Gusano

University of Florida
EEL 5666
Intelligent Machine Design Lab

Student: Christian Yanes
Date: December 4, 2001
Professor: Dr. A. Arroyo

Table of Contents

Abstract

Gusano is an autonomous playing robot. Its main goal is to find a ball and toss in the air. It’s behaviors are controlled by Motorola 68HC11 micro-controller. Gusano has a total of three actuators; two of them are used to control the navigation of the robot and the other to operate an arm that picks the ball up.

Executive Summary

This robot was originally design to climb stairs but troubles with platform mechanisms avoided the implementation of it. The robot is designed to search for a playing ball. Once it finds the ball, it will pick it up and toss it in the air. While doing this action the robot is able to avoid any obstacles it encounters. Gusano is controlled by a Motorola 68HC11 micro-controller which controls three actuators and four IR emitter-detector pair. Three of the IR are used for obstacle avoidance situated in the front and both sides. The fourth IR is used to detect the playing ball and is situated three inches above the middle IR. The micro-controller board is used in conjunction with another board, the MRSX01 to form an extensive, sophisticated micro-controller data acquisition and control system.

Introduction

Gusano started as a stair climber robot. Its first platform design looked like a warm, therefore the name Gusano which means warm in Spanish. Design and mechanical problems prompted me to redesign the entire platform to a much simpler one. Dr. Arroyo and Dr. Shwartz at the Intelligent Machine Design Lab suggested that stair climbing was very difficult to achieved and gave me the idea of using the robot arm for some other purpose. The new platform would accommodate well to achieve the task of finding a ball and try picking it up, so that's what I did.

Mobile Platform

The platform is completely made out of wood. Its design consists of two front wheels and a round rear support that allows the robot to move in any direction when it is turning. The base of the platform is a six sided polygon, this design allows more mobility on corners when the robot is doing obstacle avoidance. At the top of the platform there is a big servo that drives an arm that resembles a dusting pan. The tips of the arms have self-adhesive material to ensure that the ball attach to it.

Actuation

Gusano uses three servos. Two of them are used to drive the wheel. These two servos have a strength of 100oz/in, and they were fully hacked. The other servo is used to drive the arm and is rated at 300oz/in. I controlled the servos using the output compare of the micro-controller. This allowed me to controlled the duty cycle of the pwm signal which regulate the speed of the motor.

Electronics

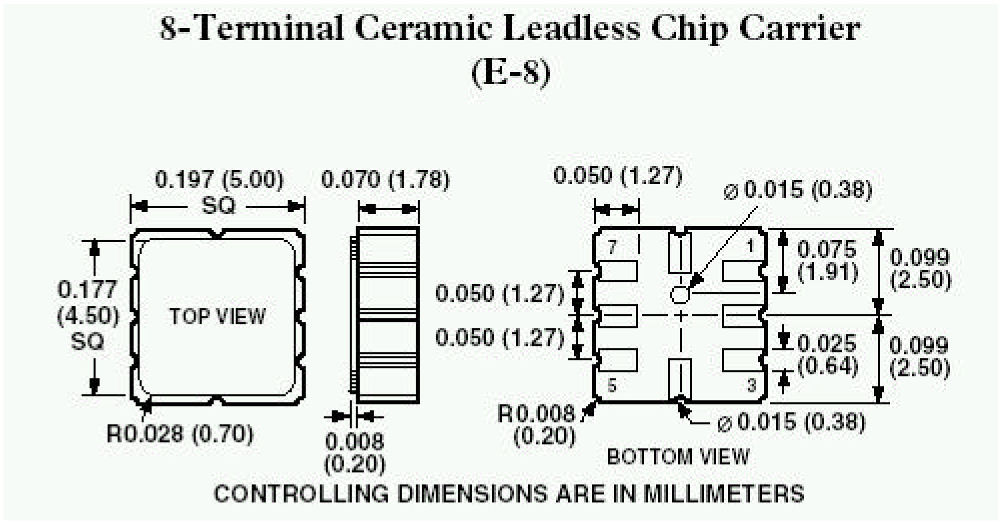
The MRC11 controller board paired with the MRSX01 expansion board were the electronics I used for my robot. The MRC11 was readily equipped with a Motorola 68HC11 micro-controller and 64 Kbytes of RAM/ROM which provided me with a very versatile package. The MRSX01 sensor expansion board was also very helpful because it provides an extensive control system. It has the all analog ports available in a bus, as well as the bus for the servos and motors

Sensors

Gusano has two different kind of sensors. It uses four IR to achieve obstacle avoidance and to find the playing ball. Three of them are in the base of the robot, in the two sides and in the front. The fourth one in the middle an on top to detect the ball.
The other sensor which is the special sensor is not included in the current behavior of Gusano because it was originally developed for stair climbing, but could be included to implement any additional behavior that requires tilt sensing.

The special sensor is the ADXL202E a dual-axis accelerometer with duty cycle output manufactured by Analog Devices. This device features a 2-axis acceleration sensor on a single IC Chip. This sensor can be modified to create a 2-axis tilt sensor with faster response than electrolytic, mercury or thermal sensors.

The ADXL202E is a low-cost, low-power, complete 2-axis accelerometer with a digital output, all on a single monolithic IC. It can measure both dynamic acceleration and static acceleration. The outputs are analog voltages or digital signals whose duty cycles are proportional to acceleration. The dimensions of the sensor are shown next, in inches and millimeters.

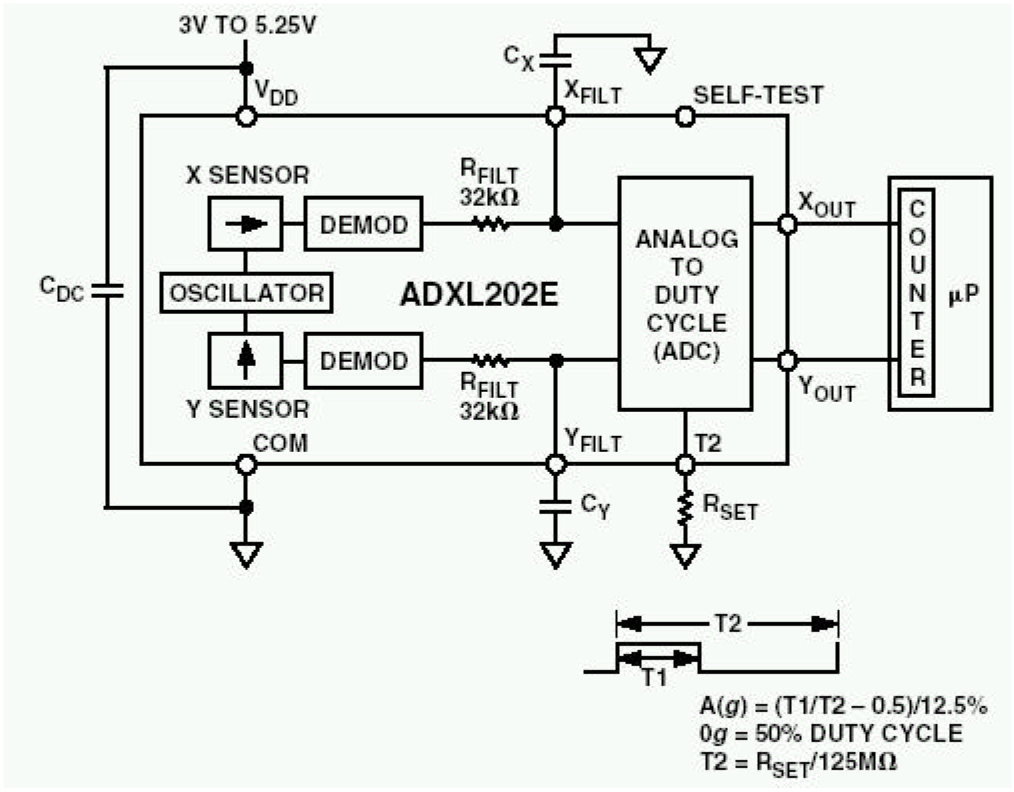


Theory of operation

This accelerometer contains a polysilicon surface-micromachined sensor and signal conditioning circuitry to implement an open loop acceleration measurement architecture. For each axis, an output circuit converts the analog signal to a duty cycle modulated (DCM) digital signal that can be decoded with the input capture system in the 68HC11.

The ADXL202E is capable of measuring both positive and negative accelerations to at least +/ – 2g. Since the accelerometer can measure static acceleration forces such as gravity I will use it as a tilt sensor.

The sensor structure is built on top of the silicon wafer. Polysilicon springs suspend the structure over the surface of the wafer and provide a resistance against acceleration forces. Deflection of the structure is measured using a differential capacitor that consists of independent fixed plates and central plates attached to the moving mass. The fixed plates are driven by 180 degrees out of phase square waves. An acceleration will deflect the beam and unbalanced the differential capacitor, resulting in an output square wave whose amplitude is proportional to acceleration. Phase sensitive demodulation techniques are then used to rectify the signal and determine the direction of the acceleration. A block diagram of the sensor is shown next.

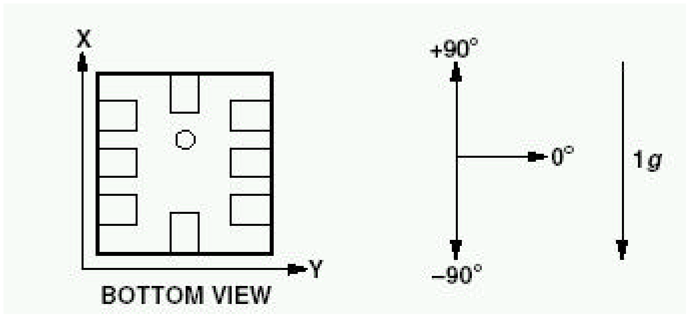


The output of the demodulator drives a duty cycle modulator (DCM) stage through a 32K ohm resistor. At this point a pin is available on each channel to allow the user to set the signal bandwidth of the device by adding the capacitor. This filtering improves measurements resolution and helps prevent aliasing.

After being low-pass filtered, the analog signal is converted to a duty cycle modulated signal by the DCM stage. A single resistor sets the period for a complete cycle (T2), which can be set between 0.5 ms and 10 ms. A 0g acceleration produces a nominally 50% duty cycle. The acceleration signal can be determined by measuring the length of the T1 and T2 pulses with the input capture system of the microprocessor.

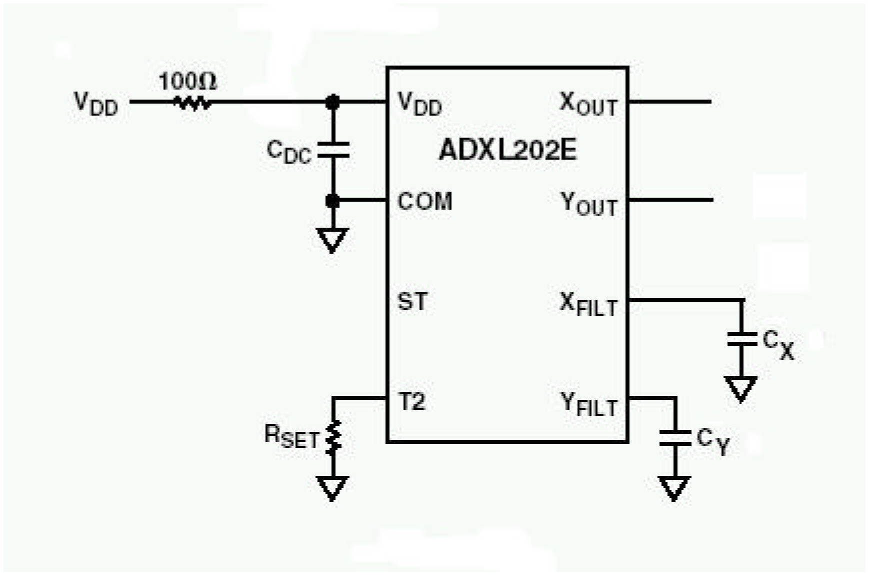
An accelerometer is most sensitive to tilt when its sensitive axis is perpendicular to the force of gravity, i.e., parallel to the earth's surface. At this orientation its sensitivity to changes in tilt is highest. When the accelerometer is oriented on axis to gravity, i.e., near its +1g or –1g reading, the change in output acceleration per degree of tilt theoretically negligible. When the accelerometer is perpendicular to gravity, it is theoretically expected that its output will change nearly 17.5 mg per degree of tilt, but at 45 degrees it should change only 12.2 mg per degree and resolution declines.

When the accelerometer is oriented so both its X and Y axes are parallel to the earth's surface one of the channels can be used for a roll and the other for pitch.



Once the output signals from the accelerometer has been converted to an acceleration that varies between –1g and +1g, the output tilt in degrees can be calculated using these equations: Pitch = Sin (Ax/1g) and Roll = Sin(Ay/1g).

In order to use the ADXL202E with a duty cycle output I had to select a duty cycle period and a filter capacitor. I used a 0.1 uF capacitor (Cdc), to adequately decouple the sensor from signal an noise on the power supply. The following figure shows how it was done.



A 100 ohms resistor was used as recommended to eliminate interference on the output from the power supply. Extreme care should be maintained during the connections made on this phase. A misunderstanding of the schematics made me blow one accelerometer and a new one had to be ordered. The ADXL202E's digital output is a duty cycle modulator. Acceleration is proportional to the ratio T1/T2. This sensor has provisions for bandlimiting the Xfilt and Yfilt. Capacitors must be added to these pins to implement low-pass filtering for antialiasing and noise reduction. The equation for the 3 dB bandwith is: F –3dB = 1 / (2 pi * 32 Kohm * C(x,y)) or, more simply
F –3dB = 5 uF / C(x,y).

The tolerance of the internal resistor (R filt), can vary typically as much as +/-15% of its nominal value of 32 Kohm; so the bandwidth will vary accordingly. A minimum capacitance of 1000pF for C(x,y) is required in all cases. I used two capacitors of 0.1uF each for Cx, and Cy, according to the above equation the bandwidth comes out to 50 Hz.

The period of the DCM output is set for both channels by a single resistor from Rset to ground. The equation for the period is **T2 = Rset (ohms) / 125 Mohms.** I used a 120 K ohm resistor that will result in a frequency of approximately 1KHz, or 1 ms

Behaviors

Gusano has to main behaviors search for a playing ball and obstacle avoidance. Once it is activated it will start to search for a ball. While it searches it avoids any obstacle it might encounter. When operating in an enclose space with a playing ball in the middle it will eventually find the ball. Once the ball is found it will reverse a small distance to lower the arm. Then it will move forward to attempt to pick up the ball, next it will move around in circles in the same place, showing happiness for finding the ball. Then it will toss the ball to the air and attempt to find it again.

Conclusion

The project Gusano a stair climbing robot was never achieved. However I feel that I have learned millions of new things while designing this robot. I have put in practice some of my knowledge in a very challenging way and that’s all that counts.

Appendix

Gusano code

```
/***************************************************************************
*
 * Title    gu_avnew.c
 * Programmer          Christian Yanes
 * Date        December, 2001
 * Version      1
 *
 * Description
 *   This program allows gusano to search for the ball an avoid
 *         obstacles
 *
 * note: motor and servo routines borrowed from
 * TALRIK's code.

***************************************************************************
/

/**************************** Includes **********************************/

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

/************************* End of includes *******************************/

/**************************** Constants *********************************/
#define STOP 0
#define RFORWARD 100
#define LFORWARD -100
#define RFWRDSLW 10
#define LFWRDSLW -10
#define RREVERSE -100
#define LREVERSE 100
#define RREVRSLW -20
#define LREVRSLW 20
#define RHFORW 50
#define LHFORW -50
#define RHREV  -50
#define LHREV  50
#define IR_THRESHOLD 60
```

```
/************************** End of constants ****************************/


void motor_tst(void);


void main(void)
/****************************** Main **********************************/
{
  unsigned int bl, cv, fb, rb, run_test;


  unsigned int IR_delta[NIRDT], IR_Threshold[NIRDT];
  unsigned int lpw, upw, servo_step;
  int i,j, rspeed, lspeed, delta_rspeed, delta_lspeed;

  init_analog();
  init_motortk();
  init_clocktk();
  init_serial();
  init_servos();


  /*Initialize servos variables*/
  lpw =1000;
  upw = 5000;
  servo_step = 100;



  /*Initial movement of GUSANO is forward*/

  lspeed = LFORWARD;
  rspeed = RFORWARD;

  motortk(RIGHT_MOTOR, rspeed);
  motortk(LEFT_MOTOR, lspeed);




  while(1)
  {
```

```
read_IR();
if(IRDT[7] > (IRDT[5] + 40) )
{

        /*reverse if see the ball*/
        lspeed = LREVRSLW;
        rspeed = RREVRSLW;
        motortk(RIGHT_MOTOR, rspeed);
        motortk(LEFT_MOTOR, lspeed);

  wait(1500);
  lspeed = STOP;
  rspeed = STOP;

  motortk(RIGHT_MOTOR, rspeed);
        motortk(LEFT_MOTOR, lspeed);



  /*move the arm up*/
  for( i=upw; i> lpw - 1; i -=servo_step)
  {
    servo(0,i);
          wait(100);
  }



   /*go forward a little bit*/
         lspeed = LFORWARD;
         rspeed = RFORWARD;
         motortk(RIGHT_MOTOR, rspeed);
         motortk(LEFT_MOTOR, lspeed);


   wait(1000);


   /*stop gusano*/
   lspeed = STOP;
   rspeed = STOP;
   motortk(RIGHT_MOTOR, rspeed);
         motortk(LEFT_MOTOR, lspeed);


  /*lift the arm half way*/
```

```
 for( i=lpw; i< upw / 2; i += servo_step)
 {
   servo(0,i);
   wait(100);
 }



 /*turn a cuple of times with the ball*/
 lspeed = LFORWARD;
 rspeed = RREVERSE;
 motortk(RIGHT_MOTOR, rspeed);
        motortk(LEFT_MOTOR, lspeed);

 wait(3000);

 /*throw the ball*/
 servo(0,5000);



 /*keep going fordward*/
        lspeed = LFORWARD;
        rspeed = RFORWARD;
        motortk(RIGHT_MOTOR, rspeed);
        motortk(LEFT_MOTOR, lspeed);




        }




read_IR();
if((IRDT[10] > IR_THRESHOLD ) || (IRDT[12] > IR_THRESHOLD ) ||
   (IRDT[7] > IR_THRESHOLD + 30) || (IRDT[5] > IR_THRESHOLD))

 {




   if(IRDT[10] > IRDT[12])
   /* Start turning when something in front and keep turning until nothing
```

```
        is in front */
        {

             lspeed = RHFORW;
             rspeed = LHREV;
            }
      else
            {

              lspeed = RHREV;
              rspeed = LHFORW;
            }


      motortk(RIGHT_MOTOR, rspeed);
      motortk(LEFT_MOTOR, lspeed);

      while((IRDT[10] > IR_THRESHOLD ) || (IRDT[12] > IR_THRESHOLD )
             || (IRDT[7] > IR_THRESHOLD + 30) || (IRDT[5] > IR_THRESHOLD))
      read_IR();

    }
  else
    {
      lspeed = -100;
      rspeed = 100;
    }

    motortk(RIGHT_MOTOR, rspeed);
    motortk(LEFT_MOTOR, lspeed);





    } /* end of while */



}
/**************************** End of Main ******************************/
```