Hamelin

IMDL Spring 2005

Matt King

Table of Contents

Abstract

My robot performs different movements based on which musical note it perceives. It has bump sensors to keep it from running into obstacles and an IR sensor to keep it from falling off of surfaces.

Executive Summary

My robot is designed to discern between five different musical pitches. I developed a tone detection circuit that interfaces to an Atmega 128 microcontroller by Atmel. My software controls the motors based on which note it perceives. I have bump switches that will stop the motors if it runs into obstacles and an IR sensor that will keep it from falling off of surfaces. It can also use the IR to track a black line.

Introduction

This is my report on Hamelin. I chose the name Hamelin after the fairy tale of the Pied Piper of Hamelin. In the story he leads all of the rats out of town of Hamelin by playing his musical instrument. My robot moves according to different musical pitches played by an instrument. I developed a tone detection circuit that will discern between five different notes. I have one for reverse, forward, turn left, turn right and brake. The robot looks like a rat and it has bump switches for its whiskers and tail. I also placed an IR sensor on its nose to track a line or detect the edge of a surface.

Integrated System

My robot is controlled by an Atmega 128 microprocessor designed by Atmel. My development board was created by Mekatronix. My robot operates by detecting certain musical pitches and then moves according to which one is played. The tone generation is done by my alto saxophone. I used a microphone to feed a bank of 567 tone decoders tuned to different frequency ranges as a sensor for the microprocessor. My software detects which output is true and then turns the motors in the appropriate direction. It also has bump sensors in several places to avoid breaking the robot frame. I also included a sensor to make sure the robot will not fall off of any surfaces.

Mobile Platform

I designed the body of my robot to look like a rat so that the person controlling the robot would be like the Pied Piper of Hamelin. I started off with a rectangular 12x8 inch design in AutoCad. I cut out wheel wells and I curved the front to look like a nose. I designed motor mounts and attached them to the platform. I simply placed my tone detection circuit, motor driver, batteries, and microcontroller board on the platform. I put bump sensors with coat hangers attached on the front for whiskers and one in the back for a tail. I built a cover in AutoCad and had Justin Robinson help me fit it to the platform. I painted it orange and blue for the commercial that the University made of my robot.

Actuation

I used two D.C. gear head motors controlled by a dual H-Bridge motor driver. I ordered both from Lynxmotion. The motors are Hsiang Neng 200rpm 30:1. The motor driver board was designed and distributed by Lynxmotion. I found these very easy to use and I would recommend them to future students.

I used the unregulated power rail on my microcontroller board to power the motors. I used the non-inverted PWM channels on the Atmega128 to generate various waveforms. For forward, and reverse I used 50% duty cycles, and 25% for the turns. I had two wheels in the back and a caster in the front. When I turned right I held the right motor and rotated the left motor counter clockwise. When I turned left I held the left motor and rotated the right motor clockwise.

Sensors

I created a special sensor using a microphone and a bank of 567 chips. Each tone decoder chip is an input to the microprocessor. When a particular note is detected the robot moves in the appropriate direction. I used five different notes: one for reverse, forward, turn left, turn right, and brake.

I used bump sensors on the nose and tail of the robot so that it does not crash into any objects.

I used an IR sensor so that the robot stops its motors before it falls off of any surface. It also uses the same sensor to detect a black line.

Behaviors

My robot waits for notes to be played and then generates an interrupt once one is detected. The interrupt routine sets a global variable and then initializes an ouput compare counter sequence to eliminate bouncing. Once there is a compare match the OC interrupt routine will check the global variables for the pitches. It will then move in the appropriate direction. I have five notes that control the motors to move forward, reverse, turn left, turn right and stop. The interrupts simply set the motor direction and speed. So they will continue to drive until an obstacle or brake note is detected. I have two different modes: Freestyle and Instructional.

Freestyle: In this mode the robot roams around freely responding to whatever notes it picks up. If it bumps into an obstacle it will stop. Also my robot will need to be held in the air upon reset in order to calibrate the IR sensor for edge of the world detection. If the IR sensor receives no reflection while the robot is driving the motors will brake immediately.

Experimental Layout and Results

I did very thorough experiments on my tone detection circuit before I had a satisfactory design. At first I had one 567 chip to see if these devices would work at all. I discovered that they could be tuned to activate upon the detection of particular frequencies. The data sheet didn't say anything about precision tuning with a potentiometer so I did very lengthy calculations to find four notes that were spaced out. I then had to find common resistor/capacitor values to combine to get close to my desired frequencies. This was a very difficult process. In hindsight this was a waste of time and money. Once I got two notes implemented on the bread board I realized that it was difficult to get the intonation of the recorder and resistor/capacitor network to match up precisely. William Dubel then suggested that I use precision potentiometers. This improved my design immediately. I was then able to tune four different notes while playing the recorder. I had to play around with the capacitor values for the filters on the chips. I realized that I had to use much lower values than what was recommended on the data sheet. The opamp also took some work to get a satisfactory gain. William Dubel also gave me a circuit schematic from which to build my foundation. I changed the feedback resistor several times to get the appropriate gain. I just exchanged several different values into the breadboard.

At first I wanted to use the recorder so I prepared the entire tone detection circuit for it. I used a relatively large gain and I researched and bought a mic with higher sensitivity. I finally decided that even these compensations were not enough and the

recorder was just not stable enough. I decided to use my sax and so I had to retune the circuit. It was then more responsive and predictable as well as being far more aurally pleasing.

My bump switches were fairly straightforward. It took me a little bit to figure out the exact orientation to implement a pull up resistor configuration. Once I had this settled I just connected my signal line to an open port on the microcontroller and then polled the pin and waited for the signal to go low.

My IR sensor took a considerable amount of calibration. The devices that I used were not very accurate so I decided to use them for only extreme values (i.e. reflection or no reflection). The ADC conversion result register is 10 bits and so I left shifted the result and read only the high byte. I used my lcd to display the values of this register. I had to split the byte up and send it one nibble at a time. I took some readings and found the threshold between some reflection and no reflection. I realized that I could generally get the same value over and over for the black tape and suspended in air. To account for error I calibrated the ADC value at the beginning of the code and then assigned a range of acceptable values.

When testing my motors I had to try several different values for my PWMs till I got comfortable speeds. I did not want to chase the robot or have it run out of my range so that it would not be able to detect the brake note.

Conclusion

My robot worked. I hope that future students will expand upon my tone detection circuit and use it for more complex applications. I never got my PCB to work but my breadboard worked fine. I would challenge anyone to see if they can bridge the gap. If any one has any questions in the future I can always be reached at my email address: mattking@ufl.edu.

This was probably the hardest and most stressful class that I have ever taken, but it was also the most rewarding. I am impressed with how much real world experience I gained from the class. I would encourage future students to take the class but only if they have nothing else to do. If you do the robot full time you can design some pretty useful and amazing things. Make sure that you plan ahead and always expect Murphy to pop at the least opportune times.

Documentation

The brains of William Dubel and Steven Pickles.

The Data Sheet for the JRC567 chip located on www.digikey.com.

List of frequencies of musical pitches:
http://www.people.virginia.edu/~pdr4h/pitch-freq.html

Recorder fingering chart.

LCD interfacing notes from EEL4744.

Atmega 128 Data Sheet.

Talrik-iv_excerpt.pdf located on www.mekatronix.com

Instruction manual for the Lynxmotion motor driver.

Sample code from www.bdmicro.com

Downloading, Installing and Configuring WinAvr by: Colin O'Flynn. Located on www.avrfreaks.com

Protel Tutorial by: Jason Plew. Located at http://mil.ulf.edu/5666

Appendices

Code for Hamelin

```
//This is the code for Freestyle mode.
//Hamelin.c
//Matt King

#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>


int note_Fsharp, note_A, note_Csharp, note_E, note_B, ircalib;

void init_interrupts (void);
void init_adc(void);
void delay (void);


main()
{
  int temp;
  init_interrupts();
 // init_adc();
  note_Fsharp = 0x00;
  note_A =0x00;
  note_Csharp = 0x00;
  note_E = 0x00;
  note_B = 0x00;
  TCCR0 = 0x64;
  TCCR2 = 0x63;
  DDRA=0xFF;
  DDRB=0xFF;
  DDRC=0x00;
  DDRD = 0xF0;
  DDRE=0xCF;
  /*
  while(1)
  {
                    DDRB = 0xff;
                    PORTB = 0x06;
  }*/
  while(1){
  temp=PINC;
  temp = temp & 0x07;
  if (temp!=0x07)
                     {
                      PORTA = 0x0F;
                      OCR0 = 0xFF;
                      OCR2 = 0xFF;
                      }
    }
```

```
 }


SIGNAL(SIG_INTERRUPT0)
{
  EIMSK  = 0x00;
  note_Fsharp = 0x01;
  TCCR1B = 0x05;
  TCCR0 = 0x64;
  TCCR2 = 0x63;
  TCNT1H = 0x00;
  TCNT1L = 0x00;
  OCR1AH = 0x01;
  OCR1AL = 0xFF;
  TIMSK  = 0x10;
}


SIGNAL(SIG_INTERRUPT1)
{
  EIMSK  = 0x00;
  note_A = 0x01;
  TCCR1B = 0x05;
  TCCR0 = 0x64;
  TCCR2 = 0x63;
  TCNT1H = 0x00;
  TCNT1L = 0x00;
  OCR1AH = 0x01;
  OCR1AL = 0xFF;
  TIMSK  = 0x10;
}

SIGNAL(SIG_INTERRUPT2)
{
  EIMSK  = 0x00;
  note_Csharp = 0x01;
  TCCR1B = 0x05;
  TCCR0 = 0x64;
  TCCR2 = 0x63;
  TCNT1H = 0x00;
  TCNT1L = 0x00;
  OCR1AH = 0x01;
  OCR1AL = 0xFF;
  TIMSK  = 0x10;
}

SIGNAL(SIG_INTERRUPT3)
{
  EIMSK  = 0x00;
  note_E = 0x01;
  TCCR1B = 0x05;
  TCCR0 = 0x64;
  TCCR2 = 0x63;
  TCNT1H = 0x00;
```

```
  TCNT1L = 0x00;
  OCR1AH = 0x01;
  OCR1AL = 0xFF;
  TIMSK  = 0x10;
}

SIGNAL(SIG_INTERRUPT5)
{
  EIMSK  = 0x00;
  note_B = 0x01;
  TCCR1B = 0x05;
  TCCR0 = 0x64;
  TCCR2 = 0x63;
  TCNT1H = 0x00;
  TCNT1L = 0x00;
  OCR1AH = 0x01;
  OCR1AL = 0xFF;
  TIMSK  = 0x10;
}

/*SIGNAL(SIG_ADC)
{
   int temp;
          temp = ADCH;
          if (temp >= ircalib)
          {
                    SREG = 0x00;
                    PORTA = 0x0F;
      OCR0 = 0xFF;
           OCR2 = 0xFF;
          }
}*/

SIGNAL(SIG_OUTPUT_COMPARE1A)
{
          TIMSK = 0x00;

   if (note_Fsharp == 0x01)
                    {
                    PORTA = 0x05;
                    OCR0 = 0x80;
              OCR2 = 0x80;
                    note_Fsharp = 0x00;
                    }
          else if (note_A == 0x01)
                    {
                    PORTA = 0x0B;
                    OCR0 = 0x40;
              OCR2 = 0xFF;
                    note_A = 0x00;
                    }
          else if (note_Csharp == 0x01)
                    {
              PORTA = 0x0A;
                    OCR0 = 0x80;
              OCR2 = 0x80;
```

```
                        note_Csharp = 0x00;
                        }
    else if(note_E == 0x01)
                        {
                        PORTA = 0x0E;
                        OCR0 = 0xFF;
                        OCR2 = 0x40;
                        note_E = 0x00;
                        }
            else if(note_B == 0x01)
                        {
                        PORTA = 0x0F;
                        OCR0 = 0xFF;
                        OCR2 = 0xFF;
                        note_B = 0x00;
                        }
            EIMSK = 0x2F;
}
void init_interrupts(void)
{
            SREG = 0x80;
            EIMSK = 0x2F;
            EICRA = 0x00;
            EICRB = 0x00;
}

/*void init_adc(void)
{
            int temp;
            DDRF  = 0x00;
    PORTF = 0x00;
            ADCSRA = 0xEF;
            ADMUX = 0x62;
            delay();
            delay();
            delay();
            temp = ADCH;
            temp = temp - 0x06;
            ircalib = temp;
}*/

void delay (void)
{
    volatile uint16_t time1;
    for(time1 = 0; time1 < 2000; time1++);
}
```