Obstacle Avoiding Bot: Report

Mukalel Ashish

December 2017

An obstacle avoiding bot is designed. The design and the way it works is outlined in this report.

1 Working Principle

The working of the bot is as follows: It moves in a straight line when it doesn't detect any obstacle in front of it. Once an obstacle is detected, it scans to the left and right and then turns to the direction with more space.

2 Implementation

2.1 Design

The basic design of the bot contains a wooden plank on which all the objects are placed. Two wheels, connected to geared motors, were stuck under the wooden plank, which would help the bot to move and turn. A specifically designed spherical wheel was attached to the front in order to support the weight. An Arduino Uno was used as the micro-controller. A ultrasonic range sensor was used to measure the distance to an obstacle. This was placed on top of a 180^o servo motor which enables the possibility to measure the free space to both the sides in case of detection of an obstacle. 9 volt batteries were used to power the motors and the servo. 4×1.2V rechargeable batteries were used to power the arduino uno. Everything was stuck in place using glue and double-sided tapes.

2.2 Electronics

The arduino uno was powered using 4×1.2V batteries. Most of the elctronics were pretty much straight forward. The servo motor had a V_{cc} pin connected to +9V (external), GND pin connected to the common ground and a control pin connected to the arduino via a breadboard. The ultrasonic range sensor had a V_{cc} pin connected to the +5V on the arduino, GND pin connected to the common ground, a Trig pin and an Echo pin, both connected to the arduino via the breadboard.

A motor driver was not available, so a circuit with a transistor, a capacitor and a diode was used in order to individually control the motors. The circuit diagram for motor control is shown in Figure 1.

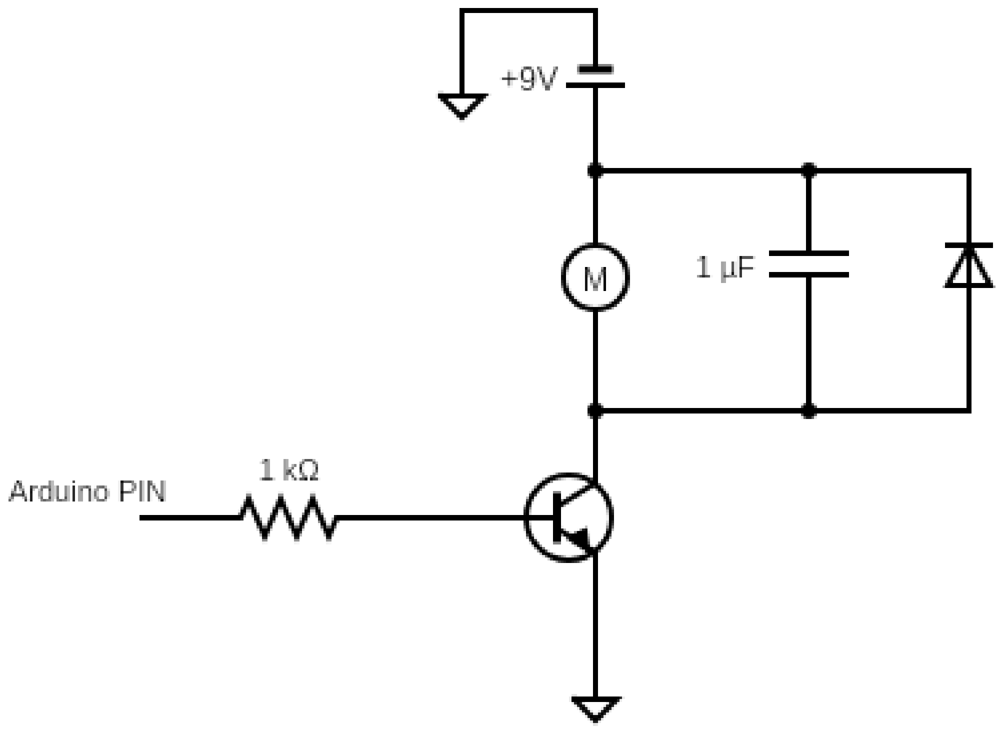

Figure 1: Circuit for powering the motor.

2.3 Program

The code used for controlling the servo motor is:

```
#include<Servo.h>

int servopin = 8;
int angle = 90;
Servo myservo;

void setup() {
  myservo.attach(servopin);
}

void loop() {
  myservo.write(angle);
}
```

where angle is the angle to which the servo motor would turn.

For the ultrasonic range sensor, the code used is:

```
int trigPin = 9;
int echoPin = 10;

long duration;
int distance;

void setup(){
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
}
```

The code for controlling the motors were pretty straight forward. The pins corresponding to both the motors were set to HIGH to move forward, the left motor HIGH and the right motor to LOW to turn right and vice versa to turn left.
The complete code can be found here.

3 Issues Encountered

One of the major issues encountered was during the initial stages of the project. An IR distance sensor was used initially as the sensor instead of the ultrasonic range sensor. As the effective range of the IR sensor was in the range of a few millimeters, the bot didn't have enough time to stop after detecting an obstacle. This problem was solved using a ultrasonic range sensor, which has an effective range in the order of a few meters.

Another issue faced was that the motors would keep rotating but the bot wouldn't move forward. This was because the floor was too smooth. This was overcome by putting some weight on the wooden plank. Another thing that was learned the hard way during this project was that servo motors shouldn't be powered using the power supply on the arduino uno board, but instead should be powered externally.

4 Scope for Improvement

Although this bot is a very basic one, a lot more features can be added to make this a pretty useful one. I'm planning on adding a camera to the bot and make the bot follow a path using OpenCV and Python.