Sumo Robot

Report

This is the project report for the robot “Interpreter” made for the class MAE 6194

Kalpesh Patil
kalpeshpatil33@gwu.edu

Kanishke Gamagedara
kanishkegb@gwu.edu

Spring 2017

The George Washington University

Contents

Summary

Figure 1: Interpreter

We made the “Interpreter” for the Sumo Robot Competition, which was held as a part of the course MAE 6194. This robot has an LCD serial monitor, which is operated through XBee, which displays (or interprets) everything it does. This lead to the name Interpreter. The basic functions and features of the robot are listed below.

- Finds the opponent using sonar sensors
- Detects the line using QTI sensors
- If and opponent is detected, accelerates towards it
- If the opponent is close enough, raise the flipping shield
- Send every task the robot is performing through XBee
- Another Arduino board reads the transmitted data via an XBee, and prints to an LCD display

This report is organized as follows. First, we will discuss the details on the body structure and the sensors. Then we will be explaining how the attacking mechanism works. Next section is on details of the LCD serial monitor system which is used to continuously monitor and debug the robot. Then we will provide an overview of the overall algorithm which is used to control the robot. Finally the issues we faced and the proposed future developments. The codes are attached in the appendix.

Physical Structure

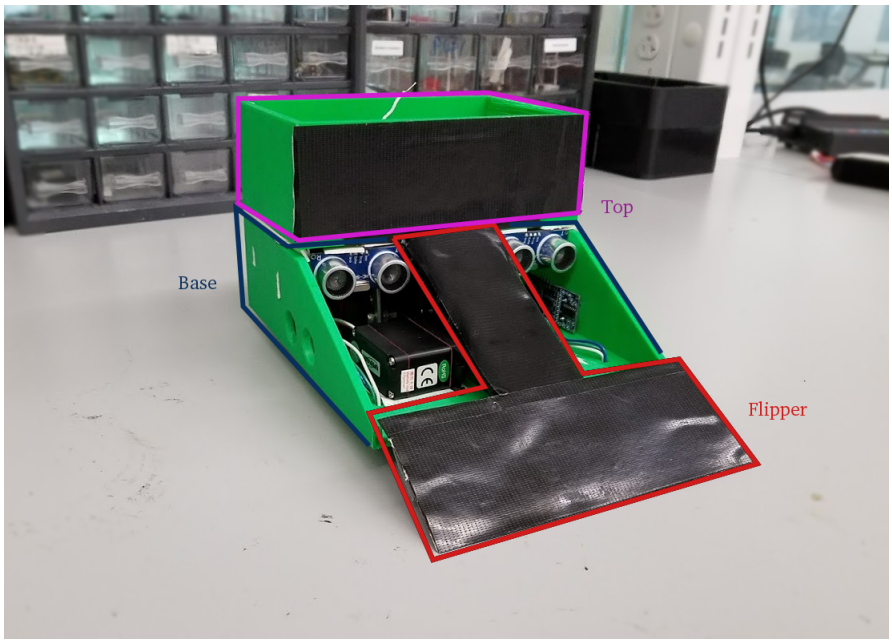

Figure 2: Structure

In this section, we discuss about the body structure of the robot and the sensors. The robots structure mainly has three parts.

1. Base
2. Top
3. Flipper

Base holds all the motors, back and side sonar sensors, and the QTI sensors at the bottom. On top of the Base, we have placed the Top which securely holds the Arduino board, batteries, Xbee unit, and the two front sonar sensors. Flipper is hinged to the front bottom edge of the Top and a servo motor with an arm is placed on the base, directly under the flipper, to activate the attacking mechanism.

Sensors and Components

A physical structure of a robot is nothing without sensing or actuation devices. In this section, we describe the sensors and components we used on our robot.

Line Detection

The most important objective of a Sumo Robot is to detect the boundary line of the ring. For this, we used QTI sensors by Parallax[1], where QTI stands for Q - charge, T - transfer and, I - infrared. We used three QTI sensors, one in the center and two in the front left and front right.

Figure 3: QTI Sensor

Referring fig. 3, pin W is supplied with 5 V and pin B is connected to the ground. Pin R is the sensor pin. As name implies, the sensor need to be charged. This is done by setting the sensor pin to HIGH during the initialization. In the case of middle QTI sensor, we need to shutdown the robot as soon as it detects the white line. For this, we used interrupts. Example code can be found below.

```
void setup() {
    pinMode(MIDDLE_pinQTIsensor, INPUT);  // set sensor pin as an input
    digitalWrite(MIDDLE_pinQTIsensor, HIGH);  // charging phase of the QTI

    attachInterrupt(digitalPinToInterrupt(MIDDLE_pinQTIsensor), MIDDLE_whiteLineISR, LOW);
    // attach the interrupt
 }

void MIDDLE_whiteLineISR() {
    goStop();  // stop the robot as soon as middle QTI detects a line
}
```

[1] https://www.parallax.com/product/555-27401

Sonar Sensors

The next objective of the robot is to identify the opponent robot. We used five HC-SR04 sonar sensors[2]. Two sensors were used in the front, and other three were used on left, right and back. The sensor sends a sonar signal and receives it back. The time difference between sending and receiving is used to calculate the distance of and obstacle. The sensor has a range of 2 - 500 cm for obstacle detection.

Figure 4: HC-SR04

This sensor has four pins: power, ground, trigger, and echo. Power in is supplied with 5 V. We need to write the pulse to the trigger pin and then read the echo. For this we use built-in Arduino function "pulseIn"[3]. This function returns the time the length of the pulse in microseconds. Since we know that the speed of sound is 340 $m/s = 0.034\ cm/\mu s$, we can calculate the distance to the obstacle.

```
int FRONT_distance = 0  // initialize the distance to an obstacle facing front
                        // sensor as a global variable

void FRONT_HC() {
    // writes the pulse
    digitalWrite(FRONT_trigPin, LOW);
    digitalWrite(FRONT_trigPin, HIGH);
    delay(100);
    digitalWrite(FRONT_trigPin, LOW);

    FRONT_duration = pulseIn(FRONT_echoPin, HIGH);  // read the echo
    FRONT_distance = (FRONT_duration / 2) * 0.034;  // calculate the distance
}

void loop {
    FRONT_HC();  // call the front sonar function
}
```

[2] https://goo.gl/ZwPIIL
[3] https://www.arduino.cc/en/Reference/pulseIn

Motors

Motors are the actuators of our robot. We used three motors: two for the motion of the robot and one for the attacking mechanism. For the wheel motors, we use Parallax continuous rotation servo motors[4]. Each of the motors has a power pin (5 V), ground pin, and a servo pin. In this section we mainly focus on wheel motors. The motor we used for the attacking mechanism is described in section 4.

Figure 5: Parallax continuous rotation servo motor

First thing you have to do with the servo is to calibration. For example, servo motor is supposed to stop when the servo pin is written 90. But, there might be cases where the motor might not be 100% stopped when 90 is written to the servo pin. To calibrate, we have to write 90 to servo and rotate the small screw until the motor is completely stopped. We have two servo motors facing opposite sides. The list of servo pin values that need to written to each motor is given in table 1

Table 1: Servo pin values to be written for each operation

Operation	Left motor	Right motor
Forward	0	190
Backward	190	0
Stop	90	90
Left rotate	0	0
Right rotate	190	190

Further, depending on the time period you write these values, we can determine the angle of rotation. If you left rotate for 200 ms and 400 ms, robot will turn 90 and 180 degrees respectively.

Below is snippet gives a basic idea on this.

[4] https://www.parallax.com/product/900-00008

```
// define servo pins as global variables
#define LEFT_servo = 9
#define RIGHT_servo = 10

void setup() {
    // attach the servos to servo object
    leftServo.attach(LEFT_servo);
    rightServo.attach(RIGHT_servo);
}

void goRight() {
    leftServo.write(190);
    rightServo.write(190);
}

void goStop() {
    leftServo.write(90);
    rightServo.write(90);
}

void loop() {
    // rotate 90 degrees right
    goRight();
    delay(200);

    // stop
    goStop();

    // rotate 180 degrees right
    goRight();
    delay(200);
}
```

Attacking Mechanism

In this section, we explain details about the attacking mechanism installed in our robot.

Display System

Algorithm

Sumo Bot Code

LCD System Code

List of Components

Future Developments