**PROGRAM 1:-**

/\* A program to implement the basic maneuvers of the robot\*/

#include<stdio.h>

#include<khepera/khepera.h>

staticknet\_dev\_t \* dsPic; // for microcontroller access

voidmove\_front();

voidrotate\_robot();

int main(intargc, char \*argv[])// argc and argv[] are two input parameters of main function, argc will store the no. of arguments passed while execution and argv[] will store these arguments as a character array

{

if ( kh4\_init(argc ,argv)!=0)

{

printf("\nERROR: could not initiate the libkhepera!\n\n");

return -1;

}

dsPic = knet\_open( "Khepera4:dsPic" , KNET\_BUS\_I2C , 0 , NULL );//open robot socket

if ( dsPic==NULL)

{

printf("\nERROR: could not initiate communication with Kh4 dsPic\n\n");

return -2;

}

move\_front();// moving the robot forward by a given distance

rotate\_robot();// rotating the robot by 90 degrees in clockwise direction

}

voidmove\_front()

{

int distance=300;

int speed=50;

int count=(int)((distance\*400)/151);//151mm distance travel for 400 steps

kh4\_SetMode(kh4RegSpeed,dsPic );// set motors to control mode

kh4\_set\_speed(speed,speed,dsPic );// to define the left and right wheel speeds

while(count!=0)

{

kh4\_set\_speed(speed ,speed,dsPic );

usleep(10000); // wait some ms

count=count-1;

}

kb\_change\_term\_mode(0); // revert to previous state

kh4\_set\_speed(0,0,dsPic ); // stop robot

kh4\_SetMode(kh4RegIdle,dsPic );//set motors to idle

}

voidrotate\_robot()

{

int angle=90;

int speed=50;

int count=(int)((angle\*438)/180);//180 degree rotation for 438 steps

kh4\_SetMode(kh4RegSpeed,dsPic );

kh4\_set\_speed(speed ,-speed,dsPic );

while(count !=0)

{

kh4\_set\_speed(speed ,-speed,dsPic );

usleep(10000);

count=count-1;

}

kb\_change\_term\_mode(0);

kh4\_set\_speed(0,0,dsPic );

kh4\_SetMode(kh4RegIdle,dsPic );

}

**Program 2:-**

/\* A program for processing an image\*/

#include<stdio.h>

#include<khepera/khepera.h>

staticknet\_dev\_t \* dsPic; // for microcontroller access

intcamera\_example();

int main(intargc, char \*argv[])// argc and argv[] are two input parameters of main function, argc will store the no. of arguments passed while execution and argv[] will store these arguments as a character array

{

if ( kh4\_init(argc ,argv)!=0)

{

printf("\nERROR: could not initiate the libkhepera!\n\n");

return -1;

}

dsPic = knet\_open( "Khepera4:dsPic" , KNET\_BUS\_I2C , 0 , NULL );//open robot socket

if ( dsPic==NULL)

{

printf("\nERROR: could not initiate communication with Kh4 dsPic\n\n");

return -2;

}

camera\_example();

}

intcamera\_example()

{

int ret;

unsignedintdWidth=192;//192; // image width (max 752)

unsignedintdHeight=144;//144; // image height (max 480)

unsignedchar\* buffer=NULL; // pointer for image buffer

unsignedchar\* output\_buff=NULL; // pointer for another image buffer for sobel filter output

printf("\r\nCapturing the Image and processing it");

kh4\_SetMode(kh4RegSpeed,dsPic );

// camera initialisation

if ((ret=kb\_camera\_init(&dWidth, &dHeight))<0)

{

fprintf(stderr,"camera init error %d\r\n",ret);

free(buffer);

return -1;

}

// allocating memory for image

buffer=malloc(dWidth\*dHeight\*3\*sizeof(char));

if (buffer==NULL)

{

fprintf(stderr,"could alloc image buffer!\r\n");

free(buffer);

return -2;

}

/\*\*\*\* taking image \*\*\*\*\*\*\*/

if ((ret=take\_one\_image(buffer))<0)

{

fprintf(stderr,"take image error %d\r\n",ret);

free(buffer);

kb\_camera\_release();

return -3;

}

// saving image

if ((ret=save\_buffer\_to\_jpg("original.jpg",70,buffer))<0)

{

fprintf(stderr,"save image error %d\r\n",ret);

free(buffer);

kb\_camera\_release();

return -4;

}

/\*\*\*\* transform into greyscale \*\*\*\*/

into\_greyscale(buffer);

// saving image

if ((ret=save\_buffer\_to\_jpg("greyscale.jpg",70,buffer))<0)

{

fprintf(stderr,"save image error %d\r\n",ret);

free(buffer);

kb\_camera\_release();

return -4;

}

/\*\*\*\* apply Sobel filter for detecting edges \*\*\*\*\*/

printf("\nApplyingSobel filter for detecting edges...\n");

// allocate memory for output

output\_buff=malloc(dWidth\*dHeight\*3\*sizeof(char));

#define FILTER\_SIZE 3

intfilterx[FILTER\_SIZE\*FILTER\_SIZE]={-1,0,1,-2,0,2,-1,0,1}; // mask in x direction

intfiltery[FILTER\_SIZE\*FILTER\_SIZE]={1,2,1,0,0,0,-1,-2,-1}; // mask in y direction

apply\_filter(buffer,output\_buff,filterx,filtery,FILTER\_SIZE,FILTER\_SIZE);

// saving image

if ((ret=save\_buffer\_to\_jpg("edges.jpg",70,output\_buff))<0)

{

fprintf(stderr,"save image error %d\r\n",ret);

free(buffer);

kb\_camera\_release();

return -4;

}

// releasing camera

kb\_camera\_release();

// free image memory

free(buffer);

free(output\_buff);

kb\_change\_term\_mode(0); // switch to normal key input mode

kh4\_set\_speed(0,0,dsPic ); // stop robot

kh4\_SetMode(kh4RegIdle,dsPic );

return 0;

}