University at Buffalo

CSE 453: Hardware and Software Integrated Systems

Robotics and Programming

Design Documentation

Thalaikya Voggu
Satya Kranthi Penumanchili
Kai Zheng
Edward Selig
Jesse Taylor

May 17, 2017
Contents

1 Goals ................................................. 2

2 Implementation ...................................... 3
   2.1 3D Printed Parts ..................................... 3
      2.1.1 3D Parts List Table ................................. 4
      2.1.2 3D Parts Design Sketch ....................... 4
      2.1.3 Tires .................................................. 6
   2.2 Circuit ............................................... 7
   2.3 Programming for the Motion of the Robot ............... 8
      2.3.1 Controlling the Motion of the Car Robot ........ 7
      2.3.2 Controlling the LEDs of the Car Robot .. 8
      2.3.3 Controlling the Sensors of the Car Robot ... 9
      2.3.4 Flowchart for the robot functions ............ 10

3 Parts List ............................................. 11

4 Future Improvements ................................. 11

5 References ............................................ 11
1 Goals

Using a obstacle avoidance robot and a website teach high school students how to code incrementally in order to generate interest in STEM fields. The obstacle avoidance robot should be design and implemented so that it is budget friendly. The project should help students learn how to code therefore having a website that teaches how to program the robot step by step will allow the kids to get started. In addition, the website will contain a resources page to allow the students to go beyond the scope of this project so that they are able to learn more.

Since the project has to be of low cost the project uses an Arduino Uno. Reason for choosing the Arduino Uno is that it has multiple uses. The Arduino can be used for a different project later one. Another reason is so that the programming platform is very close to the one that is currently being used at the high school (ROBOTC).
2 Implementation

2.1 3D Printed Parts

Fig. 1 is the major robot car body of the whole project. The dimension of the board is 150x251x7 (Unit: mm). More details of each 3D printed part lists will be below. Servos and axles of back wheels is show in Fig. 10 and Fig. 11. The hole of screw is smaller than actual screw size.
Note: All 3D models is designed in Tinkercad[1]. The Tinkercad 3D model is not accurate. The error range is +/- 0.5mm.

2.1.1 3D Parts List Table

<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Figure</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Single Battery Template</td>
<td>Fig.7</td>
</tr>
<tr>
<td>1</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>2</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>3</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>4</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>5</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>6</td>
<td>5x12 Wire Hole</td>
<td>Fig.8</td>
</tr>
<tr>
<td>7</td>
<td>Arduino Template</td>
<td>Fig.6</td>
</tr>
<tr>
<td>8</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>9</td>
<td>5x12 Wire Hole</td>
<td>Fig.8</td>
</tr>
<tr>
<td>10</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>11</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>12</td>
<td>Breadboard-Mini Template</td>
<td>Fig.4</td>
</tr>
<tr>
<td>13</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>14</td>
<td>4 Battery Template</td>
<td>Fig.5</td>
</tr>
<tr>
<td>15</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>16</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>17</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>18</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>19</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>20</td>
<td>9V Battery Template</td>
<td>Fig.9</td>
</tr>
<tr>
<td>21</td>
<td>5x12 Wire Hole</td>
<td>Fig.8</td>
</tr>
<tr>
<td>22</td>
<td>3mm Screw Hole</td>
<td>Fig.2</td>
</tr>
<tr>
<td>23</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>24</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
<tr>
<td>25</td>
<td>Breadboard-Mini Template</td>
<td>Fig.4</td>
</tr>
<tr>
<td>26</td>
<td>4mm Screw Hole</td>
<td>Fig.3</td>
</tr>
</tbody>
</table>

2.1.2 3D Parts Design Sketch

![Diagram](image)

Fig 3. Big Hole
Fig 3. Big Hole

Fig 4. Breadboard mini template

Fig 5. Four AA Battery template

Fig 6. Arduino template
The Arduino Template allows the Arduino to be placed in this section without moving too much.

Fig 7. Single AA battery template
The AA Battery Template allows the battery holder to be placed in this section without moving too much.
2.1.3 Tires

Please make sure to take the grip off the rear wheels to allow the robot to turn better.
2.2 Circuit

The circuit for the robot consists of Arduino Uno, two Continuous Rotational Servo, 9V battery, 5 AA Batteries, 3 Ultrasonic Sensor, 220Ω resistor, and LEDs. The following is the circuit Schematics for the Obstacle Avoidance robot.

2.3 Programming for the Motion of the Robot

2.3.1 Controlling the Motion of the Car Robot

The robot uses a series of different programs in order to accomplish the various tasks that it needs to accomplish. The robot uses various software functions and sensors in order to detect and avoid objects as well as to move in different directions. This document will explain the various functions that are used to implement the logic of the robot.

First in order to control the motion of the robot two servos were added into the robot and attached to a wheel, giving the robot the ability to move.

The program for the robot has an object called Servo,

```
Servo myServoRight;  // Create servo object to control a servo
Servo myServoLeft;
```

The myServoRight object is used to control the right servo motor, and the myServoLeft object is used to control the left servo motor. Each Servo object is needed in order to implement the speed and direction of each servo motor, changing the speed and direction are done via Servo function calls via the software.

Before doing anything first the pins on the Servo need to be assigned.

```
myServoRight.attach(3);  // attaches the servo on pin 3 to the servo object
myServoLeft.attach(11);  // attaches the servo on pin 11 to the servo object
```

The attach(int n) function assigns a pin from the Servo to the Arduino, this is done so that software has direct access to the Servo Motors via pins 3 and pins 11 on the server. For the letter n in the attach argument can be any arbitrary pin number on the Arduino.
In order to control the motion of the robot the Servo.write(int n) function is used, Servo.write(int n) is an object function for the Servo Object.

According to the Arduino website,

Writes a value to the servo, controlling the shaft accordingly. On a standard servo, this will set the angle of the shaft (in degrees), moving the shaft to that orientation. On a continuous rotation servo, this will set the speed of the servo (with 0 being full-speed in one direction, 180 being full speed in the other, and a value near 90 being no movement).

The following function below void break_car() is used to stop the robot car

```c
void break_car(){
    rightwrite(90);
    leftwrite(90);
}
```

The rightwrite and leftwrite functions calibrate the servos so that 90 will break stop the servo which is standard for a continuous rotation servo. By passing 90 to the rightwrite and leftwrite functions both servos stop and the car breaks.

```c
void go_forward(){
    myServoRight . write (0) ;
    myServoLeft . write (180) ;
}
```

Here the angles are exactly opposite to each other in the motors assigned, myServoRight.write(0) has a parameter value of 0 degrees passed into the Servos Object write function, and myServoLeft.write(180) has a parameter value of 180 degrees passed into the Servos Object write function, these angles are exactly opposite of one another, and because of this both motors will be rotating in the same direction since they are on the opposite side.

```c
void rotate_left(){
    myServoLeft . write (0) ;
    myServoRight . write (0) ;
}
```

Here a value of 0 is passed into the myServoLeft.write(0) myServoRight.write(0) which makes each servo run at maximum speed in the opposite direction causing the robot to rotate left.

```c
void rotate_left(){
    myServoLeft . write (180) ;
    myServoRight . write (180) ;
}
```

Here a value of 180 is passed into the myServoLeft.write(180) myServoRight.write(180) which makes each servo run at maximum speed in the opposite direction causing the robot to rotate right.

```c
void go_backwards(){
    myServoRight . write (180) ;
    myServoLeft . write (0) ;
}
```

Here the angles are exactly opposite to each other in the motors assigned, myServoRight.write(180) has a parameter value of 180 degrees passed into the Servos Object write function, and myServoLeft.write(0) has a parameter value of 0 degrees passed into the Servos Object write function, causing the robot car to go backwards.

### 2.3.2 Controlling the LEDs of the Car Robot

In order to assign the different pins for each LED on the car robot, three int primitive constants are used to keep track of the pin numbers for each LED on the robot.

```c
const int LED_left = 10;
const int LED_center = 7;
const int LED_right = 6;
```
After each primitive int is initialized, the pins on the Arduino are assigned each LED via the pinMode function. The pinMode function calls are used below. Each pinMode function is called in the void setup() function.

```cpp
pinMode(LED_left, OUTPUT);  // This assigns the left led pin number (10) to be output
pinMode(LED_center, OUTPUT); // This assigns the center led pin number (7) to be output
pinMode(LED_right, OUTPUT);  // This assigns the right led pin number (6) to be output
```

The first parameter of the pinMode functions assign a specific pin on the Arduino to the physical hardware being utilized, the second parameter determines whether the pin assigned to the physical hardware is either an INPUT or an OUTPUT. In this case the hardware being used here is an LED, and therefore the second parameter will be OUTPUT for every LED that is assigned here.

### 2.3.3 Controlling the Sensors of the Car Robot

In order to control the sensors of the robot various objects and primitive values were used. The NewPing.h library was used to help control the sensors. The following code creates 3 sensor objects in an array. The first argument is the pin number the trigger pin is connected to, the second argument is the pin number the echo pin is connected to, and the third argument is the max distance the sensor will detect (defined above).

```cpp
NewPing::sonar[SONAR_NUM] = {  
  // Sensor object array.
  // Each sensor’s trigger pin, echo pin, and max distance to ping.
  NewPing(4, 5, MAX_DISTANCE),  // Right sensor 4 is trigger pin 5 is echo pin
  NewPing(8, 9, MAX_DISTANCE),  // Center
  NewPing(12, 13, MAX_DISTANCE) // Left
};
```

The following code sets up the timing of the sensors. This code staggered the sensors so a different sensor will receive input every .033 seconds (stored in PING_INTERVAL).

```cpp
pingTimer[0] = millis() + 75;  // First ping starts at 75ms, gives time for the Arduino to chill before starting.
for (uint8_t i = 1; i < SONAR_NUM; i++) // Set the starting time for each sensor.
  pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;
```

The following code resets the timing of the sensors so that they will keep running on a .1 second cycle (.033 seconds per sensor * 3 sensors = .1 second cycle). When a cycle is finished the oneSensorCycle() function is called and the input from the sensors are processed. The echoCheck() function assigned the input distance to a value stored in the cm[] array.

```cpp
for (uint8_t i = 0; i < SONAR_NUM; i++) {  // Loop through all the sensors.
  if (millis() >= pingTimer[i]) {  // Is it this sensor’s time to ping?
    pingTimer[i] += PING_INTERVAL * SONAR_NUM;  // Set next time this sensor will be pinged.
    if (i == 0 && currentSensor == SONAR_NUM - 1) oneSensorCycle();  // Sensor ping cycle complete, do something with the results.
    sonar[currentSensor].timer_stop();  // Make sure previous timer is canceled before starting a new ping (insurance).
    currentSensor = i;  // Sensor being accessed.
    cm[currentSensor] = 50;  // Make distance zero in case there’s no ping echo for this sensor.
    sonar[currentSensor].ping_timer(echoCheck);  // Do the ping (processing continues, interrupt will call echoCheck to look for echo).
  }
}
```

Since the sensors react on a timer the standard delay(time t) function will cause the sensors to not work properly. The delay_car(int t) function was created to fix this problem. The function delays for t microseconds and after the delay will reset the timing of each sensor. When using sensors this function must be used instead of the delay() function.

```cpp
void delay_car(int t){
  delay(t);
  currentSensor = 0;
}
```
pingTimer[0] = millis() + 75; // First ping starts at 75ms, gives time for the Arduino to chill before starting.
for (uint8_t i = 1; i < SONAR_NUM; i++) // Set the starting time for each sensor.
    pingTimer[i] = pingTimer[i - 1] + PING_INTERVAL;

2.3.4 Flowchart for the robot functions

![Flowchart Image]
3 Parts List

<table>
<thead>
<tr>
<th>Part Name</th>
<th># of Parts</th>
<th>Source</th>
<th>Single Item Price</th>
<th>Total Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensor to avoid obstacle</td>
<td>3</td>
<td>SparkFun</td>
<td>$3.95</td>
<td>$3.95</td>
</tr>
<tr>
<td>3.10&quot; Press Fit Wheel (2 pack)</td>
<td>2</td>
<td>Servocity</td>
<td>$11.98</td>
<td>$11.98</td>
</tr>
<tr>
<td>9 V Battery (Package of 4)</td>
<td>1</td>
<td>SparkFun</td>
<td>$1.95</td>
<td>$1.95</td>
</tr>
<tr>
<td>9 V Battery Holder</td>
<td>1</td>
<td>SparkFun</td>
<td>$2.95</td>
<td>$2.95</td>
</tr>
<tr>
<td>High Speed Continuous Rotation Servo</td>
<td>2</td>
<td>Parallax</td>
<td>$16.99</td>
<td>$33.98</td>
</tr>
<tr>
<td>BreadBoard Mini Modular (White)</td>
<td>1</td>
<td>SparkFun</td>
<td>$3.95</td>
<td>$3.95</td>
</tr>
<tr>
<td>Arduino Uno R3 SMD</td>
<td>1</td>
<td>SparkFun</td>
<td>$29.95</td>
<td>$29.95</td>
</tr>
<tr>
<td>Jumper Wire Kit</td>
<td>1</td>
<td>SparkFun</td>
<td>$6.95</td>
<td>$6.95</td>
</tr>
<tr>
<td>Jumper Wire Premium</td>
<td>1</td>
<td>SparkFun</td>
<td>$1.95</td>
<td>$1.95</td>
</tr>
<tr>
<td>USB Cable A to B</td>
<td>1</td>
<td>SparkFun</td>
<td>$3.95</td>
<td>$3.95</td>
</tr>
<tr>
<td>Electrical Tape</td>
<td>1</td>
<td>Mouser</td>
<td>$5.95</td>
<td>$5.95</td>
</tr>
<tr>
<td>LED</td>
<td>1</td>
<td>SparkFun</td>
<td>$2.95</td>
<td>$2.95</td>
</tr>
<tr>
<td>4 x AA Battery Holder</td>
<td>1</td>
<td>Adafruit</td>
<td>$2.95</td>
<td>$2.95</td>
</tr>
<tr>
<td>Single AA Battery Holder</td>
<td>1</td>
<td>Amazon</td>
<td>$4.46</td>
<td>$4.46</td>
</tr>
<tr>
<td>AA Battery (Package of 8)</td>
<td>1</td>
<td>CVS</td>
<td>$10.29</td>
<td>$10.29</td>
</tr>
<tr>
<td>In-line Power Switch</td>
<td>1</td>
<td>Adafruit</td>
<td>$2.50</td>
<td>$2.50</td>
</tr>
<tr>
<td>Jump Wires Female to Female</td>
<td>1</td>
<td>SparkFun</td>
<td>$4.95</td>
<td>$4.95</td>
</tr>
</tbody>
</table>

4 Future Improvements

This robot can be further improved by modifying parts of the design. With the current design obstacles that are too low to be detected by the sensors will be hit. By attaching the sensors lower the robot will be able to avoid low down obstacles also. The current robot uses an Arduino Uno with only digital pins 2-13 usable (pins 0 and 1 are meant for serial communication). The robot uses 2 pins for each sensor and 1 pin for each sensor LED and 1 pin for each servo for a total of 11 digital pins in use. If more sensors are to be utilized it would be best to upgrade to an Arduino Mega which has pins 22-52 for digital pins. By using more sensors the robot could detect objects behind it. This could make for a more sophisticated collision avoidance algorithm.

5 References

https://www.arduino.cc/en/tutorial/blink
http://playground.arduino.cc/Code/NewPing
https://www.arduino.cc/en/Reference/ServoWrite