GAROB

University of Aveiro

Department of Electronics and Telecommunications

Dr. Keith L. Doty

Scott Jantz

Tiago Oliveira

Alfredo França

July 1995

1. Introduction

The main objective of this course is to build a robot. The robot we built, GAROB (GAROB stands for GArbage ROBot), is intended to collect garbage and pushs it to the wall. The robot moves randomly and have to avoid obstacles that appear in his way. Other feature of GAROB is the capacity to automatic recharging when the batteries are low.

2. Mobile Platform

The mobile platform is a wood made, circular shaped platform, with three wheels. The robot is pushed by the two front wheels and the third wheel serves to stand the robot.

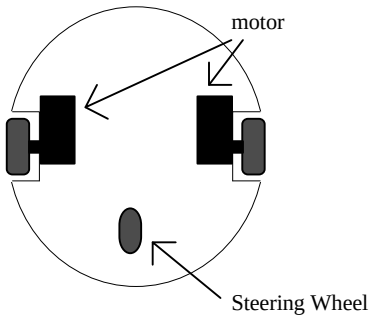


Figure 2.1 Down view of the GAROB.

3. Actuation

At the moment, we only have two DC-motors, which control the GAROB's movement.

4. Sensors

For detecting obstacles, we supplied GAROB with three infra-red (IR) LED's and three sensors. Each sensor is associated with one LED. The sensors are disposed as figure 4.1 shows. The sensors measure the signal level emited by the corresponding LED after reflection by the obstacles. The LED's used are MLED81 by MOTOROLA. The sensors used are SHARP sensors with a little modification to allow receiving an analog signal instead of the original digital signal.

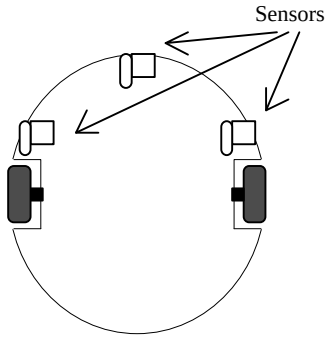


Figure 4.1 Sensor position.

5. Behaviors

At the moment, GAROB has two behaviors, which permit it to "walk" arround without crashing. The distance sensors are placed in the front of GAROB and are adjusted for three levels of proximity sensitivity: far, close and danger (see figure 5.1).

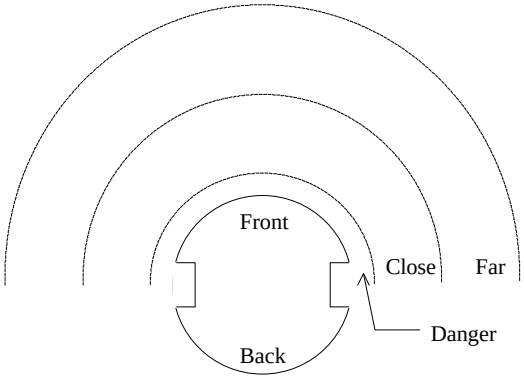


Figure 5.1 Levels of proximity sensitivity

5.1. Collision Avoidance Behavior

When the robot enters the Far zone, he slows down to allow more accuracy. When he reaches the Close zone, he moves away from the obstacles. In the Danger zone, the behavior is not yet defined.

5.2. Random Turn Behavior

This behavior is characterised by puting the robot turning arround in a random direction during a random interval of time. GAROB swaps to this behavior when he detects an obstacle in the Close zone with the front sensor.

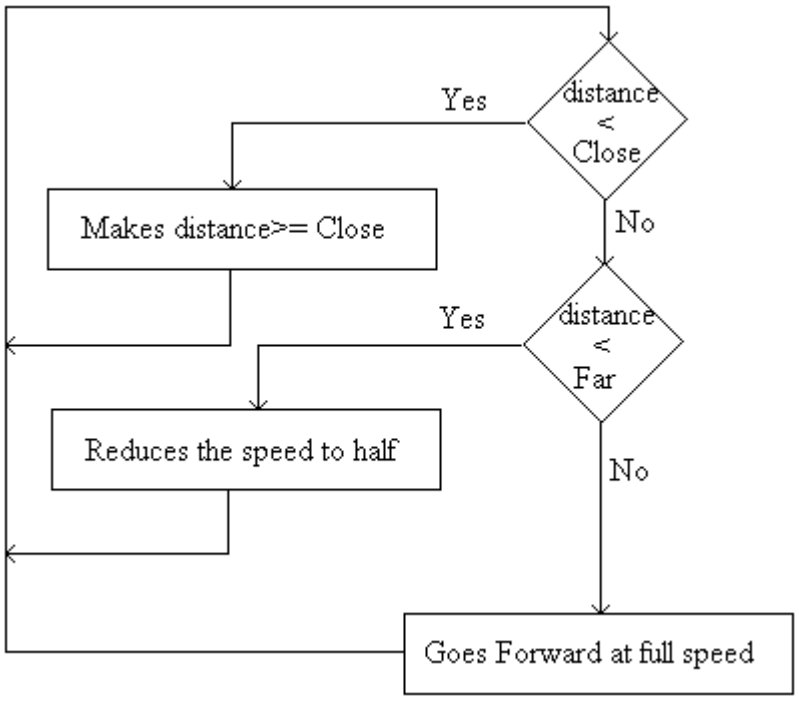
7. Conclusion

GAROB was tested, with the current version of software, and he avoid obstacles with accuracy, so at this moment we think that we are in the right way.

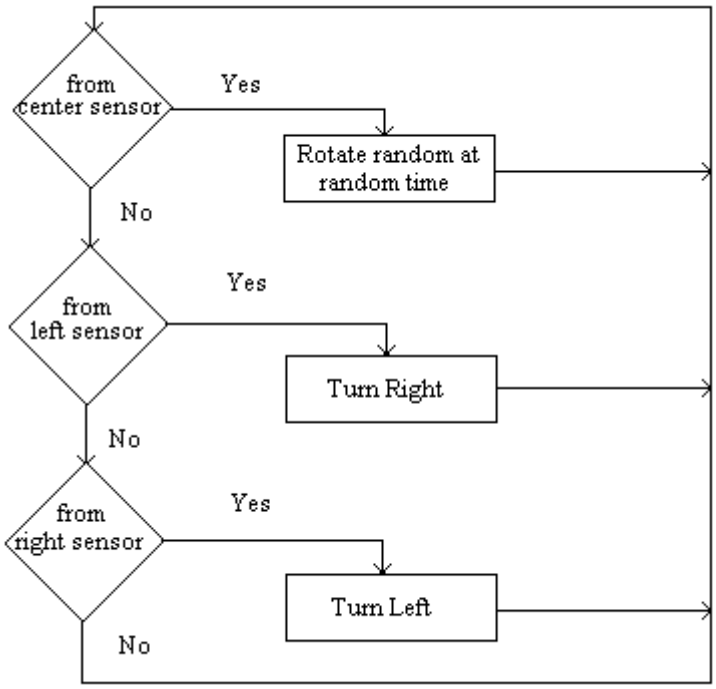
8. Appendices

8.1. Flow Chart

The next flow chart represents the combined behavior avoidance and random turn.



Next chart represents the “Makes the distance>=Close”.



8.2. Program Listing

```
/*
 *
 * July 1995
 *
 */


/* DEFINITIONS */
int RIGHT=0,
    LEFT=1;

int MotorRight=0,       /* motor positions */
    MotorLeft=1;

float SpeedRight=0.0,   /* motor speeds */
      SpeedLeft=0.0;

int EyeAddress=0x4000,  /* eyes memory address */
    EyeMask=0x07,       /* eyes data position */
    EyeRightBit=0,      /* right eye data position */
    EyeCenterBit=1,     /* center eye data position */
    EyeLeftBit=2;       /* left eye data position */

int IrDelay=50;         /* delay time for IR commutation */

float s0= 10.0,         /* motor speeds */
      s1= 20.0,
      s2= 40.0,
      s3= 60.0,
      s4= 80.0,
      s5=100.0;

int ThresFrontFar=95,      /*  97 *//* threshold levels for IR sensors */
    ThresFrontClose=115,   /* 112 */
    ThresFrontDanger=132,  /* 132 */
    ThresSideFar=95,       /*  95 */
    ThresSideClose=110,    /* 110 */
```

```
    ThresSideDanger=130;   /* 130 */


/* GLOBAL VARIABLES */
int eyeRight=0,
    eyeCenter=0,
    eyeLeft=0;

void wait(int m_second) {
  long stopTime;

  stopTime = mseconds() + (long)m_second;
  while(stopTime > mseconds()) defer();
}

void freeze() {
  SpeedRight=0.0;
  SpeedLeft=0.0;
}

void forward_fullSpeed() {
  SpeedRight=s5;
  SpeedLeft=s5;
}

void forward_halfSpeed() {
  SpeedRight=s3;
  SpeedLeft=s3;
}

void forward_slowSpeed() {
  SpeedRight=s1;
  SpeedLeft=s1;
}

void turn_left() {
  SpeedLeft=0.0;
  SpeedRight=s3;
}

void turn_right() {
  SpeedRight=0.0;
  SpeedLeft=s3;
}

void rotate_clockwise() {
  motor(MotorRight, 0.0);
  motor(MotorLeft, 0.0);
  SpeedRight=-s1;
  SpeedLeft=s1;
}

void rotate_anticlockwise() {
  motor(MotorRight, 0.0);
  motor(MotorLeft, 0.0);
  SpeedRight=s1;
  SpeedLeft=-s1;
}

void sensor_read() {
  while (1) {
    poke(EyeAddress, EyeMask);
    wait(IrDelay);
    eyeRight=analog(EyeRightBit);
    eyeCenter=analog(EyeCenterBit);
    eyeLeft=analog(EyeLeftBit);
    poke(EyeAddress, 0);
    wait(IrDelay);

    wait(50);
  }
}

void motor_write() {
  while(1) {
    motor(MotorRight, SpeedRight);
    motor(MotorLeft, SpeedLeft);
    wait(100);
  }
```

```
}

void collision_avoidance() {
  int timeRotation,
      dirRotation;

  while (1) {
    if( (eyeRight > ThresSideDanger) ||
        (eyeLeft > ThresSideDanger)  ||
        (eyeCenter > ThresFrontDanger) ) {
      timeRotation = (int)mseconds() & 0x03FF;
      dirRotation = (int)mseconds() & 0x0001;
      freeze();         /* APAGAR */
      wait(1000);        /* APAGAR */
      if ( dirRotation == LEFT ) rotate_anticlockwise();
      else rotate_clockwise();
      wait(timeRotation);
    }
    else
      if ( eyeCenter > ThresFrontClose ) {
        timeRotation = (int)mseconds() & 0x03FF;
        dirRotation = (int)mseconds() & 0x0001;
        if ( dirRotation == LEFT ) rotate_anticlockwise();
        else rotate_clockwise();
        wait(timeRotation);
      }
      else if ( (eyeLeft > ThresSideClose) && (eyeLeft > eyeRight) )
        turn_right();
      else if ( eyeRight > ThresSideClose )
        turn_left();
    else
      if ( (eyeRight > ThresSideFar) ||
           (eyeLeft > ThresSideFar)  ||
           (eyeCenter > ThresFrontFar) )
        forward_slowSpeed();
      else forward_fullSpeed();
  }
}

void main()
{
  int sensor_read_pid,
      motor_write_pid,
      collision_avoidance_pid;

  sensor_read_pid=start_process(sensor_read());
  motor_write_pid=start_process(motor_write());
  collision_avoidance_pid=start_process(collision_avoidance());
}
```

8.3. Schematic Diagram