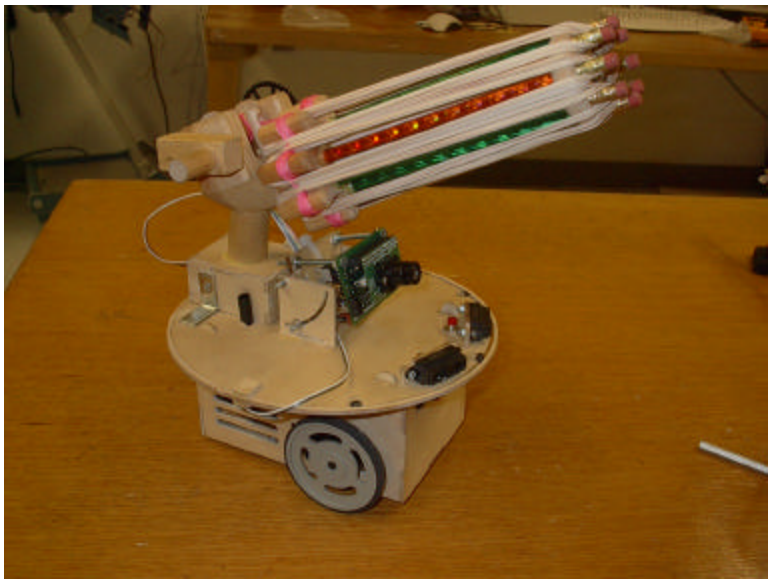
University of Florida

Department of Electrical and Computer Engineering

EEL 5666
Intelligent Machine Design Laboratory



Written Report I

By
Jordan T. Wood
January 25, 2003

Table of Contents

Abstract

S.N.I.P.E.R. is autonomous assassinating agent. S.N.I.P.E.R. can navigating and hide itself until a target comes into view. Once the target is acquired, S.N.I.P.E.R. tracks and eliminates the human target. This is all done without human control.

Executive Summary

S.N.I.P.E.R. is a small round autonomous assassinating agent. This is done by using IR proximity sensors to detect wall and other objects, a CMUcam system to track peoples movements, photo cells to detect a hiding space, and a rotating rubber band gun to assassinate its target.

S.N.I.P.E.R. is controlled with an Atmel mega-AVR 323 microcontroller. All subsystems are connected to this microcontroller and behaviors are executed though it. Its environment dynamically affects the robot's behaviors. S.N.I.P.E.R. moves around until it has determined a hiding spot. Once the hiding spot is located, S.N.I.P.E.R. tracks a person with a specific color and then moves with that person to keep itself align. After a while S.N.I.P.E.R. will then shoot its target.

Introduction

S.N.I.P.E.R. stands for Self Navigating Intelligent Personal Eliminating Robot. S.N.I.P.E.R. will navigate by looking for dark spaces to hide. To do this it has obstacle avoidance system, so it can move around walls and structures. Obstacle avoidance system has two IR sensors to detect a structure or a wall that is in front of the robot. If the wall is there the robot will turn to avoid it. If this fails there is a bump sensor around the robot, once the sensor hits the structure, then the robot will backup and turn. It has a system to detect light and dark to help it find a hiding space. This behavior is achieved by using a photocell that is directed upwards, so when the photocell detects dark, then the robot is in the right space.

S.N.I.P.E.R. will wait in the dark until it sees a human target. The robot uses a CMUcam digital camera, to detect a color that the human target is wearing. Once the robot sees the target, it will go into tracking mode. In tacking mode, the robot finds the center of mass of the color and adjusts itself with the wheels to always point at the target. Once the target passes the center the required number of times, it will shoot the target with a projectile. The projectile shooter is a rotating rubber band gun that S.N.I.P.E.R. can fire up to 48 times. Finally, once the target is assassinated, S.N.I.P.E.R. will relocate and escape, by finding another hiding space.

This report covers all the major components of the robot design in IMDL laboratory, including the mobile platform design, actuation, sensors, and the behavior of the overall system.

Integrated System

The robot is controlled by an Amtel megaAVR 323 microcontroller on a Progressive Resource LLC MegeAVR-development board see fig 1. Each behavior of the robot is controlled by this microcontroller. The behaviors each have subsystem as indicated. The obstacle avoidance system uses two Sharp GP2D12 IR Sensor that will be mounted on the front of the platform. A bump switch sensor is used for physical contacts and servos for movement. The hiding system uses two photocells to detect light and dark. When the robot detects darkness it will know it is hiding. The tracking system uses a CMUcam digital camera to detect a color that a target will be wearing and the movement of this color. The projectile system is a servo connected to a rotating rubber band gun. The robot will fire the rotating rubber band gun by using a servo to move a string, which rotates the gun and fires the rubber bands.

The following is how S.N.I.P.E.R. operates. Once S.N.I.P.E.R. is turned on it can determine if it surroundings are light or dark. If it is light then it goes in to obstacle avoidance, moves around until it finds a dark spot. Once a dark spot is detected, S.N.I.P.E.R. will stop and turn on the camera. The camera will tack a color that a person is wearing. As the person moves S.N.I.P.E.R. detects the distance off center and moves it self back on center. After the targeted person is centered with S.N.I.P.E.R. a number of times, the gun servo will fire for a few seconds to hit the target.

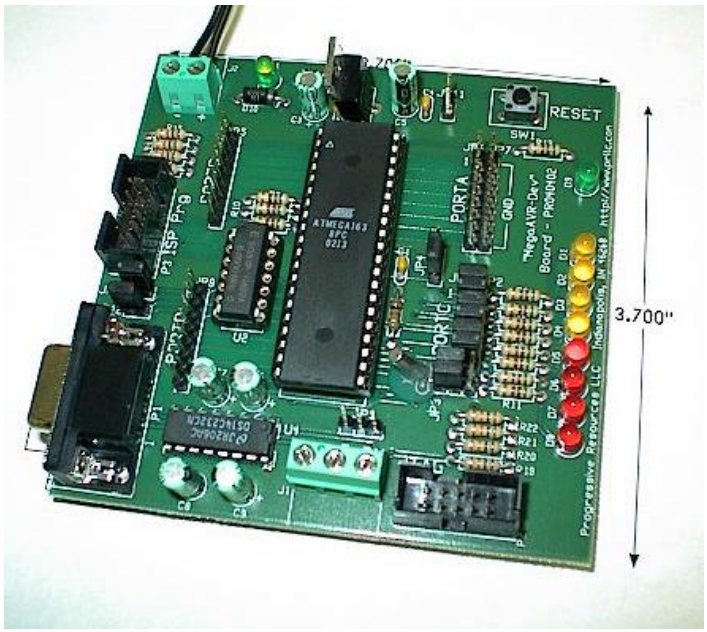


Figure 1. Amtel MegaAVR 323 on a Progressive board.

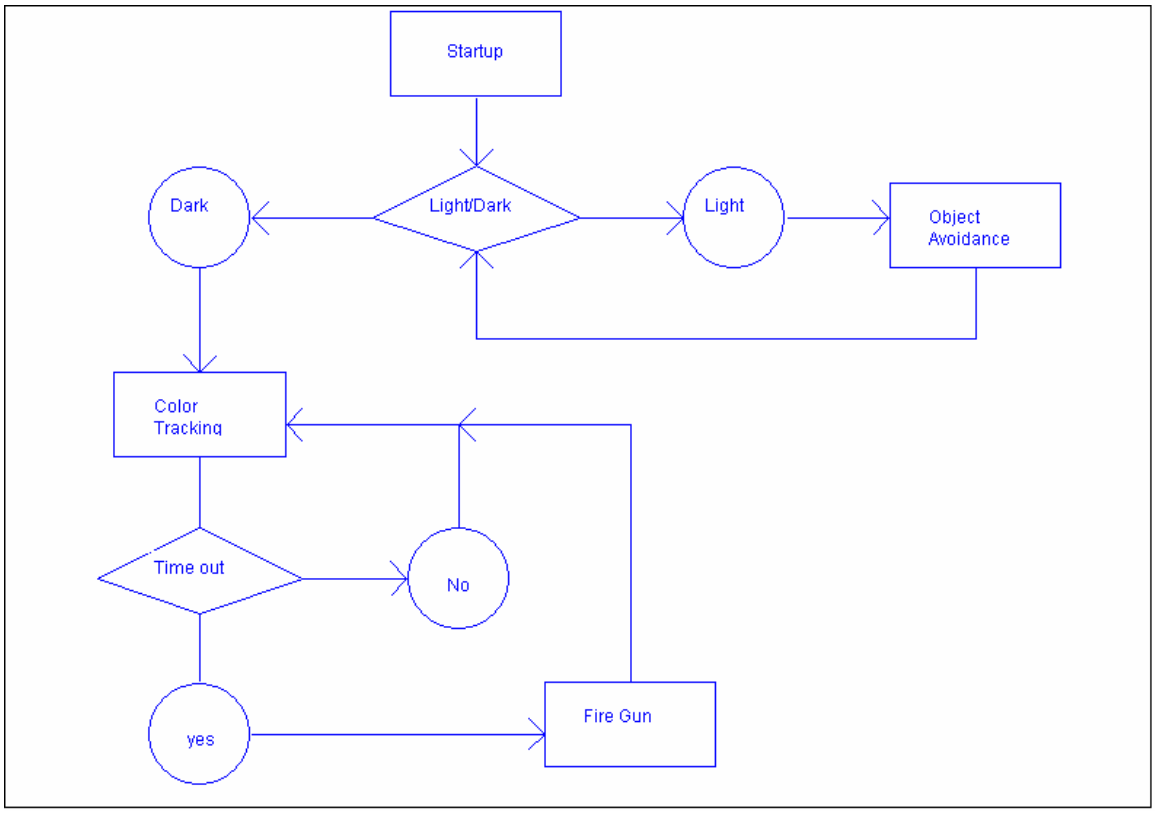


Figure 2. State Machine

Mobile Platform

The platform that all systems are connected to is an eight inches in diameter by 6 inches tall (see fig 3.) The platform is design to be small, so it can avoid detection, and to be able to hide easier. Two continuous rotating servos dive two wheels making the platform able to rotating 360 degrees around. On this platform the camera will be on top of platform and the projectile shooter is mounted on top of the camera. The IR sensors are mounted on the front. The servos, mircocontroller, and batteries will be housed inside the platform. The projectile launcher was redesigned from the original idea. (see fig 4.)

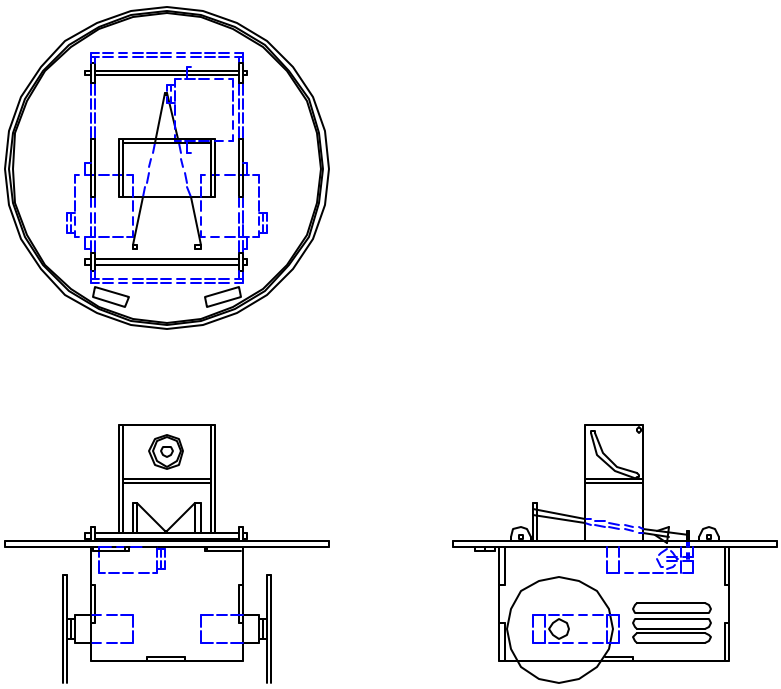


Figure 3. (Prototype design).

The following picture is the final design of S.N.I.P.E.R.; notice the rotating rubber band gun.

Figure 4. (Final design.)

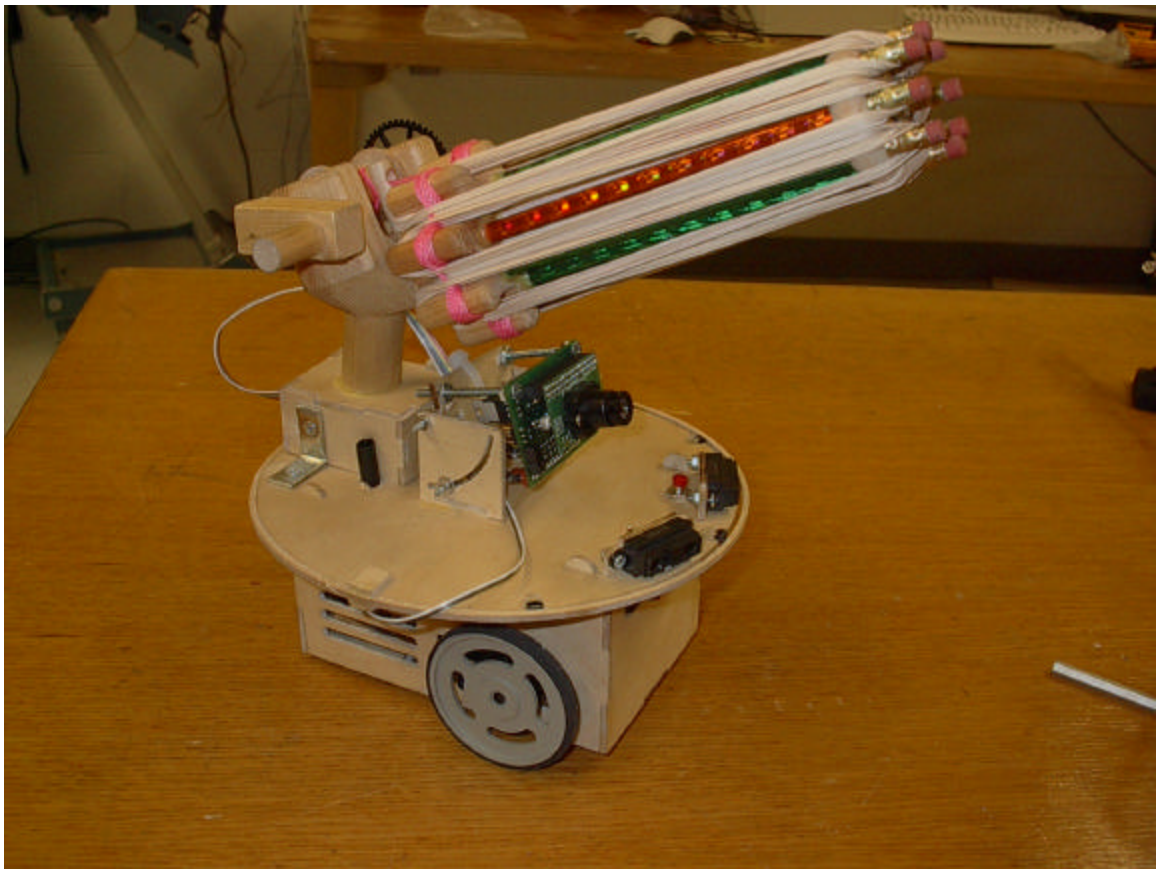


Figure5. (Platform Layout.)

Actuation

There are two systems of actuations. First two Parallax standard BB servos (see fig 7.) are hacked for continuous movement and connected to the wheels. These servos come pre-hacked, instead of hacking them myself. Originally, I hacked the servos myself, but found that they lost calibration very easy and burned out. These servos used for movement for the robot. These servos are connected to the PWM (Pulse Width Modulation) on the microcontroller. PWM is setup to generate a 60Hz signal. The duty cycle of the signal determined the direction and speed of the servos.

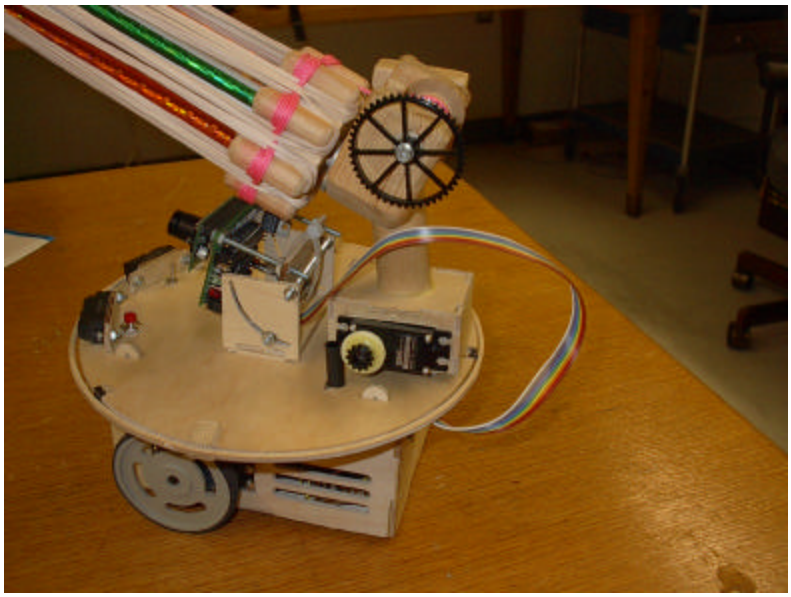


Figure 6. Chain drive for gun.

Figure 7. Parallax Servos

The second system is the firing of the projectile shooter. The projectile shooter is a rotating rubber band gun (see fig 6.) The gun is connected to a hacked servo and a chain drive system that rotates the gun while firing the rubber band projectile. This servo is a GWS S03TXF 2BB High Torque servo (see fig. 8.) This servo has a better motor, which generates a larger 69in-oz of torque instead of a standard 47in-oz torque. This high torque is needed to rotate the gun. This servo is also connected to the PWM on the microcontroller. In designing a motor control system there were some things to think about. Motor and electronics do not like to be on the same circuit. Motors require a lot of current and electronics do not, so I had to make sure the motors were on different power sources. Also, I had to make sure that I had a proper servo for this job. In the case of S.N.I.P.E.R. a standard servos was not enough to turn the gun. The motor's rotor stopped moving, which caused the motor to draw the maximum amount of current. This destroyed all the electronic on the rest of the servos on the circuit making them useless.

Fig 8. GWS High Torque Servos

Sensors

Sensors are used so the robot can interact with its environment. Main sensors used are, IR proximity sensors, CMUcam digital camera for vision purposes, and Bump ring system.

Each sensor is described below.

Two Sharp GP2D12 IR proximity sensors (see fig 9.) are used on the front to detect a structure in front of the robot. The IR sensors are self modulated and demodulated. The analog signal comes out of each sensor and is connected to the A/D on the microcontroller. Using an 8-bit resolution the values form 0 to 255 can be determined on each sensor. This will allow the robot detect items from a few inches to a foot away.

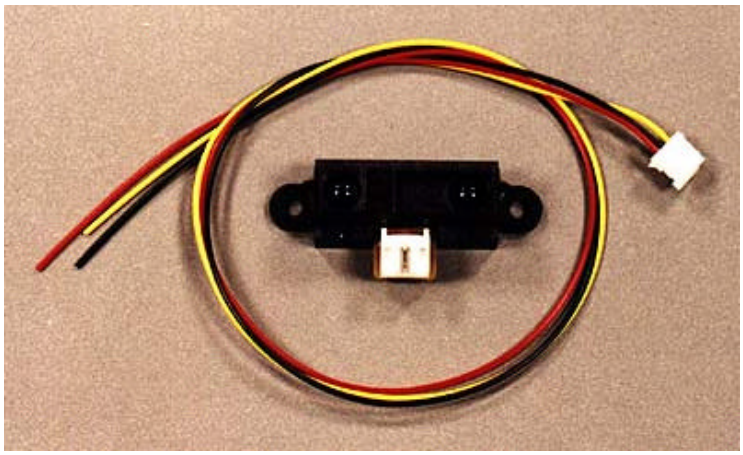


Figure 9. Sharp GP2D12 IR proximity sensor

The bump switch sensor is a set of four switches that are used to determine if a collision has occurred with the robot. The purpose of the bump switch is if the IR sensors did not pick up a structure the bump switch is the last effort in getting the robot to change direction. The circuit is from Uriel Rodriquez, and is used to save I/O pin on the mircocontroller. The system works by using four switches connected around a ring. The output of the network is tied to an A/D pin. Using four different resistor values, a

multiple voltage divider network can be made. The voltage divider gives each switch or combination of switches a different voltage value. The mircocontroller take these voltages and use the move to move from the structure.

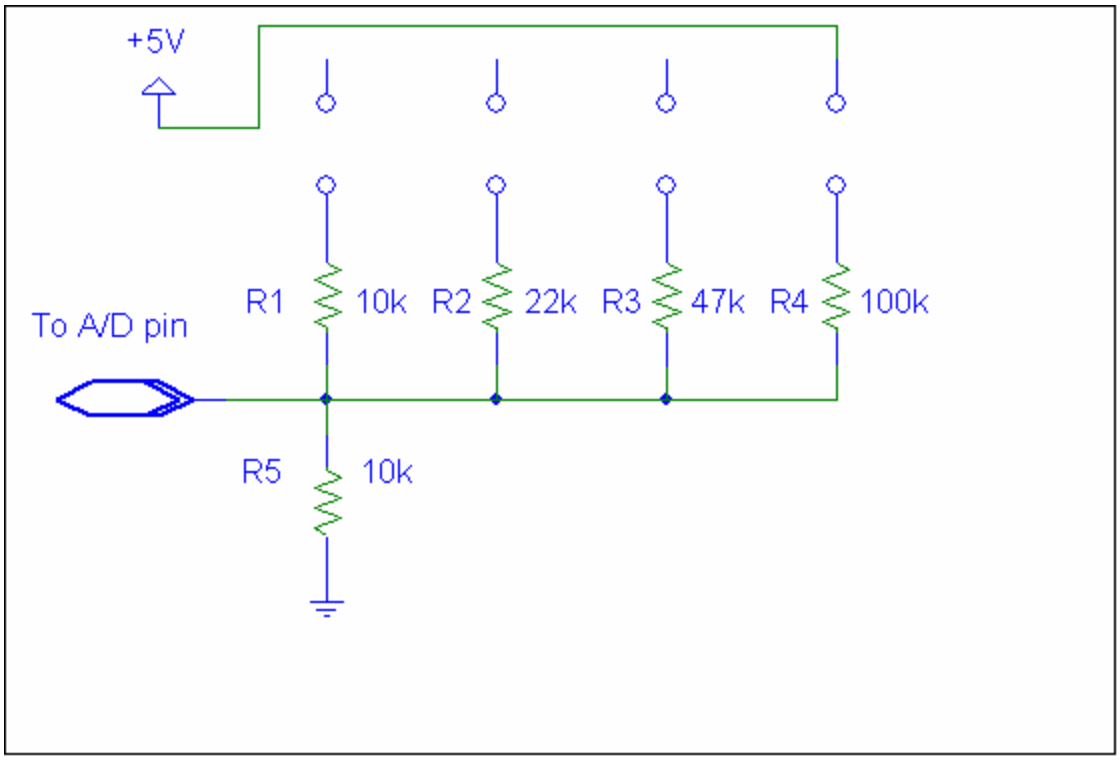


Figure 10. Bump switch network

The photocell sensor detects light and dark. The photocell is a resistor that varies depending on the light level. Photocells are very sensitive to light, so it is necessary to columnize the photocell to control the focus of the light (see fig. 11.) Like the bump sensors if the photocell is connect to a voltage divider and then connected to the A/D port a range from 0 to 255 to determine light value.

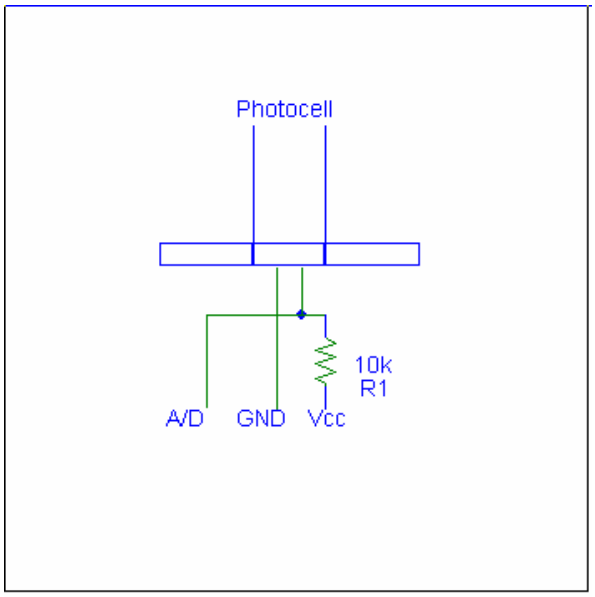


Figure 11. Photocell schematics

CMUcam Digital Camera

S.N.I.P.E.R. uses a CMUcam to track people's movement. The CMUcam is a CMOS digital camera that is controlled with a SX-28 microcontroller (see fig 12.)

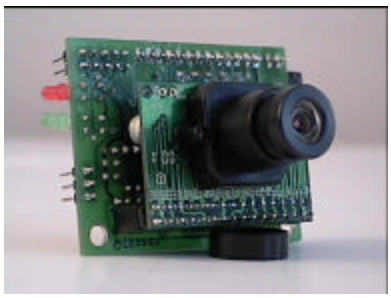
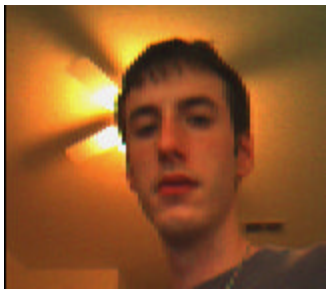


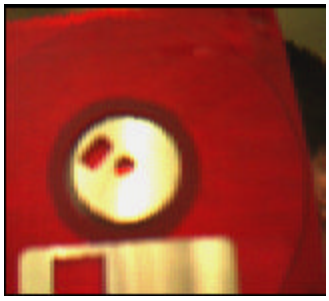
Figure 12. CMUcam Digital Camera

Image information is transmitted serially to the main microcontroller. The camera will work by taking the RGB values of the tracked item. It then, determines the center of mass of the tacked item. The camera then sends mass Y, mass X, to the main microcontroller. The main microcontroller then uses this information to control the motor movement to keep S.N.I.P.E.R. on center of the tracked item. S.N.I.P.E.R. I have at this time only connected the camera to my laptop. I have had success tracking a color

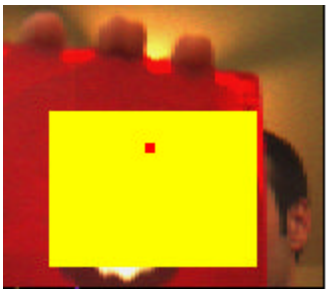
that I determined. The next step to attached the CMUcam to S.N.I.P.E.R. This is how I will connect things. First the Atmel 323 Microprocessor has a serial connection that I can use with the camera. I then can send command to the camera to track color. The camera will send mean values of the RGB and since the camera viewing size is 80 X143 pixels I can deter mind if the color is center, left, or right. With this data I can move S.N.I.P.E.R left or right two keep in line with the object that is being tracked. Here is some test images.



Picture of me taken by CMUcam



A Item to track



CMUcam Tracking the color

Behaviors

S.N.I.P.E.R. use four behaviors depending on what sensor it is using. The first behavior is to find a hiding place with obstacle avoidance. This behavior is achieved by roaming around using obstacle avoidance until the photocell finds a dark place. S.N.I.P.E.R. constantly checks the photocells to check the lighting level at a given time. Depending on lighting level S.N.I.P.E.R. will choice which behavior to do next. If S.N.I.P.E.R. determined that the light level to high then it will move the obstacle avoidance behavior. If S.N.I.P.E.R. determined that the light level is low then it will move to tracking behavior.

Obstacle avoidance uses two IR sensors to determine which motor to move. If it detect a object in front of the left IR sensor, then S.N.I.P.E.R. will move the left forward and the right to stop, so it will turn right. If it detect a object in front of the right IR sensor, then S.N.I.P.E.R. will move the right forward and the left to stop, so it will turn left. Finally if both IR sensors detect an object then S.N.I.P.E.R. will back and turn left. Once the dark place is found the S.N.I.P.E.R. goes to the hiding behavior.

In the hiding behavior the robot use the camera to find a color to appear. When the color appears goes to the next behavior, tracking. The tracking behavior uses the camera to find the center of mass of the color and move the robot to keep itself align with the target.

Finally, the last behavior is firing the projectile. When the target passes through the center of the camera a number of times. S.N.I.P.E.R. will fire a few projectiles at the target and then go back to the tracking behavior.

Experimental Layout and Results

The main code of S.N.I.P.E.R. was developed in parts. After successful completion of each part of the code, the code was then put together into the main code.

All robots in the IMDL are required to have obstacle avoidance, so that was the first to be developed. First the IR sensors where wired and connected to the A/D on the microcontroller. I wrote a simple code that setup the A/D and would store the value of the sensor on LEDS of the mircocontroller to determine the value the sensor found. In the case of S.N.I.P.E.R. it uses the value of 70 on the IR sensor to determine to move. This gives S.N.I.P.E.R. about 4"-5" away from the object to move. The next part was to get the motors to work by microcontroller control. The PWM (Pulse Width Modulation) was a little tricky. The Atmel 323 has four PWM channels, two 8 bits, and one 16 bits that can be split into two 8 bits channels. Instead of using interrupts to make a pulse, the PWM is a pulse that the microcontroller makes, and only require a prescaler bit to make the pulse width, and a value to change the duty cycle. I had trouble figuring this out. Kyle Tripician helped me out in this area. Once the IR sensor and the motor are under microcontroller control, obstacle avoidance is just "if" statements stating if the left sensor see something turn right, ect.

The photocell and bump switch where easily completed. Both are voltage divider connected to the A/D. To determining their values I simply connected them with the A/D and store the number out on the LEDS.

The CUMcam is a difficult sensor to work with. The camera itself is fairly easy to work. It has pre-program commands to certain functions, like tacking color, middle mass, mean of color, etc. The hard part is these commands send in ASCII through a

serial port. Kyle Tripician develop a code to communicate with the camera with the UART on the mircontroller, this simplify the task using the camera. Using a camera lighting is very important, because lighting level will change the color value that you determined. It is very important that if you order a CMUcam make sure that you get a wide lens with an IR coating. This improves the quality of the picture it takes and making light level less important.

Conclusion

S.N.I.P.E.R. is a successful robot. It completed all of its original design behaviors. It is also a great achievement and a dream for me to design and build a real one of a kind robot. S.N.I.P.E.R. is able to obstacle avoid objects every time. It is able to hide, track colors, and fire like it has been designed to do. It was a lot of hard work it use every thing I have learned in electrical engineering. I took this class as an undergraduate electrical engineering student. It was the best real work class I have taken. I finally got to create sometime, which is the reason why I wanted to become an engineer.

The most trouble I have was with grounds. Part of the robot would stop working, because I would forget to connect all the ground together. Another problem I have was the gun. It was a great Idea to use it, but it was hard to get a servo to be able to turn it. I burned up many servos because the gun would require so much torque that the motor's rotors would stop and cause I current spike, and burn out the servo, and very servos circuit board on the circuit.

This project used so many aspects of electrical engineering that I feel I really learned a lot. It is one thing to learn from theory in classes and a whole new ball game to apply it in the real world. Overall, I found this robot project to be a very difficult and time-consuming, but also a very great experience which I have gained much needed confidence in myself for future projects. I suggest this class for everyone who actually wants to learn and experience this learning hands-on.

Documentation

Credit

I would like to give credit and thanks to all the help I was given by Dr. Arroyo for teaching this class. Jason Plew and Uriel Rodriquez for help on concepts and being great T.A. Specially thanks to Kyle Tripician and Brain Ruck, for being great friend and helping with the code, and ideas. I would not be able to finish without their help.

Websites of relative importance

(Mircocontrollers)
www.atmel.com
http://www.prllc.com
Good site for data sheets, same code, etc. for the ATMEL mircocontroller.

(CMUcam)
www.seattlerobotics.com
Origin of the camera, ask for the proper IR lens.

www.acroname.com
More CMUcam stuff, they sell it too, sample codes, IR sensor, Servos, more

Appendix

/*SNIPER Code Created by Jordan Wood 03/20/03*/

```
#include <io.h>
#include <sig-avr.h>
#include <stdlib.h>
#include <interrupt.h>
#include <progmem.h>

typedef unsigned int u8;
typedef unsigned char u08;
typedef          char s08;
typedef unsigned short u16;
typedef          short s16;
u08 lir;
u08 cir;
u08 rir;
u08 rmotor=2;
u08 lmotor=1;
u08 distc=0x70;
u08 distf=0x67;
u08 distrlf=0x30;
int fwd=100;
int turn=50;
int rev=-100;
int stop=0;
int s,d,x,y,z,a,b,c,j,k;
u08 irval[5];
int adchan[5];
int bump;
int photo1;
int photo2;
int motors=0x12;
int motorg=0x20;
int track;
int count;
#include "uart.h"
#include "delay.h"
volatile u8 temp;
volatile u8 i=0;
volatile u8 cmudat[9];
```

```
//Initialize UART
//'baud' is the baud register divider
void uartinit(void)
{
  /* Set baud rate */
  outp(0x98,UCSRB);
  outp(0x86,UCSRC);
  outp(0x00,UBRRH);
  outp(0x09,UBRRL);
  /* Enable Receiver and Transmitter */

          // 1 stop bit
}

//Send a single byte of data
//'data' is the byte sent
void uarttransmit(unsigned char data)
{
  /* Wait for empty transmit buffer */
  while (!(UCSRA & (1<<UDRE))){}
  /* Put data into buffer, sends the data */
  outp(data,UDR);
}



//Send a given EOS terminated string
void uartstring(unsigned char * myStringIn)
{
  unsigned char *myString = myStringIn;
  unsigned char ch1;
  unsigned char gotNULL = 0;
  ch1= *myString++;
  while(!gotNULL){
    uarttransmit(ch1);
    ch1 = *myString++;
    if(ch1 == '\r'){
      gotNULL = 1;
      uarttransmit(ch1);
      }

    }
}
```

```
SIGNAL(SIG_UART_RECV){
temp=inp(UDR);
if(temp != 0x3A){
if(temp == 0x20 || i==9){
i=0;
}

else if(i==0){

if(temp == 0xFF){

cmudat[i]=temp;
i++;
}
}

else if(temp !=0x20 && i<9){

cmudat[i]=temp;
i++;
}
}

}
/*void blink(void){
uartstring("L1 1\r");
delay(1500);
delay(1500);
delay(15000);
uartstring("L1 0\r");
delay(15000);
delay(1500);
delay(1500);

}*/

void cmu(void){
outp(0xFF,DDRC);
delay(10000);
uartinit();
delay(10000);
uartstring("PM 1\r");
delay(1000);
uartstring("RM 3\r");
delay(1000);
```

```
sei();
count=25;
while(1){
uartstring("TC 42 60 33 50 30 90\r");
track=cmudat[2];
outp(cmudat[2],PORTC);
delay(10000);


if(track>0 && track<15){
servo(lmotor,rev);
servo(rmotor,fwd);
delay(500000);
servo(lmotor,stop);
servo(rmotor,stop);
}

if(track>30 && track<50){
servo(lmotor,stop);
servo(rmotor,stop);
for(x=0;x<40;x++){}
  count--;
if(count==0){

outp(0x21,OCR2);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);
delay(900000000);

outp(0x12,OCR2);

count=25;
}

}

if(track>65 && track<85){
```

```
servo(lmotor,fwd);
servo(rmotor,rev);
delay(500000);
servo(lmotor,stop);
servo(rmotor,stop);

}
}
}

void servo(s,d){
//s represents "side" lmotor=1 and rmotor=2, d is direction, i.e fwd, rev//
if(s==lmotor){
if(d==fwd){
outp(0x00,OCR1BH);
outp(0x14,OCR1BL); //left motor fastest forward//
}
else if(d==rev){
outp(0x00,OCR1BH);
//left motor reverse//
outp(0x10,OCR1BL);
}
else if(d==turn){
outp(0x00,OCR1BH);
//left motor slows to enable left turn//

outp(0x0e,OCR1BL);
}
else if(d==stop){
outp(0x00,OCR1BH);
//left motor stops//

outp(0x12,OCR1BL);
}
}

else if(s==rmotor){
if(d==fwd){
outp(0x00,OCR1AH);
outp(0x0f,OCR1AL);
//right motor fastest forward//
}
else if(d==rev){

outp(0x00,OCR1AH);
//right motor reverse//
```

```
outp(0x13,OCR1AL);

}
else if(d==turn){
outp(0x00,OCR1AH);
//right motor slows to enable left turn//

outp(0x15,OCR1AL);
}
else if(d==stop){
outp(0x00,OCR1AH);
//right motor stops//

outp(0x12,OCR1AL);
}
}
}

void conv(void){
c=0x00;
//intialize rangers for conversion//

adchan[0]=0x20;
//left adjusted channel 0//
adchan[1]=0x21;
//left adjusted channel 1//

adchan[2]=0x22;
//left adjusted channel 2//
adchan[3]=0x23;
adchan[4]=0x24;
for(b=1;b<=5;b++){
outp(adchan[c],ADMUX);
sbi(ADCSR,ADSC);
/* start conversion */
loop_until_bit_is_clear(ADCSR,ADSC);
/* checking if done */
  irval[c]=inp(ADCH);
//place value in irval array//
c++;
}
lir=irval[0];
//store val in left ir//
rir=irval[1];
//store val in right ir//
bump=irval[2];
```

```
//store bump value//

photo1=irval[3];
//store photo1 value//
photo2=irval[4];
//store photo2 value//
outp(photo1,PORTC);
}

int main(void){                         //set up PWM//

outp(0xFF,DDRD);    //set up PWM, PORTD sent send signals to motors//
outp(0x00,DDRA);    //set bit7:0 PORTA as inputs//
outp(0xFF,DDRC);    //set outputs for leds
outp(0xA1,TCCR1A);          //8 bit PWM//
outp(0x04,TCCR1B); //prescale clock by 256//
outp(0x80,ADCSR);   //turn on A/D
sbi(ADMUX,ADLAR);          //left adjust the A/D conversion


outp(0x66,TCCR2);
outp(0x00,TCNT2);


while(1){
conv();

if(photo1<=20 && photo2<=20){

servo(lmotor,stop);
servo(rmotor,stop);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
for(z=0;z<255;z++){}

cmu();
}



else if(photo1>=20 && photo2>=20){

            if(bump>=40 && bump<=50){
servo(lmotor,stop);
servo(rmotor,stop);
for(x=0;x<255;x++){}
```

```
for(y=0;y<255;y++){}
for(z=0;z<255;z++){}

servo(lmotor,turn);
servo(rmotor,turn);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
for(z=0;z<255;z++){}
}


else if(bump>=150 && bump<=160){
servo(lmotor,stop);
servo(rmotor,stop);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
for(z=0;z<255;z++){}
}


else if(bump>=10 && bump<=30){
servo(lmotor,stop);
servo(rmotor,turn);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
}


 else if(bump>=100 && bump<=120){
servo(lmotor,turn);
servo(rmotor,stop);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
}


if(lir<distf && rir<distf){
servo(lmotor,fwd);
servo(rmotor,fwd);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
for(z=0;z<255;z++){}
}

if(lir<distf && rir>distf){
servo(lmotor,fwd);
```

```
servo(rmotor,stop);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
}



if(lir>distf && rir<distf){
servo(lmotor,stop);
servo(rmotor,fwd);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}
}


if(lir>distc && rir>distc){
servo(lmotor,rev);
servo(rmotor,rev);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}

servo(lmotor,stop);
servo(rmotor,fwd);
for(x=0;x<255;x++){}
for(y=0;y<255;y++){}


}
}
}
}
```