Implementation & testing of control algorithm for obstacle avoidance & object following on the e-puck2 robot (ACS6501)

Ankur Tiwari, Dehao Zhang

Abstract*— The purpose of this report is to describe the designing, implementing, and testing of a control strategy for the e-puck2 robot to accomplish two tasks i.e., Exploring a constrained environment with obstacles and chasing an object in an open environment. To accomplish these two tasks, inbuilt IR proximity sensors and ToF (time of flight) distance sensors were used for taking logical decisions. Both the tasks were performed successfully with some minor bugs in robots' performance due to inherent uncertainties associated with real robots. Performance of the mentioned undertakings helped in understanding the problems associated with implementation of logic on real robots.

I. Strategies

A. Obstacle Avoidance

There were two main objectives associated with this task , the primary goal was to avoid any collision with surrounding objects and secondary objective was to completely explore the surroundings. For the primary task the key was to sense anything coming in the path of the robot and take a left or right turn to avoid the obstacle. To accomplish the second task successfully it was important that robot did not follow the walls and explore the arena completely. For this to happen it was important that when robot moved in a certain direction for long it should take a turn in the direction which had more open space. The logic associated with these two tasks has been explained with the help of these two diagrams given below:

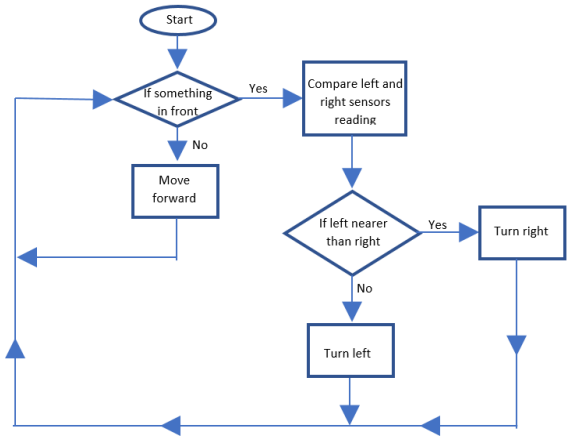

Figure 1 Flowchart for obstacle avoidance

Ankur Tiwari studies at the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, United Kingdom (e-mail: atiwari1@sheffield.ac.uk)

Dehao Zhang studies at the Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, United Kingdom (e-mail: dzhang1@sheffield.ac.uk)

B. Object Following

This task can be broken into two basic components. The first component involves detecting the position of the object when it is not nearby. Once the position is detected the robot needs to move near the object and maintain its position with respect to the object. To accomplish the first task robot checks whether the object is at proximity of ToF distance sensor if it's not Infront of the ToF distance sensor the IR sensors at the periphery are used to detect the object. Once the object is detected robot turns towards the object and then move towards it to stop at a fixed distance. The algorithm to explain the above process has been shown in form a flowchart below.

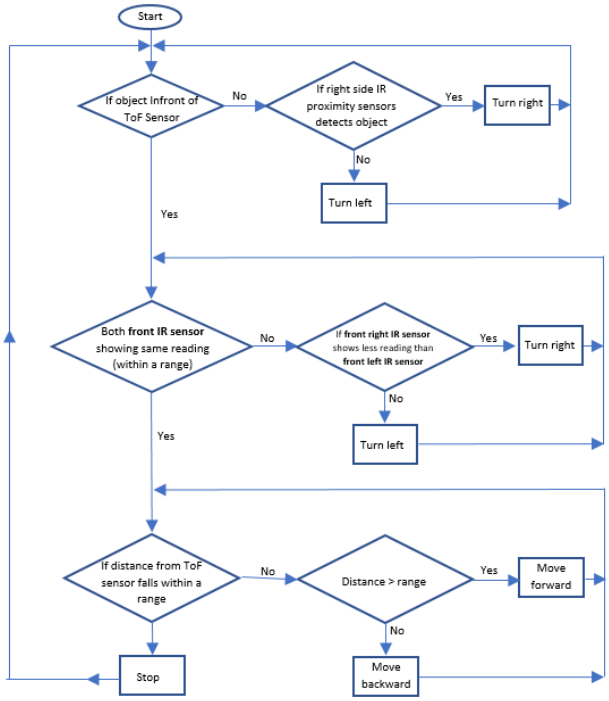

Figure 2 Flowchart for object following

II. Implementation

A. Movement

In both the tasks for moving the robot forward , backward , rotating the robot right or left, functions have been implemented. Implementation of these functions can be seen in Appendix A & B.

B. *Speed of Robot*

To set the speed of robot a variable has been assigned that sets the speed of the robot in movement functions.

C. *Obstacle Avoidance*

The strategy followed for obstacle avoidance and arena exploration was shown in form for Flowchart in Fig 1. IR sensor 1 and 2 are used to detect if something is Infront of the robot. If there is an obstruction Infront of the robot Sensors 3 and 6 are used to make the turning decision. Whichever sensor shows less proximity turning is made in that direction. To avoid collision at edges sensors 2 and 7 are also used if anything offset to sensors 1 and 8 comes in the way of the robot it is detected by these two sensors and a turning decision is made. This can be understood by the help of diagram below.

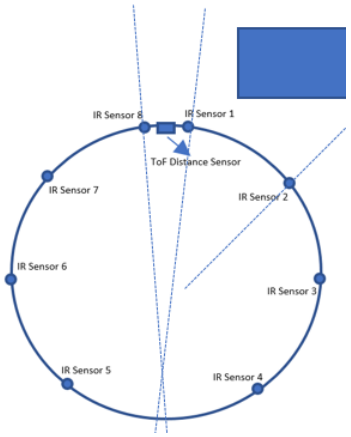

Figure 3 Detecting an object offset to sensor 1 and 2

The figure explains how readings from sensor 2 and 7 avoids robot to collide at edges. If something is detected by sensor 2 a right turn is taken and vice versa.

D. *Environment Exploration*

No extra measures were taken to do environmental exploration. Since the reading from the IR sensors are subject to change all the time. This induces randomness in the system it was just ensured that the robot didn't take actual 90 degree turn and that was achieved by using random delay while calling the turn left() and turn right() functions.

E. *Object Detection*

To detect the Time-of-Flight distance sensor is used along with IR proximity sensors . If the distance sensor points away from the object it gives a value > 120 mm then the IR sensors are used to detect the object and turn the robot towards the object. Once the distance sensor faces the object then to orient the robot properly towards the object and to detect any movement of the object sensors 1 and 8 are used if object moves towards right sensor 1 detects the movement and turns the robot towards right side . similar for left side movement but this time sensor 8 comes into action . Once the robot faces the object it moves towards the object to maintain a distance of 40-50 mm . A range is taken to avoid vibrations due to any inherent disturbances.

III. RESULTS

A. *Obstacle Avoidance*

The robot performed well in obstacle avoidance and exploration task. The motion of the robot was fast and smooth. For most of the part robot didn't collide with any obstacle but there were few collisions when a rectangular box was kept in robots path offset to the front IR sensors as shown in Figure 3 . Sometimes it detected the box and sometimes collided because of lack of calibration. Moreover, there were some instances when robot was unable to take a decision where to turn because the proximity of both the surfaces from the sensor on side were same. This was when it had to enter a narrow passage surrounded from all three sides. However, after several iterations it was able to escape such situations because of randomness of sensor data.

B. *Following an object*

The robot performed the task of following the object smoothly .As the object was placed randomly the robot was able to detect the object and made a clockwise or counter clockwise rotation depending on which side object was placed. Once the ToF sensor was facing the object robot moved towards the object and stopped at a fixed distance .If the object was brought closed the robot moved back and if the object was moved backward the robot moved towards the object. It was able to sense any lateral movement in the object and adjust accordingly. The only problem faced during the performance of the above task is when the lateral movement was small robot was unable to detect this movement due to fact that object was still facing the sensors.

IV. DISCUSSION

A. *Obstacle Avoidance*

- During obstacle avoidance task there were collisions when something came Infront of the robot but eccentric to the front sensors. For this purpose, sensors 2 and 7 were used as shown in figure 3. although the use of these sensors solved the problem, but they should be calibrated very carefully using geometric calculations.
- The value of IR sensor tends to vary in a range hence are not very reliable. Average sensor value can be considered for better results
- Bluetooth was used for getting the sensor values Realtime on the monitor. However, transmitting Bluetooth value delayed the overall response of the robot.

B. *Object Following*

- e-puck2 was able to respond to changes in object state if the changes were slow but as the object moved faster the response was not up to the mark.
- Small lateral displacement of the object was not sensed because the object was still Infront of the sensors.

V. APPENDIX

A. Code for Obstacle Avoidance using the e-puck2

```
// Header files
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "ch.h"
#include "hal.h"
#include "memory_protection.h"
#include <main.h>
Communication
#include "stdio.h"
#include "serial_comm.h"
#include "motors.h"                    // Motor Control
#include "sensors/proximity.h"         // Proximity Sensor


messagebus_t bus;
MUTEX_DECL(bus_lock);
CONDVAR_DECL(bus_condvar);

// Defining speed
int speed=600;

//Defining Movement functions
void turn_left(void)
{
   left_motor_set_speed(-1*speed);
   right_motor_set_speed(speed);
}

void turn_right(void)
{
   left_motor_set_speed(speed);
   right_motor_set_speed(-1*speed);
}

void move_forward(void)
{
   left_motor_set_speed(speed);
   right_motor_set_speed(speed);
}

void move_backward(void)
{
   left_motor_set_speed(-1*speed)
   right_motor_set_speed(-1*speed)
}

void stop(void)
{
   left_motor_set_speed(0);
   right_motor_set_speed(0);
}

//Main Function Begins

int main(void)
{
   halInit();
   chSysInit();
   mpu_init();
   messagebus_init(&bus,&bus_lock,&bus_condvar);
   clear_leds();
   spi_comm_start();
   proximity_start();
   calibrate_ir();
   motors_init();

/* Giving calibration values for IR Sensor Beforehand*/
   int sensor1_cal=300;
   int sensor2_cal=200;
   int sensor3_cal=800;
   int sensor4_cal=800;
   int sensor5_cal=800;
   int sensor6_cal=800;
   int sensor7_cal=200;
   int sensor8_cal=300;


   while (1)
   {

   int sensor1=get_prox(0);
   int sensor2=get_prox(1);
   int sensor3=get_prox(2);
   int sensor4=get_prox(3);
   int sensor5=get_prox(4);
   int sensor6=get_prox(5);
   int sensor7=get_prox(6);
   int sensor8=get_prox(7);


if (sensor1>sensor1_cal || sensor8>sensor8_cal)
   {
   if (sensor6>sensor3)
      {
      turn_right();
      }
   else if(sensor6<sensor3)
      {
      Turn_left();
      }
   }
   else if (sensor7>sensor7_cal)
   {
      turn_right();
   }
   else if (sensor2>sensor2_cal)
   {
      turn_left();
   }
   else if (sensor6>sensor6_cal)
   {
```

```
      turn_right();
   }
   else if (sensor3>sensor3_cal)
   {
      turn_left();
   }
   else
   {
      move_forward();
   }

   chThdSleepMilliseconds(100);

   }
}

#define STACK_CHK_GUARD 0xe2dee396
uintptr_t __stack_chk_guard = STACK_CHK_GUARD;

void __stack_chk_fail(void)
{
   chSysHalt("Stack smashing detected");
```

B. Code for Object Following using the e-puck2

```
// Header files
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "ch.h"
#include "hal.h"
#include "memory_protection.h"
#include <main.h>
Communication
#include "stdio.h"
#include "serial_comm.h"
#include "motors.h"                    // Motor Control
#include "sensors/proximity.h"         // Proximity Sensor
#include "sensors/VL53L0X/VL53L0X.h"   // Distance
Sensor

messagebus_t bus;
MUTEX_DECL(bus_lock);
CONDVAR_DECL(bus_condvar);

// use macro for succinct reference of these functions
#define leftMotor(speed) left_motor_set_speed(speed)
#define rightMotor(speed) right_motor_set_speed(speed)
#define delay(time) chThdSleepMilliseconds(time)

// Defining speed
int speed=600;

//Defining Movement functions
void turn_left(void)
{
   left_motor_set_speed(-1*speed);
   right_motor_set_speed(speed);
}

void turn_right(void)
{
   left_motor_set_speed(speed);
   right_motor_set_speed(-1*speed);
}

void move_forward(void)
{
   left_motor_set_speed(speed);
   right_motor_set_speed(speed);
}

void move_backward(void)
{
   left_motor_set_speed(-1*speed)
   right_motor_set_speed(-1*speed)
}

void stop(void)
{
   left_motor_set_speed(0);
   right_motor_set_speed(0);
}

int main(void)
{
   messagebus_init(&bus, &bus_lock, &bus_condvar);
   halInit();
   chSysInit();
   mpu_init();
   motors_init();
   proximity_start();
   calibrate_ir();
   serial_start();
   VL53L0X_start();

   while (1) {
   int distance=VL53L0X_get_dist_mm();
   int sensor1=get_prox(0);
   int sensor2=get_prox(1);
   int sensor3=get_prox(2);
   int sensor4=get_prox(3);
   int sensor5=get_prox(4);
   int sensor6=get_prox(5);
   int sensor7=get_prox(6);
   int sensor8=get_prox(7);

   if (distance<=40){
      move_backward();
      delay(200);
   }
   else if ( distance>40 && distance<=50)
   {
      stop();
   }
   else if ( distance>50 && distance<=120)
```

```
  {
   move_forward();
   }

   else if ( distance>120 && (sensor1-sensor8>15))
    {
      turn_right();
     }
   else if(distance>120&&(sensor8-senso1>15))
   {
      turn_left();
   }
else if (distance > 120 && (sensor1<50) && (sensor2<50)
&& (sensor3<50) && (sensor4<50) && (sensor5<50) &&
(sensor6<50) && (sensor7<50) && (sensor8<50))
{
   stop();
}
else if (distance>120&&sensor2>120)
{
      turn_right();
 }
else if (distance>120&&sensor3>120)
{
          turn_right();
 }
  else if (distance>120&&sensor4>120)
{
          turn_right();
 }
else if (distance>120&&sensor5>120)
{
 if(sensor4>120)
{
           turn_right();
 }
else
{
  turn_left();
 }
 }
   else if (distance>120&&sensor6>120)
{
turn_left();
}
else if (distance>120&&sensor7>120)
{
turn_left();
}
}

#define STACK_CHK_GUARD 0xe2dee396
uintptr_t __stack_chk_guard = STACK_CHK_GUARD;

void __stack_chk_fail(void)
{
   chSysHalt("Stack smashing detected");
}
```