

FINAL PROJECT: LOST

By Brian Cohen
Mithun Jacob
MEAM 510: Mechatronics
Professor Yim/Fiene
December 19, 2007

BACKGROUND THEORY & BRAINSTORMING

Concept

The constraints for this project were as follows:

- ✚ One base robot which needs to fit within a 22 cm cube
- ✚ Two droid robots which need to fit within 7.5 cm cubes
- ✚ All robots must carry their own power and operate autonomously
- ✚ No more than \$50 is allotted for the entire system
- ✚ Robots should be able to navigate 15 cm tall obstacles with and without mirrors

Since these requirements were fairly broad, we were able to brainstorm various different concepts that would satisfy the objective.

One idea was to have all three robots search for each other and subsequently meet at some specified (or unspecified) location. A second option would be to have the larger base robot remain immobile and have the two other robots move towards and contact it. A third alternative would be to keep the droid robots stationary and have the base robot move around to pick up the droid robots. Of these three alternatives, the second seemed the most intuitive. Having either the base or droid robots immobile would make the design, construction and assembly easier while lowering cost (less motors and driver chips, etc.). Thus we ruled out the first concept. Additionally, having an ultra-smart base robot navigate the course and acquire the two smaller droids included the problem of hauling a droid robot around and adequately securing it to the base robot during transport. Moreover, since the obstacles could be spaced as small as 15 cm apart, the base robot would need more complicated navigation algorithms to traverse the course and retrieve the droid robots (assuming it would be close to its maximal allowed size).

Another primary concern, based on previous group projects, was familiarity. We wanted to use systems with which we had prior experience using. Given the relatively short amount of time that we had for this project, we put a high weighting on simplicity and familiarity of sub-systems and designs.

Locomotion

The next important consideration was locomotion. The two ideas we strongly considered were conventional wheels and treads (such as those used on a tank). The benefit of using treads would be a more stable configuration for the droids and possibly more accurate direction control (the robot would move straight given a command to run both motors forwards). However, since treads would be much more difficult to assemble (and more expensive given that we didn't have spare rubber) we decided on conventional wheels. Moreover, future problems involving directional control and stability could be solved using software algorithms and precision manufacturing/assembly.

Long Range Sensors

We realized that devoid of long range sensors, the droid robots would have to randomly move around the field until they found the base robot. Since there was a significant

weight on the time it would take for the robots to reach the final configuration, we quickly ruled out haphazard motion. Instead our second step was to determine a feasible long range sensor. We listed every type of long range detection scheme we could think of including but not limited to the following sensor systems:

- ✚ LED/phototransistor (IR)
- ✚ Speaker/microphone (Sound)
- ✚ Ultrasonic/ flat surface (Sound)
- ✚ Lasers Optics (Monochrome Light)
- ✚ Antenna/receiver (Radio Waves)

Mithun had used previously used speakers and the course had already provided us with more than enough practice using LED/Phototransistor systems. Testing several loaned speakers and microphones, we quickly realized that sound was a poor choice given the size of the field. The microphones could not pick up sound past approximately 2 feet when pointed directly at a computer speaker outputting loud sounds (approximately 40-60 decibel level). Moreover, the voltage level output from the microphone was fairly constant in all directions (pointing towards and away from the speaker).

Additionally, we had a cost constraint of \$50. Devices such as ultrasonic rangefinders cost around \$20 per system and would quickly exhaust our funds. The same problem arose for laser-based systems. Finally, taking Mark Yim's advice about radio waves, we decided to steer clear of that system given its complexity.

Short Range Sensors

Short range sensors would be critical for obstacles as well as the final configuration (all robots touching each other). Therefore, some of the sensors we considered are listed below:

- ✚ Thermal sensors/heater or flame (Temperature)
- ✚ Hall Effect sensors/magnets (Magnetic)
- ✚ Touch sensors/ wall or obstacle (Force)

Considering thermal sensors (such as the zener diodes provided in the GM store) we realized that if one or several of the robots had heat producing elements, they would be able to distinguish between each other and regular obstacles. However, placing a heater or flame on either the droids would be impossible given the small size restraint (and dangerous as well as non-reusable if they were lit on fire!). However, the base robot was large enough that such a device could be placed on the base robot.

Hall Effect sensors were much more practical given that magnets are cheap and easy to come by. Likewise, although we did not have any experience with these sensors, the circuit setup was pretty simply (power, ground, output).

Finally, touch sensors are excellent for sensing walls or other robots. The circuit is simply a switch with a pull-up (or pull-down) resistor attached to power and ground. However, the big issue with touch sensors is their inability to distinguish contacted objects (such as a wall from another robot). Hence their applicability was limited.

Obstacle Avoidance

In order to solve the problem of obstacles there were two primary concerns that we immediately turned our attention to. The first was the fact that our droid robots could not see over the obstacles, and thus would need some sensors to recognize walls (apart from the actual base robot) as well as complex navigation algorithms to determine in what direction the base robot is most likely located. Second, the mirrors could give the droid robots the impression that a wall is actually the base robot, unless additional sensors were implemented. Moreover, they could confuse the droids about which direction the base robot is from their locations.

DESIGN





Mechanical Design: Droid Robots

Our solution accounted for both issues concerning obstacle avoidance. Using some basic geometry and a bit of creativity, we were able to design droid robots which could carry an arm that would extend above the obstacles. The arm had a phototransistor attached to the end of it which could locate a LED beacon. Effectively implementing this solution, we completely diffused the problem of obstacles, with the exception of a simple navigation algorithm. Otherwise the droid robots always knew where the base robot is from their positions, except at extremely large distances (greater than 3 meters).

The issue still remained, however, that the droid robots would not know whether an obstacle was the base robot or a plain wall. To distinguish between the two, we used a Hall Effect sensor to output a logic level low when it came into contact with one of the many magnets located around the periphery of the base robot.

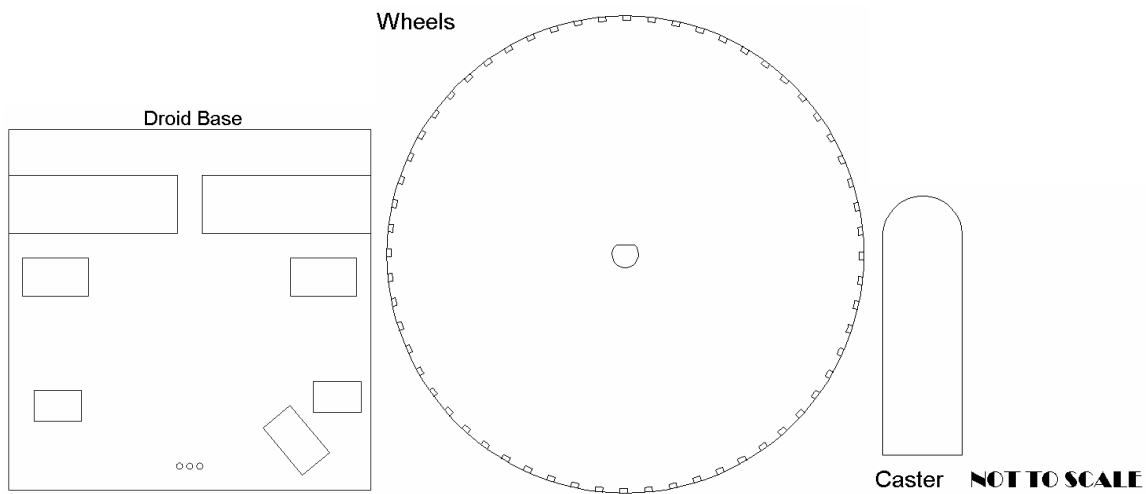
Finally, we used two separate switches to act as bumpers such that when the droid collided with an obstacle, it would read a logic level high on one or both of the input pins. The bumpers were thin rectangular acrylic pieces hot-glued to sensitive tactile switches (25 gmf).

In order to assemble the entire droid robot, we had to come up with some fairly nifty ways of making all the subsystems fit compactly into a 7.5 cm cube. All in all, we had the following components to include:

-  Circuit board
-  Arm with mounted phototransistor (including DC pager motor)
-  Two motors/wheels
-  Two switches/bumpers

-  9 Volt battery
-  Castor

To mount the circuit board, we used two standoffs and kept the third edge pressed against the arm (which acted as a vertical support). The standoffs were kept to a minimal height to allow the arm to rest as low as possible on the circuit board. Geometrically, the lower the arm was allowed to rest, the longer it could be while still fitting the size constraint. It was also important to attach the arm such that it lay along the diagonal of the base, once again allowing for maximal length. Directly beneath the circuit board we cut two slots, helping increase the usable space by removing material from the base to accommodate the motors (press-fit into the slots). Since the entire topside of the base consisted of a circuit board and arm, we glued the two switches underneath the base. Likewise, a 9 Volt battery was glued underneath the base, in between the switches and below the motors. Subsequently, we designed wheels with the appropriate diameter to accommodate the dimensions of the battery. Moreover, since the provided wheels were too thick to allow the motors to be mounted coaxially, thinner wheels needed to be manufactured. The thinner wheels were the width of MDF, or approximately 0.22". The castor, a single piece of acrylic curved at one end, was glued underneath the base as well, directly in front of the battery and in between the bumpers. Please refer to the pictures below:





Initial State



Activated

Mechanical Design: Base Robot

The primary requirements of the Base Robot were to act as a high powered omnidirectional beacon for the droid robots to converge at and provide a large periphery so both robots would converge comfortably without bumping into each other. These requirements were satisfied by designing a rotating beacon with LEDs uniformly distributed around the periphery of the base. Initially, we had planned to mount the Nanocore onto the base robot, making it responsible for pulsing the LEDs and controlling the beacon's rotary movement. With this in mind, the *motor shaft* was kept stationary and the rest of the motor with the leads (connected through a MOSFET to the Nanocore) would rotate along with the Nanocore, and the LEDs.

But as mentioned before, we wished to make our design as simple as possible and once we learned about the application of 555 Timers, we decided to directly connect the motor to the power supply and run the PWM power supply of the LEDs off a MOSFET-555 Timer combination. Thus the mass atop the rotating base robot was significantly reduced by replacing the Nanocore with a 555 Timer and we feel this is a highly elegant solution compared to using the bulky Nanocore/9s12 as the PWM generator.

The Hall Effect sensor on the droid robot generates a steady voltage at about 2.5 V in the absence of any magnetic fields and goes low to ~250 mV the presence of one of the pole, but also goes high to ~5V in the presence of the other pole. This situation was highly dangerous for the TI since it would certainly damage it. The magnets were uniformly distributed about the periphery of the base robot and since the magnets were so closely spaced, the area between two magnets would have a magnetic field which would generate a voltage of ~5V. To prevent the Hall Effect sensor from ever entering this field, a toothed disk was manufactured with the teeth placed above the space between magnets thus physically preventing the Hall Effect sensor from ever entering the aforementioned magnetic field.



Docking at the Base

Electrical Design

The phototransistor circuit was the most critical subsystem on our robot. It effectively allowed the droid robots to locate the base robot and change make the necessary course corrections. Essentially, the phototransistor circuit consisted of three sections: the phototransistor in a transresistive configuration, an RC filter, and an op-amp used as an inverting gain. The transresistive configuration guaranteed a 5.7 volt supply across the phototransistor, maintaining the relationship between IR irradiance and output current. This relationship was important when using the A2D converter to indicate if certain thresholds in the irradiance levels were reached (and subsequently determining which actions to take). Due to an offset voltage (which we later realized was not necessary because of the inverting gain) the following equation governed the output from this initial stage in the circuit:

$$V_{out} = V_{Offset} - i_{phototransistor} R$$

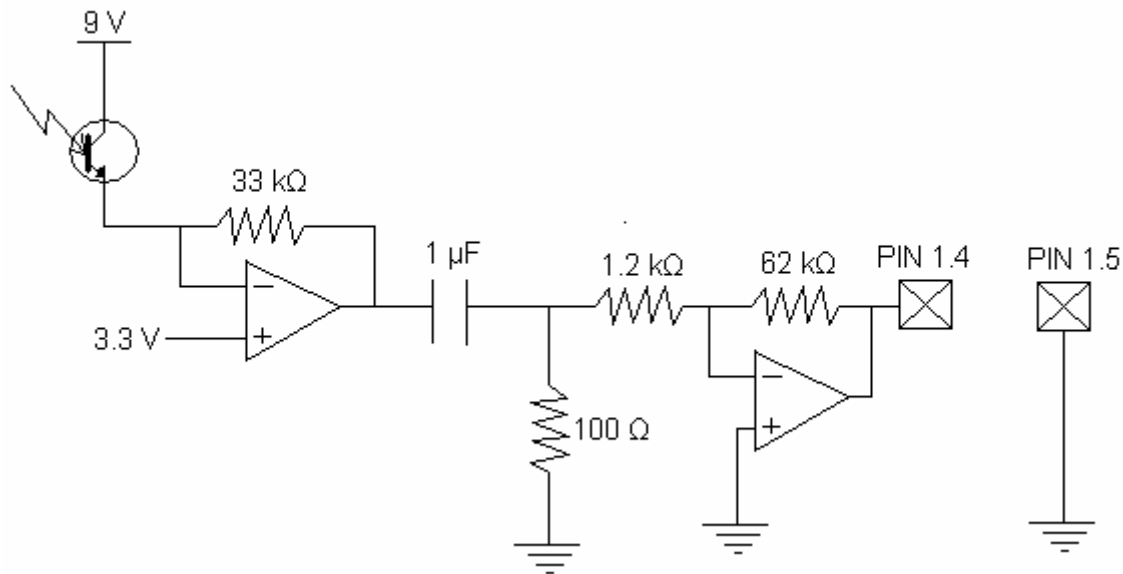
Given the large value for R, the output signal was always negative. The second stage of the circuit, a high-pass RC filter, blocked out low frequency signals such as noise from the ambient environment. The equation used to determine the cut-off frequency is the following:

$$f_c = \frac{1}{2\pi RC}$$

Based on the resistor and capacitor values, the cut-off frequency was set at around 1600 Hz, which allowed the LED signal pulsed at 3300 Hz to pass through without attenuation. Lastly, the op-amp in an inverting gain configuration provided amplification of around 50x the input signal. This extremely large gain was necessary to acquire the LED signal

at large distances. Below is a schematic of the phototransistor circuit:

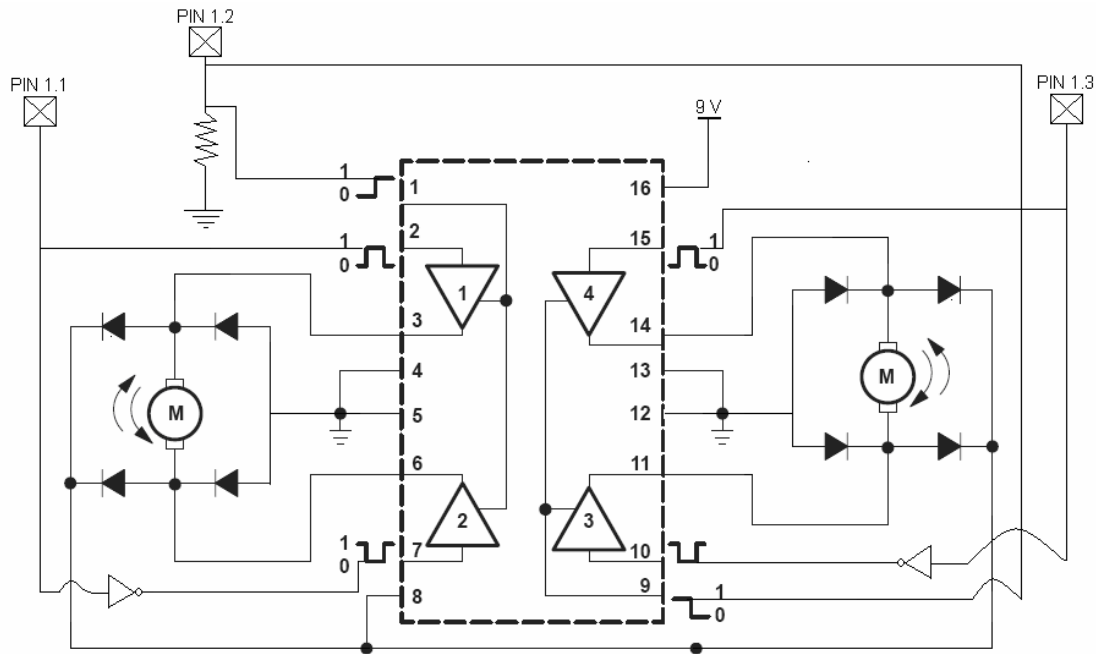
Phototransistor:



Note: A potentiometer was included immediately before pin 1.4 to vary the peak voltages so that the pin was not blown when the droid robot was at close proximity to the base robot

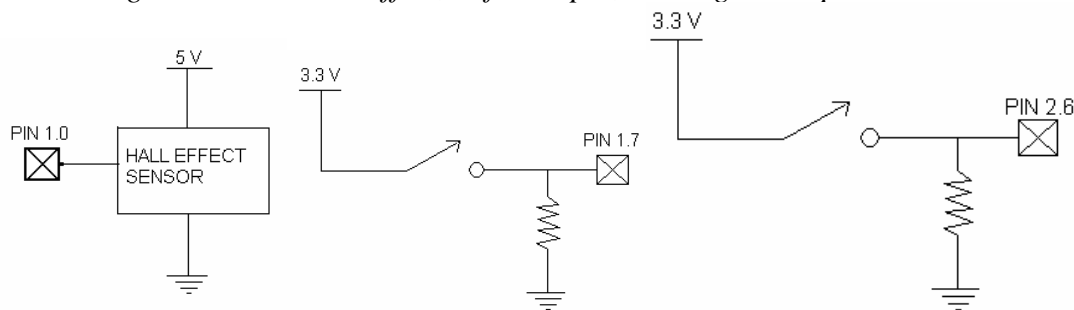
Another critical subsystem was the motor circuit. The main component of this system was the L293D driver chip, which includes two built-in H-bridges. This chip also has the added benefit of a darlington by increasing the current output to sufficient levels in order to drive the motors. Moreover, since there were a limited number of pins, we used a single PWM signal (split into two lines) to drive the motors. Thus, the robot was limited to motion forwards, backwards and zero-radius turning. Two pins (one for each motor) were used to control directionality. Splitting each logic level into two lines and inverting one of those lines, we were able to run each motor in two directions. The inverter we used was a 74HC04 chip. Again, using an inverter helped to limit the number of pins required to control locomotion on each robot. Below is a schematic of the motor driving circuit:

Motor Controller:



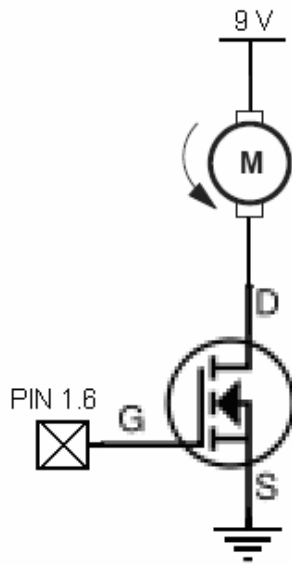
Several other subsystems controlled the on-board short range sensors. These included the bumper and Hall Effect circuits. The bumpers circuits simply consisted of a switch (the bumper) attached to an input logic pin and a 3.3 voltage regulator. Additionally, a pull-down resistor was added to ensure that the pin recorded a logic level low when the switch was open (no collision). The Hall Effect sensor simply required power from the 5 volt regulator and ground. The output from the sensor was attached to an input pin on the microcontroller. Given the orientation of the magnets, the sensor outputted a logic level low in close proximity to the base robot. Normally, this output was a logic level high (2.5 volts). Below are the schematics for these circuits:

Short Range Sensors: HALL effect, Left Bumper, and Right Bumper



The arm on the droid robot was controlled using a MOSFET and a logic pin. The logic pin was connected to the gate of the MOSFET so that when a logic level high was output, the gate would open and run the motor. Conversely, if a logic level low was output to the MOSFET, the motor would turn off. Below is a schematic of the arm circuit:

Arm:



The electrical design of the base robot consists of:

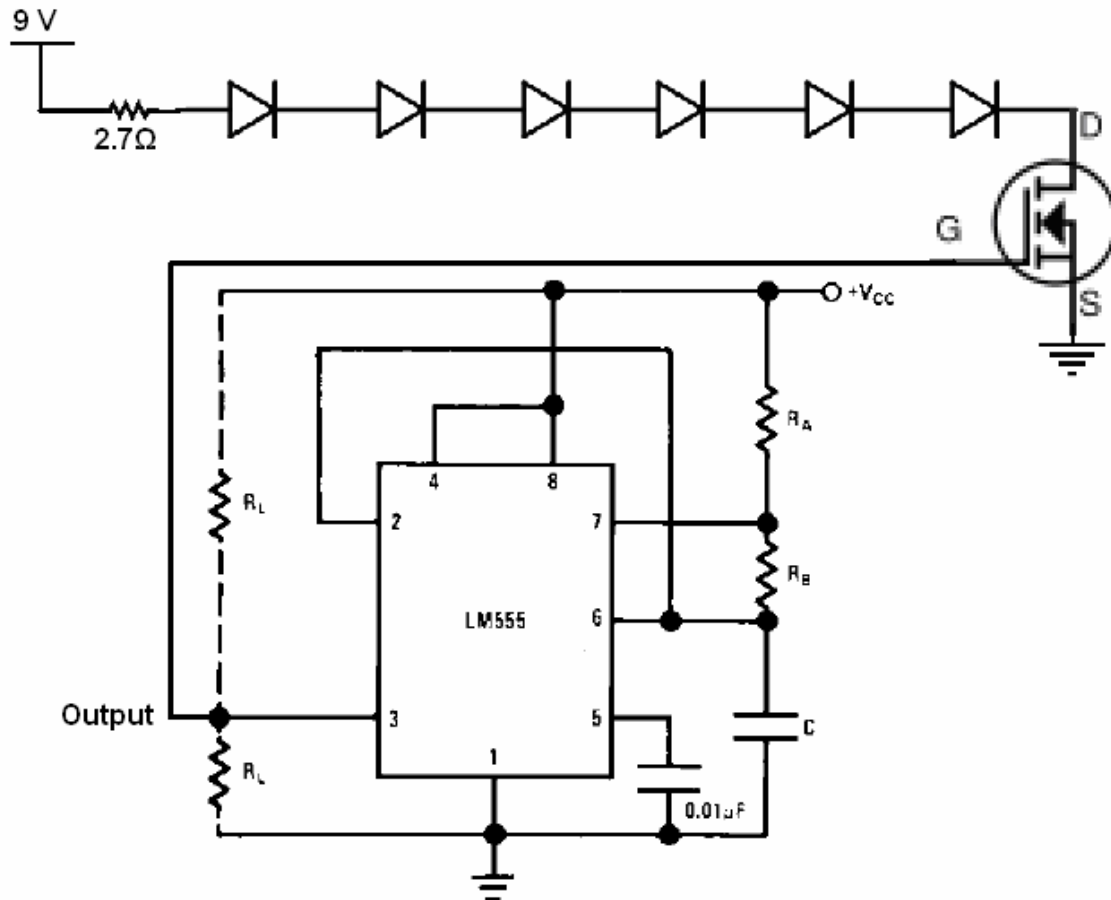
- ✚ 6 LEDs pulsed at 4.5 kHz with a current of 667 mA
 - Since, we wanted the current to be as high as possible without burning out the LEDs, we chose a medium amount of current since the LEDs would be on for a considerable amount of time. Thus using a 2.7Ω resistor,

$$V = IR \Rightarrow \frac{9 - 1.2 \times 6}{2.7} = 667 \text{ mA}$$

- ✚ Motor
- ✚ MOSFET driven by the PWM from a 555 Timer to run the LEDs
 - The 555 Timer had 2 10K pots for R_a and R_b and it used a $0.01\mu\text{F}$ capacitor. The pots were set at 10K each thus it should have a frequency of 4.8 kHz

$$f = \frac{1.44}{(R_a + R_b)C} = \frac{1.44}{(10000 + 20000)0.01 \times 10^{-6}} = 4.8 \text{ kHz}$$

The 555 Timer was in astable mode to provide a continuous PWM to the MOSFET which powered the LEDs. The motor was connected directly to another 9V supply which allowed the LEDs to spin quite fast and thus the droid was able to detect the beacon at a distance of 3-4m away.



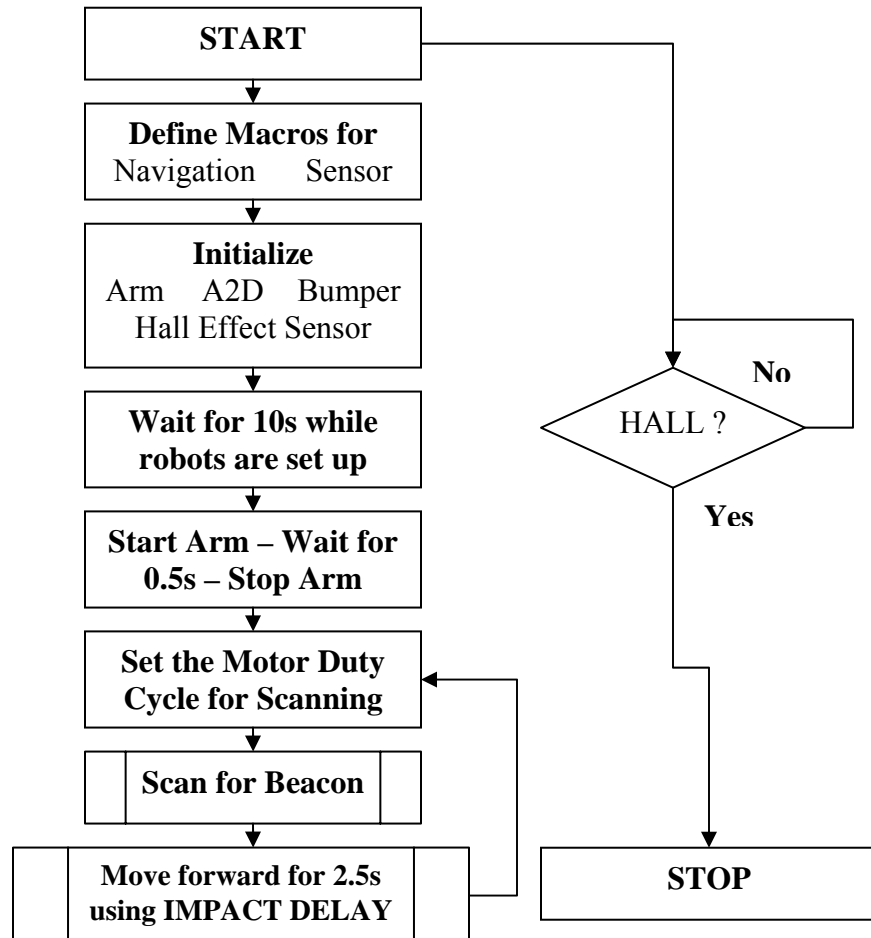
Programming

An efficient obstacle navigation system and beacon detection algorithm was the key to clearing all the rounds of the contest. Since our robots were able to detect the beacon above the obstacle, the only difference between successfully completing round 1 and round 2 was the efficiency of the obstacle navigation system. Before, we delve into the details of the implemented algorithm; here are some of the important tasks handled by the droid program.

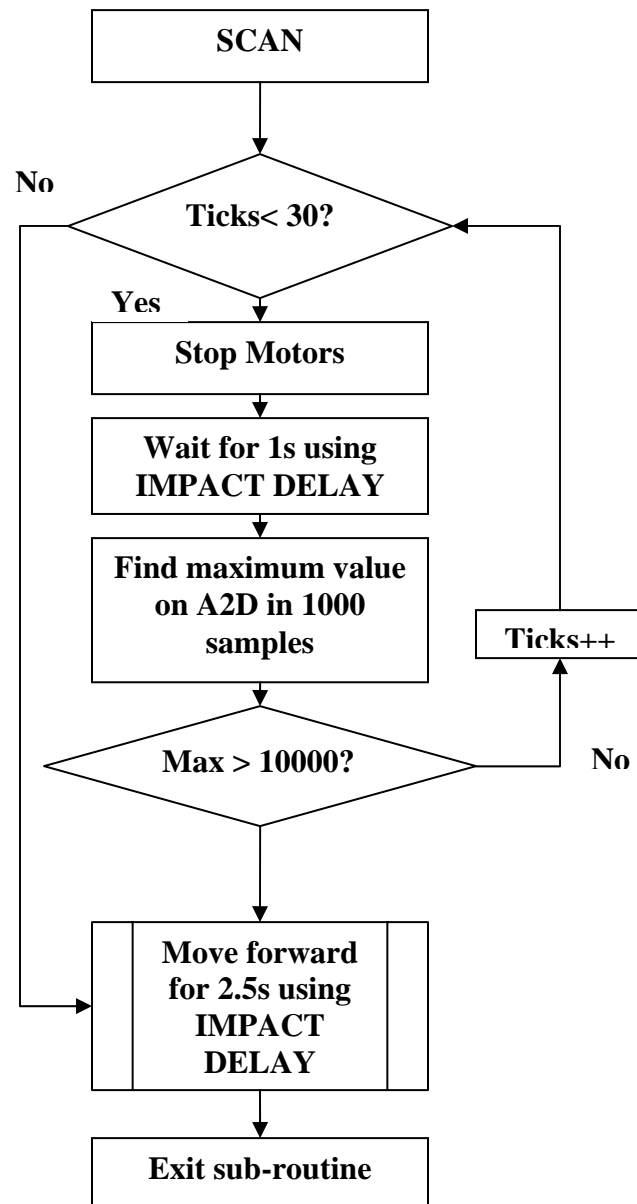
1. Run the motor which lifts the arm above the obstacles for a short time period. If the pager motor ran for too long, or wasn't stopped, it would eventually burn out.
2. Efficiently navigate obstacles and tracks direction of the beacon
3. Continuously monitor the Hall effect sensor using interrupts
4. Include the capacity to act randomly in the event when the robot was stuck in an unplanned scenario (eventual exploration over exploitation of a policy when the policy fails).

When the droid program was developed, it was apparent that a sizeable number of tasks would have to be handled by the program efficiently guiding the robot through a number of tasks. With this in mind, the overlying architecture for the program was designed to be

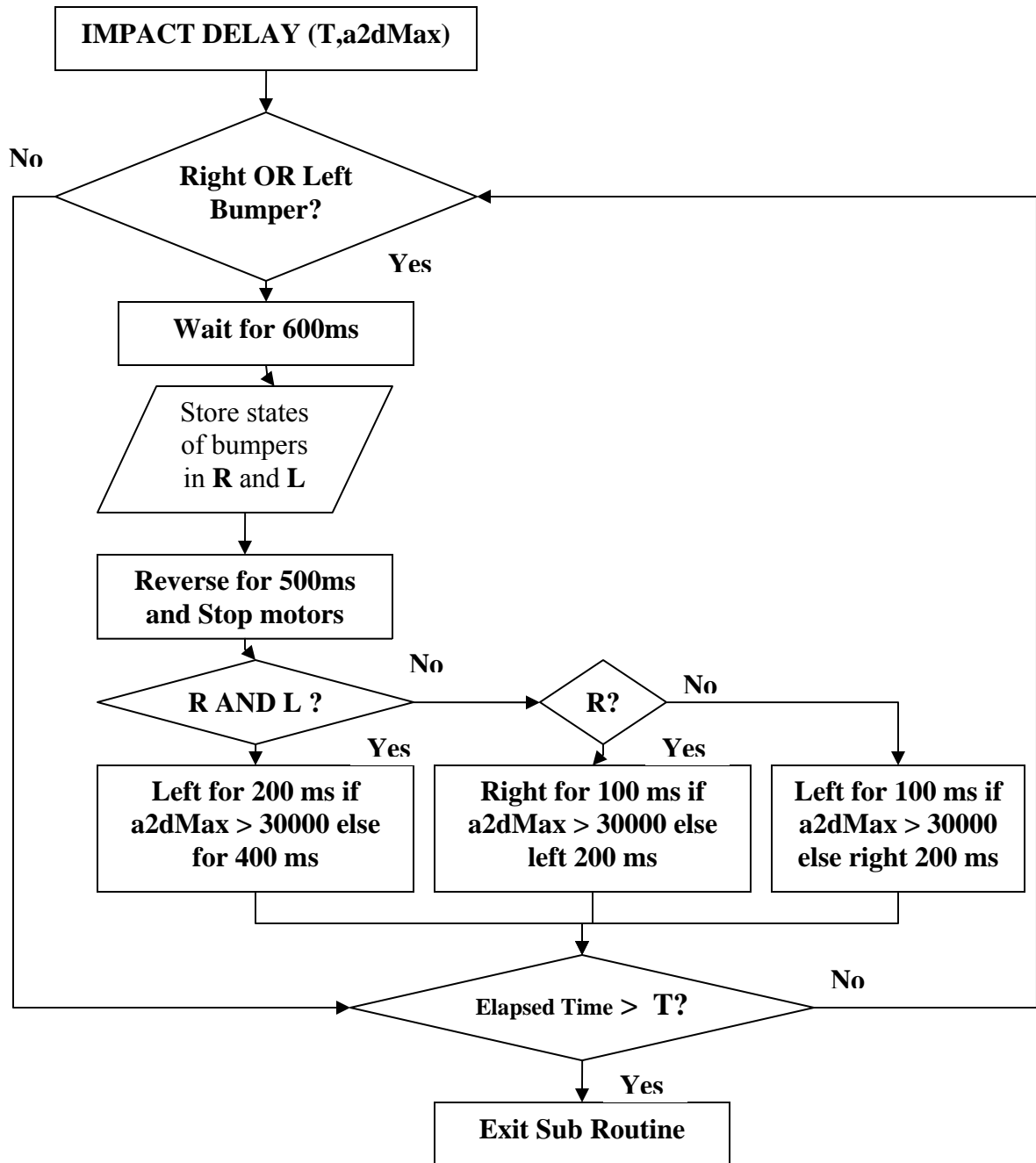
modular to allow for easy debugging and modification to the code over time. The general algorithm can be demonstrated in the flowchart below:



The algorithm for the SCAN sub routine is described below. This subroutine was used to detect the beacon and move towards it. More importantly, it stored the value of the voltage at the A2D as a global variable, allowing the obstacle navigation system to make smart decisions based on how close it was to the base robot.



The IMPACT DELAY function takes care of obstacle navigation. Using values obtained from the A2D, it makes shorter turns if near the base robot (to increase the chances of the Hall Effect sensor making contact with a magnet) and longer turns if it is far from the base robot (to speed up the obstacle navigation process).



The complete code is included in Appendix A.

CONCLUSIONS

Overall Success

Please see our YouTube video located at the following web-address:

<http://www.youtube.com/watch?v=228wENLYX4o>

The droid robots were able to reach the base robot with and without obstacles on the field. During out testing, it was clear the obstacles were not a problem for our robots because they could “see” over the with the arm mechanism. The algorithms written for

Improvements

There are several aspects of our system that could have been improved. These include electrical, mechanical and programming features.

First, the electrical circuit for the phototransistor could have been designed differently. The transresistive configuration did not require an offset voltage since the output signal was already negative (due to a large gain). Using an inverting configuration for the gain, this negative signal would always be made positive before entering the A2D converter. Thus, the non-inverting input of the op-amp could simply be tied to ground. Additionally, the Hall Effect sensors had some difficulty in recognizing the magnets when the droid robots contacted the base robot. A better solution would have been to have magnets on both the droid and base robots so that the droids would literally be pulled to the base robot. Then a circuit would be closed when the two magnets collided and indicate that contact has been made.

With regards to mechanical design, the wheels of the robot could possibly have been smaller, and hence less problematic for controlling the direction of the droid robots’ motion. By mounting the wheels underneath the base, offsetting them farther towards the rear, it is possible that there might be enough room for the battery to still fit in its original location. Furthermore, designing a set screw into the wheel, although time-consuming, would have helped maintain an orthogonal position on the motor axle and prevent the wheel from wobbling. Finally, mounting the phototransistor on the arm such that it maintained a consistent orientation every time it was raised would have helped maintain alignment with the LED’s on the base robot. A small ledge or inset on the arm would have been sufficient to keep the phototransistor from moving and ensure good alignment.

Finally, the algorithm development and implementation segment of our process was quite smooth. Everything worked as expected and the system worked flawlessly. Of course, there's a lot of room for improvement since the obstacle navigation algorithm could have been extended to handle corners and other complicated scenarios.

Feedback

Brian: This final project was an excellent way in which to test our skills and knowledge acquired throughout the semester. Given that multiple robots were needed, there was a lot of manufacturing required but in the end, it brought to our attention important issues about mechanical design (for example using one instead of two castors). It was quite rewarding to see the robots working in the end, especially since they were built from scrap material and cheap electronics components. For the future, it might be a good idea to cover more material on other sensors and provide these sensors in the GM store (such as ultrasonic rangefinders) so that students will be more familiar with them. A project

that requires using sensors other than Phototransistor/LED's would also be interesting. Finally, allocating more time to work on the final project would have been preferable. Thanks for a great semester!

Mithun: Throughout the semester, and through all the labs, we've never actually built an autonomous robot. The first robot which would oscillate between walls in a (nearly!) straight line or the teleoperated robot pale in comparison to two mobile autonomous robots capable of navigating obstacles and converging at a point. It was an excellent opportunity to integrate all the electronic systems we developed throughout the semester into one compact package and actually run them with a program! This project covered everything; design, manufacturing, electronics, quality testing, programming. It was a lot of fun, and an excellent learning experience. Thanks!

TIME LOG

Brian

Time Spent in the lab: 70 hours

Time Spent outside the lab: 2 hours

Mithun

Time Spent in the lab: 70 hours

Time Spent outside the lab: 2 hours

Appendix A:

Droid Robot Code

```
#include <msp430x20x3.h>
```

```
/*
1 PWM pin           1.2           P4
Extra PWM           1.6           P8
2 Enable Pins       1.1,1.3       P3 & P5
A2D (+,-)           1.4,1.5       P6 & P7
RBumper             1.7           P9
LBumper             2.6           P13
Touch Sensor        1.7           P9
Hall Effect         1.0           P2
*/

//Red
/*
#define LFT (P1OUT |= BIT1,P1OUT &= ~BIT3)
#define RGT (P1OUT &= ~BIT1,P1OUT |= BIT3)
#define FWD (P1OUT |= BIT1,P1OUT |= BIT3)
#define REV (P1OUT &= ~BIT1,P1OUT &= ~BIT3)
#define STOP (P1OUT &= ~BIT1,P1OUT &= ~BIT3,P1SEL &= ~BIT2,P1OUT &= ~BIT2)
#define START (P1SEL |= BIT2)

#define LBUMP ((P2IN&BIT6)!=0)
#define RBUMP ((P1IN&BIT7)!=0)
#define HEADON ((RBUMP&&LBUMP))
#define HALL ((P1IN&BIT0)==0)
```



```

#define SENS 200

#define ARMDELAY 400
#define ARMPER 400
#define ARMDTY 160*/

//Purple - The Purple robot has different navigational settings. Hence each
//robot is loaded with a slightly different program.

#define REV (P1OUT |= BIT1,P1OUT &= ~BIT3)
#define FWD (P1OUT &= ~BIT1,P1OUT |= BIT3)
#define RGT (P1OUT |= BIT1,P1OUT |= BIT3)
#define LFT (P1OUT &= ~BIT1,P1OUT &= ~BIT3)
#define STOP (P1OUT &= ~BIT1,P1OUT &= ~BIT3,P1SEL &= ~BIT2,P1OUT &= ~BIT2)
#define START (P1SEL |= BIT2)

#define LBUMP ((P2IN&BIT6)!=0)
#define RBUMP ((P1IN&BIT7)!=0)
#define HEADON ((RBUMP&&LBUMP))
#define HALL ((P1IN&BIT0)==0)

#define SENS 200

#define ARMDELAY 500
#define ARMPER 500
#define ARMDTY 500

void InitA2D();
void InitSensors();
void SetMotors(int Per, int Dty);

void SetArm();
void StartArm(int Per, int Dty);
void StopArm();

int Scan(int Togg);

void Delay(int T);
void InitDelay();
int ImpactDelay(int ID);

int HandleSensors(int BT);

int Scans=0;
int atdMax=0;

#pragma vector=PORT1_VECTOR
__interrupt void Port_1(void)
{
    STOP;
    for(;;){
    }
}

int main( void )
{
    WDTCTL = WDTPW + WDTHOLD;

```

```

SetArm();           //Initialize the arm parameters
InitA2D();          //Initialize the A2D
InitSensors();      //Initialize bumper & Hall effect sensor
InitDelay();        //Initial Delay period to set bots up


StartArm(ARMPER,ARMDTY); //Starts the arm at specified parameters
Delay(ARMDELAY);        //Lets it run for ARMDELAY ms
StopArm();             //Stops the arm


    SetMotors(400,160); //Sets the motor duty cycle
    Scan(0);           //Scans once around
    for(;;){
        STOP;          //Stops
        FWD;           //Sets Forward motion parameters
        START;         //Starts the PWM
        ImpactDelay(2500); //Delay function which scans sensors too

        STOP;          //Stops

        SetMotors(400,160); //Sets motor duty cycle again
        Scan(0);           //Scans around

    }
    return 0;
}


void Delay(int Time) //Timer-independent Delay Function
{
    float i,Lim=Time*3;
    for(i=0;i<Lim;++i){}
}


int HandleSensors(int BT)
{
    int R,L;
    if(RBUMP || LBUMP) //If either one of the bumpers are on
    {
        STOP;
        Delay(600);    //Stop & wait for 600 ms

        R=RBUMP;
        L=LBUMP;

        REV;           //Reverse for 500 ms and stop
        START;
        Delay(500);
        STOP;
        if(R && L)      //If it's a head on collision, turn left massively
        {              //except if it's near the Beacon..else, smaller turn
            LFT;
            START;
            if(atdMax >30000)
                Delay(BT);
            else
                Delay(BT*2);
        }

    }

    else if(R){        //Same goes for R except: Small Right near Beacon

```

```

        if(atdMax >30000)      //Bigger Left otherwise
        {
            RGT;
            START;
            Delay(BT*5/10);
        }
        else
        {
            LFT;
            START;
            Delay(BT);
        }
    }

    else if(L)                //Same goes for L except: Small Left near Beacon
    {                          // Bigger Right otherwise
        if(atdMax >30000)
        {
            LFT;
            START;
            Delay(BT*5/10);
        }
        else
        {
            RGT;
            START;
            Delay(BT);
        }
    }

    STOP;                    //Stops and moves forward for a shorter time if near Beacon
    FWD;                      //else it moves forward for a longer time
    START;
    if(atdMax >30000) Delay(BT);
    else Delay(BT*2);

}

return 0;
}

```

```

void InitDelay()
{

```

```

    Delay(10000);
}

```

```

int ImpactDelay(int ID)
{

```

```

    //HandleSensors checks for Hall effect too in its delay

    float i,Lim=ID*3;
    for(i=0;i<Lim;++i){HandleSensors(SENS);}
    return 0;
}

```

```

void InitA2D()
{

```

```

    SD16CTL = SD16SSEL_1 | SD16REFON; // SMCLK source, Vref ON
    SD16CCTL0 |= SD16UNI; // unipolar
    SD16AE = SD16AE4 | SD16AE5; // enable analog on A2 pins

```

```

    SD16CCTL0 &= ~SD16SNGL; // single conversion
    SD16INCTL0 = SD16INCH_2; // input channel A2
}

```

```

void SetArm()

```

```

{
//Sets the PWM for the arm
    P1SEL |= BIT6;
    P1DIR |= BIT6;
    P1OUT &= ~BIT6;
}

```

```

void StartArm(int Per, int Dty)

```

```

{
    //1.6 direction set in SetArm
    //Starts the PWM for the arm
    TACCR0 = Per;
    TACCR1 = Dty;
    TACCTL1 = OUTMOD_7;
    TACTL = TASSEL_2 | MC_1;
}

```

```

void StopArm()

```

```

{
//Stops the PWM
    TACTL |= MC_0;
    P1OUT &= ~BIT6;
    P1SEL &= ~BIT6;
}

```

```

void SetMotors(int Per, int Dty)

```

```

{
//Sets the PWM for the motors
    P1SEL |= BIT2;
    P1DIR |= BIT1 | BIT2 | BIT3;
    TACCR0 = Per;
    TACCR1 = Dty;
    TACCTL1 = OUTMOD_7;
    TACTL = TASSEL_2 | MC_1;
    STOP;
}

```

```

void InitSensors()

```

```

{
    //P1.7 (Left)
    P1SEL &= ~BIT7;
    P1REN |= BIT7;
    P1DIR &= ~BIT7;
    //P2.6 (Right)
    P2SEL &= ~BIT6;
    P2REN |= BIT6;
    P2DIR &= ~BIT6;
    //P1.0 (HALL)
    P1SEL &= ~BIT0;
    P1REN |= BIT0;
    P1DIR &= ~BIT0;
    P1IE |= BIT0; // P1.0 interrupt enabled
    P1IES |= BIT0; // P1.0 Hi/lo edge
    P1IFG &= ~BIT0; // P1.0 IFG cleared
    _BIS_SR(GIE); // Enable interrupts
}

```

```
}
```

```
int Scan(int Togg)
```

```
{
    int i,Avg=100,MaxSample=1000,Ticks=0;
    unsigned int atdSum=0;

    SD16CCTL0 |= SD16SC; //Starts conversion
    for(Ticks=0;Ticks<30;++Ticks)//Number of sweeps
    {
        STOP;
        ImpactDelay(1000);          //Stops for 1000ms

        atdMax=0;
        for(i=0;i<MaxSample;++i)
        {

            while(!(SD16CCTL0 & SD16IFG))
            {
                HandleSensors(SENS);      //Checks sensors while scanning
            }

            if(SD16MEM0 > atdMax) atdMax=SD16MEM0;
            SD16CCTL0 &= ~SD16IFG;
        }

        RGT;
        START;
        ImpactDelay(100);
        if(atdMax>10000) return 0; //If above threshold, exit
    }

    STOP;
    FWD;
    START;
    ImpactDelay(2500);      //If it scans and no voltage was found move FWD
    Scans=0;

    return 0;
}
```