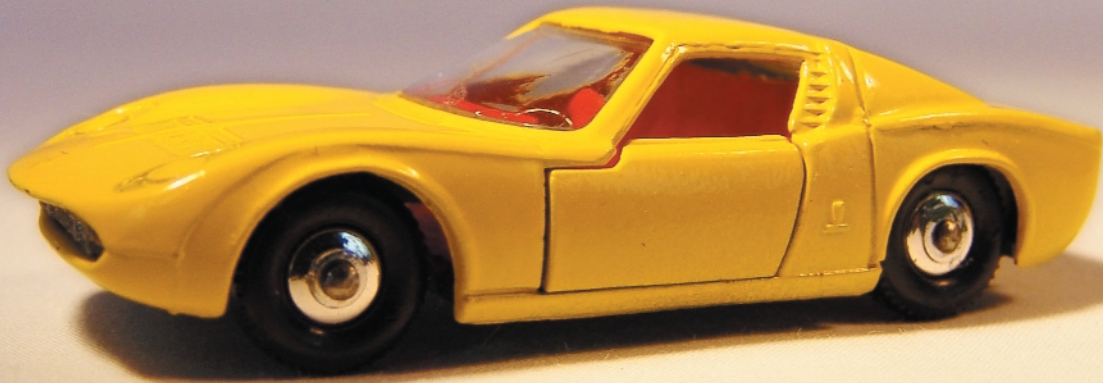


Control de un Coche Teledirigido desde Linux (II)

MONTANDO A HERBIE



Ha llegado el momento de finalizar las modificaciones del mando a distancia del coche que emprendimos el mes pasado, y de preparar nuestros *scripts* que lo controlarán desde el ordenador.

POR ALBERTO PLANAS Y VÍCTOR TIENDA

Recordemos cómo terminamos el mes pasado. Extraíamos el circuito del mando a distancia encargado del controlar remotamente el coche, así como el compartimento para la pila de 9V. Además diseñamos y construimos un cable paralelo para transmitir la información enviada desde las ocho líneas de datos (pines del 2 al 9). Explicamos el funcionamiento de un transistor en modo corte / saturación y planteamos el problema de manera general. Estamos, por tanto, en la parte más emocionante del proyecto: vamos a ensamblar el resto de los componentes.

El Optoacoplador

Ya comentamos de pasada en el número anterior los peligros que conlleva para nuestro puerto paralelo el conectarlo a un circuito alimentado de manera externa. Podríamos dañar los componentes de nuestro ordenador si por una mala soldadura o algún error en el diseño, entrara algún voltaje por alguno de los pines de sólo salida. Estamos experimentando y por tanto es previsible que cometamos errores en el proceso: necesitamos proteger nuestro ordenador. Lo ideal sería separar o desacoplar los dos circuitos: el del puerto paralelo y el circuito de radio-

control con el controlador que estamos diseñando. Pues bien, esto lo vamos a lograr con los optoacopladores.

Un optoacoplador no es más que un transistor controlado por la base mediante un diodo fotoemisor. Es decir, nosotros controlaremos la corriente que pasa por cada lado del diodo (ánodo y cátodo) y éste, como respuesta a dicha corriente, emitirá fotones que serán recogidos por el fotorreceptor acoplado a la base de transistor. Este fotorreceptor transformará la energía luminosa otra vez en energía eléctrica, que se usará para alimentar la base del transistor. Dependiendo de con cuánta intensidad y voltaje alimentemos el fotodiodo, así emitirá más o menos luminosidad, y por ende tendremos más o menos potencial y corriente en el transistor. De esta manera tan sencilla hemos desacoplado los dos circuitos, no hay ningún cable que una la parte posterior al transistor con nuestro ordenador.

El opto viene en forma de integrado con 6 pines. La norma es marcar el pin número 1 con un punto en la superficie, y a partir de ahí tendremos localizados el resto de los pines. La finalidad de cada uno de ellos podemos consultarla en el *datasheet* del fabricante[1]. En la

Figura 1 podemos ver un esquema del interior del optoacoplador que hemos seleccionado para este proyecto, el **4N25**, y la disposición de sus pines. La finalidad de cada uno de ellos es:

- Pin 1: Ánodo del diodo
- Pin 2: Cátodo del diodo
- Pin 3: No se usa, sin conexión
- Pin 4: Emisor del transistor
- Pin 5: Colector del transistor
- Pin 6: Base del transistor

Nosotros usaremos solo los pines 1, 2, 4 y 5. Con el 1 y el 2 controlaremos el diodo. Hay que tener cuidado en este punto, el diodo sólo deja pasar la corriente en dirección ánodo -> cátodo. La diferencia de potencial entre estos dos debe ser, por tanto, positiva. Si cambiamos el potencial y hacemos que circule corriente en la dirección cátodo -> ánodo correremos el riesgo de quemar el diodo. De la parte del transistor solo usaremos los pines del colector (5) y del emisor (4), ya que la base es controlada automáticamente por el diodo fotorreceptor que comentamos antes. El esquema de conexionado que seguiremos será similar al expuesto en el artículo anterior. Sería abusar de la paciencia del lector que repitamos la explicación del funcionamiento del transistor, pero hay que recordar que solo nos intere-

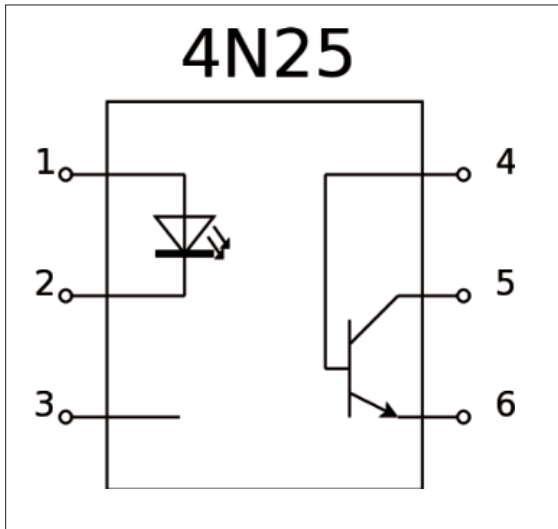


Figura 1: La correspondencia de pines puede encontrarse en el texto.



Figura 3: Nuestro cochecito, comprado en una tienda de todo a 60 céntimos, con la placa creada y el cable para el puerto paralelo.

sa el comportamiento del mismo en modo corte / saturación. Es en este modo cuando el potencial V_1 que pintamos en la figura 6 del artículo referido pasa a valer 0 V ó 9 V a nuestro antojo.

A partir de todo lo expuesto mostraremos el diagrama global que deberemos construir. Un esquema lo podemos encontrar en la Figura 2. En él vemos que los pines 2 y 3 del puerto paralelo están conectados a los pines 1 y 2 del optoacoplador, que corresponden, respectivamente, al ánodo y al cátodo del diodo. Por otro lado el extremo del transistor tiene una conexión en el colector a una resistencia que a su vez está conectada en el polo positivo de la pila (extremo V_{cc}) y por otro lado a parte positiva del interruptor del mando a distancia. El emisor del transistor queda colocado en tierra, es decir, a la parte negativa del mismo interruptor. Si nosotros emitimos un 1 (3.3V) por el pin 2 del puerto paralelo y simultáneamente emitimos un 0 (0V) por el pin 3, estaremos encendiendo el diodo del optoacoplador con la suficiente intensidad como para poder

saturar el transistor. Éste, al saturarse pondrá 0V (aproximadamente) en el borne positivo del interruptor que estamos controlando, y simulará su pulsación. En cambio, si ponemos a 0 los pines 2 y 3 del puerto paralelo, el diodo dejará de emitir cualquier tipo de luz lo que provocará que el transistor pase directamente a corte, dejando V_{cc} (9V) en la parte positiva del interruptor.

Este esquema deberemos seguirlo por cada uno de los interruptores, como tenemos cuatro direcciones (adelante, atrás, derecha e izquierda) necesitaremos repetir este circuito cuatro veces. De esta manera los pines 2 y 3 controlarán un interruptor, el 4 y 5 otro ... así hasta completar los cuatro indicados.

¡A Soldar!

Esto es un arte. Con un pincel eléctrico vamos repartiendo trozos de estaño por aquí y por allí. El problema es que soy un artista incomprendido y mis obras se niegan a permanecer funcionales durante mucho tiempo. Es aquí donde hay que llamar a un amigo experto en la materia para que te eche una mano. Él nos enseñará que bajo esta expresión artística hay también unas reglas lógicas, trucos que harán que la soldadura se acerque al ámbito de la ciencia y de lo predecible. Lo primero que aprendí como

soldador padawan es a tomar una postura cómoda que impida que tropecemos en el dichoso cable del soldador. También es aconsejable que nos agenciemos un trozo de esponja húmeda para ir limpiando el estaño del soldador cada vez que procedamos a una nueva soldadura. Un truco eficaz es calentar el cable antes de aplicar estaño. Colorearemos la punta del soldador sobre el cobre del cable y esperaremos a que este tome temperatura, luego derretiremos una gotita de estaño sobre el cable y con un giro de muñecas extenderemos rápidamente este estaño (cuidado aquí con los ojos). Si hacemos esto en ambas partes de la soldadura lograremos

unos resultados más limpios y eficientes. Otra de las cosas que aprendí es que es mejor tener cuatro manos que solo dos. Si alguien nos puede ayudar a sujetar los extremos que queremos unir, nos resultará mucho más sencilla la tarea de soldarlos. Existen altas probabilidades de que sin querer unamos dos patillas con estaño. En estos casos hay que proceder sin compasión, tendremos que quitar el estaño sobrante con el soldador. Una suerte es que los componentes escogidos (las resistencias y los optos) son duros y no se romperán fácilmente al aplicarle excesivo calor a las patillas (cosa que pasa cuando nos entretenemos demasiado al intentar fijar la soldadura).

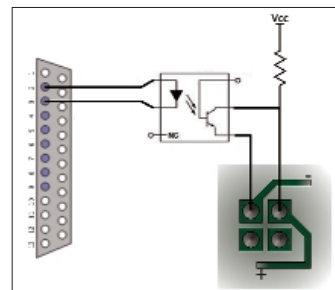


Figura 2: Diagrama global de conexiones del circuito.



Ven a Telefónica Campus Party unimos talento .creamos futuro

Valencia, del 25 al 31 de julio www.campus-party.org

Más de 5.000 participantes acuden a Campus Party, el mayor encuentro de entusiastas de todo el mundo, para disfrutar con infinidad de actividades relacionadas con el ordenador y las comunicaciones.

Últimas plazas a la venta. Entra en www.campus-party.org y apúntate a una de sus áreas: Acción y Astronomía, Deportes, Simulación, Software Libre, CampusCrea, Robótica, Modding, Astronomía y Desarrollo.

5.500
PARTICIPANTES
ÚLTIMAS PLAZAS
A LA VENTA

Telefónica

GENERALITAT
VALÈNCIANA

ES
Futuras
www.esfuturas.com

AYUNTAMIENTO DE VALÈNCIA
CONSEJO REGULADOR DE ACTIVIDADES CULTURALES

PartyCite

PS2
PlayStation 2

PC CITY
COMPUTER SUPERMART

LINUX
DISTRIBUTION

Hay una solución mucho más cómoda que soldar, pero es también más cara. En las tiendas de electrónica se pueden conseguir una placas de pruebas. Son placas de plástico perforado sobre una malla de cobre. Esta malla transmite la corriente a los puntos de su misma fila o columna, con lo que nuestro trabajo se reduce a pinchar los componentes en el orden adecuado para simular el circuito que deseemos construir. Muy recomendable.

Scripts

Por fin tenemos nuestro circuito montado. Vamos a testearlo. Para esto hay que hacer un programa que use la librería *parapin* presentada en el artículo anterior y el programa mostrado en el Listado 1. Este programa puede controlar cuatro pines del puerto paralelo de manera simultanea

(con cuatro es suficiente, ya que son a lo sumo dos los interruptores que desearemos controlar a la vez). Compilaremos el programa de la siguiente manera:

```
gcc send_lpt.c -o send_lpt -lparapin
```

El programa generado tiene cinco parámetros. Con los cuatro primeros indicaremos que pines activamos y cuales desactivamos y con el último el tiempo en el que permanecerán activado dichos pines. Si hemos conectado en el orden *adelante, atrás, derecha e izquierda* los interruptores del mando a distancia, entonces el comando `./send_lpt 2 3 0 0 2000` encenderá el pin 2 y apagará el pin 3 durante dos segundos (el tiempo se expresa en milisegundos). Esto hará que

el coche se desplace hacia adelante durante estos dos segundos (menos el tiempo en el que el coche tarda en responder a la orden).

Si con los dos primeros parámetros controlamos un interruptor y con los otros dos el otro interruptor, la tabla de comandos quedaría de la siguiente manera:

- 2 3 0 0 Adelante
- 4 5 0 0 Atrás
- 6 7 0 0 Giro a la derecha
- 8 9 0 0 Giro a la izquierda
- 2 3 6 7 Adelante con giro a la derecha
- 2 3 8 9 Adelante con giro a la izquierda
- 4 5 6 7 Atrás con giro a la derecha
- 4 5 8 9 Atrás con giro a la izquierda

Podemos ir creando ficheros *scripts* que se encarguen de realizar cada una de las tareas de movimiento, por ejemplo, si mi coche se mueve a una velocidad tal que tarde un segundo y medio en completar una circunferencia, el *script* que hace que el coche gire hacia la derecha sería:

```
#!/bin/sh
send_lpt 2 3 6 7 1500
```

Sólo nos queda ir programando el resto de las tareas. Un proyecto interesante sería crear un lenguaje similar al LOGO para hacer que el coche dibuje con su trayectoria las figuras que deseemos. Pero eso es otra historia. ■

Listado 1: send_lpt.c

```
01 /*                               25  }
02 * Uso: send_lpt pin_a pin_b      26
    pin_c pin_d tiempo              27  pin0 = atoi(argv[1]);
03 *                               28  pin1 = atoi(argv[2]);
04 * Envía un pulso alto (aprox.    29  pin2 = atoi(argv[3]);
    3.3V) a pin_a y a pin_c, y uno  30  pin3 = atoi(argv[4]);
    bajo (0V) a                      31  /* Pasamos los milisegundos
05 * los pines pin_b pin_c,        a microsegundos */
    espera 't' milisegundos y baja  32  t = atoi(argv[5]) * 1000;
    la entrada a y c.                33
06 */                               34  /* Encendemos pin0 y pin2 y
07 #include <stdio.h>                apagamos pin1 y pin 3*/
08 #include <stdlib.h>               35  pin_output_mode(LP_DATA_PINS
09 #include <unistd.h>               | LP_SWITCHABLE_PINS);
10                                   36
11 #include "parapin.h"              37  /* Metemos 0 V primero para
12                                   evitar dañar el diodo */
13 int main(int argc, char            38  if (pin1 > 0)
    *argv[])                          clear_pin(pin1);
14 {                                  39  if (pin3 > 0)
15  int pin0, pin1, pin2, pin3,      clear_pin(pin3);
    t;                                  40  if (pin0 > 0) set_pin(pin0);
16                                   41  if (pin2 > 0) set_pin(pin2);
17  if (argc < 6) {                  42
18  printf("Uso: %s pin_a            43  usleep(t);
    pin_b pin_c pin_d tiempo\n",      44
    argv[0]);                          45  /* Ponemos los pines 0 y 2 a
19  exit(-1);                          0 V otra vez */
20  }                                  46  if (pin0 > 0)
21                                   clear_pin(pin0);
22  if (pin_init_user(LPT1) < 0)      47  if (pin2 > 0)
    {                                    clear_pin(pin2);
23  printf("Error al                  48
    inicializar el puerto paralelo    49  return 0;
    (LPT1).\n");                       50  }
24  exit(-1);
```

RECURSOS

- [1] Datasheet del 4N25 <http://www.alldatasheet.co.kr/datasheet-pdf/view/QT/4N25.html>

EL AUTOR

Alberto Planas es desarrollador de aplicaciones bajo entornos libres desde hace varios años. Aficionado a la tecnología desde siempre, alterna sus horas de sueño con las horas dedicadas al estudio de las Redes Bayesianas, programación con las QT, perfeccionamiento de C++, desarrollo en Java y mil cosas más.

