UNIVERSITY OF FLORIDA

Intelligent **M**achines **D**esign **L**aboratory

(Toddler-Tail)

Submitted by:
Shalom Darmanjian

Final Report

Table of Contents

Abstract

Toddler-Tail is an autonomous car that uses a vision system to monitor and follow a small child. In tandem with the vision system, a voice playback chip is used to report negative behavior to both the child and the caretaker. An Atmel micro-controller is used to process localized obstacle detection, decode RF data, and actuate the wheel servos. In its entirety, this small sub-100 dollar robot can mimic the functions of a multi-thousand dollar vision system.

Executive Summary

Four main components were designed and implemented for toddler-tail. The first and central component was the vision system. It was charged with the task of locating and tracking a child using a real-time video signal. The simple solution assumed that the child would wear the same clothes (with distinct colors) throughout the tracking/monitoring session. With this assumption, color detection was used for the monitoring/tracking. To make the vision system robust and dynamic, it was designed to calibrate for different lighting environments and different tracking colors.

Another important component to the robot was RF communications. RF modules were chosen so that the computer (employing vision) could communicate to the ATMEL board (and the on-board mechanisms i.e. servos, alarm, etc). To lessen the complexity, the parallel port (on the computer) controlled the RF transmitting module. Because the parallel port latches the data sent to it, it was necessary to use a pre-amble when sending a new command to the receiver unit. This preamble was implemented so that the Atmel board could distinguish between a repeating command and a latched command. The siren-alarm is an example of when not to continuously repeat the command (just a quick warning) unless instructed (child goes out of view).

The voice playback chip was another integral part of this robot car. Not only does it add to the human-machine interaction, it also serves to alert the child or caretaker when something is wrong. The correct feedback is accomplished with the aid of the vision system; certain behaviors would cause a particular voice segment to be played (no running, I can not see you, etc…).

The final module to this composite system was the micro-controller. The Atmel mega-163 was selected for its ease of use and versatile instruction set. It mainly served as a conduit; all the commands were sent and received through it. This Atmel chip was also directly wired to four bump sensors and an IR sensor. These sensors were employed for localized obstacle detection and would immediately override any commands that originated from the vision system.

Introduction

The initial project involved the construction of an architecture that could be transplanted into multiple platforms. This form of architecture would subsequently increase the likelihood of a completed robot. A robotic helicopter was the first platform researched. A robotic boat was the second candidate researched. And as a last alternative, a robotic car was investigated.

Since the four main components of the robot (vision, RF, voice, MCU) could be designed and implemented independently from the platform choice, work began on these modules. Since the eventual robot needed to posses certain qualities and behaviors, one application investigated was tracking and monitoring of a person. For the helicopter design, it could follow and monitor hitchhikers as they traveled. Hitchhikers could also request for the robot to search the trail ahead. The helicopter could also be used in search and rescue missions for lost people. Police departments or military personnel could use the helicopter for surveillance or tracking of criminals/combatants. For the boat design, the same tracking/monitoring could be applied to boats or lost people at sea (hundreds of these boats could scour the ocean cheaply and quickly). The robot car could track and follow a child in a separate room (or outside) from a caretaker. It could also provide real time video of the child to the caretaker.

Voice synthesis/playback was another behavior researched. The three platforms were amplified with their usefulness with this particular component. It could serve to either attract a lost person at sea or in the forest. It could serve to notify a child or caretaker of negative behavior occurring. Police or military personnel could use it to communicate actions or requests to criminals/combatants.

The insufficient amount of time coupled with lack of mechanical and technical experienced force the helicopter platform to be temporarily abandoned. The cost of replacement components is what eventually warranted the removal of the boat platform. If water leaked into the electronics, the ability to complete a successful robot would be diminished greatly. The final decision was to use the robotic car platform.

Integrated System

As stated earlier, the crux of the entire robotic architecture is dependant on the vision system. From the vision system the next track of communication transpires across the RF modules. Then from this domain, it enters the MCU. After the MCU decodes the preamble and the desired command, the appropriate message is sent to the actuators (servos), the alarm circuit, or the voice playback chip. The obstacle detectors are the last members of this system (which have a direct line of communication to the MCU). Figure 1 below demonstrates this integration.

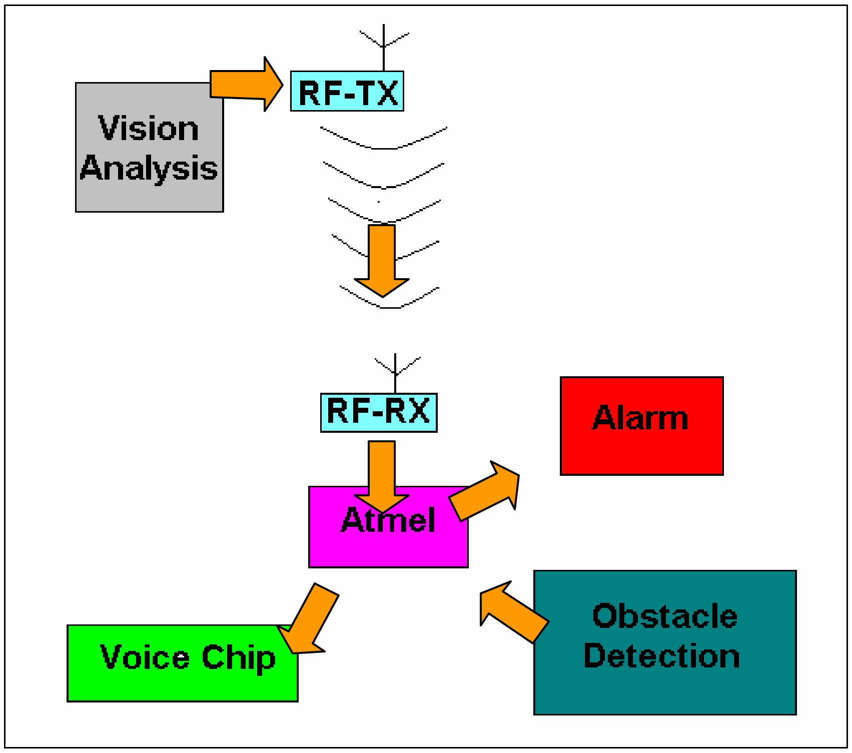


Figure 1

Since all four components were modular, testing of this system was simplified and could take place outside of the physical platform. Each individual module could also be tested independently from one another. Often, tests were run on a single component, then two then three then all four. Finally, all four components were tested with the obstacle detection/alarm. This sort of gradient testing can allow for sequential testing and a confidence that previous components were working.

Mobile Platform

Due to time constraints, it was decided that a pre-existing car should be used as the platform, rather than building one from scratch. Figure 2 below shows an example of the pre-made car of choice (PT-Cruiser model).

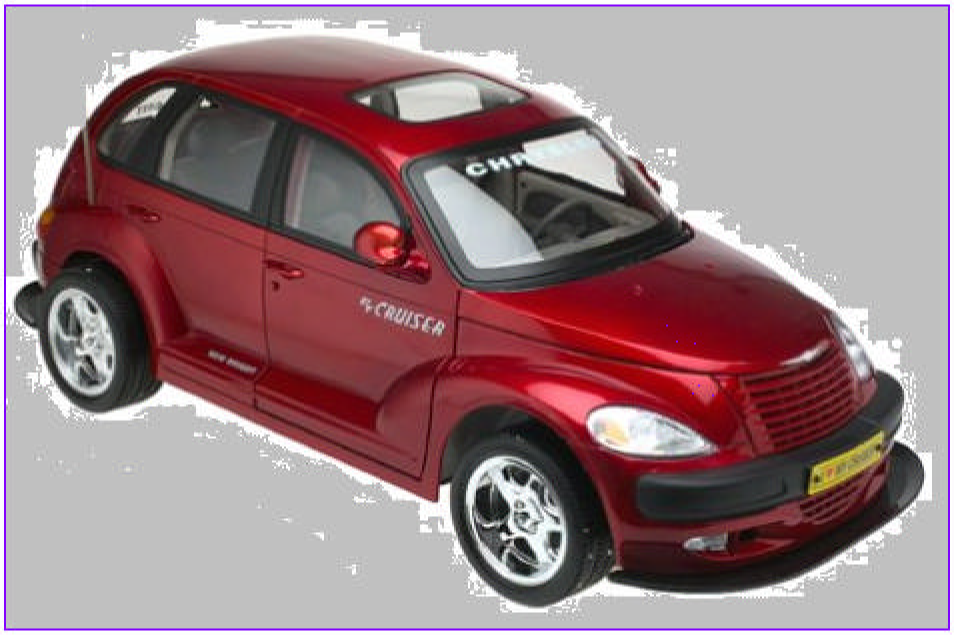


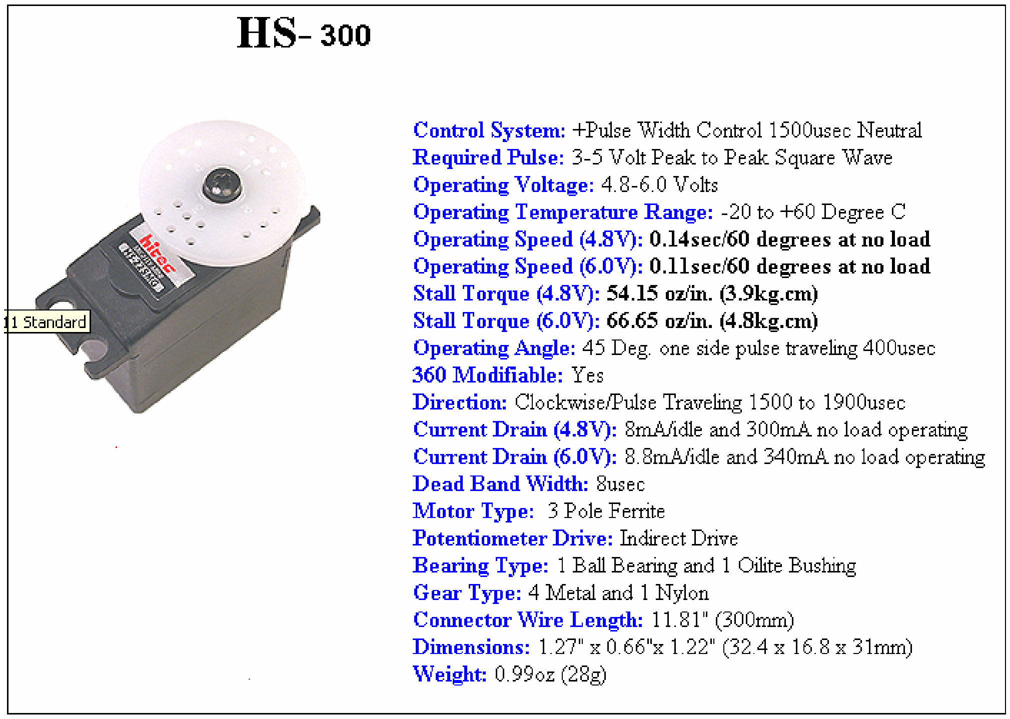
Figure 2

The rear wheels were removed and servos were inserted in their place. In order to get more speed out of the servos, wheels with large diameters were attached to the servos. With an increase diameter, theoretically more displacement can be gained for the same amount of torque. The electronics were housed within the car body (with chairs and etc removed). The camera was mounted on top of the vehicle behind the sunroof. Three speakers were mounted in a triangular formation on top of the vehicle. The four bump sensors were strategically placed around the front grillwork of the vehicle. And finally, the IR sensor was placed on rear of the vehicle above the bumper.

Choosing a purple colored car was the first embarrassing lesson that was learned. Underestimation of weight constraints was another lesson learned. It was also learned that physical dimensions could have a drastic effect on torque delivery. Having the elongated car caused the weight to be distributed lengthwise and consequently decreased the torque when turning (due to a narrow base). Another contributing factor to the poor turning ability was the weak grip on the rear wheels.

Actuation

Actuation for each wheel was accomplished with an HS-300 servo. Figure 3 below is not the same servo used in this project but it is similar (servo-city doesn't sell the 300 any longer). Each server was hacked and calibrated for a stopping point at a 10% duty cycle. The Atmel board was solely in charge of generating the PWMs for each servo. The servos were instructed move at their full speed at all times.



HS-300

Control System: +Pulse Width Control 1500usec Neutral
Required Pulse: 3-5 Volt Peak to Peak Square Wave
Operating Voltage: 4.8-6.0 Volts
Operating Temperature Range: -20 to +60 Degree C
Operating Speed (4.8V): **0.14sec/60 degrees at no load**
Operating Speed (6.0V): **0.11sec/60 degrees at no load**
Stall Torque (4.8V): **54.15 oz/in. (3.9kg.cm)**
Stall Torque (6.0V): **66.65 oz/in. (4.8kg.cm)**
Operating Angle: 45 Deg. one side pulse traveling 400usec
360 Modifiable: Yes
Direction: Clockwise/Pulse Traveling 1500 to 1900usec
Current Drain (4.8V): 8mA/idle and 300mA no load operating
Current Drain (6.0V): 8.8mA/idle and 340mA no load operating
Dead Band Width: 8usec
Motor Type: 3 Pole Ferrite
Potentiometer Drive: Indirect Drive
Bearing Type: 1 Ball Bearing and 1 Oilite Bushing
Gear Type: 4 Metal and 1 Nylon
Connector Wire Length: 11.81" (300mm)
Dimensions: 1.27" x 0.66"x 1.22" (32.4 x 16.8 x 31mm)
Weight: 0.99oz (28g)

Figure 3

Lesson that was learned is there is always a need for more torque and speed. The quality of the project outweighs the costs of increasing the torque and speed. Also increasing the power would have been able to assist in battery-drainage issues.

Input and Output

Vision System

The vision system was comprised of three main pieces of equipment. An old $20 TV capture card, a $60 wireless camera and a desktop computer.

Since the IEEE robot team was unwilling to allow their $800 frame grabber from being borrowed, other avenues were pursued. A friend of mine had an old tv capture card that he was no longer using. After hours of research, linux drivers were found for that particular card (haupenpage win-tv tuner card). As is the case with linux, the source code was available. After an immeasurable amount of time, the raw frame data could finally be manipulated.

Once the raw data could be accessed, the algorithms could be designed for that particular card. Traditionally, a lot of vision research involves the RGB space. For these particular drivers, a YUV format was the available color space. It was decided early on that there might be an advantage to using the YUV for biasing against differing lighting conditions (as opposed to transforming the YUV space back into RGB space).

A simple histogram method was employed to define the color space of the calibrated shirt color. The largest frequencies were used to bias against other colors. Then the mean x and y of the color blob was found, as well as the variance (or spread) of the pixels. Using these simple statistical measures, the robot was able to accurately track the colored shirt through space. The variance was able to tell the robot when to move forward and when to move backward. It was also able to distinguish when the child was climbing or moving towards the camera (since moving forward would cause y-values to increase and the variance to increase, whereas climbing would not produce a change in variance on in y). Simple global pointers were used to keep track of the last centroid coordinates. With a history of the centroid movement, the robot would know when the child was running by comparing in time the change in x (basically a velocity). Figure 4 shows the tv tuner card. Figures 5 and 6 show the wireless camera.

http://cgi.ebay.com/ws/eBayISAPI.dll?ViewItem&item=2076349744

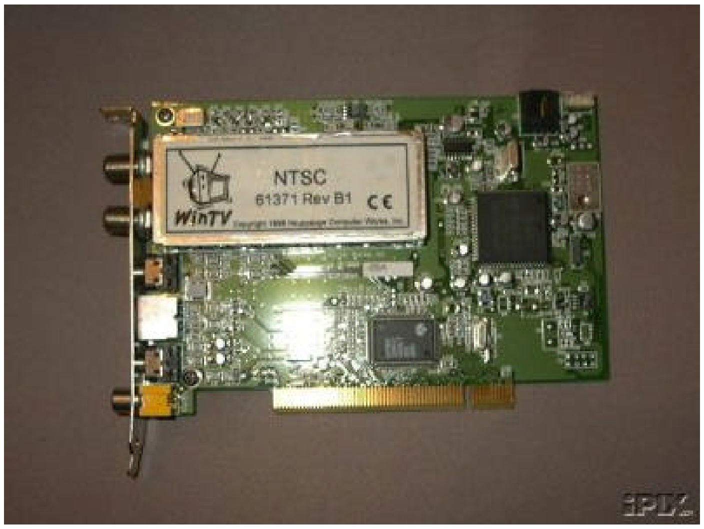
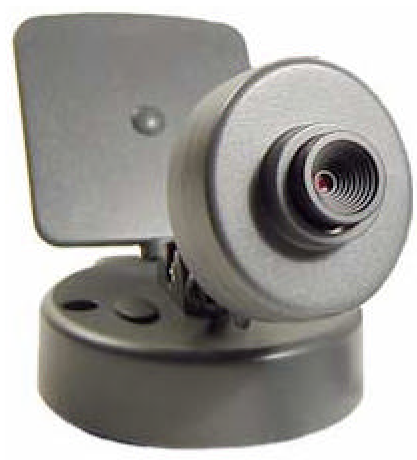


Figure 4



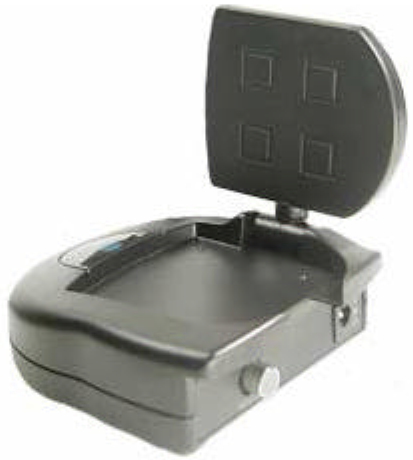
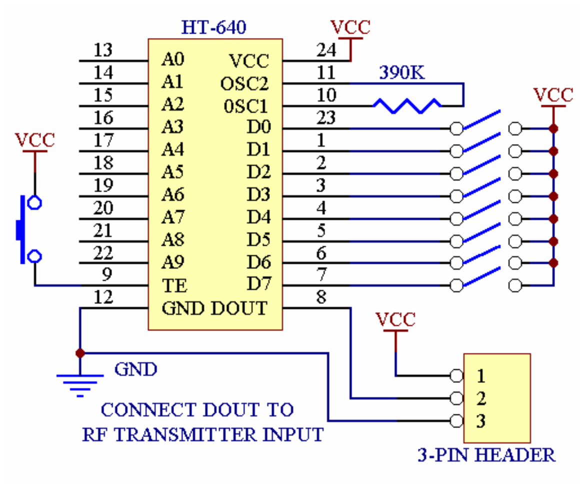


Figure 5 **Figure 6**

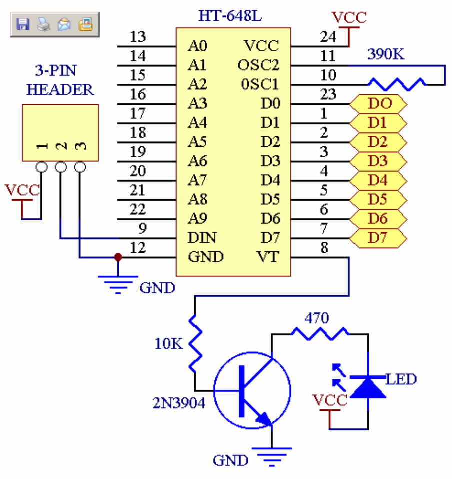
RF Communication

Reynolds electronics provided the transmitting and receiving RF modules. The particular model HT-648, was capable of transmitting 8-bits at a time. The carrier frequency was 433 Mhz and enabled the RF to transmit across hundreds of feet undisturbed. The modules were consistent and relatively trouble free. The schematics are shown below in figures 7 and 8. www.rentron.com can provide these modules for about $20.



HT-640
VCC
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 TE GND DOUT
VCC OSC2 0SC1 D0 D1 D2 D3 D4 D5 D6 D7
390K
GND
CONNECT DOUT TO RF TRANSMITTER INPUT
3-PIN HEADER

Figure 7



HT-648L
3-PIN HEADER
VCC
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 DIN GND
VCC OSC2 0SC1 D0 D1 D2 D3 D4 D5 D6 D7 VT
390K
GND
10K
470
2N3904
LED

Figure 8

Voice Playback

The ISD 256 voice chip was used to play speech segments back to the child and caretaker. You can see the schematic for the circuit that incorporated the voice record chip in figure 9 below. This chip is around $5 and can be purchased through digi-key.

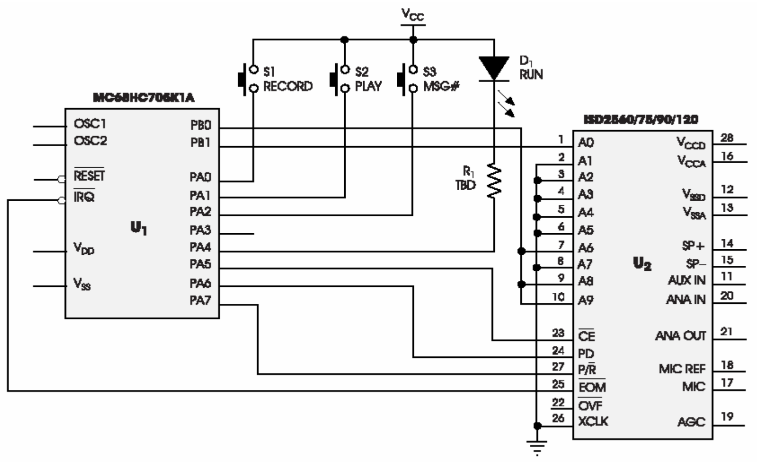


Figure 9

Local Obstacle Detection

The four bump sensors and an IR sensor were directly connected to the Atmel board. They had the ability to alert the Atmel which part of the car the obstacle had been detected. The IR sensor was purchased through www.acroname.com and was about $20. The bump sensors were obtained at radio shack for about $1.

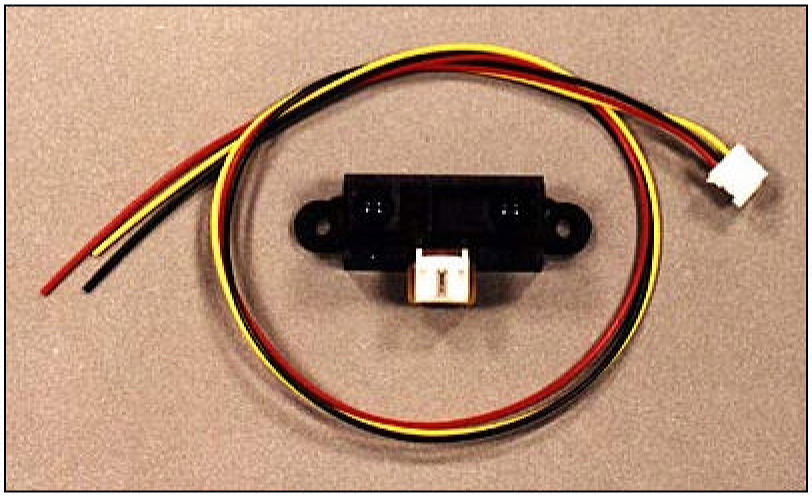


Figure 10

Behaviors

The main behavior of the robotic car was to follow and monitor a child. If the child had engaged in any negative behavior, then the robot would notify the child and caretaker using a voice segment. It was meant to detect climbing, running and the movement of a restricted object. The quality of the camera was the main issue as to why some of the vision algorithms were not consistent. Another culprit was the computer-to-RF communications. The robot could accurately print the text form of when the child was running or climbing, but with the two minor flaws it sometimes could not communicate the appropriate voice segment.

Conclusion

A realistic summery of the work accomplished would include the word success. While some of the components had some inter-communication problems and the servos had power issues, overall the robot could do the tasks it was assigned. A great personal accomplishment was the development of an independent and novel vision system. A normal frame grabber can range from 700-2000 dollars, which this novel solution could emulate for only $20. Building the circuit for the voice chip save the project about $60 and using a cheap $60 camera save $500 from buying the usual DV camera researchers employ in vision work.

Having a real-time vision test bed can serve to improve the vision algorithms for this robot. If the project were to be redone, then frugalness would not be the way to go. Spending more money on a better camera and better servos would propel this project further along a successful path. Changing the platform to a tank-like platform could increase the “cool” factor and allow for more stable control of the robot.

Appendix

(Motorfunc.h)

```
/***************************************
 * Motor Control Header File
 * Created by Kristen Allen                                      *
 *
 * June 19, 2002                                                        *
 *
 * This header file is for use with the *
 * motor control functions.                                      *
 ***************************************/

#ifndef MOTORFUNC_H
#define MOTORFUNC_H



#define SERVO_STOP 102
#define SERVO_RIGHT 1
#define SERVO_LEFT 1000
#define FULL_RIGHT 100
#define FULL_LEFT -100
#define GAIN                           10
#define K 10
#define FUNCTION ((K * tmp_speed) + speed)/(K+1)

extern void SERVO_MOTOR_init(void);
extern void motor_speed(int mot_num, int speed);
#endif
```

(Motorfunc.c)

```
/*********************************************************
 * Motor Control Functions                                                *
 * Created by Kristen Allen                                               *
 * -SERVO_MOTOR_init and motor_speed functions written -*
 * -by Rolando Panez and modified by Kristen Allen to  -*
 * -work with an ATMEGA163                             -*
 * - modified by SHALOM DARMANJIAN
 * December 10, 2002
 *
 * These functions are used to initialize the PWM                *
 * generator for use with a hacked servo. A hacked               *
 * servo can be calibrated at a 10% duty cycle.  Once                       *
 * it is calibrated, the servo will turn in different                       *
 * directions with a smaller duty or larger duty cycle.                     *
 ********************************************************/

#include <io.h>
#include <math.h>
#include "motorfunc.h"
#include "iom163.h"

void SERVO_MOTOR_init(void)
{
                           outp(SERVO_STOP, OCR1AL);
                           outp(SERVO_STOP, OCR1BL);
                            outp(SERVO_STOP, OCR2);
       outp((1<<COM1A1) | (1<<COM1B1) | (1<<PWM10) | (1<<PWM11), TCCR1A);

                outp((1<<CS22) | (1<<COM21) | (1<<PWM2), TCCR2);
                      outp((1<<CS11) | (1<<CS10), TCCR1B);
                                sbi(DDRD, PD4);
                                sbi(DDRD, PD5);
                                sbi(DDRD, PD7);

}

void motor_speed(int mot_num, int speed){
                             unsigned short OCRTMP;
                                 int TMP_SPEED;
                          unsigned short SERVO_SPEED;


                              if(mot_num == 0){
                                                             OCRTMP = inp(OCR2);


                                                                  if(speed == 0)
                                                                                 SERVO_SPEE
                                                  else if( speed >= FULL_RIGHT)
                                                                                 SERVO_SPEE
                                                   else if( speed <= FULL_LEFT)
                                                                                 SERVO_SPEE
                                                                           else
```

```
                         SERVO_SPEED = SERVO_STOP + speed;

                                                             outp(SERVO_SPEED, OCR2);
                                        }//if
                              else if(mot_num == 1){
                                                                OCRTMP = inp(OCR1AL);


                                                                       if(speed == 0)
                             SERVO_SPEED = SERVO_STOP;
                                                        else if( speed >= FULL_RIGHT)
                             SERVO_SPEED = SERVO_RIGHT;
                                                         else if( speed <= FULL_LEFT)
                             SERVO_SPEED = SERVO_LEFT;
                                                                                 else
                         SERVO_SPEED = SERVO_STOP + speed;

                                                          outp(SERVO_SPEED, OCR1AL);
                                    }//else if

                                      return;
}
```

(toddler.c)

```
// ***** CREATED BY SHALOM DARMANJIAN  *******//



#include <io.h>
#include "motorfunc.h"
#define SPEED_CHANGE 500       //Delays the change
#define TURNTIME 32000                  //Used to change time needed for change (max. 32000)

typedef unsigned char  u08;
int hold=1600;
void waitms(int delay)
{
                                  while (delay)
                                        {
                                      delay--;
                                       int i;
                            for(i=1597;i;i--)asm("nop");
                                    }//while
```

```
}//waitms
void message1(void)
{
         reset();
  //      sbi(PORTC,6);
        cbi(PORTC,3);//  this plays message
        sbi(PORTC,3);//




   //     nextmsg();
   //    play();


}
void message2(void)
{
  reset();
    sbi(PORTC,6);//   this tells it which mode to use

     cbi(PORTC,3);//  this jumps to next message
      sbi(PORTC,3);//     by pulsing CE





  //      nextmsg();
  //      nextmsg();
        play();


}// end of message2

void message3(void)
{
  reset();
        sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE
```

```
       nextmsg();
//       nextmsg();
 //      nextmsg();
       play();


}// end of message3
void message4(void)
{
  reset();
        sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE



 //      nextmsg();
 //      nextmsg();
       nextmsg();
       nextmsg();
       play();


}// end of message4
void message5(void)
{
  reset();
        sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE



   //    nextmsg();
   //    nextmsg();
       nextmsg();
       nextmsg();
                                      nextmsg();
       play();

}// end of message5
```

```
void message6(void)
{
  reset();
        sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE



  //     nextmsg();
  //     nextmsg();
       nextmsg();
       nextmsg();
                                     nextmsg();
                                     nextmsg();
       play();


}// end of message6
void message7(void)
{
  reset();
        sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE



   //    nextmsg();
    //   nextmsg();
       nextmsg();
       nextmsg();
                                     nextmsg();
                                     nextmsg();
                                     nextmsg();
       play();


}// end of message7
void message8(void)
{
  reset();
```

```
        sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE


     //  nextmsg();
     //  nextmsg();
       nextmsg();
       nextmsg();
                                    nextmsg();
                                    nextmsg();
                                    nextmsg();
       nextmsg();
       play();




}// end of message8
void message9(void)
{

  reset();





    sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE


      // nextmsg();
     //  nextmsg();
       nextmsg();
       nextmsg();
       nextmsg();
                                    nextmsg();
                                    nextmsg();
```

```
                                     nextmsg();
       nextmsg();
       play();


}// end of message9
void message10(void)
{

  reset();




    sbi(PORTC,6);//   this tells it which mode to use

        cbi(PORTC,3);//  this jumps to next message
        sbi(PORTC,3);//     by pulsing CE


      // nextmsg();
       nextmsg();
       nextmsg();
       nextmsg();
       nextmsg();
                                     nextmsg();
                                     nextmsg();
                                     nextmsg();
       nextmsg();
       play();


}// end of message10

void nextmsg(void)
{
             waitms(25);   //
                            cbi(PORTC,6); //  switch the modes
                        waitms(25);   //      and then switch back
                                     sbi(PORTC,6); //

                              cbi(PORTC,3);  // Pulse CE low
             sbi(PORTC,3);  //
}
void play(void)
{
                                   waitms(50);

                      sbi(PORTC,7);// mode M6 (pushbutton)
   cbi(PORTC,6);// this clears the mode and puts it back into push button mode
                      sbi(PORTC,5);// this tells it to play



                                                                  cbi(PORTC,3);//
                                                    cbi(PORTC,3);//  Pulse low CE
                                   sbi(PORTC,3);//

}

void reset(void)
{
       cbi(PORTC,7);
                                   sbi(PORTC,4);
       cbi(PORTC,4);
       sbi(PORTC,7);

       cbi(PORTC,6);
                                   sbi(PORTC,5);




} //end of reset
void alarm(void)
{
       outp(0xff,DDRD);
       sbi(PORTD,2);
                                   waitms(500);
                                   cbi(PORTD,2);



} //end of reset
void stopcar(void)
{
                               SERVO_MOTOR_init();
```

```
    motor_speed(0,0);
    motor_speed(1,0);
     waitms(50);
} //end of reset
void moveforward(void)
{
                 stopcar();
                 motor_speed(0,-20);  //this is forward
                 motor_speed(1,20);  //right wheel
       //          cbi(PORTC,0);
        //         waitms(500);
} //end of reset

void moveback(void)
{
                      stopcar();
                    motor_speed(0,20);  //this is backward
                    motor_speed(1,-20);
          //          cbi(PORTC,0);
           //         waitms(500);
} //end of reset
void moveleft(void)
{
                        stopcar();
                       motor_speed(0,20);  //this is left
                       motor_speed(1,20);
             //          cbi(PORTC,0);
              //         waitms(500);
} //end of reset
void moveright(void)
{


} //end of reset

void main(void){

                                   int i,j,k;
                                  int count=0;
        char res=0x0f;
        char oldres=0x00;
        char oldoldres=0x00;
        char ir=0x00;
        outp(0xff,DDRC);
        sbi(PORTC,7); //A6-9
                                                            cbi(PORTC,6); //A0
                                                                 sbi(PORTC,5);
                                                                 cbi(PORTC,4);
                                                                 sbi(PORTC,3);
                                                                          j=0;

                               SERVO_MOTOR_init();

// ************** START OF THE BUTTON CRAP

      for(;;){



       outp(0xfe,DDRB);
        ir=inp(PINB);
       if (ir==0x01){
                                    message10();
                                   moveforward();
                                    waitms(500);
                                     stopcar();
        cbi(PORTC,0);

}
 else sbi(PORTC,0);
      if(ir==0x02){//most right pushbutton
       message6();
       moveleft();
       waitms(350);
       moveforward();
       waitms(450);
       moveright();
       waitms(160);
       stopcar();
       cbi(PORTC,1);

   }
       else sbi(PORTC,1);

    if(ir==0x04){ //most left push button
```

```
     message6();
                                                        moveright();
                                                        waitms(350);
                                                       moveforward();
                                                        waitms(450);
                                                         moveleft();
                                                        waitms(160);
      stopcar();

       cbi(PORTC,2);

  }
   else sbi(PORTC,2);

             if(ir==0x08){  //center right
                                                            message6();

                                                               moveback();

                                                         stopcar();

                                                        cbi(PORTC,2);



                                                          }
      else sbi(PORTC,2);


             if(ir==0x10){//center left
             message6();
```

```
                                         cbi(PORTC,2);




   }
       else sbi(PORTC,2);

//***** END OF THE BUTTON CRAP

//*********** THIS IS FOR THE RF!!!
       outp(0x00,DDRA);
        res=inp(PINA);

//       outp(res,PORTC);
      if (res==0x01 && j==0&&oldres==0x00){

       j=1;

        cbi(PORTC,0);
        message1();

     //RECORD
       }
                /* invert the output since a zero means: LED on */
       else if(res==0x02&&j==0&&oldres==0x00){

       j=1;


      //PLAY
        cbi(PORTC,1);
       message2();
stopcar();


       }

       else if(res==0x03&&j==0&&oldres==0x00){
        j=1;
        cbi(PORTC,1);//this is the light
        cbi(PORTC,0);//this is the light....
        message3();

   }
       else if(res==0x04&&j==0&&oldres==0x00){
        j=1;

        count++;
   if (count==2){
                                             alarm();
     count=0;
 }
 else  message4();
stopcar();
   }
       else if(res==0x05&&j==0&&oldres==0x00){
        j=1;

        message5();

   }

       else if(res==0x06&&j==0&&oldres==0x00){
        j=1;


 message6();



   }
          else if(res==0x07&&j==0&&oldres==0x00){
           j=1;

           message7();

   }
             else if(res==0x08&&j==0&&oldres==0x00){
              j=1;

              message8();

   }
```

```
          else if(res==0x09&&j==0&&oldres==0x00){
           j=1;

           message9();

}
          else if(res==0x0A&&j==0&&oldres==0x00){
           j=1;

           message10();

}

          else if(res==0x0B&&j==0&&oldres==0x00){
              j=1;

              alarm();

}
          else if(res==0x0C&&j==0&&oldres==0x00){
              j=1;

              moveforward();
}
             else if(res==0x0D&&j==0&&oldres==0x00){
                 j=1;

         moveback();
   }

                else if(res==0x0E&&j==0&&oldres==0x00){
                    j=1;
              moveleft();

   }
                      else if(res==0x0F&&j==0&&oldres==0x00){
                                                j=1;
                           stopcar();
                                motor_speed(0,-20);  //this is left
                                        motor_speed(1,-20);
                                //          cbi(PORTC,0);
                                 //         waitms(500);
```

```
      }
             else if(res==0x10&&j==0&&oldres==0x00){
                 j=1;
                               SERVO_MOTOR_init();
             motor_speed(0,0);
             motor_speed(1,0);
             cbi(PORTC,1);
             waitms(500);

   }

       else {



        j=0;
//        sbi(PORTC,7);
    //    cbi(PORTC,6);
     //   sbi(PORTC,5);

   //    sbi(PORTC,0);
  //     sbi(PORTC,1);
 //     sbi(PORTC,2);

        }
oldres=res;

if (res>0x00)
 oldoldres=res;
   }
//*********** END OF RF STUFF

/* //*****************THIS IS FOR THE MOTOR- working!!!!!

                               SERVO_MOTOR_init();
                           motor_speed(0, 10); //Right
                               //MAX is -20 and +something?
                           motor_speed(1, -20); //Left

                                                              outp(0x00,PORTC);
*/ //***************** END OF MOTOR STUFF
}//main
```