What's the Vector, Victor?

By Brian Cohen & Mithun Jacob
November 21, 2007
MEAM 510 Mechatronics
Dr. Yim/Fiene

INTRODUCTION

Design Requirements/Brainstorming

The assignment designated that there would be two robots: one base robot and one droid robot. The base robot had fewer constraints in terms of dimensions and power supply. However, the base robot, as its name might indicate, needed to be stay at it designated spot throughout the demonstration. Because the droid robot was free to move around within the designated arc, there was a great deal more freedom in determining how exactly the droid would react to the base.

As such, our group came to several conclusions about the robot designs and functionalities:

- The base robot should probably perform the brunt of the computational work since it can be built large enough to hold the nanacore-12 which is a superior microcontroller compared to the TI.
- Likewise the droid robot will not be able to perform many of the functions needed to accurately measure the distance vector since the size is severely constrained.
- Since the droid robot will be powered off of batteries, it is likely that the supply voltage and current will be significantly smaller than that supplied to the base robot and thus is important when allocating receivers and detectors.
- There is no need to use differential drive to orient the robots; only one part of one of the robots needs to indicate direction.
- The distance measurement has a high margin of error, and thus non-optical methods need to be considered for the sake of novelty.

Our brainstorming sessions covered a lot of ideas to complete this task. Some included using laser triangulation, gas sensors, noise (amplifier and a microphone) detection and even a third robot which would be launched from the base robot to determine distance (using either a touch or Hall-effect sensor to indicate that it has reached the droid robot). According to our pre-lab report, we had decided on the latter idea, but soon realized that it required a lot more precision manufacturing and would not help for the final project. As a result, we resorted back to our LED emitter/receiver idea which may or may not be useful for the final project.

The idea was simply this: If an LED is pointed at a phototransistor, the distance may be inferred from the voltage level of the incoming signal (assuming that the nothing in the circuitry is changing). This would require testing a beacon and phototransistor to determine the relationship between distance and voltage (the results of which are shown later in this report). According to theory, the intensity an LED falls as $1/(distance)^2$. Furthermore, since the irradiance and current output from the phototransistor are linear (and current is obviously linear with voltage according to $V=IR$) then we would expect the voltage output to fall according to the relationship $1/(distance)^2$. Using the right gains on our operational amplifiers and proper signal conditioning, we could tune the base robot to detect the droid robot between 1 and 3 meters as desired.

Some reasons why this concept is superior is its inherent simplicity, the familiarity that we have with using the various components of the system, and the fact that this system would stay well within the original budget constraints of $20. In order to implement any of the other ideas, we would need to wait for parts to come in and would not be able to begin building the circuits until late.

System Overview

Once we had decided on using LED's and phototransistors, we followed through with narrowing down to a suitable design. Our finalized concept was the following:

The droid robot would act first by turning in back and forth in a 180 degree arc until it faced the base robot and found any of the six orientation LED's. Then, after a pre-programmed time, the base robot would sweep a turret back and forth along a 90 degree arc (covering the entire field where the droid robot could be located) several times. Each sweep would return a maximum voltage which would be stored and later averaged. This final average would be used to determine distance from a derived relationship. In summary, the task was as follows:

PHASE I: Droid robot turns until it finds the base robot and then stops, pointing a LED beacon towards the base robot.

PHASE II: Base robot sweeps several times, finding a maximum voltage per sweep that pertains to the signal voltage level from the droid robot's beacon.

PHASE III: Calculate the distance using a pre-determined relationship between voltage level and distance. Output this distance to the computer from the serial port.

The remainder of the report describes in detail the design and programming of the robots as well as some problems that were encountered and general conclusions.

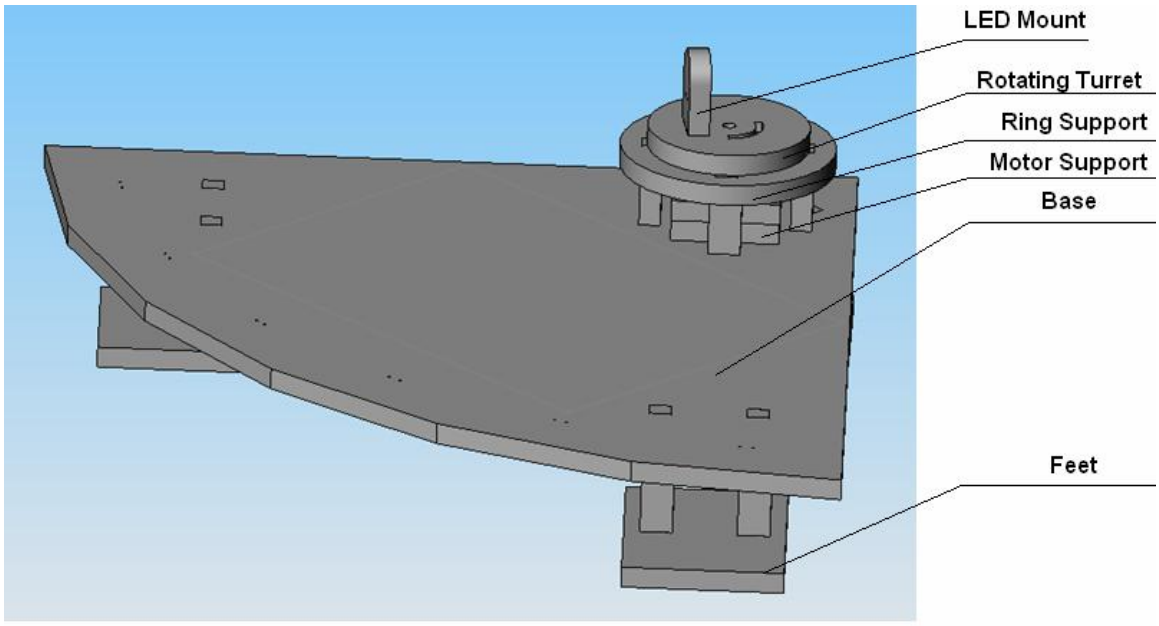
MECHANICAL DESIGN

One useful feature of our mechanical design which we are most proud of is that the assembly required almost no glue. The only parts that did require glue was between each of the motor shafts and rotating disks, basically to make sure the disk rotated with the shaft, and the feet of the droid robot to attach them to the base. Otherwise, every connected part was press-fitted. Each of the final assemblies is robust and provided solid frames on which to mount our electronic components

List of Components

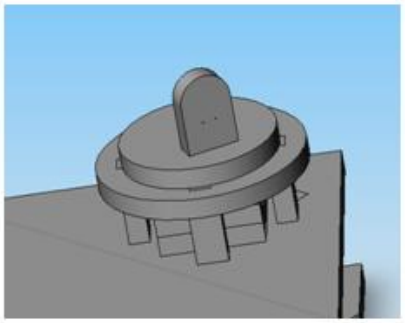
- MDF
- Hot Glue

Base Robot



As mentioned above, we had realized that it made little sense to use differential drive and thus we designed a turret on the base robot that would spin in place. This turret required only one motor at the base, turning a circular disk holding a phototransistor. The disk was supported by a ring support construction ensuring that the disk rotated in a plane and

henceforth does not wobble. Because we used MDF for the entire design, there was little friction between this ring support and rotating disk. Later we realized that the wires leading to the phototransistor were causing too much resistance for the motor to turn the disk slowly (as required to sweep for a high signal). Therefore, we used a higher voltage on the motor and simply weighed down the disk with a solid steel weight. Increasing the rotor inertia helped to generate more controlled motion from the motor.

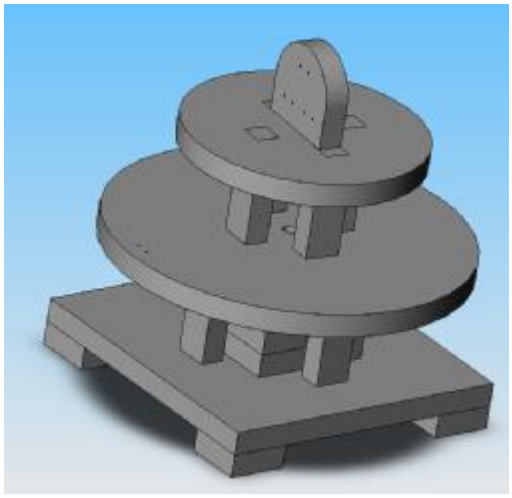


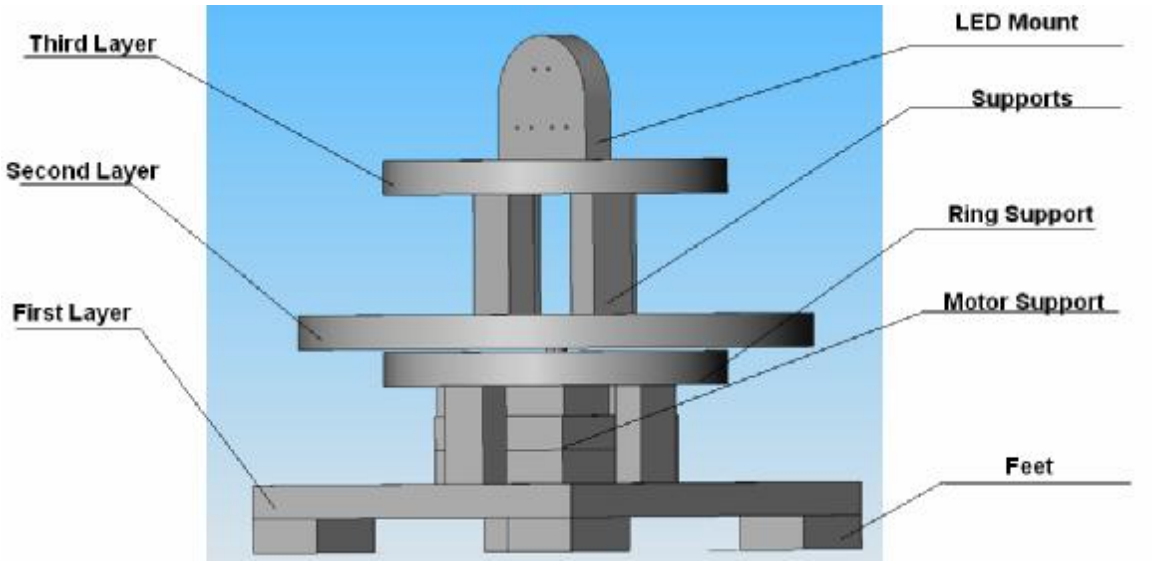
Base Robot Turret

Since the field was a 90 degree arc, it made little sense for the base robot to be circular and surrounded by LED's. Instead, we laser cut a base that was a 90 degree arc with six LED's directed outward at 15 degree increments. This division was determined from the LED data sheet which indicated a significant drop-off in emission at angles wider than 10 degrees. The LED's were placed underneath the base at exactly the same height from the ground as the orientation phototransistor on the droid robot. This was done using appropriately sized feet to hold the base off the ground at a specified height. Three feet held the robot at each corner

Finally, we originally planned to have the microcontroller located across the top of the base but underneath the turret. Since the height of the turret was constrained to that on the droid robot, we were not able to keep the path between the phototransistor and the droid clear of wires, etc. Thus we drilled holes into the base and mounted the nanocore-12 underneath the robot. This made the design not only look more elegant, but allowed it to function as desired with no obstructions.

Droid Robot





The droid robot had to be designed within a very small size requirement (7.5 cm x 7.5 cm x 7.5 cm). This constraint was undoubtedly the toughest to work with. We essentially designed a stationary turret that could spin. In order to keep things simple we designed the droid robot much like the turret on the base robot, except bigger. We used a ring support to hold up a disk that was turned by a single motor.

Because the phototransistor and LED's needed to not only rotate but point in the same direction, we placed each on a separate disk (one higher than the other). It was critical to keep our LED's and phototransistor as far apart as possible to prevent noise. Thus the phototransistor was located on the lower disk and the three LED's were located on the upper disk, held up on a small semi-circular mount.

Space was later made in between the top and middle layers by removing two supports to accommodate alternate parts for the modified designs. This will be discussed later in the report.

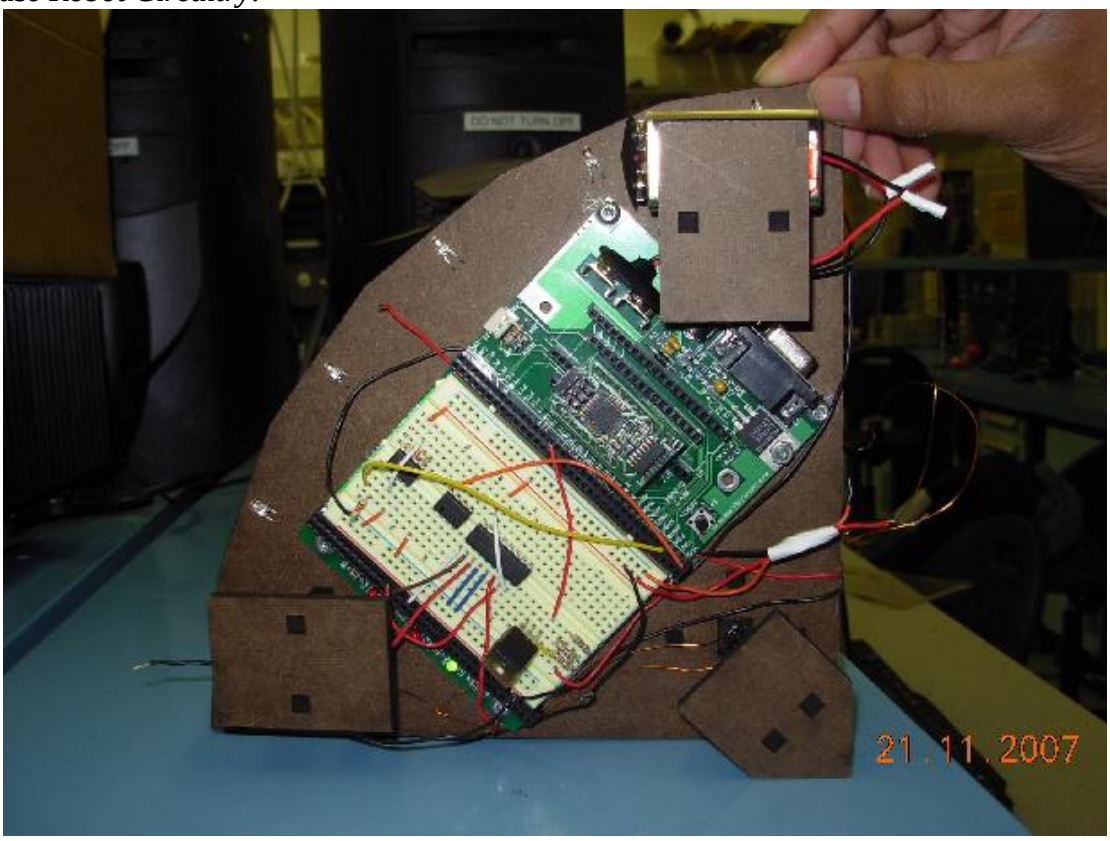
ELECTRICAL CIRCUITS DESIGN

Electrical Components

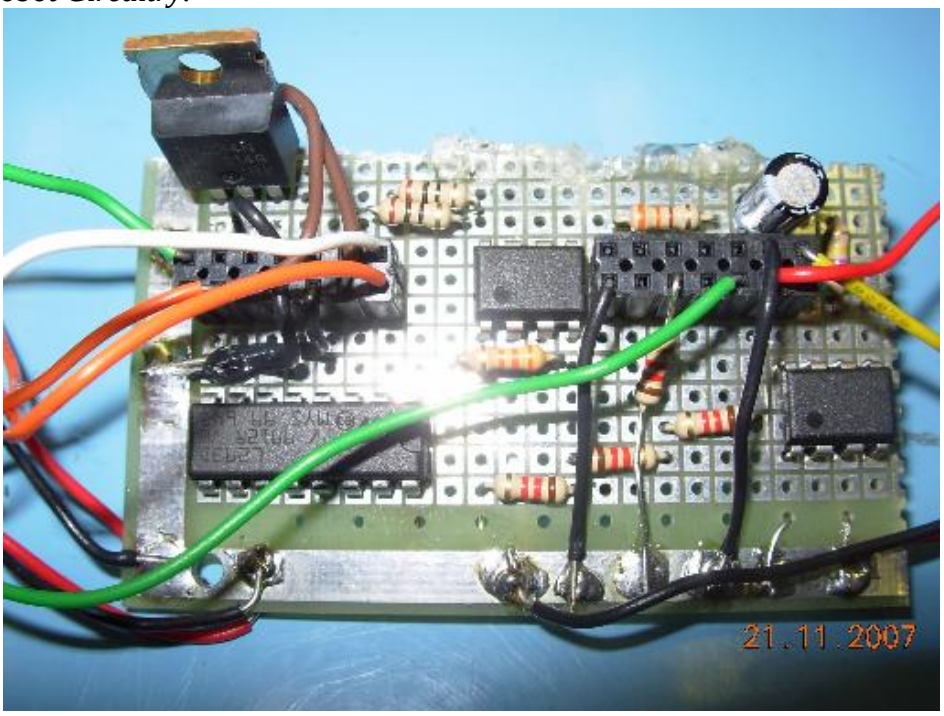
- 2 L293D Motor Drivers
- 2 IRLZ34N N-Channel MOSFET's
- 2 LTR 4206-E Phototransistors
- 9 LTE 4206 LED's
- 2 TL7726 Hex Clamp
- 2 TLV272 Low Voltage Operational Amplifiers
- Assorted wiring, resistors and 2 capacitors

Electrical Design

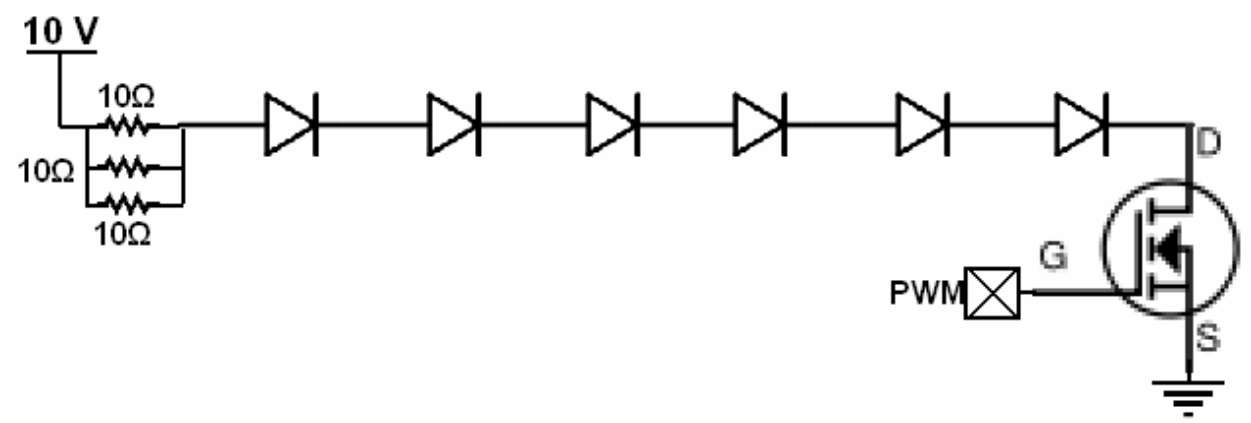
Base Robot Circuitry:



Droid Robot Circuitry:

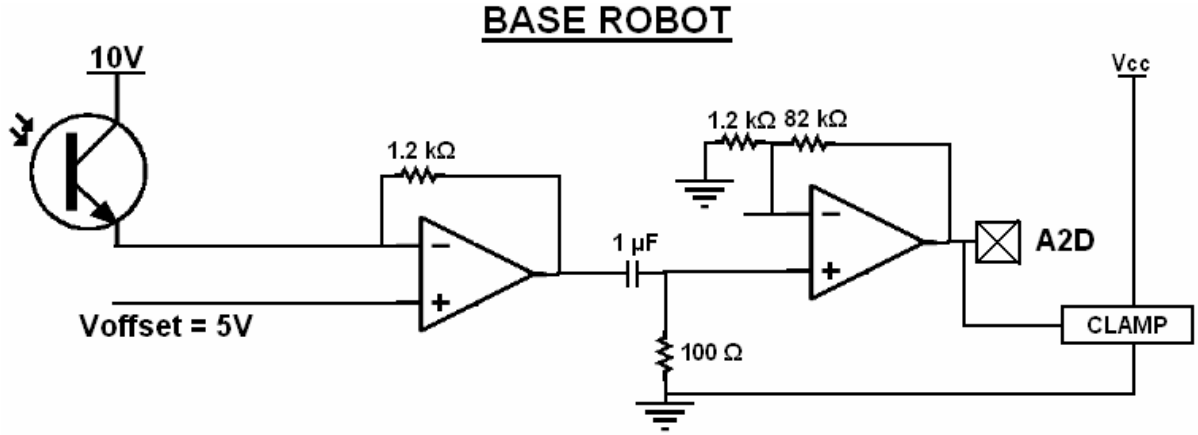


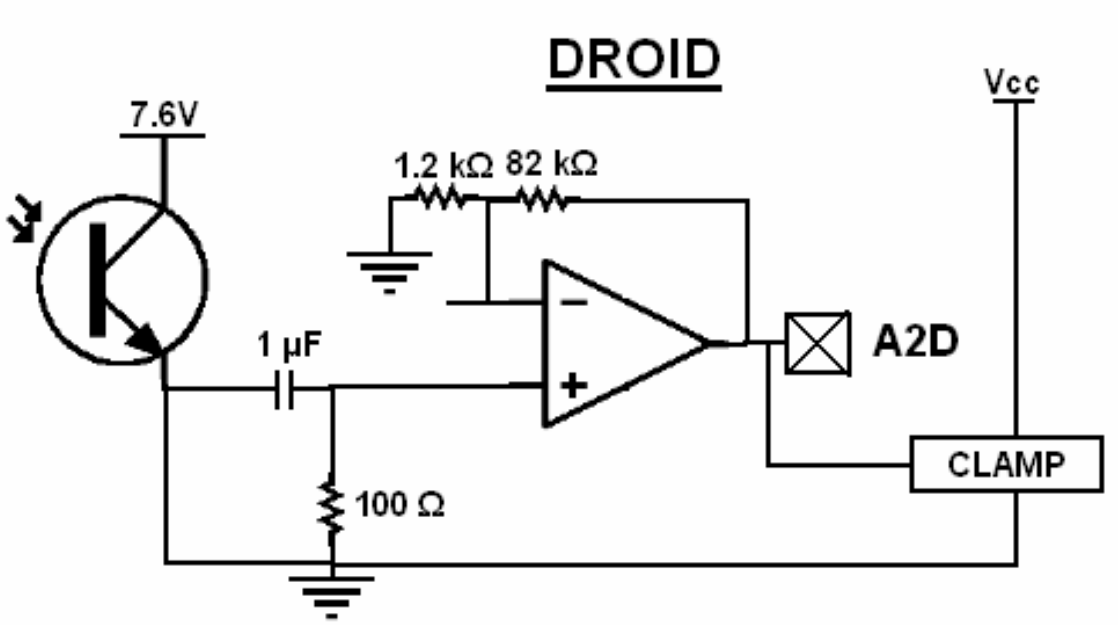
Please refer to the circuit diagram for the LED's below:



The 3 LED's on the droid robot were pulsed using the TI microcontroller. Essentially, when the PWM signal was at a logic level high, the MOSFET would turn on and allow a pulse to travel through the LED's and resistors. When the signal was at a logic level low, the MOSFET was turned off and no current flowed through the LED's. Since the LED's were directly connected to the batteries, we has a much higher voltage and current source than if we used the PWM signal itself from the TI. Additionally, this configuration did not require operational amplifiers or Darlington's to operate. The exact same configuration was used on the base robot with the exceptions that only two 10Ω resistors in parallel, six LED's were used in series, and the power supplied was 7.2 V.

Please refer to the circuit diagram for the phototransistor circuit on both the base and droid robot below:



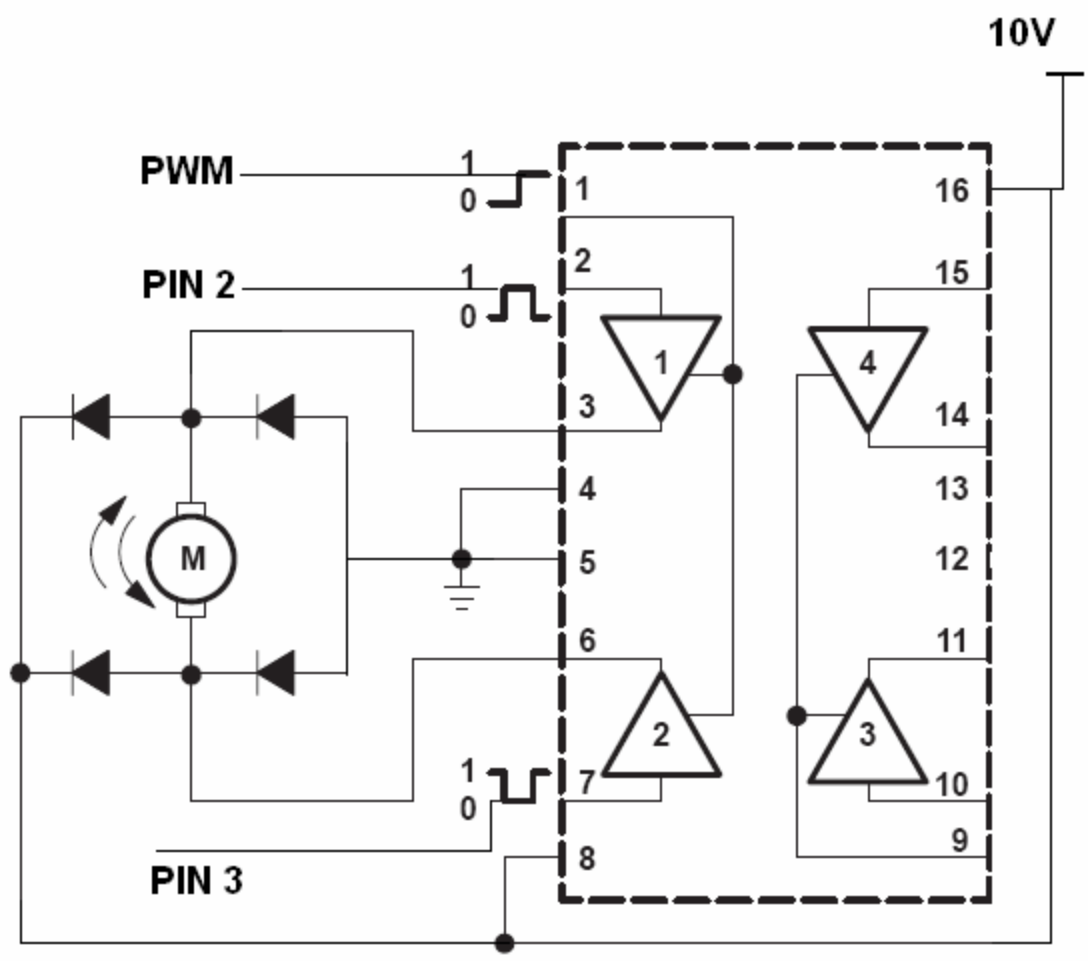


The phototransistor on the droid was necessary to determine orientation. When the phototransistor found a specified frequency of 214 kHz, the droid robot was oriented towards the base robot and then stopped spinning. However, the phototransistor on the base robot was used to determine distance between itself and the droid. It measured the voltage from the droid's beacon and manipulated the value to output distance. Because the distance between the droid and the base robot could vary up to 3 meters it was critical to use a photo-transitive configuration. This guaranteed that the current output from the phototransistor is constant for any given irradiance and hence that there would be no discrepancy in our distance measurements.

It was also critical to use operational amplifiers to amplify the signal to the analog to digital converter pin and input/output pin for both robots. A clamp was used in both cases to protect the pins on both the TI and nanocore-12 microcontrollers. The circuits also used high pass RC filters to filter any noise out from the ambient.

The resistor values were basically determined through trial and error. The gains on the op amps were increased and decreased depending on the signal as displayed by the oscilloscopes. The resistor going from the phototransistor to ground on the droid robot was likewise determined by trial and error. It was important however, that this phototransistor be sensitive (hence use a higher resistor value) in order to use the frequency program on the TI microcontroller.

Please refer below to the circuit diagram for the motors below:



The motor circuit was the same for both robots with the exception that the base robot supplied 10 V and the droid supplied approximately 7.2 V. The L293D motor driver was a great device for driving the motors because it acted as an "all-in-one" device. Not only did it have a built in H-bridge to change directions, but it also increased the output current up to approximately 1 Amp. The bi-directional functionality was necessary for the droid robot in case it accidentally did not find the base robot in its initial 360 degree sweep. In the case of the base robot, this function was not necessary although it was helpful to include it in case we altered the distance measuring procedure to include multiple sweeps (which we did at one point by making the base sweep 10 or more times to take an average of the maximum voltage readings). What was required from the microcontrollers was a PWM signal and two high/low signals to change directionality. Otherwise, the L293D was a great device for simplifying our electronic circuitry.

Electronics Calculations

In order to find the resistor and capacitor values for the high pass RC filters, we used the following equation:

$$f_c = \frac{1}{2pRC}$$

Essentially, any frequencies below f_c would be filtered out. In the case of the orientation phototransistor on the droid robot, we wanted any signals below 30,000 Hz to be filtered out. Thus the calculation went as follows:

$$50{,}000 = \frac{1}{2pRC}$$
$$RC = 3.183x10^{-6}$$

Since 1 µF capacitors were in abundance, we decided to use a 10Ω resistor which guaranteed us a cutoff frequency of approximately 16 kHz. Because the lowest frequency for the LED's that we used was 30 kHz. This also gave us a nice safety factor of about two.

Finally we also needed to calculate the necessary resistor values for each LED circuit. The LED's had a typical voltage drop of 1.2 V and the voltage drop across the drain and source on the MOSFET was minimal (resistance between the drain and the source was less than 0.1Ω when it is turned on). Therefore, the governing equation was:

$$\frac{V_{Supply} - n(1.2)}{I} = R$$

Where n is the number of diodes in the circuit. Since the LED's can take a maximum of 1 Amp when being pulsed, we tried to attain around 800 mA in order to maximize the intensity and project pulsed signals between the robots. Using this value of I, we calculated a resistance of 3.5Ω for the base robot (using 6 LED's and a supply of 10 Volts) and 4.5Ω for the droid robot (using 3 LED's and a supply of 7.2 Volts). Because the lowest resistor available was 10Ω, we simply used three 10Ω resistors in parallel for the base robot (3.33Ω total) and two 10Ω resistors in parallel for the droid robot (5Ω total).

All other specifications were met for the motor and phototransistors circuits. The clamps for the microcontrollers were set at the reference voltages (5 V for the nanocore-12 and 3.6 V for the TI using a voltage divider).

PROGRAMMING

Phase 1: Orientation

The Base robot has circular array of 6 LEDs used to orient the Droid robot in the first phase of operation. The Droid robot uses the phototransistor output to detect the frequency of the received light and once it matches that of the Base Beacon, it stops and overshoots by a fixed amount so that it is facing the base robot. The orientation will be

perfected during the distance measurement phase (Phase 2). The Base robot will keep the PWM output to the Base Beacon going for a predetermined time period. We assume that the droid robot will be able to find the base robot within this time period of *DroidTimeout.*

Phase 2: Distance Measurement

The Base robot has a rotating phototransistor and using PWM output, it moves in short, powerful bursts essentially stepping the phototransistor around the 90^{o} arc. The phototransistor output is read *only* when the phototransistor is stationary (between bursts). The voltage is read is verified by checking if the values preceding it exhibit a constant slope (be it rising or falling). If the verified voltage is large than the current maximum, it is stored. Once the sweep about the arc is completed, the maximum voltage is used to find the distance by comparing it to an inverse
square curve which has been approximated in the program as a 3 linear equations.

Event	Base	Droid
Phase 1	• Initialize Timer, PWM and operational variables • Start PWM channel 1 for the Beacon LEDs • Delay(DroidTimeout)	• Switch off the Watchdog • Get the Frequency off the Phototransistor output. • Step the motor using PWM • If 30 kHz, exit the GetFrequency function.
Phase 2	• Switch on PWM output for Channel 2 to the motors • Set PWMPER0 to 0x53 and initialize it so it moves in an anticlockwise direction. • Continuously read the voltage off the phototransistor for 3000 ms. First the maximum voltage every 10 ms is determined (since the frequency of the Droid Beacon is 100 kHz); this is done 60 times and the average of these voltages is set as the maximum voltage in the 600 ms cycle. The maximum voltage of every 600 ms cycle in a 3000 ms cycle is used to determine the voltage of every sweep. • Then the duty cycle is set to PWME0 = 20 and the motor is turned on for a brief period of 20 ms after which it is turned off and the cycle repeats (starting from Voltage measurement) after a 2000ms delay.	• The PWM output to the LEDs is turned on, pulsing them at 100 kHz.

Code:

Base Robot:

```
#include <hidef.h>          /* common defines and macros */
#include <mc9s12c32.h>      /* derivative information */

#pragma LINK_INFO DERIVATIVE "mc9s12c32"

void Delay(int Time) {

unsigned int delta_time,start_time;


start_time=TCNT;            //Stores starting value
delta_time=Time;        //Stores time left to run function

do{
 if (TCNT < start_time){      //If TCNT has overflowed
   if(65535-start_time+TCNT > 187){//Check if 1 ms has passed
     --delta_time;                 //Decrement delta_time
     start_time=TCNT;               //Set the new Start Time
     }
 }

 else if (TCNT - start_time > 187 ) {//If 1 ms has passed
   --delta_time;
   start_time=TCNT;
 }

}while(delta_time);

}

int GetVoltage(int Avg, int SampleTime, int Per) {

unsigned int atd, atdSum, atdSumMax;
int Final;
 int i,atdMax,Sample;
 unsigned int delta_time,start_time;

SampleTime=SampleTime/(Avg*Per);    //Number of times the voltage is
sampled


ATDCTL2_ADPU = 1;     // power up the a/d converter
ATDCTL5 =  0;       //start conversion
ATDCTL4_SRES8 = 0; // Set to 10 bit resolution

//The timer works on the same principle as Delay(T).
start_time=TCNT;

 for(Sample=0;Sample<SampleTime;++Sample){//Ensure it only Samples for
2000ms
 atdSum=0;
```

```
for(i=0;i<Avg;++i){     //Averages every Avg times

 atdMax=0;
 atdSumMax=0;
 delta_time=Per;

//The following timer code ensures that the Voltage is sampled for only
//Per milliseconds.

do{
if (TCNT < start_time){
if(65535-start_time+TCNT > 187){
--delta_time;
start_time=TCNT;
}
}
else if (TCNT - start_time > 187 ) {
--delta_time;
start_time=TCNT;
}

while(ATDSTAT0_SCF == 0) ; /* wait till conversion is done */

 atd =  (ATDDR0>>6);    //Read the Voltage
 if(atd > atdMax) atdMax=atd;
}while(delta_time);

   atdSum+=atdMax/60;
        }

 if(atdSum > atdSumMax) atdSumMax = atdSum;

 myprintstring("\n\rVoltage: ");
 myprint16bit(atdSum);
 }

return atdSumMax;

}

int CalculateDistance( unsigned int atdMax)
//Using linear equations described below
{
      if (atdMax >= 288)
      {
            return -381*atdMax + 370;
      }

      if (atdMax > 288 & atdMax =< 1000)
      {
            return -140*atdMax + 300;
      }

      if (atdMax > 1000)
      {
            return -26*atdMax + 186;
      }
```

```
}

void main(void) {

 int INP1,INP2,Sweep,i;
 int Per=10,DroidTimeout=3000,MaxSweeps=20,KickStart=2000;
 int MotorTurnaroundTime2=1000,MotorTurnaroundTime1=1000;
 unsigned int atd, atdMax;
 int Ticks;
 for(i=0;i<7;++i) History[i]=0;

 DDRT = 0x1F;

 MODRR = 3;
//map PWM0 to PT0 and PWM1 to PT1 since MOTOR's OFF in the beginning

 PWMPOL_PPOL0=1; //swap polarity
 PWME_PWME0 = 0  ; //Disable M PWM


 PWMPOL_PPOL1=1; /* swap polarity */
 PWMPER1 = 0x70; //Sets LED pulsing period and Duty Cycle
 PWMDTY1 = 56;
 PWME_PWME1 = 1  ;      //ENABLE LED PWM

 TSCR2_PR=7; //Sets the prescaler to the max
 TSCR1_TEN=1; //Starts timer


 Delay(DroidTimeout);//Allows LEDs to pulse for DroidTimeout ms

 PWME_PWME1 = 0;  //Switches off the LED PWM output


PWMPER0 = 0x53;   //Sets Period for Motor PWM

PTT_PTT2=0;       //Sets the input pins for the Motor Driver
PTT_PTT3=1;

atdMax=0;

for(Ticks=0;Ticks<40;++Ticks){ //Described above in report

 PWME_PWME0 = 0;
 Delay(2000);
 atd=GetVoltage(60,2000,10);
 if(atd > atdMax) atdMax=atd;
 PWMDTY0 = 20;
 PWME_PWME0 = 1;
 Delay(20);
 PWMDTY0 = 0;
 PWME_PWME0 = 0;
}

 myprintstring("\n\rVoltage: ");
 myprintnum(CalculateDistance(atdMax));
```

```
PWME_PWME0 = 0;

}
```

Droid Robot:

```
#include "io430.h"
#include <stdio.h>

void Delay(int Time)          //Timer-independent Delay Function
{
 float i,Lim=Time*3.07692;
 for(i=0;i<Lim;++i){}
}

int GetFrequency()
{
unsigned short f,end,start;
unsigned long Conv = 252944;

f=end=start=0;
BCSCTL1 = CALBC1_1MHZ; //Setting DCO Frequency at 1 MHZ
DCOCTL = CALDCO_1MHZ;
P1SEL |= P1SEL_1; // set pin 3 to TA1 mode
P1DIR &= ~BIT1; // set pin 3 to input

TACTL |= TASSEL_2 | ID_2 | MC_2;//SMCLK input, prescale / 4,continous
TACCTL0 |= CAP | CM_1 | CCIS_0;//capture rising edge on P1.1*/
while (!(TACCTL0 & CCIFG)); //wait till it overflows

start = TACCR0; //capture time ticks
TACCTL0 &= ~CCIFG; //resetting the flag

while (!(TACCTL0 & CCIFG)); //same as above
end = TACCR0;
f = Conv/(end - start);
if(f > 25000 && f < 30000) return 1;
TACCTL0 &= ~CCIFG; //resetting the flag

return 0;
}



int main( void )
{
  int Ticks;

  WDTCTL = WDTPW + WDTHOLD;

  P1DIR |= BIT2  | BIT3 | BIT4;     //Sets PWM for motor control
  TACCR0 = 0x80;
  TACCR1 = 0x48;
  TACCTL1 = OUTMOD_7;
  TACTL = TASSEL_2 | MC_1;
```

```
  P1OUT |= BIT3;              //Sets Motor Input Pins
  P1OUT &= ~BIT4;


  for(Ticks=0;Ticks<5;++Ticks){  //Steps the Motor
      Delay(1000);
     if (GetFrequency()) break;
      P1SEL |= BIT2;
      Delay(20);
      P1SEL &= ~BIT2;
      P1OUT &= ~BIT2;
  }

  TACTL |= MC_0;              //Sets PWM for LEDs

  P1SEL |= BIT6;
  P1DIR |= BIT6 | BIT3 | BIT4;
  TACCR0 = 0x80;
  TACCR1 = 0x70;
  TACTL = TASSEL_2 | MC_1;

  for(;;){
  }
}
```

Extension to myprint.c:

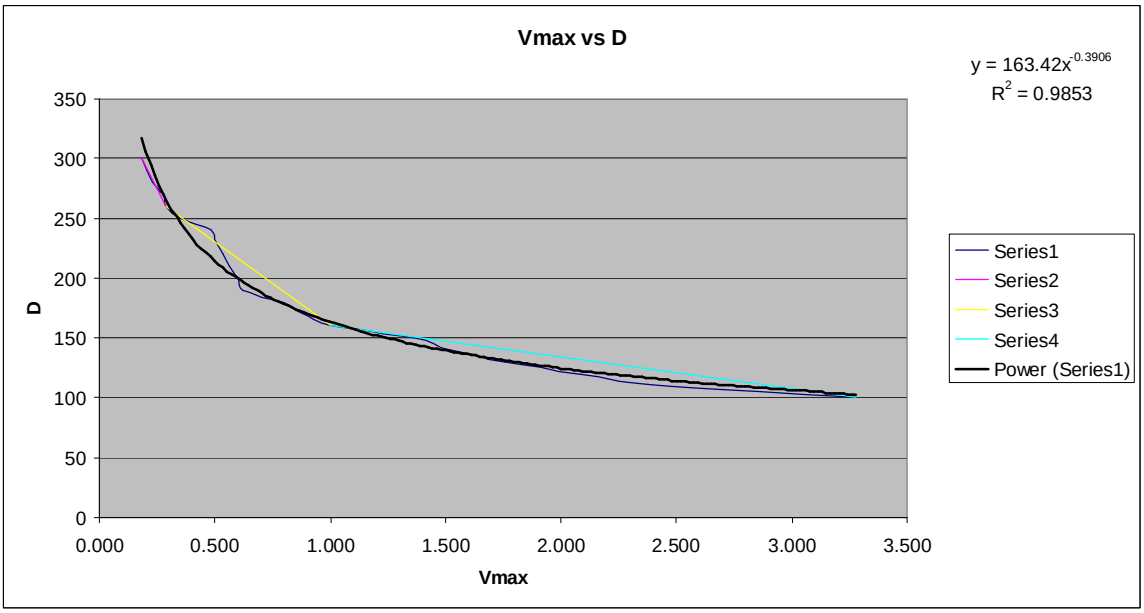
```
void myprint16bit(unsigned int num) {
 /* writes a number to the terminal */
 char numstrptr[6]="     ";
 int i = 4;


 /* start backwards from the 0 termination */

 do {
  numstrptr[i--] = '0'+ (num%10);
/* put in ascii value of least sig digit*/
 } while ((num = num/10)>0);
/* truncate off least significant digit, while digits exist */
 myprintstring(numstrptr);
}
```

Conversion from Vmax to Distance:

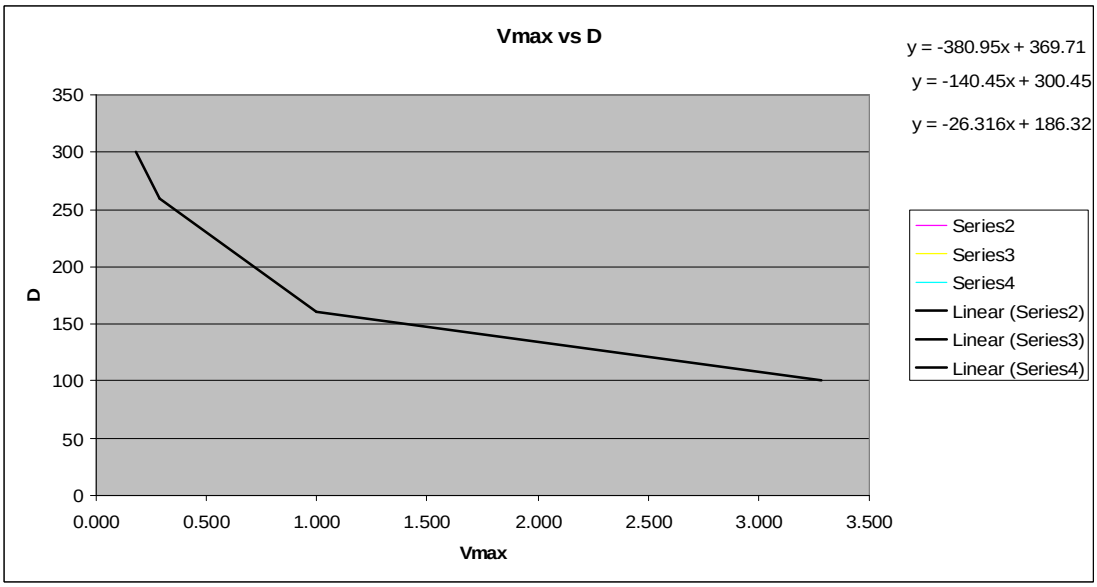
With the test values we obtained, we described the relation between Vmax and Distance with 3 linear equations (Series 2, Series 3 and Series 4).



The linear equations used are:

y = -380.95x + 369.71
y = -140.45x + 300.45
y = -26.316x + 186.32

These equations were approximated and used in the CalculateDistance(unsigned int) function.



CONCLUSIONS

Although we did not have a working system by the time of the demonstration, the design was elegant and many of subsystems worked well. The major issue that kept the original system from working was the TI microcontroller which did not operate as we expected. The PWM signal could not be generated while the TI was supplied with the requisite power to operate. This prevented our droid robot from emitting a pulsed infrared signal for the LEDs. Subsequently, we downgraded the droid several times in order to make up for the lack of a microcontroller. This required corresponding changes on the base robot. It is clear that these changes avalanched into a whole mess of problems. For example, our newly designed phototransistor circuit on the base robot was not functioning correctly (although we did get a nice sine wave curve).

Our design worked well in several areas. Our DC motor was effectively converted into a stepper motor by sending bursts of PWM signal into the motor driver for short periods of time. Our rate of turn and step size was fairly slow yet repeatable. Also, the RC filter on the base robot worked really very well and our signals were configured to indicate distance measurements perfectly. Not only did the motor and LED circuits on the droid robot worked as planned, but they were manufactured on a very small workspace. Finally, the programming was quite modular which allowed for changes easily. It was efficient in terms of memory usage and worked exactly as expected.

This report was based on our initial design which will be tested next week.

Brian's Reflections

I was disappointed that the project did not go according to plan. We started out early and completed several milestone tasks, including design and building the robots and testing the distance voltage measurements with a phototransistor and 3 LED's, way ahead of schedule with plenty of leeway time. However, because of the TI microcontroller and several other inexplicable errors that cascaded afterwards, we could not finish on time. Please refer to the time log to understand why I am this frustrated.

I though the overall project was a good way to familiarize ourselves with the many electronic devices we had covered over the past few weeks. I also appreciated the broad requirements for the task, allowing us to brainstorm different ideas, methods and sensors/receivers. However, some problems that I noted was the low budget constraint of $20 initially stated in the lab handout. Because the budget was not changed until later, we ruled out many alternatives simply because they were too expensive (ultrasonic rangefinder for example). Furthermore, I don't think we covered enough material on alternate sensors in class or more importantly in previous labs. It seemed that the only easy and cheap way to complete this task was with LED's and phototransistors.

Mithun's Reflections

The Lab was quite challenging since we had to measure the distance as well as orient two robots together. Our design methodology ensured that the design for the droid could be duplicated on the base robot thus reducing the overall design time and effort. The mechanical design of the robots was quite interesting and we felt that bypassing the obvious differentially driven robots and only moving the important components (phototransistors) greatly increased precision. Our circuits may have been complex, but it ensured excellent control over the motors (as seen by the stepping action) and the reception of a relatively noise-free signal for data analysis.

We believe that our Lab could have been highly successful if the TI was able to run independently. Once we discovered and exhausted all possible options of running it independently, we downgraded our Droid to make it run without the TI and this led to a chain of events which basically demolished any chance of us completing the Lab in time.

I must agree with Brian regarding the initial low budget. A solution using ultrasonic sensors would have been quite elegant and precise and I feel that if the budget had been set at $50 initially we could have bought the sensors and implemented them. But that is left for retrospection since we haven't had any experience using ultrasonic sensors and anything could have happened.

Individual Time Log

Brian Cohen:
Time Spent Inside the Lab: 70 hours
Time Spent Outside the Lab: 3 hours

Mithun Jacob:
Time Spent Inside the Lab: 70 hours
Time Spent Outside the Lab: 3 hours

Ibrahim Emre Celikkale:
Time Spent Inside the Lab: 0 hours
Time Spent Outside the Lab: 0 hours