Intelligent Machines Design Laboratory

EEL5666
Summer 2002

Final Report
Thursday, August 8, 2002

Cyrus Harrison

Table of Contents

1. Abstract

This document outlines my progress and design for creating Ranzor, a robot which can recognize the faces of rolled dice. My goals for this project are to learn about the process of creating an intelligent mobile robot and to focus my image analysis interests into a concrete project. I also want to familiarize myself with Atmel's AVR line of microcontrollers and gain more experience with embedded programming. The end result will be an autonomous robot which can extract true random numbers using the dice.

2. Introduction

Frustrated with the restriction of only pseudo random numbers in programming, I decided to create an agent which can extract random numbers. Ranzor will roam the treacherous terrain of standard table hunting dice while trying not to run into things and avoiding the certain death which lurks over the edge. Upon finding a die it will recognize the value of its top face and speak its value then knock it off of the edge of the table. From there it is up in the air, maybe it will become a friendly craps dealer or perhaps be enslaved by a statistician to check the probably distribution of a dice roll ad infinitum.

In creating this system I hope to gain experience with simple image recognition, an introduction to autonomous robotics, strengthen my embedded programming skills and have some fun. It is sure to be an interesting experience.

3. Executive Summary

This was quite a learning experience for me. I am very glad to have had the opportunity to work on a project of this magnitude. My original goals for this project were to create a robot which could talk, collect dice, recognize their face value, and reroll them. I accomplished everything except the rerolling, which I accomplished partially. The rerolling was kind of fun cap to the robots action, but the collecting and image processing of the dice was my main goal. I am proud to say that I have completed this recognition with accuracy greater than 75%. If I had a camera that was less susceptible to interference and a little bit better mechanical design for my camera lift I believe I could get 90%+.

If the die is placed correctly in the staging area, the camera lowers correctly every time, and there is no signal interference it will always (95%+) recognize the face value. I am impressed with my success in this area since I have never done any image processing before; it took a little bit of research and a lot of tweaking. I have been interested in this area for awhile and this project gave me a chance to explore and learn about it.

The reason why I did not fully implement the rerolling was because I purchased two solenoids and could not use either. One was too small and just moved the dice a little bit, and the other was two big and pulled to much current off of the batteries. Instead of ordering a new one and have to worry about hooking up and tweaking it during the last part of the class, I felt my time would be better spent tweaking the other aspects of the design, especially the voice playback. I decided to replace the kick rolling with a knock off of the table behavior. After a die is found Ranzor now looks for the edge of the table and shakes the die off of the edge. This behavior rolls the die but after it does so it is obviously not accessible to Ranzor, which is why I say I did not fully implements the reroll behavior. This behavior also shows another use of my table edge sensors and I think it is a bit more interesting than simple kicking.

I suffered a few other set backs during the process but overcame these. I burned a few boards due to loose solder joints and placed a few chips in backwards, etc. I was disappointed with the SPO256 Chip. This chip would have been the perfect solution for my robot but I could not get it to work. I eventually had to give up on it and switch to the ISD2500 chip which was very easy to use and brand new. In the future I am going to give the SPO chip another chance, but it is quickly becoming scarce on the internet.

4. Integrated System

Ranzor, my robot, collects dice from a table and recognize values on their top face. It then finds the edge of the table and pushes the die off. The robot might not be too useful, but is a unique and fun approach to robotics.

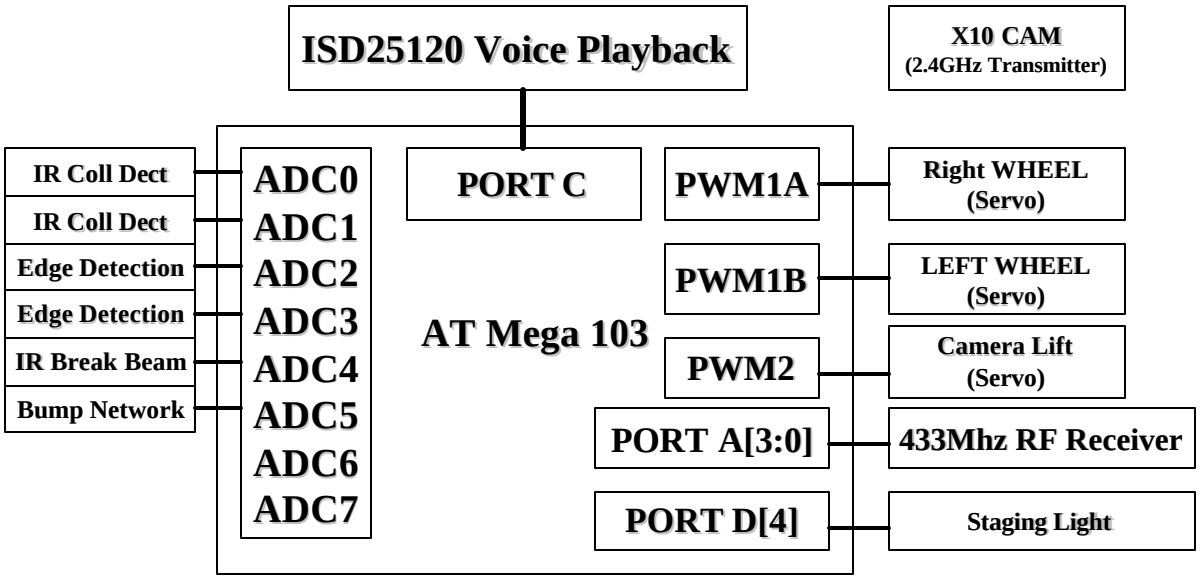
The Dice Recognition system has to major components:

- Mobile Unit: Ranzor
- PC Image Analysis

4.1 Mobile Unit

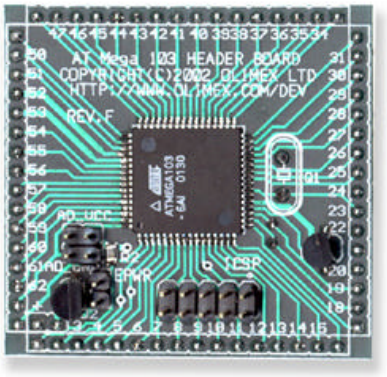
Ranzor, the mobile unit, has the task of navigating a table and gathering a die into the video staging area for recognition. It roams the table randomly avoiding the edge using Sharp IR Sensors. Once the Die enters the staging area, the break beam will sense its presence. Then it will lower the Camera Enclosure and turn on the staging light. At this point the PC Image Recognition program will notice the change in video signal and will begin the recognition. After the recognition is complete it will send the face value back to Ranzor using a LINX 433 MHz Transmitter connected to the Parallel Port. A receiver on Ranzor will receive the value and interpret it. To accomplish these behaviors a suite of systems is needed. Figure 4.1.A shows an overview of Ranzor's Systems.

Figure 4.1.A Ranzor's System



The brain of Ranzor is an Atmel ATMega 103 Microprocessor, it provides all necessary to integrated and control the various systems needed. I purchased a board for this chip that contains a voltage regulator, 6Mhz crystal, ISP Header and headers for every pin of the Mega103. It is shown in Figure 3.1.B.

Figure 4.1.B
AVR-H103B from Olimex
www.olimex.com/dev/



It is a small board at 1.85”x1.85”. To fully use the board, I created two extension boards:

The first adds a LM74805 Voltage Regulator, which can handle up to an amp. This board also pulls out the Analog Signals to three pin ports (Signal, VCC, GND), powered by this voltage regulator. It also pulls out the UART to a 4 pin interface, so it can connect to a MAX232 Chip for debugging on the PC.

The second pulls out the PWM outputs to three pin ports (Signal VCC, GND) to control servos. It also contains the simple transistor switch to turn on and off the staging area light and has a few two pin power ports for connecting more components.

All and all I really like this board, It has plenty of IO pins, I originally bought it because you can expose the address and data bus on it for a memory expansion, however I had so many IO Pins this was not necessary. It has four PWM Outputs using two 8-bit timers, and splitting up its one 16-bit timer into two more. The eight analog ports were adequate as well.

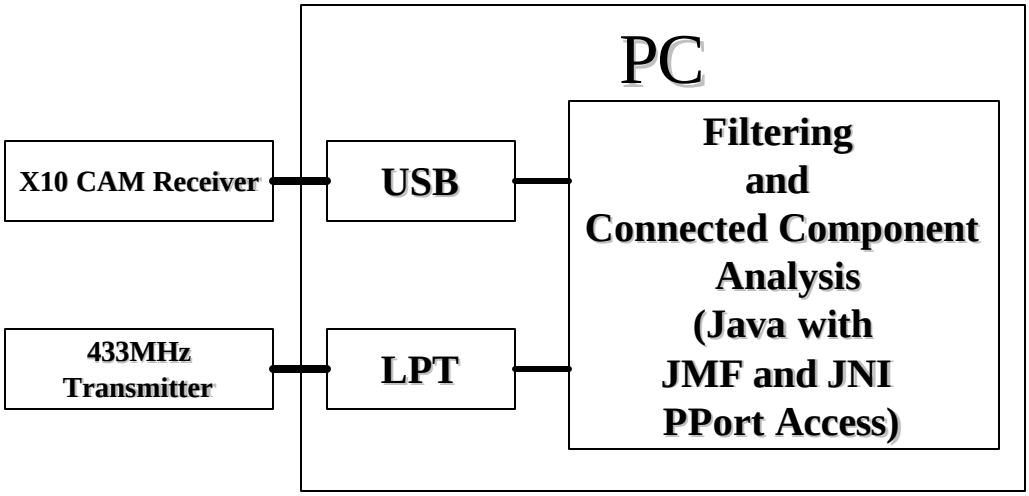
I used AVR GCC, AVR Studio, and Pony Prog 2000 for my development. I liked using AVR GCC it was very easy to get started using examples. The Simulator in AVR Studio (4) helped a lot also for testing out interrupts and making sure timing was correct for my PWMs. I would recommend using AVR Micros for IMDL and other Microcontroller projects, they are very powerfully and well documented.

4.2 PC Image Analysis

The RGB Video Signal is processed with a PC using the Java Media Framework. By filtering out the light spots of the image we are left only with the dots on the face of the die and the dark background surrounding the die in the staging are. From here connected component analysis is used to extract how many visible entities are in the picture, and this count is the die value. This process is covered in detail in section eight.

Figure 4.2.A shows an overview of the Image Analysis System.

Figure 4.2.A Image Analysis System

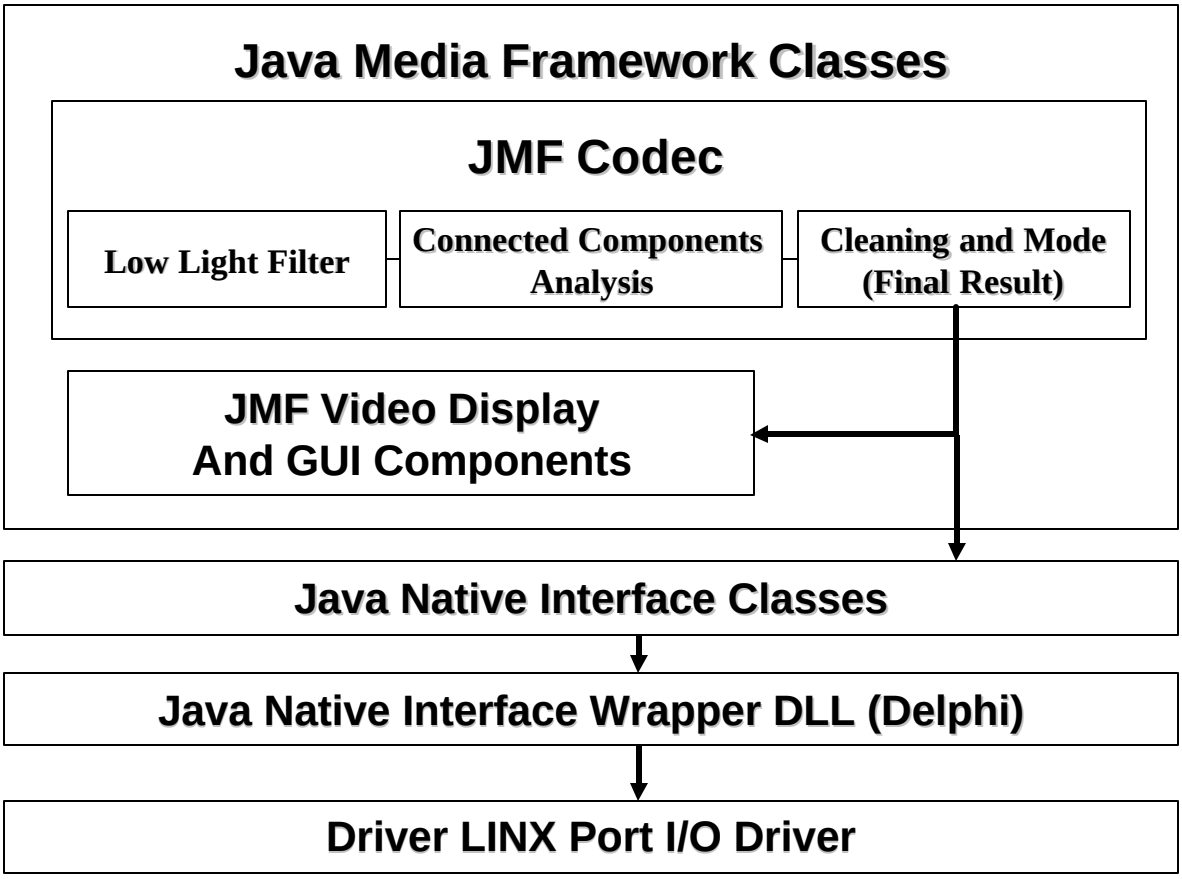


The Java Media Framework gives access to the RBG Data for each frame. To send back the value, I was originally going to use the PC's Serial Port, however I found that using the Parallel Port was much less of a hassle. In order to access the Parallel Port in Java under Win XP, I used the DriverLINX Port I/O Driver for parallel port access. I used this DLL and wrapped it in another DLL so Java could Access it. This was done using Delphi and Java's Native Interface (JNI). This code is included in the appendix.

To use RF in a parallel manner I use a Holtek Serial Encoder/Decoder Pair. A 4 bit Serial Encoder (Holtek HT-12E) is used to take the Parallel Data from the PC and transmit it via the Linx RF Module and the reverse is done on Ranzor with an RF Receiver and a Holtek HT-12D 4-bit Serial Decoder. I had some problems with these RF chips, however I think I damaged the receiver early on and this is likely the cause of these problems. They were pretty easy to hook up and I would recommend using them with a decent antenna for RF projects.

The software used to filter the image is covered in the sensors section in great detail. Figure 4.2.B shows an overview of the Software components and the flow of data as described on the previous page.

Figure 4.2.B Software Components Interaction



4.3 Subsystems

The following is a listing and short description of systems on Ranzor:

Analog Sensors

- Two Sharp GP2Y0A02YK IR Sensors will serve as eyes and allow for collision avoidance and in my final configuration as Table Edge Detection Sensors.
- Bump Network with bump switches and resistors provides a last resort collision resolution and miscellaneous calibration.
- An IR Break beam Sensor detects when a die has been acquired.

Movement

- Two Hacked PWM Servos provide the robot with its wheels
- A small servo lifts and lowers the Camera Enclosure for face recognition

Communication

- Face result is displayed on the Mobile Unit in addition to the PC, using Voice via an ISD2500 Voice Playback Module
- Die result is sent back to Mobile unit using 433 Linx Transmitter / Receiver Pair.
- The PC Image Analysis Program knows when to start the recognition by a light in the Die Staging Area.

Video

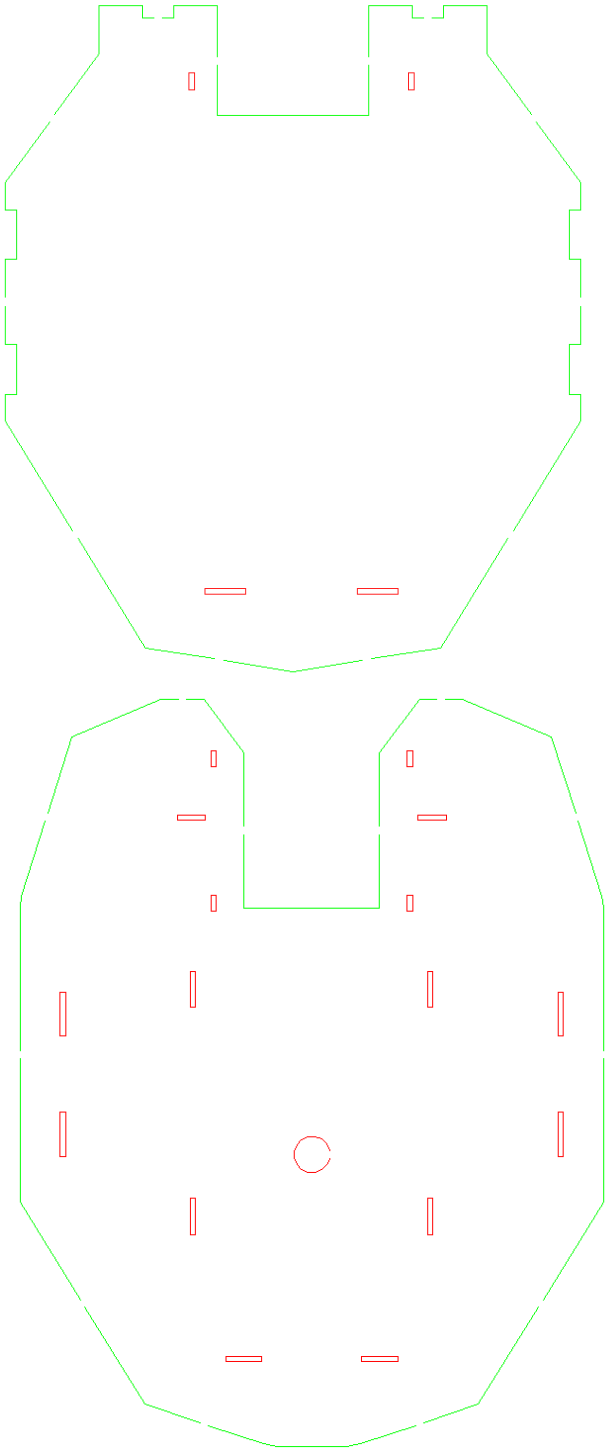
- X10 Camera provides images of dice faces, PC will receive the signal using X10 Receiver and VA11A Video to USB converter
- Die recognition will be done using a custom codec for Java Media Framework

All of these systems are discussed in detail in the Actuation and Sensors Sections of this paper.

5. Mobile Platform

I designed my mobile platform from scratch using AutoCAD. My restrictions on the design were very simple, make it as small as possible and make sure it will accomplish its purpose. Ranzor did not require a marvelous design; the main design difficulty was creating a working die staging area. Figure 5.1 shows the Autocad Drawings for the bottom and top platform.

Figure 5.1 Ranzor Auto CAD

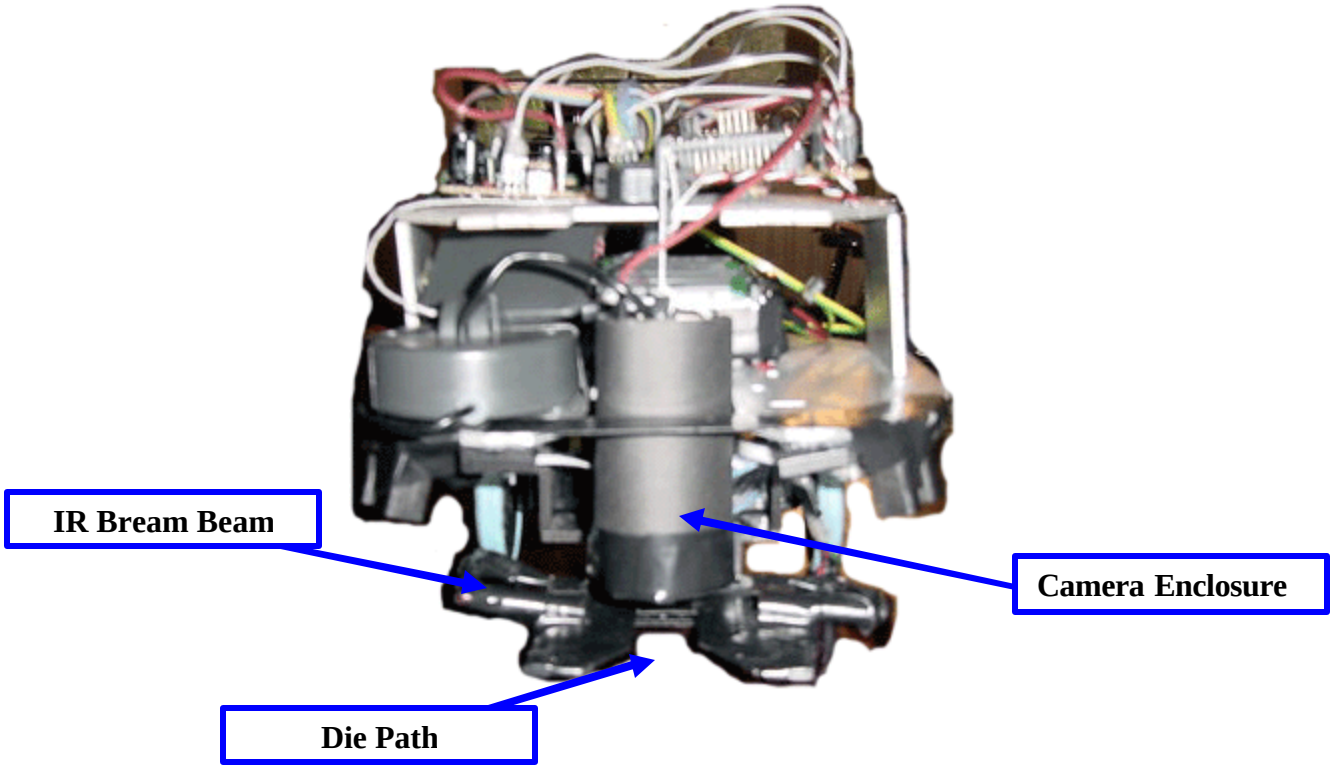


Top Level Platform

Lower Level Platform

Below the lower platform is a box which houses the wheel servos.Figure 5.2 shows a front view of Ranzor. I based my design on a die staging area that would rise to the bottom of the Camera Enclosure. This pulley lift turned out to not be exactly what I needed, so I changed it so the Camera was lowered to the die. This did not change the design of the platform at all, just how the Camera and Staging Area were mounted.

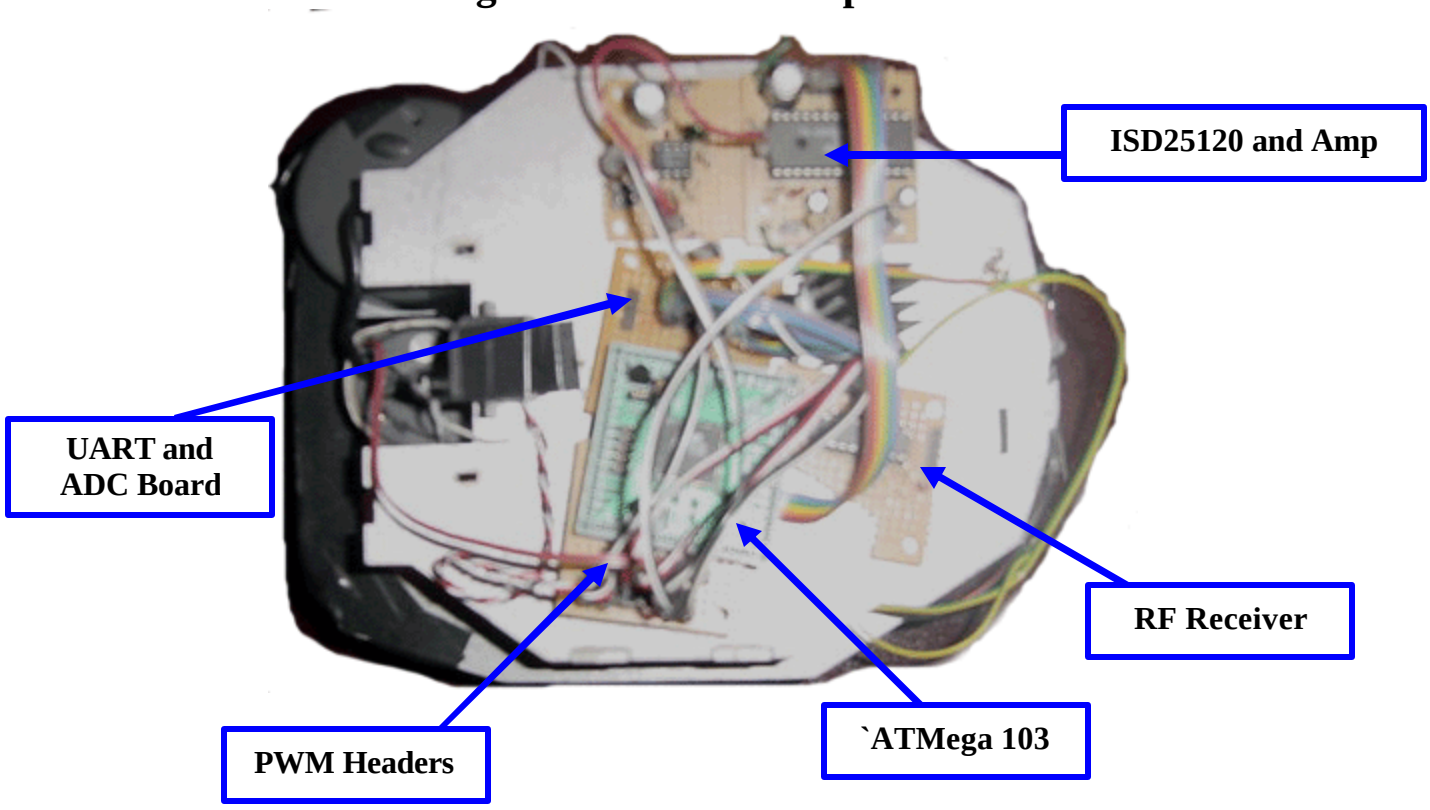
Figure 5.2 Ranzor Front View



For locomotion I wanted to have a simple control so I chose a run of the mill two wheel center steered design. I am happy with my platform design. Before this project I have never used AutoCad so it was an interesting experience.

I chose a two level design because this gives me versatility for mounting a bunch of stuff. The top level is for electronics and the bottom level is for the camera, speaker and battery packs. The X10 requires a separate battery back from all other electronics. Figure 5.3 (next page) Shows a Top View of Ranzor and its mounted systems.

Figure 5.3 Ranzor: Top View



6. Actuation

The simple table top environment limits the necessary complexity of my platform's movement. As stated previously the robots movement on the table will be created via two centered hacked servos. Here are the details of the components I chose for Ranzor's Actuation

Locomotion

Servos

Locomotion is created via two prehacked servos. I choose the prehacked servos for ease of use. Plug them in and experiment with PWM timing and you are ready to roll. They are not very high quality compared to other servos I have used, but they get the job done. I have noticed that they go a little faster forward than backward. Because they have to be mounted in reverse on opposite sides, this causes the robot to turn a bit when attempting to move forward. It has not caused any problems; actually this helps my robot get out of possible traps when both IRs are switching on reporting obstacle proximity.

Wheels

I purchased these wheels because they could plug directly into a servo, and they were the correct size for my platform. They are nice wheels made specifically for hobby robotics and I am very happy with them. Using the rubber bands they slip very little on normal surfaces.

Camera Lift

Servo

For lifting my camera from up and down the staging area I needed something with a small profile. I original planned to lift the dice up to the camera with a platform, but this was unnecessary. I selected something much simpler. Because I wanted to have an aesthetically pleasing solution I chose the smallest servo I could get, and hoped the torque would be adequate. It turned out to work fine. This tiny servo is around the size of a nickel.

Die Rolling

Solenoid

To accomplish die rolling, I tried using two solenoids. One of them was much too small and could not provide enough power to roll the die. The other was much too big and pulled too much current from my batter pack causing a reset. Since I already have two battery packs on the platform, I decided against using a separate battery pack roll a die. I was not able acoomplish die rolling because of this, but a solenoid in the middle of the two I bought would probably do the trick.

Voice

My original plan was to use a SPO256 Speech Generator Chip. This chip was found in early video game systems, and sold at Radio Shack in the 1980's. The chip is nice because you can create any word needed combining a series of 59 phonetical sounds. The only problem with this chip is it is not sold any more. I purchased one from a private seller. I created a board from diagrams online for this chip, and tested it. I was only able to obtain some gibberish. The timing sounded correct for the words but I had no real luck.

Since I have no real way of knowing if the chip ever worked I decided to use another chip for speech. The chip I choose is an ISD2500 ChipCoder from Winbond. Using the documentation it was pretty easy to wire up. I also used a LM386 Op Amp to amplify the Audio. The only regret I have with this chip is that I chose the 120 second Version. All of the ISD2500 Series Chips have the same amount of memory for playback; the only difference is the sample rate, so the 120 Second version sounds kind of crappy. It turned out I did not need that much play back and I wish I had better voice quality.
Figures 6.1 and 6.2 show the Schematics for my ISD2500 Setup

Figure 6.1 ISD25120 Circuit

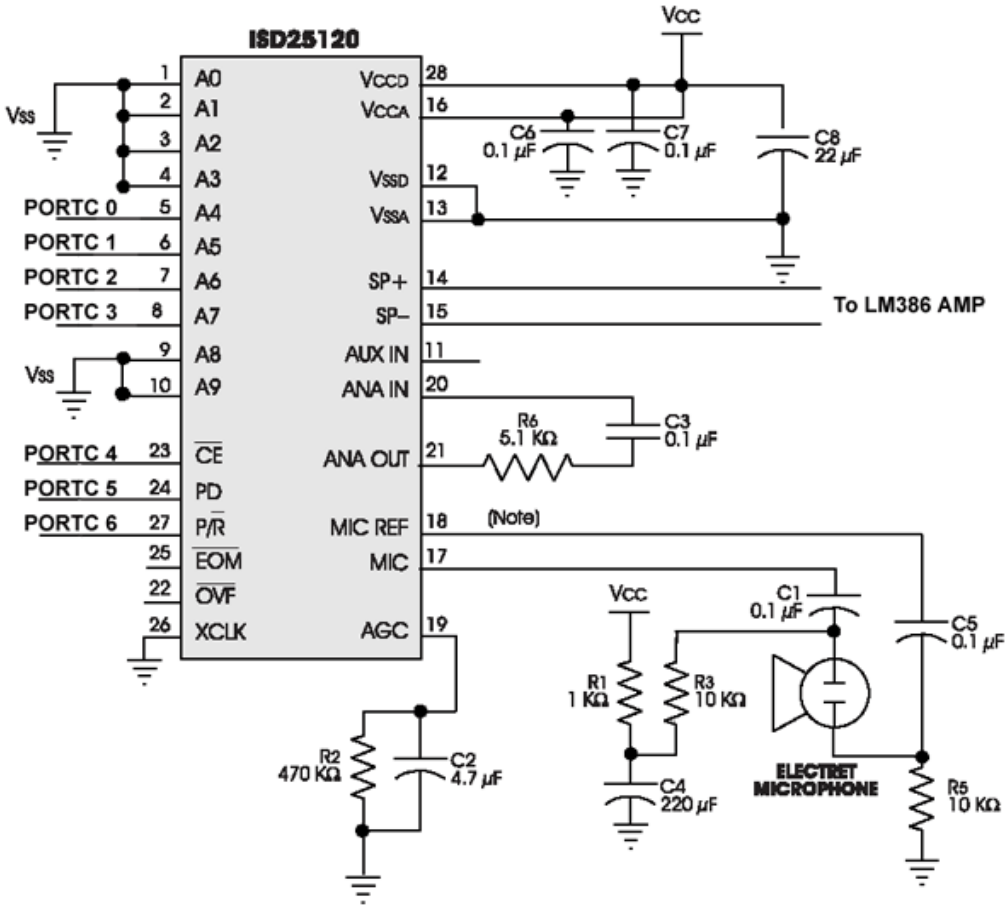
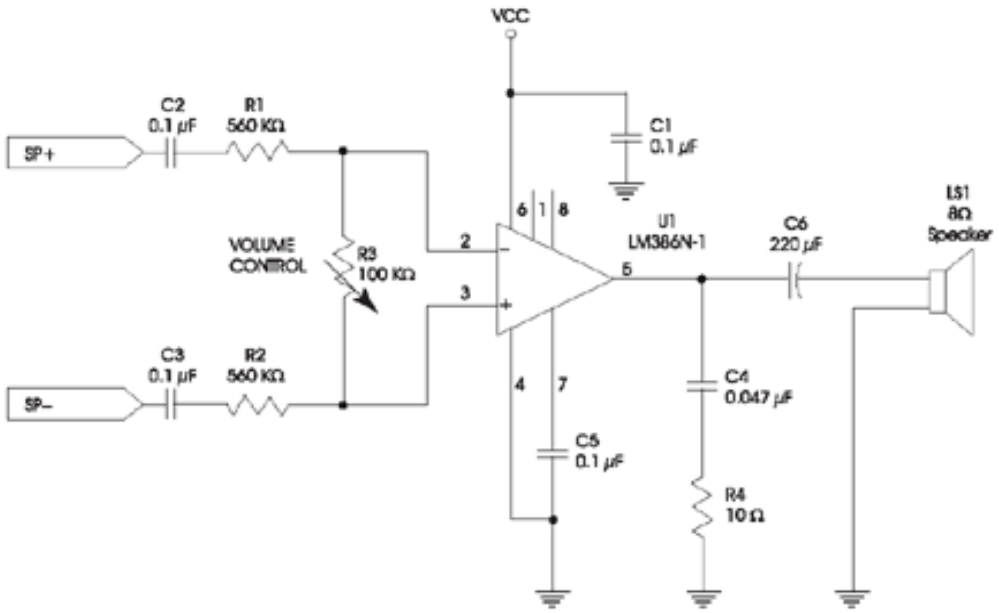


Figure 6.2 LM386 Op Amp Circuit



I had to put some work in to make the voice noticeable. I wanted to make sure that it could be heard in a semi noisy environment. First I tried just a simple LM386 Amplifier. This gave me some more volume but it did not give me the level I wanted. To solve the problem I used a Speaker Amplifier a friend loaned to me that runs off of two nine volt batteries. The LM386 was not a lost cause because I needed this circuit to preamplifier for the more powerful amplifier.

7. Sensors

The Sensors on Ranzor help avoid the table edge, find table edge, detected a die, and recognize the die face value. Here are the details of the sensors components I chose:

Image Recognition with X 10 Camera:

Camera Mounting:

The X10 Camera is mounted on the front of the mobile platform using a thick paper tube to isolate any environmental light. The light source for dice staging is a white led. Using a 100 Ohm resistor the light is very bright and illuminates the dice well enough for the image analysis algorithm to work. Fooling with many distances of the X10 lens, I found that the best image was produced when placed around 2 inches.
At this distance the camera focuses on the die well and external light from the bottom of the staging area is causes minimal interference.

When the IR Bream Beak is triggered (die has been found), a small servo lowers the camera tube down so its bottom touches the dice platform. From here a clear image of a die can be obtained.

To power the X10 with batteries an X10 battery pack is needed. The battery pack contains a small circuit which amplifies four AA batteries to the voltage necessary for the camera. Since the Battery back is very bulky, I removed this unit from the plastic molding and attached it to a smaller battery pack. I also shortened the X10's Power Cord because it was a nuisance.

Image Analysis:

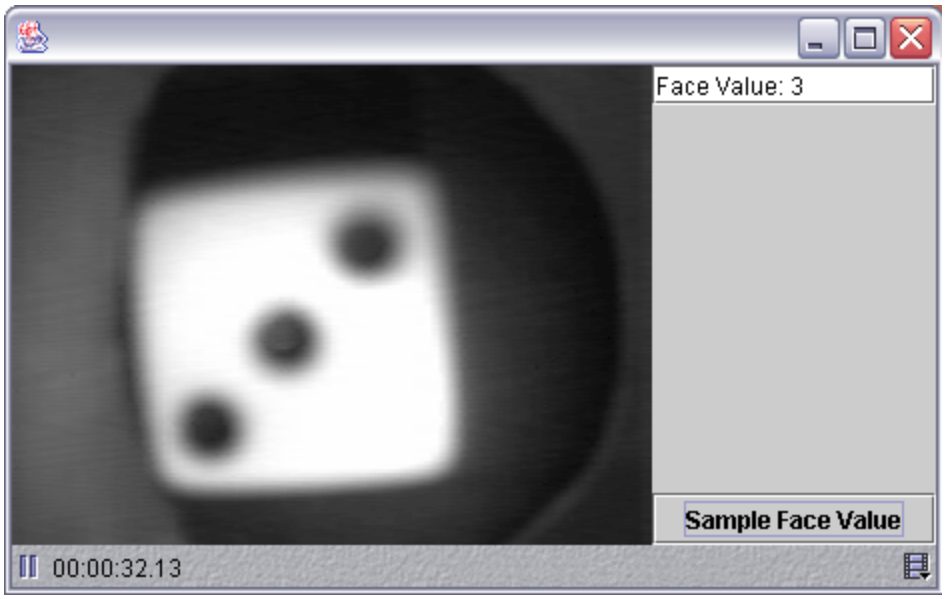
The algorithm looks for similar regions in the image and counts how many of these separate components exist. The camera area is staged so that any part of the image that is not part of a die should be dark. This means counting the components will result in one more than dots are on the die observed.

The image analysis is done in three steps using connected components labeling algorithm as a base.

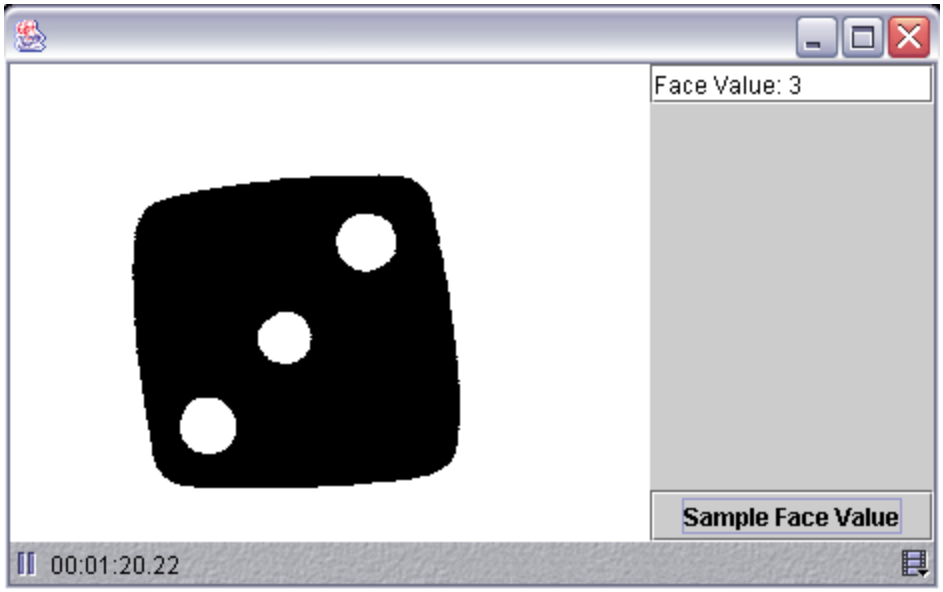
Low Light Filter:

Examines each pixel of the image and filters out anything that is not "dark". This gives a new image which has the same resolution. Each new pixel is binary to represent if the pixel was dark or light based on the filter threshold.

Before LLFilter:



After LLFilter:



Labeling of Components:

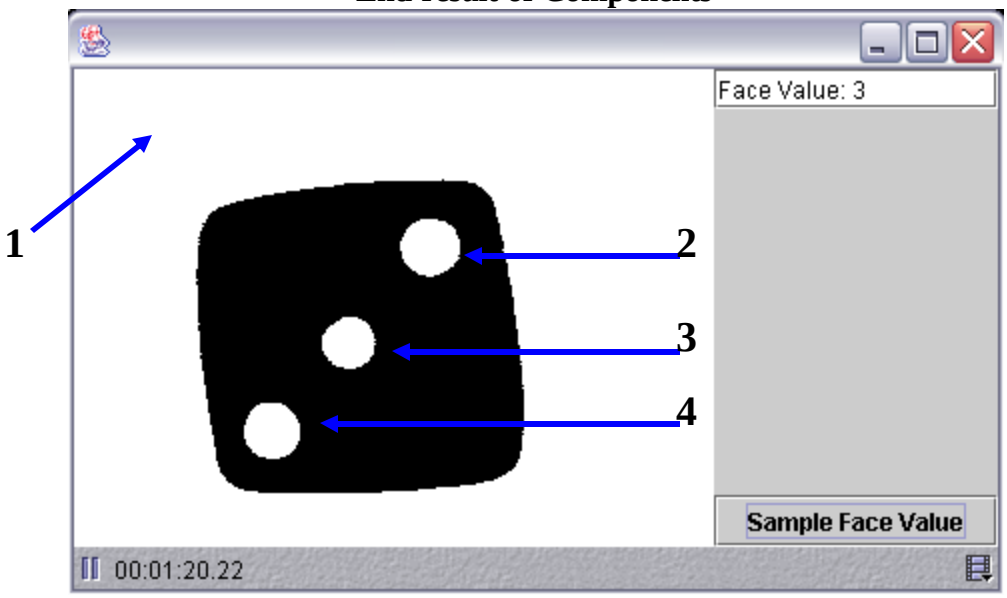
Examines each pixel to see if its adjacent pixels are of the same nature. With a binary image if a dark pixel has neighbors that are dark, then they will be labeled as part of the same component. If a dark pixel is newly discovered it is considered a new component. This scan is done left to right, top to bottom on the image to discover how many distinct components.

Cleaning of Components:

Cleans the discovered components to avoid errors from a messy signal. This process eliminates errors in the process because of a jagged image, and throws out any

components that are to small to be considered dots. Several samples are taken and the mode of these sample is used for the end result.

End result of Components



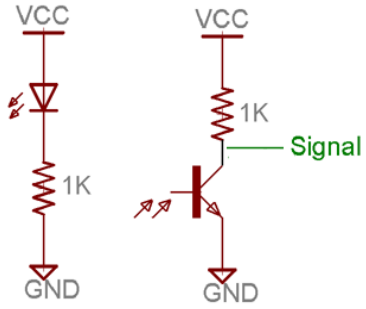
One is subtracted from the found components.

Thanks to http://www.dai.ed.ac.uk/HIPR2/label.htm for showing me the use and theory of the Connected Components Algorithm

Sharp GP2Y0A02YK Distance Measuring Sensor:

The Sharp GP2Y0A02YK despite the ridiculous name is a very easy to use IR Proximity Detector. It has a simple three connection interface (Signal, GND, VCC) which can be plugged directly into the ADC header on my extension board. The Sensor Datasheet says it can accurately measure between 20cm and 150cm. This range is much more than needed. The sensor voltage rises the more its emitted IR is reflected back. Through experimentation I found that when something is close enough to cause concern, the value is about 2 Volts (0x190 when read by 10bit ADC). When the sensor is very very close (collision will be shortly unavoidable) the sensor saturates and the value begins to drop. This choice of the 0x190 gives a handle on the proximity of an approaching object and also will snag it if the sensor saturates.

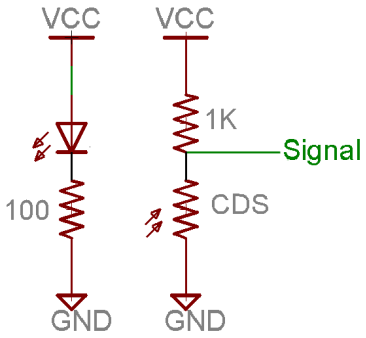
IR Break Beam:



The IR Break Beam is used to detect when a die has entered the image recognition staging area. It is a simple IR Emitter LED and IR Detector Transistor pair. I used heat shrink tubing to collimate the light from the emitter and aim it directly at the detector. When the beam is broken the voltage raises from 3 volts to almost 5 volts.

As long as there is not a very bright light directed on to the detector (easy enough to avoid) this characteristic will prevail. The Break Beam detection takes precedence over IR collision detection, and causes the platform to stop and lower the Camera. After it receives the value from the computer via RF receiver, Ranzor will lift the Camera and resume normal operation.

Edge of Table Sensors:

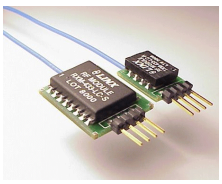


To create the Edge of Table Sensors I used Cadmium Sulfide Cells and bright white LEDs. The White LED is connected with a 100 Ohm Resistor exactly like the one used to light the die staging area. I used heat shrink tubing to collimate the light into a small bright circle, and placed the CdS Cell in series with a 1K resistor next to it. The sensor is very similar to Vinh Trinh's from Spring 2002. The concept for edge detection using this sensor is pretty simple. When the sensor is on a surface, the CdS Cell will see the reflection from the white led and change resistance accordingly. When the sensor is placed over nothing (over the edge of the table) the light will not affect the CdS Cell. By examining the voltages produced in these situations the robot can prevent itself from falling off of the table. I have tested it on different colors of surfaces and it should work for any surface light or dark.

To ensure that external light does not reach the CdS Cells, I encapsulated the pair in a plastic film canister. I cut and added to the film canister so when mounted on the front of the platform it would extend to the table, and shield the sensor from external light.

I created these sensors and they worked well, but I found that just using the Sharp GP2Y0A02YK worked just as well if not better, so I decided on just using these for the final design. If you are doing table edge sensing in the future, I would recommend you consider using the Sharp Sensors because they are very easy to use. Just mount them downwards on your platform and simply look for a drop in the analog value to indicate the absence of the table. Ranzor has not fallen off of the table yet.

RF Modules:



I purchased a pair TX/RX of Linx RF Modules from Rentron.com. They were very easy to use. I chose to Holtek 4-bit Encoder / Decoder Pair to simplify the transmission. I would recommend using these if you are going you using RF Communication but make sure you have good antennas.

8. Behaviors

Ranzor's behaviors support its objective of finding die, and rolling them off the edge of the table. Figure 7.1 shows an overview of Ranzor's behaviors:

Figure 8.1 Behavioral Flow Chart

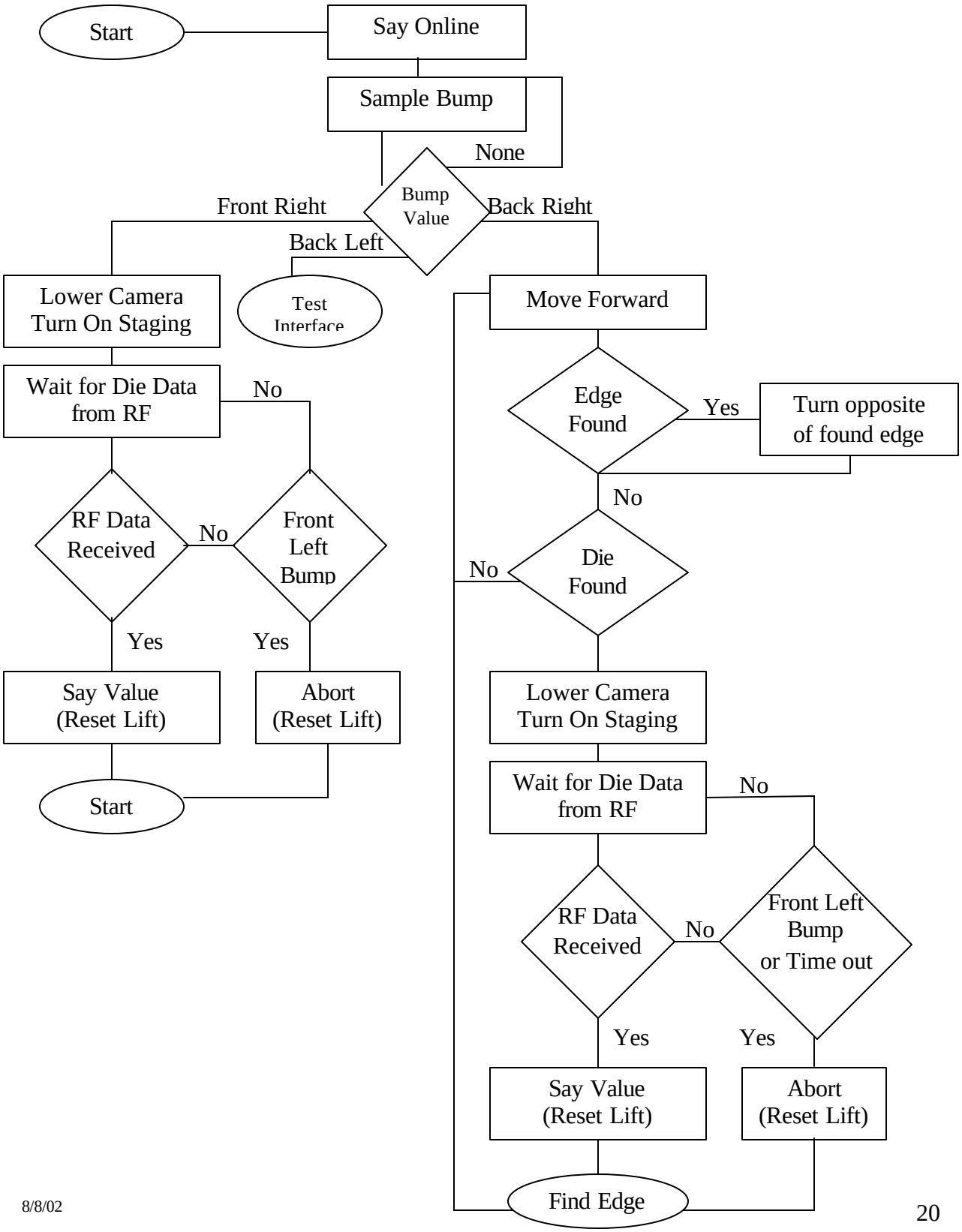
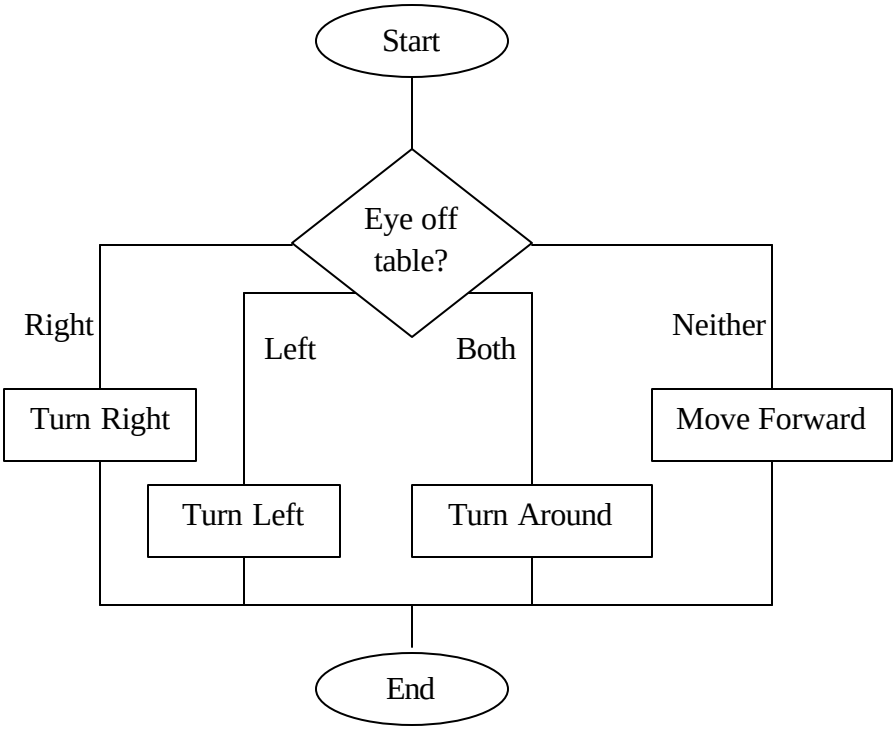


Figure 8.2 shows the find edge sub behavior.

Figure 8.2 Find Edge Behavioral Flow Chart



These flowcharts represent Ranzor's final behaviors; I experimented with many different behaviors during the semester including simple collision avoidance, following the edge of the table using the Sharp IRs, etc.

The only emergent behavior I noticed is that when doing edge of the table avoidance with the Sharp IR's it can also detect obstacles that are low on the table. This is do to the fact that they saturate and causes a value drop. This was the main reason why I chose to use the IR edge detection instead of the CdS Cell method. Tilting the IR out a little bit also gives more of a warning of the table edge than a CdS Cell Method can.

9. Experimental Layout and Results

The X10 Camera auto adjusts the brightness and contrast of the image based on the amount of light it sees. This makes reproducing results with the X10 a pain. To determine the best setup for the camera I tried several configurations. Table 9.1 shows these configurations with the accuracy of the die recognition for each setup. For each setup I did a series of trials (around 20) and found the reliability. The tests were done in different rooms and lighting situations to try to get handle on how it will work in many situations

Table 9.1 Die Experiments

Setup 1:

Overhead Room Lighting

Reliability: 0%

Results: Worked well in my room, but would have to be adjusted for each room the robot was used in, not a good solution

Setup 2:

Bright White Led, No Isolation

Reliability: 0 %

Results: The auto adjust depends on environmental lighting, so it is almost as bad as just having the environmental light, but is preformed a little better.

Setup 3:

Box camera with camera mounted at opening

Reliability: 20 %

Results: The box works better but still suffers from a large opening where environmental light can enter, will not work well

Setup 4:

Paper Cylinder Isolation with LED Light Bleeding through side

Reliability: 30%

Results: This many sound like a bad idea to start with and it was, but I was trying not to saturate the X10 so much so the auto adjust would not take so much of an effect. It didn't work well in all situations.

Setup 5:

Thick Paper Cylinder with two LEDs Mounted on at bottom to illuminate background and one overhead white led.

Reliability: 40%

Results: Positioning the bottom light was too difficult and the amount of light will be to hard to control on the robot, so it did not work well

Setup 6:

Thick Paper Cylinder with frilly paper filter at bottom to provide constant background and one overhead white led

Reliability: 70%+

Results: Placing the die in the setup produces and easily repeatable

10. Conclusion

This has been a very interesting project for me. I really liked using the Atmel AVR Microcontroller, the development environment and complier made things easy for me and the support on the internet was vast. I will probably use Atmels for any of my future hobby or school projects.

Developing the image analysis system was a nice goal for me because I wanted to have my hand at something in this field and having a real project to focus my efforts on helped. The method I used was rather simple, but I had a few hurdles I had to jump over because of the noise in the X10 signal. The X10 is a nice solution for cheap wireless video but for projects where more precision is necessary I would not recommend it. Also the 2.4 Ghz Range is subject to interface, especially when the camera is powered by the battery pack.

I am happy with exposure to Auto Cad I gained. I have never done any mechanical design before and it was nice to take a swing at designing my platform.

During the process of the course I burned some boards and some fingers but this class was a very valuable learning experience. I fell that accomplished my main goals in this project. The only thing that I did not accomplish fully that I wanted to was the rolling of the die. I was able to roll a found die , but this placed it out of reach from Ranzor in the future. I feel this was a good alternative to solenoid rolling.

11. References and Acknowledgements

I would like to thank Uriel and Tae for their suggestions and help and Dr Arroyo and Dr Schwartz for their guidance as well.

Websites Used:

www.olimex.com/dev/
www.avrfreaks.com
www.atmel.com

Connected Components Analysis Explanation
http://www.dai.ed.ac.uk/HIPR2/label.htm

JMF Docs
http://java.sun.com/products/java-media/jmf/

Java Native Interface and Dephi
http://home.pacifier.com/~mmead/jni/delphi/

DriverLINX Port I/O DLL Use (Python)
www.geocities.com/dinceraydin/python/indexeng.html

ISD2500 (ChipCoder) Docs
www.winbond.com

SPO256 Docs
http://support.tandy.com/support_supplies/17518.htm (Radio Shack)
home.rmci.net/jimboh/spo256/spopcb.htm (SPO Speech Board)

MAX232 Examples
http://www.seattlerobotics.org/encoder/aug97/cable.html

RS232 and Parallel Port Pinouts
http://www.pin-outs.com/datasheet_11.html (RS232 DB9)
http://www.rfcafe.com/references/electrical/parallel_port_pinout.htm (PPort)

All of the places I purchased stuff from (that I can think of):

Websites:
www.olimex.com/dev
www.x10.com
www.parallaxinc.com
www.fmadirect.com
www.junun.org/MarkIII/Store.jsp
www.digikey.com
www.rentron.com
www.solienodcity.com

Stores:
Radio Shack
Electronics plus
Skipper's Electronics

12. Appendix

Appendix contents are on the Ranzor's CD
They are also zipped so that they be placed on the web easily.
Code Contents on the CD

- AVR Code
  - Modules and Headers
- JMF Code
- Delphi JNI Code
  - Delphi DLL Code
  - JNI Code