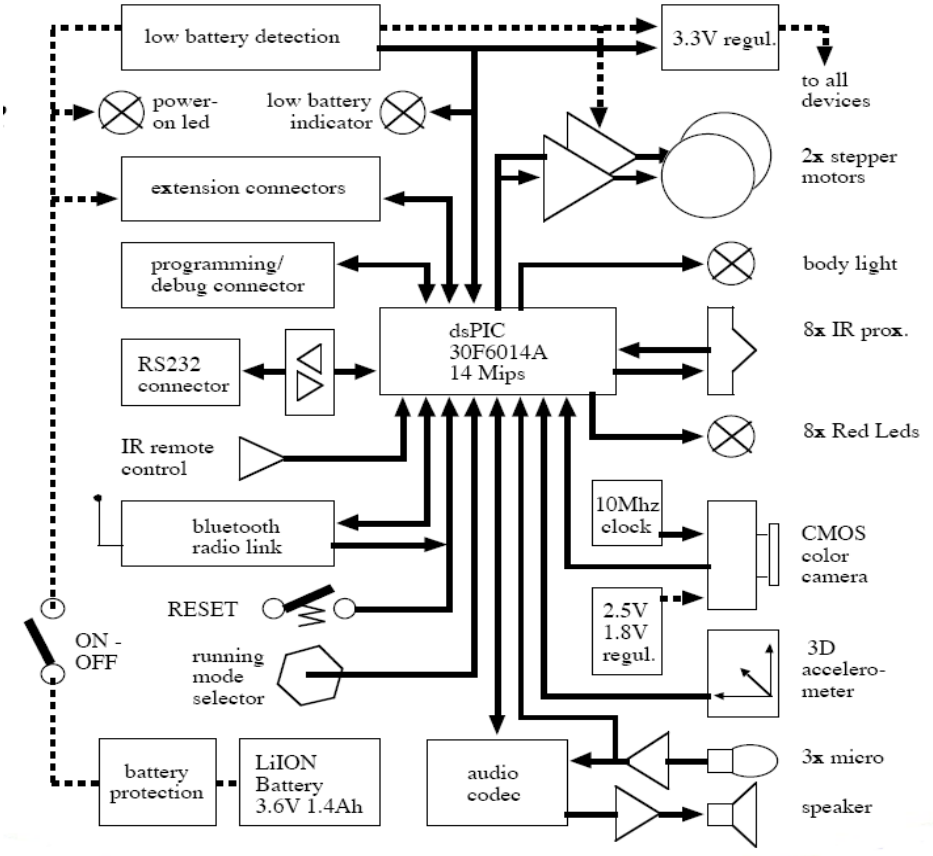
INTRODUCCIÓN AL E-PUCK

1.- Introducción

E-puck es un robot desarrollado como hardware abierto en la Escuela Politécnica Federal de Lausana (EPFL). En la página web del proyecto (www.e-puck.org) están disponibles librerías, ficheros de fabricación, ejemplos y una amplia documentación sobre el robot y su funcionamiento. En esta sección haremos una pequeña introducción al hardware del robot, pero para complementar información sobre el mismo se recomienda visitar la página oficial.

El robot dispone de un microcontrolador de Microchip, concretamente el dsPIC30F6014A, de 16 bits. Dispone de 144KB de Memoria de Programa, 8KB de RAM y 4KB de EEPROM.

El mapa sensorial y actuador del robot se puede observar en la siguiente figura:



Observamos que la placa base del robot dispone de:

- 2 motores paso paso.
- 8 Sensores de Infrarrojo.
- 8 Leds alrededor del anillo.
- 1 Led apuntando frontalmente.
- 1 Led que ilumina el cuerpo del robot.
- 1 acelerómetro de 3 ejes.
- 3 micrófonos
- 1 altavoz
- 1 cámara CMOS
- Comunicación bluetooth

Actualmente los robots del laboratorio disponen de dos placas de expansión.

- Una desarrollada en el EPFL: Tarjeta de sensores de suelo. Dicha tarjeta incorpora 3 sensores de infrarrojos que se encargan de distinguir colores en función de cuanto refleje (el infrarrojo) el color que se encuentra

justo debajo de los sensores.
- Una desarrollada en la ETSIT: Tarjeta de comunicación. Dicha tarjeta incorpora 12 módulos de emision/recepción, para comunicación local entre los robots.
- Se ha eliminado la tarjeta que incorpora el altavoz.

2.- Instalación y descarga de programas

Este apartado presenta la instalación del compilador y programa de descarga en el PC. En el laboratorio todo se encuentra YA instalado.

2.1.- Instalación del compilador cruzado

Descargar los paquetes Debian:

Binutils: http://robolabo.etsit.upm.es/descargas/epuck/pic30-binutils_2.01-1_i386.deb
Modificación de gcc: http://robolabo.etsit.upm.es/descargas/epuck/pic30-gcc_2.01-1_i386.deb
Soporte para gcc: http://robolabo.etsit.upm.es/descargas/epuck/pic30-support_2.01-1_all.deb

Ir al directorio donde se descargaron los paquetes e instalarlos:

```
alex@plop:~/factory/e-puck/packages/$ sudo dpkg -i pic30-binutils_2.01-1_i386.deb
Password:
Selecting previously deselected package pic30-binutils.
(Reading database ... 230447 files and directories currently installed.)
Unpacking pic30-binutils (from pic30-binutils_2.01-1_i386.deb) ...
Setting up pic30-binutils (2.01-1) ...

alex@plop:~/factory/e-puck/packages/$ sudo dpkg -i pic30-gcc_2.01-1_i386.deb
Password:
Selecting previously deselected package pic30-gcc.
(Reading database ... 230447 files and directories currently installed.)
Unpacking pic30-gcc (from pic30-gcc_2.01-1_i386.deb) ...
Setting up pic30-gcc (2.01-1) ...

alex@plop:~/factory/e-puck/packages/$ sudo dpkg -i pic30-support_2.01-1_all.deb
Password:
Selecting previously deselected package pic30-support.
(Reading database ... 230447 files and directories currently installed.)
Unpacking pic30-support (from pic30-support_2.01-1_i386.deb) ...
Setting up pic30-support (2.01-1) ...
```

La modificación de los ficheros de compilación así como la inclusión del compilador de los paquetes debian han sido realizadas por D. Alexandre Campo. Para más información sobre el "port" de los ficheros y los paquetes debian generados ver (http://iridia.ulb.ac.be/~e-puck/wiki/tiki-index.php):

2.2.- Instalación del sistema de descarga

La descargas de programas en el robot se realiza mediante la interfaz de comunicación bluetooth. De esta forma existe un programa en el cliente (PC) llamado epuckupload que descarga el fichero hexadecimal al robot.
Es un archivo ejecutable por lo que sitúalo en el directorio que más te interese. (Recomendamos /usr/local/bin/).

El programa se encuentra en http://robolabo.etsit.upm.es/descargas/epuck/epuckupload

La sintaxis de descarga es:

epuckupload -f *fichero.hex num_robot*,

donde fichero.hex es el hexadecimal del programa compilado y num_robot el número del robot asignado en el fichero de comunicación bluetooth.

Este programa también permite descargar el mismo fichero a diferentes robots simultáneamente:

epuckupload -f *fichero.hex num_robot1 num_robot2 num_robot3*,

hasta un máximo de 5 robots. Para más información teclear.

epuckupload

y obtener la ayuda.

2.3.- Prueba de descarga

Antes de poder descargar un fichero a un robot es necesario comprobar tanto que el dispositivo bluetooth del ordenador está funcionando como que se ha creado una clave entre el PC y el robot.

Una vez encendido el robot comprobar que es visto por el ordenador *hcitool scan*.

```
hcitool scan
Scanning ...

10:00:E8:52:AE:48       e-puck_1325
10:00:E8:52:A6:AF       e-puck_1258
```

Una vez comprobado que el robot es visto por el ordenador debemos incluir su MAC dentro del fichero de configuración de dispositivos bluetooth /etc/bluetooth/rfcomm.conf. En este fichero incluimos en cada linea una asociación entre un número y un robot. (Dicha asociación ya se encuentra hecha en el laboratorio)

```
rfcomm03{device 10:00:E8:52:A6:E8;}
rfcomm58{device 10:00:E8:52:A6:AF;}
                ...
                ...
```

Para poder conectarnos a un robot necesitamos fijarnos en el número que se le ha asignado a cada robot (Etiqueta en la parte frontal del robot). Si queremos comprobar la conexión con el robot 10 ejecutaremos el comando:

```
rfcomm connect 10
```

y él nos deberá responder

```
Connected /dev/rfcomm58 to 10:00:E8:52:A6:AF on channel 1
                Press CTRL-C for hangup
```

La primera vez que se produce una conexión con el robot debemos introducir la clave de conexión. Esta clave son los 4 números de serie que tiene el robot encima del chip de bluetooth o en el frontal del mismo. La aplicación necesaria para que el sistema nos solicite el pin de conexión es bluetooth-applet. En el caso de ubunto, viene instalada por defecto con el entorno gráfico.

Una vez conectado, vamos a descargar un programa de ejemplo para ver que todo se encuentra correctamente instalado. Descargamos el programa de ejemplo de http://robolabo.etsit.upm.es/descargas/epuck/ejemplo.hex.

Una vez descargado tecleamos:

```
epuckupload -f ejemplo.hex 10
```

El sistema se queda esperando hasta que el presionemos el botón de reset del robot (botón que se encuentra en la placa superior del e-puck). Una vez presionado y si todo va bien, ya estará el programa dentro del robot y empezará a ejecutar el programa ejemplo. Durante la descarga, se observará lo siguiente por pantalla:

```
[18:46:38] Reading file ejemplo.hex
[18:46:38] [/dev/rfcomm58] Connecting to /dev/rfcomm58 (robot ID 58)
[18:46:38] [/dev/rfcomm58] Uploading ejemplo.hex
R..............*********************************************************************************
**********************************************************************************************
```

```
**********************************************************************
[18:46:51] [/dev/rfcomm58] Transmission of ejemplo.hex complete!
```

3.- Compilación en el laboratorio

3.1.- Estructura de directorios

La estructura de ficheros del e-puck se encuentra dentro del directorio /home/alumnos/epuck.

En ese directorio encontramos otros 2 directorios:

- e-puck: De aquí cuelgan todas las librerías del robot así como las fuentes y hexadecimales de los programas realizados.
- pc: En este directorio se encuentra el programa ircomTest, que se encarga de comunicarse con el robot cuando está en ejecución y del que hablaremos más adelante.

3.1.2.- e-puck:

Este directorio se encuentra estructurado en 4 subdirectorios:

- src: En este directorio se encuentran las fuentes de los programas a generar. Actualmente hay algunos ejemplos. Cada proyecto debería ir en un directorio aparte con un makefile que habría que modificar EXE_TARGET por el nombre que se quiera dar al fichero de descarga.
- include: En este directorio se encuentran diferentes cabeceras de las librerías existentes en el e-puck.
- lib: Se encuentran librerías ya compiladas.
- bin: Este directorio almacena los ficheros de descarga para el robot de los proyectos compilados.

Dentro del directorio src/Epfl encontramos todos ficheros fuente de las librerías de manejo de sensores/actuadores del robot. Sobre estas librerías hablaremos en el apartado 3.3.

3.1.3.- pc:

Este directorio almacena un programa de comunicación con el robot. Se encuentra tanto el ejecutable como las fuentes para que el alumno lo modifique según son necesidades. El ejecutable existente se encarga de comunicarse con el robot y espera una pulsación de INTRO para enviarle un comando al robot. A partir de ahí empezará a imprimir por pantalla toda la información que sea transmitida desde el robot al ordenador.

3.2.- Programa Ejemplo

A continuación se muestra uno programa que se encarga de mover el robot en línea recta:

```
----------------------------
1  #include <p30f6014a.h>
2  #include <stdio.h>
3  #include <string.h>
4  #include <math.h>
5
6  #include <e_uart_char.h> /* Needed for the Bluetooth conection */
7  #include <e_init_port.h> /* Init all the IO, AD, etc. Ports
8  #include <e_motors.h>    /* Init motors */
9  #include <e_agenda.h>    /* Init an agenda */
10
11 #define SPEED 300
12
13 int main() {

14   /*system initialization */
15   e_start_agendas_processing();
16   e_init_port();
17   e_init_uart1();
18   e_init_motors();
```

```
19
20   /*Reset if Power on (some problem for few robots)*/
21   if (RCONbits.POR) {
22     RCONbits.POR=0;
23     __asm__ volatile ("reset");
24   }
25
26   while(1){
27     /* Set Robot going forward */
28     e_set_speed_left(SPEED);
29    e_set_speed_right(SPEED);
30   }
31
32   return 0;
33 }
```

El programa consta de un único main.
En el programa podemos observar diferentes partes:

- Includes [1-9] : Ficheros a incluir necesarios para el funcionamiento del mismo. Observamos la inclusión del microcontrolador así como librerías generales matemáticas, de conversión de datos y librerías específicas del robot. Prestaremos especial atención a la inclusión del fichero agenda.h que se encarga de ejecutar periódicamente las instrucciones de orden a los motores.
- Inicialización de los periféricos [14-18]: Uart, puertos, motores y agenda.
- Reset en caso de problema [20-24]: Algunos problemas detectados en algunos robot obligan a incluir esta pieza de código que resetea el controlador en caso de problemas con la alimentación.
- Bucle infinito [26-30]: Se le mandan la velocidad a la rueda derecha e izquierda. Valor definido en SPEED.

Encontramos este programa bajo el nombre de motors en los ejemplos disponibles en el laboratorio.

3.3.- Librerías

A continuación mostramos una colección de funciones utilizadas para el acceso a los sensores y actuadores del robot. En este documento no se encuentran especificadas todas ellas, por lo que para una ampliación de las mismas se recomienda una lectura del código fuente.

- Generales:
  - **Librerías**:
    - *e_uart_char.h.*
    - *e_init_port.h.*
    - *e_epuck_ports.h.*
  - **Funciones**:
    - *void e_init_port (void)*: Inicializa los pines del robot .
    - *void e_init_uart1 (void)*: Inicializa el puerto serie para el bluetooth.
- Leds:
  - **Librerías:**
    - *e_led.h.*
  - ***Funciones:***
    - *void e_set_led (unsigned int led_number, unsigned int value)*: Pone el led *led_number* [0-7] al valor *value* (0-1).
    - *void e_set_body_led(unsigned int value):* Pone el led del cuerpo al *valor value (0-1).*
    - *void e_set_front_led(unsigned int value)*: Pone el led frontal al valor *value* (0-1).
    - *void e_led_clear(void)*: Apaga todos los leds.
- Agenda:
  - **Librerías**:
    - *e_agenda.h.*
  - **Funciones**:
    - *void e_start_agendas_processing (void*): Inicializa la agenda periódica que se encarga de mandar datos a los motores.

- Motores:
  - **Librerías**:
    - *e_motors.h.*
  - **Funciones**:
    - *void e_init_motors (void)*: Inicializa los motores.
    - *void e_set_speed_left (int value)*: Escribe un valor entre [-1000,1000] en el motor izquierdo.
    - *void e_set_speed_right (int value)*: Escribe un valor entre [-1000,1000] en el motor izquierdo.
- Sensores de Infrarrojo:
  - **Librerías**:
    - *e_ad_conv.h.*
    - *e_prox.h*
  - ***Funciones****:*
    - **v***oid e_init_prox (void)*: Inicializa los sensores de infrarrojo.
    - *int e_get_prox (int number)*: Devuelve la medida del sensor de infrarrojo. Cuando no hay nada cercano **0** y cuando tiene un objeto muy cerca **4096**. Observar que el rango de distancia de medida de los e-pucks es de solo unos pocos centímetros.
    - *int e_get_ambient_light (int number* **)**: Devuelve la mediad luminosidad ambiente. **4096 c**uando está oscuridad y **0** punto más brillante. (Nota: téngase en cuenta que los sensores de infrarrojos son los usados para la medida de luz, por lo que una modificación en la misma afectará ligeramente a la medida de los sensores de infrarrojos).
- Acelerómetro:
  - **Librerías**:
    - *e_accelerometer.h.*
  - ***Funciones****:*
    - *void e_init_acc (void)*: Inicializa los acelerómetros.
    - *void e_get_acc(int* acc_x, int* acc_y, int* acc_z)*: Obtiene los valores de los 3 ejes del acelerómetro. Los valores devueltos van de **0 – 4096**.
- Bluetooth:
  - **Librerías:**
    - *btcom.h*
  - **Funciones:**
    - *void btcomWaitForCommand (char cmd)*: Esperar a un comando específico proveniente del bluetooth.
    - *void btcomSendString (string):* Enviar una cadena de caracteres por la conexión bluetooth.
- *Sensores de suelo:*
  - ***Librerías:***
    - *e_ground_sensor.h*
  - **Funciones:**
    - *void e_init_ground_sensor (void):* Inicializa los sensores de suelo.
    - *void e_get_ground_sensor (int value[3]).* Obtiene las lecturas de los 3 sensores de suelo. Aunque los sensores deberían ofrecer un valor entre 0 – 1024, en los experimentos realizados se observa que para un color blanco se obtiene un valor entorno a **950** y para un color negro, entorno a **200**.
- *Placa de Comunicación (e-randb):*
  - ***Librerías:***
    - *e_randb.h*
  - **Funciones:**
    - *void e_init_randb (void)*: Inicializa la tarjeta
    - *void e_randb_set_range (unsigned char distance):* Establece el rango de emisión de la tarjeta. 0-255 (1 m-0 cm).
    - *void e_randb_store_light_conditions (void):* Almacena los valores iniciales de luz ambiente
    - *void e_randb_set_calculation (char value):* Establece donde se realizan los cáluclos. En la tarjeta (ON_BOARD) o en el robot (ON_ROBOT).
    - *void e_randb_send_all_data(unsigned int data):* Envia el dato "data" (0-65535) por los 12 canales de la tarjeta.
    - *void e_randb_store_data (unsigned char channel , unsigned int data)*: Almacena el dato a enviar "data" por un canal específico "channel".

- *void e_randb_send_data (void):*Envía los datos almacenados mediante la instrucción anterior en los canales específicos
- Si los cálculos se realizan en la tarjeta (ON_BOARD)
  - *unsigned char e_randb_get_if_received(void):*Comprueba si ha nuevos datos recibidos. 0 si no se ha recibido nada, 1 si ha habido recepción.
  - *unsigned int e_randb_get_data(void):* Obtiene el dato recibido (0-65535)
  - *unsigned int e_randb_get_range(void):* Extrae la estimación de distancia al emisor. (0-4096) 0 muy lejos y 4096 muy cerca.
  - *double e_randb_get_bearing(void):* Extrae la estimación de ángulo del emisor (0-2*pi)
  - *unsigned char e_randb_get_sensor (void):* Devuelve el sensor de máxima recepción.
- Si los cálculos se realizan en el robot (ON_ROBOT)
  - *unsigned int e_randb_get_all_data (unsigned int *peaks, unsigned int *data):* Devuelve un puntero a un array (peaks) de 12 valores (los valores recibidos en los 12 sensores de recepción) y el dato transmitido (data).
  - *int e_randb_all_reception(unsigned int *range, float *bearing, unsigned int *data)*: Obtiene el dato (data), distancia (range) y ángulo (bearing) del emisor.

3.3.- Ejemplos

Existen una serie de ejemplos dentro del directorio src, estos ejemplos son los siguientes:

- leds: Se encarga de ir encendiendo y apagando los leds del anillo del robot.
- motors: Se encarga de hacer andar el robot en línea recta.
- ir: Envía por bluetooth los valores medidos de los 8 sensores de infrarrojo.
- light: Envía por bluetooth los valores de luz medidos por los sensores de infrarrojos.
- acel: Envía por bluetooth el valor del los 3 ejes del acelerómetro.
- ground: Envía por bluetooth el valor de los 3 sensores de suelo.
- randbEmitter: Envía contínumente por la tarjeta e-randb una secuencia de números entre 0-65535.
- randbReceiver: Envía por bluetooth los datos recividos por la tarjeta e-randb.
- randbDoubleReceiver: Envía por la tarjeta e-randb una secuencia de número entre 0-65535 y si recibe datos los envía por bluetooth. (El programa es una suma de randbReceiver y randbDoubleReceiver).