**Don Lair**

**University of Florida**
**Department of Electrical and Computer Engineering**
**EEL 5666 Spring 2002**

# Intelligent Machines Design Laboratory
# Final Report

# Table of Contents

**Section** **Page**

## Abstract

In the interest of building a robot that had a personality I decided to build a robotic version of my favorite cartoon character, Homer Simpson. I thought of three functions that were required to capture the essence of Homer: walking, talking, and watching TV. I used two types of IR sensors, a universal remote, and a sound module with speaker to implement this. An Atmel ATmega8L microcontroller was used for HomerBot's brains.

## Executive Summary

As soon as I came up with the idea of HomerBot during the Spring 2001 semester I knew that I would take EEL5666 so that I could build it. It was also important to me that I take a class where I could put my engineering education to use.

Dr. Arroyo encouraged us to use a chip other than the HC11 to control the robot. After some research I decided to use the Atmel ATmega8L microcontroller. There were some advantages and disadvantages to using this chip instead of an HC11, but overall it was a very similar development environment and almost everything that I learned in EEL4744 was applicable.

I used two hacked servos for HomerBot's movement. The servos were driven by the PWM generating capabilities of the Atmel microcontroller. The two-wheel setup was chosen because it greatly simplifies movement and navigation since there is no turning radius.

HomerBot's speaking is handled by a Quadravox QV306M4 sound module. The module is controlled with serial communications from the microcontroller. The module has an on-boar amplifier and is connected to a small speaker.

There is a hacked universal remote that HomerBot can access once he finds the TV. HomerBot can use any TV that the universal remote is designed to support.

# Introduction

The intention of this robot was to take a different approach to the class than the other robots that I have seen. My main goal for this class is to build something that has a distinct personality, rather than a robot that has complicated mechanical functions since that is not my forte.

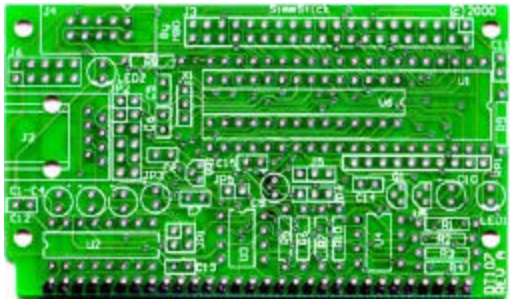
Once HomerBot is powered and started he immediately begins to search for the TV. He is actually looking for the signal from an IR beacon that is placed near the TV. Once he finds the TV he will watch it until he gets bored and will then start wandering around.

HomerBot has several (currently 18) sounds clips that he accesses randomly while searching for the TV.

## Integrated System

I chose the Atmel ATmega8L microcontroller for HomerBot's brains. There are five main advantages of using this chip instead of the HC11. It's cheap, it is easier to generate a PWM for driving the servos, the memory is non-volatile so you don't have to re-program your robot every time you power down, every operation takes only one clock cycle, and there is an active community of developers that use Atmel's microcontrollers.

The chip is currently running on a DT107 SimmStick board from DonTronics (http://www.dontronics.com/dt107.html ) that cost just $5 when I bought it but now costs $9. This board supports nearly all of the DIP microcontrollers made by Atmel.



On top of spending only $5 for the board Atmel sent me samples of their ATmega8L chip. They sent 2 free chips very quickly (within one week of ordering).

There are several free development tools that are available for the Atmel chips. There is a GNU C compiler, a development suite (AVR Studio) which can be used to write and compile both assembly and C code (you must download and interface the C compiler separately), and a programmer (Pony Prog) for programming via the parallel port since AVR Studio only supports programming via the serial port. All these programs can be found at http://www.avrfreaks.com. In addition to all the software they have a lot of code

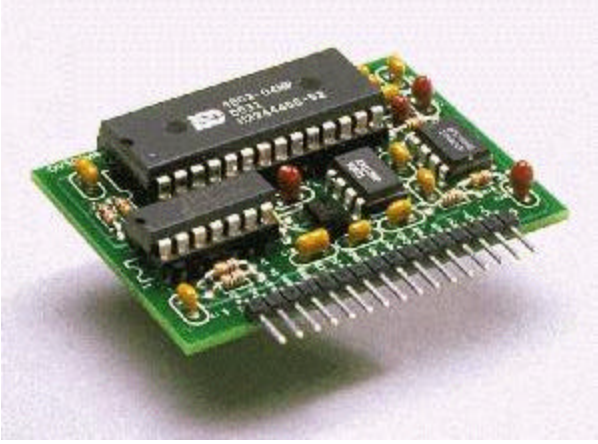
and designs available for download so you can see examples of other people's code. There is also an active community of very smart people using and discussing the Atmel chips on their forums. The mega8L is a newer chip so not many people have used it yet but there's still a lot of useful information there.

The most immediate problem with the DT107 board is that the output pins are useless for robots since it is the signal only and you will need power and ground to connect it to any electronics. It is necessary to build a header board to so you can bring out the signals that you need to interface IR sensors, etc. I built a header board using a small proto board from Radio Shack. The VCC from the board is not strong enough to drive the servos or the Sharp GP2D12 IR sensors so you will need a separate regulated voltage. Another problem with the DT107 is that it only supports the DIP version of the chips and there are it would be nice to be able to use the other packages since there are more pins.

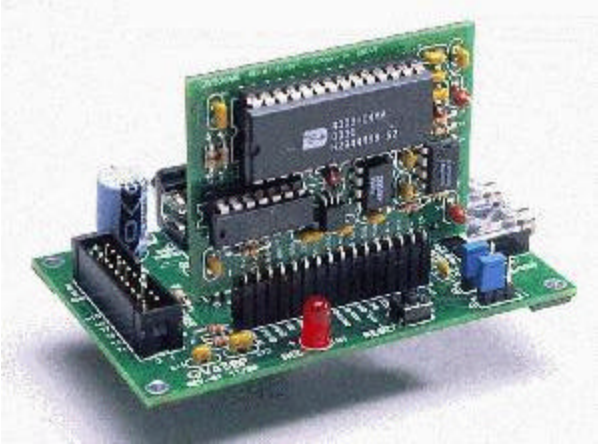
The most limiting factor of using the DIP version of this chip is that there are only 6 analog-digital converter channels available. If you need more than that you will need to interface an analog mux to expand the number of sensors you use.

## Sound Module

I ordered the Quadravox QV306m4 sound module ($57) from http://www.quadravox.com. I chose this module because it can store up to 240 different messages and 4 minutes of audio. There are also three different playback modes: direct, random, and looping. I used the direct and random modes. Direct was used when I wanted to play a specific sound clip, such as playing the Simpson's theme song at startup. Random mode is used while he is driving around searching for the TV.



The module comes with software used to program the chip via the PC. I bought the programming cradle (QV430P, $49: http://www.quadravox.com/qv430p.htm), which allows you to connect the sound module to the PC. The cradle is an optional purchase since you can design your own interface to the PC but I decided that it was worth $49 to not have to worry about it. The software and module support WAV files.



The module itself is interfaced with the Atmel chip through serial communications. The first step is to initialize the sound chip, i.e. Setting the playback mode and volume level on the amplifier. After this has been done all you have to do is send the byte code for the sound clip that you want to play.

## Remote Control

I bought a very cheap ($5) universal remote from Best Buy for HomerBot to use. Using a universal remote was important so that I wouldn't be limited to only one TV set. I hacked the remote so that HomerBot could access the Channel Up button when he wants to change the channel.

The other option for controlling the TV set would be to write code similar to the remote control lab from EEL4744. I didn't go this route because I felt that the universal remote would be much more reliable.

## Mobile Platform

My robot has no need for complicated mechanics so the platform is very simple. It consists of two round platforms for holding the electronics and wheels.

- 9” diameter base containing wheels, navigation IR and bump sensors, microcontroller/header board, sound module, and switches.
- 9” diameter raised platform containing the speaker, beacon detector IR and the remote control.

## Actuation

Actuation largely serves to give HomerBot his basic functions and personality, such as movement, speech, and the ability to “veg out”.

- The servos are driven by the microprocessor. There are two servos and they are designed solely for navigation.
- The sound module is activated when specified situation are encountered, giving HomerBot his personality.
- The remote control is activated and controlled when HomerBot goes into TV-watching mode.

## Sensors

- IR is used for obstacle avoidance and navigation.
- Bump sensors are used to detect if an obstacle has been encountered.
- An IR beacon is used for HomerBot to be able to locate the TV. There are two IR detectors, separate from the navigation IR sensors, that are used to detect the beacon.

## Behaviors

- The default mode is “search” until he locates the beacon so that he can control the TV.
- Once the TV is found he begins to surf the TV channels, until he stops and randomly decides if he “likes” the channel that he has found. If he does he stops surfing, otherwise he continues surfing until he finds a channel that he likes. He can also become bored with watching the TV. At the point he turns away from the TV and begins driving around again.
- He has several (currently 18) sound clips saved that will be accessed and played when appropriate situations are encountered. Instances of this would be HomerBot saying “DOH!” when he runs into something or “WOOHOO” when he finds a channel that he likes.

## Experimental Layout and Results

I don’t have any special sensor experimental results since I added special peripherals instead. I could put the operation levels for the IR and bump sensors as those are the only parts of HomerBot that have give feedback to the microprocessor, but this is well documented already.

## Conclusion

I am very happy with the way HomerBot has turned out. At the beginning of the semester I said that my goal was to create a robotic Homer Simpson and I think I have done that. The best proof of this is that HomerBot always gets a laugh out of people. He sings, he burps, he comments on life, he quotes movies, and he even talks about how much he loves Duff beer.

His method of finding the TV isn't as nice as it could be since he simply searches randomly for the beacon, which doesn't have the greatest range since it is IR. Developing and an accurate positioning system is enough for an entire project by itself so the IR beacon is the most realistic answer to the problem of finding the TV.

I am happy with my choice of the Atmel Atmega8L chip, but there are situations in which other chips would be a better solution. I would not use this chip on a robot that has a need of more than 6 analog-digital converter channels or more than three servos. Other than that this chip should be adequate. The most serious problem that I ran into using this chip was not having serial communications back to the PC. The chip supports this but I could not get a working RS-232 interface working correctly. Since I had no communications with the PC it was impossible for me to print any results to the computer screen. This made debugging code very difficult since I had to output all my data to LED arrays if I wanted to see it.

# Documentation

**ATmega8(L) Preliminary (Complete)** (247 pages, updated 12/01)

http://www.atmel.com/atmel/acrobat/doc2486.pdf

**QV306m4 Sound Module**

http://www.quadravox.com/qv306.pdf

# Appendices

```
/****************************************
 * Homer Bot Code
 * Rev. 1.8 3-14-02
 *      written by Don Lair
 ****************************************/

#include <io.h>
#include <wdt.h>
#include <stdlib.h>

#define SERVO_FREQ_H 0x27
#define SERVO_FREQ_L 0x10
#define ZERO 0x00
#define HALF_DUTY_H 0x23
#define HALF_DUTY_L 0x28
#define MAX_SPEED_Right_H 0x22
#define       MAX_SPEED_Right_L 0x60
#define nMAX_SPEED_Right_H 0x27
#define       nMAX_SPEED_Right_L 0x09
#define MAX_SPEED_Left_H 0x27
#define       MAX_SPEED_Left_L 0x09
#define nMAX_SPEED_Left_H 0x1E
#define       nMAX_SPEED_Left_L 0x78
#define HALF_SPEED_Right_H 0x22
#define HALF_SPEED_Right_L 0x90
#define HALF_SPEED_Left_H 0x25
#define HALF_SPEED_Left_L 0x28


typedef unsigned char u08;

u08 ir_right;
u08 ir_left;
u08 bump;
u08 beacon_right;
u08 beacon_left;

void ad_init(void)
{
      outp(0x86, ADCSR);
}

void servo_motor_init(void)
{
      outp(0xF0, TCCR1A);       /* Set output compare OC1A/OC1B clear on
compare match, */
```

```
	outp(0x12, TCCR1B);		/* store TOP in ICR1, and prescaler
Clk(I/O)/8. */
	outp(SERVO_FREQ_H, ICR1H);		/* Set TOP to match 20ms period
*/
	outp(SERVO_FREQ_L, ICR1L);

	sbi(DDRB, PB1);			/* Set output direction on PORT B for
output compare pin A */
	sbi(DDRB, PB2);			/* Set output direction on PORT B for
output compare pin B */
	outp(HALF_DUTY_H, OCR1AH);	/* Set PWM to 50% duty cycle on channel
A */
	outp(HALF_DUTY_L, OCR1AL);	/* servo should be calibrated to this
reference value */
	outp(HALF_DUTY_H, OCR1BH);	/* Set PWM to 50% duty cycle on channel
B */
	outp(HALF_DUTY_L, OCR1BL);	/* servo should be calibrated to this
reference value */
}

u08 analog(u08 channel)
{
	u08 sample_val_L, sample_val_H;
	outp(channel, ADMUX);

	sbi(ADCSR, ADSC);		/* begin conversion */
	loop_until_bit_is_set(ADCSR, ADIF);

	sample_val_L = inp(ADCL); /* get low 8 bits */
	sample_val_H = inp(ADCH);/* high 2 bits, not used */

	sbi(ADCSR, ADIF);		/* clear ADC interrupt flag */

	return sample_val_H;
}

void motor_speed(int mot_num, int speed){

	if(mot_num == 0){

		if(speed == 100){
			outp(MAX_SPEED_Right_H, OCR1AH);
			outp(MAX_SPEED_Right_L, OCR1AL);
		}

		else if(speed == -100){
			outp(nMAX_SPEED_Right_H, OCR1AH);
			outp(nMAX_SPEED_Right_L, OCR1AL);
		}

		else if (speed == 50){
			outp(HALF_SPEED_Right_H, OCR1AH);
			outp(HALF_SPEED_Right_L, OCR1AL);
```

```
		}

		else {
			outp(HALF_DUTY_H, OCR1AH);
			outp(HALF_DUTY_L, OCR1AL);
		}

	}
	else if(mot_num == 1){

		if(speed == 100){
			outp(MAX_SPEED_Left_H, OCR1BH);
			outp(MAX_SPEED_Left_L, OCR1BL);
		}

		else if(speed == -100){
			outp(nMAX_SPEED_Left_H, OCR1BH);
			outp(nMAX_SPEED_Left_L, OCR1BL);
		}

		else if (speed == 50){
			outp(HALF_SPEED_Left_H, OCR1BH);
			outp(HALF_SPEED_Left_L, OCR1BL);
		}

		else {
			outp(HALF_DUTY_H, OCR1BH);
			outp(HALF_DUTY_L, OCR1BL);
		}

	}


	else {
		outp(HALF_DUTY_H, OCR1AH);	/* Set PWM to 50% duty cycle on
channel A */
		outp(HALF_DUTY_L, OCR1AL);	/* servo should be calibrated to
this reference value */
		outp(HALF_DUTY_H, OCR1BH);	/* Set PWM to 50% duty cycle on
channel B */
		outp(HALF_DUTY_L, OCR1BL);	/* servo should be calibrated to
this reference value */

	}

	return;
}

void read_sensors(void)
{
	ir_right = analog(0x60);
	ir_left = analog(0x61);
	bump = analog(0x62);
```

```
	beacon_right = analog(0xE3);
	beacon_left = analog(0xE5);
}

void Wait_opt(int time)
{
	volatile int a, b, c, d;

	for (a = 0; a < time; ++a) {
		wdt_reset();
		for (b = 0; b < 10; ++b) {
			for (c = 0; c < 66; ++c) {
			d = a + 1;
			}
		}
	}
	return;
}

void USART_Init(void)
{
	/* Set baud rate */
	outp(0x00, UBRRH); // 9600 baud @ 8MHz
	outp(0x33, UBRR);

	/* enable receiver and transmitter */
	outp((1<<RXEN)|(1<<TXEN), UCSRB);
	outp(0x8E, UBRRH);
}

void USART_Transmit(unsigned char data)
{
	/* wait for empty tansmit buffer */
	loop_until_bit_is_set(UCSRA, UDRE);
	outp(data, UDR);
}

void sound_init(void)
{
	USART_Transmit(0xF0);
	loop_until_bit_is_set(PIND, PIND2);
	USART_Transmit(0xFC);
	loop_until_bit_is_set(PIND, PIND2);
	USART_Transmit(25);
	loop_until_bit_is_set(PIND, PIND2);

}

void drive(void)
{
	sbi(PORTD, PD6);	// Search LED on
	cbi(PORTD, PD5);	// Surf LED off
```

```
            if((bump>0x30 && bump<0x3D)){               // check front right
bump
                  USART_Transmit(19);
                  loop_until_bit_is_set(PIND, PIND2);

                  motor_speed(0,-100);              // reverse
                  motor_speed(1,-100);
                  Wait_opt(625);
                  motor_speed(0,-100);              // super hard left
                  motor_speed(1,100);
                  Wait_opt(625);
            }
            if((bump>0x50 && bump<0x60)){         // check front center bump
                  USART_Transmit(19);
                  loop_until_bit_is_set(PIND, PIND2);

                  motor_speed(0,-100);              // reverse
                  motor_speed(1,-100);
                  Wait_opt(625);
                  motor_speed(0,-100);              // super hard left
                  motor_speed(1,100);
                  Wait_opt(625);
            }
            if ((bump>0x18 && bump<0x22)){        // check rear center bump
                  USART_Transmit(19);
                  loop_until_bit_is_set(PIND, PIND2);

                  motor_speed(0,100);               // forward
                  motor_speed(1,100);
            }
            if ((bump>0x75 && bump<0x80)){        // check front left bump
                  USART_Transmit(19);
                  loop_until_bit_is_set(PIND, PIND2);

                  motor_speed(0,-100);              // reverse
                  motor_speed(1,-100);
                  Wait_opt(625);
                  motor_speed(0,100);               // super hard right
                  motor_speed(1,-100);
                  Wait_opt(625);
            }

            if((ir_right>0x60) && (ir_left>0x00)){     // super hard left cases
                  motor_speed(0,-100);
                  motor_speed(1,100);
            }

            else if((ir_right<0x37) && (ir_left>0x60)){       /* super hard right
cases */
                  motor_speed(0,100);
                  motor_speed(1,-100);
            }
```

```
		else if((ir_right<0x50) && (ir_right>0x30) && (ir_left>0x00)){ /*
hard left cases */
			motor_speed(0,0);
			motor_speed(1,100);
		}

		else if((ir_right<0x30) && (ir_left<0x50) && (ir_left>0x30)){   /*
hard right cases */
			motor_speed(0,100);
			motor_speed(1,0);
		}

		else if((ir_right<0x50) && (ir_right>0x13) && (ir_left<0x19)){ /* left
cases */
			motor_speed(0,50);
			motor_speed(1,100);
		}

		else if((ir_right<0x19) && (ir_left<0x50) && (ir_left>0x13)){   /*
right cases */
			motor_speed(0,100);
			motor_speed(1,50);
		}

		else if((ir_right<0x19) && (ir_left<0x19)){       /* straight cases */
			motor_speed(0,100);
			motor_speed(1,100);
		}

		else if((ir_right>0xA3) && (ir_left>0xA3)){       /* stuck cases */
			while(ir_left>60){
			motor_speed(0,100);
			motor_speed(1,-100);
			read_sensors();
			}
		}

}

void watch_tv(void)
{
	u08 val;
	if((beacon_right>0xB5) || (beacon_left>0xB5)){
		u08 j;

		cbi(PORTD, PD6);    // Search LED off
		sbi(PORTD, PD5);    // Surf LED on
		cbi(PORTD, PD4);    // Bored LED off

		motor_speed(0,0);
		motor_speed(1,0);
		USART_Transmit(0xF0);
		USART_Transmit(20);
```

```
		while((beacon_left>0xB0) || (beacon_right>0xB0))
		{
			read_sensors();
			val = inp(TCNT1L);			// generate random number
			if((val>120) && (val<255)){		// common case 1
				sbi(PORTD, PD7);		// change channel
				Wait_opt(50);
				cbi(PORTD, PD7);		// release button
			}
			else
				cbi(PORTD, PD7);		// release button

			Wait_opt(500);

		if(val<60 && val>0)
		{
			sbi(PORTD, PD6);	// Search LED off
			cbi(PORTD, PD5);	// Surf LED on
			sbi(PORTD, PD4);	// Bored LED off

			motor_speed(0,100);
			motor_speed(1,-100);
			Wait_opt(500);
			drive();
		}
		}
	}
}

int main(void)
{
	u08 i,j;
	i=0;
	j=0;
	u08 random, random2;

	outp(0x00, DDRC);		/* port c set to input */
	outp(0xFB, DDRD);

	cbi(PORTD, PD3);	// reset
	sbi(PORTD, PD3);	// reset pin high

	ad_init();
	servo_motor_init();
	USART_Init();
	sound_init();

	outp(0x01, TCCR2);  // enable clock to 8-bit counter
	USART_Transmit(21);			// play theme song
//	loop_until_bit_is_set(PIND, PIND2);

	while(i==0)
```

```
	{
		read_sensors();
		if(bump<0x22 && bump>0x18){
			i=1;
		}
		else	i=0;
	}

	for (;;)// infinite loop
	{

	read_sensors();

	drive();
	watch_tv();

	random = inp(TCNT1L);
	random2 = inp(TCNT2);
	if(random<10 && random>0 && random2<20 && random2>10)
USART_Transmit(1);
	if(random<25 && random>15 && random2<80 && random2>70)
USART_Transmit(2);
	if(random<70 && random>60 && random2<40 && random2>30)
USART_Transmit(3);
	if(random<130 && random>120 && random2<50 && random2>40)
USART_Transmit(4);
	if(random<210 && random>200 && random2<160 && random2>150)
USART_Transmit(5);
	if(random<120 && random>110 && random2<225 && random2>215)
USART_Transmit(6);
	if(random<250 && random>240 && random2<140 && random2>130)
USART_Transmit(7);
	if(random<110 && random>100 && random2<20 && random2>10)
USART_Transmit(8);
	if(random<90 && random>80 && random2<25 && random2>15)
USART_Transmit(9);
	if(random<15 && random>5 && random2<140 && random2>130)
USART_Transmit(10);
	if(random<65 && random>55 && random2<180 && random2>170)
USART_Transmit(11);
	if(random<35 && random>25 && random2<30 && random2>20)
USART_Transmit(12);
	if(random<85 && random>75 && random2<120 && random2>110)
USART_Transmit(13);
	if(random<180 && random>170 && random2<130 && random2>120)
USART_Transmit(14);
	if(random<230 && random>220 && random2<165 && random2>155)
USART_Transmit(15);
	if(random<160 && random>150 && random2<70 && random2>60)
USART_Transmit(16);
	if(random<140 && random>130 && random2<215 && random2>205)
USART_Transmit(17);
```

```
        if(random<190 && random>180 && random2<150 && random2>140)
USART_Transmit(18);

        }       // end infinite loop
}
```