# PROYECTO DE MICROPROCESADORES: BRAZO ROBÓTICO

Jorge Bennasar Vázquez, Enrique Alonso Álvarez

El proyecto consiste en un brazo robótico que se encarga de transportar objetos de un punto a otro. Un sensor de presión analógico detecta si hay un objeto disponible para recoger y, a continuación, el brazo se encarga de cogerlo y moverlo al sitio deseado. El sistema, por tanto, cuenta con cuatro estados: robot en posición inicial (reposo) (1), objeto disponible para recoger (2), objeto recogido (3) y objeto depositado (4). Adicionalmente, se envía el estado del microprocesador al ordenador (que funciona en C++) por comunicación UART. Por otro lado, unos LEDs indican si hay un objeto disponible para recoger o depositado.

## I. HARDWARE

El hardware del proyecto es el mostrado a continuación en la *Figura 1*:

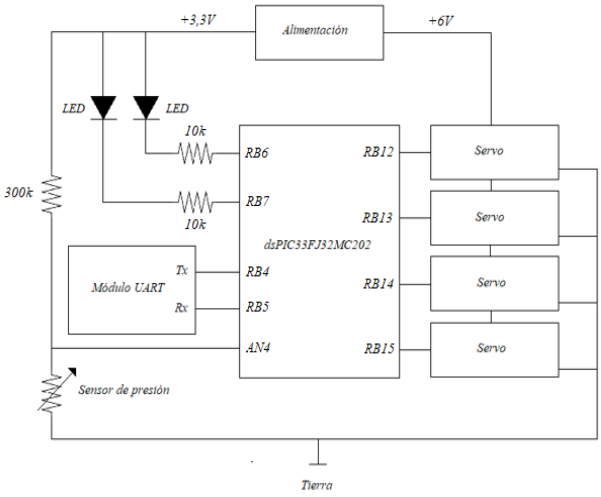


**Figura 1: esquema del diseño hardware del proyecto**

Como se puede observar en el esquema, hemos usado los siguientes componentes:

- Tres servos MAXPRO 3003 (20 ms, 0,5-2,5 ms) para el giro y las articulaciones del brazo.
- Un servo SG90 (20 ms, 1-2 ms) para el accionamiento de la pinza.
- Un sensor de presión analógico.
- Dos diodos LED (uno verde y otro rojo).
- Resistencias ($2 \times 10\ k\Omega$ y $2 \times 150\ k\Omega$ (en serie $300\ k\Omega$)).
- Microprocesador dsPIC33FJ32MC202 (con módulo UART, conversor AD…).

Además, hemos necesitado una alimentación externa a $6\ V$ debido a las exigencias de corriente de los servos.

## II. SOFTWARE A NIVEL DE MICRO

El software a nivel de micro está desarrollado en lenguaje C. El código se ha estructurado de la siguiente forma:

- Un main (*main.c*) en el que están programados el timer 1 y la máquina de estados.
    - El timer 1 produce una interrupción cada $0{,}125\ ms$ y va incrementando un contador (*contador_servos*) hasta que este alcanza el valor de $160$ ($160 \times 0{,}125\ ms = 20\ ms$). De esta manera, y con la ayuda del vector *ancho_pulso*, se asignará a las salidas RB12-RB15 los pulsos necesarios para que los servos hagan los movimientos deseados en cada momento del funcionamiento. Además, el timer también tendrá otro contador (*contador_pinza*) que se encarga de administrar el tiempo que duran los estados 2, 3 y 4.
    - La máquina de estados consiste en cuatro estados:
        - Robot en posición inicial (reposo) (1): el robot se encuentra en este estado hasta que se detecta un objeto en la zona de recogida. En ese momento pasa al estado 2.
        - Objeto disponible para recoger (2): se enciende el LED 1 y, tras haber pasado un tiempo determinado en el que el robot ya debe de haber recogido el objeto, se pasa al estado 3.

- Objeto recogido (3): se apaga el LED 1 y, cuando ya se ha depositado el objeto (tras un tiempo determinado), el sistema pasa al estado 4.

- Objeto depositado (4): se enciende el LED 2 y, después de que el robot haya vuelto a su posición de reposo, se pasa al estado 1, apagándose el LED 2.

- Varios drivers:

  - Driver *config* (*config.h* y *config.c*): se encarga de la configuración general.

  - Driver *servos* (*servos.h* y *servos.c*): inicializa el timer 1 (*init_servos()*) y actualiza las posiciones de los servos en función del estado (*estado_actual*) y el *contador_pinza* (*ancho_pulso[0] = get_servo_1(estado_actual, contador_pinza)*…).

  - Driver *ad* (*ad.h* y *ad.c*): inicializa el convertidor AD (*init_ad(0x3F)*) y se encarga (en el estado 1) de detectar si hay un objeto en la zona de recogida (*detector_objeto = get_ad(4)*).

  - Driver *uart* (*uart.h y uart.c*): inicializa (*InicializarUART()*) y se encarga de las comunicaciones UART con el ordenador.

El esquema de la organización software se muestra en la *Figura 2*:

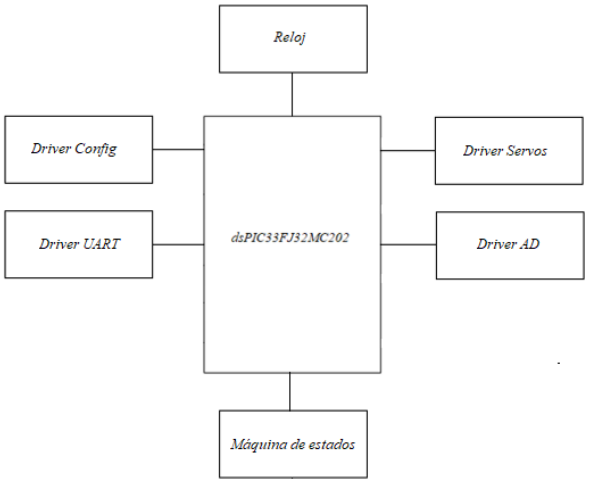


**Figura 2: esquema del diseño software a nivel de micro**

El diagrama de estados sería (*Figura 3*):

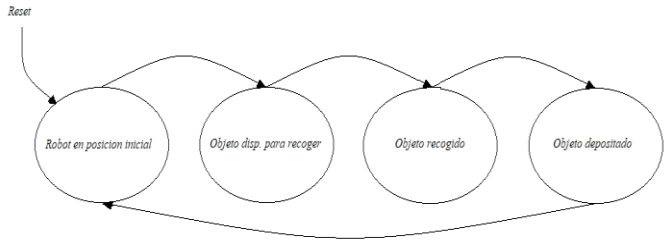


**Figura 3: diagrama de estados**

## III. SOFTWARE A NIVEL DE ORDENADOR

Para desarrollar el software a nivel de ordenador se ha utilizado el lenguaje C++. Para ello, hemos hecho uso de una clase (*CSerial* (*serial.h* y *serial.cpp*)) y un main (*UartRobot.cpp*). Sus funciones son:

- La clase *CSerial* se encarga de organizar las comunicaciones UART entre micro y ordenador.

- El main *UartRobot.cpp* se encarga de mostrar el estado del robot (1, 2, 3 o 4) por pantalla siempre que este haya cambiado.

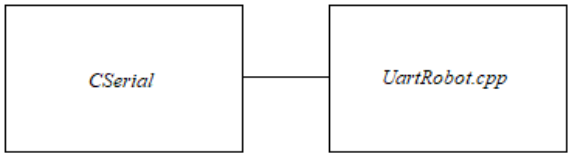


**Figura 4: organización del código en C++**

## IV. TRABAJO MECÁNICO

Para la creación del brazo robótico utilizamos una impresora 3D. Tras ello, nos encargamos de perfeccionar las piezas (6 en total: una base, una pieza de giro, dos articulaciones y una pinza dividida en dos partes) con la ayuda de lijas y una taladradora. A continuación, juntamos las piezas con pasadores (creados a partir de cuerdas de piano de acero) y prisioneros. Posteriormente unimos el brazo a una plataforma e hicimos los arreglos necesarios para que los servos hicieran la función que les correspondía. Por ultimo colocamos los puntos de recogida (con el sensor) y depositado. La idea inicial fue la mostrada a continuación (*Figura 5*):

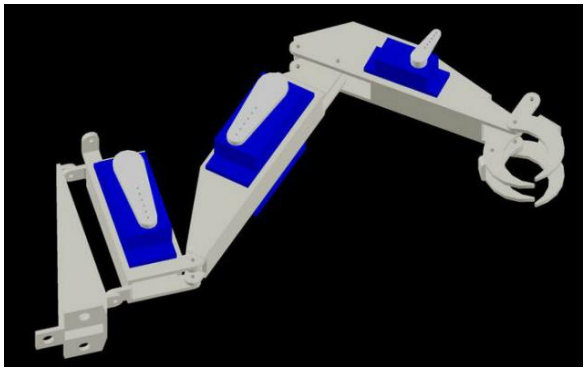

**Figura 5: idea original para el diseño del brazo robótico**

El resultado final, por otro lado, fue:

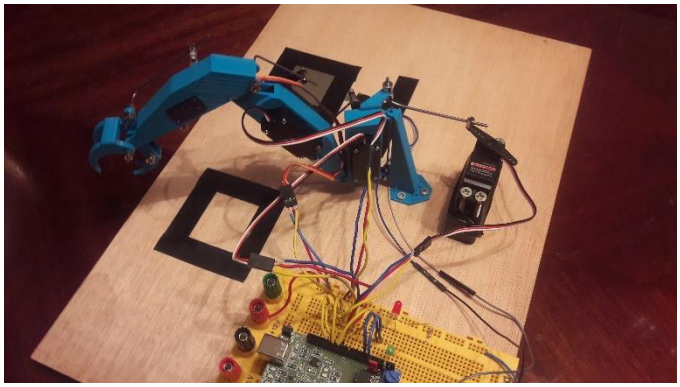

**Figura 6: resultado final del proyecto (1)**

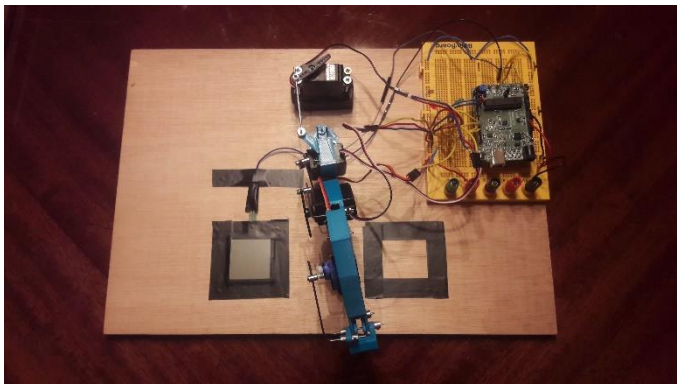

**Figura 7: resultado final del proyecto (2)**

## V. PLANIFICACIÓN

Nuestra planificación a la hora de hacer el proyecto fue la siguiente (*Tabla 1*):

| Tarea | Persona | 1ª semana | 2ª semana | 3ª semana |
|---|---|---|---|---|
| Driver *ad* | Jorge y Enrique | X | | |
| Driver *servos* | Jorge | X | | |
| Driver *uart* | Enrique | | | X |
| Main (máquina de estados y timer 1) | Jorge | | X | |
| C++ | Enrique | | | X |
| Trabajo mecánico | Jorge | | | X |

**Tabla 1: planificación**

## VI. CONCLUSIONES

A lo largo de este proyecto hemos aprendido a usar con fluidez la mayoría de herramientas aprendidas a lo largo del semestre en la asignatura:

- En el lenguaje C: timers e interrupciones, módulo UART, convertidor AD…
- En el lenguaje C++: utilización de clases y comunicación UART con el microprocesador. Mostrar información del funcionamiento del sistema al usuario en vivo.

Además, hemos repasado temas dados en asignaturas anteriores como, por ejemplo, máquinas de estado.

Adicionalmente, nos hemos familiarizado con el trabajo mecánico que supone desarrollar proyectos relacionados con la robótica.