Enhancing E-Puck Robot Performance in Autonomous Navigation Tasks ACS6501

Elena Manoli[1] and Saloni Sunil Badave[2]

***Abstract*— This report outlines strategies and implementation details aimed at enhancing the performance of the e-puck robot in two autonomous navigation tasks. In the first task, the robot is tasked with detecting obstacles using infrared sensors and applying obstacle avoidance behaviors. The second task requires the robot to chase an object while simultaneously implementing collision avoidance. This is achieved through the integration of a Time-of-Flight sensor and infrared sensors to ensure efficient and secure navigation. The results and discussions section critically analyzes the effectiveness of these strategies and proposes possible improvements.**

I. STRATEGIES

A. Task 1: Navigation and Obstacle Avoidance Strategy

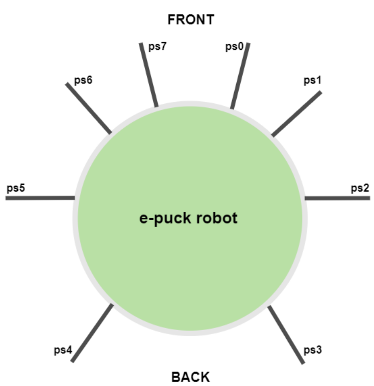

Fig. 1. Drawing of proximity sensors in the e-puck robot

The e-puck robot is equipped with a total of eight infrared sensors, as illustrated in Figure 1. However, our focus for obstacle avoidance involves the utilization of four sensors. More specifically, ps0 (Front Center Right Sensor) and ps1 (Front Right Sensor) identify obstacles on the right side, while ps7 (Front Center Left Sensor) and ps6 (Front Left Sensor) detect obstacles on the left side. When the readings from these sensors exceed a predefined threshold, determined through testing, it indicates the presence of an obstacle. The robot should respond by turning counter-clockwise if the obstacle is on the right side and turning clockwise if the obstacle is on the left side.

During testing and experimentation, a challenge occurred during the robot's navigation through narrow paths with obstacles on both sides. In such scenarios, the robot could enter an infinite loop, detecting an obstacle on one side, initiating a rotation, and subsequently encountering an obstacle on the opposite side. To address this challenge, a new function was introduced to evaluate the readings from both the Front Left Sensor and Front Right Sensor, verifying if they exceeded the predefined threshold. When both sensors indicate the presence of obstacles, the robot should perform a 180-degree rotation to effectively navigate through narrow passages. The flowchart provided in Figure 2 illustrates the key steps involved in obstacle detection and the subsequent control of the e-puck robot's movements.

B. TASK 2

The objective of Task 2 is to enable the robot to chase an object within an open environment while maintaining a safe distance to prevent collisions. If any sensors on the robot's right side detect an object, the robot should execute a clockwise rotation, allowing the front sensors to locate the object, and subsequently, the robot should proceed forward. In the event that sensors on the left side detect the object, a counter-clockwise rotation is warranted to position the front sensors for object detection, followed by forward movement. If the front sensors detect an object, the robot should move forward, but upon reaching a predefined distance threshold, the robot should stop to avoid collision. It is crucial to ensure that the robot maintains a constant distance from the object. When the object approaches too closely, the robot should move backward, and when it moves farther away, the front sensors should detect the change and prompt the robot to move forward. This dynamic adjustment in distance helps to maintain a consistent separation between the robot and the object. This task requires the use of six infrared sensors and the Time-of-Flight sensor. The decision to employ the Time-of-Flight sensor for the front detection is rooted in their ability to provide more precise and accurate distance measurements. Unlike proximity sensors, which primarily detect the presence or absence of an object within a certain range, distance sensors offer continuous distance readings.

II. IMPLEMENTATION

The robot's behavior is determined by the position of the selector. When the selector is set to '1', Task 1 is executed. Similarly, when the selector is set to '7', Task 2 is performed. Finally, when the selector is set to '3', the robot remains idle and stops moving. For both tasks, the first step is to calibrate the sensor values. At the beginning, a constant value called

[1]Manoli is with the Department of Automatic Control and Systems Engineering, The University of Sheffield, UK emanoli1@sheffield.ac.uk

[2]Badave is with the Department of Automatic Control and Systems Engineering, The University of Sheffield, UK ssbadave1@sheffield.ac.uk

MAX_SPEED is initialized with a value of 400. A total of 7 functions were used to control the robot's movement:

1. stop(): This function stops the robot from moving by setting the speed of each wheel to 0.

2. moveForward(): The robot moves forward with both wheels at the predefined speed set by MAX_SPEED.

3. goFast(): This function doubles the speed of both wheels instead of using the predefined MAX_SPEED value, allowing the robot to move forward faster.

4. goBackwards(): To make the robot move backward, the speed is set to a negative value, causing both wheels to move in reverse while maintaining the same MAX_SPEED magnitude.

5. rotateRight(): This function makes the robot turn to the right. It involves setting the speed of the right wheel to a negative value and the speed of the left wheel to the same magnitude but positive.

6. rotateLeft(): Similar to 'turnRight()', this function causes the robot to turn to the left. It sets the speed of the left wheel to a negative value and the speed of the right wheel to the same positive value.

7. trappedMovement(): This function is designed for Task 1. When the robot becomes trapped, meaning it detects obstacles on both sides, it performs a 180-degree rotation to escape. This is accomplished by calling rotateRight() once, followed by stop(), and concluding with another call to rotateRight().

These functions collectively allow the robot to perform various movements and maneuvers in response to sensor inputs.

In Task 1, the robot's body LED signals readiness, turning off only when obstacles are detected.

Three additional functions were implemented for Task 1 to implement predefined logic, as depicted in Figure 2. These functions objectOnTheLeft(), objectOnTheRight(), and trapped() are boolean functions. objectOnTheLeft() assesses the presence of an object on the left side by examining proximity sensors 6 and 7. If either sensor value exceeds the predefined threshold, the function returns true. Similarly, objectOnTheRight() checks for an object on the right side using proximity sensors 0 and 1, returning true if either sensor value surpasses the threshold. The trapped() function evaluates proximity sensors 1 and 6; if both sensor values exceed the predefined threshold, it indicates the robot is trapped, and the function returns true.

The main logic of Task 1 is that if the robot is trapped between two obstacles (trapped()), it executes a 180-degree rotation to escape by calling function trappedMovement(). If an object is detected on the right side (objectOnTheRight()), the robot performs a left rotation (rotateLeft()). If an object is detected on the left side (objectOnTheLeft()), the robot performs a right rotation (rotateRight). If no obstacles are detected, the robot continues moving forward (moveForward()). These functions allow the robot to autonomously adapt to various scenarios.

For Task 2, four additional boolean functions were implemented: objectOnTheLeft2(), objectOnTheRight2(), objectOnTheBackRight2(), and ObjectOnTheBackLeft2(). objectOnTheLeft2() checks proximity sensors 5 and 6 for left-side obstacles, returning true if either value exceeds the distinct predefined threshold. Similarly, objectOnTheRight2() assesses right-side obstacles using sensors 1 and 2, returning true if either value surpasses the threshold. objectOnTheBackRight2() and objectOnTheBackLeft2() check sensors 3 and 4, respectively, for back right and back left obstacles, returning true if either value exceeds the threshold. The robot utilizes a VL53L0X distance sensor to identify objects in its front, while proximity sensors detect objects on the robot's right, left, or back.

The main logic of Task 2 is that if an object is detected within the optimal range (between 40 and 50 millimeters), the robot promptly halts, extinguishing its body LED. In instances where an object is in close proximity (less than 40 millimeters), the robot initiates a backward movement while illuminating its LED in a distinct manner. Additional conditions check for objects on the right, left, back right, and back left, prompting appropriate rotational maneuvers to circumvent potential collisions. Speed adjustments are made depending on the distance of the object, with the robot accelerating when the object is within a range of 50 to 300 millimeters. If the distance exceeds 300 millimeters, the robot halts to avoid unnecessary movement. In scenarios where no obstacles are detected, the robot moves forward to continue its autonomous navigation.

III. RESULTS AND DISCUSSIONS

In Task 1, the e-puck successfully avoided collisions with its surroundings and navigated through the entire arena. It effectively maneuvered through narrow passages in most cases, demonstrating its capability to adapt to challenging situations. However, when the robot became trapped between obstacles, there was an observed need for improved adjustment of the proximity sensors, particularly Sensor 1 and Sensor 6, to enable obstacle detection at a closer distance from the walls. Additionally, considering that the e-puck robot exhibited a relatively slow movement, adjustments to its speed might be beneficial for enhanced performance.

During the demonstration of task 2, the robot successfully chased an object while maintaining a safe distance. It demonstrated the ability to adjust its position when the object approached, moving backward to avoid a collision. As the object surrounded the robot, it successfully identified it and executed a rotation, allowing the front sensors to detect the object as well. The robot's speed was well-calibrated so when the object was placed at a greater distance, the robot accelerated to chase it until it approached, at which point it reduced its speed appropriately. However, the robot required the object to be exceptionally close for reliable detection. An improvement suggestion involves fine-tuning the proximity sensor values to enhance object detection, ensuring the robot's capability to identify objects even when placed at greater distances.

APPENDIX

```c
#include <main.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "ch.h"

#include "epuck1x/uart/e_uart_char.h"

#include "hal.h"

#include "leds.h"

#include "memory_protection.h"

#include "motors.h"

#include "selector.h"

#include "sensors/VL53L0X/VL53L0X.h"

#include "sensors/proximity.h"

#include "serial_comm.h"

#include "spi_comm.h"

#include "stdio.h"

#define MAX_SPEED 400

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

// Function to stop the robot from moving
void stop() {
  left_motor_set_speed(0);
  right_motor_set_speed(0);
}

// Function for the robot to move forward
void moveForward() {
  left_motor_set_speed(MAX_SPEED);
  right_motor_set_speed(MAX_SPEED);
}

// Function for the robot to go backward
void goBackwards() {
  left_motor_set_speed(-MAX_SPEED);
  right_motor_set_speed(-MAX_SPEED);
}

// Increase the speed of the wheels so the robot can move faster
void goFast() {
  left_motor_set_speed(800);
  right_motor_set_speed(800);
}

// Adjust the speed of each wheel to achieve left rotation (left wheel negative, right wheel positive)
void rotateLeft() {
  left_motor_set_speed(-MAX_SPEED);
  right_motor_set_speed(MAX_SPEED);
}

// Adjust the speed of each wheel to achieve right rotation (right wheel negative, left wheel positive)
void rotateRight() {
  left_motor_set_speed(MAX_SPEED);
```

```c
  right_motor_set_speed(-MAX_SPEED);
}

// When the robot is trapped between 2 walls and the only way to escape is to rotate 180 degrees,
//otherwise it will be stucked in an infinite loop rotating left and right
// call the rotateRight() function twice
void trappedMovement() {
  rotateRight();
  chThdSleepMilliseconds(500);
  stop();
  chThdSleepMilliseconds(500);
  rotateRight();
}

//-----------------Functions for Task 1--------------------------

// If there is an object on the right side (ps0 or ps1) then return true since the function is boolean
bool objectOnTheRight() {
  bool detectObjectOnTheRight = false;
  int prox0 = get_calibrated_prox(0);
  int prox1 = get_calibrated_prox(1);

  if (prox0 > 450 || prox1 > 450) {
    char str[100];
    int str_length =
        sprintf(str, "Object on the Right  ---- Sensor0: %d, Sensor1: %d\n",
                prox0, prox1);
    e_send_uart1_char(str, str_length);
    detectObjectOnTheRight = true;
  }
  return detectObjectOnTheRight;
}

// If there is an object on the Left side (ps6 or ps7) then return true since the function is boolean
bool objectOnTheLeft() {
  bool detectObjectOnTheLeft = false;
  int prox6 = get_calibrated_prox(6);
  int prox7 = get_calibrated_prox(7);

  if (prox6 > 450 || prox7 > 450) {
    detectObjectOnTheLeft = true;
    char str1[100];
    int str_length1 =
        sprintf(str1, "Object on the Left  ---- Sensor6: %d, Sensor7: %d\n",
                prox6, prox7);
    e_send_uart1_char(str1, str_length1);
  }

  return detectObjectOnTheLeft;
}

// If the robot is trapped meaning that both left and right sensors detect an object (ps1 and ps6) then
    return true since the function is boolean
bool trapped() {
  bool robotTrapped = false;
  int prox6 = get_calibrated_prox(6);
  int prox1 = get_calibrated_prox(1);

  if (prox1 > 250 && prox6 > 250) {
    robotTrapped = true;
    char str3[100];
    int str_length3 = sprintf(
        str3, "Robot is Stuck  ---- Sensor1: %d, Sensor6: %d\n", prox1, prox6);
    e_send_uart1_char(str3, str_length3);
  }

  return robotTrapped;
}



//----------Functions for Task 2-------------------------

// Check if there is an obstacle on the right side using ps1 and ps2
bool objectOnTheRight2() {
```

```c
  bool detectObjectOnTheRight = false;
  int prox1 = get_calibrated_prox(1);
  int prox2 = get_calibrated_prox(2);

  if (prox1 > 200 || prox2 > 200) {
    char str[100];
    int str_length =
        sprintf(str, "Object on the Right  ---- Sensor1: %d, Sensor2: %d\n",
                prox1, prox2);
    e_send_uart1_char(str, str_length);
    detectObjectOnTheRight = true;
  }

  return detectObjectOnTheRight;
}


// Check if there is an obstacle on the left side using ps5 and ps6
bool objectOnTheLeft2() {
  bool detectObjectOnTheLeft = false;
  int prox5 = get_calibrated_prox(5);
  int prox6 = get_calibrated_prox(6);

  if (prox5 > 200 || prox6 > 200) {
    char str1[100];
    int str_length1 =
        sprintf(str1, "Object on the Left  ---- Sensor5: %d, Sensor6: %d\n",
                prox5, prox6);
    e_send_uart1_char(str1, str_length1);
    detectObjectOnTheLeft = true;
  }

  return detectObjectOnTheLeft;
}

// Check if there is an obstacle on the back right side using ps3
bool objectOnTheBackRight2() {
  bool detectObjectOnTheBackRight = false;
  int prox3 = get_calibrated_prox(3);

  if (prox3 > 200) {
    char str3[100];
    int str_length3 =
        sprintf(str3, "Object on the back Right  ---- Sensor3: %d", prox3);
    e_send_uart1_char(str3, str_length3);
    detectObjectOnTheBackRight = true;
  }

  return detectObjectOnTheBackRight;
}

// Check if there is an obstacle on the back left side using ps4
bool objectOnTheBackLeft2() {
  bool detectObjectOnTheBackLeft = false;
  int prox4 = get_calibrated_prox(4);

  if (prox4 > 200) {
    char str4[100];
    int str_length4 =
        sprintf(str4, "Object on the back left  ---- Sensor4: %d", prox4);
    e_send_uart1_char(str4, str_length4);
    detectObjectOnTheBackLeft = true;
  }

  return detectObjectOnTheBackLeft;
}


int main(void) {
  halInit();
  chSysInit();
  mpu_init();
  serial_start();
  messagebus_init(&bus, &bus_lock, &bus_condvar);
  proximity_start(0);
```

```
  calibrate_ir();
  motors_init();
  clear_leds();
  spi_comm_start();
  VL53L0X_start();

  while (1) {

    // When selector is rotated to 1 Task 1 is executed
    if (get_selector() == 1) {
      //---Task1---
      // The LED is always on, it turns off only when the robot detects an object on either side, and
    after moving forward, it turns on again
      set_body_led(1);

      // If the robot is trapped between 2 walls, it rotates 180 degrees
      if (trapped()) {
        trappedMovement();
      }

      // If there is an object on the right side, the robot rotates left
      else if (objectOnTheRight()) {
        set_body_led(0);
        rotateLeft();
      }

      // If there is an object on the left side, the robot rotates right
      else if (objectOnTheLeft()) {
        set_body_led(0);
        rotateRight();
      }

      // If the robot detected nothing, then move forward
      else {
        moveForward();
      }

      chThdSleepMilliseconds(100);


    // When selector is rotated to 7 Task 2 is executed
    } else if (get_selector() == 7) {
      // ---Task2----
      // The LED is always on, it turns off when the distance sensor detects something.
      // It blinks when the robot is moving backward
      set_body_led(1);

      // For Task 2, the distance sensor was utilized to identify if there is an object in front of the
    robot.
      // Proximity sensors were used to identify if there is an object on the right, left, or back of the
     robot

      if (VL53L0X_get_dist_mm() >= 40 && VL53L0X_get_dist_mm() < 50) {
        stop();
        set_body_led(0);
      } else if (VL53L0X_get_dist_mm() < 40) {
        goBackwards();
        set_body_led(2);
      }

      else if (objectOnTheRight2()) {
        rotateRight();
      } else if (objectOnTheLeft2()) {
        rotateLeft();
      } else if (objectOnTheBackRight2()) {
        rotateRight();
      } else if (objectOnTheBackLeft2()) {
        rotateLeft();
      } else if (VL53L0X_get_dist_mm() > 50 && VL53L0X_get_dist_mm() < 300) {
        goFast();
      } else if (VL53L0X_get_dist_mm() > 300) {
        stop();
      }

      else {
```

```c
        moveForward();
      }

    // If the selector is at position 3, stop both motors
    } else if (get_selector() == 3) {
      left_motor_set_speed(0);
      right_motor_set_speed(0);
    }
    chThdSleepMilliseconds(100);
  }
}

#define STACK_CHK_GUARD 0xe2dee396
uintptr_t __stack_chk_guard = STACK_CHK_GUARD;

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