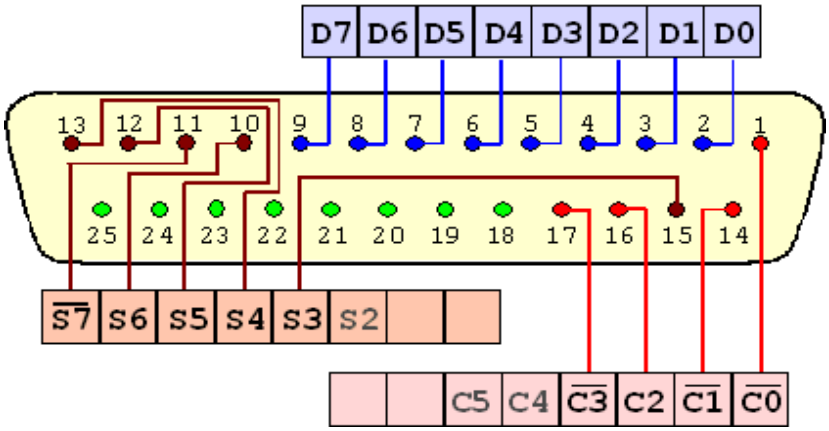
Para hacer uso del puerto paralelo es necesario comprender que éste contiene un puerto de de datos y un puerto de estado. Estos puertos se corresponden con direcciones de E/S, siendo lo común la dirección 378h para el puerto de datos (pines 2 al 9) y 379h para el puerto de estado (pines 11, 10, 12, 13 y 15).

El puerto de datos contiene 8 bits por los cuales se puede sacar una señal de 5 volt (nivel alto, 1) o una señal de 0 volts (nivel bajo, 0).

El puerto de estado contiene 8 bits de los cuales podemos utilizar los 5 bits mas significativos para el ingreso de señales. Cabe destacar que el bit mas significativo (pin 11) funciona con lógica negativa, es decir, tiene un nivel alto (1) al recibir una señal de 0 volt, y su nivel es bajo (0) al recibir 5 volt.

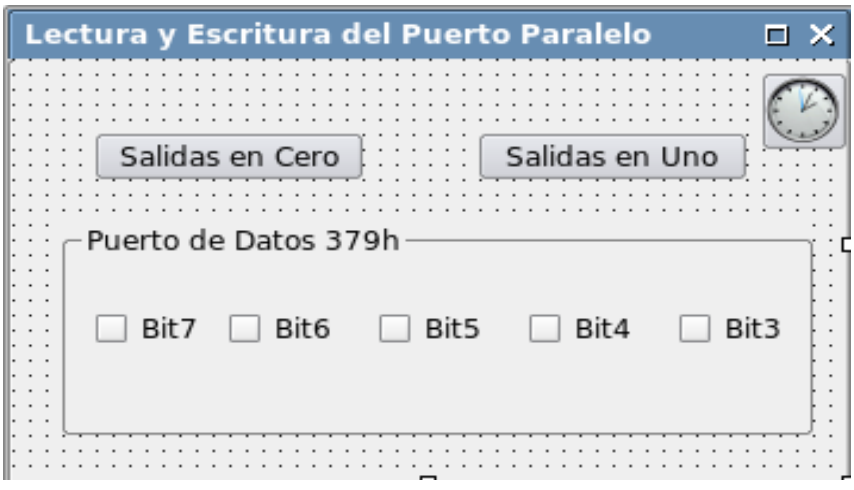
En la siguiente figura se puede apreciar la numeración de los pines del puerto paralelo y a que puerto pertenece.



Lectura y escritura del puerto paralelo en Gambas

Antes de iniciar, es importante aclarar que en linux solo es posible el uso del puerto paralelo al usuario root, por lo cual debemos ejecutar Gambas como root (sudo gambas2) o crear el ejecutable y luego ejecutarlo como superusuario.

Para hacer la primer práctica de escritura y lectura del puerto paralelo, vamos a diseñar el siguiente formulario:



Iniciamos la programación declarando las variables, en este caso una variable para el puerto de datos y otra para el puerto de estado, como en linux se accede a cada dispositivo como un archivo las declaramos del tipo File. También declaramos una variable para recibir el byte de entrada y otra en donde se almacena el byte de salida. Las constantes Pin3, Pin4, Pin5, Pin6 y Pin7 son necesarias para realizar un enmascaramiento de bits para poder determinar que valor tiene cada pin en particular, esto se explicará mas adelante. La declaración completa de variables queda de la siguiente manera:

```
PUBLIC PuertoDatos AS File
PUBLIC PuertoEstado AS File
PUBLIC Entrada AS Byte
PUBLIC Salida AS Byte
CONST bit3 AS Byte = 8
CONST bit4 AS Byte = 16
CONST bit5 AS Byte = 32
CONST bit6 AS Byte = 64
CONST bit7 AS Byte = 128
```

Escritura, salida de un byte (8 bits) por los pines del 2 al 9

La salida de datos será programada desde los dos botones de nuestro formulario (controles Button), el valor de salida es un byte, cuyo rango posible es de 0 a 255. Estos valores están expresados en sistema decimal. Al convertirlos en binario tendremos una idea de cuales pines recibirá un nivel alto y cuales un nivel bajo.

Si se desea activar todas las salidas, el numero binario es 11111111, es decir, sale un 1 por cada uno de los pines. Si convertimos el binario 11111111 a decimal obtenemos el número 255, este valor es el que debemos escribir en el puerto de datos. El código para hacerlo se observa a continuación:

```
PUBLIC SUB Button2_Click()
    puertoDatos = OPEN "/dev/port" FOR WRITE
    SEEK #puertoDatos, &H378
    Salida = 255
    WRITE #puertoDatos, Salida
    CLOSE puertoDatos
END
```

Para enviar un nivel bajo a todos los pines el número binario es 00000000, es decir 0 en sistema decimal. Lo hacemos de la misma manera que el caso anterior pero cambiando el valor de salida.

```
PUBLIC SUB Button2_Click()
    puertoDatos = OPEN "/dev/port" FOR WRITE
    SEEK #puertoDatos, &H378
    Salida = 0
    WRITE #puertoDatos, Salida
    CLOSE puertoDatos
END
```

El acceso al puerto de datos se realiza como si se tratase de un archivo, aquí el modo de apertura del puerto de datos 378h es escritura (write).

Lectura, entrada de un byte (solo se utilizan los 5 mas significativos) por los pines 11, 10, 12, 13 y 15.

La constantes que declaramos al inicio del proyecto se corresponden al peso de cada uno de los pines por los cuales ingresan las señales. Por lo tanto, la constante bit7 hace referencia al pin 11, bit6 al pin 10, bit 5 al pin 12, bit4 al 13 y bit5 al pin 15.

La lectura del puerto de estado arrojará un número decimal correspondiente al byte que este ingresando. Suponiendo que la lectura del puerto de estado devuelve el numero decimal 170, el equivalente en binario es 10101010. De estos 8 bits, como se ha dicho, nos interesan los 5 mas significativos.

```
Pin11   Pin10  Pin12  Pin13   Pin15
bit 7    bit6    bit5   bit4    bit3
  1       0       1      0       1       0       1       0   ---------> 170
```

Ahora, a simple vista, podemos comprender que ese valor de 170 que se leyó, indica que por el pin11 hay un nivel alto, por el pin 10 un nivel bajo, por el pin 12 un nivel alto, por el pin 13 un nivel bajo y por el pin 15 un nivel alto.

Para determinar el valor de cada pin en particular es necesario realizar un enmascaramiento de bits. El enmascaramiento de bits consiste en combinar un valor binario con otro para aislar los bits que nos interesan. Para determinar el valor del bit7 (pin 11) realizamos la siguiente operación:

```
      Pin11   Pin10  Pin12  Pin13   Pin15
      bit 7    bit6    bit5   bit4    bit3
        1       0       1      0       1       0       1       0   ----------> 170
AND                                                                     AND
        1       0       0      0       0       0       0       0   ----------> 128
-------------------------------------------------------------------------------
        1       0       0      0       0       0       0       0   ----------> 128
```

El resultado es 128, lo cual indica la presencia de un 1 en el bit 7. Vemos el caso del bit 6.

```
      Pin11   Pin10  Pin12  Pin13   Pin15
      bit 7    bit6    bit5   bit4    bit3
        1       0       1      0       1       0       1       0   ----------> 170
AND                                                                     AND
        0       1       0      0       0       0       0       0   ---------->  64
-------------------------------------------------------------------------------
        0       0       0      0       0       0       0       0   ----------->   0
```

El resultado es cero, lo cual indica la presencia de un cero en el bit 6.

Teniendo en cuenta el enmascaramiento de bits, se puede programar la lectura del puerto de estado cada 300 milisegundos, es decir 0.3 segundos. Esto lo hacemos con un control Timer de la siguiente manera:

```
PUBLIC SUB Timer1_Timer()

    puertoEstado = OPEN "/dev/port" FOR READ
    SEEK #puertoEstado, &H379
    READ #puertoEstado, Entrada
    CLOSE puertoEstado

    IF (Entrada AND bit3) = bit3 THEN
       Chk3.Value = 1
    ELSE
       Chk3.Value = 0
    END IF


    IF (Entrada AND bit4) = bit4 THEN
       Chk4.Value = 1
    ELSE
       Chk4.Value = 0
    END IF

    IF (Entrada AND bit5) = bit5 THEN
       Chk5.Value = 1
    ELSE
       Chk5.Value = 0
    END IF

    IF (Entrada AND bit6) = bit6 THEN
       Chk6.Value = 1
    ELSE
       Chk6.Value = 0
    END IF

    IF (Entrada AND bit7) = bit7 THEN
       Chk7.Value = 1
    ELSE
       Chk7.Value = 0
    END IF
END
```

En el código anterior la lectura del puerto se efectúa como si fuese un archivo, se accede al puerto de estado 379h en modo lectura (read). Luego de leer el puerto, se realiza el enmascaramiento de cada uno de los bits que nos interesa, con el fin de indicar aquellos bits que se encuentran en nivel alto, por medio de CheckBox tildados, y en nivel bajo, por medio de CheckBox sin tildar.

De esta manera logramos utilizar el puerto paralelo para recibir señales del exterior o enviarlas, lo cuál nos abre un amplio abanico de posibilidades, ya sea para controlar motores o encender lámparas, como así también activar ciertos procesos a partir de señales externas.

Pablo Mileti
pablomileti@gmail.com