Bluetooth RC Car

CS122A: Fall 2018

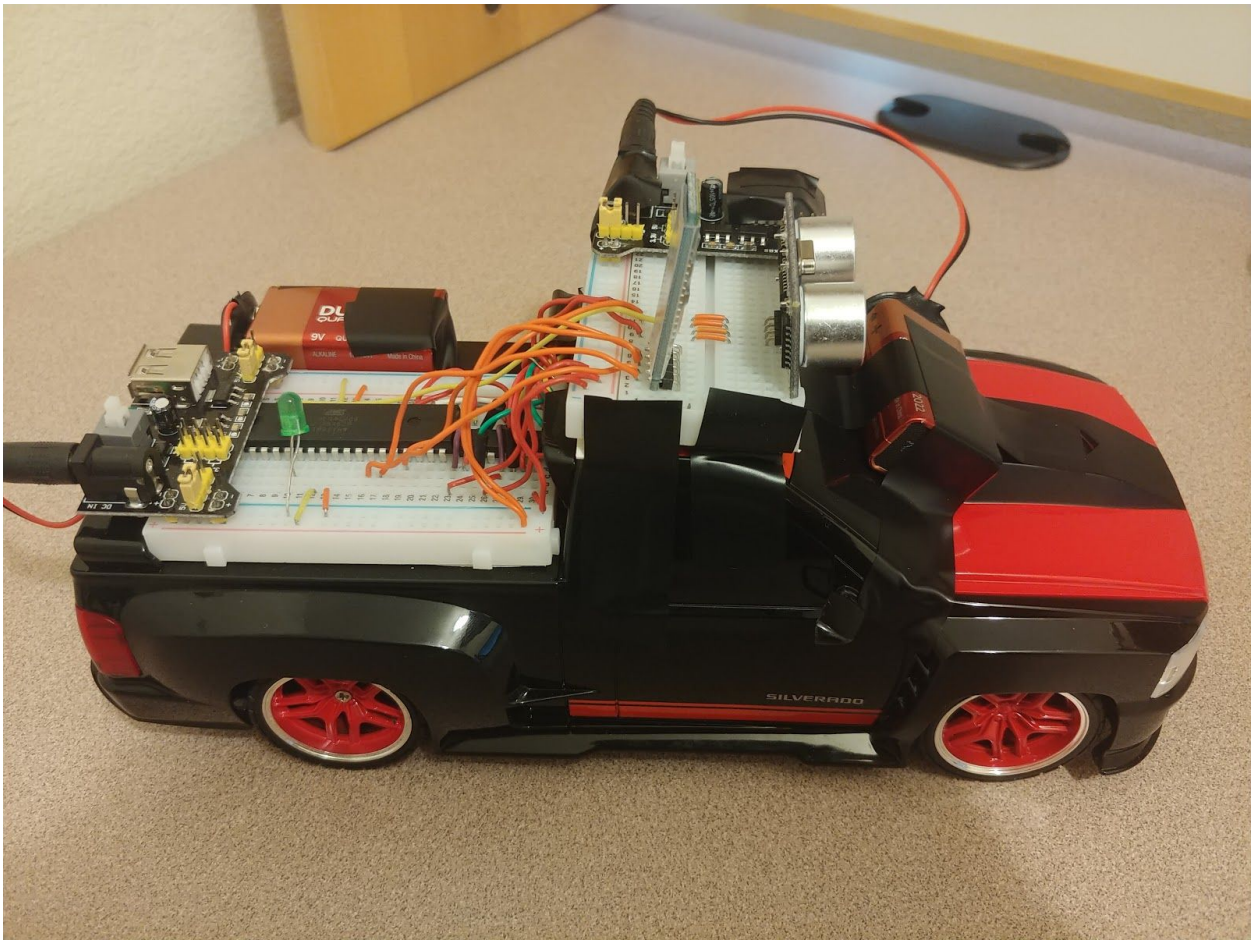
Jose Peralta

Table of Contents

Introduction

This project is an RC car that uses Bluetooth instead of radio to communicate. I bought a regular RC car from Amazon, and took out the PCB that came with the car. I used the DC motors that came with the car, and used the car shell as the place to put the remaining components and peripherals. One of those peripherals is an ultrasonic sensor that detects when an object is close to the front of the car, and stops it so the car does not crash.

I decide to customize an RC car because RC cars are interesting to me. I am also very interested in embedded systems and artificial intelligence, and I wanted to see if this project would be fun for me.While an ultrasonic sensor is not too related to artificial intelligence, it was an interesting concept to experiment with. In the future, I hope to work with self driving cars.



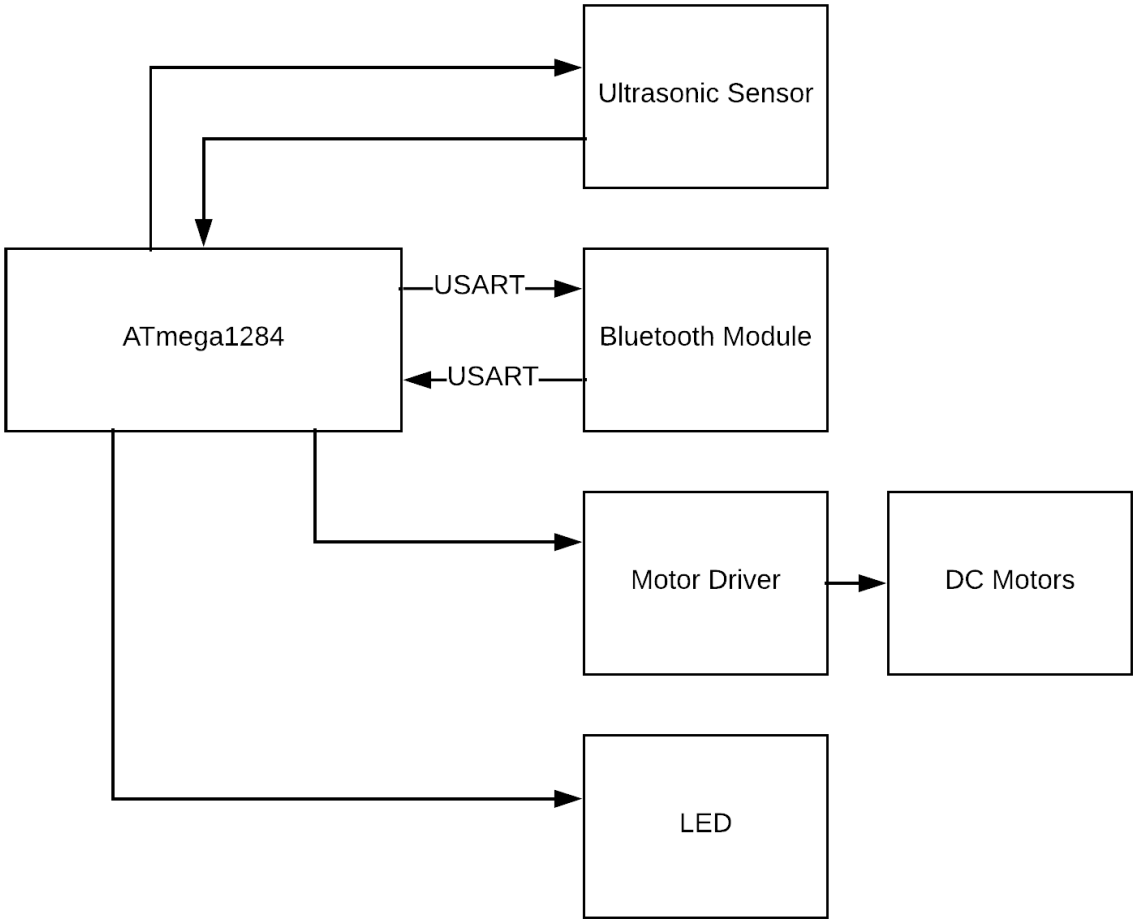
Hardware

Parts List

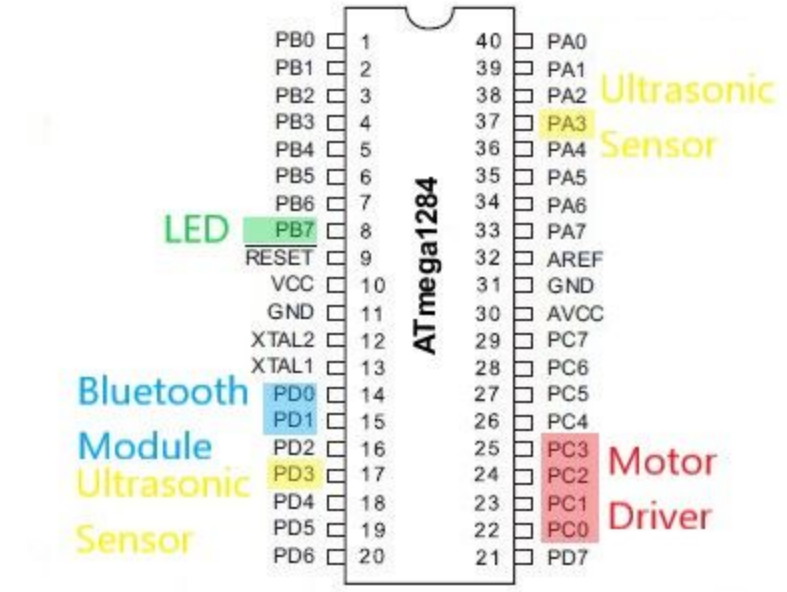
The hardware that was **used** in this project is listed below. The equipment that was not taught in this course has been bolded.

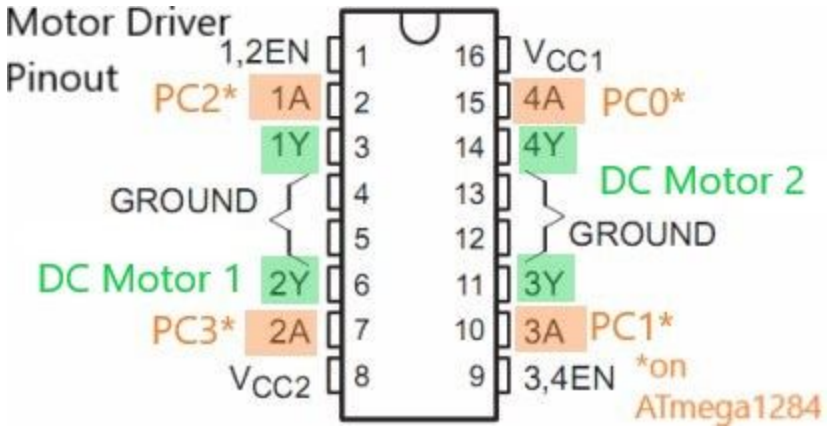
Part	Part #	Quantity	Price (optional)
ATMega1284	ATMega1284	1	$5
Bluetooth Module	HC-05	1	$11.11
Ultrasonic Sensor	HC-SR04	1	$3.95
Motor Driver	L293D	1	$6.15
Brushed DC Motor	Tamiya 9801112M Mabuchi FA-130	2	$1.99 (each)
5V Power Supply		2	$5 (each)
9V Battery	9V Battery	2	~$3 (each)
Male DC Power Plug For 9V Battery	Male DC Power Plug For 9V Battery	2	$5.88
Solderless Breadboard	BB400	2	$5.90
Wires	Wires	~35	
LED	LED	1	
		Total	60.97

Block Diagram



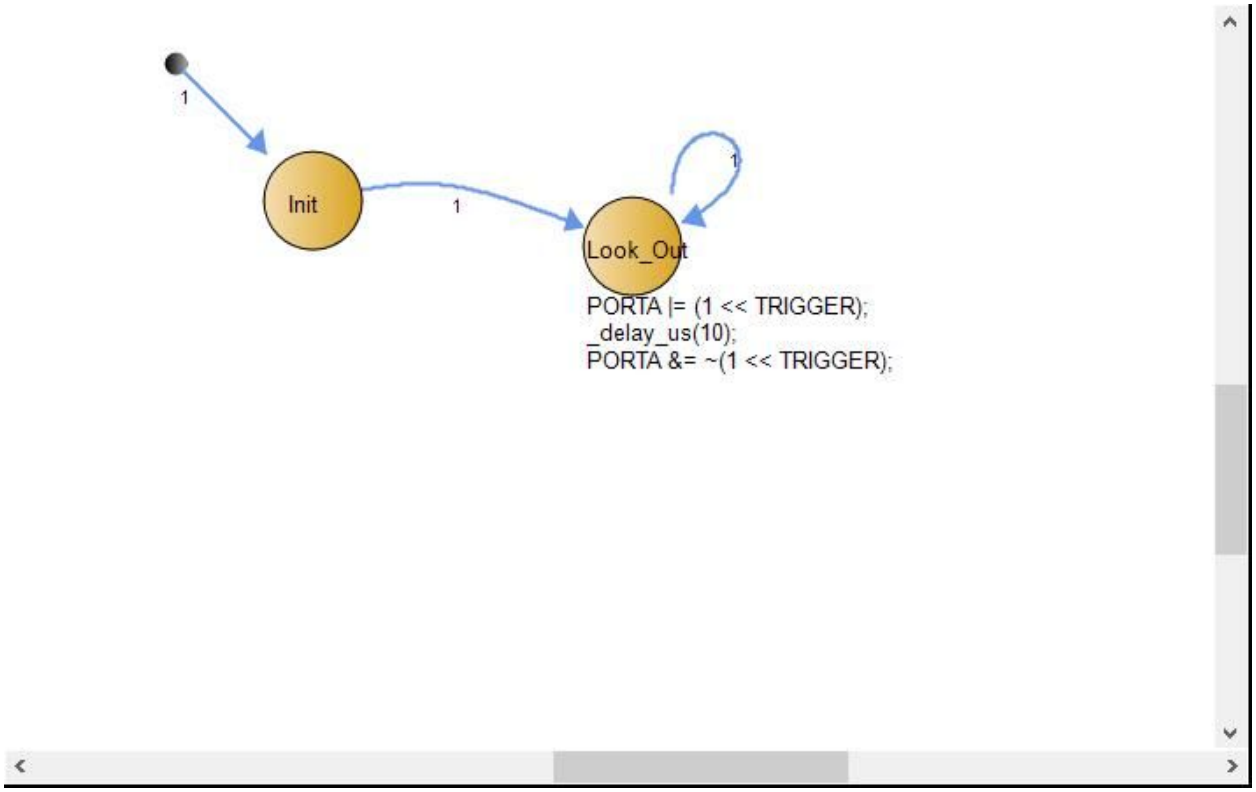
Pinout

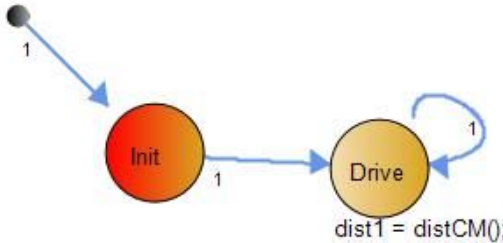




Software

The software designed for this project was implemented using the PES standard. The overall design as a task diagram is included below.


1
Init
1
1
Look_Out
PORTA |= (1 << TRIGGER);
_delay_us(10);
PORTA &= ~(1 << TRIGGER);




```
dist1 = distCM();
dist2 = distCM();
dist3 = distCM();

dist = (dist1 + dist2 + dist3) / 3;
if(dist <= 50) {
    halt = 1; }
else {
     halt = 0;}
var = USART_Receive(0);
if(var == 'L') { // left
    USART_Flush(0);
    PORTC = 0x01;
}
else if (var == 'R') { // right
    USART_Flush(0);
    PORTC = 0x02;
}
else if (var == 'F') { //forward
```

```
}
else if (var == 'F') { //forward
    USART_Flush(0);
     if(halt == 1) {
          PORTC =  0x00;
      }
      else {
          PORTC = 0x04;
      }

}
else if (var == 'B') { // back
    USART_Flush(0);
    PORTC = 0x08;
}
else if (var == 'H') { //back left
    USART_Flush(0);
    PORTC = 0x09;
}
else if (var == 'G') { //forward left
USART_Flush(0);
    if(halt == 1) {
        PORTC =  0x00;
    }
    else {
        PORTC = 0x05;
    }
```

```
else if (var == 'B') { // back
    USART_Flush(0);
    PORTC = 0x08;
}
else if (var == 'H') { //back left
    USART_Flush(0);
    PORTC = 0x09;
}
else if (var == 'G') { //forward left
USART_Flush(0);
    if(halt == 1) {
        PORTC =  0x00;
    }
    else {
        PORTC = 0x05;
    }
}
else if (var == 'I') {//foward right
    USART_Flush(0);
    if(halt == 1) {
        PORTC = 0x00;
     }
     else {
          PORTC = 0x06;
     }
}
```

```
else if (var == 'G') { //forward left
USART_Flush(0);
    if(halt == 1) {
        PORTC =  0x00;
    }
    else {
        PORTC = 0x05;
    }
}
else if (var == 'I') {//foward right
    USART_Flush(0);
    if(halt == 1) {
        PORTC = 0x00;
     }
     else {
          PORTC = 0x06;
     }
}
else if (var == 'J') { //back right
    USART_Flush(0);
    PORTC = 0x0A;
}
else{
     USART_Flush(0);
     PORTC = 0x00;
}
```

The Ultrasonic Task (shown first) simply polls the ultrasonic sensor every 50ms by sending a 10 microsecond pulse wave and reading the input.

The Drive Task (remaining images) control the drive actions based on the output of the ultrasonic sensor.

Implementation Reflection

I had fun and I learned a lot from this project. I was not very content with my progress at first, because it took me awhile to get the easiest thing working: communication via Bluetooth and USART from the Atmega to my phone. However, once I got over that obstacle, I had a better idea of what I was doing. I think 5 weeks was plenty of time to get this project done, especially since I lost a couple of days because my parts did not come in on time.

If I were to do this project again, I would make sure to start earlier, and plan out my supplies better. I had to rely on inadequate tools to strip some wires and solder them, until I decided to go out and buy a solder kit and wire strippers. My parts came 3 days late, so I would make sure I either express shipped them or ordered them earlier. Despite all of the road bumps, I am personally content with my project. My favorite part about it is that the ultrasonic sensor works, despite some bugs. That sensor was the most difficult part of my project: I spent the last half of my time trying to figure out how to implement it. It was such a relief when I finally got it working.

Milestone

My first milestone was to make the car communicate with the phone, and make it drive forward. I was able to successfully meet this milestone, and I believe this was a perfect point to break, because the next milestone was to make the car communicate with the ultrasonic sensor. Once forward drive was implemented, the remaining directions were simple.

My second milestone, as stated above, was to make the car communicate with the ultrasonic sensor and make the car stop at a given distance from an object. The car should've also be able to turn in every direction as well. I also successfully met this milestone. At this point, my project was done for the most part, and it was time for debugging, so this milestone was just just right.

Completed components

- All components were successfully wired and coded: ATmega1284, Bluetooth module, motor driver, DC motors, and ultrasonic sensor.

Incomplete components

- None

Youtube Links

- Short video:
    - https://youtu.be/3PQHZkG9LJ8
- Longer video:
    - https://youtu.be/wSfDBKevmIU

Testing

Bluetooth Module/USART

Upon programming the Atmega1284, I download an app and paired my phone using Bluetooth. I sent characters to the Atmega, and checked the correctness of the characters by blinking LEDs if the characters sent matched the characters received. I had two classmates check this part.

Test cases:

- Sent ‘a’, received ‘a’, LED lights up
- Sent ‘b’, received ‘b’, LED lights up
- Sent ‘z’, received ‘c’, LED lights up
- Sent ‘10’, received ‘10’, LED lights up

DC Motors/Motor Driver

After snipping the wires from the supplied PCB, I soldered an extension to the existing wires so I could plug them in to the breadboard. Afterwards, I plugged them into the motor driver, and downloaded the joystick app that I used for the remainder of my project. I tested forward, backward, left, right, forward-left, forward-right, back-left, and back-right. I gave this semi complete project to two classmates and two non-engineering friends to test. It worked successfully.

Test cases:

- Sent forward, car goes forward
- Sent backward, car goes backward
- Sent left, car turns left
- Sent right, car turns right
- Sent forward-left, car goes forward-left
- Sent forward-right, car goes forward-right
- Sent backward-left, car goes backward-left
- Sent backward-right, car goes backward-right

Ultrasonic Sensor

After successfully driving the car in all directions, it was time for the ultrasonic sensor. This was the most difficult part of the project. In order to test it, I kept on setting different distances to find out which ones were safe, and used an LED to indicate when an object was in “dangerous range”. The car would then stop moving forward as soon as the object was detected. I had 3 classmates and 3 friends outside of class test my fully complete project

Test cases:

- Object is 5 cm in front of car, car does not stop quick enough and crashes
- Object is 10 cm in front of car, car crashes
- Object is 20 cm in front of car, car nearly crashes
- Object is 30 cm in front of car, car stops at an appropriate time. Car does not crash.

Known Bugs

- Car does not stop when going full speed, even if object is detected
    - Cause: DC motors do not stop spinning fast enough for car to stop
    - Attempted solution: I have increased the stopping distance, but that makes it so that when the car is going half speed, it stops really far away from the object.
    - Next possible solution: read the speed of the DC motor in some way, maybe by calculating the RPMs, and based on the speed (half vs full), change the stopping distance. For half speed, the stopping distance will be smaller. For full speed, the stopping distance will be greater.
    - This bug cannot be turned into a feature. In a real world scenario, this would crash the car.
- Left/right turns are not as sharp as they should be
    - Possible cause #1: I am not correctly sending the turn signals to the DC motor.
    - Possible cause #2: Early on, I disassembled the gearbox because I was curious to look at the steering mechanism. I may have incorrectly lined up the gears upon reassembly
    - Attempted solution: there is a left/right adjust handle at the bottom of the car. I have attempted to set it at different degrees, but it does not help
    - Next possible solution: reexamine the gearbox and see if any gears/pieces are not lined up properly.
    - This bug cannot be turned into a feature.
- When driving in a combination of left/right/forward/backward, car slows down significantly
    - Cause: not enough power is being delivered to the two DC motors

- Attempted solution: bought an extra 5V power supply and connected the DC motors' VCC and GND to that power supply only. Nothing else is being powered by that power supply. Same issue is still happening.
- Next possible solution: buy a voltage regulator with a different rating, so that I can slowly increase the voltage provided to the motors. The original car provided 6V to the motors, whereas this power supply only gives 5V
- This bug cannot be turned into a feature.

Resume/Curriculum Vitae (CV) Blurb

'Crash Me If You Can' is a bluetooth RC car designed to stop whenever it detects an obstruction in its path. It will stop at a safe distance and will not allow the user to move forward until the object is no longer detected. The car is powered by an ATmega1284, which communicates via USART to a Bluetooth module which is connected to a joystick app on a phone. The ultrasonic sensor uses timer interrupts to receive information about obstacles every 10 microseconds, in order to properly detect when objects are present. The DC motors controlling the car are driven by a motor driver, which is also connected to the ATmega1284.

Future work

The next feature I would add to this project would be PIR motion sensors at key points in the front of the car. Currently, the ultrasonic sensor sits on top of the car and is able to detect objects only within its viewing angle. It does not detect objects that are lower to the ground. I would attach at least one PIR motion sensor at the front center of the car, and use it as an additional peripheral to detect obstructions.

I could also create a custom PCB to reduce the size of my project. This car originally came with a PCB built in, I could design a similar one and place mine in the same spot as the previous one. More than size, it would help a lot with weight. All of the components are particularly heavy. The car also came with a premade battery holder socket, so I could also take out the two 9V batteries and replace them with the six 1.5V batteries it was originally designed to use.

This project was kind of similar to the project I proposed in my Company Research report in the fact that I am using sensors to stop a moving vehicle from crashing into an object. I am not entirely sure if this is along the lines that Pacific Railway works on Positive Train Control, but I believe it would be in my best interest to show them this project and express that I am interested in Positive Train Control.

Extra Credit

- Did not attempt

References

- Mr. Jonathan Woolf was a great help when I was learning how to use the ultrasonic sensor. He gave me some pointers, and I could not have done this without his help.
- https://www.instructables.com/id/How-to-use-the-L293D-Motor-Driver-Arduino-Tutorial/
    - I drew inspiration from the wiring diagram when wiring my motor driver.

Appendix

All state machines have been included in the software portion above because they are all relevant. No other material is included.