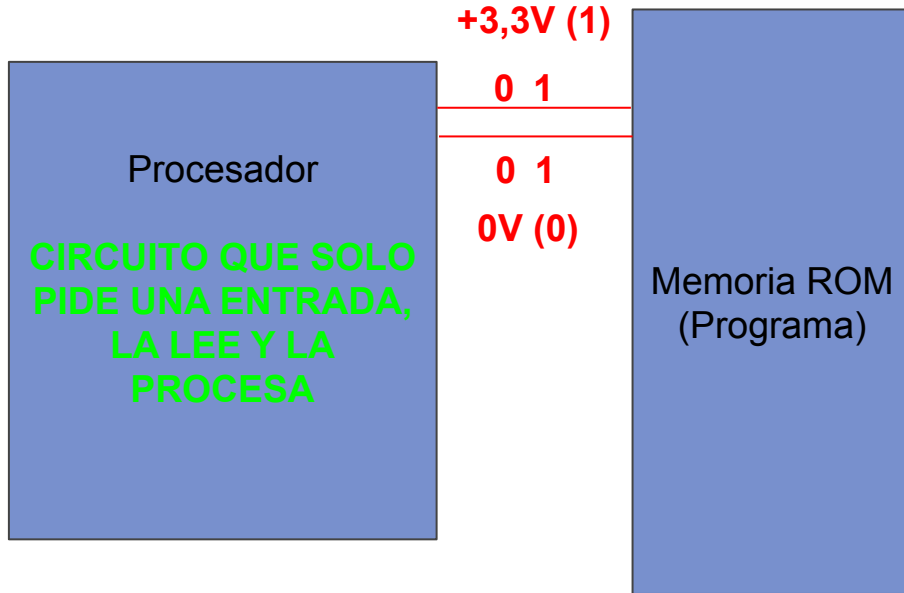
Un procesador es un circuito que, básicamente, y en forma muy esquemática, posee una entrada (bus de datos), formada por N “cables”, y una salida (bus de direcciones) y cuya única funcionalidad es incrementar el número que se envía por el bus de salida, recibir un dato por el bus de entrada, y modificar sus registros internos (y el bus de datos) según el número que se reciba por el bus de entrada.

De esta manera, si ponemos un procesador con un solo cable en el bus de entrada, el mismo solo podrá hacer 2 operaciones (instrucciones 0 y 1) posibles con sus registros. A medida que incrementamos la cantidad de “cables” que posee el bus de entrada, incrementamos la cantidad de operaciones que puede hacer el procesador.



¿Qué es un circuito electrónico inteligente?

¿Cómo funciona un circuito electrónico inteligente?



1 cable:	0 ó 1	- 2 instrucciones
2 cables:	00, 01, 10 ó 11	- 4 instrucciones
3 cables:	000, 001, 010, 011 100, 101, 110, 111	- 8 instrucciones

N cables: - 2^N instrucciones

LA CANTIDAD DE CABLES (TAMAÑO DEL BUS)
ME INDICA CUÁNTAS INSTRUCCIONES PUEDO
PEDIR, O SEA CUANTAS COSAS DISTINTAS
PUEDO HACER EL PROCESADOR

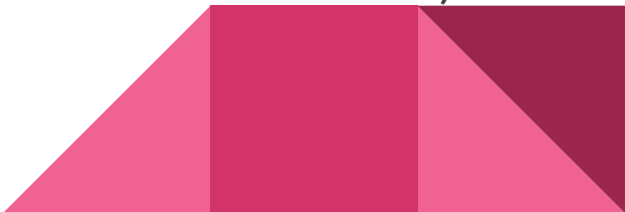
Programando un circuito electrónico inteligente

Cada procesador se diseña de manera que pueda hacer una cantidad limitada de instrucciones, que son el SET DE INSTRUCCIONES de ese procesador.

Esas instrucciones LIMITADAS son las que se combinan para generar un programa.

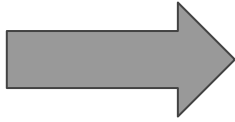
Las instrucciones se guardan en la memoria ROM como combinaciones de ceros y unos.

El código assembler es un código que le asigna a cada instrucción un nombre, que permite escribirla más fácilmente



De C a Assembler, a código de máquina

var++;



Traductor

MOV A,@var
INC A
MOV @var,A

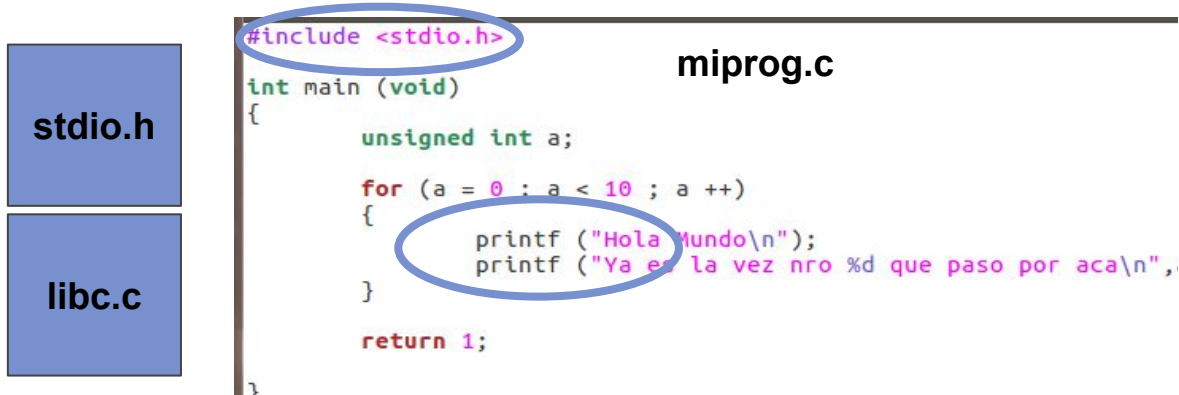


Compilador

1000101010000110
1001011110011100
1000101000010101

Resumiendo: Cuando escribimos un programa en C, lo compilamos y ejecutamos, estamos traduciendo de un lenguaje sencillo (C) a un lenguaje de MENOR NIVEL (Assembler), el cual es fácilmente transportable a un grupo de instrucciones que pueden guardarse en una memoria. Lo único que debe hacer el procesador a continuación es ejecutar de una en una las instrucciones que nosotros grabamos en la memoria de programa (ROM).

¿Y con programas más complejos?




De acuerdo al dispositivo que vamos utilizando, se crean conjuntos de funciones en C que ofrecen al usuario una serie de funcionalidades ya desarrolladas para el manejo específico de periféricos, como el teclado o la pantalla. Esto no implica que el procesador venga con instrucciones para manejar estos periféricos, sino que los desarrolladores de los mismos se toman el trabajo de generar funciones que les permitan a los usuarios acceder a sus dispositivos mediante funciones apropiadas para cada procesador.

A estas funciones se las conoce como **DRIVERS**.

Para que nosotros como programadores no tengamos que saber que periférico en particular tiene cada plataforma, y hagamos programas “estándar”, existen las funciones **primitivas**, que luego utilizan internamente el driver apropiado en cada entorno.

¿Qué es lo que hace entonces nuestro entorno de desarrollo cada vez que compilamos un programa?

1. El compilador busca todos los archivos que son necesarios para construir nuestro programa (`#include "myheader.h"`)
 2. Chequea que los llamados a función sean coherentes con los prototipos que aparecen en los headers incluidos.
 3. Chequea errores de sintaxis en todos los archivos y los **COMPILA** por separado.
 4. Toma TODOS los archivos de código compilados y genera un programa a partir de todos ellos (los **LINKEA**)
- 

Microprocesador, nuestro circuito electrónico inteligente

Resumiendo todo lo anterior, una vez que escribimos un programa, lo linkeamos y compilamos, terminamos ejecutando una serie de instrucciones en código máquina (traducidas en un archivo binario). Este conjunto de instrucciones terminará en la memoria ROM del dispositivo.

Finalmente, nuestro MICROPROCESADOR, como dijimos, se encargará de pedir secuencialmente todas estas instrucciones, y modificar sus registros internos y sus salidas en función de lo que diga cada instrucción.

De esta manera, un microprocesador solo hace esto:
Pedir una instrucción, modificar sus registros y salidas

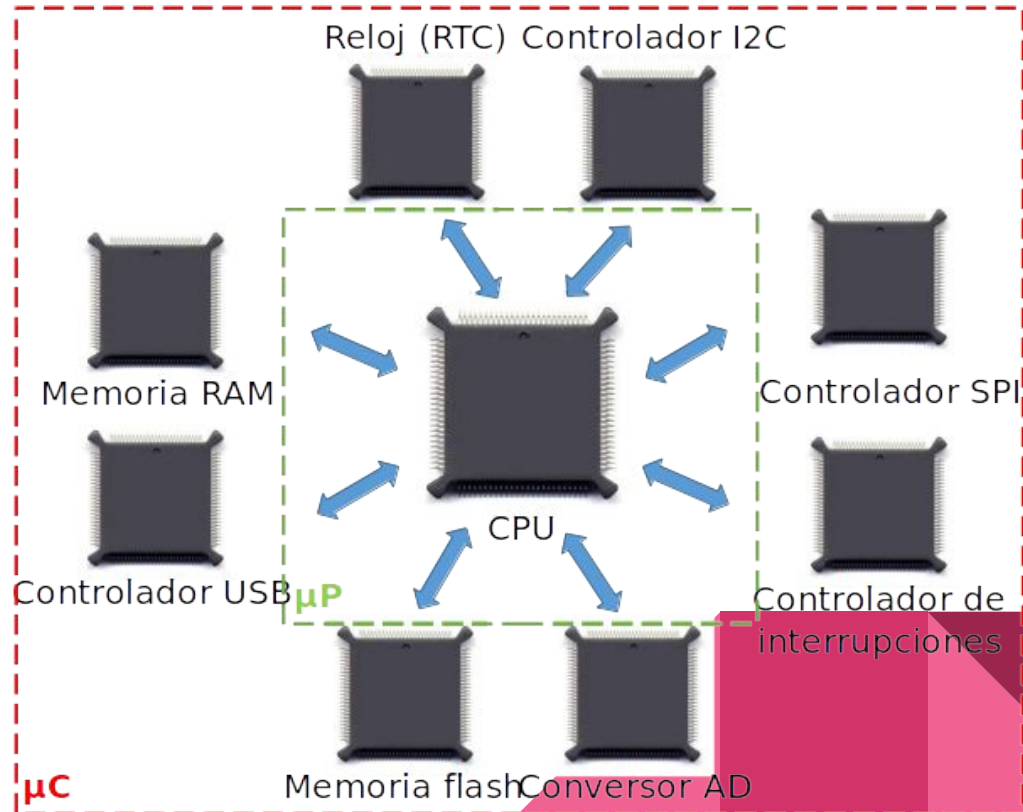


Microprocesadores y Microcontroladores

De la misma manera, un Microprocesador (μP) es un dispositivo que posee una unidad central de proceso (CPU), y una serie de componentes lógicos que permiten enlazarlo con otros dispositivos (memorias, controladores, puertos de E/S, etc.)

Un Microcontrolador (μC), en cambio, integra muchos de los dispositivos dentro del mismo circuito.


Un μP es un sistema abierto, mientras que un μC se desarrolla para un fin más específico, con los dispositivos necesarios para tal fin



Migrando de la PC a un microcontrolador...

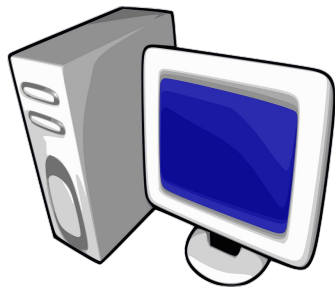
Nuestro objetivo en la segunda parte del año es deshacernos de todos los dispositivos y periféricos que no son esenciales para las aplicaciones que vayamos a realizar, de manera de poder diseñar programas que controlen dispositivos electrónicos que tengan fines específicos (que pueden ir desde autos a control remoto, hasta sistemas de control de acceso, controles inteligentes del hogar, sistemas de telemetría, sistemas para automatizar procesos, etc.)

Desarrollaremos los programas en la PC, y los bajaremos directamente al microcontrolador, con los periféricos mínimos necesarios para que el mismo pueda ejecutar los programas.



Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?

MEMORIA (RAM Y ROM)



Mucha memoria

(Gigabytes) para almacenar programas y datos



Poca Memoria

(Kilobytes) para almacenar programas y datos

Nuestros programas deben tener código corto y eficiente, no utilizar funciones de librería que no sepamos cómo están desarrolladas y debemos optimizar el uso de los datos.

Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?

Ejemplo:

```
float a;  
for (a = 0 ; a < 10 ; a++ )  
{  
    ...  
}
```

```
unsigned char a;  
for (a = 0 ; a < 10 ; a++ )  
{  
    ...  
}
```

¿Es lo mismo?

y...

```
unsigned char a;  
for (a = 10 ; a ; a-- )  
{  
    ...  
}
```



Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?

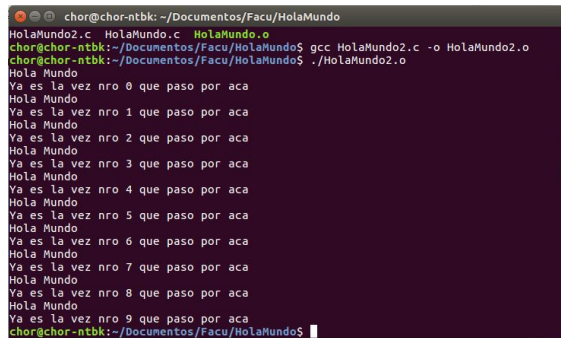
En una PC:

```
#include <stdio.h>

int main (void)
{
    unsigned int a;

    for (a = 0 ; a < 10 ; a ++ )
    {
        printf ("Hola Mundo\n");
        printf ("Ya es la vez nro %d que paso por aca\n",
    }

    return 1;
}
```



```
chor@chor-ntbk: ~/Documentos/Facu/HolaMundo
HolaMundo2.c  HolaMundo.c  HolaMundo.o
chor@chor-ntbk:~/Documentos/Facu/HolaMundo$ gcc HolaMundo2.c -o HolaMundo2.o
chor@chor-ntbk:~/Documentos/Facu/HolaMundo$ ./HolaMundo2.o
Hola Mundo
Ya es la vez nro 0 que paso por aca
Hola Mundo
Ya es la vez nro 1 que paso por aca
Hola Mundo
Ya es la vez nro 2 que paso por aca
Hola Mundo
Ya es la vez nro 3 que paso por aca
Hola Mundo
Ya es la vez nro 4 que paso por aca
Hola Mundo
Ya es la vez nro 5 que paso por aca
Hola Mundo
Ya es la vez nro 6 que paso por aca
Hola Mundo
Ya es la vez nro 7 que paso por aca
Hola Mundo
Ya es la vez nro 8 que paso por aca
Hola Mundo
Ya es la vez nro 9 que paso por aca
chor@chor-ntbk:~/Documentos/Facu/HolaMundo$
```

¿Quién ejecuta mi programa?

¿Cuándo se ejecuta?

Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?

En un sistema embebido:

```
miprogram.c
```

```
void main (void)
```

```
{
```

```
    while ( 1 )
```

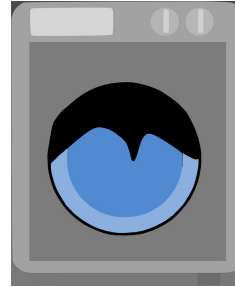
```
    {
```

```
        //Aquí el programa
```

```
    }
```

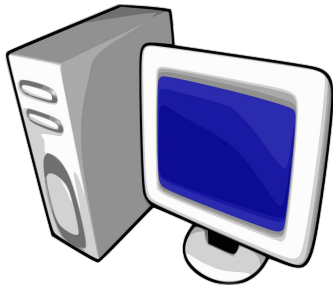
```
}
```

**NO TENGO UN SISTEMA
OPERATIVO**



**¡EL PROGRAMA NO LO INVOKA NADIE Y
NUNCA SE DETIENE!**

Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?



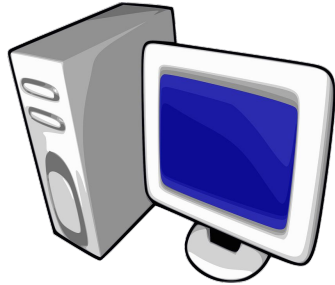
stdio:

printf (...)
scanf (...)
fopen (...)
connect (...)

ACCESO AL HARDWARE EN LA PC

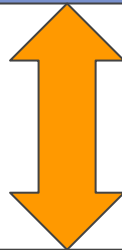
Trabajando en un entorno de desarrollo como la PC, tengo resuelto todo el acceso a los periféricos (al hardware), gracias a un conjunto de funciones (primitivas) que permiten al programador hacer llamadas estándar a estos dispositivos, y a los drivers (invocados por el sistema operativo) que a su vez diferencian el formato de esta llamada según la marca y modelo del hardware conectado

Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?

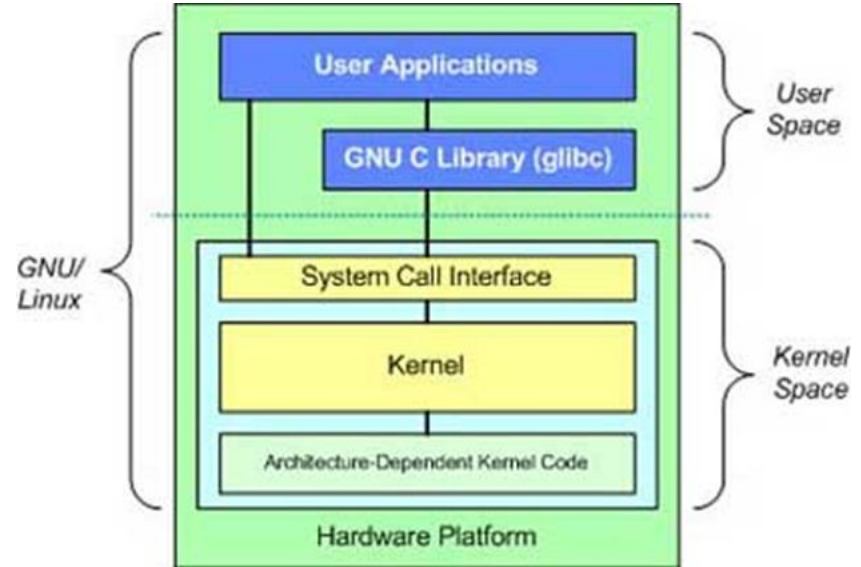


miprogram.c

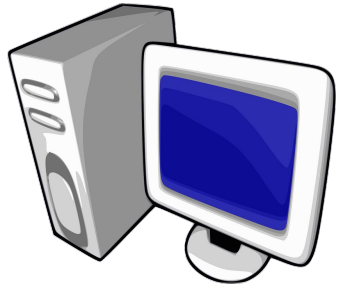
¿Cómo llego?



HARDWARE



Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?



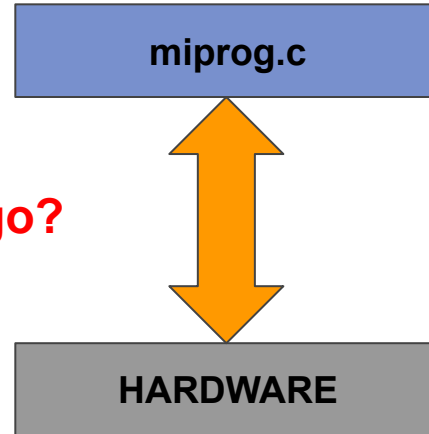
stdio:

printf (...)
scanf (...)
fopen (...)
connect (...)

ACCESO AL HARDWARE EN EL MICRO



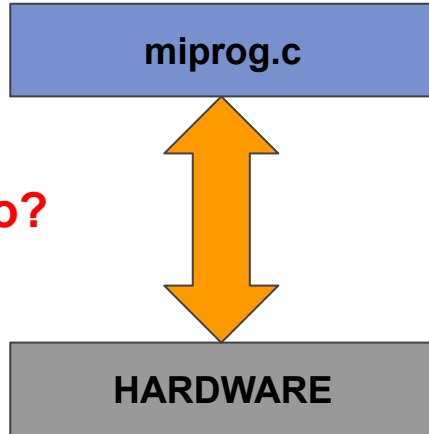
¿Cómo llego?



Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?



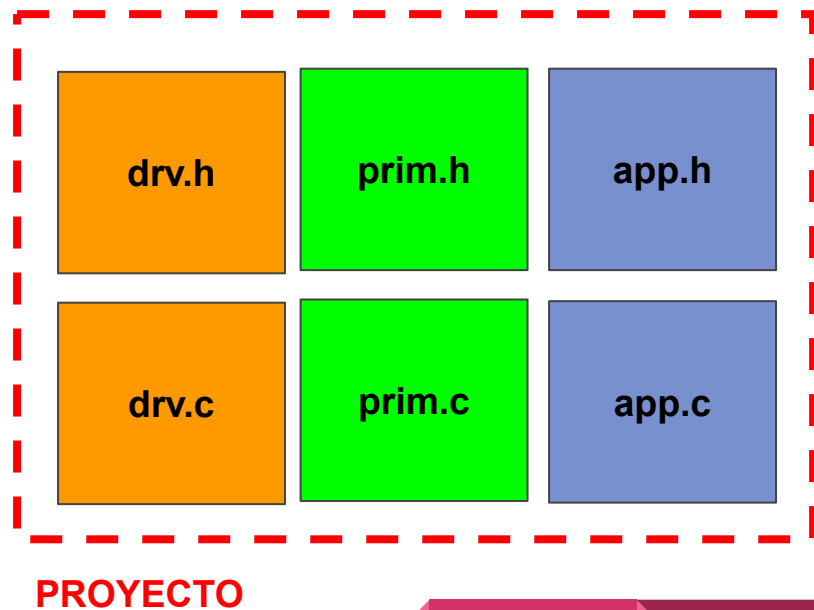
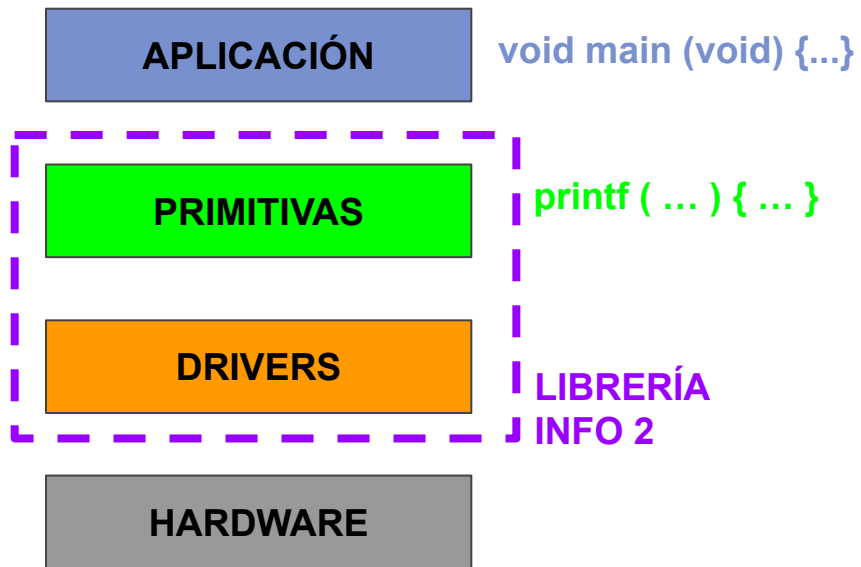
¿Cómo llego?



LO TENEMOS QUE HACER NOSOTROS

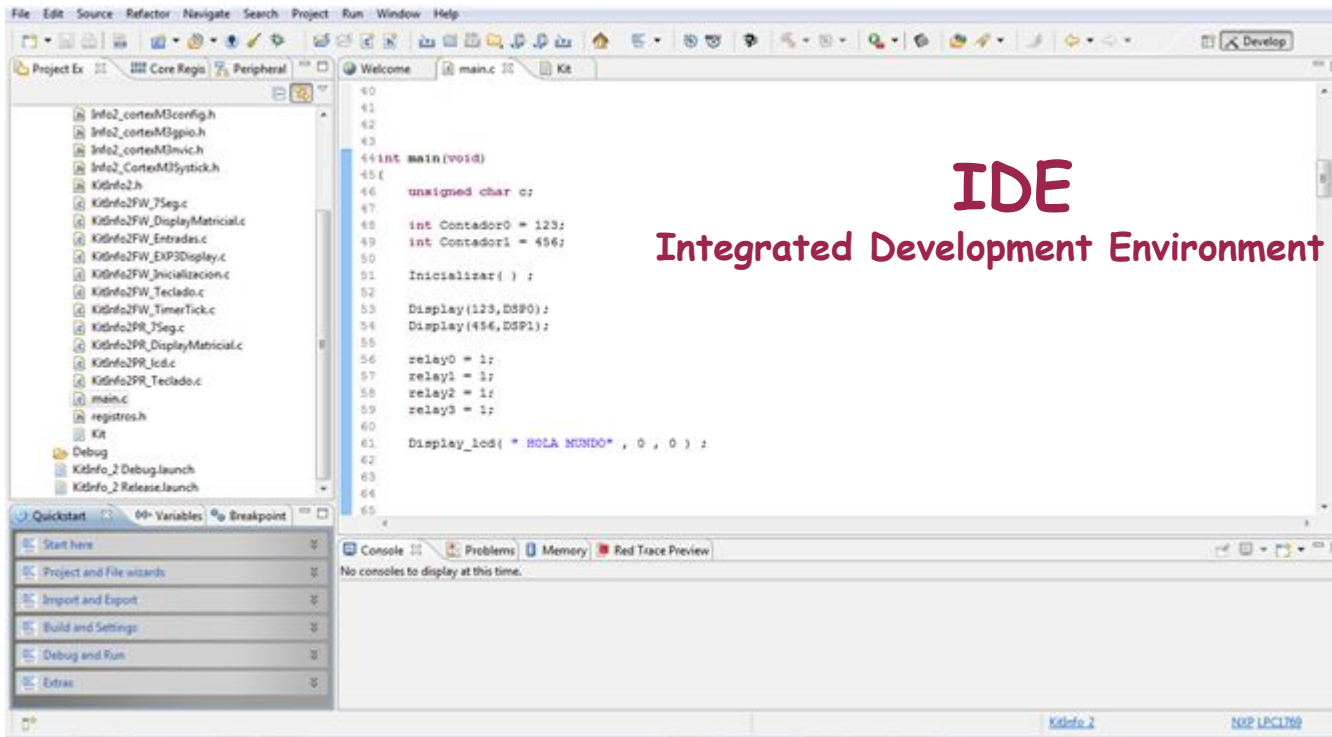
1. Hojas de datos (me dicen cómo tengo que hacer para tocar el Hardware)
2. Desarrollo de DRIVERS (funciones que están diseñadas para manipular el Hardware de un determinado sistema)
3. Desarrollo de PRIMITIVAS (funciones de fácil acceso desde la aplicación que ayudan a que no tenga que modificar mi programa cada vez que cambio el Hardware)

Entonces... ¿Qué diferencias hay entre trabajar sobre una PC que sobre un sistema embebido?



En un primer momento, desde la cátedra les daremos las funciones primitivas, para que solo nos tengamos que concentrar en desarrollar las aplicaciones. En la segunda mitad del año, nos concentraremos en el desarrollo de las primitivas y los drivers para que estas aplicaciones funcionen en entornos reales

¿Donde creo mi proyecto?



Todo integrado

- Editor de texto
- Administración de proyectos
- Compilación y linkeo *para el micropocesador* Cortex M0

Quién es quién en Informática II - Primera Parte

ARM



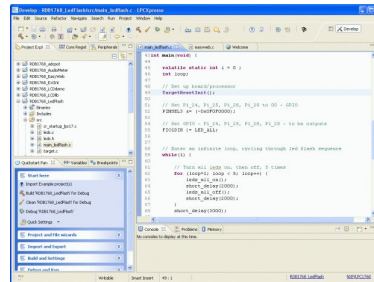
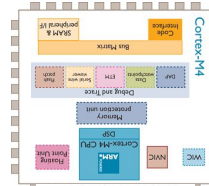
NXP
founded by Philips



Embedded
Artists



NXP
founded by Philips



Microprocesador Cortex M3

Microcontrolador LPC845

Stick LPCXpresso

IDE MCUXpresso

Link de descarga



https://www.nxp.com/design/software/development-software/mcuxpresso-software-and-tools/mcuxpresso-integrated-development-environment-ide:MCUXpresso-IDE?tab=Design_Tools_Tab#nogo

Conectamos Parte I y Parte II en el TPO



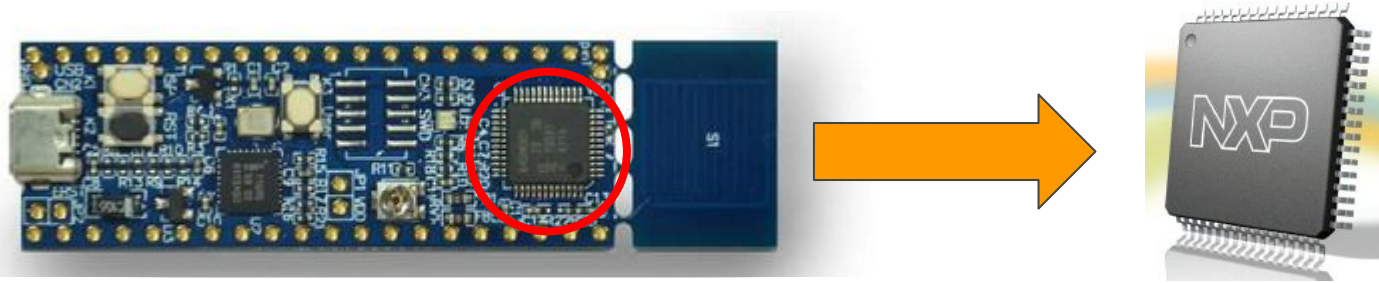
PC



Sistema Embebido

El trabajo práctico obligatorio (TPO) de la materia, buscará poder vincular el desarrollo de aplicaciones gráficas en la PC con la aplicación que desarrollaremos para el control de un sistema embebido, pudiendo entonces controlar el mismo en forma remota.

Stick y LPC 845



[UM11029.pdf](#)

Un sistema embebido es un dispositivo FUNCIONAL (o sea, que cumple con una función), integrado por un microcontrolador y los periféricos necesarios para que el mismo funcione.

Todo dispositivo electrónico (integrado, microcontrolador, sistema embebido) tiene una hoja de datos que nos ayuda a entender como funciona y como interactuar (un manual del usuario)

¿Cómo nos comunicamos con el microcontrolador para descargar un programa?

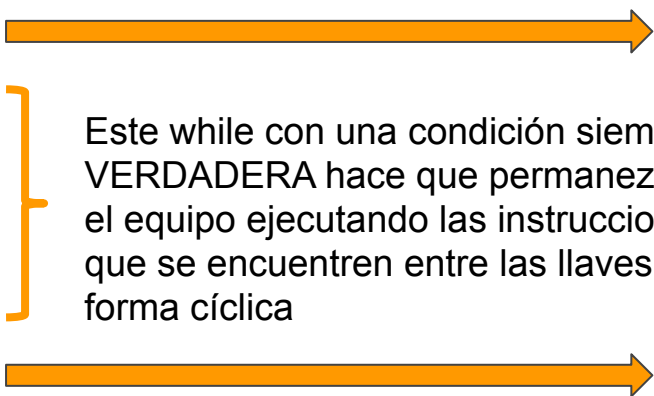


Diseño de un algoritmo para un sistema embebido

1. Funcionamiento continuo

¿Cómo hacemos para que nuestro programa comience al encender el equipo y nunca se detenga?

```
void main (void)
{
    while ( 1 )
    {
        // Instrucciones del bucle
    }
}
```



Punto de arranque del programa (cuando se energiza el sistema el microcontrolador empieza a buscar instrucciones a partir de este punto)

Finalización del programa (a partir de este punto el micro se va a un bucle que vuelve sobre si mismo, o sea que NO HACE NADA)

¿Podemos calcular cuánto tarda el microcontrolador en hacer un ciclo del while?

Diseño de un algoritmo para un sistema embebido

2. Atención “inmediata” de las entradas y salidas

¿Cómo hacemos para que el microcontrolador esté siempre pendiente de lo que está pasando?

```
while ( 1 )  
{  
    while ( GetKey() == ON_TECLA1 ) {  
        PrenderLampara1();  
    }  
    ApagarLampara1();  
  
    while ( GetKey() == ON_TECLA2 )  
    {  
        PrenderLampara2();  
    }  
    ApagarLampara2();  
}
```



¿Funciona?



¿Sigue
Funcionando?

¡¡EN INFO2 NO PODEMOS USAR LOOPS BLOQUEANTES!!

Diseño de un algoritmo para un sistema embebido

2. Atención “inmediata” de las entradas y salidas

¿Cómo hacemos para que el microcontrolador esté siempre pendiente de lo que está pasando?

```
while ( 1 )  
{  
    if (GetKey() == ON_TECLA1)  
        EncenderLampara1();  
    else  
        ApagarLampara1();  
  
    if (GetKey() == ON_TECLA2)  
        EncenderLampara2();  
    else  
        ApagarLampara2();  
}
```



Diseño de un algoritmo para un sistema embebido

¿En qué estado está mi sistema embebido cuando lo energizo?



¡SIEMPRE antes de comenzar nuestro programa debemos inicializar nuestro sistema en un estado seguro! (para el equipo y para los usuarios)

```
void main (void)
{
    Inicializar();

    while (1)
    {

        //Aqui va nuestro programa

    }
}
```