DOCUMENTATION
-Robot Project-

Students:

Novacean Ligia

Samarghitan Flaviu

(Group 30434)

Teaching Assistant:

Razvan Itu

Problem Statement:

The main purpose of this project is to develop a car-like robot that can perform certain tasks, starting from a given kit. Therefore, the car has been augmented with: obstacle detection and avoidance, remote control from a mobile application, an object following mode and a light following mode.

Hardware Components:

- 1 Arduino Uno R3
- 2 DC Motors
- 1 Servo Motor
- 1 Sonar Sensor
- 1 HC-05 Bluetooth Module
- 1 Buzzer
- 2 Breadboards

Usage

The car starts by default in Bluetooth mode, where the control needs to be taken from the mobile application. We use the "Bluetooth RC Controller" application found on the Google Play Store, but any kind of similar application will do fine. In this mode, the car can receive different commands from the user about its movement.

Once a device is connected to the Bluetooth module the mode can be changed to either the Obstacle Avoidance mode or the Object Following mode.

In the Obstacle Avoidance mode the car will dodge any collision with elements of the environment. Whenever any such element arises the car will stop and look to its sides, choosing the option where the distance is greater.

In the Object Following mode the car will follow an object that should initially be in front of it. If the followed object changes direction the car will stop and emit a sound to mark the fact that it has lost its path. It will look in all directions for the object and move accordingly.

Design Overview:

- Ultrasonic Sensor Reading

    - Implemented in ReadSensor.cpp

- The sensor has an echo and a trigger pin, set as output pins
- Initially, we clear the trigger pin by setting it to low voltage for 2 microseconds. After setting it on high for 10 microseconds, by using the pulseIn functions we read the echo pin, returning the sound wave travel time in microseconds and the value is converted to distance by multiplying with the speed of sound and dividing to 2 (because the distance was traveled twice until the receiver could read the information).

Motion

- Implemented in MoveRobot.cpp
- Motors have been adjusted to assure proper forward movement (using a 95% factor for the left motor when moving forward)
- The motion of the robot is assured by the two DC Motors. Each of them has 2 pins:
  - En: connected to 1 through jumpers, so the user has nothing to do to it
  - In1 and In2: control the robot's movement
- *moveForward(int speed)*: In1 = 0, In2 = 1 (for both motors)
- *moveBackward(int speed)*: In1 = 1, In2 = 0 (for both motors)
- *moveRight(int speed)*: the right motor is stopped, while the left one is working
- *moveLeft(int speed)*: the left motor is stopped, while the right one is working
- *stopMotor()*: both motors are stopped(In1 = In2)

Environment analysis

- Implemented in ControlServo.cpp
- In order to look around the scene the Ultrasonic sensor has to move its position in all directions. This is where the servo motor intervenes, rotating the head of the robot
- For this we use the *Servo.h* library and declare a Servo type object that will be used to implement all the operations
- *rotateServo(int angle):* rotates the head of the robot (where the Ultrasonic Sensor located) to the given position (0^0 for right, 90^0 for forward and 180^0 for left)

Obstacle Detection and Avoidance

- Implemented in AvoidObstacle.cpp
- *detectObstacle()*: function which performes 3 measurements of the distance in 3 directions (left, middle and right) and then filters this results. Therefore, if 2 out of those 3 measurement done in one direction have the result smaller than 25 cm (the

obstacle is in front of us), the function returns true, meaning that an obstacle has been detected.

- *avoidanceType():* indicates the type of avoidance that should be performed, i.e. whether the car should turn right of left. It rotates the servo motor to the right (servo.write(0)) and to the left (servo.write(180)). It returns an integer corresponding to the direction where the detected object was further
- *avoidObject():* we power the two DC motors in such a way that the car turns left or right with 90 degrees. We empirically detected that by using the middle speed of 128 for about 500 ms, the turn is an almost perfect one.

Remote Control

- We communicate with the HC-05 Bluetooth Module by means of serial communication (since the main serial ports are free we will use pins 0 and 1) and a mobile app.
- We constantly check in the main loop if a new message has been transmitted through the serial port (*if (Serial.available())* and read said message (*c=(char)Serial.read()*)
- The application sends characters indicating the action that the car should perform and we interpret them. Therefore, using the app we also control the functioning mode of the car.

  F: move forward
  B: move backward
  R: move right
  L: move left
  W/w: enter remote control mode
  V/v: enter light follow mode
  U/u: enter obstacle avoidance mode
  X/x: enter follow object mode

Object following

- Implemented in Follow.cpp
- In the main loop we look for the object with the *detectObstacle()* method and just keep moving forward if we see the object
- If the car loses sight of the object the buzzer will alert us of this *analogWrite(buzzerPin, 127);*

- Otherwise, we use the following methods:
  - *findObject():* that looks around 5 directions (0^0, 45^0, 90^0, 135^0, 180^0) using the *rotateServo()* method and finds the angle that held the smallest distance using the *minD(int *d)*
  - *followObject():* after the decision has been made we move the car in the direction of the object

- When a change of direction is detected, the car's direction switches to the one indicated by *findObject(),* the Servo is rotated to 90^o and it continues moving forward on the new selected direction