CLASE 12
PROGRAMACIÓN EN ALTO NIVEL

Programación en alto nivel

C es un lenguaje de programación estructurado de propósitos generales. Por su diseño permite "mapear" eficientemente a instrucciones de bajo nivel. Esta es una ventaja en aplicaciones que involucran interacción con hardware (drivers, sistemas operativos, PLCs y sistemas embebidos).

La **biblioteca estándar** de C: https://en.wikipedia.org/wiki/C_standard_library
Estándar: disponible en diferentes implementaciones. ANSI/ISO y POSIX.
Contiene:

- Conjuntos de funciones más utilizadas (math, string, io, etc.)
- Tipos de datos
- **Mecanismos de entrada/salida** ← Sólo la interfaz, hardware/drivers a definir en cada caso

Microchip Studio - ANSI C
AVR: Programación en C y estilo: *"preferred coding style for programming the AVR microcontrollers"*
http://ww1.microchip.com/downloads/en/Appnotes/AVR1000b-Getting-Started-Writing-C-Code-for-AVR-DS90003262B.pdf

"Los lenguajes de programación de alto nivel se han vuelto una necesidad debido al acortamiento de los tiempos de desarrollo y los requerimientos de calidad. Éstos hacen que el código sea más fácil de mantener y reutilizar debido a una mayor portabilidad y facilidad de lectura respecto de las instrucciones de bajo nivel específicas de cada arquitectura. El lenguaje de programación por sí mismo no asegura la facilidad de lectura y la reusabilidad. Ésto se logra con un ***buen estilo de codificación****."*

C standard library

La API (**application programming interface**) de la biblioteca estándar de C (**C standard library**) está declarada en los archivos de cabecera (**header files**).

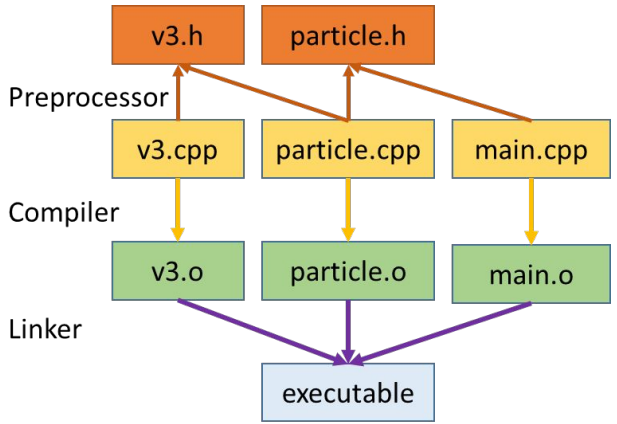
Cada archivo de cabecera contiene una o más declaraciones de:

- Funciones (**function declarations**)
- Tipo de datos (**data type definitions**)
- **Macros**

```
#include <math.h>          // exp, sqrt, sin, rand, etc.
#include <stdio.h>         // archivos, printf, etc.
```

Los headers contienen la “interfaz de programación” o API, número de argumentos, tipos, return value. El compilador los usa para verificar la sintaxis de nuestro programa (**preprocessor**).

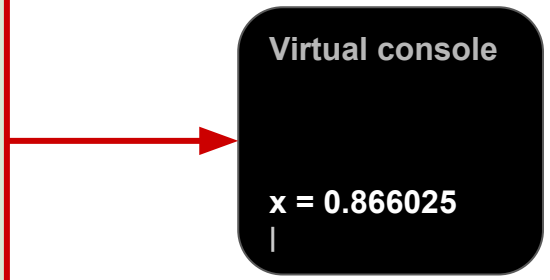
El código de las funciones está en otro archivo que debe ser linkeado con nuestro programa (**linker**).

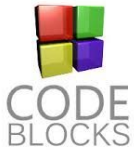


C standard library

Ver en la Práctica 3

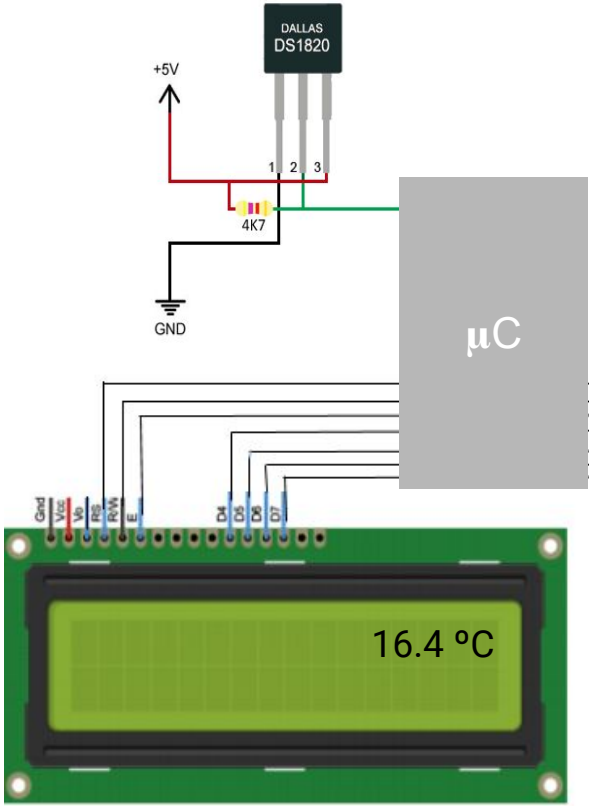
```
#include <stdio.h>
#include <math.h>

int main(void)
{
   float pi = 3.141592;
   float x;

   x = sin(pi/3.0);
   printf("x = %f\n", x);
}
```





Otras bibliotecas

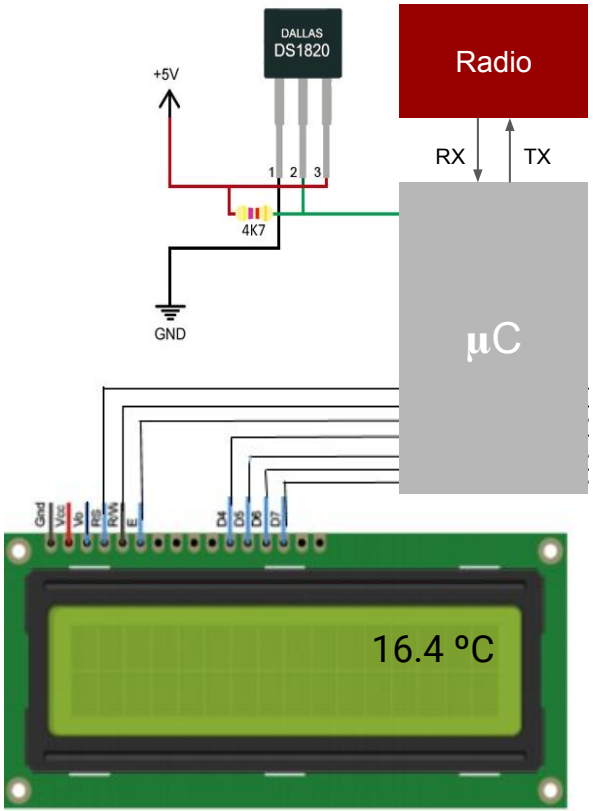
https://github.com/Jacajack/avr-ds18b20




```
#include <ds18b20/ds18b20.h>
#include <displayLCD/displayLCD.h>
#include <util/delay.h>

int main( )
{
	int temp;

	while(1)
	{
		//Start conversion
		ds18b20convert( &PORTB, &DDRB, &PINB, (1<<0), NULL);

		//Delay (sensor needs time to perform conversion)
		_delay_ms(1000);

		//Read temperature
		ds18b20read( &PORTB, &DDRB, &PINB, (1<<0), NULL, &temp);

		displayLCDout(temp);
	}

	return 0;
}
```

Otras bibliotecas

https://github.com/Jacajack/avr-ds18b20




```
#include <ds18b20/ds18b20.h>
#include <displayLCD/displayLCD.h>
#include <util/delay.h>
#include <radio/radio.h>

int main( )
{
	int temp;
	RadioInit();

	while(1)
	{
		//Start conversion
		ds18b20convert( &PORTB, &DDRB, &PINB, (1<<0), NULL);

		//Delay (sensor needs time to perform conversion)
		_delay_ms(1000);

		//Read temperature
		ds18b20read( &PORTB, &DDRB, &PINB, (1<<0), NULL, &temp);

		displayLCDout(temp);
		RadioSend(temp);
	}

	return 0;
}
```

Subiendo un nivel más de abstracción (con cuidado)

El proyecto Arduino: https://www.arduino.cc/

ARDUINO PROGRAMMING LANGUAGE
Basado en Wiring. No es C++.
https://www.arduino.cc/reference/en/

WIRING
“Wiring is an open-source programming framework for microcontrollers”
http://wiring.org.co/

ARDUINO IDE
Arduino Software (IDE), basado en Processing
https://www.arduino.cc/en/Main/Software

PROCESSING
“Processing is a flexible software sketchbook and a language for learning how to code within the context of the visual arts”
https://processing.org/

Sistemas operativos de tiempo real (RTOS)

Materia optativa: Sistemas Embebidos (FreeRTOS sobre Cortex M3)

https://www.freertos.org/
https://en.wikipedia.org/wiki/FreeRTOS

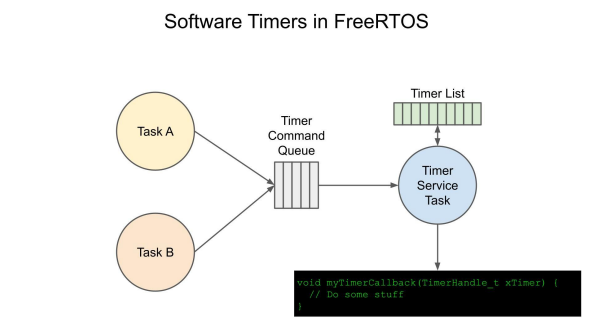
Kernel (3 archivos fuente). Disponible para 35 modelos de microcontroladores.

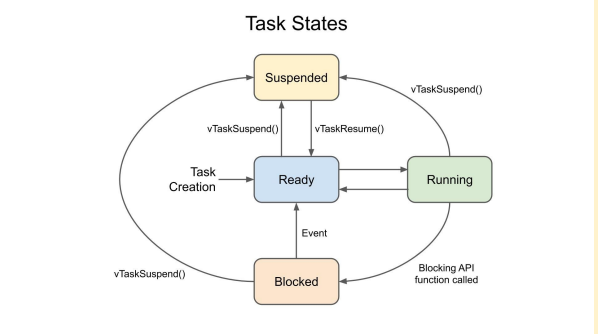
MIT license, Amazon.

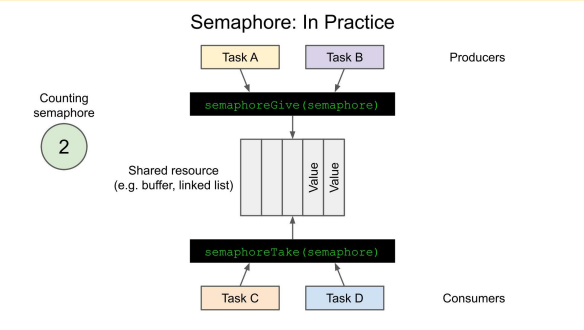
Escrito en C y ensamblador.
Provee métodos para:

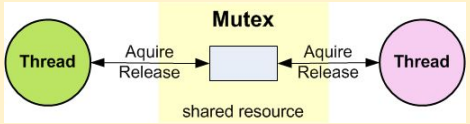
- multiple threads o tareas (prioridades, thread tick method),
- mutexes, semaphores y software timers.

No tiene advanced memory management, user accounts o networking, como Linux. Es más una **thread library** que un sistema operativo.







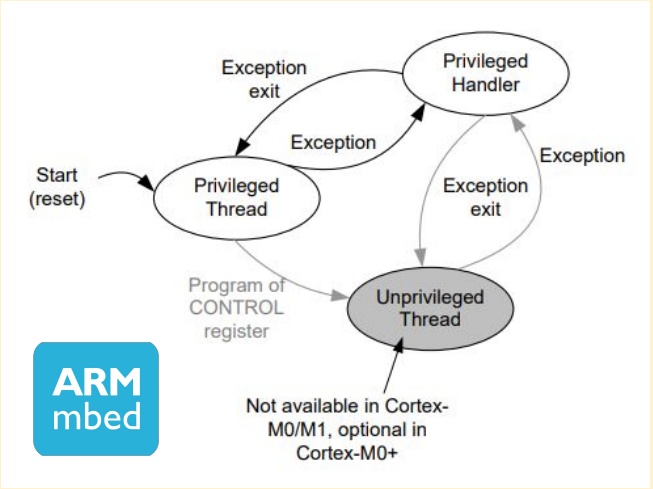


Sistemas operativos de tiempo real (RTOS)

Materia optativa: Sistemas Embebidos (FreeRTOS sobre Cortex M3)

Algunos procesadores (por ejemplo la arquitectura Cortex-M) fueron diseñados pensando en un OS. Algunas prestaciones relacionadas:

- Shadowed stack pointer.
- SVC (supervisor call) and PendSV (pendable service call) exceptions, interfaz para proveer acceso al HW a las aplicaciones.
- SysTick timer – a 24-bit down counter for generating periodic OS exception for time keeping and task management.
- Unprivileged execution level and Memory Protection Unit (MPU) in Cortex-M0+/M3/M4 and M7.



CMSIS-RTOS provee la interfaz a sistemas operativos de terceros: https://os.mbed.com/handbook/CMSIS-RTOS
Tread, osDelay, Mutex, Semaphore, Signals, Message Queue, Memory Pool, Mail Queue, Timer, Interrupt Service Routines, Status and Error Codes, etc.
mbed OS es un sistema operativo desarrollado por ARM usando la interfaz CMSIS-RTOS: https://os.mbed.com/
Mbed Online Compiler (free)