PRÁCTICA DE ROBÓTICA 2 : Sensores

Carlos Tardón Rubio
Marcos Herrero Agustín
Martín Fernández de Diego

1 de febrero de 2022

Resumen

En esta práctica se montarán varios sensores que pueden utilizarse como elemento digital o analógico, pero que en nuestro caso serán utilizados como digital. Se programará su lectura y se mostrará su funcionamiento por pantalla. Después se implementará una arquitectura software que permita programar tareas repetitivas y utilizar el robot sin que se vea afectado por estas tareas.

1. Objetivos

- Montaje de los sensores que se usarán en el robot
- Uso básico de sensores con la RaspBerry Pi: lectura de sensores digitales.
- Diseño de programa para robot móvil: lectura de sensores.

2. Desarrollo Parte I : Programación de sensores en robótica

Usamos el siguiente código para realizar las lecturas:

```
#include <stdio.h>
#include <wiringPi.h>

// LED Pin - wiringPi pin 0 is BCM_GPIO 17.

#define LED     0

int main (void){
  printf ("Raspberry Pi blink\n") ;

  wiringPiSetup () ;
  pinMode (LED, INPUT) ;

  int time = 500;
  int i = 0;
  for (;;){
    delay (time);
    printf("%d\n", digitalRead(LED));
    i++;
  }
  return 0 ;
```

}

2.1. LDR

Desarrollo

Para entender el funcionamiento del sensor LDR basta con tratarlo como una resistencia dinámica que varía en función de la intensidad de la luz que reciba. Por lo tanto, basta con conectar los extremos del sensor a los 3.3V y al GND de la Raspberry Pi. Probamos su funcionamiento tomando con el voltímetro la tensión que hay entre dichos extremos.

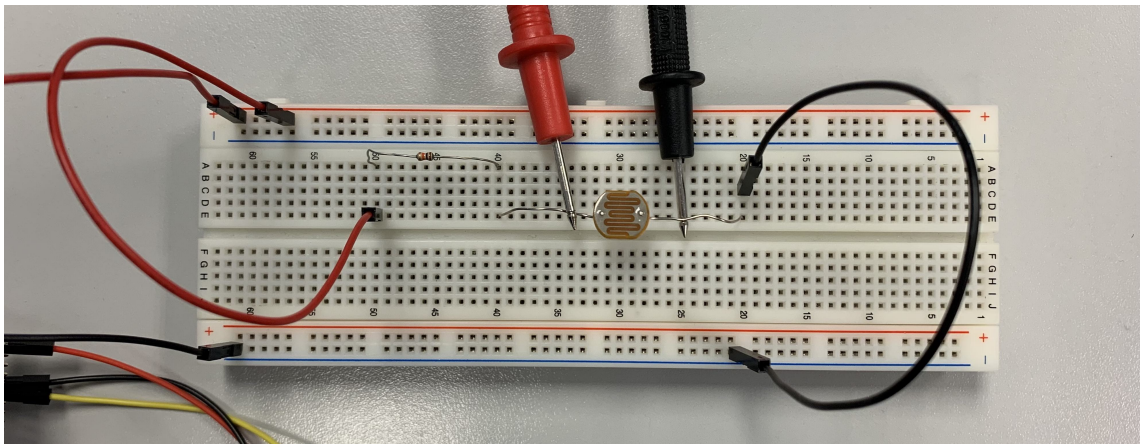

Figura 1: Esquema de conexiones del test con voltímetro del sensor LDR

Ejercicios

1.- Medir con el polímetro el valor de voltaje cuando:

a) Está iluminado únicamente con luz ambiente.

b) Se ilumina con una fuente de luz el LDR (por ejemplo con el led de un móvil).

c) Se tapa el LDR para que esté a oscuras

a) 0.4 V cuando está iluminado con luz ambiente.

b) 0.1 V cuando está iluminado con el flash del móvil.

c) 1.5 V cuando está tapado a oscuras.

Concluimos que cuanto más iluminado esté el sensor, menor es la resistencia que opone y, por tanto menor, la diferencia de potencial eléctrico entre los extremos del sensor LDR.

2.- Medir con la Raspberry Pi en los mismos casos anteriores. Indicar qué se obtiene. ¿Qué se puede concluir?

a) 0 cuando está iluminado con luz ambiente.

b) 0 cuando está iluminado con el flash del móvil.

c) 1 cuando está a oscuras.

Por tanto, leyendo la diferencia de potencial con un pin digital de la Raspberry Pi obtenemos un sensor con respuesta binaria. Devuelve 1 en estado de oscuridad y 0 en claridad sin dar información acerca de cambios de intensidad.

2.2. Sensor de Contacto

Desarrollo

El botón con tres láminas siempre permite el paso de corriente a través de dos de sus pines. Si el pin C ó COM (Common) está conectado a voltaje, es el encargado de suministrarla a los otros dos dependiendo del estado del sensor de contacto.

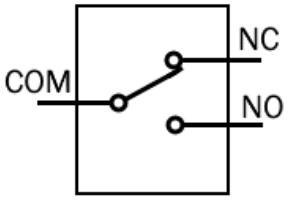

Figura 2: Esquema básico de conexiones botón de tres láminas

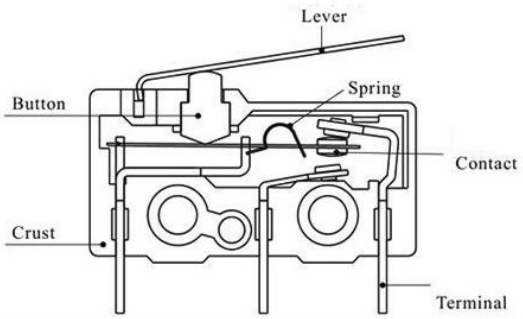

Figura 3: Esquema de conexiones palanca de tres láminas

Ejercicios

a) Comprobar con el polímetro qué dos láminas de las tres disponibles se unen cuando está pulsado el interruptor. ¿Y cuando no está pulsado?.

b) Indicar un circuito para que el sensor dé Vcc cuando no está pulsado y GND (0) cuando lo está.

c) Utilice 3.3V de la RaspBerry Pi para Vcc y lea los valores del interruptor utilizando una entrada digital.

a) Si no está pulsado, está en su posición normal, el pin NO (Normally Open) no está conectado con COM y el NC (Normally Closed) sí. Si está pulsado, el pin NO está conectado con COM y el NC no.

b) Conectamos VCC a NC, y GND a NO. El output del circuito es COM. De esta manera, cuando no está pulsado, el circuito tiene como output NC, es decir VCC, y cuando lo pulsamos obtenemos GND.

c) Basta con poner en el output del circuito (COM) el pin digital GPIO 17 de entrada. Leerá 1 cuando no está pulsado, y 0 cuando lo pulsemos.

Problemas presentados

- Dado que no teníamos el componente físico, no pudimos comprobarlo empíricamente pero sí recreamos el circuito interno del botón manualmente.

2.3. Sensor digital CNY70

Desarrollo

Conectamos el sensor CNY70 como se muestra en la figura 4.

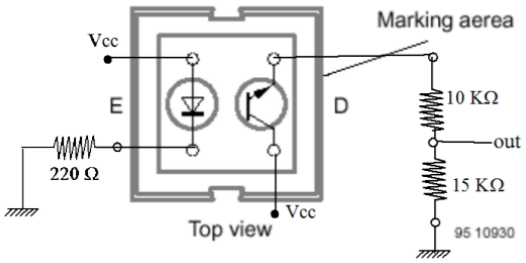

Figura 4: Esquema de conexiones del CNY70. Intento 1

Hicimos pruebas midiendo el voltaje entre el punto out y GND. Tras darnos cuenta de que debíamos situar el sensor muy cerca del objeto para que detectar el color adecuadamente, conseguimos que el voltaje se mantuviera máximo al mostrarle un objeto negro y cayera hasta 0 al mostrarle un objeto blanco. Sin embargo, la Raspberry no leía adecuadamente el valor (leía siempre 1).

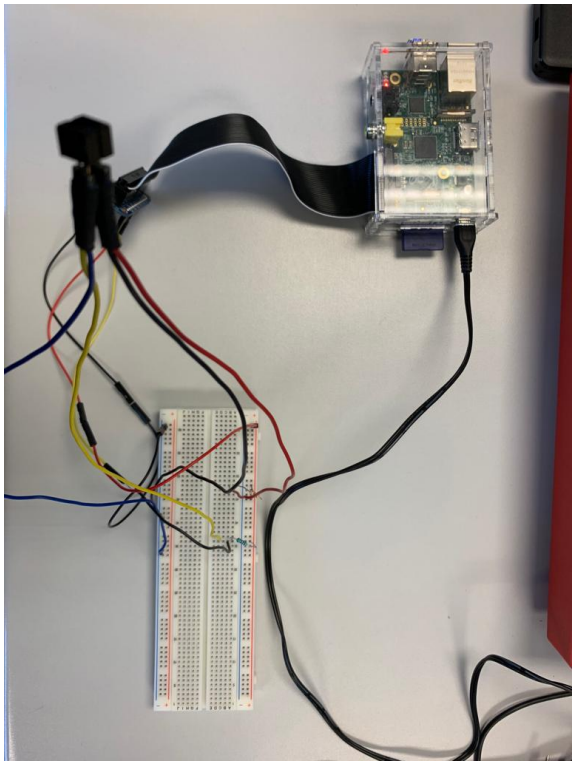

Figura 5: Esquema de conexiones del CNY70. Intento 2

El profesor nos ayudó a diseñar un circuito ligeramente distinto (el que se muestra en la figura 5) que, ahora sí, nos funcionó correctamente tanto en la lectura con voltímetro como en la lectura con la Raspeberry.

Ejercicios

a) Comprobar con el polímetro el rango de funcionamiento del sensor. Observe la distancia o alcance dependiendo del color del objeto; y los valores de voltaje que da cuando ve el color blanco y cuando ve el color negro.

b) Conecte la entrada digital de la RaspBerry Pi en el punto out y verifique que mide cuando se le muestra un objeto negro y uno blanco. Compruebe el efecto según a la distancia que se coloque el objeto.

a) Al apuntar el CNY70 a un objeto negro, se obtiene voltaje máximo (3,3 V) a distancia corta (1-2 mm), que disminuye progresivamente al alejar el sensor del objeto.
Al dirigirlo a un objeto blanco, se obtiene voltaje 0 a cualquier distancia inferior a 10 mm.

b) La Raspberry lee el valor 1 cuando el CNY70 se apunta a un objeto negro y lee el valor 0 cuando el sensor se apunta a un objeto blanco. Esto es cierto para cualquier distancia inferior a 10 mm. A mayores distancias, el resultado es impredecible.

Problemas presentados

- Uno de nuestros primeros fallos fue no acercar suficientemente el sensor al objeto. El CNY70 tiene un alcance muy bajo, de unos 2 mm
- Inicialmente, funcionaba correctamente la caída de voltaje al enfrentar el sensor con un objeto blanco (pasaba a leerse 0 entre out y GND). Sin embargo, esta diferencia de voltaje no era captada adecuadamente por la Raspberry Pi.
- Después de muchos intentos y varios diseños alternativos, no solo no conseguíamos que la Raspberry leyera los valores adecuados sino que además dejamos de ser capaces de replicar el experimento inicial en el que el voltaje si caía adecuadamente.
- Una de las cosas que nos fallaron durante mucho tiempo fue utilizábamos el pin 18, que no está pensado para este propósito.
- Desde los primeros ensayos que funcionaron bien hasta que conseguimos replicarlos pasó mucho tiempo.

2.4. Arquitectura software del robot

Desarrollo

Utilizamos los aprendido en las asignaturas de Sistemas Operativos y Programación Concurrente para realizar el programa pedido. Nuestro programa (en concreto, el thread llamado ldrThread) lee del pin 17 el valor del LDR, pone la variable Vldr a 1, y actualiza la variable global ldr con el valor leído. Como usamos threads, es imprescindible leer y escribir en las variables Vldr, Vcny, ldr y cny con exclusión mutua, para lo que utilizamos los locks de wiringpi, a los que se accede con piLock(id) y piUnlock(id), donde id es el identificador del semáforo accedido. Nosotros usamos 0 para el semáforo que bloquea las variables ldr y Vldr, y 1 para el semáforo de las variables cny y Vcny. En cuanto al hilo principal, el main se encarga de comprobar que haya lecturas disponibles (mediante las variables Vldr y Vcny), e imprime el valor correspondiente. El bucle del main se ejecuta durante MAXTIME microsegundos.

Problemas presentados

- No nos decidíamos sobre cómo hacer para que cada iteración del bucle durara Tldr milisegundos. Al final, optamos por un diseño rudimentario pero efectivo : medir cuánto llevábamos de iteración (usando gettimeofday) y haciendo, al final de ella, un delay de lo que falta hasta completar los Tldr milisegundos.

Código

```
#include <stdio.h>
#include <wiringPi.h>
#include <time.h>
#include <sys/time.h>
#include <stdlib.h>

#define Tldr     100
#define Tcny     100
#define TMAX     10000
#define MAXTIME  10000000 // 10s en microsegundos
#define LDR      0
```

```
#define CNY    1

int Vldr = 0; // indican si hay lectura o no
int Vcny = 0;
int ldr; // aqui aparece el valor leido
int cny;


PI_THREAD (ldrThread){ // Usa el lock con key=0
    wiringPiSetup () ;
    pinMode (LDR, INPUT);

    int i = 0;
    struct timeval tv;
    int last;
    while(1){
        gettimeofday(&tv,NULL); //llamada al sistema para obtener el tiempo actual
        last = tv.tv_usec/1000; // el tiempo actual

        int lectura = digitalRead(LDR);
        printf("Leyendo ldr en thread: %d\n", lectura);
        piLock(0);
        Vldr = 1; // indicamos que hay lectura disponible
        ldr = lectura; // copiamos en la variable global
        piUnlock(0);
        i++;

        gettimeofday(&tv,NULL);
        delay(Tldr - (tv.tv_usec/1000 - last)); // Queremos que cada iteracion del bucle dure Tldr,
    }
}

PI_THREAD (cnyThread){ // Usa el lock con key=1
    wiringPiSetup () ;
    pinMode (CNY, INPUT);

    int i = 0;
    struct timeval tv;
    int last;
    while(1){
        gettimeofday(&tv,NULL); //llamada al sistema para obtener el tiempo actual
        last = tv.tv_usec/1000; // el tiempo actual

        int lectura = digitalRead(CNY);
        printf("Leyendo CNY en thread: %d\n", lectura));
        piLock(1);
        Vcny = 1; // indicamos que hay lectura disponible
        cny = lectura; // copiamos en la variable global
        piUnlock(1);
        i++;

        gettimeofday(&tv,NULL);
        delay(Tcny - (tv.tv_usec/1000 - last)); // Queremos que cada iteracion del bucle dure Tldr,
    }
}
```

```
int main (void){
    printf ("Raspberry␣Pi␣blink\n") ;
    piThreadCreate(ldrThread);
    piThreadCreate(cnyThread);
    struct timeval tv;
    gettimeofday(&tv,NULL);
    int initialTime = tv.tv_usec;

    while( tv.tv_usec - initialTime < MAXTIME){

        piLock(0);
        if(Vldr == 1){
            printf("Leyendo␣ldr␣en␣MAIN:␣%d\n", ldr);
            Vldr = 0;
        }
        piUnlock(0);

        piLock(1);
        if(Vcny == 1){
            printf("Leyendo␣cny␣en␣MAIN:␣%d\n", cny);
            Vcny = 0;
        }
        piUnlock(1);

        gettimeofday(&tv,NULL);
    }

    return 0 ;
}
```

3. Desarrollo Parte 2 : Programación de sensores en el robot

3.1. GP2D12

Desarrollo

Conectamos el chip MCP3008 y el sensor GP2D12 como se observa en la figura 1.

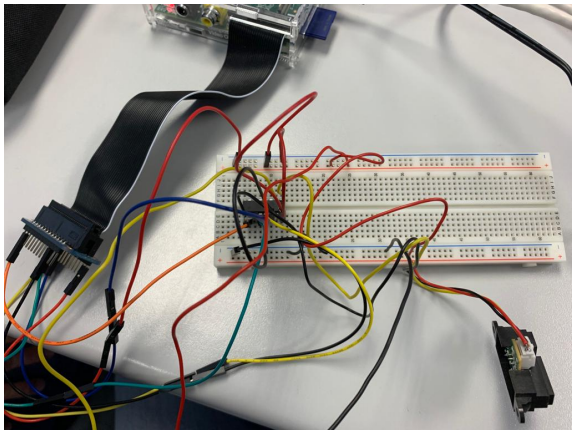

Figura 6: Esquema de conexiones GP2D12

Ejercicios

- **Conecte la salida out al canal 0 (CH0) del MCP3008 y muestre el valor del canal por pantalla. Anote los valores para diferentes distancia del obstáculo. Represente gráficamente los resultados.**
- **Conecte el otro GD2Y0A41SK0F al canal 1 (CH1) y realice la misma operación que anteriormente. Puede mostrar los dos a la vez en la mísma gráfica.**

Para tomar los valores para diferentes distancias, marcamos en un papel algunas distancias representativas. Poniendo el sensor en el borde del papel, y objetos a la altura de cada una de las marcas, observamos los valores que imprimía el programa de ejemplo mcp3008.c.

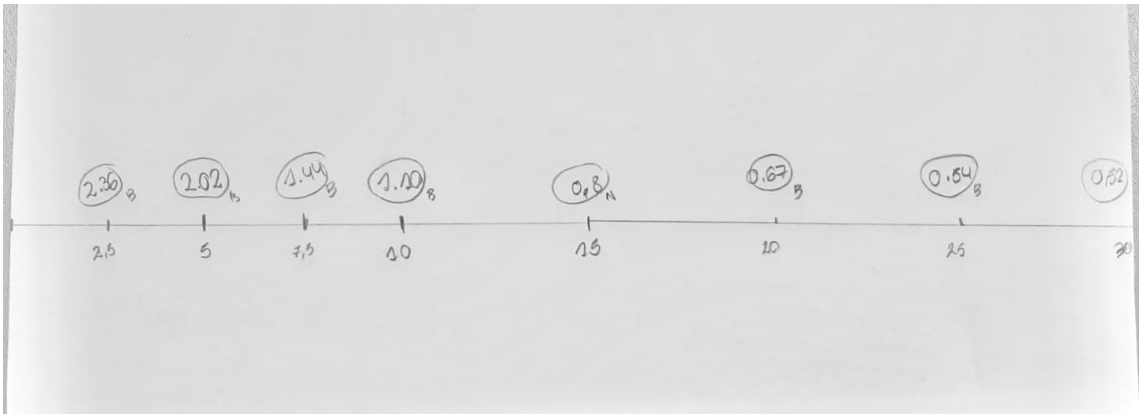

Figura 7: Marcas en el papel

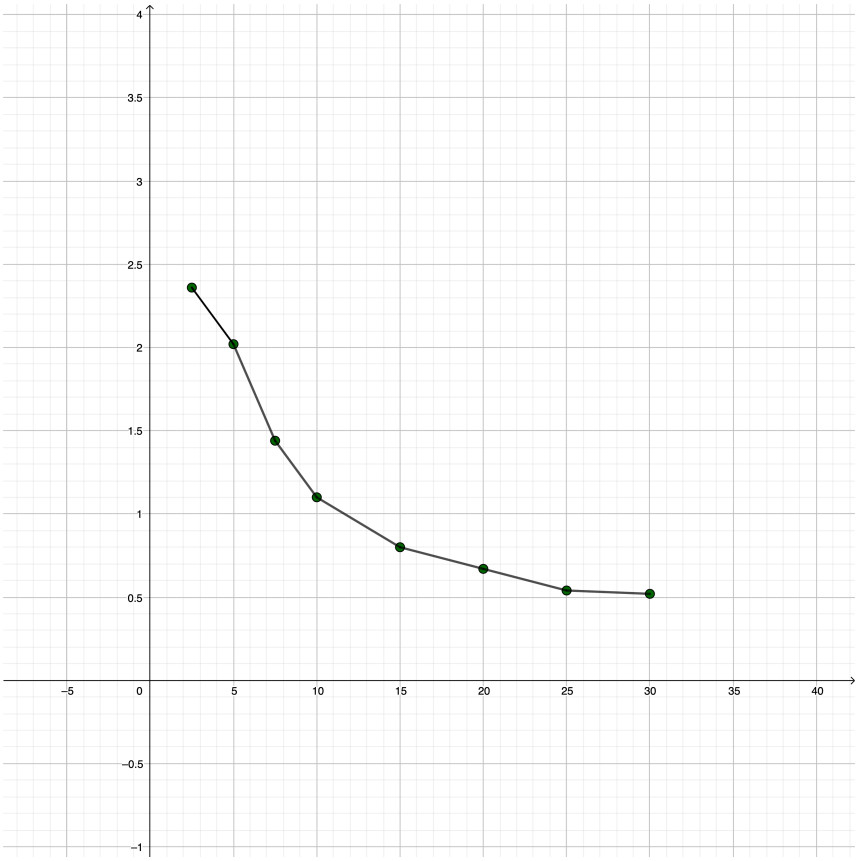

Figura 8: Graficado de las marcas

En el eje X la distancia respecto del sensor, y en el eje Y el voltaje del canal leído. Tras haber tomado los ocho pares y haberlos colocado en el plano, observamos como la variación de voltaje del canal disminuye con el aumento de la distancia; es decir, su precisión disminuye con la distancia.

Problemas presentados

- El voltaje producido por el circuito variaba mucho con distancias largas (25cm o más), por lo que decidimos hacer una media de los valores.