GESCOM_MEGA2560 V3

GESTOR DE COMANDOS

VERSIÓN PARA ARDUINO MEGA 2560

Fecha Doc.: **01 de marzo de 2016**
Versión: 3.00

1 TABLA DE CONTENIDO

2 LIBRERÍA GESCOM_MEGA2560

La librería GESCOM (Gestor de Comandos), versión 3.0 implementa un gestor de comandos de propósito general que permite enviar comandos entre dispositivos (Robots, PCs etc).
Esta librería se ha creado con el objeto de que sea independiente de la implementación física de las operaciones que ejecute cada comando, de forma que el gestor se pueda utilizar en cualquier Dispositivo.

Esta versión está adpatada para que se pueda utilizar con la placa Arduino MEGA2560 porque permite la utilización de cualquier puerto serie.

2.1 ABREVIATURAS Y DEFINICIONES

- **GC, Gestor de comandos.**
  Es el gestor de comandos descrito en este documento.

2.2 DOCUMENTOS RELACIONADOS

Documentos relacionados con Gescom_MEGA2560

- **Gescom_MEGA2560_V3.mdb**
  Base de Datos Access para creación y gestión de comandos.
  Esta BBDD permite también la obtención de los dos ficheros (*Gescom_MEGA2560_V3_CMD.h* y *Gescom_MEGA2560_V3_FProceso.cpp*) que implementan los comandos definidos en la BBDD y que deben ser incluidos en la librería Gescom del directorio de librerías del IDE de Arduino cada vez que se actualiza/crean comandos.

- **Instrucciones_BBDD.doc**
  Documento word con instrucciones de uso de la BBDD Gescom_MEGA2560_V3.mdb

- **Instrucciones_Libreria.doc**
  Documento Word para creación de la librería Gescom_MEGA2560 que se debe copiar en el directorio de librerías del IDE de Arduino.

3 COMANDOS

GC se comunica mediante comandos que se envian por un puerto serie.
Después de la ejecución de un comando, GC envía una respuesta, si se produce algún error en la transmisión de los caracteres la función que implentan el gestor *exeGesComando()* retornará un error.

Un comando es una secuencia de caracteres ASCII que al ser recibidos se procesan y dan como resultado la llamada a una cierta función.

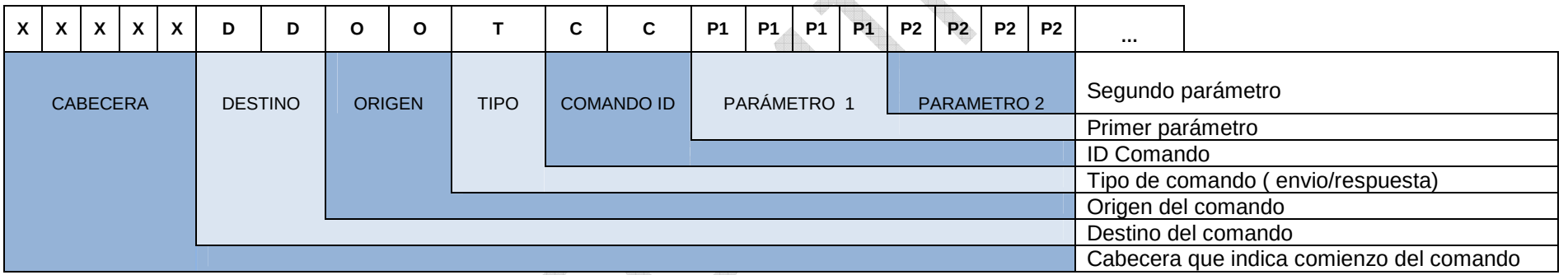
Para mantener activo el GC es necesario llamar a la función *exeGesComando()* por lo que un buen lugar para colocar la llamada a esta función es la función *loop()* de los programas de Arduino.

3.1 DEFINICIÓN DE COMANDO

Seguidamente se muestra la sintáxis de un comando (comando de envío) y la respuesta que se recibe desde el GC después de su ejecución (comando respuesta), es importante destacar que después de la ejcución de un comando, el GC siempre genera un comando de respuesta.
Los comandos son secuencias de caractéres ASCII de longitud fija (20 caractéres), salvo para comandos que envíen datos adicionales.

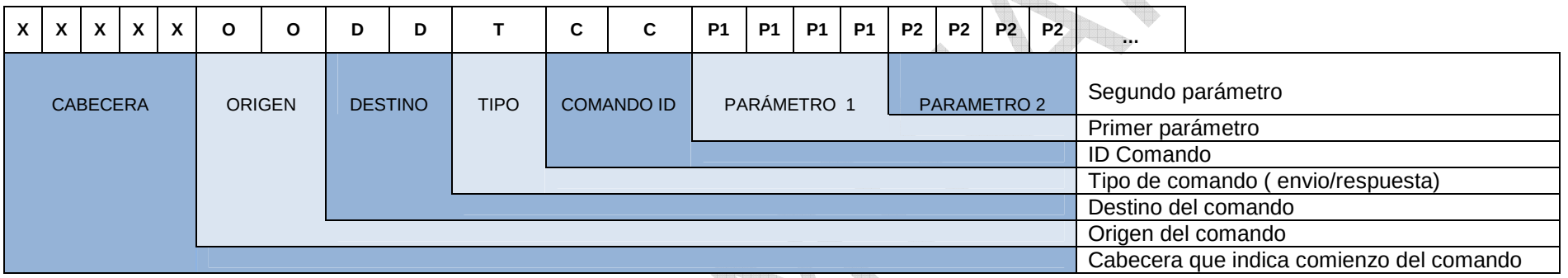
3.1.1 COMANDOS DE ENVÍO

Los comandos que se envían desde el sistema de control al dispositivo para su ejecución tienen la siguiente estructura,



3.1.2 COMANDOS DE RESPUESTA

Los comandos que se envían desde el dispositivo hacia el sistema de control (respuesta de ejecución) como resultado de la ejecución de un comando de envío tienen la siguiente estructura,



En ambos casos, (envío y respuesta) los diferentes parámetros tienen el siguiente significado:

TIPO	DESCRIPCIÓN	TIPO
X	Carácter # (almohadilla), ASCII 35	char (byte)
D	Identifica el receptor del comando	char (byte)
O	Identifica quien genera el comando	char (byte)
T	Identifica el tipo de comando	char (byte)
C	Identificador del comando	char (byte)
C	Identificador del comando	char (byte)
P1	Parametro 1 (*) ver detalle, en tabla siguiente	char (byte)
P2	Parametro 2 (*) ver detalle, en tabla siguiente	char (byte)

- **X**, cabecera
  5 caracteres que indican el comienzo de un comando, son **siempre** caractéres # (almohadilla), ASCII 35.

- **D**, destino
  2 caracteres que indican en hexadecimal el dispositivo **destino** al que se dirije el comando.

- **O**, origen
  2 caracteres que indican en hexadecimal el dispositivo **origen** que envía el comando.

Nota:
(ver la tabla "Tbl_Dispositivos" en *gescom_MEGA2560_V3.mdb*)

- **T**, Tipo
  1 caracter que indica:
  0 Comando de envío
  1 Comando de respuesta

- **C**, comando ID
  2 caracteres que identifican en hexadecimal el comando.

Nota:
(ver la tabla "Tbl_Comandos" en *gescom_MEGA2560_V3.mdb*)

- CARÁCTER **P1**, parámetro 1.
  4 carcteres que identifican en hexdecimal el parámetro 1.

  Importante:
  Si es un comando de envío: **valor**
  Si es un comando de respuesta: **0**

Nota:
(ver la tabla "Tbl_Parametros" en *gescom_MEGA2560_V3.mdb*)

- CARÁCTER **P2**, parámetro 2.
  4 carcteres que identifican en hexdecimal el parámetro 2.

  Importante:
  Si es un comando de envío: **valor**
  Si es un comando de respuesta: **0** Si el comando no genera datos de respuesta
  Valor longitud en bytes del string que retorna el comando como respuesta.
  La longitud MÁXIMA que se puede retornar es de IDE_MAXL_RESP_CMD (actualmente fijado en 25 bytes), este define se encuentra en Gescom_MEGA2560_V3.h

Nota:
(ver la tabla "Tbl_Parametros" en *gescom_MEGA2560_V3.mdb*)

El mecanismo de funcionamiento es el siguiente:

- El comando se envía o recibe utilizando la comunicación serie, através del puerto indicado en el constructor del objeto,

```
#define IDE_SERIAL_0
#define IDE_SERIAL_1
#define IDE_SERIAL_2
#define IDE_SERIAL_3
```

- El Gestor de comandos recibe el comando, comprueba su sintáxis y una vez determinado que es un comando válido llama a una determinada función de usuario para procesarlo.

En la carpeta de "Gescom_MEGA2560_V3\examples", existe un fichero .ino con una función de interfaz creada vacia.

Los comandos y parámetros soportados por esta versión se gestionan desde la base de datos Gescom_MEGA2560_V3.mdb.

3.1.3 CREACIÓN, MODIFICACIÓN COMANDOS

Para crear, modificar y eliminar comandos se debe utilizar la base de datos Access Gescom_MEGA2560_V3.mdb.
Para más información sobre la utilización de esta BBDD ver el documento Instrucciones_BBDD.doc

3.2 CAMBIOS

La definición de la estrcutura de comandos de esta versión es incompatible con las anateriores versiones de la librería.

3.3 UTILIZACIÓN DE LA LIBRERÍA GESCOM

La forma típica de utilización de esta librería será:

```
#include <Gescom_MEGA2560_V3.h>
#include<Gescom_MEGA2560_V3_CMD.h>




GESCOM3 gc = GESCOM3(puertoID,modo,dispositivoID,velocidad);
// ------------------------------------------------------------
// . puertoID:
//   puede ser cualquiera de los defines (incluidos en el
//   define “Gescom_MEGA2560_V3.h”)que identifican los
//   puertos serie del Arduino MEGA2560
//
//   IDE_SERIAL_0
//   IDE_SERIAL_1
//   IDE_SERIAL_2
//   IDE_SERIAL_3
//
//  . modo:
//    true  → Modo DEBUG
//    false → Modo real
//
//  . dispositivoID:
//    Valor en hexadecimal que identifica el Robot (0x00 … 0xFF)
//
//  . Velocidad
//    Velocidad del puerto (tipicamente 9600)
//
// ------------------------------------------------------------


// ------------------------------------------------------------
//
// void setup()
//      Funcion setup de proyectos Arduino
//
// ------------------------------------------------------------

void setup()
{
      .
      .
      .
      .
      .
      .
}
```

```
// ---------------------------------------------------------
//
// void loop()
//      Funcion setup de proyectos Arduino
//
// ---------------------------------------------------------

void loop()
{
      .
      .
      .
  gc.exeGesComando();
      .
      .
      .
}
```

Para más información, ver el ejemplo *gescom_demo.ino* del directorio examples de la librería.

3.3.1 INSTALACIÓN

Para poder utilizar la librería es necesario copiar la carpeta "gescom_MEGA2560_V3" en el directorio de librerias de la versión del IDE de Arduino que se esté utilizado, por ejemplo para la versión arduino-1.0.4,

arduino-1.0.4\libraries\Gescom_MEGA2560_V3

3.3.2 DESCRIPCIÓN DE LA CLASE GESCOM

Todos los defines relacionados con el gestor de comandos se encuentran en el include *gescom_MEGA2560_V3_CMD.h*.

En la actual versión, la clase GESCOM3 tiene tres miembros públicos:

```
void GESCOM3::gescom( int puertoID,
                      int debug,
                      byte robID,
                      long int velocidad
                    );
```

Inicializa la clase y abre el puerto serie indicado por el parámetro "puertoID", inicializándolo a la velocidad indicada en "velocidad", se recomienda 9600.

- puertoID puede ser:

  ```
  IDE_SERIAL_0 Identifica el puerto serie 0
  IDE_SERIAL_1 Identifica el puerto serie 1
  IDE_SERIAL_2 Identifica el puerto serie 2
  IDE_SERIAL_3 Identifica el puerto serie 3
  ```

  Si se introduce un valor no válido se seleccionará el puerto 0

- debug puede ser:

true Habilitar depuración.
false Deshabilitar depuración.

- robID puede ser:
  1 valor (0x00…0xFF) que identifica el Robot.
  Gescom solo procesará comandos cuyo "robID" sea el especificado en este parámetro.

- Velocidad:
  Velocidad del transmisión (9600, etc).

int GESCOM::exeGesComando(void);

Ejecuta el gestor de comandos.
Retorna:

- IDE_BUFF_RX_OK — Se ha recibido algo valido.
  En este caso el comando se habrá ejecutado

  NOTA:
  Al recibir un comando siempre se comprueba el ID que identifica el dispositivo al que se dirige ese comando.
  Si el ID del comando NO es el de este dispositivo, el comando se ignora y no se ejecuta.

- IDE_BUFF_RX_ER — Se ha recibido algo erroneo.
  No se ha ejecutado nada

- IDE_BUFF_RX_VACIO — NO se ha recibido nada o si el ROB ID del comando recibido no corresponde con el indicado en el constructor de la clase.
  En este caso se retorna inmediatamente al programa principal.

int GESCOM3::getBytes(char* buff, int nBytes);

Lee del puerto serie asociado a la clase, el número de bytes indicados en el parámetro nBytes y los almacena en el buffer que recibe en el parámetro "buff", que debe contener espacio para los mismos.
Retorna el número de caracteres leidos, si todo ha ido correcto coincidirá con el número de bytes requeridos en el parámetro nBytes.
Si se produce error o no hay caracteres retorna 0.