Drunk Ride

Autonomous Automobile

Jonathan Palgon
Report Date: 4/23/02

University of Florida
Department of Electrical and Computer Engineering
EEL5666
Intelligent Machine Design Laboratory

Instructor: A. A. Arroyo
TA: TaeHoon Choi, Aamir Qaiyumi, Uriel Rodriguez

Email: jnplgn@ufl.edu

Table of Contents

Abstract

Everywhere today, people are dying at alarming rates due to drunk drivers. The alcohol impairs the decision making abilities of inebriated drivers leading to gory automobile accidents. If these drunken citizens would wait to sober up or ride with a designated driver, the roads would be a safer place. However, many of these people feel they can drive perfectly well while inebriated.

Another option to make the roads a safer place to be would be an automobile that would safely navigate city streets to the intoxicated's final destination (home). This project attempts to make the city streets safe again by examining the possibility of an autonomous automobile. A small scale model, "Drunk Ride", is designed using a microcontroller as the "brain," many different sensors as the "eyes," and motors as the "muscles." This project could be the foundation for a large scale autonomous vehicle.

Executive Summary

After several courses with Professor Schwartz, I had heard his story of the autonomous car many times. Since then, I have always wanted to make an autonomous automobile. Because I live in the college world where everybody wants to get drunk and nobody wants to be designated driver, I came up with an idea for my robot: a "Drunk Ride" robot. The car would drive inebriated passengers through the city streets to a home.

Unfortunately, I am not well funded and therefore was not able to build a full scale model. Due to lack of cash flow, I decided to find a small microcontroller board that would fit on a tiny car that could roam the streets of a relatively small (cheap) city. I built the city streets using cardboard for the foundation, black construction paper in place of asphalt, and white masking tape for lanes. The home is a sonar transmitting beacon.

I used some of the features of the "Trans Am" robot by Jason Grzywna as a guide. The automobile is able to stay within a lane, avoid obstacles (other cars, pedestrians, etc.), react to impacts, and steer the vehicle safely to the sonar transmitting destination. I used one servo as a drive motor, infrared detectors for obstacle avoidance, bump sensors for impact reaction, CdS sensors for locating the lanes, and a sonar receiver for locating "home." I borrowed additional ideas and implementations from other previous projects such as Michael Apodaca's and Megan Grimm's sonar circuits.

Introduction

The University of Florida is always in the top 5 party school in the nation. Yet, the title of the "Number 1 Party School in the Nation" seems to elude us. I want to see the University succeed in every aspect possible, and being the top party school is no exception. I feel that we could easily achieve this task if we did not have designated drivers. Designated drivers are necessary to keep the streets safe. However, what if there was no need for designated drivers? My robot is meant to be a safe alternative to drunk driving without the necessity for a designated driver. The automobile is able to stay within a lane, avoid obstacles (other cars, pedestrians, etc.), react to impacts, and steer the vehicle safely to the sonar transmitting destination.

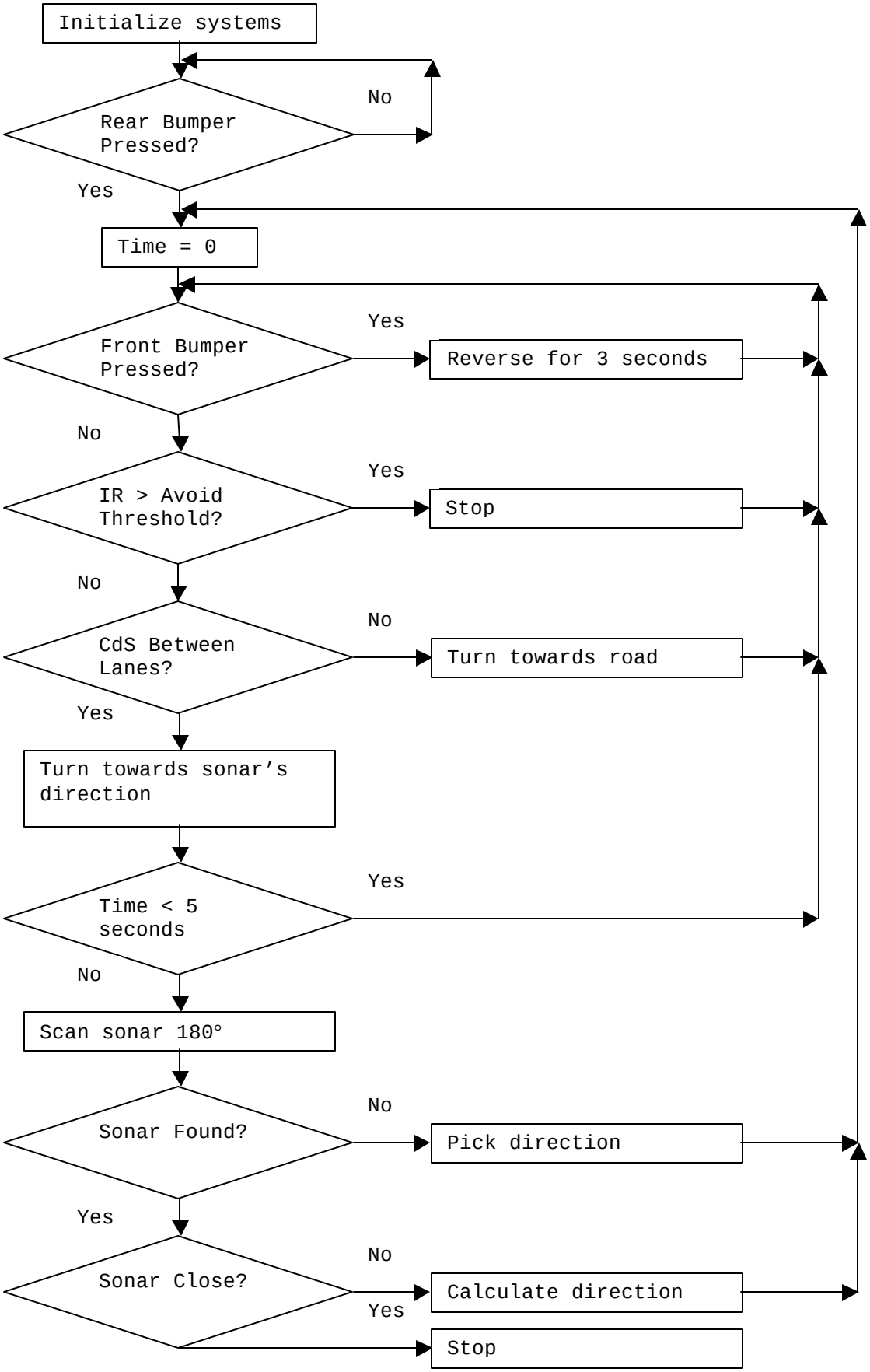
This paper covers the steps involved in the design, assembly, and programming of "Drunk Ride." In the sections that follow, I will touch on the integrated system, platform design, actuation, sensors, expected behaviors, and the experimental layout.

Integrated System

The "brain," the microcontroller, is responsible for interpreting the input from the sensors and controlling the motors in reaction to the input. The smallest microcontroller I could find that fit my needs was the TJPRO11 microcontroller from Mekatronix. I needed at least 7 analog channels (for sensor measurements) as well as 2 8-bit digital output ports (for IR LEDs and feedback) and 3 output compare pins (for motor control). The way the different components interface with the different ports on the microcontroller is modeled in the table below.

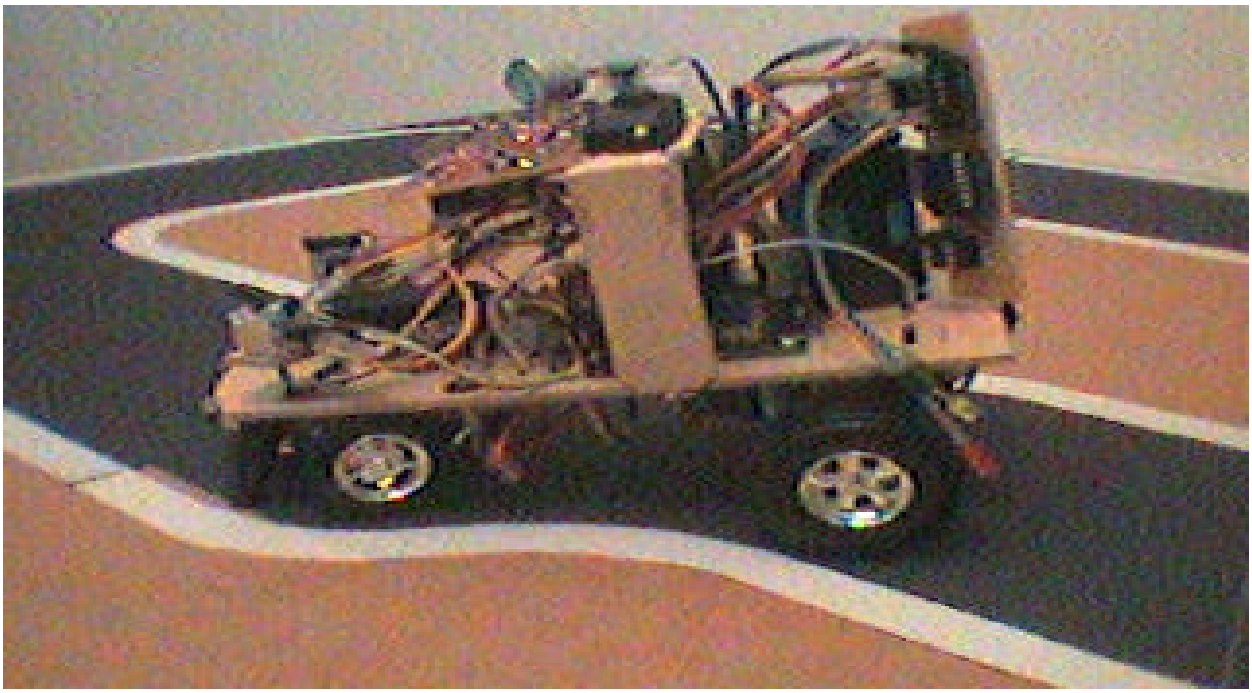
Analog Port		Digital Port		Digital Output Port	
Port	Connection	Port	Connection	Port	Connection
PE0	Bumper Network	PA0	Unused	0x4000	Feeback
PE1	Sonar Receiver	PA1	Unused	0x8000	40kHz IR LED
PE2	Right IR Receiver	PA2	Unused		
PE3	Left IR Receiver	PA3	Drive "Motor"		
PE4	Right CdS	PA4	Steering Servo		
PE5	Center CdS	PA5	Sonar Servo		
PE6	Unused	PA6	Unused		
PE7	Left CdS	PA7	Unused		

The flowchart on the following page shows the basics of the program. The program continuously scans the sensors and puts the motors into action based on the sensors' readings. The order the sensors are scanned is in order of safety importance. When the vehicle is close enough to home, the vehicle stops and shuts off, allowing the drunken passengers to stumble out of the vehicle.


Initialize systems
Rear Bumper Pressed?
No
Yes
Time = 0
Front Bumper Pressed?
Yes
Reverse for 3 seconds
No
IR > Avoid Threshold?
Yes
Stop
No
CdS Between Lanes?
No
Turn towards road
Yes
Turn towards sonar's direction
Time < 5 seconds
Yes
No
Scan sonar 180°
Sonar Found?
No
Pick direction
Yes
Sonar Close?
No
Calculate direction
Yes
Stop

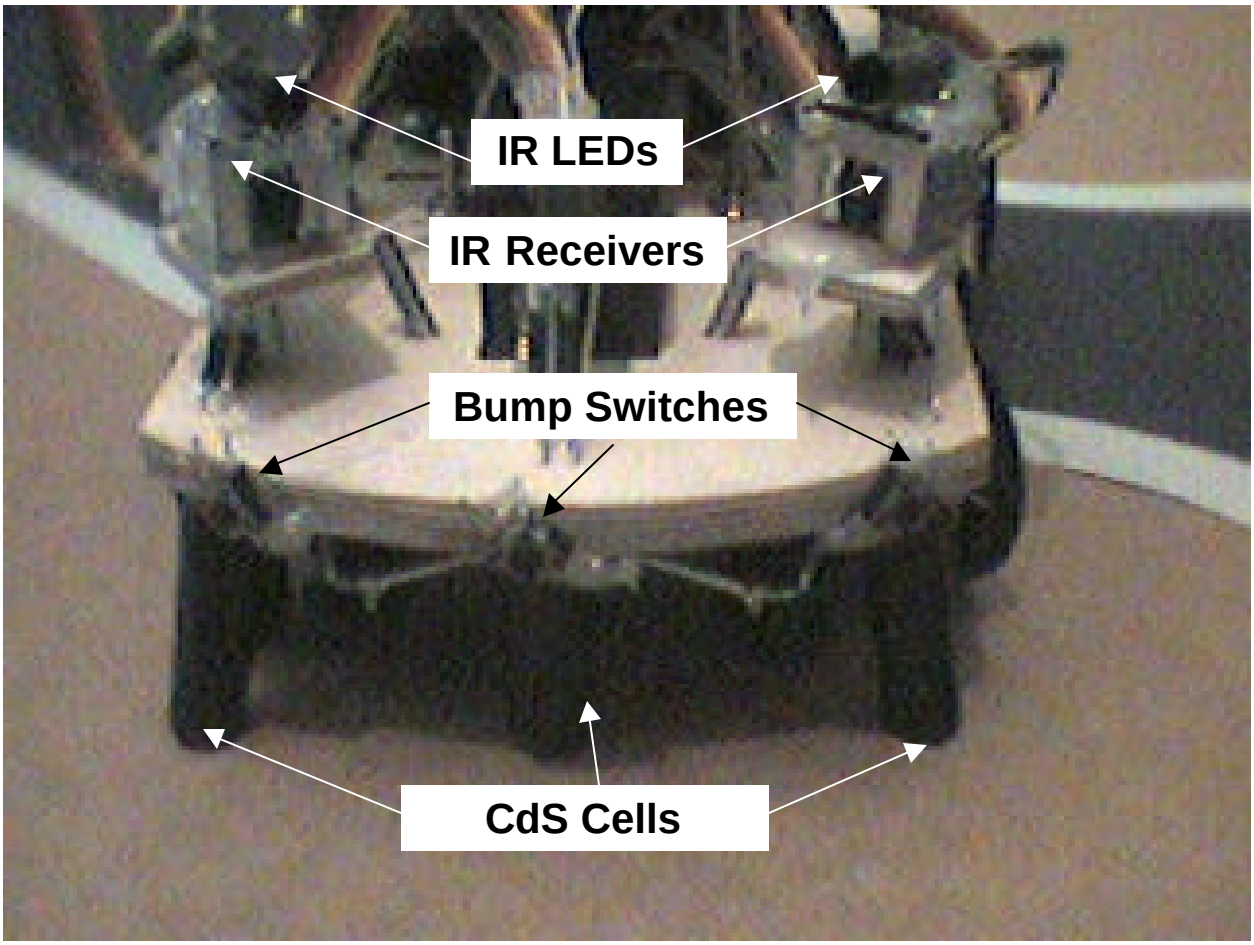
Mobile Platform

Because the two wheeled automobile is not an accurate model for a real world autonomous automobile, I chose to use a four wheel design. The chassis houses all motors and sensors in a compact manner, small enough to navigate a model city. Below is a picture of the final assembled platform.

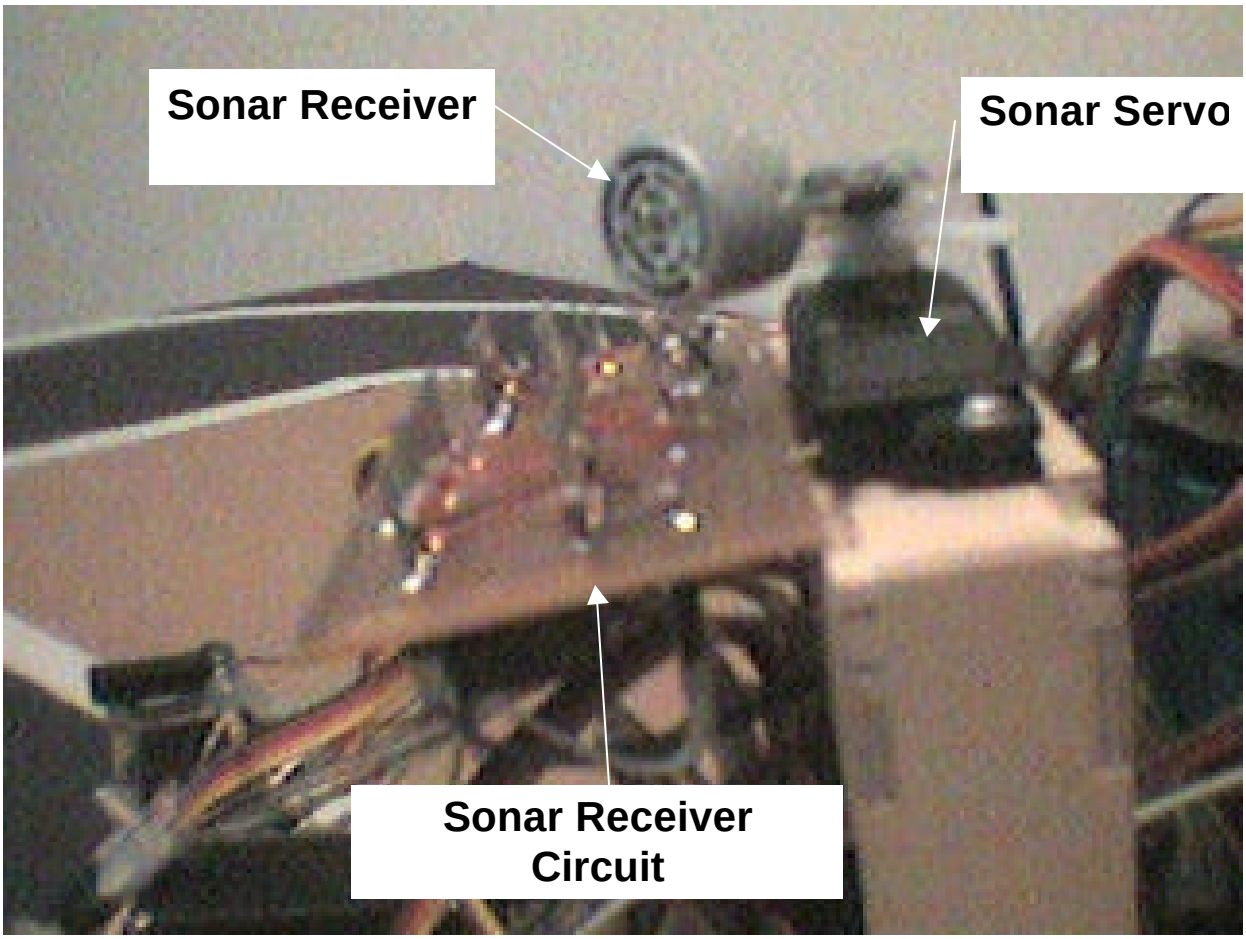


The platform was designed around the microcontroller, which is located on the center of the main platform.

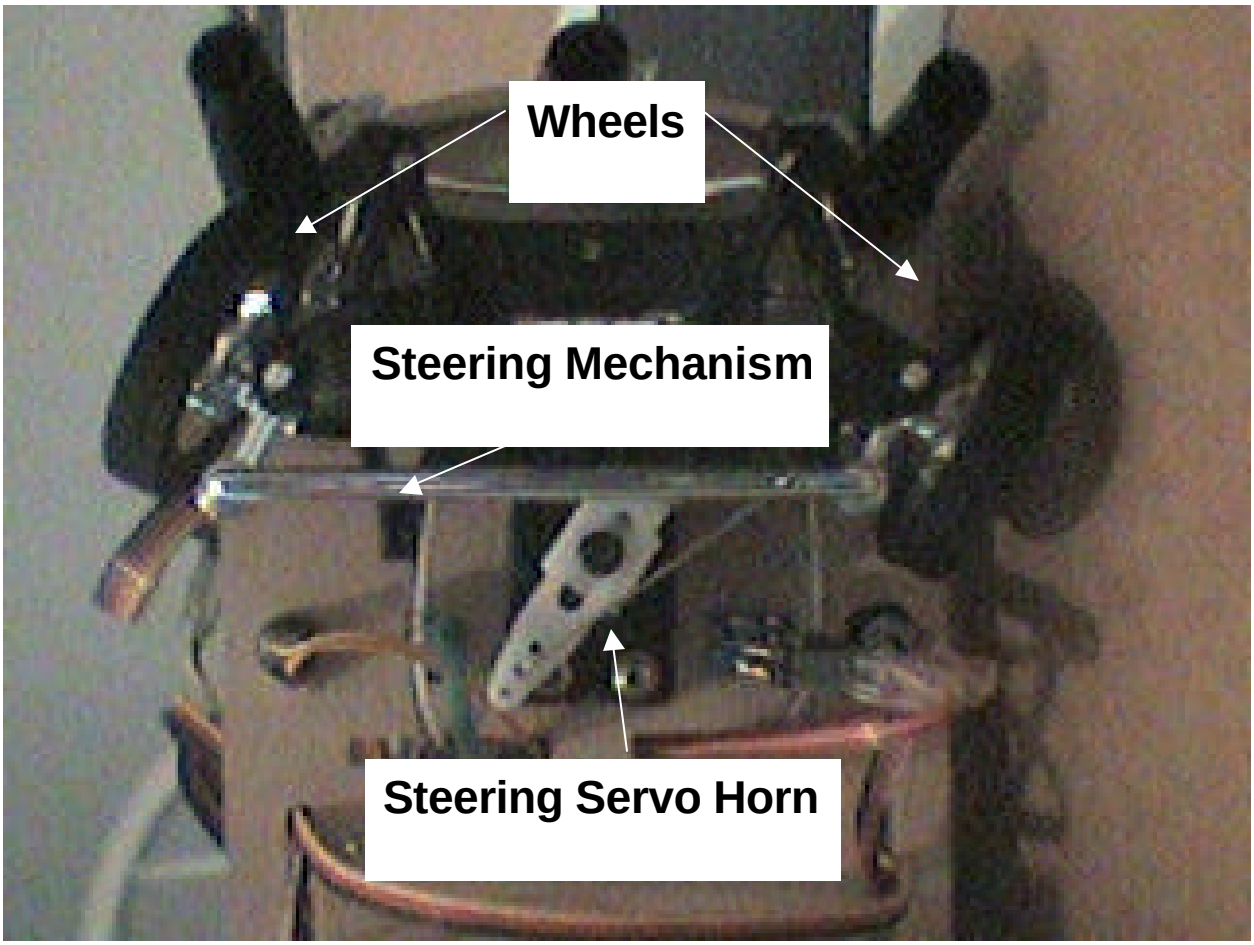
The picture on the following page shows the placement of the CdS cells, IR LEDs, IR receivers, and bump switches. The CdS cells are placed underneath the front of the vehicle, the IR receivers and LEDs are located above the front of the vehicle, and the bump switches are located on the rim of the front of the vehicle.



The picture on the following page shows the placement of the sonar receiver and the sonar receiver circuit. The sonar receiver and its circuit are located on a platform which is raised from the main platform in the center of the vehicle.



The picture on the following page shows the placement of the steering servo and the steering mechanism. The steering servo and steering mechanism are located beneath the front of the vehicle. After a failed attempt to make a steering mechanism using balsa wood and pins, I used the steering mechanism from a model ’57 Chevy. A modified sewing pin connects the steering mechanism to the servo.

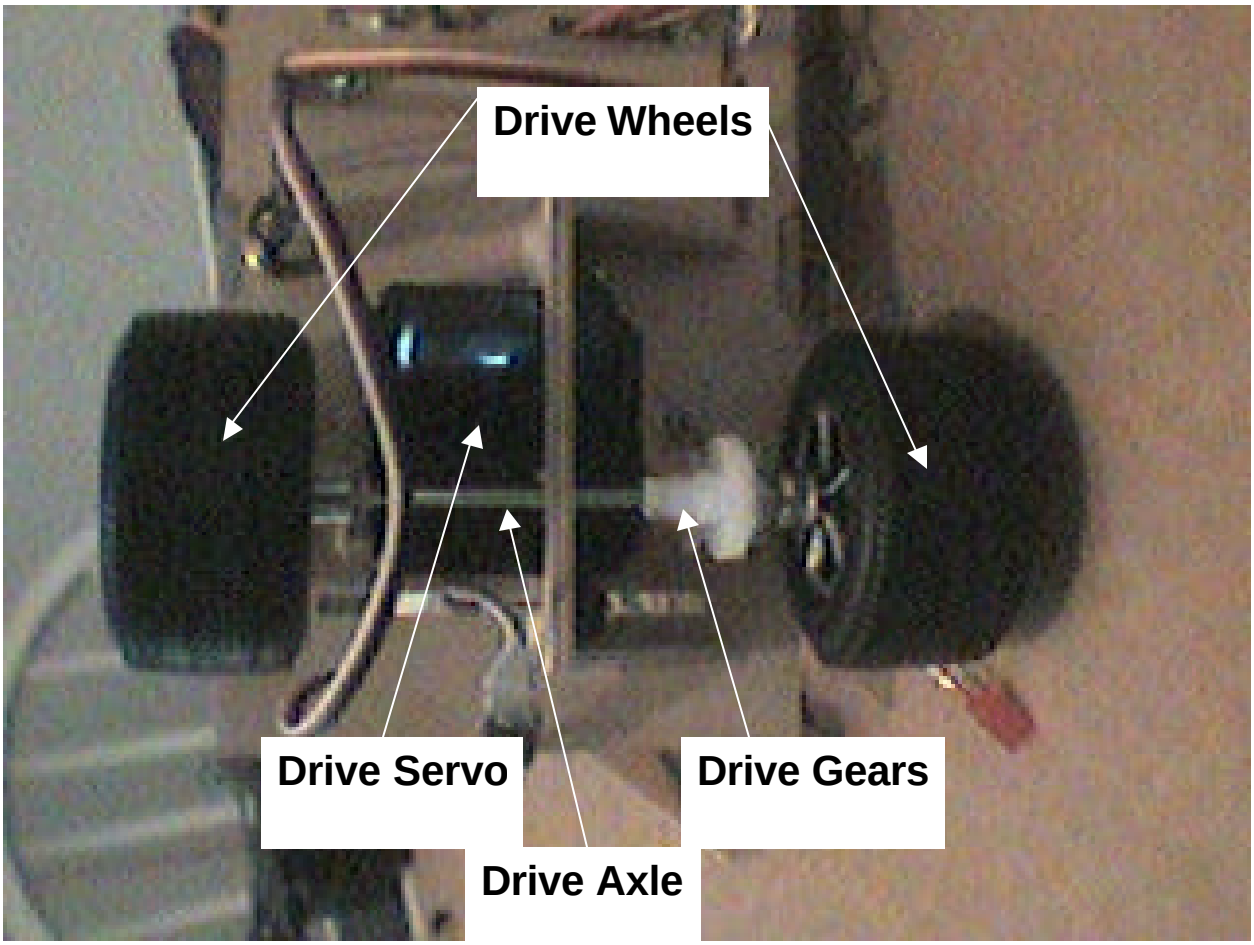


The picture on the following page shows the placement of the drive servo and the drive axle. The drive servo and drive axle are located beneath the rear of the vehicle. My initial platform design included two steering servos to enable a tighter turning radius. This idea failed in two areas:

1. It was rather difficult to make the vehicle travel rather straight.

2. A real car does not have two different drive motors.

For the final platform, I decided on using a single steering servo. This servo would spin a gear, which would be attached to the drive axle (stolen from the same '57 Chevy). This in turn would spin the wheels attached to the drive axle allowing the vehicle to move forward or backward.


Drive Wheels
Drive Servo
Drive Gears
Drive Axle

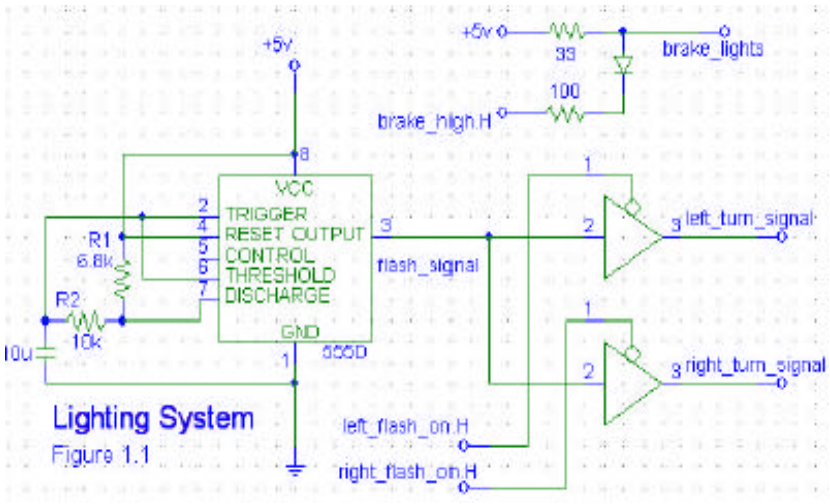
Actuation

After the brain makes a "decision" on which direction and at what speed the vehicle should go, the drive motor and steering servo activate. Borrowing from Jason Grzywna's design, the drive motor was made from a "hacked" servo, and the steering motor was made from a regular servo. The drive motor can move forward and reverse at several different speeds, as well as not move at all. The servos are the main mechanism for getting the vehicle from point A (party) to point B (home).

Note: The servo hack was performed as documented in the "Manuals" section of the www.mekatronix.com website.

Standard servos are controlled using a pulse width modulation (PWM) signal. This tells the servo which direction, how far, and at which speed the servo should turn. The modified servo has been altered so that it never knows when it reaches the location it is instructed to find, but it can distinguish the speed and direction at which to go.

Also borrowing from Jason Grzywna's design, brake lights and turning signals are used for feedback. When the vehicle is stopped the brake lights are on. If the vehicle has detected sonar, the direction signal is on in the direction the vehicle is trying to turn. If the sonar does not detect "home", the hazard signals are turned on. The circuit on the following page is used to implement the rear lights.



Feedback Lighting System

Courtesy Jason Grzywna's Report

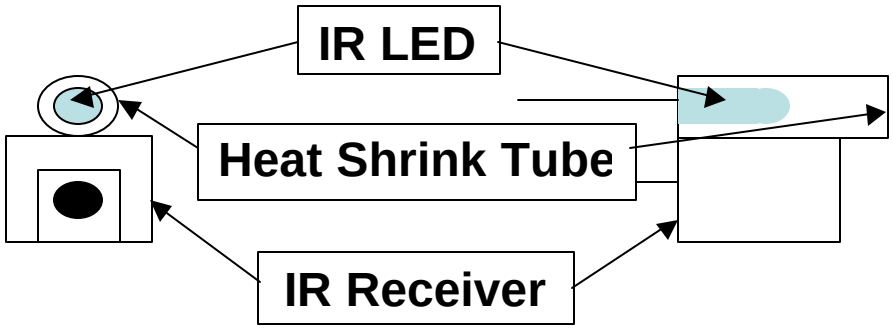
Sensors

Infrared System

The vehicle uses "analog hacked" IR emitters and detectors to advise the vehicle in which direction to head. The IR emitters and detectors are used to locate "obstacles" in a relatively close vicinity. There are two forward facing IR emitter and detector pairs.

Note: The analog hack was performed as documented in the "Manuals" section of the www.mekatronix.com website.

In the first platform design, the IR LEDs were to be located on the underside of the vehicle. During testing of this configuration, the IR receiver theoretical range of 85 – 130 on a 68HC11 analog port was not achieved. Instead, the actual measured range was from 120 – 130. It was determined that the IR LEDs were located such that they saturated the ground with IR which reflected back to the IR receivers (i.e., the IR receivers were detecting the ground as an obstacle). To achieve the maximum theoretical range of the "analog hacked" IR receivers, it was suggested that I columnate the IR LEDs in heat shrink tubing. Also, I was instructed place the IR LEDs directly on top of the IR receivers as pictured in the figure below.



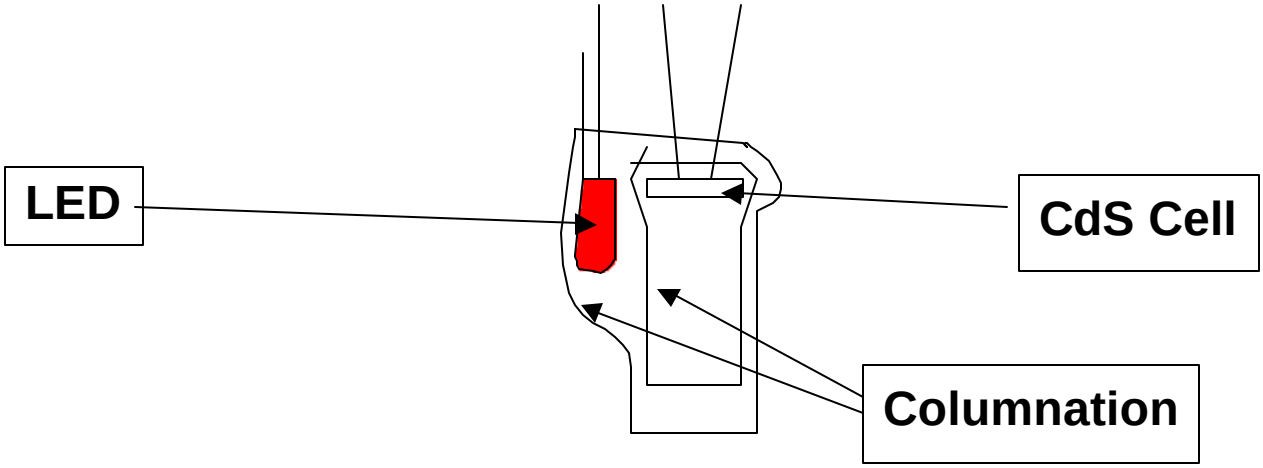
Bumper Switches

Just like a human being, the infrared system cannot see everything. Therefore, there are a series of four switches that notify the microcontroller when the vehicle has made contact with another object. Because the car will most likely run into objects from the front, it would be better served if there were more sensors located toward the front of the vehicle. There is one switch located at the center of the rear bumper, and three switches located on the front of the vehicle. One of the front switches is located at the center of the front bumper. The other two switches are located on the left and right side of the front bumper. The switches are connected to the resistor network divided PE0 analog port on the TJPRO11 board. Depending on which switch(es) is(are) pressed, a different voltage is measured on the analog port. When a front switch is pressed, the microcontroller tells the drive motor to go in reverse and wait for further feedback.

CdS Cells

To notice the difference between lanes and the road, CdS cells are used. CdS cells change resistance depending on the lightness or darkness of the object directly beneath it. To interface the CdS cells with the microcontroller, a simple voltage divider network is made with the CdS cell (a variable resistor) and a 48kÙ resistor. There are three CdS cells; one located at each of the left, center, and right front underside of the vehicle.

To achieve environment independence, an ultra-bright LED is used in combination with each CdS cell. To shield out ambient light, the CdS cell is columnated in heat shrink tubing. Then, the ultra-bright LED and columnated CdS cell are columnated in heat shrink tubing as shown in the figure on the following page.



This configuration was tested in an extremely well lit environment and a poorly lit environment. Both scenarios yielded the same results, proving environment independence.

Sonar

Sonar is typically used for obstacle detection. An ultrasonic sound wave is emitted from a sonar emitting transducer and the wave reflects back off objects. If a sonar receiving transducer detects a sound wave that has reflected off an object, one can calculate the distance away from the object using the following equation:

Distance of Obstacle = [(Time of emission) – (Time of reception)] * (Rate of speed of sound in air) / 2

In this project, sonar is used to detect how far "Drunk Ride" is away from home. The manner in which this is accomplished is as follows:

A sonar emitting beacon, "home", will continuously emit a 40 kHz pulse. "Drunk Ride" will pause while navigating the streets to scan the area for the beacon. When the sonar receiver on the vehicle has detected the beacon, it will listen for two consecutive pulses. From the time lapse

between the receptions of the two consecutive pulses, one can calculate the distance from the beacon using the following equation:

Distance to Beacon = [(First time of reception) – (Second Time of reception)] * (Rate of speed of sound in air)

However, to keep the calculation simple, just the difference between the two times will be used as a determination of "distance."

In this project, sonar is also being used to direct "Drunk Ride" to home. The direction of "Home" in relation to "Drunk Ride" will be located in the following manner:

The sonar receiver will be mounted on a servo that can turn 180°. The servo will scan from left to right and save the servo location in memory when the receiver detects a beacon. The servo will continue to scan from left to right until the receiver no longer detects a beacon. From the current servo location and the servo location stored in memory, one can calculate the general direction to head in the following manner:

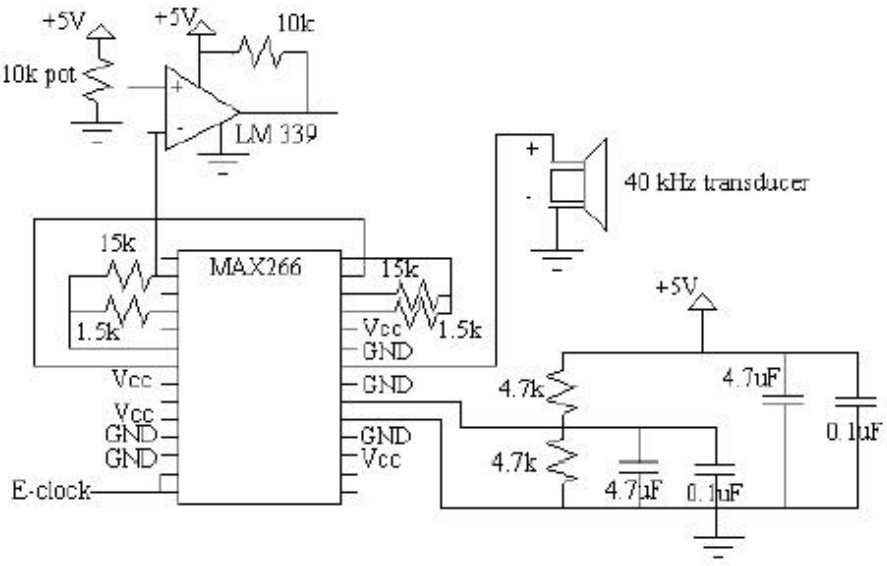
Direction to Head = [(Current Servo Location) – (Servo Location Stored in Memory)] / 2

If the sonar receiver does not detect the beacon, "Drunk Ride" will wander the streets randomly looking for the beacon.

Circuits

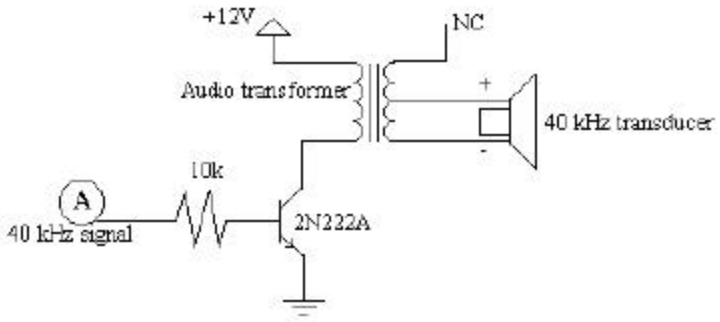
I am using the circuit found on the IMDL website (www.mil.ufl.edu/imdl). This schematic was modified to work "correctly" originally used by Michael Apodaca in the spring of 1998 and later modified by several people including Megan Grimm in the fall of 1998. The schematic for the receiver circuit is pictured on the following page, as well as the schematic for the transmitter

circuit. The output of the LM 339 op-amp comparator on the sonar receiver circuit will yield +5 V if the sonar transducer detected a pulse, and 0 V otherwise.



Sonar Receiver

Courtesy Megan Grimm's Report

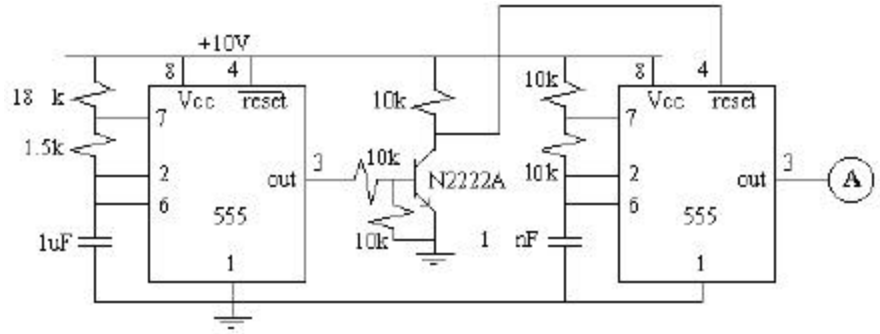


Sonar Transmitter Circuit

Courtesy Megan Grimm's Report

Because I am using a sonar beacon that is not attached to the TJPRO11 board with onboard 40 kHz generation, I needed a way to generate a 40 kHz signal for input into the sonar transmitter circuit. I first attempted to use Megan Grimm's 40 kHz generation circuit with a few modifications as pictured on the following page. I could not find an 18.3 kÙ resistor or a 1.2 nF

capacitor. Therefore, I replaced them with a 18 kÙ resistor and a 1 nF capacitor respectively. Also, Megan's original calculations called for a 10 kÙ potentiometer between pins 2 and 7 of the rightmost 555 chip. However, when she implemented her circuit, she found she needed a 15 kÙ resistor to achieve a 40 kHz signal. When I implemented her circuit with my two changes, I found I needed the 10 kÙ resistor that she originally calculated.

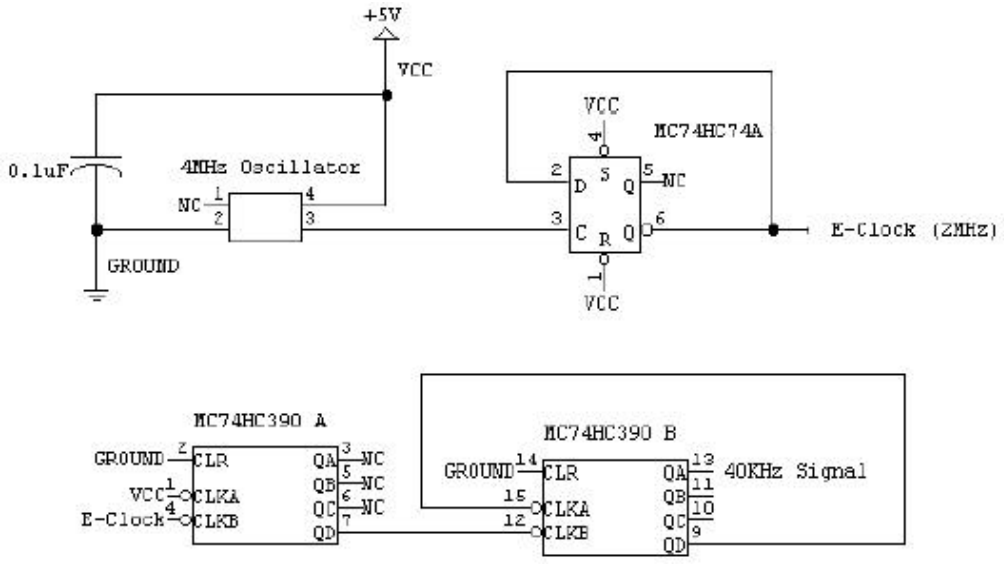


40 Khz Generation Circuit

Courtesy Megan Grimm's Report

When powered, a clicking noise from the sonar emitter could be heard. Because I thought this was not normal, I decided to find another alternative for generating a 40 kHz signal.

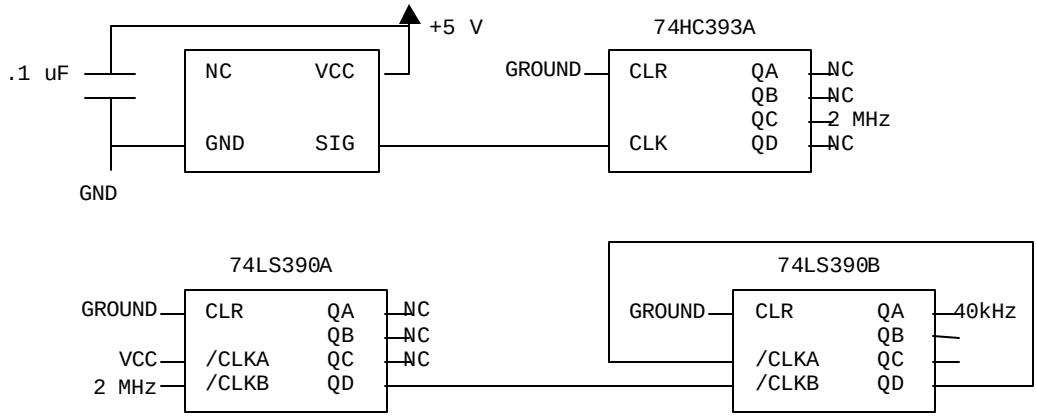
Looking back at Michael Apodaca's report, I found a different implementation of a 40 kHz signal generator, pictured in the figure on the following page.



40 Khz Generation Circuit

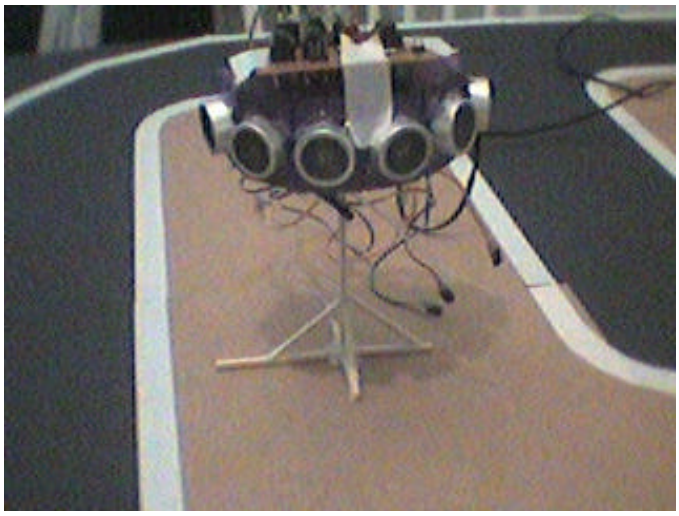
Courtesy Michael Apodaca's Report

Several factors prevented me from building an exact copy of his circuit implementation. Because I could not find a 4 MHz oscillator, my circuit, pictured in the figure below, looks a little different. I found a 16 MHz oscillator, which could produce a 2 MHz signal easily with a "divide-by-8" chip. I decided to use the 74HC393 chip for the "divide-by-8" function. Also, I could not find a 74HC390 chip, so I used a 74LS390. After testing out this circuit, no clicking noise could be heard from the sonar emitter.



40 kHz Generation Circuit

This circuit seemed to work correctly. However, after a while, the circuit would no longer produce an ultrasonic sound wave. Therefore, to prove the project worked, I used a sonar beacon from a previous IMDL project pictured below.

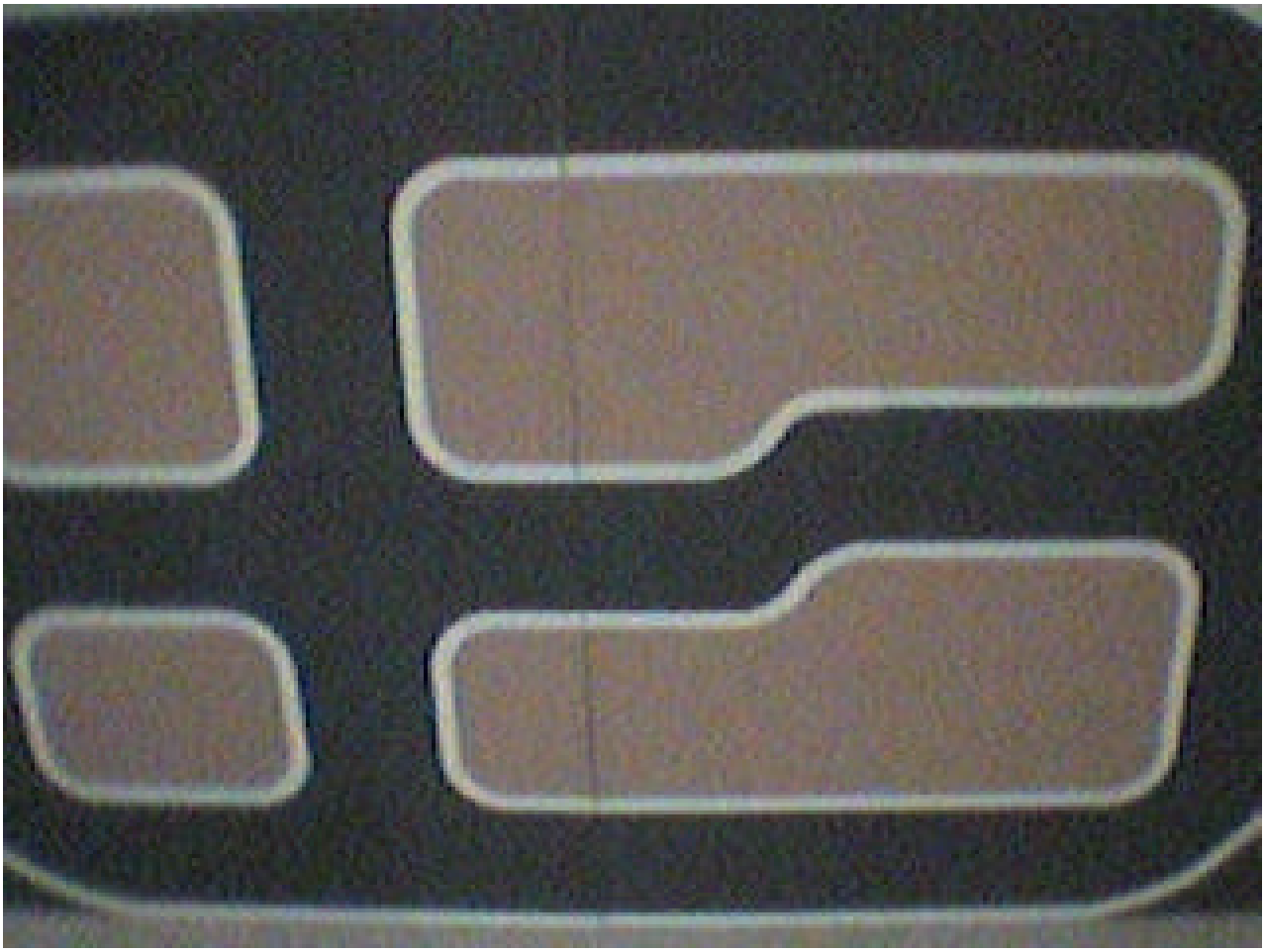


Behaviors

The automobile is able to stay within a lane, avoid obstacles (other cars, pedestrians, etc.), react to impacts, and steer the vehicle safely to the sonar transmitting destination. It was rather difficult to get the vehicle to stay within the rather narrow lanes that I had designed. I had to add in a case when the vehicle totally crossed over a line, it would back up until it had gotten back onto the road. Another difficult situation was making sure the sonar receiver knew when it was close enough to the sonar transmitter. This was achieved using trial and error.

Experimental Layout

The city streets were constructed using cardboard for the foundation, black construction paper in place of asphalt, and white masking tape for lanes. The home is a sonar transmitting beacon. To prove the correct operation of the vehicle, the vehicle was placed in random locations throughout the city as was the "home". "Drunk Ride" then navigated the streets and ended up at "home." The course is pictured below.



Conclusion

The robot successfully accomplished the tasks of staying within the lanes, avoiding obstacles, and navigating to the "home." While the project was rather successful, there were some portions that I was disappointed with. I never could get my own sonar transmitting beacon to work correctly, and the sonar transceiver did not yield the range I was looking for. Also, the small design of the robot led to a cramped environment. All in all, this robot exhibited artificial intelligence and could be the basis for a larger scale autonomous vehicle someday.

For future work, if this project was to be implemented in a life-size scale, the location detection most likely would be accomplished by GPS as opposed to sonar. Location detection should be implemented by using sonar as opposed to IR. Both of these modifications would allow for more environment independence.

Appendix A – Parts

Main Processor

Microcontroller – MTJPRO11A - $85 – Mekatronix

Serial Communications Board – MB2325A - $5 – Mekatronix

DB9 to DB25 - $5.95 – Radio Shack

Sensors

CdS cells – pack of 5 - $1.95 – Radio Shack

IR Sharp Can - $1.95 – Radio Shack

IR LED - $.95 – Radio Shack

Heat Shrink Tube - $2.95 – Radio Shack

Ultra-bright LED - $1.95 – Radio Shack

Sonar Receiver

Sonar Transducer – Donated to me by another IMDLer

Filter – MAX266 – Free – www.maxim-ic.com

Comparator – LM339 - $1.95 – Radio Shack

Sonar Transmitter

Sonar Transducer – Donated to me by another IMDLer

Audio Transformer – 273-1380 - $2.95 – Radio Shack

Timer – LM555 - $.65 – Radio Shack

16MHz Oscillator - $4.00 – Electronics Plus

Appendix A – Code

```
/*********************************************************************
 * Title            tjpbasej.h
*
 * Programmer       Jon Palgon                                      *
 * Date             April 22, 2002                                     *
 * Version          1.0
*
 *                                                                   *
 * Description
             *
 *   Collects include files and general constants into one file.     *
 *********************************************************************/

/************************** Includes *********************************/

#include <analog.h>
#include <clocktjp.h>
#include <motortjp.h>
#include <servotjp.h>
#include <serialtp.h>
#include <isrdecl.h>
#include <vectors.h>

/************************ End of Includes ****************************/

/************************** Constants ********************************/
#define LEFT_MOTOR      0
#define RIGHT_MOTOR     1
#define STEERING_SERVO 0
#define SONAR_SERVO 0
#define MAX_SPEED      100
#define HALF_SPEED      75
#define ZERO_SPEED       0
#define RIGHT_TURN            3500
#define LEFT_TURN             4500
#define NO_TURN         4000

#define BUMPER     analog(0)
#define RIGHT_IR   analog(2)
#define LEFT_IR    analog(3)
#define LEFT_CDS   analog(7)
#define CENTER_CDS analog(5)
#define RIGHT_CDS  analog(4)
#define SONAR      analog(1)

#define START     while(BUMPER<120)
#define FRONT_BUMP  (BUMPER>10)&&(BUMPER< 120)
#define BACK_BUMP  BUMPER>120

/* Enable OC4 interrupt and all servo operations */
#define SERVOS_ON     SET_BIT(TMSK1,0x10)
```

```
/*Disable OC4 interrupt: Stops all servo holding torques, useful for energy
savings*/
#define SERVOS_OFF    CLEAR_BIT(TMSK1, 0x10)

#define IRE_ON  *(unsigned char *)(0x7000) = 0x81
#define IRE_OFF   *(unsigned char *)(0x7000) = 0x00

/*Turn the BRAKE lights ON and OFF*/
#define BRAKE_ON  *(unsigned char *)(0x4000) = 0xC3
#define BRAKE_OFF *(unsigned char *)(0x4000) = 0x42

/*Turn the LEFT_TURN signal ON and OFF*/
#define LEFT_TURN_ON  *(unsigned char *)(0x4000) = 0x40
#define LEFT_TURN_OFF *(unsigned char *)(0x4000) = 0x42

/*Turn the RIGHT_TURN signal ON and OFF*/
#define RIGHT_TURN_ON  *(unsigned char *)(0x4000) = 0x02
#define RIGHT_TURN_OFF *(unsigned char *)(0x4000) = 0x42

/*Turn the RIGHT_TURN and LEFT_TURN signals ON and OFF*/
#define HAZARD_ON  *(unsigned char *)(0x4000) = 0x00
#define HAZARD_OFF *(unsigned char *)(0x4000) = 0x42

/************************ End of Constants ****************************/




/*****************************************************************************
 * Title:         finalpro.c
                  *
 * Programmer:    Jon Palgon
                  *
 * Date:          4/22/02
                           *
 * Version:       1.0
                           *
 *
                                 *
 * Description:
                           *
 *          This is a very simple proof of concept program.
            *
 *****************************************************************************/

/******************************** Includes ********************************/
#include <tjpbasej.h>
/***************************** End of Includes ****************************/

/******************************** Constants *******************************/
#define ZERO 0
#define AVOID_THRESHOLD 100
#define SONAR_THRESHOLD 5
#define SONAR_SERVO_0_DEG 1750
#define SONAR_SERVO_45_DEG 2700
#define SONAR_SERVO_135_DEG 5000
```

```
#define SONAR_SERVO_180_DEG 6000
#define SONAR_SERVO_INCREMENT 212
/***************************** End of Constants ***************************/

/******************************** MAIN() ********************************/
void main(void)
{
      unsigned int i, j, counter, lower_bound, upper_bound, dir;
      unsigned int turnvar, nextturn, nextspeed;
      unsigned int counter_num;
      unsigned rand;
      int irdr, irdl, CdSr, CdSc, CdSl, speed;

      init_analog();
      init_motortjp();
      init_servotjp();
      init_clocktjp();

      IRE_ON;             /* turn on IR emitters */
      BRAKE_ON;     /* turn on BRAKE lights */

      START;  /*Press the rear bumper to start the program*/
      BRAKE_OFF;    /* turn off BRAKE lights */

      nextspeed = ZERO_SPEED;
      nextturn = NO_TURN;

      while(1)
      {
            for(j = ZERO; j < 100; j++)
            {
                  irdr = RIGHT_IR;
                  irdl = LEFT_IR;
                  CdSr = RIGHT_CDS;
                  CdSc = CENTER_CDS;
                  CdSl = LEFT_CDS;

                  if (FRONT_BUMP)
                  {     motorp(RIGHT_MOTOR, MAX_SPEED);
                  wait(3000);
                  }
                  if ((irdl > AVOID_THRESHOLD)  || (irdr > AVOID_THRESHOLD))
            {
                        speed = ZERO_SPEED;
                        turnvar = NO_TURN;
                  }
                  else if ((CdSr > 50)  && (CdSr < 100) && (CdSc < 125))
                  {
                        wait(100);
                        servo(STEERING_SERVO, RIGHT_TURN);
                        motorp(RIGHT_MOTOR, MAX_SPEED);
                        while (CdSr < 100)
                              CdSr = RIGHT_CDS;
                        wait(100);
                        motorp(RIGHT_MOTOR, ZERO_SPEED);
                        wait(100);
                  }
```

```
                else if ((CdSl > 60)  && (CdSl < 100) && (CdSc < 125))
                {
                      wait(100);
                      servo(STEERING_SERVO, LEFT_TURN);
                      motorp(RIGHT_MOTOR, MAX_SPEED);
                      while (CdSl < 100)
                            CdSl = LEFT_CDS;
                      wait(100);
                      motorp(RIGHT_MOTOR, ZERO_SPEED);
                      wait(100);
                }
          else if ((CdSr < CdSc) && (CdSr < CdSl) && (CdSr < 100))
                {
                      speed = HALF_SPEED;
                      turnvar = LEFT_TURN;
          }
          else if ((CdSl < CdSc) && (CdSl < CdSr))
          {
                speed = HALF_SPEED;
                      turnvar = RIGHT_TURN;
                }
          else
                {
                      speed = nextspeed;
                      turnvar = nextturn;
          }

          if (j == 99)
          {
                speed = ZERO_SPEED;
                BRAKE_ON;
          }

          motorp(RIGHT_MOTOR, -speed);
          servo(STEERING_SERVO, turnvar);

                wait(50);

                if (j == 99)
                {
                      lower_bound = ZERO;
                      upper_bound = ZERO;
                      for(i = SONAR_SERVO_0_DEG; i < SONAR_SERVO_180_DEG; i
+= SONAR_SERVO_INCREMENT)
                      {
                            servo(1, i);
                            counter_num = ZERO;
                            for(j = ZERO; j < 20; j++)
                            {
                                  counter = ZERO;
                                  while((SONAR > 200) && (counter < 1000))
                                  {
                                        counter = counter + 1;
                                  }

                                  if (counter != ZERO)
                                  {
```

```
                                        counter_num++;

                                        if (counter >= SONAR_THRESHOLD ||
counter_num >= SONAR_THRESHOLD)
                                        {     BRAKE_ON;
                                              wait(2000);
                                              BRAKE_OFF;
                                              return;
                                        }
                                  }
                            }

                            if (lower_bound == ZERO && counter_num > ZERO)
                                  lower_bound = i;
                            if (counter_num > ZERO)
                                  upper_bound = i;

                            wait(500);
                      }

                      if (lower_bound != ZERO && upper_bound != ZERO)
                      {
                            dir = (lower_bound + upper_bound)/2;
                            servo(SONAR_SERVO, dir);
                      }
                      else dir = ZERO;

                      if (dir == ZERO)
                      {
                            rand = TCNT;

                            if (TCNT & 0x0001)
                                  nextturn = LEFT_TURN;
                            else
                                  nextturn = RIGHT_TURN;

                            nextspeed = HALF_SPEED;
                      HAZARD_ON;
                      }
                      else if (dir <= SONAR_SERVO_45_DEG)
                      {
                            nextturn = LEFT_TURN;
                            nextspeed = HALF_SPEED;
                      LEFT_TURN_ON;
                      }
                      else if (dir <= SONAR_SERVO_135_DEG)
                      {
                            nextturn = NO_TURN;
                            nextspeed = MAX_SPEED;
                      BRAKE_ON;
                      }
                      else
                      {
                            nextturn = RIGHT_TURN;
                            nextspeed = HALF_SPEED;
                      RIGHT_TURN_ON;
```

```
                        }
                    }
                }
            }
}
/****************************** END OF MAIN() ****************************/
```