University of Florida
Department of Electrical and Computer Engineering

EEL5666
Intelligent Machines Design Laboratory

Formal Report

Towbot

Table of Contents

Abstract

In light of recent events, automating airport baggage handling has tragically become a necessity. By eliminating baggage handling personnel, a human can be spared the tedious job of transporting baggage cars to and from the terminal. Towbot is the answer to this problem. As an autonomous mobile agent, Towbot is capable of performing a useful task for the airline industry. By utilizing feedback sensors, a Motorola 68HC11 micro-controller directs Towbot without human assistance.

Executive Summary

Overall, the Towbot project was a success. Firstly, I learned that I am capable of designing and implementing a functional mobile robot. Building a robot has been something that I have wanted to do since early childhood.   If I had it to do over again, I would have made a more elaborate body, instead of a Frisbee (which is ultimately what it is). I would have liked to have object manipulators on Towbot. Instead, it basically just moves around – just like most of the robots that have come before – and surely like many that will follow.

Secondly, building Towbot gave me the confidence that I am able to undertake a serious engineering problem, come up with a solution and then apply it.

Time management was my biggest shortcoming. I should have allocated more time to programming complex behaviors. This is not something that I am going to discontinue as soon as the semester is over. Towbot will be an ongoing project – a testing ground for research, and a trial platform for new ideas.

Introduction

Towbot is a small robot that is capable of following a black painted line, chasing the brightest light source, avoiding obstacles, leaping tall buildings in a single bound, and saving humanity from the forces of evil.

Platform

Original plans call for Towbot to be rectangular shaped, similar to conventional baggage tow trucks. However, a platform similar to Talrik was used. A round platform will make obstacle avoidance less of an issue because it will have a 0 degree turning radius. A rectangular vehicle has blind edges that may get caught on objects while maneuvering throughout the airport. Using a pre-designed platform will also save time and money in designing the robot. In retrospect, I wish I had built something bigger. The 10" diameter circle Towbot is based upon is a bit too small to house all of the electronic circuitry I wanted to squeeze underneath. By having all of the wiring, circuitry, and batteries underneath the main platform, Towbot was designed to have a simplistic appearance. Two Futaba S-1003 Servos were hacked to turn a full 360 degrees. Jack's Hobbies in Ocala, while expensive, had two really fun looking tires from an old RC car that Towbot used for locomotion.

A large magnet was housed in a wooden structure and securely attached to the rear or Towbot. This design proved flawed. All of the weight was resting on the ground in one spot. To make matters worse, there was not opportunity to put a caster under the rear end because the 8 battery Ni-Cad pack was also located in the rear. This put a lot of weight on the ground, and thus created a lot of friction. Future plans include a complete redesign of Towbot. An active actuating mechanism will be used to find the cargo vehicle. Towbot was supposed to find the target and back into it– passively latching on with the magnet.

Actuation

Two Futaba S3003 servo motors will provide locomotion for Towbot. By hacking the stop inside the motor, the unit is capable of 360 degree rotation. This makes it ideal for driving wheels.

A solenoid is being considered for use in the baggage car release mechanism. Further details will be available as the project proceeds.

Sensors

Modified 40kHz Sharp can-type IR sensors will be used in analog mode for obstacle avoidance. These emitters and detectors are placed strategically around the platform to give Towbot a broad field of vision.

Contact switches will be used to give Towbot a backup method of determining if it has collided with another object.

The baggage vehicle will be marked in a way that Towbot will be able to recognize it. Towbot will attach the baggage car to itself and tow it to the airport terminal. The terminal will be identified by an IR beacon operating at 29kHz. Pulse Code Modulation will be used to correctly identify the location of the terminal. The Axiom 68HC11 board will serve to modulate the coded IR signals on the beacon at the terminal.

Cadmium Sulfide light sensing photocells will give Towbot the ability to follow a black line on the ground, allowing it to quickly and efficiently find the baggage car.

One original sensor will be used to give Towbot feedback to know at all times what the status is f the car being towed.

Behaviors

Towbot will be able to follow a printed lane marker in the road. Also, it will find the baggage c ar, perform obstacle avoidance, and determine the direction of the terminal so that it may tow the car to the terminal. Once at the terminal, Towbot will drop off the cart and continue on its way.

Towbot, at the time of publication, can avoid obstacles with limited accuracy. By Wednesday, Towbot will be at the infantile stages of line following and light following.

Conclusion

Overall, Towbot was a success. The project needs work still, that will be done by Wednesday. A strong baseline has been established. Where the project goes from here is unknown – but should be interesting.

Documents / Appendix

Towbot C code.

```
/////////////////////////////////////////////
//Program:     towbot.c                    //
//Description: Towbot 2001 test/avoid porg. //
//Programmer:  Scott Kivitz                //
//Date:        October, 2001               //
/////////////////////////////////////////////

#include <stdio.h>  //Standard includes
#include <mil.h>
//#include <tkbase.h>
#include <hc11.h>
#include <analog.h>

extern void _start(void); //This MUST be placed here RIGHT after the includes, it is for
the reset vector

#define DUMMY_ENTRY (void (*)(void))0xFFFF //This is a void function declaration
for the interrupt vector table

#define PERIOD 30000  //User defines
#define amt    100
#define M1Maxf 3355                  //motor 1 max forward
#define M1Maxr 2775                  //motor 1 max Reverse
#define M2Maxf 2700
#define M2Maxr 3250                  //motor 2 max reverse
#define Center1 3025//motor 1 (toc2) center
#define Center2 2940//motor 2 (toc3) center

#pragma interrupt_handler TOC1_isr, TOC2_isr, TOC3_isr;  //This is where interrupt
service routines are specified

void init_pwm(void);   //Function Prototypes
void TOC1_isr(void);
void TOC2_isr(void);
void TOC3_isr(void);
void choices(void);
void avoid(void);
int IRcheckL(void);
int IRcheckR(void);
void checkAD(void);
void wait(void);
```

```
int motor1;           //Global variables
int motor2;
int AtoDflag;                           //flag gets set to 1 when initAD() is run, else 0
int time;


void main (void){     //Beginning of main routine



      init_pwm();
      init_analog();
      motor1 = Center1;  //stopped
      motor2 = Center2;  //stopped
  //    AtoDflag = 0;

      while(1){
                  //choices();   //test code for servos and IR sensors (CdS cells also)
                  avoid();
                  }
         } //end main

// ***********************************************
// *                    Choices                                             *
// ***********************************************
void choices(void){

                  char clear[]= "\x1b\x5B\x32\x4A\x04";
                  int menu;
                  int dc;
      dc = 0;                       //Initialize DC for while loop with garbage
           printf("%s", clear);
           printf("duty cycle 2 = %d ", motor1);
           printf("\nduty cycle 3 = %d", motor2);
           printf("\n\nWelcome to the TOWBOT Operating System\n");
                              printf("1) Motor 1 - TOC2 \n");
                              printf("2) Motor 2 - TOC3 \n");
                              printf("3) Center Servos [STOP!] \n\n\n");
                              printf("4) FULL FORWARD\n");
                              printf("5) FULL REVERSE\n");
                              printf("6) Spin CW\n");
                              printf("7) Spin CCW\n");
                              printf("8) A/D Control\n");

//                            while((menu !=0x31) || (menu !=0x32))
```

```
		menu = getchar();
		if(menu == 0x31)
		{	printf("\nYou've selected Motor 1");
				printf("\nPress 'a' to increase duty cycle or 'z' to decrease it");
		}
		if(menu ==0x32)
		{	printf("\nYou've selected Motor 2");
				printf("\nPress 'a' to increase duty cycle or 'z' to decrease it");
		}
		if(menu==0x33) //ascii for 3
		{
		 printf("Press [Enter] to center servos 1,2");
			motor1 = Center1;
			motor2 = Center2;
		}

//		while((dc != 0x61) || (dc != 0x7A))
		dc = getchar();

		if(menu==0x31) //ascii for 1
		 {
				if(dc==0x61) //ascii for a
				{
					motor1 = motor1+amt;
				}
				if(dc==0x7A) //ascii for z
				{
					motor1 = motor1-amt;
				}
			}

			if(menu==0x32) //ascii for 2
			{
				if(dc==0x61) //ascii for a
				{
					motor2 = motor2+amt;
				}
				if(dc==0x7A) //ascii for z
				{
					motor2 = motor2-amt;
				}

		 }

		if(menu==0x34) // full forward
		{
```

```
            motor1 = M1Maxf;
            motor2 = M2Maxf;
        }

        if(menu==0x35) // full reverse
        {
            motor1 = M1Maxr;
            motor2 = M2Maxr;
        }

        if(menu==0x36) // spin CW
        {
            motor1 = M1Maxf;
            motor2 = M2Maxr;
        }

        if(menu==0x37) // spin CCW
        {
            motor1 = M1Maxr;
            motor2 = M2Maxf;
        }

        if(menu==0x38) // ITS A -> D TIME!!! Howdy Doody!
        {
            checkAD();
        }


                    }//End choices
/* **************************************************
  *             checkAD                 *
  *
        *
  **************************************************
*/

void checkAD(void)
{
 int ir0;
 int ir1;
 int ir2;
 int ir3;
 int ir4;
 int ir5;
 int ir6;
 int ir7;
```

```
while(1)
        {
        ir0 = analog(0);
        ir1 = analog(1);
        ir2 = analog(2);
        ir3 = analog(3);
        ir4 = analog(4);
        ir5 = analog(5);
        ir6 = analog(6);
        ir7 = analog(7);

/*      printf("this is analog 0: ");
        printf("%u\t",ir0);
        printf("this is analog 1: ");
        printf("%u\t",ir1);
        printf("this is analog 2: ");
        printf("%u\t",ir2);
*/      printf("\tthis is analog 3: ");
        printf("%u\t",ir3);
        printf("\nthis is analog 4: ");
        printf("%u\t",ir4);
        printf("this is analog 5: ");
        printf("%u",ir5);
/*      printf("this is analog 6: ");
        printf("%u",ir6);
        printf("this is analog 7: ");
        printf("%u\t",ir7);
*/      time = 5000;
        wait();

        }

} //end checkAD


// ************************************************
// (*           init A/D                *
// *)                               *
// ()**********************************************0
//void initAD(void)
//{}
/*
    SET_BIT(OPTION, 0x80);
    time = 100;
    wait();
```

```
   ADflag = 1;
}
*/

// ***********************************************
// *          init_pwm                *
// *          --------               *
// * Initializes OCs and sets up stuff..          *
// ***********************************************
void init_pwm(void)    //Initialization function

            {

    INTR_OFF();    //Turns off interrupts

    SET_BIT(TMSK1, 0x80);  //Enable TOC1,2,3  (1000 0000)
    SET_BIT(TMSK1, 0x40);  //Enable TOC1,2,3  (0100 0000)
    SET_BIT(TMSK1, 0x20);  //Enable TOC1,2,3  (0010 0000)

            SET_BIT(TCTL1, 0x80);  //Set TOC2,3 Low on Int. (1000 0000)
            SET_BIT(TCTL1, 0x20);  //Set TOC2,3 Low on Int. (0010 0000)

    CLEAR_BIT(TCTL1, 0x40); // (0100 0000)
    CLEAR_BIT(TCTL1, 0x10); // (0001 0000)

 // OC1 STUFF
            SET_BIT(OC1D, 0x40);    // (Sets OC2,3 to go HI on OC1 interrupts)
            SET_BIT(OC1D, 0x20);      // (0110 0000)
            SET_BIT(OC1M, 0x40);     // Gives control of those pins to OC1
    SET_BIT(OC1M, 0x20);        // (0110 0000)

    INTR_ON();                    //Turns on interrupts? I guess...

            }//End init_pwm
// ***********************************************************
// *             avoid                   *
// * obstacle avoidance                          *
// ***********************************************************
void avoid(void)
{
      int holdwheel = 10000;

      time = 95000;  //approx 2 sec. delay before begin
      wait();
      wait();
      wait();
	wait();



	while(!(IRcheckL() || IRcheckR() ||Fbump() ))
	{
		motor1 = M1Maxf;
		motor2 = M2Maxf;
	}
	if(IRcheckL() )				// If Object on left, stop right wheel for
"holdwheel"
	{
		motor2 = M2Maxr;  //max reverse right motor
		time=holdwheel;
		wait();
		wait();
		wait();
		wait();
		wait();
		motor2 = Center2;
		motor1 = Center1;
		wait();
		wait();
		wait();
		motor1 = M1Maxf;
		motor2 = M2Maxf;

	}
	if(IRcheckR() )
	{
		motor1 = M1Maxr;
		//motor2 = Center2;
		time=holdwheel;
		wait();
		wait();
		wait();
		wait();
		wait();
		motor1 = Center1;
		motor2 = Center2;
		wait();
		wait();
		wait();
		motor1 = M1Maxf;
		motor2 = M2Maxf;
```

```
}
if(IRcheckR() && IRcheckL() )
{
        motor1 = Center1;
        motor2 = Center2;
        time = holdwheel;
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = M1Maxr;
        motor2 = M2Maxr;
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = Center1;
        motor2 = Center2;
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = M1Maxf;
        motor2 = M2Maxr;
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = Center1;
        motor2 = Center2;
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = M1Maxf;
        motor2 = M2Maxf;

}

if(Fbump() )
{
        motor1 = Center1;
        motor2 = Center2;
        time = holdwheel;
        wait();
        motor1 = M1Maxr;
        motor2 = M2Maxr;
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = Center1;
        motor2 = Center2;
        wait();
        motor1 = M1Maxf;
        motor2 = M2Maxr;
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        wait();
        motor1 = Center1;
        motor2 = Center2;
        wait();
        motor1 = M1Maxf;
        motor2 = M2Maxf;

}
```

```
}//end avoid

//*********************************************
//                    IRcheckL                    *
//*********************************************
int IRcheckL(void)
{
	//this function checks the IR detectors and returns a 0 or 1
	int L1;
	L1 = (analog(4) > 90);
	return L1;
}

//*********************************************
//                    IRcheckR                    *
//*********************************************
int IRcheckR(void)
{
	int R1;
	R1 = (analog(5) > 107);
	return R1;
}


//**********************************************
//                bump!
//**********************************************
int Fbump(void)
{
	int Fb;
	Fb = (analog(3) == 0);
	return Fb;
}
//******************************
//     WAIT
//******************************
void wait(void)
{
  int i;
    for (i = time; i > 0; i--);
}


//*********************************************************
//        TOC1 ISR
	*
```

```
//
                                    *
//************************************************************
void TOC1_isr(void)  //This is the interrupt service routine
         {
         CLEAR_FLAG(TFLG1, 0x80); //Clears OC1 Flag (1000 0000)
         TOC2 = TOC1+motor1;
         TOC3 = TOC1+motor2;
         TOC1 = TOC1+PERIOD;

         }


//************************************************************
//                 TOC2 ISR
         *
//
                                    *
//************************************************************
void TOC2_isr(void)  //This is the interrupt service routine
         {
         CLEAR_FLAG(TFLG1, 0x40); //Clears OC2 Flag

         }



//************************************************************
//                                                TOC3 ISR
//
//************************************************************
void TOC3_isr(void)
         {
         CLEAR_FLAG(TFLG1, 0x20); //Clears OC3 Flag

         }




#pragma abs_address:0xffd6
/* change the above address if your vector starts elsewhere */

void (*interrupt_vectors[])(void) =
      {
      DUMMY_ENTRY,   /* SCI, RS232 protocol */
      DUMMY_ENTRY,   /* SPI, high speed synchronous serial*/
      DUMMY_ENTRY,   /* Pulse accumulator input edge */
      DUMMY_ENTRY,   /* Pulse accumulator overflow */
      DUMMY_ENTRY,   /* Timer overflow */
      DUMMY_ENTRY,   /* TOC5 */
      DUMMY_ENTRY,   /* TOC4 */
      TOC3_isr,      /* TOC3 */
      TOC2_isr,      /* TOC2 */
      TOC1_isr,      /* TOC1 */
      DUMMY_ENTRY,   /* TIC3 */
      DUMMY_ENTRY,   /* TIC2 */
      DUMMY_ENTRY,   /* TIC1 */
      DUMMY_ENTRY,   /* RTI */
      DUMMY_ENTRY,   /* IRQ */
      DUMMY_ENTRY,   /* XIRQ */
      DUMMY_ENTRY,   /* SWI */
      DUMMY_ENTRY,   /* ILLOP */
      DUMMY_ENTRY,   /* COP */
      DUMMY_ENTRY,   /* CLMON */
        _start       /* RESET */
      };

#pragma end_abs_address
```