Memoria de prácticas de Robótica

Sergio Pinilla
David Rozas

Curso : 05/06

[aquí pract01 de lego]

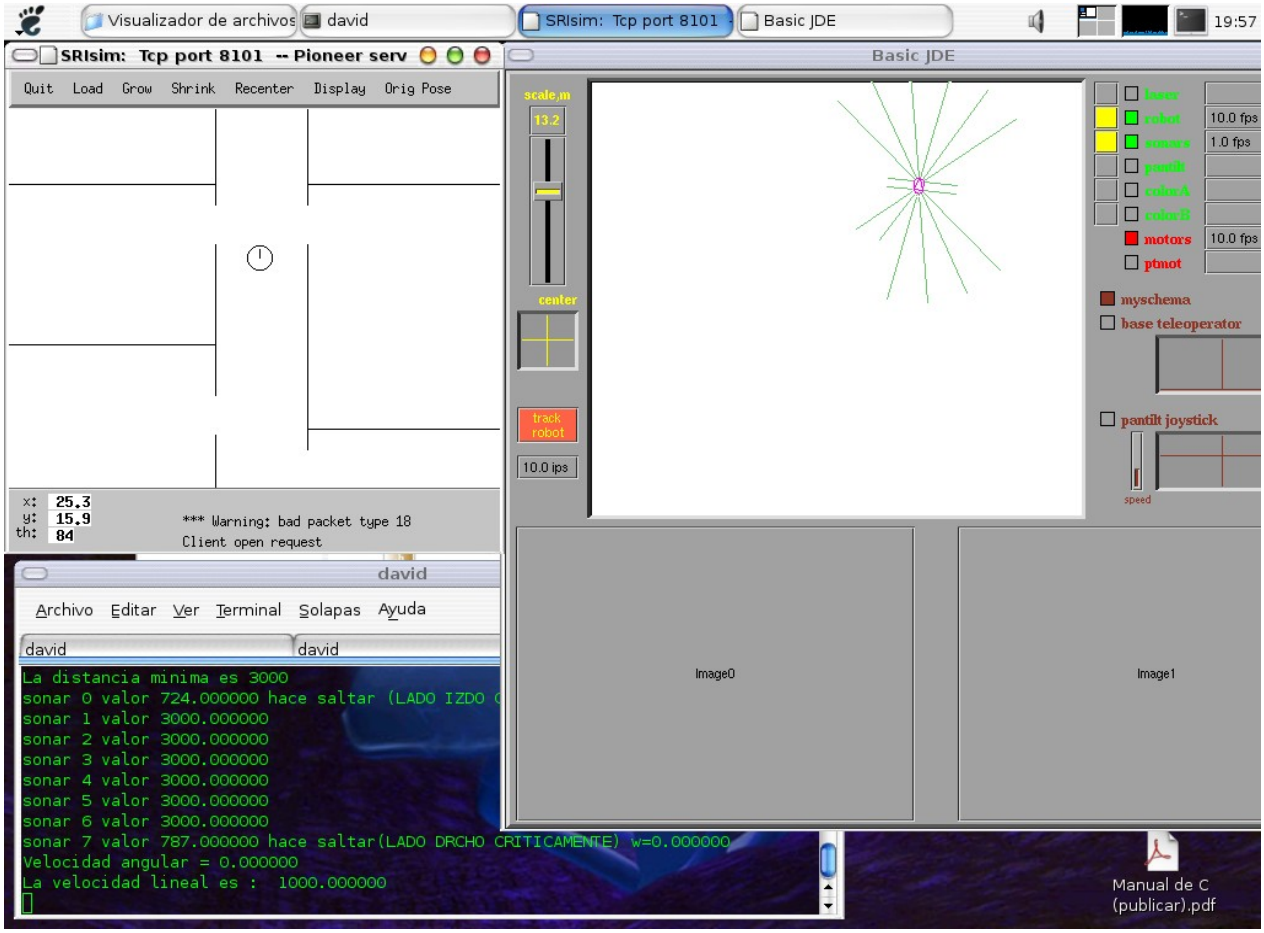
[aquí pract02 de lego]

Práctica 1 de pioneer.

Objetivo : Navegar sin chocarse por distintos mapas (departamental2.wld, office.wld, ...) de forma reactiva utilizando los sónares.

Idea : Calculamos la velocidad angular en función de las lecturas de los sónares "delanteros", a los que asignamos pesos. En cuanto a la velocidad lineal, se modula en función del mínimo valor recogido por los sónares.

Fichero: navegacion_sonares.tar.gz



Código fuente principal (modificaciones sobre myschema.c) :

```
void myschema_iteration()
{
  static int d;

        int i;
        int pesos[8];
        int pesoslin[8];
        int min_distancia;
        int min_sensor;

        /*Sensores izda*/
        pesos[0] = 4;
        pesos[1] = 3;
        pesos[2] = 2;
```

```
	pesos[3] = 1;
	/*Sensores drcha*/
	pesos[4] = 3;
	pesos[5] = 2;
	pesos[6] = 3;
	pesos[7] = 4;

// Pesos para la velocidad lineal
	/*Sensores izda*/
	pesoslin[1] = 1;
	pesoslin[2] = 2;
	pesoslin[3] = 3;
	/*Sensores drcha*/
	pesoslin[4] = 3;
	pesoslin[5] = 2;
	pesoslin[6] = 1;


	v = 250;
	i = 0;


	/*Calculamos la distancia minima*/
	min_distancia = us[1];
	min_sensor = 1;
	for (i=1; i<7; i++)
	{
		if (min_distancia>us[i]){
			min_distancia = us[i];
			min_sensor = i;
			}
	}

	printf("La distancia minima es %i \n", min_distancia);

	/*Modulamos la velocidad en funcion de esta distancia minima */
	/*Ademas si la distancia minima es muy amplia, "centramos"*/
	if(min_distancia<=3000 && min_distancia>=2500)
	{
		v = MAX_VEL/pesoslin[min_sensor];
		w=0;
	}else if(min_distancia<2500 && min_distancia>=2000){
		v = MAX_VEL/pesoslin[min_sensor];
		w=0;
	}else if(min_distancia<2000 && min_distancia>=1500){
		v = 800/pesoslin[min_sensor];
		w=0;
	}else if(min_distancia<1500 && min_distancia>=1000){
		v = 200/pesoslin[min_sensor];
	}else if(min_distancia<1000 && min_distancia>=500){
		v = 200/pesoslin[min_sensor];
	}else{
		v = 100/pesoslin[min_sensor];
	}

	for(i=0;i<=7;i++)
	{

		printf("sonar %i valor %f ",i,us[i]);
		// Miramos si ese sonar esta "pitando"
		// Sonares del lado izquierdo
		if (i<4)
		{
				if (us[i]<1000)
				{
```

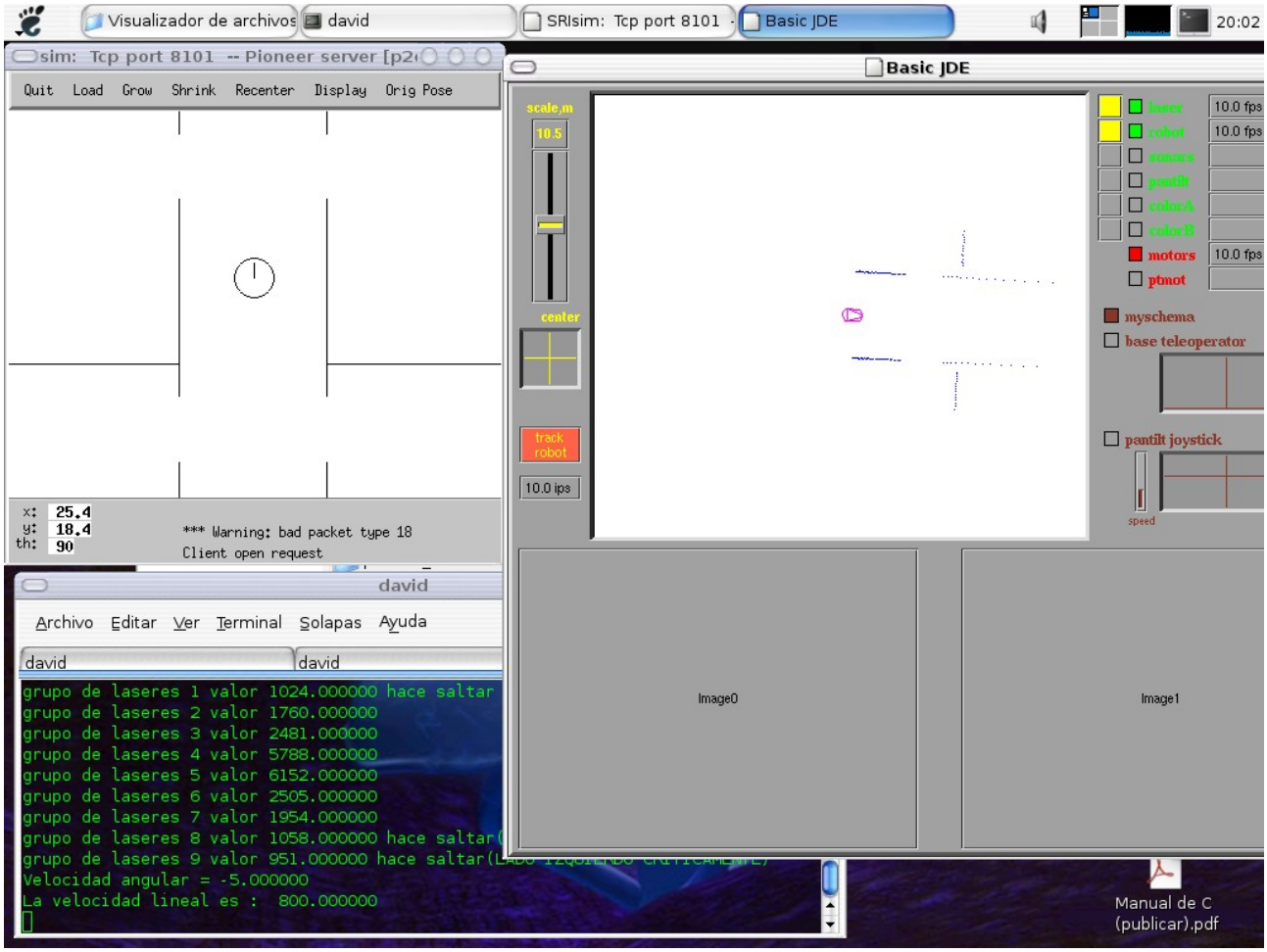
```
                                        if (w>-30)
                                        {
                                                w-=pesos[i]*1;
                                                printf("hace saltar (LADO IZDO CRITICAMENTE) w=%f",w);
                                        }
                                }
                                else if(us[i]<1500)
                                {
                                        if (w>-30)
                                        {
                                                w-=pesos[i]*0.8;
                                                printf("hace saltar (LADO IZDO) w=%f",w);
                                        }
                                }

                        }
                // Miramos si ese sonar esta "pitando"
                // Sonares del lado derecho
                        if ((i>=4) && (i<=7))
                        {

                                if (us[i]<1000)
                                {
                                        /*Controlamos una velocidad angular max, para que no pegue giros
bruscos*/
                                        if(w<30)
                                        {
                                                w+=pesos[i]*1;
                                                printf("hace saltar(LADO DRCHO CRITICAMENTE) w=%f",w);
                                        }
                                }
                                else if(us[i]<1500)
                                {
                                        if(w<30)
                                        {
                                                w+=pesos[i]*0.8;
                                                printf("hace saltar(LADO DRCHO) w=%f",w);
                                        }
                                }
                        }

                        printf("\n");

                }
                printf("Velocidad angular = %f\n",w);
                printf("La velocidad lineal es :  %f \n", v);
}
```

Práctica 2 de pioneer.

Objetivo : Navegar sin chocarse por distintos mapas (departamental2.wld, office.wld, ...) de forma reactiva utilizando los láseres.

Idea : El algoritmo es muy similar al anterior, agrupando los láseres por medias en 10 grupos de 18. Nuevamente se asignan pesos a los grupos para calcular la velocidad angular, y además se asignan pesos "inversos" para el calculo de la velocidad lineal (dando toda la relevancia a los grupos centrales).

Fichero: navegacion_laseres.tar.gz



Código fuente principal (modificaciones sobre myschema.c) :

```
void myschema_iteration()
{
  static int d;

        double laseres_agrupados[10];
        int i,j,total;
        float pesos[10];
        float pesoslin[8];
        int min_distancia;
        int min_sensor;

        /*Cargamos con sus medias en 10 grupos de 18 */
        for(i=1, j=0, total=0; i<=180; i++)
```

```
	{
		total+= laser[(i-1)];
		if (i%18==0)
		{
			laseres_agrupados[j] = total/18;
			total = 0;
			j++;
		}
	}

	for(i=0; i<10; i++)
	{
		printf("Valor de grupo %i : %f \n", i, laseres_agrupados[i]);
	}

/* Asignación de pesos para velocidad angular*/
	/*Sensores drcha*/
	pesos[0] = 3;
	pesos[1] = 2.5;
	pesos[2] = 2;
	pesos[3] = 1.5;
	pesos[4] = 1;
	/*Sensores izda*/
	pesos[9] = 3;
	pesos[8] = 2.5;
	pesos[7] = 2;
	pesos[6] = 1.5;
	pesos[5] = 1;
/* Pesos para la velocidad lineal*/
	/*Sensores drcha*/
	pesoslin[0] = 0;
	pesoslin[1] = 0;
	pesoslin[2] = 1;
	pesoslin[3] = 2;
	pesoslin[4] = 3;
	/*Sensores izda*/
	pesoslin[5] = 3;
	pesoslin[6] = 2;
	pesoslin[7] = 1;
	pesoslin[8] = 0;
	pesoslin[9] = 0;

/*	v = 250;*/

	/*Calculamos la distancia minima de los subgrupos "más centrales"*/
	min_distancia = laseres_agrupados[2];
	min_sensor = 2;
	for (i=2; i<=7; i++)
	{
		if (min_distancia>laseres_agrupados[i])
		{
			min_distancia = laseres_agrupados[i];
			min_sensor = i;
		}
	}

	printf("La distancia minima es %i \n", min_distancia);



	/*Modulamos la velocidad en funcion de esta distancia minima */
	/*Ademas si la distancia minima es muy amplia, "centramos"*/
	if(min_distancia>=2500)
	{
		v = MAX_VEL/pesoslin[min_sensor];
```

```
        w=0;
}else if(min_distancia<2500 && min_distancia>=2000){
        v = MAX_VEL/pesoslin[min_sensor];
        w=0;
}else if(min_distancia<2000 && min_distancia>=1500){
        v = 800/pesoslin[min_sensor];
        w=0;
}else if(min_distancia<1500 && min_distancia>=1000){
        v = 400/pesoslin[min_sensor];
        w=0;
}else if(min_distancia<1000 && min_distancia>=500){
        v = 200/pesoslin[min_sensor];
}else if (min_distancia>500 && min_distancia>=100){
        v = 100/pesoslin[min_sensor];
}else{
        printf("frenamos!\n");
        v = 0;
}


/* Modificamos la velocidad angular en función del grupo
        afectado */
for(i=0;i<=9;i++)
{

        printf("grupo de laseres %i valor %f ",i,laseres_agrupados[i]);

        /*Grupo de sonares del lado derecho */
        if (i<=4)
        {
                if (laseres_agrupados[i]<1000)
                {
                        if (w<20)
                        {
                                w+=pesos[i]*1;
                                printf("hace saltar (LADO DERECHO CRITICAMENTE)");
                        }
                }
                else if(laseres_agrupados[i]<1500)
                {
                        if (w<20)
                        {
                                w+=pesos[i]*0.8;
                                printf("hace saltar (LADO DERECHO)");
                        }
                }
        }

        /* Sonares del lado izquierdo*/
        if ((i>4) && (i<=9))
        {

                if (laseres_agrupados[i]<1000)
                {
                        /*Controlamos una velocidad angular max, para que no pegue giros bruscos*/
                        if(w>-20)
                        {
                                w-=pesos[i]*1;
                                printf("hace saltar(LADO IZQUIERDO CRITICAMENTE)");
                        }
                }
                else if(laseres_agrupados[i]<1500)
                {
                        if(w>-20)
                        {
```

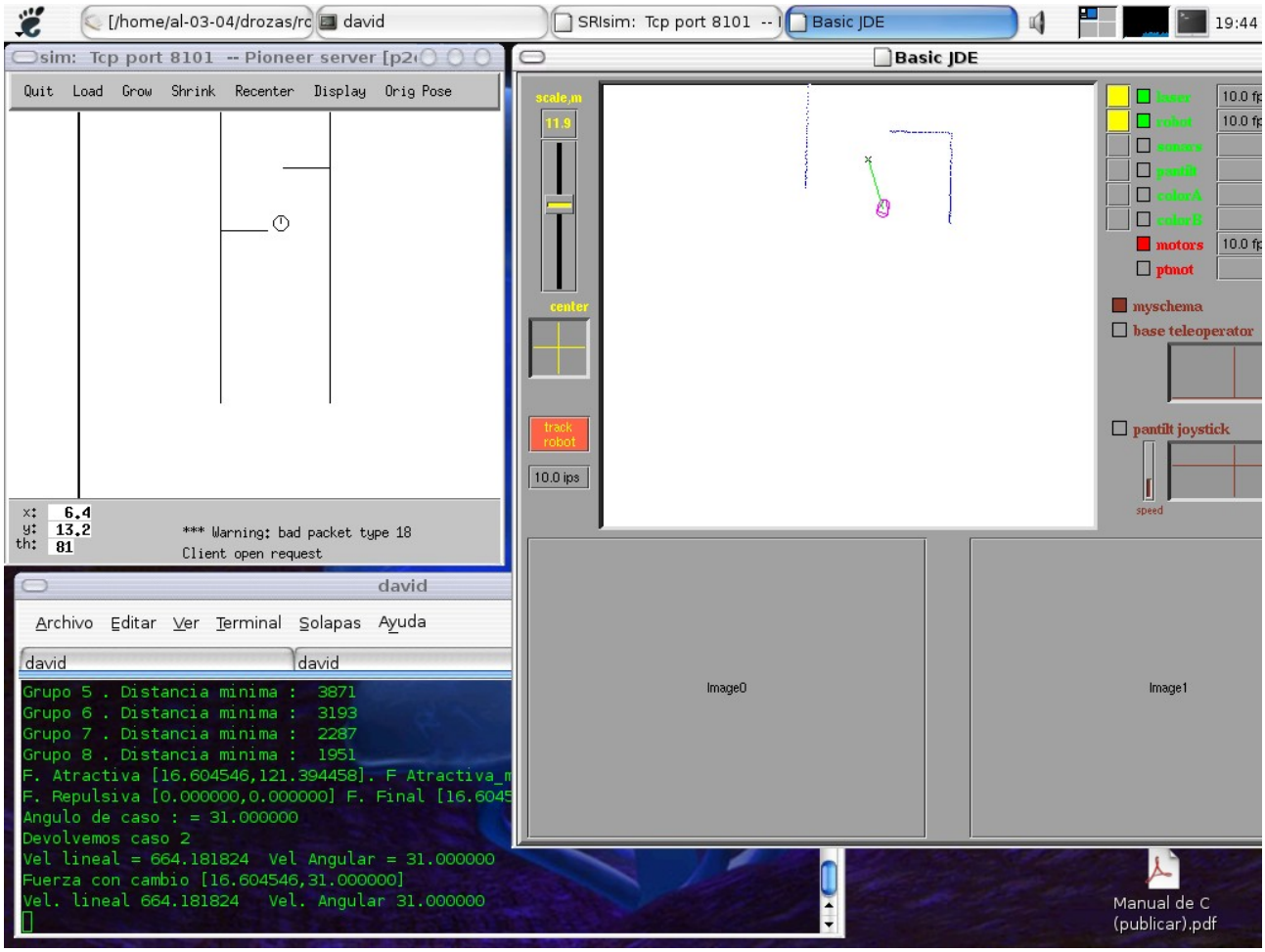
```
                                        w-=pesos[i]*0.8;
                                        printf("hace saltar(LADO IZQUIERDO)");
                                }
                        }

                }

                printf("\n");
        }

        printf("Velocidad angular = %f\n",w);
        printf("La velocidad lineal es :  %f \n", v);

}
```

Práctica 3 de pioneer.

Objetivo : Navegar reactivamente implementando VFF. El robot tiene ser capaz de navegar por escenarios y superar un slalom.

Idea : El algoritmo consiste en calcular la velocidad angular y lineal del robot a partir de la suma de las fuerzas repulsivas y atractivas. Las fuerzas repulsivas se calculan a partir de las lecturas de los láseres del robot (agrupados en 9 grupos, por mínimos); mientras que la fuerza atractiva se calcula a partir de la posición en que se clickea con el ratón en JDE.

Fichero: navegacion_vff.tar.gz



Código fuente principal (modificaciones sobre myschema.c) :
[Ademas se realizaron cambios en jde.h, guiforms.c, guixforms.h, etc para poder pintar, y se creó slalom.wld]

```
/*Agrupa los sensores en 9 grupos de 20 laseres cada uno.
  Toma el valor mínimo de cada grupo*/
void agrupa_sensores(int sensores[])
{
        int i,j,total,min_actual;

        /*Cargamos con el valor minimo, en lugar de la media */
        min_actual = 7600;
```

```
	for(i=1, j=0, total=0; i<=180; i++)
	{
		if(min_actual>laser[(i-1)])
			min_actual = laser[(i-1)];

		if (i%20==0)
		{
			sensores[j] = min_actual;
			min_actual = 7600;
			j++;
		}
	}

}


/*Otorga un peso a cada grupo de laseres, para
  regular el factor que controla la velocidad lineal*/
double obtener_peso_grupo(int grupo)
{
	double peso;

			/*Orientación
				0-> + a drcha
				8-> + a izda*/

			/*Pesos :
				+ fuerza a los centrales*/

		switch (grupo) {
			case 0:
				peso = 0.4;
				break;
			case 1:
				peso = 0.5;
				break;
			case 2:
				peso = 0.6;
				break;
			case 3:
				peso = 0.8;
				break;
			case 4:
				peso = 1.0;
				break;
			case 5:
				peso = 0.8;
				break;
			case 6:
				peso = 0.6;
				break;
			case 7:
				peso = 0.5;
				break;
			case 8:
				peso = 0.4;
		}

		return peso;
}


/*Calcula el vector de fuerza repulsiva de un grupo de sensores*/
void calcula_fuerza_repulsiva(int sensores[],int grupo,double f_repulsiva_aux[])
{
```

```
/* Modulo = Inversamente proporcional a la distancia*/
/* Angulo = 90 + angulo que abarca*/
	double peso;

	/*Se añade funcion que otorga pesos simetricos al grupo */
	peso = obtener_peso_grupo(grupo);

	f_repulsiva_aux[0] = (peso * 7600) / sensores[grupo];

	f_repulsiva_aux[1] = (20 * grupo) + 190;

	/*Valores optimos, de 3..5 (Anula ese vector)*/
	if (f_repulsiva_aux[0]<5)
		f_repulsiva_aux[0]=0;
}

/*Suma dos vectores, pasandolos previamente a coordenadas.
  Devuelve el vector suma resultante.*/
void	suma_fuerzas(double fuerza_pri[],double fuerza_seg[],double fuerza_sum[])
{
	/* Componente de la fuerza en base ortonormal*/
	/* x = Modulo * cos(angulo)			*/
	/* y = Modulo * seno(angulo)	*/
	double f_ortonormal[2];
	double x1; /* X de Fuerza 1		*/
	double y1; /* y de Fuerza 1		*/
	double x2; /* X de Fuerza 2		*/
	double y2; /* Y de Fuerza 2		*/
	double alfa1; /* Angulo en radianes de Fuerza 1	*/
	double alfa2; /* Angulo en radianes de Fuerza 2	*/
	double alfa3; /* Angulo en radianes de Fuerza suma */

	/*Pasamos los vectores a coordenadas*/
	alfa1 = (fuerza_pri[1] * M_PI) / 180;
	x1 = fuerza_pri[0] * cos(alfa1);
	y1 = fuerza_pri[0] * sin(alfa1);

	alfa2 = (fuerza_seg[1] * M_PI) / 180;
	x2 = fuerza_seg[0] * cos(alfa2);
	y2 = fuerza_seg[0] * sin(alfa2);

	/*Obtenemos el vector suma en coordenadas*/
	f_ortonormal[0] = x1 + x2;
	f_ortonormal[1] = y1 + y2;

	/* De dichas coordenadas, obtenemos el módulo del vector suma por pitágoras a²=x²+y² */
	fuerza_sum[0] = sqrt( (f_ortonormal[0] * f_ortonormal[0]) + (f_ortonormal[1] * f_ortonormal[1]));
	/* Y el ángulo del vector suma, mediante la función arcotangente*/
	alfa3 = atan2(f_ortonormal[1],f_ortonormal[0]);
	/*Y lo pasamos de radianes a grados*/
	fuerza_sum[1] = (180 * alfa3)/M_PI;

	/*Si es menor que 0, hay que sumarle 360º*/
	if (fuerza_sum[1]<0)
		fuerza_sum[1] = 360 + fuerza_sum[1];
}


/*Devuelve el vector suma repulsivo total, a partir de las medidas de cada grupo de láseres*/
void	calcula_fuerza_repulsiva_total(int sensores[])
{
	int e;
	int grupo;

	double f_repulsiva_aux[2];
```

```
	double f_repulsiva_total[2],f_repulsiva_sumatoria[2];

	/* inicializamos la fuerza total */
	f_repulsiva_aux[0]=0;
	f_repulsiva_aux[1]=0;
	f_repulsiva_total[0] = 0;
	f_repulsiva_total[1] = 0;

	for (e=0;e<=8;e++)
	{
		grupo = e;
		calcula_fuerza_repulsiva(sensores,grupo,f_repulsiva_aux);
		suma_fuerzas(f_repulsiva_total,f_repulsiva_aux, f_repulsiva_sumatoria);
		f_repulsiva_total[0] = f_repulsiva_sumatoria[0];
		f_repulsiva_total[1] = f_repulsiva_sumatoria[1];

	}
	f_repulsiva[0] = f_repulsiva_total[0];
	f_repulsiva[1] = f_repulsiva_total[1];

}

/*Cambia los angulos a la forma en que los entiende el robot*/
double cambia_angular(int f_inicial)
{
	double angulo;
	double f_final;

	/* Trasladamos los ejes 90º*/
	angulo = f_inicial - 90;

	/*Si el angulo es negativo, hay q sumarle 360*/
	if (angulo<0)
		angulo = angulo + 360;

	/*Si el angulo resultante, esta en la izda, permanece igual.
	 Si está en la derecha, le restamos 360*/

	if (angulo<=180)
		f_final = angulo;
	else
		f_final = angulo - 360;

/*	printf("Angulo con cambio [m.robot] = %f\n",f_final);*/

	return f_final;
}

/*Analiza la situación en la que se encuentra el robot
  respecto a su orientación*/
int dame_caso (double angulo)
{
	int caso;

/*Valores optimos de primer tramo...entre 8..15
	Valores optimos de segundo tramo max(t1)..20-35*/

	printf("Angulo de caso : = %f \n", angulo);

	/*Paramos v y w si hemos llegado*/
	if((f_final[0]<1) || (f_atractiva[0]<5)){
		caso = 0;
	}else if ( f_repulsiva[0] > 25){
		/*Si la fuerza repulsiva es muy grande, paramos solo la v_lineal, pero seguimos girando*/
		caso = 1;
```

```
		printf("caso 1 por f.repulsiva>25\n");
	}else{
		if ((angulo<10) && (angulo>-10))
		{
			caso = 3;
		}else if( (angulo>10 && angulo<=135) || (angulo <-10 && angulo>-135) ){
			caso = 2;
		}else{
			caso = 1;
		printf("caso 1 por angulo\n");
		}
	}

	printf("Devolvemos caso %i \n", caso);

	return caso;

}

/*Realiza el movimiento del robot*/
void mueve_robot()
{
	double fangular = 0.0;
	int factor_lineal = 300;
	int factor_angular = 1;
	int caso = 0;

	fangular = cambia_angular(f_final[1]);


			caso = dame_caso(fangular);

			switch (caso) {
				case 3:
					factor_lineal = 65;
					break;

				case 2:
					factor_lineal = 40;
					break;

				case 1:
					factor_lineal = 0;
					break;

				default:
					factor_lineal = 0;
					factor_angular = 0;

			}

	v = f_final[0] * factor_lineal;

	if (abs(fangular)<90){
		w = fangular * factor_angular;
	}else {
		if (fangular<0)
			w=-90;
		else
			w=90;
	}
	printf("Vel lineal = %f  Vel Angular = %f\n",v,w);


	printf("Fuerza con cambio [%f,%f]\n",f_final[0],fangular);
```

```
	printf("Vel. lineal %f   Vel. Angular %f\n",v,w);

}

/*Devuelve el vector de fuerza atractiva total*/
void calcula_fuerza_atractiva_total()
{
	double alfa,alfa_gr;
	double modulo;

	/* Actualizamos el valor de las x y de la fuerza atractiva con respecto al robot para que lo pinte */
	x_fatrac = mouse_x - robot[0];
	y_fatrac = mouse_y - robot[1];


	/* Modulo por pitagoras a²=x²+y² */
	modulo = sqrt( (x_fatrac * x_fatrac) + (y_fatrac * y_fatrac));
	modulo = modulo / 76;
	f_atractiva[0] = modulo;

	/* Angulo primero averiguamos en angulo absoluto con cambio de ejes x->y */
	alfa = atan2(y_fatrac,x_fatrac);

	/* Angulo será alfa - lo que se ha movido el robot*/
	alfa = alfa - robot[2];
	alfa = alfa + (M_PI/2);

	alfa_gr = (180 * alfa)/M_PI;

	/* Cambiamos a grados */
	if (alfa_gr<0)
		alfa_gr = 180 + (180 + alfa_gr);
	f_atractiva[1] = alfa_gr;
}


void myschema_iteration()
{
	/***********************************************
	 *		PROGRAMA PRINCIPAL                     *
	***********************************************/
	int sensores[9];
	int i;
	double f_atractiva_modulada[2];

/* Inicializamos fuerzas */
	f_repulsiva[0] = 0;
	f_repulsiva[1] = 0;

	f_final[0] = 0;
	f_final[1] = 0;


	agrupa_sensores(sensores);

	for (i=0;i<=8;i++)
		printf("Grupo %i . Distancia minima :  %i\n",i,sensores[i]);

	calcula_fuerza_repulsiva_total(sensores);

	calcula_fuerza_atractiva_total();

	f_atractiva_modulada[0] = f_atractiva[0];
	f_atractiva_modulada[1] = f_atractiva[1];
```

```
		if(f_atractiva_modulada[0]>20)
			f_atractiva_modulada[0] = 20;


		printf("F. Atractiva [%f,%f]. F Atractiva_modulada [%f,%f] F. Repulsiva [%f,%f]
",f_atractiva[0],f_atractiva[1], f_atractiva_modulada[0], f_atractiva_modulada[1] ,f_repulsiva[0],f_repulsiva[1]);


		suma_fuerzas(f_atractiva_modulada,f_repulsiva,f_final);
		printf("F. Final [%f,%f]\n",f_final[0],f_final[1]);

		mueve_robot();
}
```

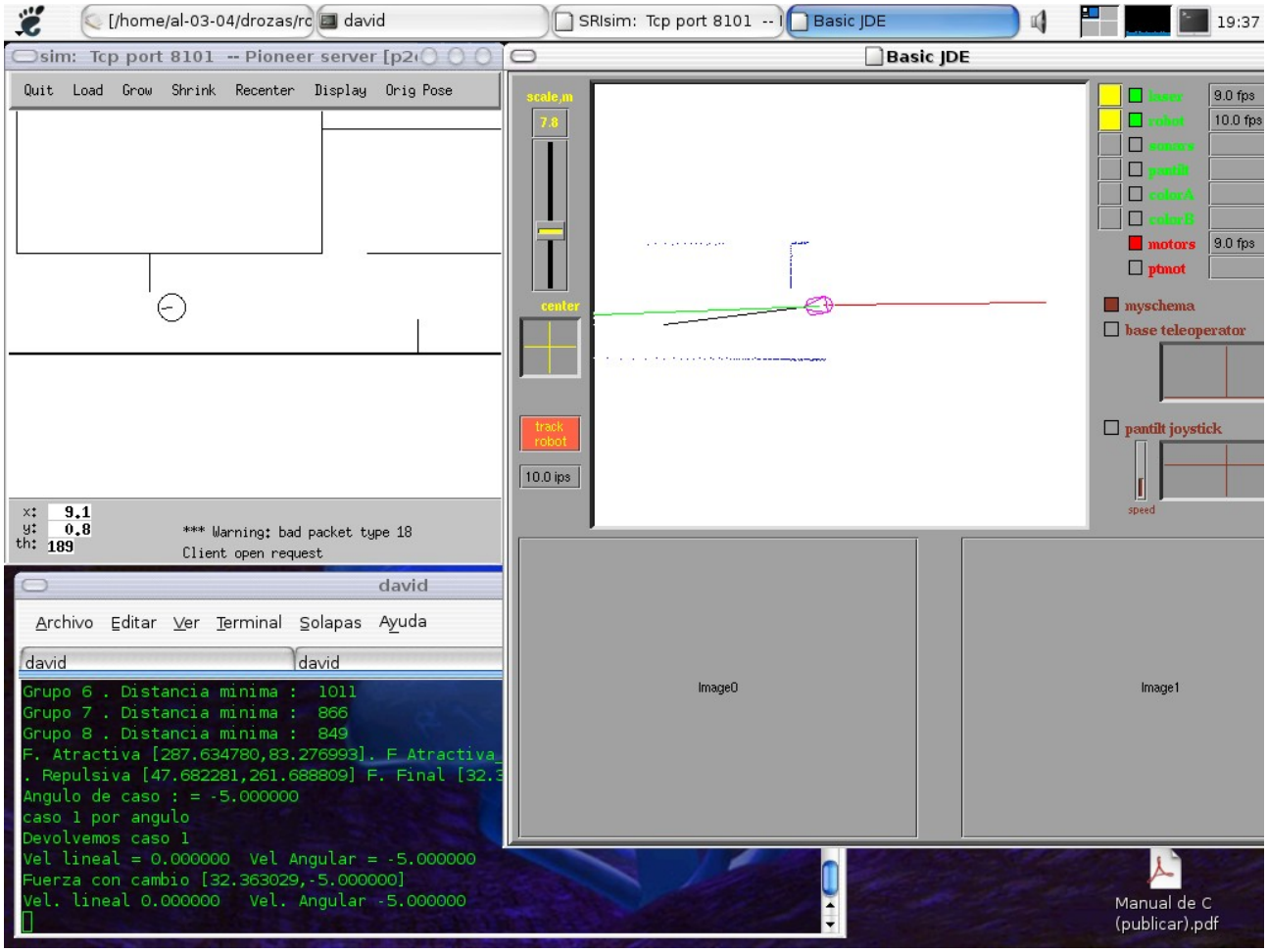
[aquí pract04 de pioneer]

Práctica 5 de pioneer.

Objetivo : Navegación híbrida, combinando deliberativa con VFF. El robot debe ser capaz de realizar un recorrido por el departamental2, y superar obstáculos con los que no contaba.

Idea : El algoritmo consiste en una modificación del VFF, en la que la fuerza atractiva la van marcando los distintos puntos de la ruta. Los obstáculos imprevistos se superan gracias a las fuerzas repulsivas, a las que se ha dado un especial peso a los sensores de los grupos "diagonales".

Fichero: navegacion_hibrida.tar.gz



Código fuente principal (modificaciones sobre myschema.c) :
[Ademas se realizaron cambios en jde.h, guiforms.c, guixforms.h, etc para poder pintar, y se creó departamental2_obstaculos.wld]

```
int inicializado = 0;
int pto_actual = 0; /*Esta sera global! con el pto actual */


/*Inicializa la lista de puntos que marca la ruta*/
void inicializar_lista()
{
```

```
	/*Ruta por los departamentales*/
	lista[0].x = 25400;
	lista[0].y = 12750;

	lista[1].x = 25200;
	lista[1].y = 700;

	lista[2].x = 5500;
	lista[2].y = 700;

	lista[3].x = 5500;
	lista[3].y = 11900;

	lista[4].x = 1300;
	lista[4].y = 11700;
}

/* Guarda en las coordenadas globales el siguiente punto. Devuelve 0
   si todavía quedan puntos, uno en caso de que no*/
int dame_puntos()
{
	double distancia = 0.0;
  //Si me quedan puntos.
	printf("Pto_actual = %i\n",pto_actual);

	if (pto_actual<N_PUNTOS)
	{
	/*Nos da los puntos de destino y origen */
			distancia = sqrt( ((destino.x-robot[0]) * (destino.x-robot[0])) + ((destino.y-
robot[1])*(destino.y-robot[1])));
		if (distancia<600)
		 {
				printf("******* Cambio de punto *******\n");
				pto_actual++;
				destino = lista[pto_actual];
			}
	return 0; //Devolvemos 0 si aun nos quedan ptos que calcular
	}else{
	 return 1; //Si ya no hay puntos, devolvemos un 1 al p.ppal.
	}
}


/*Agrupa los sensores en 9 grupos de 20 laseres cada uno.
  Toma el valor mínimo de cada grupo*/
void agrupa_sensores(int sensores[])
{
	int i,j,total,min_actual;

	/*Cargamos con el valor minimo, en lugar de la media */
	min_actual = 7600;
	for(i=1, j=0, total=0; i<=180; i++)
	{
		if(min_actual>laser[(i-1)])
			min_actual = laser[(i-1)];

		if (i%20==0)
		{
			sensores[j] = min_actual;
			min_actual = 7600;
			j++;
		}
	}
```

```
}


/*Otorga un peso a cada grupo de laseres, para
  regular el factor que controla la velocidad lineal*/
double obtener_peso_grupo(int grupo)
{
	double peso;

			/*Orientación
				0-> + a drcha
				8-> + a izda*/

	switch (grupo) {
			case 0:
				peso = 0.2;
				break;
			case 1:
				peso = 3.0;
				break;
			case 2:
				peso = 4.0;
				break;
			case 3:
				peso = 5.0;
				break;
			case 4:
				peso = 1.9;
				break;
			case 5:
				peso = 5.0;
				break;
			case 6:
				peso = 4.0;
				break;
			case 7:
				peso = 3.0;
				break;
			case 8:
				peso = 0.2;
		}

		return peso;
}


/*Calcula el vector de fuerza repulsiva de un grupo de sensores*/
void calcula_fuerza_repulsiva(int sensores[],int grupo,double f_repulsiva_aux[])
{
/* Modulo = Inversamente proporcional a la distancia*/
/* Angulo = 90 + angulo que abarca*/
	double peso;

	/*Se añade funcion que otorga pesos simetricos al grupo */
	peso = obtener_peso_grupo(grupo);

	f_repulsiva_aux[0] = (peso * 7600) / sensores[grupo];

	f_repulsiva_aux[1] = (20 * grupo) + 190;
}

/*Suma dos vectores, pasandolos previamente a coordenadas.
  Devuelve el vector suma resultante.*/
void	suma_fuerzas(double fuerza_pri[],double fuerza_seg[],double fuerza_sum[])
{
```

```
	/* Componente de la fuerza en base ortonormal*/
	/* x = Modulo * cos(angulo)                    */
	/* y = Modulo * seno(angulo)        */
	double f_ortonormal[2];
	double x1; /* X de Fuerza 1          */
	double y1; /* y de Fuerza 1          */
	double x2; /* X de Fuerza 2          */
	double y2; /* Y de Fuerza 2          */
	double alfa1; /* Angulo en radianes de Fuerza 1       */
	double alfa2; /* Angulo en radianes de Fuerza 2       */
	double alfa3; /* Angulo en radianes de Fuerza suma */

	/*Pasamos los vectores a coordenadas*/
	alfa1 = (fuerza_pri[1] * M_PI) / 180;
	x1 = fuerza_pri[0] * cos(alfa1);
	y1 = fuerza_pri[0] * sin(alfa1);

	alfa2 = (fuerza_seg[1] * M_PI) / 180;
	x2 = fuerza_seg[0] * cos(alfa2);
	y2 = fuerza_seg[0] * sin(alfa2);

	/*Obtenemos el vector suma en coordenadas*/
	f_ortonormal[0] = x1 + x2;
	f_ortonormal[1] = y1 + y2;

	/* De dichas coordenadas, obtenemos el módulo del vector suma por pitágoras a²=x²+y² */
	fuerza_sum[0] = sqrt( (f_ortonormal[0] * f_ortonormal[0]) + (f_ortonormal[1] * f_ortonormal[1]));
	/* Y el ángulo del vector suma, mediante la función arcotangente*/
	alfa3 = atan2(f_ortonormal[1],f_ortonormal[0]);
	/*Y lo pasamos de radianes a grados*/
	fuerza_sum[1] = (180 * alfa3)/M_PI;

	/*Si es menor que 0, hay que sumarle 360º*/
	if (fuerza_sum[1]<0)
		fuerza_sum[1] = 360 + fuerza_sum[1];
}


/*Devuelve el vector suma repulsivo total, a partir de las medidas de cada grupo de láseres*/
void	calcula_fuerza_repulsiva_total(int sensores[])
{
	int e;
	int grupo;

	double f_repulsiva_aux[2];
	double f_repulsiva_total[2],f_repulsiva_sumatoria[2];

	/* inicializamos la fuerza total */
	f_repulsiva_aux[0]=0;
	f_repulsiva_aux[1]=0;
	f_repulsiva_total[0] = 0;
	f_repulsiva_total[1] = 0;

	for (e=0;e<=8;e++)
	{
		grupo = e;
		calcula_fuerza_repulsiva(sensores,grupo,f_repulsiva_aux);
		suma_fuerzas(f_repulsiva_total,f_repulsiva_aux, f_repulsiva_sumatoria);
		f_repulsiva_total[0] = f_repulsiva_sumatoria[0];
		f_repulsiva_total[1] = f_repulsiva_sumatoria[1];

	}
	f_repulsiva[0] = f_repulsiva_total[0]/3.0;
	f_repulsiva[1] = f_repulsiva_total[1];
```

```
}

/*Cambia los angulos a la forma en que los entiende el robot*/
double cambia_angular(int f_inicial)
{
	double angulo;
	double f_final;

	/* Trasladamos los ejes 90º*/
	angulo = f_inicial - 90;

	/*Si el angulo es negativo, hay q sumarle 360*/
	if (angulo<0)
		angulo = angulo + 360;

	/*Si el angulo resultante, esta en la izda, permanece igual.
	 Si está en la derecha, le restamos 360*/

	if (angulo<=180)
		f_final = angulo;
	else
		f_final = angulo - 360;


	return f_final;
}

/*Analiza la situación en la que se encuentra el robot
  respecto a su orientación*/
int dame_caso (double angulo)
{
	int caso;

/*Valores optimos de primer tramo...entre 8..15
	Valores optimos de segundo tramo max(t1)..20-35*/

	printf("Angulo de caso : = %f \n", angulo);

	/*Paramos v y w si hemos llegado*/ /*OJITO AHORA AQUI, QUE SI NO SE ACHANTA*/
	if((f_final[0]<1) || (f_atractiva[0]<5)){
		caso = 0;
	}else if ( f_repulsiva[0] > 70){
		/*Si la fuerza repulsiva es muy grande, paramos solo la v_lineal, pero seguimos girando*/
		caso = 1;
		printf("caso 1 por f.repulsiva>70\n");
	}else{
		if ((angulo<5) && (angulo>-5))
		{
			caso = 3;
		}else if( (angulo>5 && angulo<=160) || (angulo <-5 && angulo>-160) ){
			caso = 2;
		}else{
			caso = 1;
		printf("caso 1 por angulo\n");
		}
	}

	printf("Devolvemos caso %i \n", caso);

	return caso;

}

/*Realiza el movimiento del robot*/
void mueve_robot()
```

```
{
	double fangular = 0.0;
	int factor_lineal = 10;
	int factor_angular = 1;
	int caso = 0;

	fangular = cambia_angular(f_final[1]);


			caso = dame_caso(fangular);

			switch (caso) {
				case 3:
					factor_lineal = 13;
					break;

				case 2:
					factor_lineal = 2.5;
					factor_angular = 1.0;
					break;

				case 1:
					factor_lineal = 0;
					factor_angular =1.0;
			}

	v = f_final[0] * factor_lineal;

	if (abs(fangular)<90){
		w = fangular * factor_angular;
	}else {
		if (fangular<0)
			w=-90;
		else
			w=90;
	}

	printf("Vel lineal = %f  Vel Angular = %f\n",v,w);
	printf("Fuerza con cambio [%f,%f]\n",f_final[0],fangular);
	printf("Vel. lineal %f   Vel. Angular %f\n",v,w);

}

/*Devuelve el vector de fuerza atractiva total*/
void calcula_fuerza_atractiva_total()
{
	double alfa,alfa_gr;
	double modulo;

	/* Actualizamos el valor de las x y de la fuerza atractiva con respecto al robot para que lo pinte */
	x_fatrac = destino.x - robot[0];
	y_fatrac = destino.y - robot[1];


	/* Modulo por pitagoras a²=x²+y² */
	modulo = sqrt( (x_fatrac * x_fatrac) + (y_fatrac * y_fatrac));
	modulo = (modulo / 76) * 6.0;
	f_atractiva[0] = modulo;

	/* Angulo primero averiguamos en angulo absoluto con cambio de ejes x->y */
	alfa = atan2(y_fatrac,x_fatrac);

	/* Angulo será alfa - lo que se ha movido el robot*/
	alfa = alfa - robot[2];
	alfa = alfa + (M_PI/2);
```

```
	alfa_gr = (180 * alfa)/M_PI;

	/* Cambiamos a grados */
	if (alfa_gr<0)
		alfa_gr = 180 + (180 + alfa_gr);
	f_atractiva[1] = alfa_gr;
}


void myschema_iteration()
{
		/**********************************/
		/*	PROGRAMA PRINCIPAL	*/
		/**********************************/
	int sensores[9];
	int i;
	int fin;
	double distancia = 0.0;
	double f_atractiva_modulada[2];

	if (inicializado == 0)
	{
		printf("*******INICIALIZAMOS LAS VARIABLES*********");
		/* Inicializo */
		inicializar_lista();
		destino = lista[0];
		inicializado = 1;
	}

		/* Mientras haya puntos */
		fin = dame_puntos();
		if (fin == 0)
		{
			/*Calculamos la distancia actual respecto a ese punto*/
			distancia = sqrt( ((destino.x-robot[0]) * (destino.x-robot[0])) + ((destino.y-
robot[1])*(destino.y-robot[1])));
			printf("Vamos hacia el punto %f %f \n", destino.x, destino.y);
			printf("Estamos a %f del punto destino \n", distancia);

			/*Si es mayor que cien, seguimos nuestro camino. Si no, consideramos que ya hemos llegado
y pediremos el sig. pto.*/
			if(distancia>600)
			{

				/* Inicializamos fuerzas */
				f_repulsiva[0] = 0;
				f_repulsiva[1] = 0;

				f_final[0] = 0;
				f_final[1] = 0;

				/*Calculamos el vector total de fuerza repulsiva*/
				agrupa_sensores(sensores);
				for (i=0;i<=8;i++)
					printf("Grupo %i . Distancia minima :  %i\n",i,sensores[i]);

				calcula_fuerza_repulsiva_total(sensores);

				/*Calculamos el vector total de fuerza atractiva, para ese par de puntos (globales)*/
				calcula_fuerza_atractiva_total();

				f_atractiva_modulada[0] = f_atractiva[0];
				f_atractiva_modulada[1] = f_atractiva[1];
```

```
                                        /*Hemos aumentado el modulo internamente, pero ahora limitamos a un máximo de
80. Asi conseguimos
                                        que los valores altos no sean tan altos, y a la vez que los valores minimos nos den
una f.repulsiva
                                        que pueda compensar a la atractiva*/
                                        if(f_atractiva_modulada[0]>80)
                                                f_atractiva_modulada[0] = 80;

                                        printf("F. Atractiva [%f,%f]. F Atractiva_modulada [%f,%f] F. Repulsiva [%f,%f]
",f_atractiva[0],f_atractiva[1],                                        f_atractiva_modulada[0],
f_atractiva_modulada[1] ,f_repulsiva[0],f_repulsiva[1]);


                                        suma_fuerzas(f_atractiva_modulada,f_repulsiva,f_final);
                                        printf("F. Final [%f,%f]\n",f_final[0],f_final[1]);

                                        mueve_robot();
                                }else{

                                        printf("Ya hemos llegado a %f,%f . A por el siguiente punto!\n",
destino.x,destino.y);
                                        v=0;
                                }

                        }
                        else
                        {
                                printf("**************** Hemos llegado al final del recorrido *************** \n");
                                v=0;
                                w=0;
                                exit(0);
                        }

}
```