

Espardino librería software

Espardino micro 214x – guía de las librerías.



Las librerías de Espardino son útiles para comenzar con tu proyecto sin tener que comprender todos los detalles de configuración hardware, permitiéndote así ir directo al desarrollo de tu aplicación

En el kit de desarrollo encontrarás librerías para acceder al sistemas de ficheros de una microSD (FAT12/16/32), construir un puerto serie virtual sobre USB para desarrollo rápido de comunicaciones con un PC, acceso a los convertidores Analógico/Digital (ADC) , las salidas PWM, entradas CCP/PWM , leds, pulsador...

LICENCIAS DE LAS LIBRERÍAS DE ESPARDINO

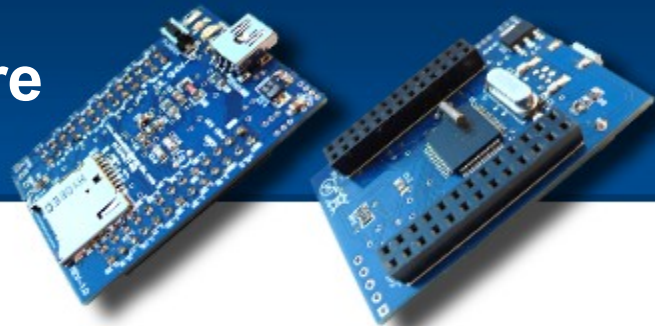
Espardino puede ser útil tanto para el aficionado a la electrónica como para el profesional que busca componentes testados de fábrica, y de rápido desarrollo. Los aficionados normalmente no deberían tener que preocuparse por licencias de software, mientras que para un desarrollador profesional esto es algo de lo que realmente debe preocuparse. Cuando está construyendo un nuevo producto.

Todas las librerías proporcionadas con Espardino, pueden redistribuirse en nuestra aplicación de forma binaria (ya sea programada o en fichero).

(*) Las librerías específicas del proyecto Espardino tienen licencia para su uso sin restricciones en aplicaciones comerciales, con la única condición de que durante el desarrollo se utilicen placas Espardino originales.

Módulo	Proyecto	Tipo de licencia
Virtual Com Port	lpcusb	BSD: Uso de binarios sin restricciones
Conversión A/D	Espardino	Uso de binarios sin restricciones (*)
Salidas PWM	Espardino	Uso de binarios sin restricciones (*)
Entradas PWM	Espardino	Uso de binarios sin restricciones (*)
Acceso MicroSD	EFSL	Uso de binarios sin restricciones ¹
Salida DAC	Espardino	Uso de binarios sin restricciones (*)
mini printf	Espardino	Uso de binarios sin restricciones (*)
USART	Espardino	Uso de binarios sin restricciones (*)
SPI	Espardino	Uso de binarios sin restricciones (*)
I2C	Espardino	Uso de binarios sin restricciones (*)

¹ GPL modificada: GPL sin necesidad de redistribuir tu aplicación como GPL ni su código fuente. (ideal para una librería, mejor que LGPL que obliga a la redistribución de los binarios de librería).



LIBRERIA DE PUERTO SERIE VIRTUAL (VCP)

La librería de puerto serie virtual está basado en las librerías lpcusb disponibles en Internet bajo licencia BSD.

Es posible configurar un puerto serie virtual con el PC, lo cual es una forma fácil y efectiva de construir las comunicaciones de la mayor parte de aplicaciones.

Utiliza los siguientes recursos: Nivel VIC de interrupción 0, Interfaz USB

Utilizando el puerto serie virtual en tus aplicaciones

1st) Edita tu archivo Makefile para incluir la librería "vcom" durante el enlazado. Puedes modificar el Makefile suministrado con Espardino añadiendo "-lvcom" a la variable MY_LIBS.

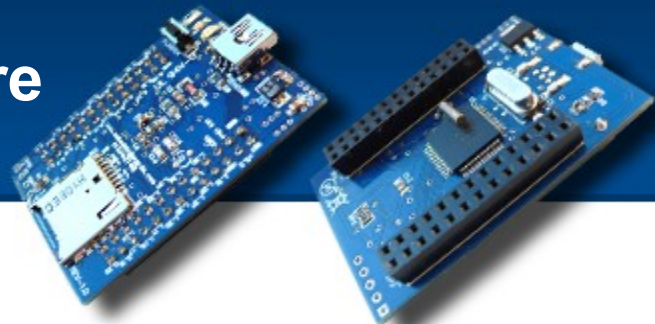
```
# System libraries to be linked with
MY_LIBS = -lefs -lvcom -lmicro214x
```

2nd) Incluye la cabecera "vcom.h" en tu aplicación:

```
#include <vcom.h>
```

3rd) Llama a VCOM_init() durante la inicialización de tu aplicación. Esto configurará todo el hardware USB así como las interrupciones para atender a las comunicaciones del puerto serie.

```
VCOM_init(); /* Inicializamos el puerto serie virtual */
```



Enviando y recibiendo datos a través del puerto serie virtual (VCP)

```
int VCOM_putchar_nonblock(int c);
```

Envía un carácter por el VCP, si el buffer de salida está lleno no esperará para enviar dicho carácter, devolviendo un valor <0 en ese caso.

```
int VCOM_getchar_nonblock(void);
```

Lee un carácter del VCP, si el buffer de entrada está vacío este no esperará a que estén disponibles nuevos datos, devolviendo un valor <0 en dicho caso.

```
int VCOM_putchar(int c);
```

Escribe un carácter en el buffer de salida del VCP, en caso de que dicho buffer esté lleno este esperará a que quede espacio disponible, deteniendo la ejecución de la aplicación.

```
int VCOM_getchar(void);
```

Lee un carácter del VCP, en caso de que el buffer de entrada del mismo esté vacío esperará a que lleguen nuevos datos.

```
void VCOM_puts(const char *str);
```

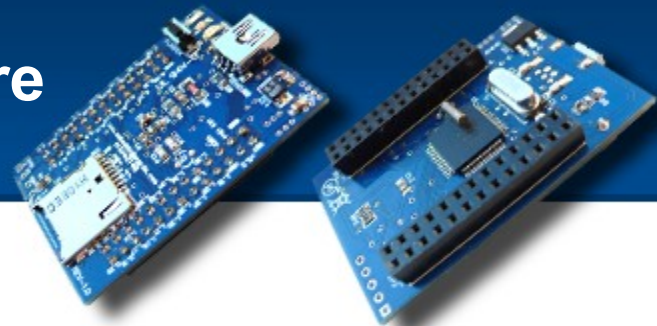
Escribe una cadena ASCII terminada en 0, esperando a que se envíe la cadena al completo antes de regresar.

Utilizando printf con la librería VCP

Las librerías de Espardino incluyen una versión minimalista de printf / sprintf con un tamaño de código muy pequeño. Únicamente están soportados los códigos %c, %s, %d, %u, %x y %X, aunque podrán ser ampliados con el tiempo.

Para configurar la función de salida de caracteres por defecto de printf existe la función printf_output, que permite asignar una función de escritura de caracteres, recibe siempre un puntero a función. En este ejemplo printf hará escritura bloqueante sobre el puerto serie virtual:

```
printf_output(&VCOM_putchar);
```



LIBRERÍA DEL CONV. ANALOGICO/DIGITAL

El modelo micro2142 dispone de 5 entradas analógicas configurables entre 3 y 10bits: AD0.1,AD0.2, AD0.3 ,AD0.4, AD0.7.

El modelo micro2148 dispone de 8 entradas analógicas de 10bit adicionales a las AD0.1,2,3,4,7. Estas son AD1.0, AD1.1, AD1.2, AD1.3, AD1.4, AD1.5, AD1.6, AD1.7.

Todos los puertos analógicos están multiplexados con otros pines multifunción, y su lectura será siempre relativa a la entrada VREF y masa. Por ejemplo, si configuramos la lectura en modo 10bits, VREF tiene un valor de 2.8V, y AD0.1 es 2.8V o mayor, la lectura de AD0.1 debería resultar "1023" (o lo que es lo mismo, 10 bits a "1").

La librería de acceso al convertidor A/D de Espardino es una forma sencilla y rápida de acceder a las entradas ADC disponibles, y debe ser útil en la mayor parte de las situaciones. Programando la lectura mediante IRQs, o lectura por ráfagas periódicas es posible conseguir mejores resultados en ciertas aplicaciones.

Utiliza los siguientes recursos:

Entrada	Puerto y función
AD0.1	P0.28,CAP0.3,MAT0.3
AD0.2	P0.29,CAP0.3,MAT0.3
AD0.3	P0.30,EINT3,CAP0.0
AD0.4	P0.25 AD0.4,AOUT
AD0.7	P0.5,MISO0,MAT0.1,AD0.7
AD1.0	P0.6,MOSI0,CAP0.2
AD1.1	P0.8,TXD1,PWM4
AD1.2	P0.10,RTS1,CAP1.0
AD1.3	P0.12,DSR1,MAT1.0
AD1.4	P0.13 ,DTR1,MAT1.1
AD1.5	P0.15,RI1,EINT2
AD1.6	P0.21, PWM5,CAP1.3
AD1.7	P0.22,CAP0.0,MAT0.0



Utilización de la librería ADC

1st) Edita tu archivo Makefile para incluir la librería “micro214x” durante el enlazado. Puedes modificar el Makefile del proyecto base de Espardino añadiendo “-lmicro214x” a la variable MY_LIBS.

```
# System libraries to be linked with
MY_LIBS = -lefs -lvcom -lmicro214x
```

2nd) Incluye las cabeceras “micro214x.h” en tu aplicación.

```
#include <micro214x.h>
```

3rd) Llama a ADC_init(mask) durante la inicialización de tu aplicación. Esto configurará todos los pins seleccionados como entradas analógicas.

```
ADC_init(AD0_1|AD0_2); /* activa AD0.1 y AD0.2 como A/D */
```

Las máscaras disponibles son: AD0_1, AD0_2, AD0_3, AD0_4, AD0_6, AD0_7, AD1_0, AD1_1, AD1_2, AD1_3, AD1_4, AD1_5, AD1_6, AD1_7.

Leyendo datos de los puertos ADC

```
void ADC_set_precision(int bits);
```

Configura la precisión de lectura de los convertidores A/D entre 3 y 10 bits: bajando la precisión conseguimos una mayor velocidad de lectura, mientras que incrementándola esta se reduce, pero conseguimos una lectura más precisa.

```
int ADC_read(int adc);
```

Inicia una lectura de un canal ADC en concreto {AD0_1 .. AD1_7}, esperando a que la muestra esté lista para devolverla como un entero positivo de la resolución configurada.

```
void ADC_start(int adc);
```

Esta es otra forma de realizar una lectura de forma no bloqueante. Con esta función se inicia la lectura en un canal {AD0_1 .. AD1_7}. Regresando inmediatamente.

Esparduino librería software

Esparduino micro 214x – guía de las librerías.



```
int ADC_done(int adc);
```

Esta función, devuelve si la lectura de la entrada analógica está terminada y lista para su lectura con ADC_get o no. Devolverá 0 siempre que la lectura no haya finalizado.

```
int ADC_get(int adc);
```

Con ADC_get leeremos el resultado de la conversión analógico digital de un canal en concreto. Podremos llamar a esta función siempre que ADC_done haya indicado que la lectura esté lista para dicho canal.



LIBRERIA DE SALIDAS PWM

La librería de salidas PWM permite la utilización de hasta 6 salidas (ver guía rápida) PWM1 .. PWM6. Las salidas PWM son útiles para manejar la posición de servomotores, control de motores y mosfets mediante PWM o generación de señales analógicas mediante un filtro R/C.

Utiliza los siguientes recursos:

Salida	Puerto y función
PWM_1	P0.0 (compartido con TXD0)
PWM_2	P0.7 (compartido con SSEL0) (microSD)
PWM_3	P0.1 (compartido con RXD0)
PWM_4	P0.8 (compartido con TXD1)
PWM_5	P0.21 (compartido con AD1.6)
PWM_6	P0.9 (compartido con RXD1)

Utilizando las librería de salidas PWM en nuestra aplicación

1) Edita tu archivo Makefile para incluir la librería "micro214x" durante el enlazado. Puedes modificar el Makefile suministrado con Espardino añadiendo "-lmicro214x" a la variable MY_LIBS.

```
# System libraries to be linked with
MY_LIBS = -lefs -lvcom -lmicro214x
```

2) Incluye la cabecera "micro214x.h" en tu aplicación:

```
#include <micro214x.h>
```

3) Llama a PWM_init(máscara) durante la inicialización de tu aplicación. Esto los puertos correspondientes en modo PWM.

```
PWM_init(PWM_1,PWM_4); /* P0.0 y P0.8 como salidas PWM */
```

4) Llama a PWM_frequency(Hz_freq) con la frecuencia en hercios que quieres producir en las salidas PWM.

```
PWM_frequency(50); /* 50Hz / 20ms / control de servos */
```

Esparidino librería software

Esparidino micro 214x – guía de las librerías.



5) Configura las salidas PWM en el ciclo de trabajo deseado con las funciones `PWM_pulsewidth_us(máscara, microsegundos)` o `PWM_pulsewidth(máscara, ciclos_de_reloj)`, normalmente el reloj estará configurado a 60.000.000 ciclos por segundo.

www.nbee.es