Informe Final

Daniela Alejandra Córdova Porta
UNIVERSIDAD COMPLUTENSE DE MADRID
Madrid, febrero de 2021

Resumen

Este documento expone el procedimiento para el montaje del robot para la asignatura Robótica en la Universidad Complutense de Madrid. Igualmente se describe cómo se realizó la construcción y configuración del robot para el circuito de seguimiento de líneas y el seguimiento de paredes/puerta.

Índice

1. Objetivos

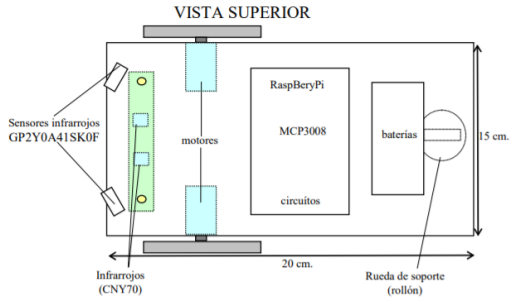
- Montaje del robot para seguimiento de líneas.
- Montaje del robot para el seguimiento de paredes/puertas.

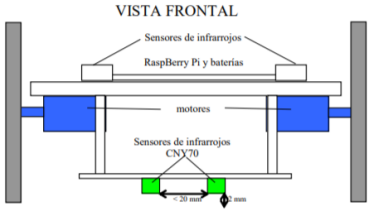
2. Procedimiento

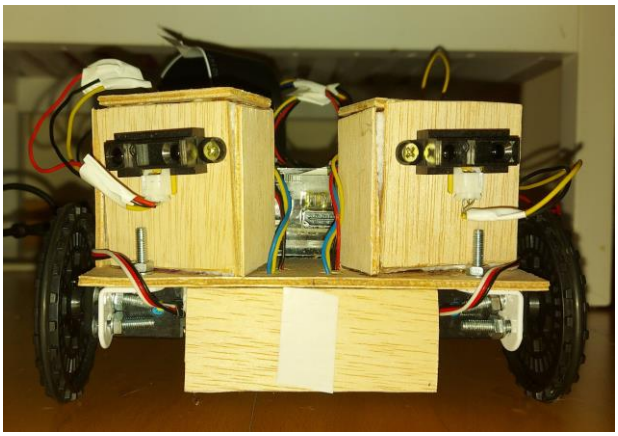
3. Montaje

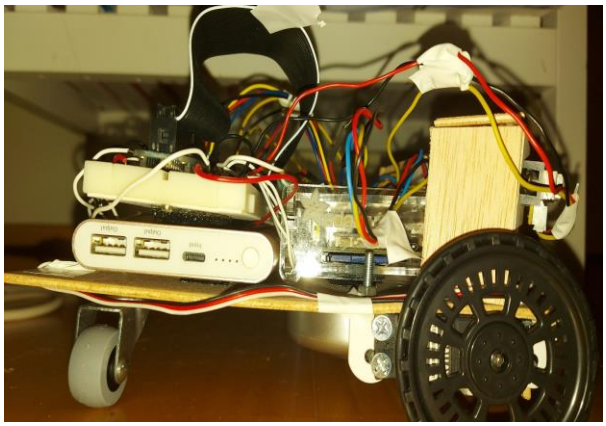
El robot consiste en una placa de madera 20x10 cm a la que se le atornilló unas ruedas a los lados usando “L’s” de metal y tornillos. Luego se colocó una rueda loca en la parte de atrás, para soportar el peso restante. Se situó la raspberry en el medio del robot detrás de los sensores GP2D12 y al final de este se encuentra la batería y encima la protoboard.

Se montaron los sensores del robot siguiendo el montaje explicado en la práctica de sensores II. Los sensores GP2D12 se colocaron en bases de poliestireno cubiertas de placas de madera a los lados del robot apuntando a los extremos y los sensores CNY70 en otro poliestireno acomodados para que tengan una distancia entre ellos de 20 mm y una distancia de 2mm con el suelo. Todo esto siguiendo la siguiente imagen:







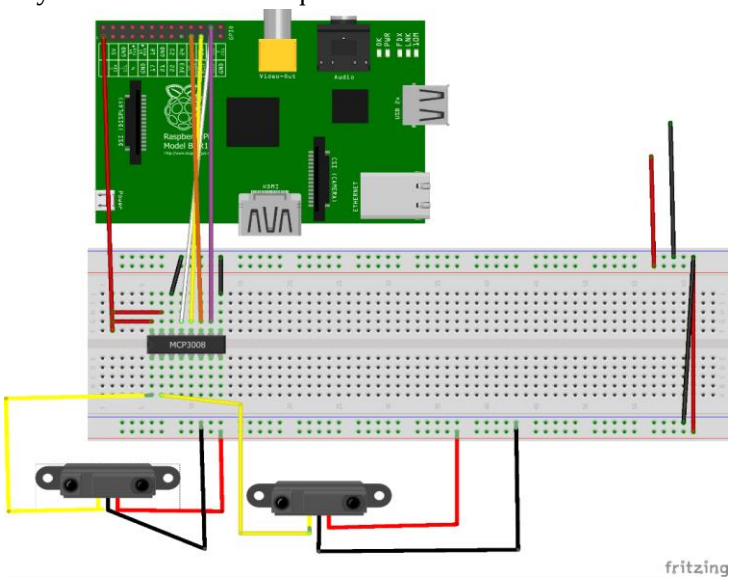


3.1 Sensores

Para los dos sensores del robot se hicieron los siguientes circuitos usando el convertidor analógico MCP3008:

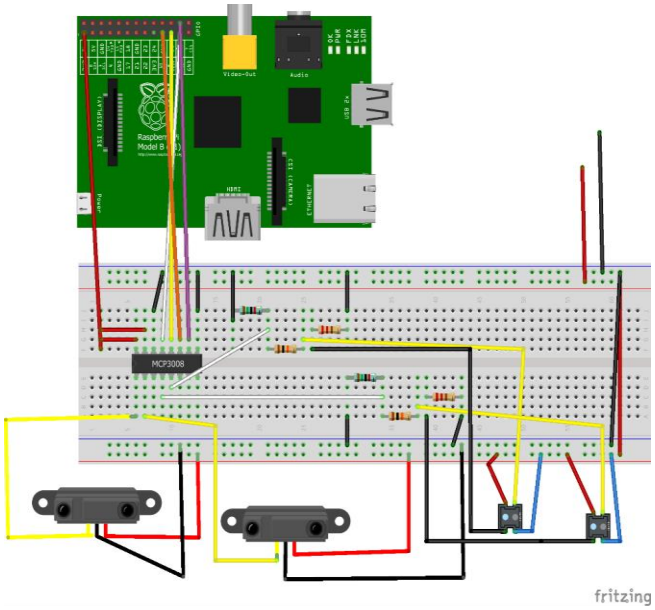
3.1.1 Sensor Analógico (GP2D12 o GD2Y0a41SK0F):

Conectamos la salida del cable amarillo del sensor del lado derecho con el canal CHO del MCP3008 y el sensor del lado izquierdo con el canal CH1.



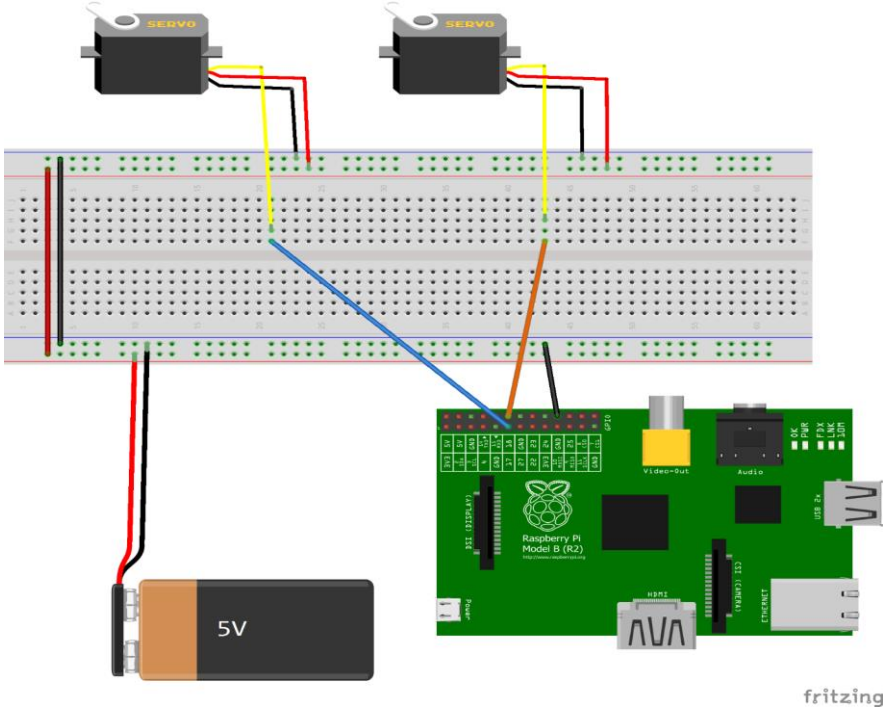
3.1.2 Sensor digital (CNY70):

Conectamos el sensor con las resistencias necesarias y la salida entre las resistencias 10K ohm y 15K ohm con uno de los canales del convertidor. El sensor del lado izquierdo se conecta con el canal CH3 y el lado derecho con el canal CH2.



3.2 Ruedas

El circuito para las ruedas es el siguiente. Se usó el GPIO 17 y el 18.



4. Programación y Configuración

Para los dos seguimientos se hicieron códigos en c que configuran las acciones del robot.
Se usaron las librerías:

```
#include <stdlib.h>
#include <softPwm.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <wiringPi.h>
#include <softPwm.h>
```

```
#include <wiringPiSPI.h>
#include <mcp3004.h>
```

Las librerías importantes para mencionar son las siguientes:

- SoftPwm: que se encarga de las ruedas y la creación del pulso que genera las velocidades y dirección.
- Mcp3004 para la configuración del chip usado para la lectura de los sensores
- WiringPi: para configurar todas las salidas y entradas de la Raspberry

4.1.Seguimiento de líneas:

En el caso de esta práctica, se configuró el uso del mcp3004, WiringPi y la creación de los pulsos Pwm. Luego de manera indefinida se hacen lecturas analógicas a las direcciones de pin de los sensores CNY. Esta da valores que fueron analizados para el caso de que refleje o no el infrarrojo (lea blanco o negro).
Dependiendo de los valores devueltos el código realiza las siguientes acciones:

- Si lee 2 valores altos: Sigue recto con velocidad normal
- Valor del sensor derecho indica no reflexión: se para rueda derecha y avanza la izquierda
- Valor del sensor izquierdo indica no reflexión: se para rueda izquierda y avanza la derecha
- Si ninguno de estos, se detiene.

Para cada condición se tomó en cuenta que, si lee valores menores, el robot realiza la acción y luego se espera unos segundos en parada para ver si es necesario seguir avanzando o se debe hacer otro ajuste para el recorrido.

4.2.Seguimiento de la pared:

Para este se hicieron estructuras que reservan una base de datos de cuánta distancia corresponde a cada valor devuelto por los sensores. Igualmente se configura el mcp3004, wiringPi y la creación de pulsos para las ruedas.

Este algoritmo de manera indefinida lee los valores analógicos y realiza una búsqueda binaria en la base de datos para determinar a qué distancia con respecto de la pared se refiere ese valor.

- Si el valor para los sensores es mayor o igual a 20: el robot avanza hacia delante.
- Si encuentra algún sensor con valores menores, determina si debe seguir avanzando según lo siguiente:
    - Si el otro sensor no encuentra obstáculo, es decir, el valor del otro sensor corresponde a un valor mayor de una distancia de 20 cm con respecto a la pared: sigue hacia adelante ya que el robot se encuentra paralelo a una pared.
    - Si el otro sensor encuentra obstáculo, se tiene que ver quién está más cerca de la pared. El de menor distancia determina el lado. Si es el izquierdo, nos movemos a la derecha y si es el derecho, a la izquierda

Se hicieron varias versiones de este algoritmo. La usada para la entrega hace que el robot gire 90° y avance como lo indicado anteriormente. Otra versión hace que el robot verifica cada 45° cómo debe ajustarse con respecto a lo que tiene a su alrededor, se hizo de esta manera ya que se veía que funcionaba mejor si se introducía el robot en un laberinto.

5. Pruebas y análisis de resultados

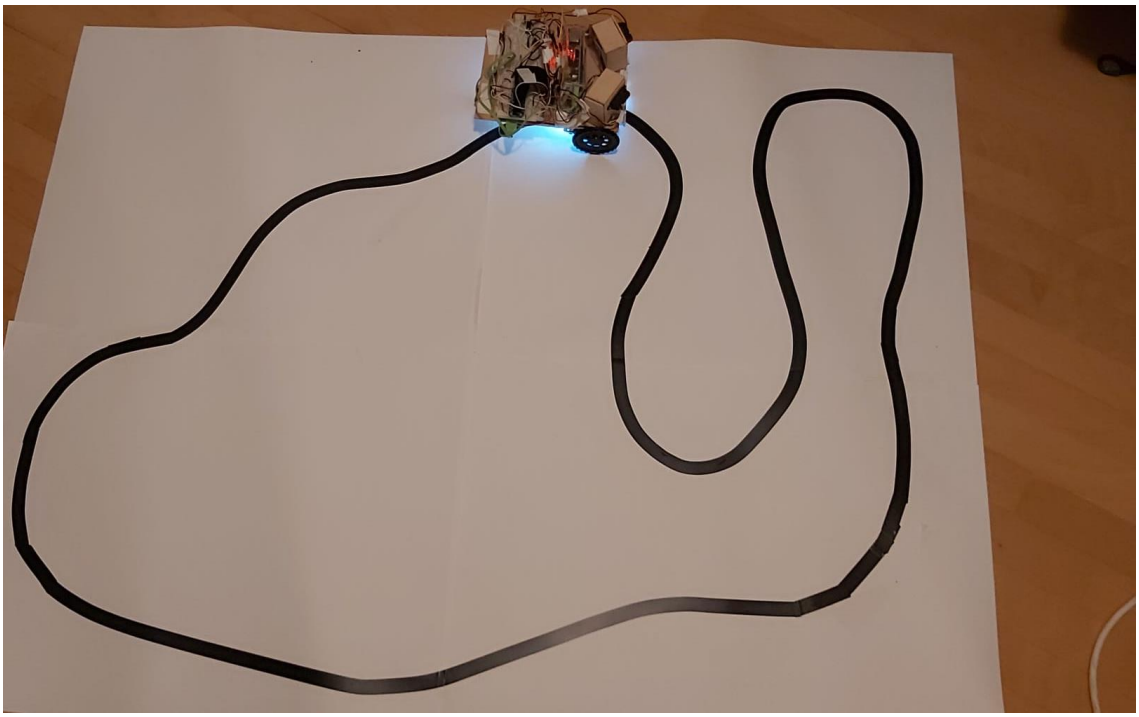
Con los programas ubicados en el apéndice, se observó lo siguiente

5.1. Seguimiento de líneas

Mientras más curvilíneo es el circuito, había mayor necesidad de ajustar delays con respecto a la lectura de datos y procesamiento del caso a ejecutar. Se debe tener cuidado en la asignación correcta de los sensores ya que puede causar confusión si se asignan incorrectamente los datos.

Se necesitan los dos sensores ya que el robot necesita ajustar correctamente su posición en las curvas, con uno no se podría determinar a cuál dirección se debe ir y necesitan estar separados por 20 cm exactos y estar muy ajustados al piso para evitar problemas.

Si el robot no está montado adecuadamente, varían los resultados por lo cual debe estar todo bien construido y con las medidas exactas recomendadas.



Ejecutando el algoritmo se obtiene el siguiente resultado que se puede ver en el video:
https://drive.google.com/file/d/1iAoXO3Q3kN0oOZPJOjqm7M0yCiDrbYH-/view?usp=sharing

5.2. Seguimiento de las paredes/puerta

Se deben colocar los sensores perpendiculares a la base del robot y dirigiéndose hacia los lados para poder determinar cuál lado tiene pared y cuál no para poder realizar el seguimiento. Los valores para determinar las distancias deben ser medidos una vez ya montados los sensores en el robot para evitar diferencias en los valores. Se debe conocer el tiempo que tarda cada rueda en hacer girar 90° al robot para seguir correctamente las paredes. Al igual que los sensores del seguimiento de líneas, se requiere parar el robot durante unos segundos para que valores de los sensores sean acertados.

https://drive.google.com/file/d/1iDHDEojDCMwizcmkJt1eQTNO9_ymrjrt/view?usp=sharing

6. **Conclusiones:**

- Se necesitan los dos sensores de cada tipo para hacer ajustes correctos para el seguimiento de lo que se desea
- El montaje debe ser apropiado y proporcionado a las medidas dadas por el profesor ya que si el robot es muy grande puede inclinarse en los movimientos. Igualmente, si los sensores están mal colocados, a pesar de que la programación es la correcta, el robot no realizará las acciones que se desean ya que no percibe los valores adecuadamente.

7. **Errores:**

El mayor error encontrado fue que no se esperaba lo suficiente para poder leer los sensores y por ello no realizaba las acciones que se deseaban. Si se ajustan los tiempos con delays para la correcta lectura de estos, el robot funciona apropiadamente.

8. Apéndice:

8.1.Código para el seguimiento de la línea negra

```
#include <stdlib.h>
#include <softPwm.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <wiringPi.h>
#include <softPwm.h>
#include <wiringPiSPI.h>
#include <mcp3004.h>

#define BASE 100
#define SPI_CHAN 0
#define RANGE 100

#define PIN0 0
#define PIN1 1

#define PIN_CNY_izq 102
#define PIN_CNY_dr 103
#define cerrojo 0

int blancoIzq=1;
int blancoDr=1;
int main(void){


   mcp3004Setup (BASE, SPI_CHAN); // 3004 and 3008 are the same 4/8 channels

   wiringPiSetup();

   softPwmCreate(PIN0, 0, RANGE);
   softPwmCreate(PIN1, 0, RANGE);

   while(1){
   int valorIZ=analogRead (PIN_CNY_dr) ;
   int valorDR=analogRead (PIN_CNY_izq) ;

   printf("Valor IZQ: %d\n", valorIZ);
   printf("Valor DR: %d\n", valorDR);

   if( valorDR >356 &&valorIZ>2){
      softPwmWrite (0, 10) ;
      softPwmWrite (1, 35) ;
```

```
    }
    else if (valorDR <=600){
        softPwmWrite (0, 0);
        softPwmWrite (1, 30);//izq
        printf("DR!!!!!!!!!!!\n");
    delay(5);
    softPwmWrite (0, 0) ;
    softPwmWrite (1, 0) ;   //Dr
    delay(20);
    }
    else if(valorIZ<=1 )
    {
        softPwmWrite (0, 10) ; //DR
        softPwmWrite (1, 0) ;
            printf("IZ!!!!!!!!!!!!!!!!\n");
    delay(5);
    softPwmWrite (0, 0) ;
    softPwmWrite (1, 0) ;   //Dr
    delay(20);
    }


    else
    {
        softPwmWrite (0, 0) ;
        softPwmWrite (1, 0) ;   //Dr
    }


    }

    return 0;
}
```

8.2.Código para el seguimiento de la pared

```
#include <stdlib.h>
#include <stdio.h>
#include <errno.h>
#include <string.h>

#include <wiringPi.h>
#include <softPwm.h>
#include <wiringPiSPI.h>
#include <mcp3004.h>

#define BASE 100
#define SPI_CHAN 0
#define RANGE 100

#define PIN0 0
#define PIN1 1

#define PIN_GP2_izq 101
#define PIN_GP2_dr 100
#define cerrojo 0
#define SIZE_OF_MAP 20


struct Map {
   int key;
   int value;
};

struct Map map1[SIZE_OF_MAP]= {{52, 60}, {98, 50}, {116, 40}, {134, 35}, {160, 30}, {184, 25}, {200, 20},{220, 18}, {271, 17}, {298, 15},
          {332, 13}, {406, 10}, {624, 7}, {674, 6}, {720, 5}, {860, 4}, {950, 3}};

struct Map mapDr[SIZE_OF_MAP]= {{52, 60}, {98, 50}, {102, 40}, {114, 35}, {126, 30}, {144, 25},{182, 21}, {195, 20},{212, 18}, {230, 17}, {274, 15},
          {332, 13}, {400, 10}, {624, 7}, {674, 6}, {800, 5}, {860, 4}, {950, 3}};
struct Map mapIz[SIZE_OF_MAP]= {{52, 60}, {68, 50}, {80, 40}, {99, 35}, {110, 30}, {149, 25},{184,22}, {204, 20},{234, 18}, {220, 17}, {245, 15},
          {332, 13}, {400, 10}, {624, 7}, {674, 6}, {770, 5}, {860, 4}, {950, 3}};


int binarySearch(struct Map arr[], int l, int r, int x, int n) ;
void haciaDelante(){

    softPwmWrite (0, 10) ;
    softPwmWrite (1, 35) ;
```

```
    delay(50);
     //delay(1500);
}

void derecha(){
    softPwmWrite (0, 10);  //Si quieres derecha, solo rueda Izq
    softPwmWrite (1, 0);
    delay(1400);
}

void izquierda(){
    softPwmWrite (0, 0);
    softPwmWrite (1, 80); //Si quieres izquierda, solo rueda Dr
    delay(2000);
}

void haciaAtras(){
    softPwmWrite (0, 38) ;
    softPwmWrite (1, 10) ;
    delay(1000);
}

void stop(){
    softPwmWrite (0, 0) ;
    softPwmWrite (1, 0) ;
    delay(2000);
}
int main(void){


    int n = sizeof(map1) / sizeof(map1[0]);

    mcp3004Setup (BASE, SPI_CHAN); // 3004 and 3008 are the same 4/8 channels

    wiringPiSetup();

    softPwmCreate(PIN0, 0, RANGE);
    softPwmCreate(PIN1, 0, RANGE);
    int valor1, valor2, indexIz=0, indexDr=0;

    while(1){
    valor1=analogRead (PIN_GP2_izq) ;
    valor2=analogRead (PIN_GP2_dr) ;


    indexIz = binarySearch(mapIz, 0, n-1, valor1, n);
    indexDr = binarySearch(mapDr, 0, n-1, valor2, n);
    printf("Valor iz: %d\n", valor1);
    printf("Valor dr: %d\n", valor2);
```

```
   printf("valorIzq: %d\n", mapIz[indexIz].value);
   printf("valorDr: %d\n", mapDr[indexDr].value);

  if(mapIz[indexIz].value>20 && mapDr[indexDr].value> 20){
      haciaDelante();
   }
   else if(mapIz[indexIz].value<= 20 || mapDr[indexDr].value<= 20){


      if(mapIz[indexIz].value> mapDr[indexDr].value)
      {
                      if(mapIz[indexIz].value>30)
                              haciaDelante();
                      else{
                          stop();
              derecha();
             }
      }
      else
      {
                 if(mapDr [indexDr].value>30)
                              haciaDelante();

                      else{
                          stop();
              izquierda();
                }

      }

   }

}
   return 0;
}

int binarySearch(struct Map arr[], int l, int r, int x, int n)
{
   if (r >= l) {
      int mid = l + (r - l) / 2;

      // If the element is present at the middle
      // itself
      if (arr[mid].key <= x && mid<n && arr[mid+1].key > x)
         return mid;
      else if (arr[mid].key >=x && mid==0)
         return mid;
```

```
        else if (arr[mid].key<=x && mid==n-1)
            return mid;
        // If element is smaller than mid, then
        // it can only be present in left subarray
        if (arr[mid].key > x)
            return binarySearch(arr, l, mid - 1, x, n);

        // Else the element can only be present
        // in right subarray
        return binarySearch(arr, mid + 1, r, x,n);
    }

    // We reach here when element is not
    // present in array
    return -1;
}
}
```