# FETCH

**Final Report**
**EEL 5666**
**Carley Thompson**
**04-23-02**

## Table of Contents

## Abstract

Fetch is an autonomous robot whose purpose is to entertain my dog. Fetch releases a ball when it hears a loud noise. Once it has released all of its balls, it collects the balls one at a time and returns them to a home base. Once it collects and returns all of the balls, it waits to be refilled and reset by a human.

## Executive Summary

The main purpose of Fetch is to entertain my ball crazy Cocker Spaniel and keep him busy. Fetch can successfully avoid obstacles while listening for a loud sound. Once he detects a loud sound, Fetch can successfully release a ball and continue avoiding obstacles until the break beam sensor detects that there are no more balls in the ball container. Once there are no more balls in the ball container, the robot searches for the balls he had released. Once one ball is in the scoop, the gate traps the found ball and Fetch takes the ball to home base. After Fetch leaves the ball at home base, he goes in search of the remaining dropped balls. Once Fetch has found all of the balls that he released, he waits at the home base until he is reloaded and reset by a human.

## Introduction

The idea for this robot came to me one day while I was playing with my dog. I have a three year old Cocker Spaniel who is absolutely ball crazy. His favorite thing is to chase balls around. So I thought, why not create a robot that can dispense a ball whenever he wants it to. I also decided that when the robot had no more balls to dispense, it should wander around and collect them. Thus came the creation of Fetch. In the rest of this paper I will discuss the whole system integrated together, the design and construction of the mobile platform, the actuation both for the wheels and the extra servos I used, a description of each sensor I used and how I used them, and the different behaviors Fetch was expected to perform.

## Integrated System

For Fetch I chose the double stack boards, the MRSX01 and the MRC11, also know and the Talrik boards. I chose these as the brains of my robot because initially I did not know exactly how many sensors I wanted to use and I wanted to have as many options as possible. The final version of Fetch uses two completely hacked servos for the wheels, two non-hacked servos for the releasing and collecting balls, four IR sensors, six tactile switches, four cadmium sulfide cells (CdS cells), one laser diode, two flex (bend) sensors, and a small condenser microphone to perform all of his designated behaviors.

## Mobile Platform

The platform for Fetch was designed originally to be round for easier maneuvering. It ended up being basically round with some parts sticking out. I decided that I wanted the balls to be racquet balls and that I wanted them to dispense through some pvc pipe. After buying the pvc pipe, I made measurements of the pvc pipe and a small bucket to be used as the ball container to determine how large the outer part of the robot should be. After I got a rough estimate I went in search of a round container that would work. I found a metal trashcan at Target® that had almost the exact dimensions I wanted. I got some large wheels from an old radio control robot that I had lying around. I built a wooden platform from an eighth inch thick piece of plywood to mount the wheels on. I then bolted this platform to the bottom of the trashcan. I cut a hole in the front of the trashcan so that the balls could come out. The electronics fit nicely under the ball container. Inside the ball container I built a ramp for the balls to enter on. Under this ramp I attached a servo to control a door that only lets out one ball at a time. I cut a hole in the bottom of the ball container on the other side of the door and attached the pvc pipe to that hole. I needed some way to collect the balls. Originally I wanted to actually scoop up the balls and dump them back into the ball container, but I decided that was too mechanically complicated. Before I decided it was too complicated, I bought an enormous Tonka® bulldozer at Toys-R-Us® which I ripped the scoop off of (see figure 1 in appendix). I decided to use this scoop anyway. I extended the sides of it with balsa wood so that it could corral the ball. Then I added a servo on one of the sides and attached a gate so that I could trap the ball.

Here are some lessons I learned:

1) It is extremely difficult to cut a hole two inches in diameter in metal.
2) If you are using a power tool without enough lubrication, the metal gets really hot.
3) Cutting soft plastic with a Dremel® is not a good idea. The plastic melts and ruins the cutting disc. (and burns your skin)
4) The heavier your robot, the faster your batteries wear out.

## Actuation

For all four of my servos, both hacked and non-hacked, I used the Futaba S3003. These servos are rated 44.4 oz-inch at 6VDC. For the wheels, I completely hacked two of these servos. I removed the plastic tabs that prevent it from rotating 360º and I removed the control circuitry. I used one of the other servos for the door in the ball container and I used that last servo for the gate that traps the found ball. I did not hack these servos at all because I wanted them to have limited ranges of motion.

## Sensors

My robot uses IR sensors, CdS cells, tactile sensors, sound detection sensor, and my unique sensor. Here is a description of each sensor.

I used four of the Sharp GP2D12 IR sensors. These have the detector and emitter in one package and they do not need to be hacked. I used three of these for obstacle avoidance and all four of them for finding the balls.

I used CdS cells to detect a bright light source. This bright light serves as the robot's home base. This is where it brings each ball that it finds.

I used tactile sensors for bump detection. Right now these sensors are useless because they do not stick out far enough to ever be touched. I am using the front sensor to start the robot.

I created a sound detection circuit, with the help of Michael Maines. This circuit takes in an input from a microphone, amplifies the signal, and chops off the negative part of the signal. (see figure 2 in appendix) The output is connected to the analog to digital ports on the expansion board. When the microphone detects a loud sound, such as a clap or a dog bark, it triggers a servo to release a ball. I had some trouble with the microphone. The servos driving the wheels and the vibration of the robot would set off the microphone circuit. Uriel Rodriguez suggested I raise the microphone up on a pole above the robot. That helped tremendously, but I still had to raise the threshold level for the microphone. This means that the sound must be incredibly loud for the robot to release a ball.

My unique sensor is actually two sensors working independently of each other. The first is a flex sensor, also called a bend sensor. The second sensor I am using is a break beam sensor of sorts. It is made up of a laser diode and a CdS cell. Originally I wanted to use a pressure sensor to determine if there where any balls left in the container. This proved to be more difficult than I imagined though. For starters, I could only find pressure sensors that measured either a liquid or a gas, not a mass. Therefore, I settled on a flex sensor. I was originally going to use two flex sensors, one for the scoop and one for the container. However, Aamir said that a flex sensor by itself is not that special and that I should use a break beam sensor. Since a normal break beam sensor is not incredibly useful to me because of the rather small distance between the light source and the detector, I decided to make my own. Next is a detailed description of each unique sensor.

The flex sensor is a long thin piece of a plastic like material with two pins sticking out of one end. As the device is bent toward a $90^o$ angle, the resistance increases from 10kΩ to around 40kΩ. This process is highly non-linear. I used the sensor in a voltage divider circuit connected to an op amp to utilize the flex sensor's ability to change resistance (see Figure 3 in appendix). I modified this circuit slightly by using negative feedback to increase the gain of the op amp. (see Figure 4 in appendix) I connected the output of the op amp to one of the IR ports on the MRSX01 board. This way I can convert the voltage into a digital number and use a program to compare the digital numbers to determine if the sensor is bent or not. I am using this sensor in the scoop on my robot. The scoop will be used to pick up a ball and carry it to a home base. This way I will know if there is a ball in the scoop. I wrote a short program to determine what the range of numbers is (see flextest1 in appendix) and then I used these numbers to write another program that will tell me whether or not there is a ball in the scoop (see flextest2 in appendix). Eventually this program will jump to the find home program when there is a ball in the scoop. If there is no ball in the scoop, the robot will search for balls. I did a screen capture of the program running in the terminal to show what the values are when the sensor is bent and unbent (see Figure 5 in appendix).

I have some experience with designing a break beam sensor because that is basically what I designed for my senior design project. So I planned to use the laser diode and the protection circuit for the laser diode from my senior design project and I decided to use a CdS cell to detect the laser beam. However, I somehow I accidentally blew up the laser diode so now I am using a laser pointer instead. The laser will shine directly at the CdS cell, which will change in resistance as it goes from light to dark and vice versa. In the dark the CdS cell has a resistance of around 3MΩ and when the laser light is shining on it, the resistance drops to around 300Ω. I connected the CdS cell to one of the CdS ports on the MRSX01 board. They are laid out on the board in such a way that the CdS cell is the top part of a voltage divider. The voltage that comes out is connected to the analog to digital converter so I get a digital value that I can use a program to tell whether the laser beam is broken or not. I am using this sensor to determine if there are any balls left in the ball container. When there are no more balls my robot will go in search of balls to take to the home base. I wrote a short program to determine what the range of numbers is (see cds1 in appendix) and then I used these numbers to write another program that will tell you whether or not there are any more balls in the container (see cds2 in appendix). Eventually the program will jump to a find balls routine when there are no more balls in the container. While there are still balls in the container, the robot will dispense the balls. I did a screen capture of the program running in the terminal to show what the values are when the beam is broken and unbroken (see Figure 6 in appendix)

Here are some lessons I learned:

1) IR sensors can not handle 10Vdc.
2) Flex sensors are extremely temperature dependent.
3) I am a horrible programmer.
4) Don't put a semicolon after your while statement if you want it to perform the functions you put in the curly brackets.

5) All IR sensors are not created equal.

## Behaviors

The behaviors Fetch can perform are: obstacle avoidance, release of balls, collection of balls and finding home.

Fetch uses three IR sensors to avoid obstacles. When an obstacle is detected he backs up a little and turns in a random direction for a random amount of time.

The ball release is sound activated. While Fetch is wandering around avoiding obstacles, he is listening for a loud sound. When a loud sound is detected, the door in the ball container releases one ball then continues wandering until another sound is detected. When there are no more balls in the container the break beam sensor detects this and the robot goes into find ball mode.

Fetch uses all four IR sensors to locate the balls. When a ball enters the scoop area and touches one of the flex sensors, the gate closes and the robot then searches for home base.

Home base is a bright table lamp. Fetch searches for home using three CdS cells. When he gets within a certain distance to home, he leaves his collected ball and goes off in search of another ball. When he has found all of the balls (a predetermined number), he waits at home until he is refilled and reset.

I used eight LEDs for feedback to show what function the robot is performing at a certain time. I connected these LEDs to the IR emitter output port on the expansion board. Here is what I used each LED for:

1) Obstacle Avoidance

2) Microphone hears a noise

3) Searching for balls

4) Searching for home

5) Found home

6) Leaving ball at home

7) There are more balls to find

8) All balls have been found

## Conclusion

I feel that this robot is still a work in progress. I need to learn more about programming because my code could be much better. If I had more time I would learn how to properly program in C. To the students of the future, I recommend you not try to make your robot do too many behaviors. I think I got carried away with all I hoped it would do. It does every behavior, but it doesn't do any behavior great.

## Appendix

Figure 1

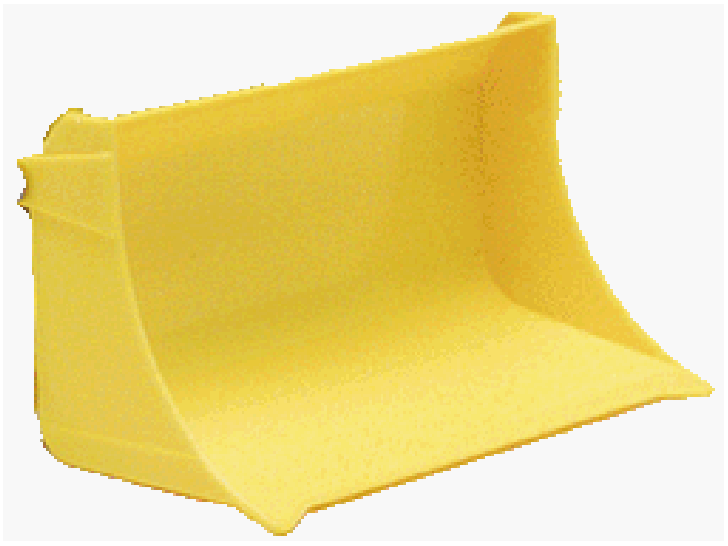


Figure 2

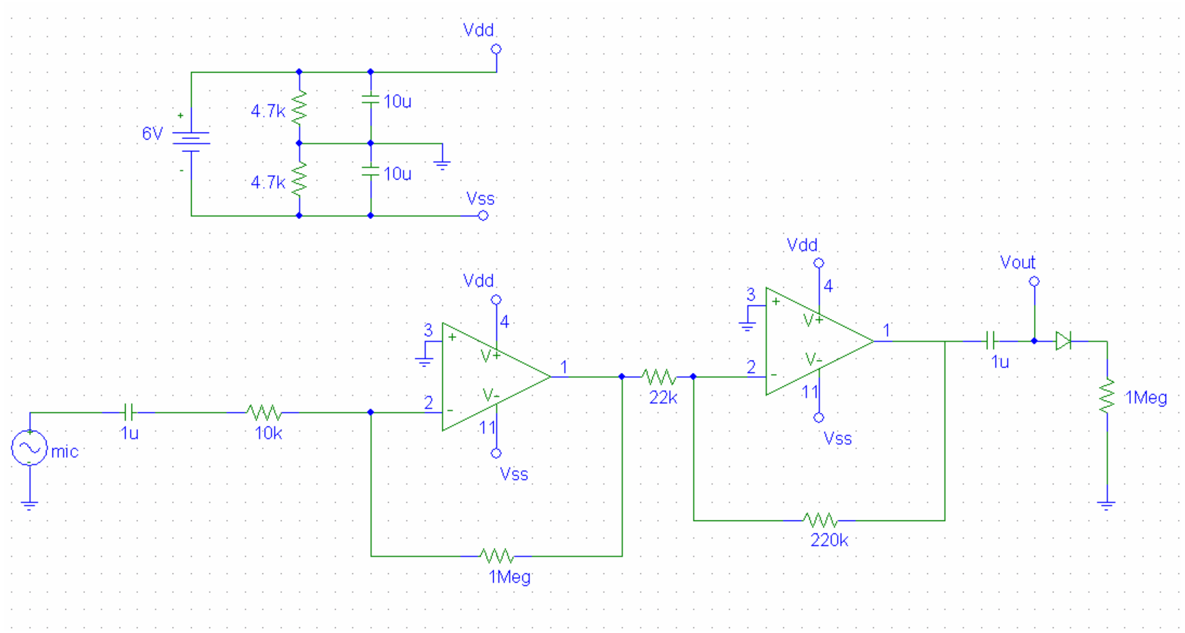
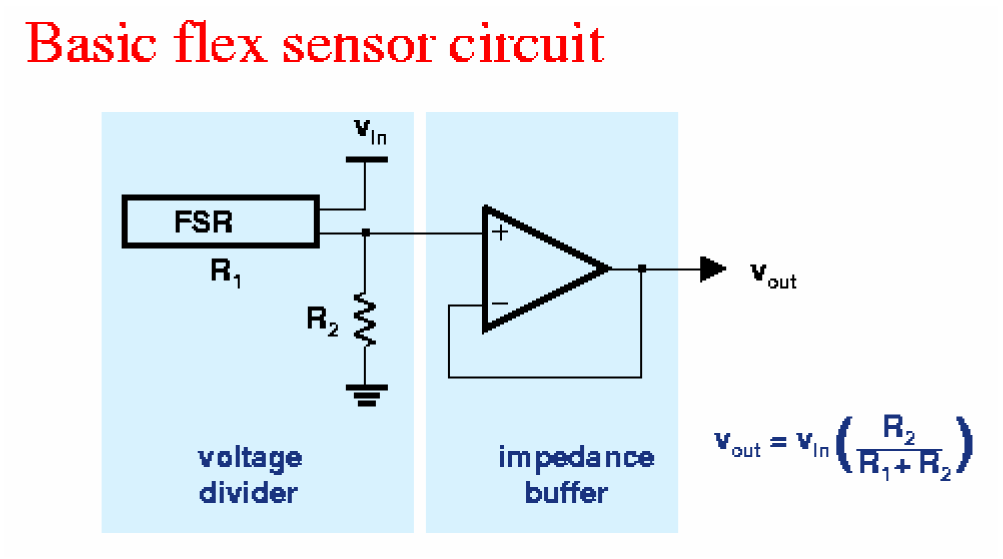
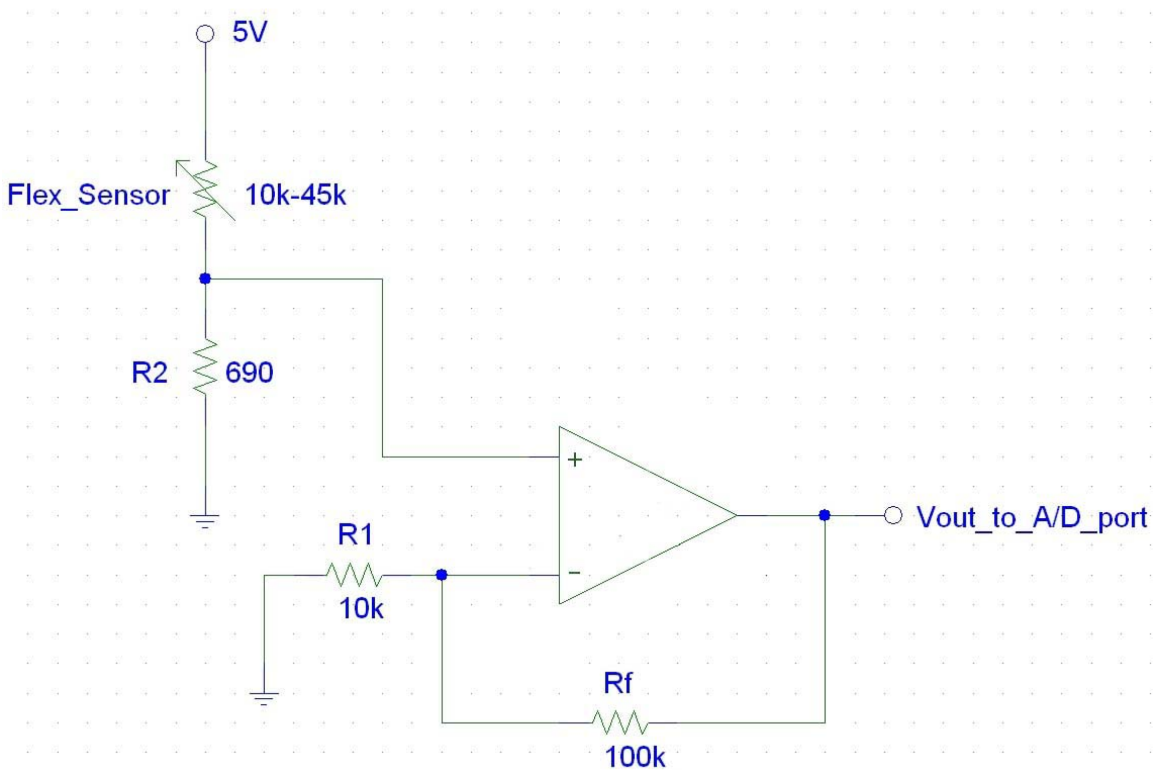


Figure 3


Basic flex sensor circuit
$v_{In}$
FSR
$R_1$
$R_2$
+
−
$v_{out}$
voltage divider
impedance buffer
$v_{out} = v_{In}\left(\frac{R_2}{R_1 + R_2}\right)$

I found this circuit at http://devices.sapp.org/component/flex.

Figure 4



**Flextest1**
```
#include <tkbase.h>
#include <stdio.h>
#define IRE_OUT   *(unsigned char *)(0xffb9)
#define IRE_ALL_ON 0xff

void main(void)
{
  init_clocktk();
  init_analog();
  IRE_OUT = IRE_ALL_ON;
  wait(300);

while(1)
   {
  read_IR();
  printf("the value = %d\n\n",IRDT[1]);
  wait(300);
    }
}
```

**Flestest2**
```
#include <tkbase.h>
#include <stdio.h>
#define IRE_OUT   *(unsigned char *)(0xffb9)
#define IRE_ALL_ON 0xff

void main(void)
{
  init_clocktk();
  init_analog();
  IRE_OUT = IRE_ALL_ON;
  wait(300);

while(1)
   {
  read_IR();
  if (IRDT[1] > 75)
   {      printf("the unbent value = %d\n\n",IRDT[1]);
        wait(300);
   }
  if (IRDT[1] < 75)
   {      printf("the bent value = %d\n\n",IRDT[1]);
        wait(300);
   }
  }
}
```

**cds1**

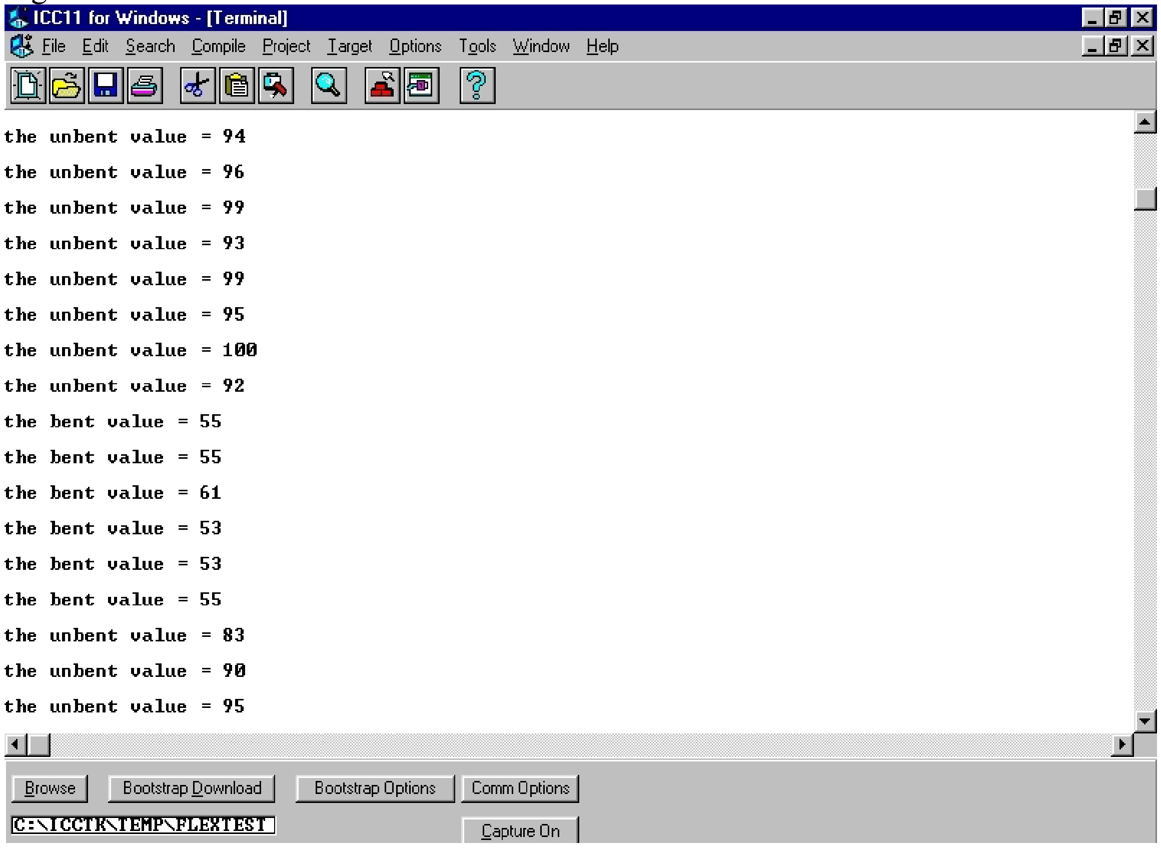
```
#include <tkbase.h>
#include <stdio.h>
#define thresh 100

void main(void)
{
  init_analog();
  init_clocktk();
  wait(300);
  while (1)
{
read_CDS();
printf("value 1 = %d\n\n", CDS[4]);
wait(300);
}
}
```

**cds2**

```
#include <tkbase.h>
#include <stdio.h>

#define hforw 50
#define hrev -50

void find_balls(void);

void main(void)
{
init_analog();
init_clocktk();
init_motortk();
  wait(300);
while (1)
            {
            read_CDS();
if (CDS[4] < 100)
{           printf("there is a ball, value = %d\n\n",CDS[4]);
            wait(300);
}
if (CDS[4] > 100)
```

```
        printf("there are no balls,value = %d\n\n", CDS[4]);
        wait (300);
        find_balls();
        }
}

void find_balls(void)
{
motortk(RIGHT_MOTOR, hforw);
motortk(LEFT_MOTOR, hforw);
}
```

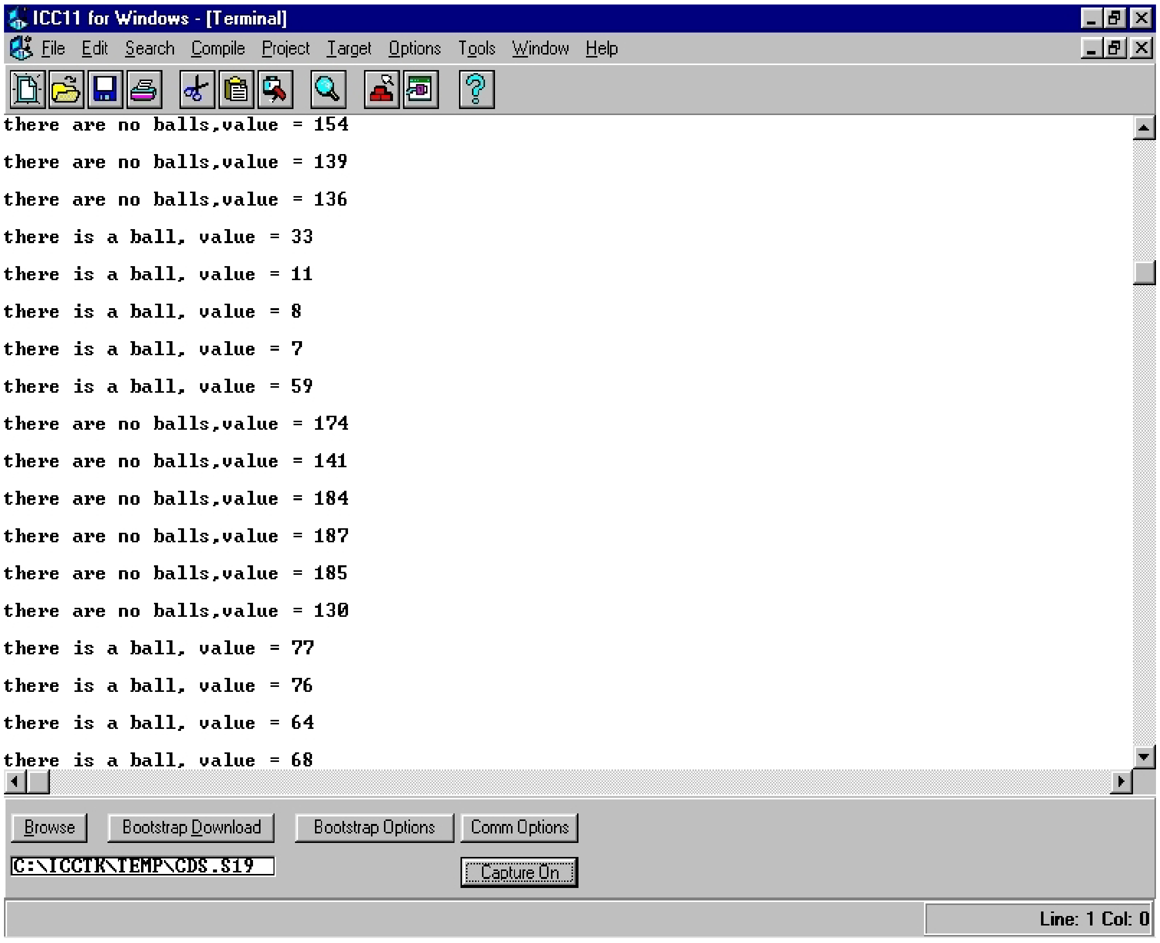
Figure 5


ICC11 for Windows - [Terminal]
File Edit Search Compile Project Target Options Tools Window Help
Browse
Bootstrap Download
Bootstrap Options
Comm Options
Capture On

```
the unbent value = 94
the unbent value = 96
the unbent value = 99
the unbent value = 93
the unbent value = 99
the unbent value = 95
the unbent value = 100
the unbent value = 92
the bent value = 55
the bent value = 55
the bent value = 61
the bent value = 53
the bent value = 53
the bent value = 55
the unbent value = 83
the unbent value = 90
the unbent value = 95
```

C:\ICCTK\TEMP\FLEXTEST

Figure 6


ICC11 for Windows - [Terminal]
File Edit Search Compile Project Target Options Tools Window Help
Browse
Bootstrap Download
Bootstrap Options
Comm Options
Capture On
Line: 1 Col: 0

```
there are no balls,value = 154
there are no balls,value = 139
there are no balls,value = 136
there is a ball, value = 33
there is a ball, value = 11
there is a ball, value = 8
there is a ball, value = 7
there is a ball, value = 59
there are no balls,value = 174
there are no balls,value = 141
there are no balls,value = 184
there are no balls,value = 187
there are no balls,value = 185
there are no balls,value = 130
there is a ball, value = 77
there is a ball, value = 76
there is a ball, value = 64
there is a ball, value = 68
```

C:\ICCTK\TEMP\CDS.S19

## Sources for Parts

Servos

www.servocity.com

Futaba standard servos $10.95 each, FREE shipping, received them in two days.

IR Sensors

www.acroname.com

Sharp GP2D12 $13.50 each, $6.95 shipping UPS ground, received then in 3 – 4 days.

Laser Pointer

Office Depot

Laser pointer key chain $7.99

Flex Sensors

www.jameco.com

Images Company flex sensor $10.95 each, $4.25 USPS Priority Shipping, received them in  2 days

Condenser Microphone

Radio Shack

PC-board condenser microphone element $1.99

## Final Complete Code

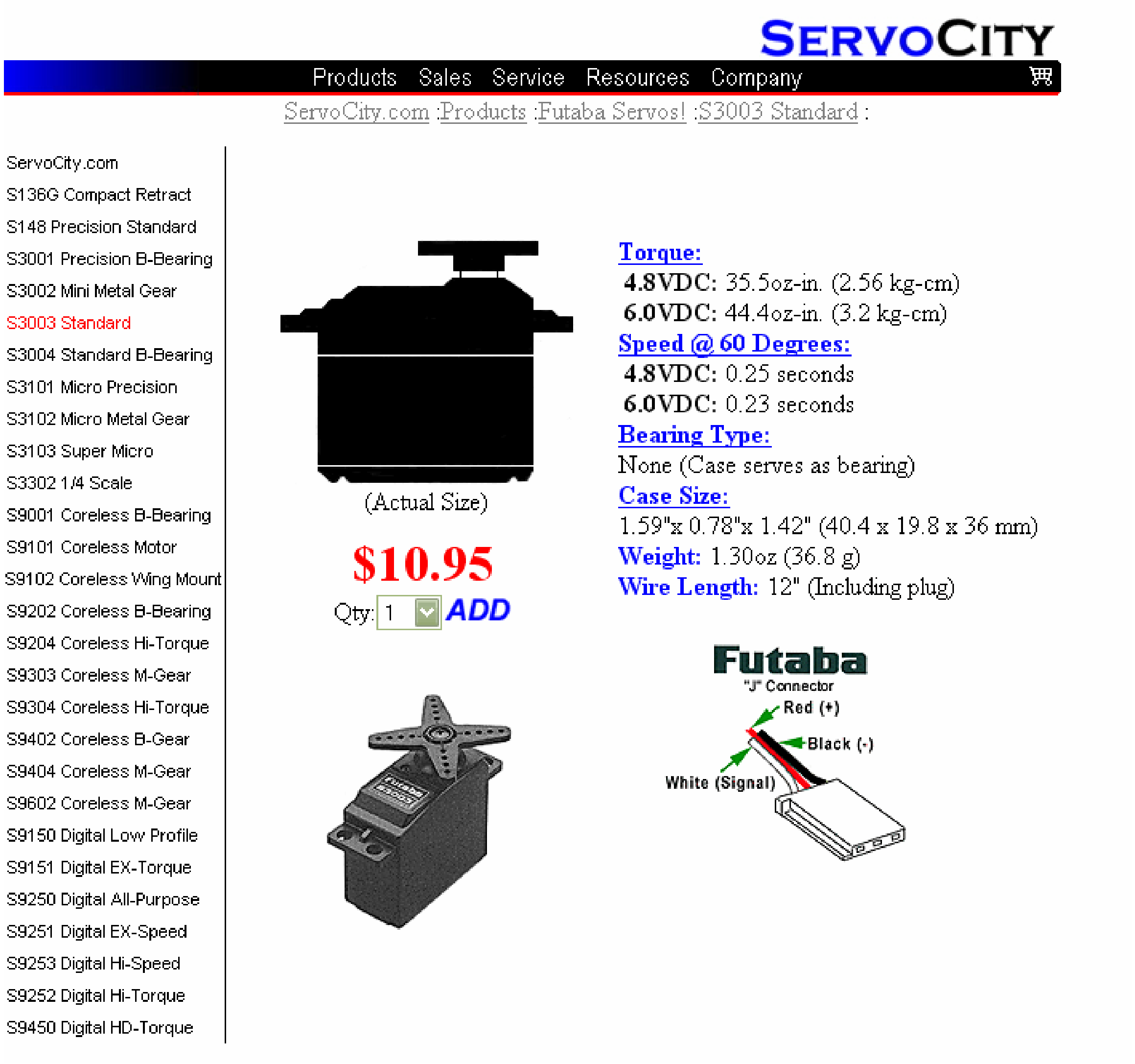
```
#include <tkbase.h>
#include <stdio.h>

#define forw 100
#define rev -100
#define hforw 75
#define hrev -75
#define sforw 35
#define srev -35
#define thresh 70
#define IR_THRESHOLD 90
#define BUMPER_FUZZY_ZERO 12
#define IRE_OUT   *(unsigned char *)(0xffb9)
#define IRE_ALL_OFF 0x00
#define IRE_ALL_ON 0xff
#define LED1 0x01
#define LED2 0x02
#define LED3 0x04
#define LED4 0x08
#define LED5 0x10
#define LED6 0x20
#define LED7 0x40
#define LED8 0x80
#define mic_thresh 100
#define offset 25

  void avoid(void);
  void mic(void);
  void breakbeam(void);
  void findballs(void);
  void turn(void);
  void findhome(void);
  void counter(void);
  void go(void);

  unsigned int bl, IR_delta[NIRDT], IR_Threshold[NIRDT], close1, open1, close, open,
stop;;
  int a, b, c, i, j, k, m, n, p, w, x, y, z;
  int fb, rspeed, lspeed, delta_rspeed, delta_lspeed;
  int average1, bent1, unbent1, average2, bent2, unbent2;

   void main(void)
  {
    init_analog();
```

```
init_motortk();
init_servos();
init_clocktk();
init_serial();
b = 1;
i = 1;
c = 0;
p = 0;
y = 0;
w = 1;
x = 1;
z = 1;
open = -500;
close = 750;
close1 = -500;
open1 = 500;
stop = 0;
IRE_OUT = IRE_ALL_ON;
wait(300);
bl = battery_level();
printf("battery: %d\n\n", bl);
read_IR();

j = IRDT[9];
m = IRDT[8];

for(k = 0; k < 19; k++)
{
  c = IRDT[9];
  j = j + c;
  wait(100);
}
average1 = j/20;
unbent1 = average1;
bent1 = unbent1 - offset;

for(n = 0; n < 19; n++)
{
  p = IRDT[8];
  m = m + p;
  wait(100);
}
average2 = m/20;
unbent2 = average2;
bent2 = unbent2 - offset;
IRE_OUT = IRE_ALL_OFF;
```

```
    while(rear_bumper()<BUMPER_FUZZY_ZERO);
    {
      motortk(RIGHT_MOTOR, forw);
      motortk(LEFT_MOTOR, forw);
    }

    while(b)
    {
      avoid();
      mic();
      breakbeam();
    }
  }

   void avoid()
  {
    IRE_OUT = LED1;
    read_IR();
    printf("right = %d\n\n",IRDT[11]);
    printf("middle = %d\n\n",IRDT[12]);
    printf("left = %d\n\n", IRDT[13]);
    motortk(RIGHT_MOTOR, forw);
    motortk(LEFT_MOTOR, forw);
    if(((IRDT[11] >IR_THRESHOLD) && (IRDT[11] < 200)) || ((IRDT[13] >
IR_THRESHOLD) && (IRDT[13] < 200)) && (IRDT[12] < IR_THRESHOLD))
    {    motortk(RIGHT_MOTOR, hrev);
      motortk(LEFT_MOTOR, hrev);
      wait(500);
      turn();
    /*if(IRDT[11] > IRDT[13])
    {
    printf("right = %d\n\n",IRDT[11]);
    printf("middle = %d\n\n",IRDT[12]);
    motortk(RIGHT_MOTOR, hrev);
    motortk(LEFT_MOTOR, hforw);
    wait (500);
    turn();
    }*/
    /*if(IRDT[13] > IRDT[11])
    {
    printf("left = %d\n\n",IRDT[13]);
    printf("middle = %d\n\n",IRDT[12]);
    motortk(RIGHT_MOTOR, hforw);
    motortk(LEFT_MOTOR, hrev);
    wait (500);
  turn();
  }*/
  }
  if((fb=front_bumper())>BUMPER_FUZZY_ZERO)
  {
    printf("fb was pressed\n");
    motortk(LEFT_MOTOR, hforw);
    motortk(RIGHT_MOTOR, hforw);
    wait(350);
    turn();
  }
  if(rear_bumper()>BUMPER_FUZZY_ZERO)
  {
    printf("rb was pressed\n");
    motortk(LEFT_MOTOR, hrev);
    motortk(RIGHT_MOTOR, hrev);
    wait(350);
    turn();
  }
  else
  {
    motortk(LEFT_MOTOR, forw);
    motortk(RIGHT_MOTOR, forw);
  }
  motortk(LEFT_MOTOR, forw);
  motortk(RIGHT_MOTOR, forw);

}


 void mic()
{
  read_IR();
/*if(IRDT[5] < mic_thresh)
{
printf("the value = %d\n\n",IRDT[5]);
 wait(300);
} */

  if(IRDT[5] > mic_thresh)
  {
  /*printf("release ball, value = %d\n\n", IRDT[5]);
  wait(200);*/
    IRE_OUT = LED2;
  /*motortk(RIGHT_MOTOR, 0);
  motortk(LEFT_MOTOR, 0);
  wait(1000);*/
    servo(0, open);
  /*   printf("door open\n");*/
    wait(250);
    servo(0, stop);
    wait(200);
    servo(0, close);
  /*   printf("door close\n");*/
    wait(800);
    servo(0, stop);
    wait(800);
    b = 1;
  }
}

 void breakbeam()
{
  read_CDS();
/*if (CDS[4] < 100)
{      printf("there is a ball, value = %d\n\n",CDS[4]);
 wait(300);
} */
  if (CDS[4] > 100)
  {    printf("there are no balls,value = %d\n\n", CDS[4]);
    wait (300);
    w = 1;
    findballs();
  }
}

 void turn()
{
  int a;
  unsigned rand;

  rand = TCNT;

  if (rand & 0x0001)
  {
    motortk(RIGHT_MOTOR, forw);
    motortk(LEFT_MOTOR, rev);
  }
  else
  {
    motortk(RIGHT_MOTOR, rev);
    motortk(LEFT_MOTOR, forw);
  }

  a=(rand % 1024) + 35;
  wait(a);

}

 void findballs()
{
  b = 0;
  while(w)
  {    motortk(RIGHT_MOTOR, hforw);
     motortk(LEFT_MOTOR, hforw);
     IRE_OUT = IRE_ALL_OFF;
     IRE_OUT = LED3;
     read_IR();

    if(IRDT[11] >90)
     {
    motortk(RIGHT_MOTOR, sforw);
    motortk(LEFT_MOTOR, srev);
    wait(700);
    motortk(RIGHT_MOTOR, hforw);
    motortk(LEFT_MOTOR, hforw);
     }
    if(IRDT[13] >90)
     {
    motortk(RIGHT_MOTOR, srev);
    motortk(LEFT_MOTOR, sforw);
    wait(700);
    motortk(RIGHT_MOTOR, hforw);
    motortk(LEFT_MOTOR, hforw);
     }
     while(IRDT[12] > 90 || IRDT[2] > 90)
     {
       read_IR();
       motortk(RIGHT_MOTOR, hforw);
       motortk(LEFT_MOTOR, hforw);

       printf("bent = %d\n\n", bent1);
       wait(300);
       printf("value = %d\n\n", IRDT[8]);
       wait(300);
       printf("value = %d\n\n", IRDT[9]);
       wait(300);
       if (IRDT[9] < bent1 || IRDT[8] < bent2)
```

```
          {
            printf("found it");
            servo(1, close1);
            wait(300);
            servo(1, stop);
            w = 0;
            z = 1;
            findhome();
          }
        }
      }
    }

    void findhome()
    {
      w = 0;
      while (z > 0)
      {
        IRE_OUT = IRE_ALL_OFF;
        IRE_OUT = LED4;
        read_CDS();
        CDS[0] = CDS[0]/1.5;
        CDS[1] = CDS[1]/2;
        CDS[3] = CDS[3]/2;


        if(CDS[0] < thresh)
        {
          if(CDS[3] > thresh)
          {
          /*printf("right sees\n");*/
            motortk(RIGHT_MOTOR, hforw);
            motortk(LEFT_MOTOR, hrev);
            wait(500);
          /*        printf("go straight\n");*/
            motortk(RIGHT_MOTOR, rev);
            motortk(LEFT_MOTOR, rev);
            wait(800);
          }
          if(CDS[1] > thresh)
          {
          /*        printf("middle sees, no left\n");*/
            go();
          }
        }
        if(CDS[3] < thresh)
        {

          if(CDS[0] > thresh)
          {
          /*        printf("left sees\n");*/
            motortk(RIGHT_MOTOR, hrev);
            motortk(LEFT_MOTOR, hforw);
            wait(500);
          /*printf("go straight\n");*/
            motortk(RIGHT_MOTOR, rev);
            motortk(LEFT_MOTOR, rev);
            wait(800);
          }
          if(CDS[1] > thresh)
          {
          /*        printf("middle sees, no right\n");*/
            go();
          }
        }
        if((CDS[0] < thresh) && (CDS[1] < thresh) && (CDS[3] < thresh))
        {
        /*          printf("doesn't see\n");*/
          motortk(RIGHT_MOTOR, hrev);
          motortk(LEFT_MOTOR, hrev);
          wait(800);
        /*printf("slow search\n");*/
          motortk(RIGHT_MOTOR, sforw);
          motortk(LEFT_MOTOR, srev);
          wait(1500);
        }
      }
    }
    void go()
    {
      IRE_OUT = IRE_ALL_ON;
      while(x)
      {
        read_CDS();
      /*   printf("going\n");*/
        motortk(RIGHT_MOTOR, hrev);
        motortk(LEFT_MOTOR, hrev);

        if(CDS[1] > 175);
        {IRE_OUT = LED5;
        /*                  printf("stopped\n");*/
```

```
        motortk(RIGHT_MOTOR, 0);
        motortk(LEFT_MOTOR, 0);
        wait(1000);
        IRE_OUT = LED6;
        motortk(RIGHT_MOTOR, hrev);
        motortk(LEFT_MOTOR, hforw);
        wait(4000);
        motortk(RIGHT_MOTOR, 0);
        motortk(LEFT_MOTOR, 0);
        servo(1, open1);
        wait(250);
        servo(1, stop);
        motortk(RIGHT_MOTOR, hrev);
        motortk(LEFT_MOTOR, hrev);
        wait(5000);
        motortk(RIGHT_MOTOR, hforw);
        motortk(LEFT_MOTOR, hrev);
        wait(2000);
        x = 0;
        z = 0;
        y = y+1;
        counter();
      }
    }
  }

 void counter()
{

  if (y == 3)
  {
    IRE_OUT = LED8;
    motortk(RIGHT_MOTOR, 0);
    motortk(LEFT_MOTOR, 0);
    while(1)
    {
    }
  }
  if(y < 3)
  {
    w = 1;
    IRE_OUT = LED7;
    x = 1;
    findballs();
  }
}
```



(Actual Size)

**$10.95**

**Torque:**
**4.8VDC:** 35.5oz-in. (2.56 kg-cm)
**6.0VDC:** 44.4oz-in. (3.2 kg-cm)

**Speed @ 60 Degrees:**
**4.8VDC:** 0.25 seconds
**6.0VDC:** 0.23 seconds

**Bearing Type:**
None (Case serves as bearing)

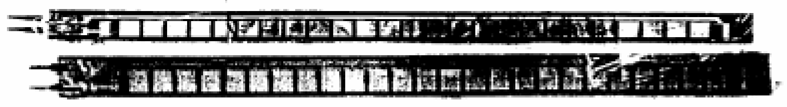
**Case Size:**
1.59"x 0.78"x 1.42" (40.4 x 19.8 x 36 mm)

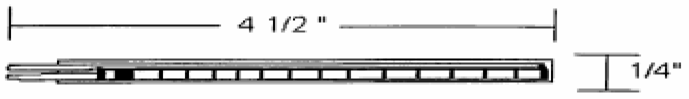
**Weight:** 1.30oz (36.8 g)

**Wire Length:** 12" (Including plug)

Jameco Part number 150551

# Flex Sensor





Nominal Resistance
Flex 0 Degrees: 10 K

Flex 90 Degrees
30-40 K

Proportional increase in resistance as sensor is bent or flexed. Maximum resistance 30K - 40K ohms.

Applications:
Virtual Reality Data Gloves
Robotic Sensor
Bio-Metric Sensor
Physics & Engineering Sensor

Nominal Resistance 10,000 ohms (10K)
Range 10K to 40K ohms
FLX-01 $ 10.00

Images Company
39 Seneca Loop
Staten Island NY 10314
(718) 698-8305 Voice
(718) 982-6145 Fax

# GP2D12/GP2D15 General Purpose Type Distance Measuring Sensors

## ■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : GP2D12
   Detecting distance : 10 to 80cm
   Distance judgement type : GP2D15
   Judgement distance : 24cm
   (Adjustable within the range of 10 to 80cm)
3. External control circuit is unnecessary
4. Low cost

## ■ Applications

1. TVs
2. Personal computers
3. Cars
4. Copiers

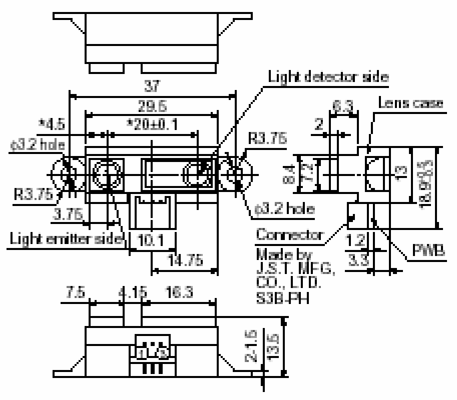
## ■ Absolute Maximum Ratings

(Ta=25°C, Vcc=5V)

| Parameter | Symbol | Rating | Unit |
|---|---|---|---|
| Supply voltage | $V_{CC}$ | −0.3 to +7 | V |
| Output terminal voltage | $V_O$ | −0.3 to $V_{CC}$+0.3 | V |
| Operating temperature | $T_{opr}$ | −10 to +60 | °C |
| Storage temperature | $T_{stg}$ | −40 to +70 | °C |

## ■ Outline Dimensions

(Unit : mm)



Terminal connection
① $V_O$
② GND
③ $V_{CC}$

※ The dimensions marked * are described the dimensions of lens center position.
※ Unspecified tolerance : ±0.3mm

## ■ Recommended Operating Conditions

| Parameter | Symbol | Rating | Unit |
|---|---|---|---|
| Operating supply voltage | $V_{CC}$ | 4.5 to +5.5 | V |

## ■ Electro-optical Characteristics

($T_a$=25°C, $V_{CC}$=5V)

| Parameter | | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Distance measuring range | | ΔL | *1 *3 | 10 | – | 80 | cm |
| Output terminal voltage | GP2D12 | $V_O$ | L=80cm *1 | 0.25 | 0.4 | 0.55 | V |
| | GP2D15 | $V_{OH}$ | Output voltage at High *1 | Vcc −0.3 | – | – | V |
| | | $V_{OL}$ | Output voltage at Low *1 | – | – | 0.6 | V |
| Difference of output voltage | GP2D12 | $\Delta V_O$ | Output change at L=80cm to 10cm *1 | 1.75 | 2.0 | 2.25 | V |
| Distance characteristics of output | GP2D15 | $V_O$ | *1 *2 *4 | 21 | 24 | 27 | cm |
| Average Dissipation current | | $I_{CC}$ | L=80cm *1 | – | 33 | 50 | mA |

Note) L : Distance to reflective object.
*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%).
*2 We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor.
*3 Distance measuring range of the optical sensor system.
*4 Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H.

Fig.1 Internal Block Diagram

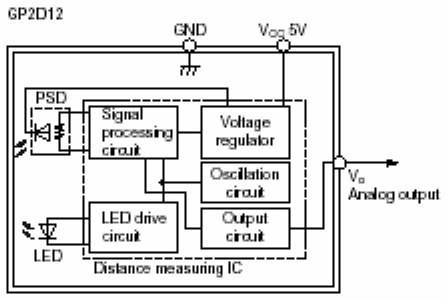


Fig.2 Internal Block Diagram

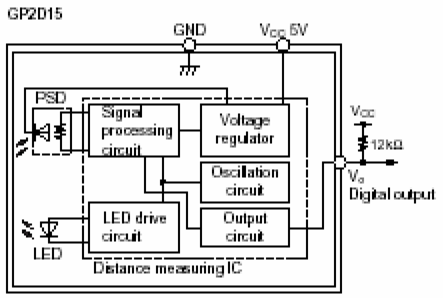
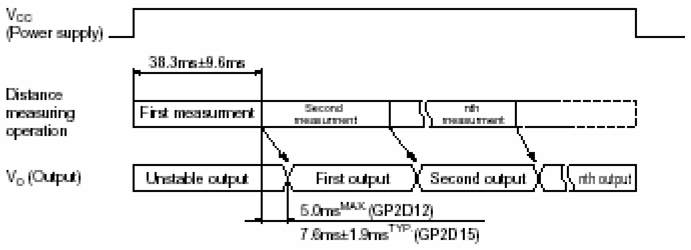
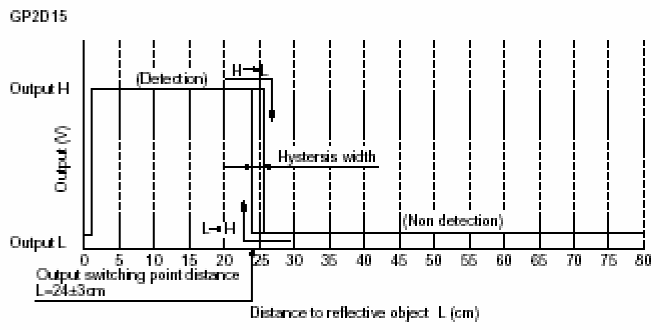


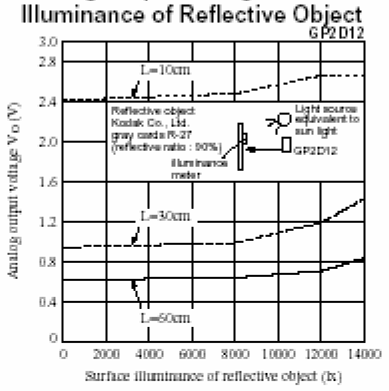
Fig.3 Timing Chart



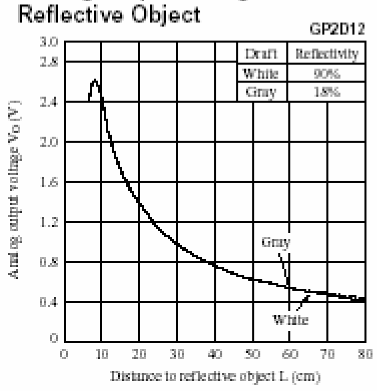
## Fig.4 Distance Characteristics



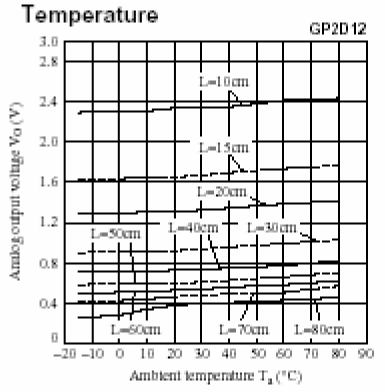
## Fig.5 Analog Output Voltage vs. Surface Illuminance of Reflective Object



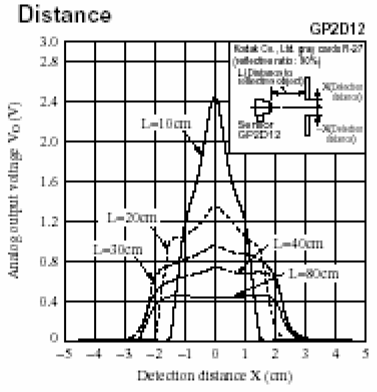
## Fig.6 Analog Output Voltage vs.Distance to Reflective Object



## Fig.7 Analog Output Voltage vs.Ambient Temperature



## Fig.8 Analog Output Voltage vs.Detection Distance


GP2D12
Kodak Co., Ltd. gray cards R-27 (reflective ratio : 90%)
L(Distance to reflective object)
X(Detection distance)
Sensor GP2D12
X(Detection distance)
L=10cm
L=20cm
L=30cm
L=40cm
L=80cm
Analog output voltage V_O (V)
Detection distance X (cm)