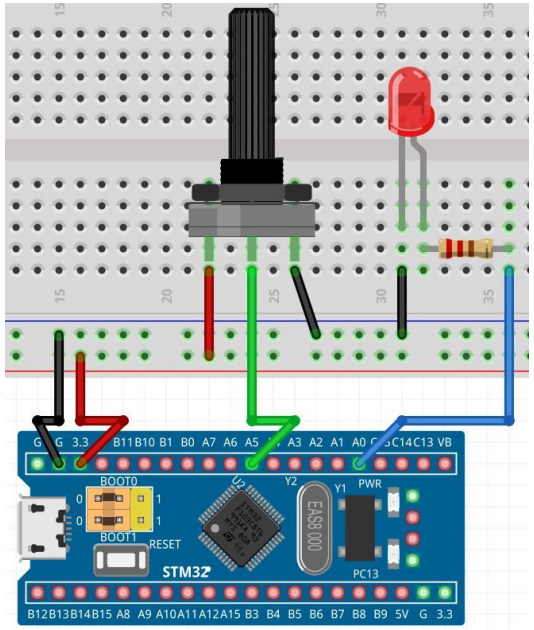

PUCP

PROGRAMACIÓN DE MICROCONTROLADORES ARM

FABRICUM - PUCP

Sesión 8 - 23/07/2024:

- Temporizadores
  - Pre Escaladores
  - Modo PWM
  - Señales en el osciloscopio
- Librerías
  - Conceptos: .h y .c
- Sistemas embebidos
  - Control de brazo robótico



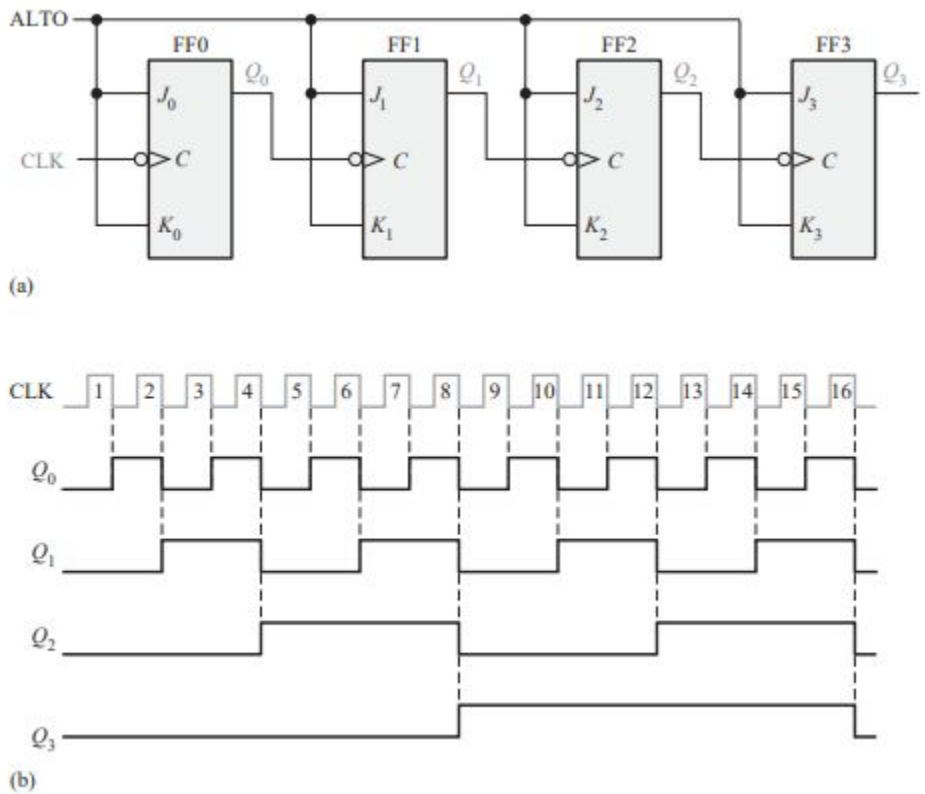
Temporizadores

Programación de microcontroladores ARM - Sesión 8

PUCP

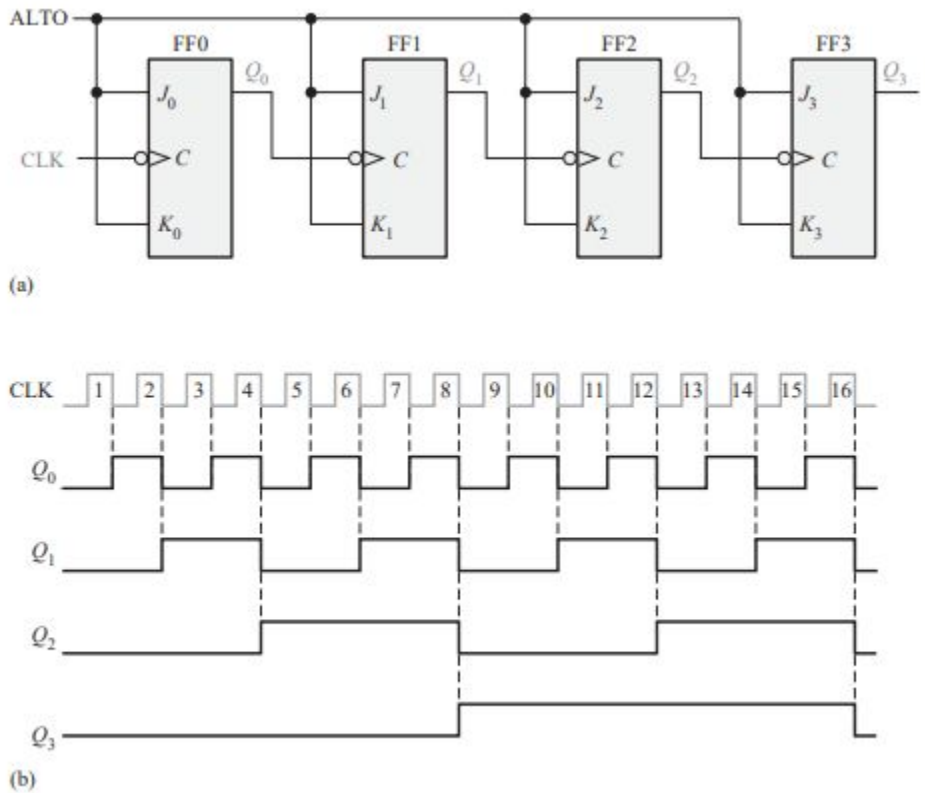
Temporizadores

El concepto de un temporizador tiene como punto de partida el uso de contadores a través de los cuales se pueden generar cierto tipo de ondas. Estos contadores se reinician al finalizar su cuenta y en el caso del microcontrolador STM32F103C8T6 se tiene una resolución de 16 bits para el contador. Dicha cuenta puede ser truncada o configurada a través de los registros del microcontrolador. Como todo circuito digital la operación del contador es síncrona

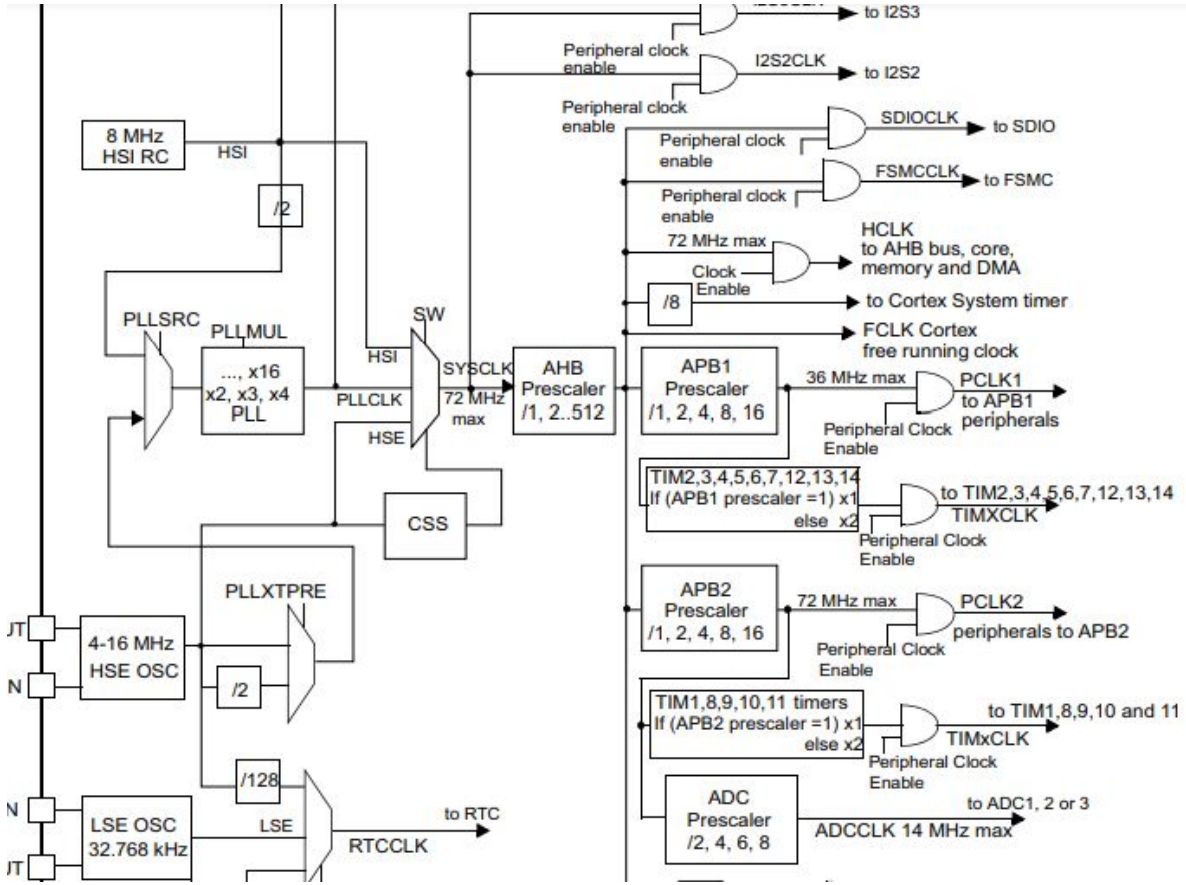


Fuentes de reloj

Para mantener una operación síncrona se necesita de una señal de reloj que pueda llegar al módulo, para esto se tienen circuitos como pre escaladores los cuales permiten dividir la frecuencia de ingreso al módulo para obtener el requerimiento deseado. La fuente de reloj de ingreso es de 8 Mhz y se puede revisar en los registros RCC_CFGR.

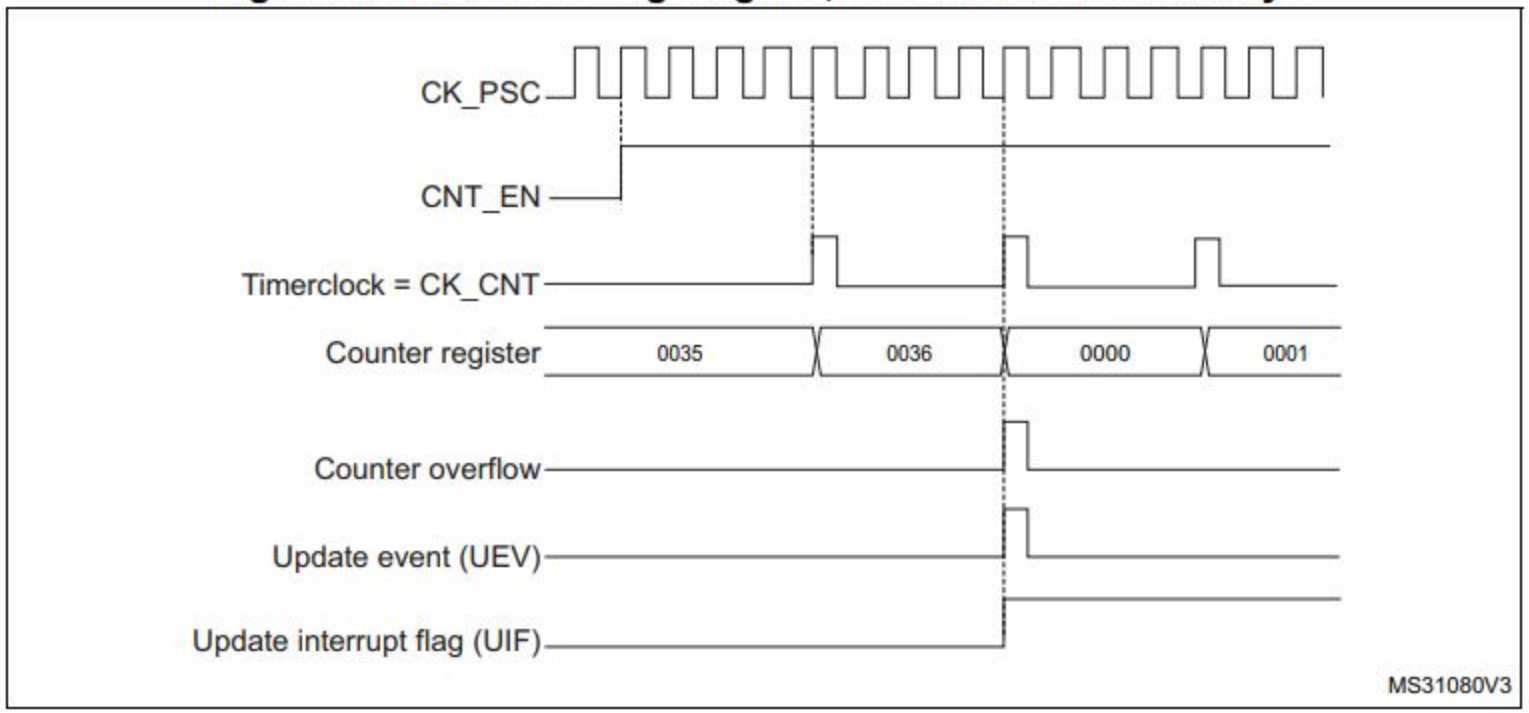


Fuentes de reloj



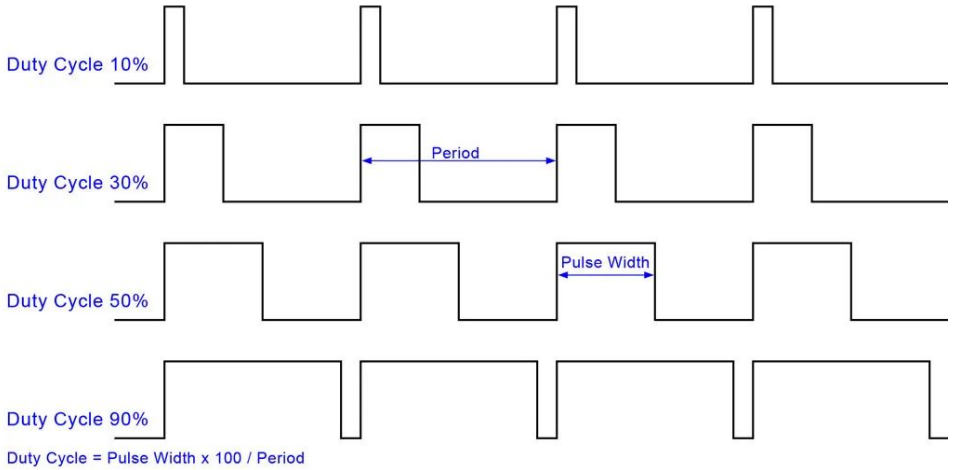
Preescalador - TIMx-PSC

Figure 57. Counter timing diagram, internal clock divided by 4



Modo PWM

Las ondas PWM son señales digitales que a partir de un ciclo de trabajo permiten emular voltajes analógicos. Este tipo de señales son muy utilizadas para control de intensidad, control de velocidad en motores o variadores de frecuencia



Modo PWM

15.3.9 PWM mode

Pulse width modulation mode allows generating a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per OCx output) by writing 110 (PWM mode 1) or ‘111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The user must enable the corresponding preload register by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the auto-reload preload register by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, the user has to initialize all the registers by setting the UG bit in the TIMx_EGR register.

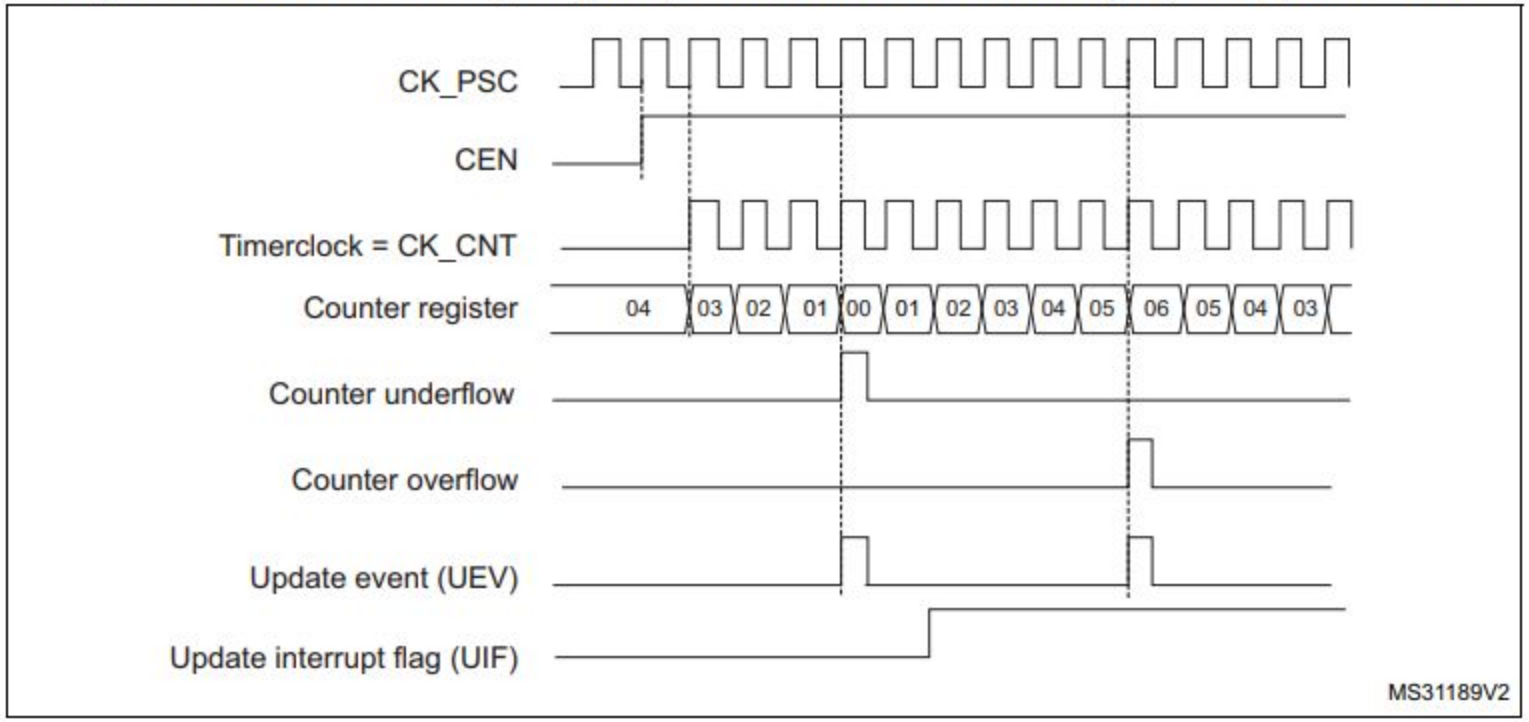
OCx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx≤ TIMx_CNT or TIMx_CNT≤ TIMx_CCRx (depending on the direction of the counter). However, to comply with the ETRF (OCREF can be cleared by an external event through the ETR signal until the next PWM period), the OCREF signal is asserted only:

- When the result of the comparison changes, or
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM=‘000) to one of the PWM modes (OCxM=‘110 or ‘111).

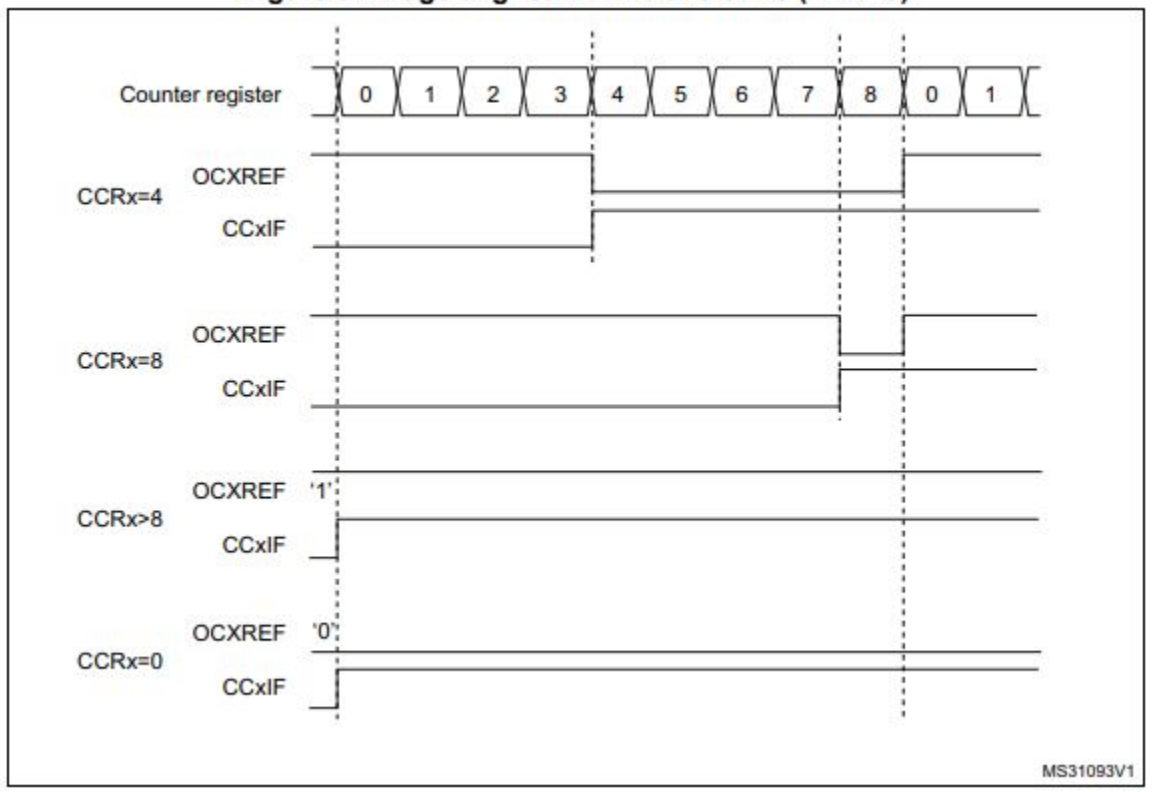
Cuenta en el temporizador - TIMx-ARR

Figure 66. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



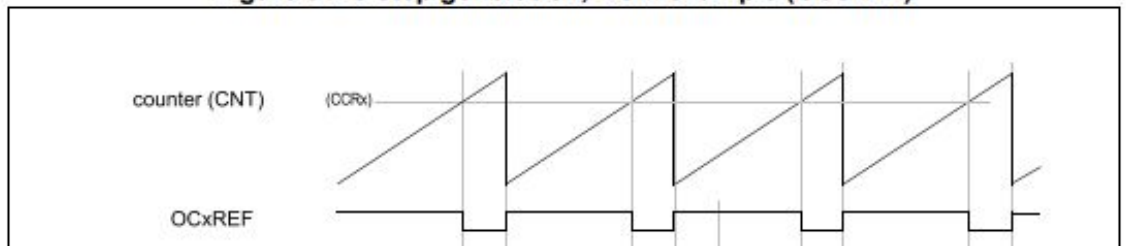
Onda PWM - TIMx-CCRx

Figure 84. Edge-aligned PWM waveforms (ARR=8)



Registros de configuración

Figure 91. 6-step generation, COM example (OSSR=1)



15.4.1 TIMx control register 1 (TIMx_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						CKD[1:0]		ARPE	CMS		DIR	OPM	URS	UDIS	CEN
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

16.4.9 TIM9/12 prescaler (TIMx_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Registros de configuración

15.4.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)

Address offset: 0x18

Reset value: 0x0000

The channels can be used in input (capture mode) or in output (compare mode). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode. For a given bit, OCxx describes its function when the channel is configured in output, ICxx describes its function when the channel is configured in input. Take care that the same bit can have a different meaning for the input stage and for the output stage.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
IC2F[3:0]				IC2PSC[1:0]				IC1F[3:0]				IC1PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15.4.8 TIMx capture/compare mode register 2 (TIMx_CCMR2)

Address offset: 0x1C

Reset value: 0x0000

Refer to the above CCMR1 register description.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
IC4F[3:0]				IC4PSC[1:0]				IC3F[3:0]				IC3PSC[1:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Registros de configuración

15.4.9 TIMx capture/compare enable register (TIMx_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		CC4P	CC4E	Reserved		CC3P	CC3E	Reserved		CC2P	CC2E	Reserved		CC1P	CC1E
		rw	rw			rw	rw			rw	rw			rw	rw

15.4.12 TIMx auto-reload register (TIMx_ARR)

Address offset: 0x2C

Reset value: 0xFFFF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Registros de configuración

15.4.13 TIMx capture/compare register 1 (TIMx_CCR1)

Address offset: 0x34

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15.4.14 TIMx capture/compare register 2 (TIMx_CCR2)

Address offset: 0x38

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15.4.15 TIMx capture/compare register 3 (TIMx_CCR3)

Address offset: 0x3C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

15.4.16 TIMx capture/compare register 4 (TIMx_CCR4)

Address offset: 0x40

Reset value: 0x0000

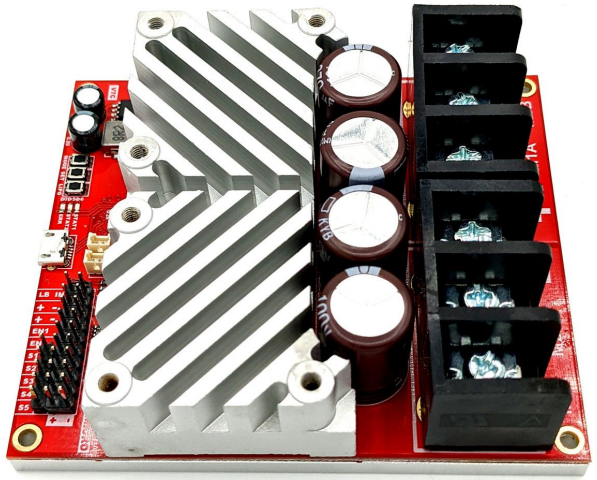
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro	rw/ro

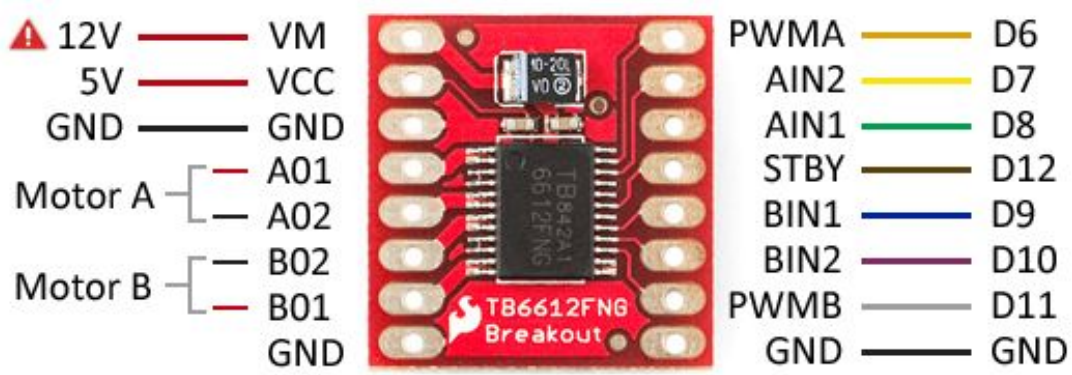
Principales aplicaciones

Las ondas PWM tienen una amplia variedad de aplicaciones relacionadas al accionamiento de motores y control de velocidad. En sistemas de propulsión marina y aérea también se puede apreciar este tipo de ondas. Dentro de los sistemas robóticos, muchos elementos de posicionamiento poseen ondas PWM como interfaces de control.

Control de motores

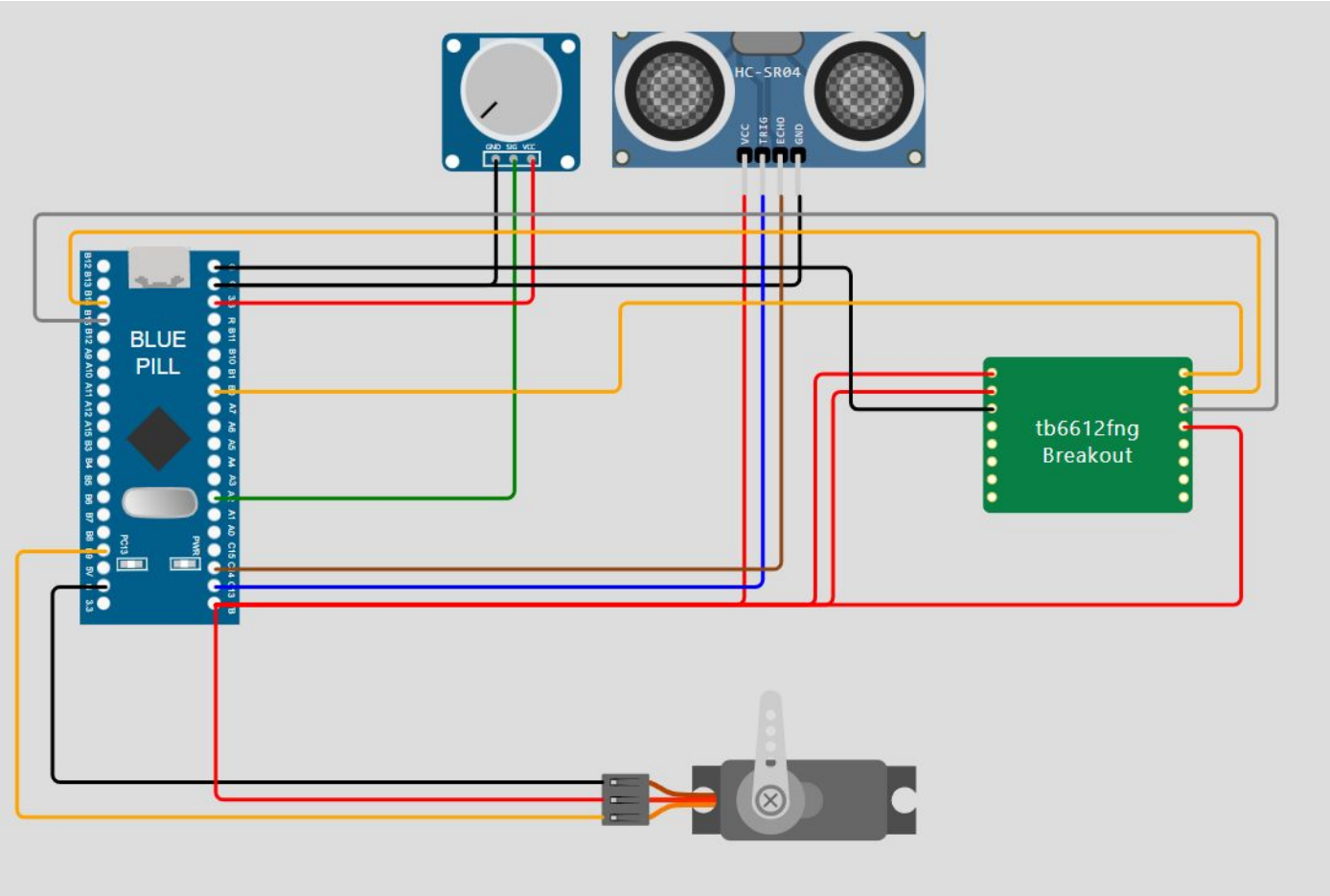
En la actualidad el control de motores se realiza a través de drivers de potencia, los cuales en su circuito de control requieren puertos PWM como interfaz. Dos de los driver muy utilizados con el TB6612FNG y RoboClaw





Vin entre 5 y 15V. Al usar alimentación externa SIEMPRE poner con GND común.

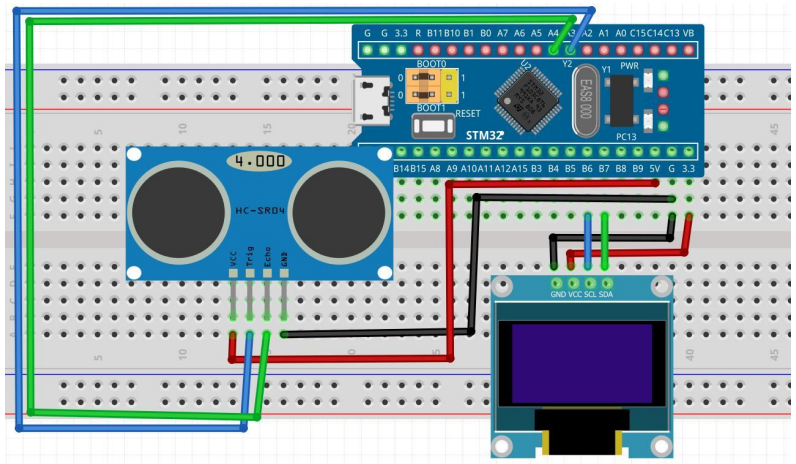
Diagrama esquemático



Ejemplos de aplicación

Para este módulo se desarrollarán los siguientes ejemplos

- Generación de onda PWM
- Control de motor DC
- Control de posición de un servomotor
- Control de un servomotor con un potenciómetro



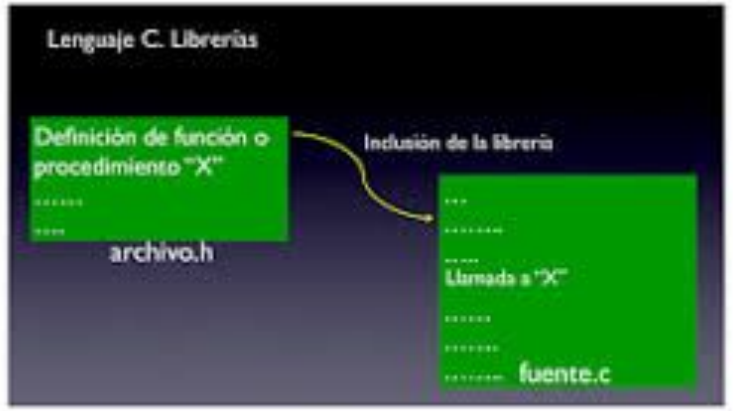
Librerías

Programación de microcontroladores ARM - Sesión 8

PUCP

Librerías en C

Las librerías en C se componen de dos archivos, los .h y los .c. Los archivos de cabecera (.h) definen las funciones y parámetros que serán implementadas en el archivo fuente (.c). No es obligatorio que se implementen ambos archivos, a veces un archivo de cabecera para definir constantes es suficiente como en el caso de la definición de punteros y estructuras para los microcontroladores ARM



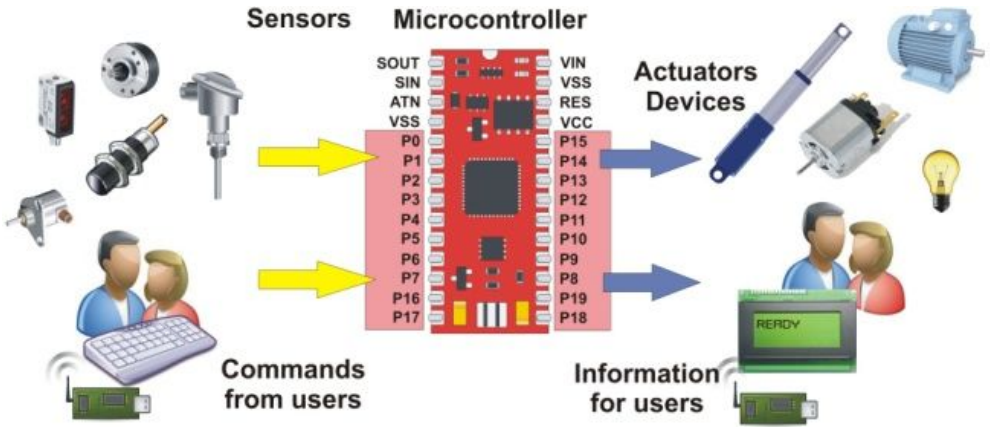
Ejemplos de aplicación

Dentro del curso se han desarrollado las librerías para los periféricos:

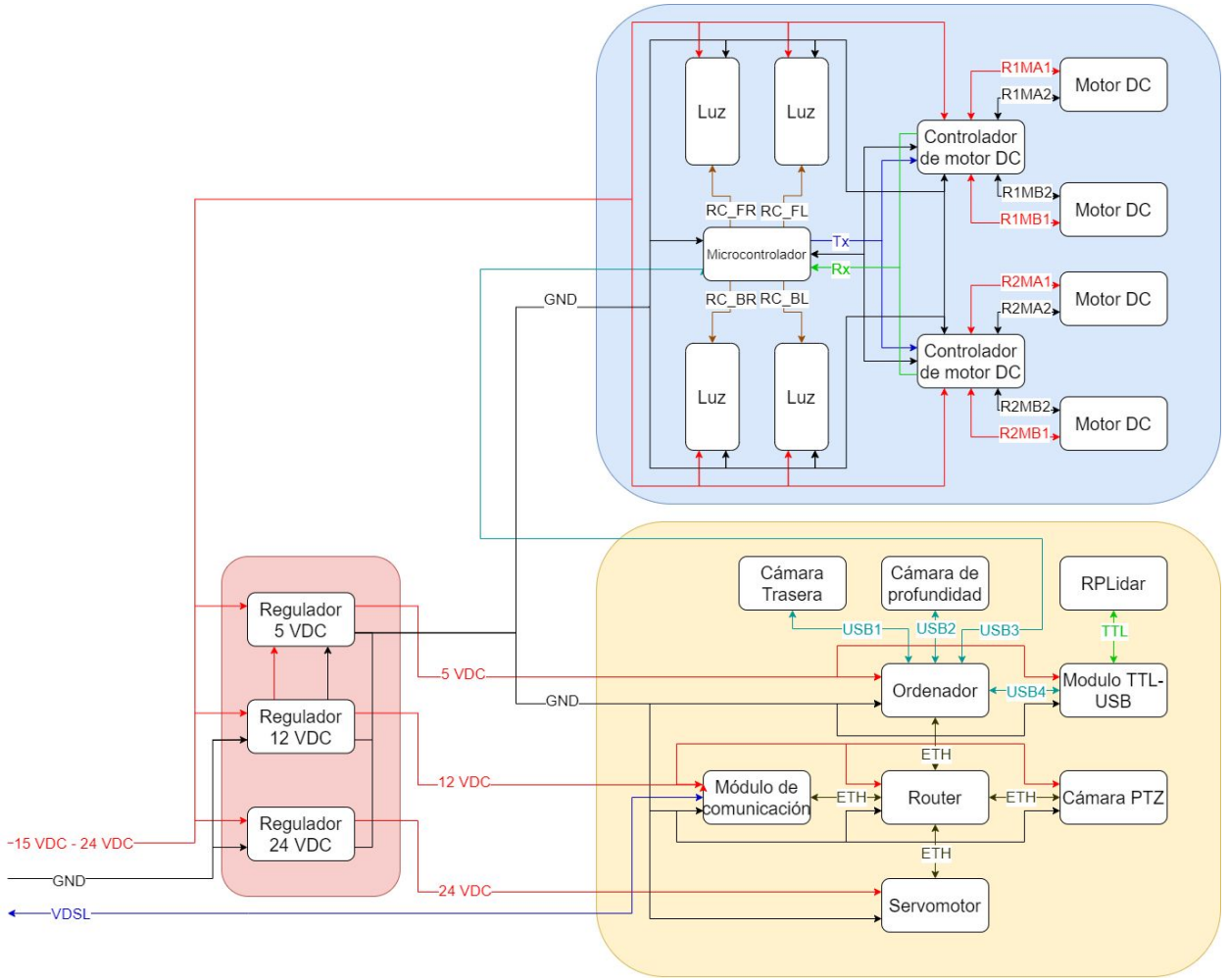
- UART
- Timer
- ADC
- HC-SR04
- BrazoRobotico

Sistemas embebidos

Un sistema embebido sigue un esquema genérico en su construcción en donde se extraen señales o comandos de usuario los cuales son interpretados por el programa de control. Como salida del sistema generalmente se tienen actuadores o pantallas en donde se pueden apreciar los resultados del procesamiento de la información. Es importante tener en cuenta que para la alimentación de actuadores se debe contar con un sistema de potencia y una interfaz que permita recibir las señales de control.



Sistemas embebidos

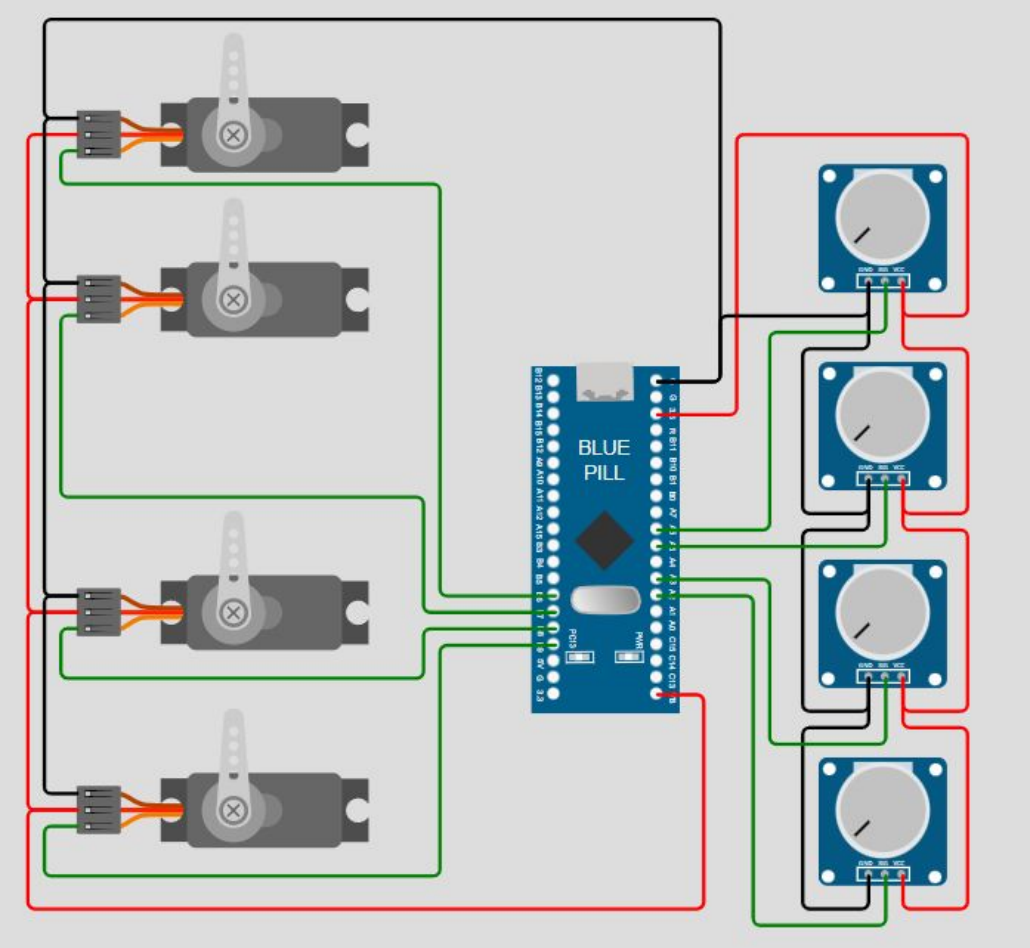


Sistemas embebidos

Proyecto de aplicación

Como actividad final del curso desarrollaremos un control de un brazo robótico, el cual tendrá potenciómetros como sensores de entrada y se buscará controlar los servomotores a través de temporizadores.

Proyecto de aplicación

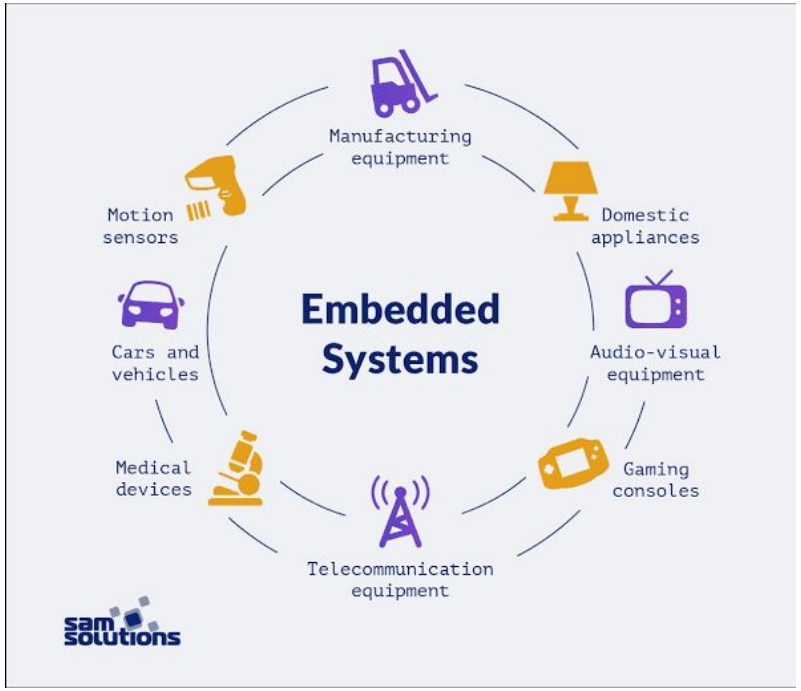


Examen final del curso

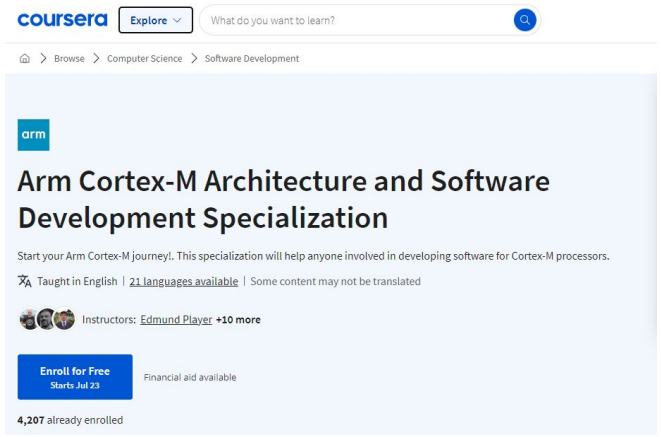
El examen final del curso consiste en la programación de una funcionalidad para un brazo robótico haciendo uso de la terminal serial. Las conexiones y el detalle del examen se encuentran en el documento adjunto en el canvas. El plazo de entrega a través de github es hasta el 30 de julio del presente año.

¿Qué sigue ahora?

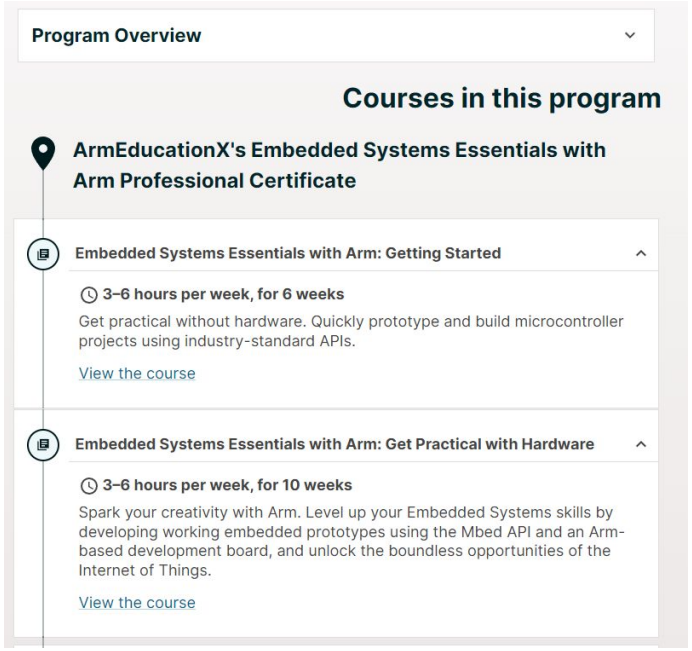
En el presente curso se han revisado los principales tópicos en la programación de sistemas embebidos tomando como referencia el microcontrolador STM32F103C8T6 pero con conceptos que se aplican a cualquier otro chip de cualquier fabricante. Este es un punto de partida, el cual puede ser reforzado con diferentes recursos disponibles para desarrollarse académicamente y profesionalmente. A nivel nacional es un campo naciente ya que Perú no es un país que exporte tecnología; sin embargo esto también puede ser visto como una ventaja ya que aún queda mucho por desarrollar y automatizar.



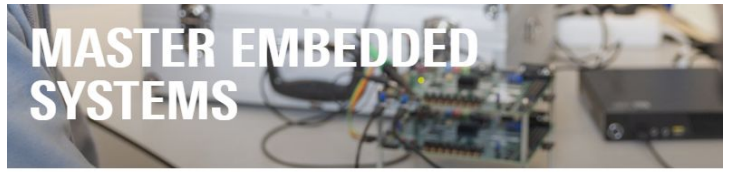
¿Qué sigue ahora?



https://www.coursera.org/specializations/cortex-m-architecture-and-software-development



https://www.edx.org/certificates/professional-certificate/armeducationx-embedded-systems-essentials

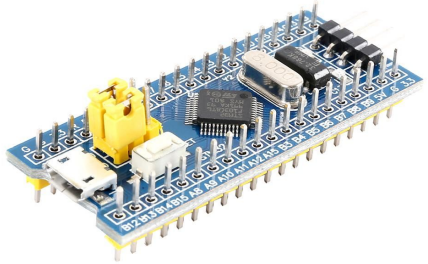


CONTRIBUTE TO **ADVANCED SOLUTIONS IN INDUSTRIAL PROCESSES BY DESIGNING THE SOFTWARE AND HARDWARE OF COMPLEX EMBEDDED SYSTEMS** IN VEHICLES, PACEMAKERS, CHIPSETS, AND OTHER DEVICES.

The design of embedded systems is at the core of technological and industrial progress. Embedded systems are building blocks of medical devices, automobiles, industrial machinery, and GPS systems. Even your car's antilock braking system (ABS) is an embedded system. The design of such systems is crucial for their functionality. If an embedded system can't perform the required task in time with the required quality, this has an immediate effect on the safety of the person. Furthermore, cost- and energy efficiency are crucial. Within **the Master's in Embedded Systems at the University of Twente**, you will learn to design, build, and program intelligent software and hardware that comply with strict requirements concerning time latency, power consumption, reliability, and cost efficiency.

https://www.utwente.nl/en/education/master/programmes/embedded-systems/

PUCP



GRACIAS