

PORLAND STATE UNIVERSITY

EMBEDDED SYSTEMS ON FPGAs

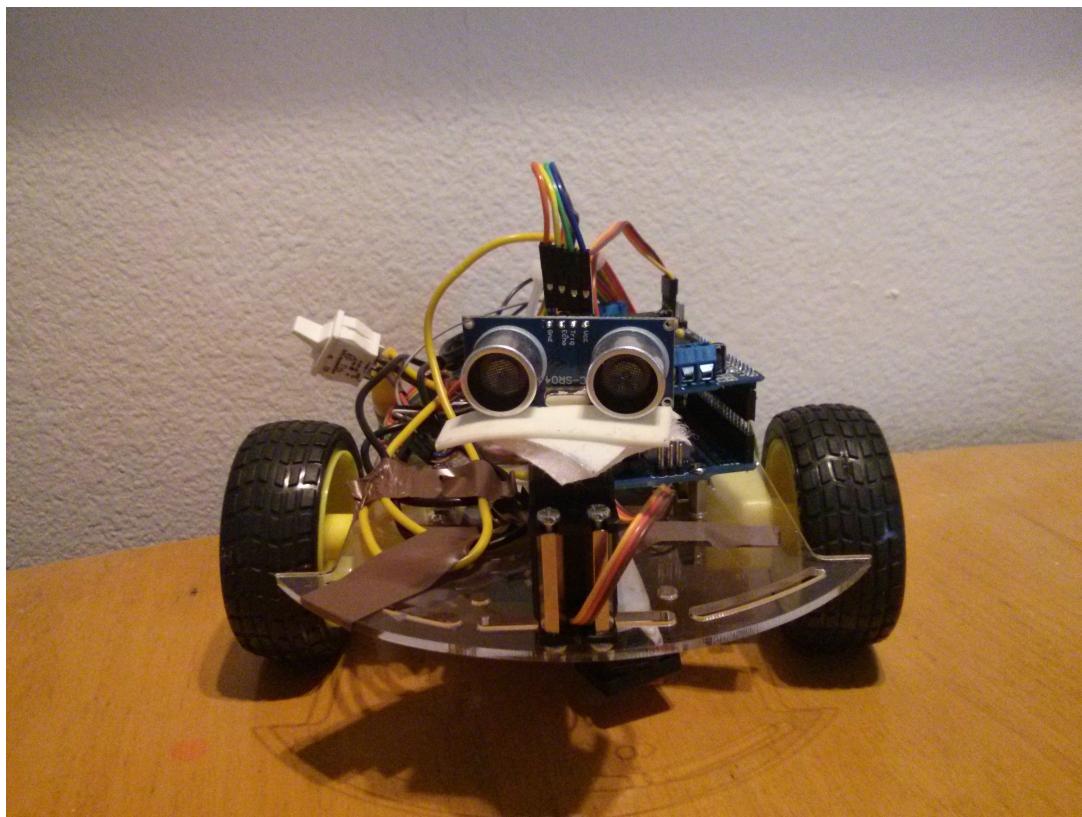
ECE544

Autonomous Robot Car

Erik Rhodes

Caren Zgheib

June 9, 2014



Contents

1	Introduction	3
2	Hardware Components	3
3	Robot Control	3
3.1	Functions	3
3.2	Algorithm	5
4	Conclusion	5
4.1	Challenges	5
4.2	Future Additions	6
5	References	7

1 Introduction

We created an autonomous robot car (SaMMY) that can successfully navigate its way through a given area without hitting any objects . It uses the popular **Arduino** platform and libraries along with a ultrasonic sensor for object detection.

2 Hardware Components

There are many different components needed to successfully and robustly create an autonomous robot. The bulk of our design used a robot kit we ordered online (ref). We also added various parts to make the production time more convenient, including switches, additional batteries, and various cables. The important components can be seen in Figure 1 and are described in Table 1.

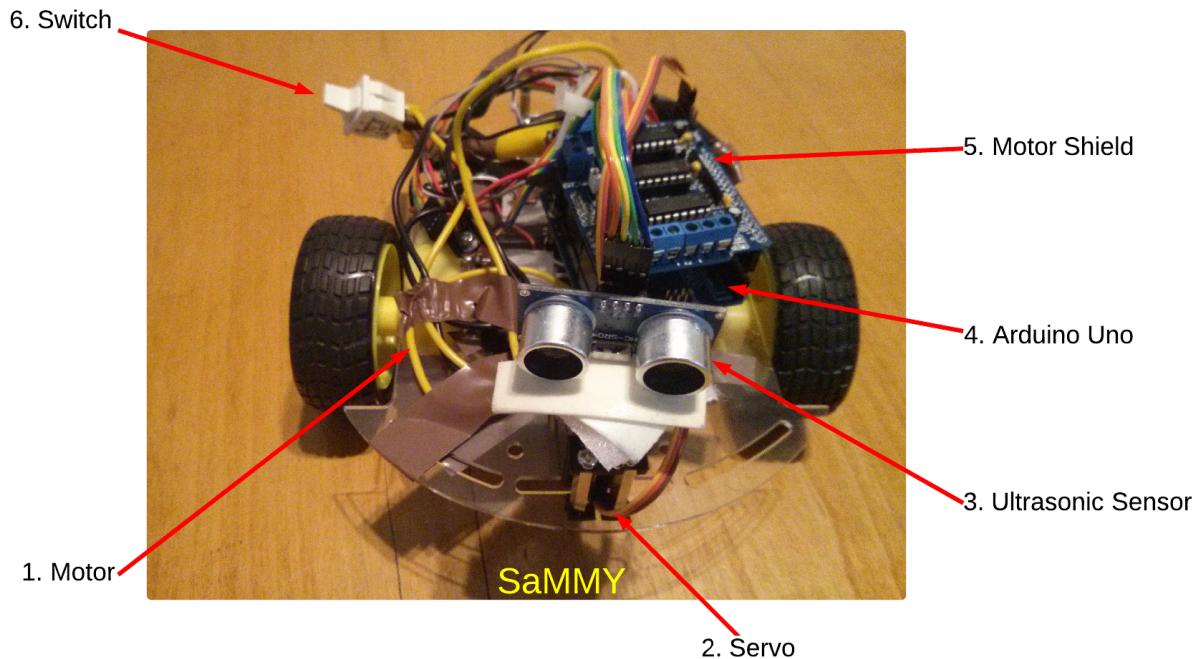


Figure 1: Block Diagram of Control System

Part Name	Function	Details
1. Motor	Drives wheels	N/A
2. RadioShack® Standard Servo	180°rotation	http://shack.net/Ty2VB4
3. Ultrasonic Ranging Module HC-SR04	Object detection	http://bit.ly/1lhqwjI
4. Arduino Uno	Microcontroller Board	http://bit.ly/1gBB1NG
5. L293D Motor Driver Shield	Controls motors	http://bit.ly/1uILISF
6. Switch	On/Off power button	N/A
7. Batteries (not pictured)	9V AA External Power	For Arduino/motors/servo

Table 1: Hardware Components List

Assembling the kit took a fair amount of time. We added a switch, soldered pins onto the motor shield for the sonar sensor, added the motor shield, changed the weight a few different times, and taped the battery holder to ensure the batteries wouldn't get displaced.

3 Robot Control

3.1 Functions

The software portion was written in the **Arduino** IDE, and we modified some of the existing libraries. The main functions for our project are listed in Table 2.

Component	Function	Details
Motor	Drive Forward	Both wheels moving forward at predefined speed
	Drive Backward	Both wheels moving backward at predefined speed
	Turn Left/Right	Both wheels spin for “in-place” turn of 90°
	Veer Left/Right	Only outside wheel turns and vehicle does not stop
	U-Turn	Backs up then turns 180°
	Victory Dance	After certain time, bot does various spins
Sonar + Servo	Look Forward	Scans a 40°range in front while moving forward
	Look Left/Right	Scans 90°left/right to determine next turn

Table 2: Robot Functions

After initializing our components, the software ran an infinite loop which scans for objects and makes decisions on which direction to go. The main control flow is seen in listing 1 below.

Listing 1: Control Loop

```

1 void loop() {
2   if (scanClear()) // If clear to go forward
3   {
4     drive_forward();
5   }
6   else // if path is blocked
7   {
8     freewheel(); // stop
9     lookLeft();
10    int distanceLeft = getAverageDistance();
11
12    lookRight();
13    int distanceRight = getAverageDistance();
14
15    // re-center the ultrasonic
16    servo_position(CENTER);
17
18    // go the least obstructed way
19    if (distanceLeft > distanceRight && distanceLeft > dangerThreshold)
20    {
21      drive_backward();
22      delay(400);
23      rotate_left();
}

```

```

24      }
25      else if (distanceRight > distanceLeft && distanceRight > dangerThreshold)
26      {
27          drive_backward();
28          delay(400);
29          rotate_right();
30      }
31      else // equally blocked or less than danger threshold left or right
32      {
33          freewheel();
34          delay(20);
35          drive_backward();
36          delay(500);
37          u_turn();
38      }
39  }
40 }
41 }
```

3.2 Algorithm

Talk about the specific algorithms, referring to the figures. Mention victory dance

4 Conclusion

Servo and Protecto was quite fun for us to build and was certainly the biggest “crowd pleaser” out of all the projects we’ve made. The assembly perhaps took the least amount of time, although we did face problems with a few of the parts not being built correctly (mainly the battery holder and motor plate mount). Fixing the power problems was the most frustrating part of the project, but once we finished everything, the final product was quite rewarding. We both spent equal times on all the different aspects of this project. Currently, SaMMY performs quite well, even if he does run into walls occasionally. This is caused by veering, which was a tough issue to fix. Listed below are some of the challenges we faced.

4.1 Challenges

- **Power Issues:** After the initial assembly, we were not able to properly move the servo and control the motors. Our initial thought was inadequate power, but the correspondent we communicated with ensured us that it could run perfectly fine on 6V of AA batteries. Even after upping the voltage we saw problems, which were due to the power demands of our servo. After increasing the power to 9V of AA batteries the problem diminished.
- **Motor control:** The motors we received are not precise enough to deliver equal power on both sides. This caused inevitable veering of the robot. We were able to use our own veering function to correct a lot of this, but were not able to control the bot perfectly.
- **CMUcam:** We were able to calibrate and test the CMUcam, which was intended for color detection. However, communicating the frames from the camera to the Arduino proved difficult. Even the example code would not work on our device. We were not able to solve the communication error before the project was due.

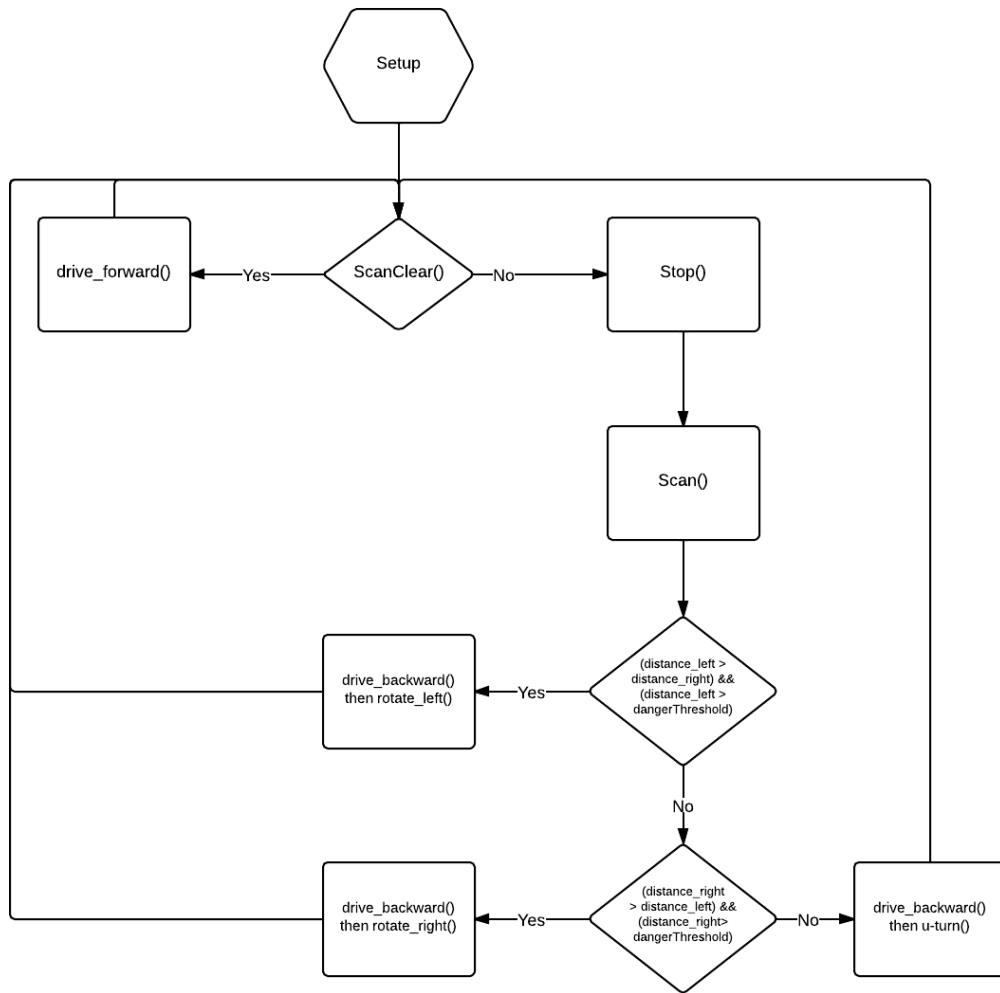


Figure 2: Main Loop Flowchart

- **Erroneous Sonar Readings:** Occasionally the ultrasonic sensor picked up erroneous values that would usually be below 10cm (possibly due to ghost echoes). This caused the robot to stop immediately. We adjusted this by filtering out the first ping that it received below 10cm.

4.2 Future Additions

The Arduino platform and components in this project provide a large amount of functionality. Because of this, the amount of improvements is virtually limitless. Our goals in the future could involve adding:

- **Battery Pack:** Buying a powerful, rechargeable battery pack would save money, improve performance and decrease hassle.
- **External Apparatus:** We initially wanted to attach a simple device that could be triggered based on certain events. While launching or shooting an object would have been

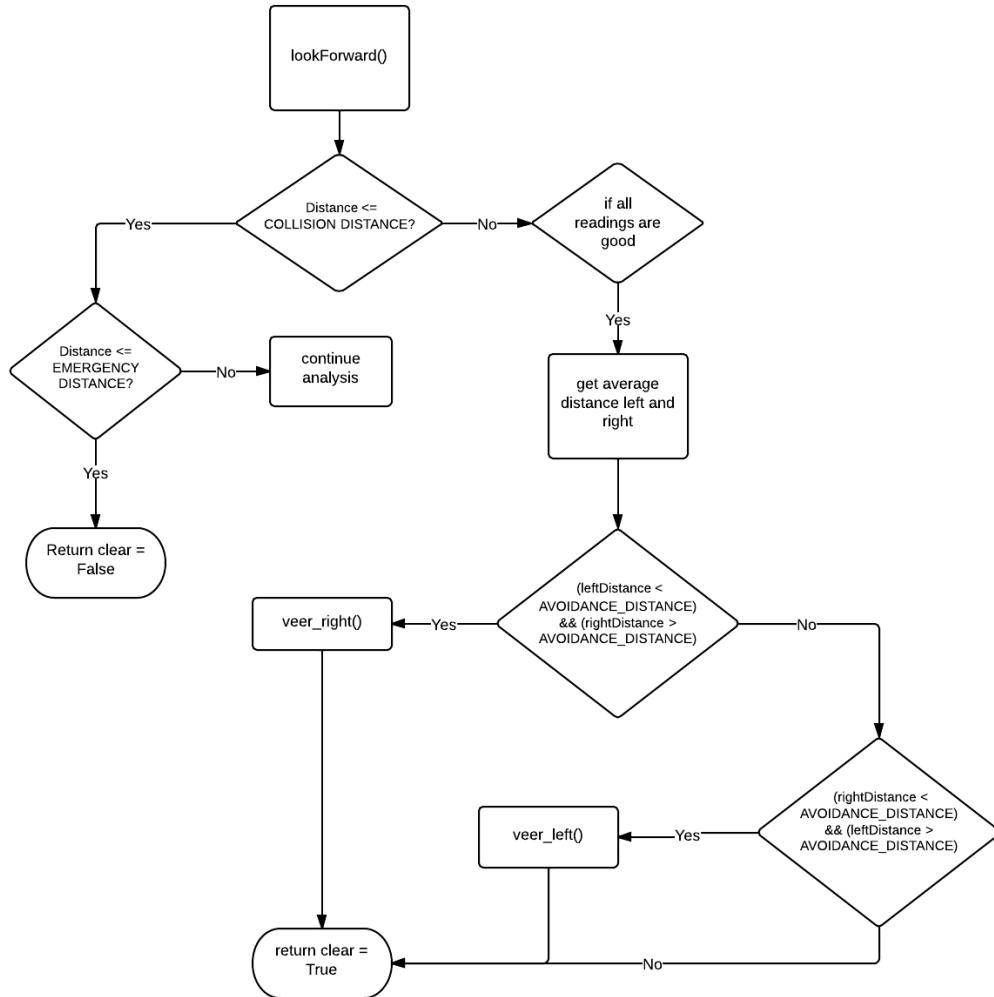


Figure 3: Scan Flowchart

interesting, in most cases this would require PID algorithms, which would involve sufficiently more work and tuning.

- **Color Detection:** One of our original ideas (and part of the reason for the name **Servo and Protecto**) was to use color detection to differentiate between “friends” and “enemies”. There would be certain objects that the robot would purposely avoid, and others that it would just run over.
- **Infrared:** Infrared sensors can be used to detect edges, and they could also be used as an external controller/remote.
- **Object Following:** The sonar and/or color detector could be used to follow an object instead of avoid them. This would be useful when if we attached a physical arm or apparatus of some sort.

5 References

Oddwires Robot Kit: <http://bit.ly/TysmCv> Arduino Libraries Pictures