Memoria

Sesión 3

Robótica Móvil

Laboratorio de Robótica

Andreu Morro Gallego

Carlos Pimentel Martorell

José Luís Delgado Arce

Joaquín Barrachina

Objetivos:

- **a)** Realizar un programa que permita que el robot se desplace paralelo a una pared.
- **b)** Realizar un programa similar al anterior pero que se posicione a una distancia indicada con la mayor precisión posible.
- **c)** Realizar un test tal que: El robot situado a 50 cm de la pared debe intentar posicionarse a 30 cm de la misma para luego alejarse hasta los 40 cm.

El conexionado es igual al de las anteriores sesiones con la diferencia principal de que los sensores ahora están situados en la misma protoboard pero en el lateral derecho del robot móvil y la distancia que los separa es de d=14.5cm, dato importante, como se verá luego, para el correcto funcionamiento de la orientación al desplazarse.

M1_Izq Pin 46 //Direccion High: avanzar 46

M1_Der Pin 48 //Direccion High:Retroceder 48

M2_Izq Pin 36 //Direccion High:retroceder 36

M2_Der Pin 38 //Direccion High: avanzar 38

PWM_Der Pin 3

PWM_Izq Pin 5

Eco Pin 50 //Sensor Izquierdo

Eco2 Pin 53 //Sensor Derecho

Disparo Pin 51 //Sensor Izquierdo

Disparo2 Pin 52 //Sensor Derecho

Nota: Solo se conservan datos de telemetría, y por ende, gráficas del último apartado pues es el más completo y el que engloba y une las acciones del primer y segundo apartado además de acciones adicionales.

a) El algoritmo de este apartado viene a ser sencillo, solo tienen que igualarse (de forma aproximada) las distancias de cada sensor de cada extremo girando sobre sí mismo) y avanzar.

   En el siguiente algoritmo se encuentra el esquema que compone la función *void loop()*, la función *void setup()* para declarar el tipo de puertos y las funciones usadas sigue siendo el mismo que el del primer ejercicio de robótica móvil.

```
switch (estado)
{

case 'a': //alinearse

 Serial1.println("Alineandose");
 distancia= calculardistancia()*10; distancia2= calculardistancia2()*10;
 dist= (int)distancia;
 dist2= (int)distancia2;
 if(dist>dist2)
 {
  if(flag1=1){stop();flag1=0;}
  girar(0, 100);
  Serial1.print("  -100  100 ");
 } //girar(0,v)
 else if (dist<dist2)
 {
  if(flag1=0){stop();flag1=1;}
  girar(1, 100);
  Serial1.print("  100  -100 ");
  } //girar(1,v)
 else {estado='p'; stop();}

break;

case 'p': //posicionarse

 phi=0;   //Para mantenerse en paralelo con la pared
 //n=tan(phi)*d*10;
 angulo=atan((dist2-dist)/d*10)*10;
 Serial1.print(int(angulo));
 Serial1.print(" ");
 distancia= calculardistancia();
 dist= (int)distancia;   //Serial1.print(distancia); Serial1.print("  ");
 distancia2= calculardistancia2();
 dist2= (int)distancia2;  //Serial1.print(distancia2);  Serial1.print(" ");
```

```
   if(int(angulo)<int(phi*10))/*dist>dist2-n)*/ //De esta manera se busca dist =
dist2+n
   {
    if(flag1=1){stop();flag1=0;}
    girar(0, 110);
    Serial1.print("0  100 ");
   }
   else if (int(angulo)>int(phi*10))/*dist<dist2-n)*/
   {
    if(flag1=0){stop();flag1=1;}
    girar(1, 110);
    Serial1.print("100  0 ");
   }
   else {
      avanzar(180); Serial1.print("180  180 "); //avanzar
      }

   break;

 }
 time=millis();
 Serial1.print(time);
 Serial1.println();
 delay(50);
}


///////Funciones

//Motor 1 rueda izquierda, relacionado con Disparo del sensor 1
//Motor 2 rueda derecha, relacionado con Disparo2 del sensor 2
void avanzar(int v){
   digitalWrite(M1_Izq, HIGH);
   digitalWrite(M1_Der, LOW);
   digitalWrite(M2_Izq, LOW);
   digitalWrite(M2_Der, HIGH);
   analogWrite(PWM_Der,v);
   analogWrite(PWM_Izq,v);
 }

void retroceder(int v){
   digitalWrite(M1_Izq, LOW);
   digitalWrite(M1_Der, HIGH);
   digitalWrite(M2_Izq, HIGH);
   digitalWrite(M2_Der, LOW);
   analogWrite(PWM_Der,v);
   analogWrite(PWM_Izq,v);
 }
```

```
void girar(int flag, int v){
  if (flag==0) {    //Giro de sentido horario
    digitalWrite(M1_Izq, HIGH);
    digitalWrite(M1_Der, LOW);
    digitalWrite(M2_Izq, LOW);// HIGH);
    digitalWrite(M2_Der, LOW);
    analogWrite(PWM_Der, v);
    analogWrite(PWM_Izq, v);
    }
  else{    //Giro de sentido antihorario
    digitalWrite(M1_Izq, LOW);
    digitalWrite(M1_Der, LOW);// HIGH);
    digitalWrite(M2_Izq, LOW);
    digitalWrite(M2_Der, HIGH);
    analogWrite(PWM_Der, v);
    analogWrite(PWM_Izq, v);
    }
  }

void stop(){
    digitalWrite(M1_Izq, LOW);
    digitalWrite(M1_Der, LOW);
    digitalWrite(M2_Izq, LOW);
    digitalWrite(M2_Der, LOW);
    analogWrite(PWM_Der,0);
    analogWrite(PWM_Izq,0);
  }

float calculardistancia(){
  unsigned long tiempo=sonar.ping();
  float distancia=tiempo*0.000001*sonido/4.0;
  Serial1.print(int(distancia));
  Serial1.print(" ");
  //Serial1.println();

  return distancia;
}

float calculardistancia2(){
  unsigned long tiempo=sonar2.ping();
  float distancia=tiempo*0.000001*sonido/4.0;
  Serial1.print(int(distancia));
  Serial1.print(" ");
  //Serial1.println();

  return distancia;
}
```

Por dar una explicación resumida, consiste en 2 casos, aunque uno de ellos es redundante pero será necesario para indicarle al robot desde que posición parte, sobretodo en el resto de apartados. Simplemente busca el ángulo que tiene con la pared en todo momento.
Las funciones arriba expuestas serán las mismas en los demás algoritmos del resto de apartados y se intuyen fácilmente su uso.

Puede verse los resultados obtenidos en el archivo *'practica3_1.mp4'*.

b) y c) Se incluyen los 2 apartados en 1 puesto que el c) es solo la realización de un test en base al primero sin ningún cambio considerable. Dicho esto se muestra abajo el algoritmo que compone a *void loop().*

```
switch (estado)
 {

 case 'a': //alinearse
  Serial1.println("Alineandose");
  distancia= calculardistancia()*10; distancia2= calculardistancia2()*10;
  dist= (int)distancia;
  dist2= (int)distancia2;
  if(dist>dist2)
  {
   if(flag1=1){stop();flag1=0;}
   girar(0, 100);
   Serial1.print("  -100  100 ");
  } //girar(0,v)
  else if (dist<dist2)
  {
   if(flag1=0){stop();flag1=1;}
   girar(1, 100);
   Serial1.print("  100  -100 ");
   } //girar(1,v)
  else {estado='d'; stop();}
 break;

 case 'd': //dato
   if(dist2!=umbral){
   Serial1.println("Esperando dato");
   //umbral=Serial1.read();
   Serial1.print("Recibida la distancia ");
```

```
    Serial1.print((int)umbral);
    Serial1.println();
    }
    else delay (1000);

    estado='p';
 break;

 case 'p': //posicionarse
 //if(dist2!=(umbral+1 || umbral-1))
 //{
  if(dist2>umbral) phi=pi/5;
  else phi=-pi/5;
  //n=tan(phi)*d*10;
  angulo=atan((dist2-dist)/d*10)*10;
  Serial1.print(int(angulo));
  Serial1.print(" ");
  distancia= calculardistancia();
  dist= (int)distancia;   //Serial1.print(distancia); Serial1.print("  ");
  distancia2= calculardistancia2();
  dist2= (int)distancia2;  //Serial1.print(distancia2);  Serial1.print(" ");
  if(int(angulo)<int(phi*10))/*dist>dist2-n)*/ //De esta manera se busca dist =
dist2+n
  {
   if(flag1=1){stop();flag1=0;}
   girar(0, 110);
   Serial1.print("0  100 ");
  }
  else if (int(angulo)>int(phi*10))/*dist<dist2-n)*/
  {
   if(flag1=0){stop();flag1=1;}
   girar(1, 110);
   Serial1.print("100  0 ");
  }
  else {
     //lptomedio=dist2;//n/sqrt(1+tan(phi)*tan(phi))/*diferencia de distancia
entre sonares perpendiculares a la pared*/*(1/2+dist/n);
     //Serial1.print(lptomedio);
     //Serial1.print(" ");
     if(dist2!=umbral) {avanzar(180); Serial1.print("-180  -180 ");} //avanzar
     //else if (dist2<umbral) {retroceder(180); Serial1.print("180  180 ");}
//retroceder
     else {estado='a'; stop(); delay(1000); umbral=40;}
     }
 //}
  break;

 }
```

```
  time=millis();
  Serial1.print(time);
  Serial1.println();
  delay(50);
 }
```

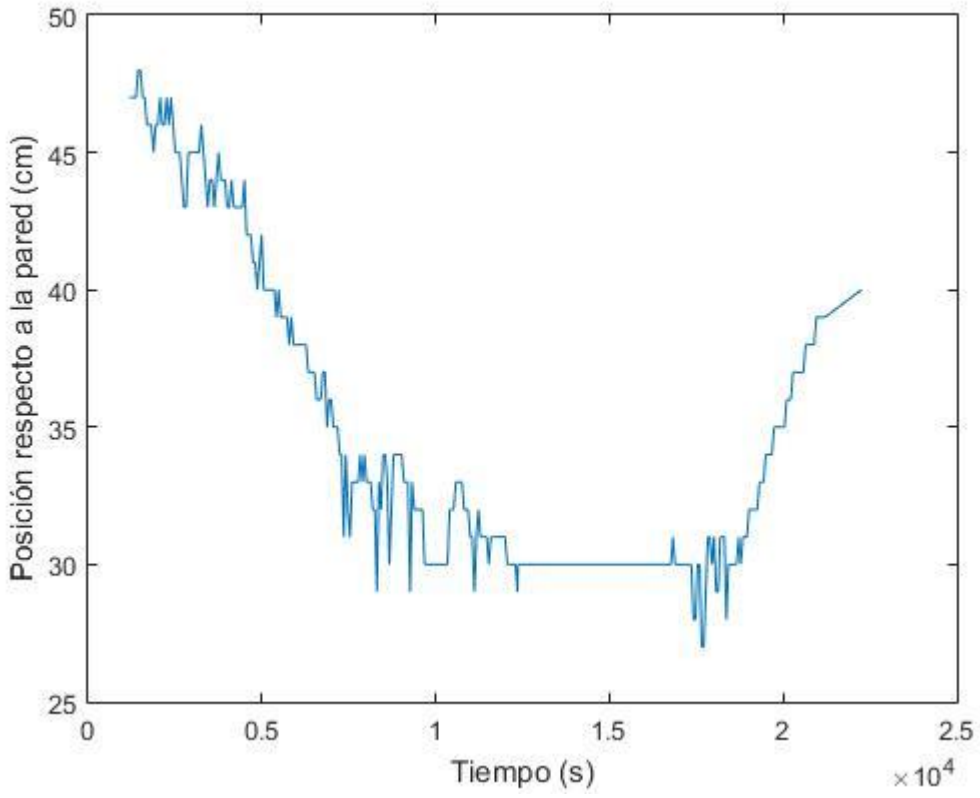
En este caso se compone la función principal de 3 estados, uno alinea al robot y mide la distancia a la que está de la pared. Una vez hecho esto se pasa al estado de 'dato' que compara la posición con la distancia objetivo e indica si se ha cumplido o no (realmente este estado espera un valor por bluetooth, es decir, la variable 'umbral' que es la distancia objetivo pero para la realización del test basta con asignar de antemano umbral=30 (cm) y luego, cuando alcanza dicho objetivo cambia a umbral=40(cm) y compara). Por último, el estado de 'posicionarse' va analizando el ángulo del robot en todo momento (el ángulo se pre-asigna de antemano para evitar complejidades siendo de phi=pi/5= 30º) y va direccionándose a 30º hacia la pared desde el plano vertical paralelo a la pared (si el objetivo está mas cerca de la pared que el robot) o a -30º (si el objetivo está más lejos de la pared que el robot).

Puede visualizar el video de los resultados en el archivo *'practica3_3.mp4'.*

Puede encontrar los resultados de telemetría en el archivo '*Medidas Telemetría 3.3.txt'*

Continua abajo |
\/

Resultados expuestos gráficamente en base a las medidas Telemétricas:



Se puede ver como aproximadamente, según los sensores de ultrasonidos, parte de 50 cm y va pegando pequeños picos de cambios para orientarse y avanzar (únicamente avanza), luego llegados a los 30 cm (que le cuesta trabajo posicionarse correctamente) espera 1 segundo y esta vez se aleja de la pared hasta alcanzar los 40 cm.