Rande T. Enderby

Dave

Table of Contents

Abstract

The name of my robot is dave. He will take a drawing or writing downloaded from a computer, written using a mouse. He will then take that image and begin to search for his "chalk board". When he has found his "chalk board", he will reproduce the image using a CO_2 laser. Once the "chalk board" is found, an LED will light, indicating to me that he is ready to draw. I will then manually turn the laser on, via a panic trigger (as suggested by Dr. Swartz). Dave will have four major modes of operation: obstacle avoidance/search, line following, mirror calibration and drawing mode. His "chalk board" will basically a piece of plywood, backed with meddle, if need be.

Executive Summary

The final product of dave was not what I wanted. I had obstacle avoidance and mirror calibration working several weeks ago, and could even burn an “L” on a piece of paper. However, two nights before the demo day, things started going wrong. The batteries started going which started a landslide of problems. It took me two days to crawl back to the same spot I had gotten to before. At this point, I could move the laser beam up and over, but it seemed to be jumpy and not very smooth. With the suggestion by Dr. Swartz, I got ride of my timing loops, and tried using the RTI system. I had very little luck with that, and could not get the RTI to interrupt. Instead, I used OC1. However, my new problem was that after about 5-15 steps, the processor would automatically reset itself. I tried isolating things onto their own batter packs, but that seemed to have no effect on my problem. I thought that maybe the He-Ne laser diode was pulling too much current, so I added a 150 Ohm resistor, but this too had no effect. Any higher resistor values would result in the laser pointer not lighting at all. Finally the last straw was when my laser pointer laser pointer blew the morning of media day. Without that, it made it impossible to test the writing capability without having to turn the laser on.

I honestly feel that this project was doable, but I just needed more time. The concept is rather simple, but problems started plaguing me. Also having to go home 3 out of 4 weekends in November certainly didn’t help things.

Introduction

This paper will walk you through the different areas of dave. I will discuss the Integrated Systems, and how dave will receive his pattern. The physical design and materials used for the Mobile Platform. I will then discuss the Actuation, and his means of locomotion. I will also discuss which sensors I will use, and his behaviors. I will also show hand drawings of the artists' rendition.

MAIN

Integrated Systems

Dave will be controlled by the MRC11, using the 68HC11F1. The pattern will be drawn with a mouse in a Java program that was written by David Pinto. The program records the pattern as X-Y coordinates. I will then clean up the output, and download it to dave. The pattern will then be stored in memory, and will reference a mirror position. See Appendix C for David's code.

Mobile Platform

The platform will be made out of wood. I wanted to design it using AutoCad, but due to the physical dimensions, was unable to. The base will be 35 inches long, by 12 inches wide. The laser will be propped up backwards at about a 20 degree angle. There will be two supports holding the laser up. There will also be another platform housing the stepper motors and mirror. Then two square rods with fasten able wing nuts at the end, so securely hold the laser in place.

Actuation

The four wheels will be controlled by servos. I will use the servos from www.balarc.com, 208 oz-in. I will partially hack them, to allow them to fully rotate 360 degrees. I have estimated dave to be about 30 lbs., since the power supply weighs about 25lbs., and the laser a few pounds. The servos will be connected to 4 inch wheels, which are filled with foam instead of air or rubber. The wheels will operate tank style. The left two will move together, and the right two will move together.

Sensors

To perform the obstacle avoidance, dave has two IR sensors (32kHz) located on front left and right, and two in the rear. He also has two slotted interrupt sensors attached the rotating X and Y axis, for mirror calibration. This was necessary to insure the mirror starts at the same location every time, so no writing exceeds the boundaries of the “chalk board”. Dave also have 3 CdS cells to perform line following.

Behaviors

The main goal of dave is to write his pattern on his “chalk board”. He will accomplish this be using a CO_2 laser etching into wood. The laser beam will bounce and be controlled by a mirror. The mirror will be able to move 180 degrees up or down, and 180 degrees left or right. I will use two stepper motors, one to control X coordinates, and one to move Y coordinates. The X motor would be mounted to a circular dowel 3 inches long. Mounted on the side of that, would be a motor attached to the mirror, to control the Y axis.

Experimental Layout and Results

I was able to get rough obstacle avoidance working and mirror calibration. The platform was built with line following capabilities, but that mode was not completed due to time restraints. I was able to move the mirror in both the X and Y directions, but writing was not completed. I was able to read in a set of coordinates, and determine whether one step left, right, up or down was needed, but after only a few steps, the board would reset itself.

I found that when writing very fast was easy to do, but when I was taking one step and waiting for a second, problems started to arise.

Conclusion:

I believe that dave is a feasible robot design. However, I am not sure I will be able to control the mirror well enough to be able to write clearly. Also, the laser is low wattage to reduce the risk of accident, so that means that I will have to leave it on a surface for a longer period of time. Assuming the laser pulls 1A, I will have limited laser on-time. I will also have to find a way to secure the laser from moving around, otherwise the beam may not line up with the mirror.

Appendix A

(Dave’s main code)

```
//rande t. enderby
//code for dave
//wheel movement and obstacle avoidance; mirror calibration and
//drawing

#include <stdio.h>  //Standard includes
#include <mil.h>
#include <hc11.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 atod1 ADR1            //port A0 and A4
#define atod2 ADR2            //port A1 and A5
#define atod3 ADR3            //port A2 and A6
#define atod4 ADR4            //port A3 and A7
#define six_max 0x80
#define siy_max 0xC8    //y interrupt blocked=200

#pragma interrupt_handler TOC1_isr, TOC2_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 avoid(void);
void init_rti(void);
void init_atod(void);
void cal_x_axis(void);
void cal_y_axis(void);

void init_portd(void);

int duty2;             //Global variables
int duty3;
int high4;
int x,y;
int tol;
static int times = 0;
static int overal = 0;
static int pos = 0;
int coords[64][2] = {
6,12,
7,13,
8,14,
6,15,
6,16,
7,17,
7,18,
8,19,
8,20,
8,21,
8,23,
8,24,
8,25,
8,26,
8,27,
8,28,
9,34,
10,34,
11,36,
12,37,
13,39,
14,39,
15,40,
16,40,
17,40,
18,40,
19,40,
20,40,
21,40,
22,39,
23,38,
24,37,
24,36,
25,34,
25,33,
25,32,
25,31,
24,31,
23,30,
22,29,
21,29,
20,29,
20,28,
19,27,
19,26,
19,25,
19,23,
19,21,
19,20,
20,20,
21,19,
22,19,
23,19,
24,19,
24,18,
25,18,
26,17,
27,17,
27,16,
29,15,
30,15,
31,15,
32,15,
32,14,
};

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

        char clear[]= "\x1b\x5B\x32\x4A\x04";
        int dc;


        init_atod();
        init_portd();
        avoid();
        cal_x_axis();
        cal_y_axis();


        duty2 = 3000;  //stopped
        duty3 = 3000;  //stopped
        dc = 0;                     //Initialize DC for while loop with
                                    garbage
        tol = 10;
        high4 = 0;            //set atod low 4
while (1)  {
      init_pwm();
}//End main

void cal_x_axis(void)  {            //calibrate x axis to starting
position
      SET_BIT(PORTD, 0x08);         //set pd3
      SET_BIT(PORTD, 0x04);         //set pd2   FORWARD
      SET_BIT(PORTD, 0x10);         //set pd4, and enable y7
      high4 = 1;                    //use high 4 atod ports
      init_atod();
      while (atod1 < six_max) {     //ad5
            CLEAR_BIT(PORTA, 0x10);       //set pa4 low
            for (x=0; x < 1000; x++);
            SET_BIT(PORTA, 0x10);         //set pa4 high
      }

}  //end cal_x_axis

void cal_y_axis(void)  {            //calibrate y axis
      CLEAR_BIT(PORTD, 0x08);       //clear pd3
      CLEAR_BIT(PORTD, 0x04);       //clear pd2
      CLEAR_BIT(PORTD, 0x10);       //set pd4 and direction, y1
      high4 = 1;                    //use high 4 atod ports
      init_atod();
      while (atod2 < siy_max) {     //ad6
            CLEAR_BIT(PORTA, 0x08);       //set oc5 low
            for (x=0; x < 1000; x++);
            SET_BIT(PORTA, 0x08);         //set oc5 high
      }
      CLEAR_BIT(PORTD, 0x08);       //set pd3
      CLEAR_BIT(PORTD, 0x04);       //set pd2
      SET_BIT(PORTD, 0x10);         //set pd4, and enable y7
      for (y=0; y<100; y++) {         //back out of si sensor
            CLEAR_BIT(PORTA, 0x08);       //set pa5 low
            for (x=0; x < 1000; x++);
            SET_BIT(PORTA, 0x08);         //set pa5 high
      }
}  //end cal_y_axis


void avoid(void)  {                 //obstical avoidance
      while((atod1 > 80) || (atod2 > 80)){
            while (((atod1 >= 80) && (atod1 < 110)) || ((atod2 >= 80)
&& (atod2 < 100))) {
                  duty2 = 2900;     //slow down
                  duty3 = 3100;
            }
            while (((atod1 >= 110) && (atod1 < 130)) || ((atod2 >= 110)
&& (atod2 < 130))) {
                  duty2 = 2950;
                  duty3 = 3050;     //slower down
            }
            if ((atod1 >= 130) || (atod2 >= 130)) {
                  duty2 = 3000;
                  duty3 = 3000;     //full stop
                  while (((atod1 - atod2) > tol))&& ((atod3 > 120) &&
(atod4 > 120))) {
                        duty3 = 2950;
                        duty2 = 4000;     //backup left
                  }
                  while (((atod2 - atod1) > tol)) && ((atod3 >120) &&
(atod4 > 120))) {
                        duty2 = 3050;
                        duty3 = 1000;     //backup right
                  }
           }
}
}  //end avoid


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(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)

 // OC1 STUFF
            SET_BIT(OC1D, 0x40);    // (Sets OC2,3 to go HI on OC1
interrupts)
```

```
            SET_BIT(OC1M, 0x40);    // Gives control of those pins to
            OC1
            CLEAR_BIT(OC1M, 0x08);

//setup oc5 stuff
            CLEAR_BIT(PACTL, 0x04);

            INTR_ON();              //Turns on interrupts

            }//End init_pwm


void init_atod(void)  {
      if (!high4)  {
      SET_BIT(OPTION, 0x80)         //turn on a to d system
      CLEAR_BIT(ADCTL, 0xFF)        //multi=scan=0, first 4 channels
      SET_BIT(ADCTL, 0x20)          //scan=1
      SET_BIT(ADCTL, 0x10)       //multi=1

      CLEAR_BIT(ADCTL, 0x04)        //multi=scan=0, first 4 channels
      }
    else if (high4)
            SET_BIT(ADCTL, 0x04);         //second 4 channels
}

void init_portd(void)  {
      SET_BIT(DDRD, 0xFF);          //set all ports to output
}

//*****************************************************************
//                     TOC1 ISR
//
//*****************************************************************
void TOC1_isr(void)  //This is the interrupt service routine
      {

      CLEAR_FLAG(TFLG1, 0x80); //Clears OC1 Flag (1000 0000)
      TOC2 = TOC2+ duty2
      TOC3 = TOC3 + duty3
      TOC1 = TOC1+PERIOD;
}  //end oc1_isr


//*****************************************************************
//                     TOC2 ISR
            *
//
                        *
//*****************************************************************
void TOC2_isr(void)  //This is the interrupt service routine
      {
      CLEAR_FLAG(TFLG1, 0x40); //Clears OC2 Flag
      if (times > 90)  {
            if (coords[pos][0] > coords[pos-1][0])  {
                  SET_BIT(PORTD, 0x08);         //set pd3
                  SET_BIT(PORTD, 0x04);         //set pd2   FORWARD
                  SET_BIT(PORTD, 0x10);         //set pd4, and enable
y7
                  SET_BIT(PORTA, 0x10);         //set pa4 high
                  for (y=0; y < 100; y++);
                  CLEAR_BIT(PORTA, 0x10);       //set pa4 low
            }
            else if (coords[pos][0] < coords[pos-1][0])  {
                  CLEAR_BIT(PORTD, 0x08);       //clear pd3
                  CLEAR_BIT(PORTD, 0x04);       //clear pd2  REVERSE
                  CLEAR_BIT(PORTD, 0x10);       //set pd4 and
direction, y1
                  SET_BIT(PORTA, 0x10);         //set pa4 high
                  for (y=0; y < 100; y++);
                  CLEAR_BIT(PORTA, 0x10);       //set pa4 low
                  }
            else if (coords[pos][0] == coords[pos-1][0]);
            if (coords[0][pos] > coords[0][pos-1])  {
                  CLEAR_BIT(PORTD, 0x08);       //clear pd3
                  CLEAR_BIT(PORTD, 0x04);       //clear pd2
                  SET_BIT(PORTD, 0x10);         //FORWARD

                  SET_BIT(PORTA, 0x08);         //set oc5 high
                  for (y=0; y < 100; y++);
                  CLEAR_BIT(PORTA, 0x08);       //set oc5 low
            }
            else if (coords[0][pos] < coords[0][pos-1])  {
                  CLEAR_BIT(PORTD, 0x08);       //clear pd3
                  CLEAR_BIT(PORTD, 0x04);       //clear pd2
                  CLEAR_BIT(PORTD, 0x10);       //set pd4 and
                  SET_BIT(PORTD, 0x10);         //FORWARD
                  SET_BIT(PORTA, 0x08);         //set oc5 high
                  for (y=0; y < 100; y++);
                  CLEAR_BIT(PORTA, 0x08);       //set oc5 low
            }
            else if (coords[pos][x] == coords[pos][x-1]);

            times = 0;
            pos++;
          }  //end for
          times++;
      }



//*****************************************************************
//                                  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 */
        DUMMY_ENTRY,   /* 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
```

Appendix B

(David Pinto’s Code)

```
import javax.swing.*;
import java.util.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;


 class RandyFrame extends JFrame
{
   int myWidth = 500;
   int myHeight = 500;

   Vector coordinates = new Vector();

    RandyFrame()
   {
      Container container = getContentPane();
      container.setLayout(null);

   //Create Drawing Panel
      MyPanel thePanel = new MyPanel();
      thePanel.addMouseMotionListener(
             new MouseMotionAdapter()
            {
                public void mouseDragged(MouseEvent e)
               {
                  int x = e.getX();
                  int y = e.getY();

                  coordinates.add(new MyCoordinate(x,y));
                  repaint();
                  System.out.println(x + ", " + (myHeight - y));
               }
            });
      thePanel.setBounds(10, 10, myWidth, myHeight);
      container.add(thePanel);

   //AddButton To Frame
      JButton button = new JButton("Save");
      button.addActionListener(
             new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
               {
               //Create File Output Stream and PrintWriter
                  try
                  {
                     FileOutputStream fos = new FileOutputStream("data.txt");
                     PrintWriter pr = new PrintWriter(fos);
                  //Write Entries To File
                     for (int i = 0;i < coordinates.size();i++)
                     {
                       MyCoordinate temp = (MyCoordinate)coordinates.elementAt(i);

                     //EDIT HERE
                     //pr.println()
                     //pr.print()
                       pr.println(temp.x + "," + (myHeight - temp.y));

                     }

                  //Flush OutputStreams and Close
                     pr.flush();
                     fos.flush();
                     fos.close();
                  }
                     catch (FileNotFoundException ee) {System.out.println(ee);}
                     catch (IOException ee) {System.out.println(ee);}
               }
            });
      button.setBounds(10, myHeight + 75, 100, 25);
      container.add(button);

   //Set Frame Attributes
      setSize(myWidth + 60, myHeight + 60);
      show();
   }

    class MyPanel extends JPanel
   {
      public void paint(Graphics g)
      {
        g.setColor(new Color(0, 0, 0));
        g.fillRect(0, 0, getWidth(), getHeight());

        g.setColor(new Color(255, 0, 0));
        for (int i = 1; i < coordinates.size();i++)
        {
            MyCoordinate a = (MyCoordinate)coordinates.elementAt(i-1);
            MyCoordinate b = (MyCoordinate)coordinates.elementAt(i);
            g.drawLine(a.x, a.y, b.x, b.y);
        }
      }
   }

    class MyCoordinate
   {
      int x, y;
      MyCoordinate(int x, int y)
      {
        this.x = x;
```

```
         this.y = y;
      }
   }

//Begins Here
    public static void main(String args[])
   {
      new RandyFrame();
   }
}
```