University of Florida
Department of Electrical Engineering

EEL 5666
Intelligent Machines Design Laboratory

Final Report

An Autonomous Domino Dispensing Robot

By: Arfath Pasha
April 15, 1999
arfath@ccgnv.net
(352) 222-2352

My autonomous robot- DomiGnome has been developed to dispense dominoes along a predefined path stored in memory in the form of an array.

Filename: domignome.doc

CONTENTS

ABSTRACT

My autonomous robot- DomiGnome has been developed to dispense dominoes along a predefined path that is stored in memory in the form of a one-dimensional array. DomiGnome required two hacked servos to power its wheels and two regular servos to power the dispensing mechanism. It also employed IR sensors and bump sensors for obstacle detection, and to check the presence of dominoes in the domino stack.

DomiGnome's behaviors include setting dominoes upright in a pre-programmed path, detecting obstacles, sensing the completion of dominoes in the stack, and, setting off the chain reaction once it has completed the path. The dispensing action is partly powered and partly gravity fed. All the parts were fabricated out of wood and aluminum by the author to provide a smooth and repeatable dispensing action.

EXECUTIVE SUMMARY

The main goal during the development of this project was to have a robot that could dispense dominoes in a pre-programmed path while it interacted with its surroundings through various sensors and actuators.

This goal was broken down into four sub tasks or behaviors, which, in their order of importance are- dispensing of dominoes in a pre-programmed path, obstacle detection, detection of the completion of dominoes in the domino stack, and, setting off the chain reaction at the end of the programmed path.

The first behavior was achieved by having two independent mechanisms, one to push the bottom most domino out of a vertical stack into an enclosure at the rear of the platform and a second slider mechanism that slid the domino out into the open from the side of the enclosure. The pre-programmed path was stored in an array. Obstacle detection was done with the help of four IR sensors and two bump sensors placed on the front end of the robot. This was due to the fact that the only operating direction of the robot was the forward direction. If the robot sensed an obstacle, it would stop dead in its tracks and wait for the obstacle to be removed. No attempt to circumnavigate the obstacle has been made as this would defeat the purpose of having a continuous pre-programmed path. The last behavior was added at the end of the program where the robot would spin around and knock the last domino that it set, thereby starting the chain reaction through the entire line of dominoes. The robot was powered by two hacked servos driving a differential drive. The robot was supported in the front and back with angle plates that kept it from rocking back and forth. The brain of the robot was a Motorola 68HC11 chip and the power source was derived from six rechargeable Ni-Cd batteries.

INTRODUCTION

In this paper, I have described in detail, the various features of DomiGnome and have also included other material that would be beneficial for anyone who takes up a similar project in the future.

In the chapter- Integrated System, I have described the robot in terms of its sub-systems. I have described the function of each susb-system and their interaction with each other and the environment. In the next chapter, I have given a brief description of the mobile platform, its layout and certain features of the assembly that I considered important The chapter on actuation contains a detailed description of the actuators on DomiGnome- the drive system and the dispenser. All the sensors used have been described in detail in the chapter- Sensors.
All the behaviors have been listed out and explained in detail in the next chapter- Behaviors. A sample path of the robot is also shown here.

The next few chapters contains some experiments that went into making certain systems work effieciently and the results of these experiments.

Besides elaborating on the robot's features, I have included other successful and unsuccessful experimental work that I did during the course of the project. I have also added some information on where one can acquire and make parts that went into making DomiGnome.

INTEGRATED SYSTEM

DomiGnome can be broken down into the following sub-systems-

1. The drive system
2. The dispensing system
3. The obstacle detection system

All these sub-systems come into play in a sequential manner in order to meet the specified objective. The flow of instructions to these systems by the microprocessor is shown in fig.1

At the beginning, the robot initializes itself by setting all the servo angles to an appropriate value and turning on all the IR emitters. Once it has done this, it checks to see if it has come to the end of the path. If it has, it turns around and tips the last domino that it set in order to start the chain reaction. If it has not come to the end of the path, it checks all the sensors for obstacle detection and also checks to see if it has run out of dominoes. If the robot finds that there is an obstalce, it will just wait in its path till the obstacle is cleared. If not, it will proceed to the next instruction level which reads a number from the array that contains the path information. It then dispenses a single domino in its current position and moves forward about half an inch in the desired direction, that is straight for about 10 steps if it read a value of 0 from the path array, left for about 10 steps if it read a value of 1, right for abut 10 steps if it read a value of 2 and it leaves a gap of 2 inches if it reads a valus of 3. The option of leaving a gap is provided if the programmer of the path wants to place something in the domino chain like ladders, toy cars etc. The robot recognises the end of the path when it reads a value of 5 from the path array. It then cycles through this routine till the end of the path is reached. As seen, all three sub-systems of the robot are constantly being addressed during its operation.

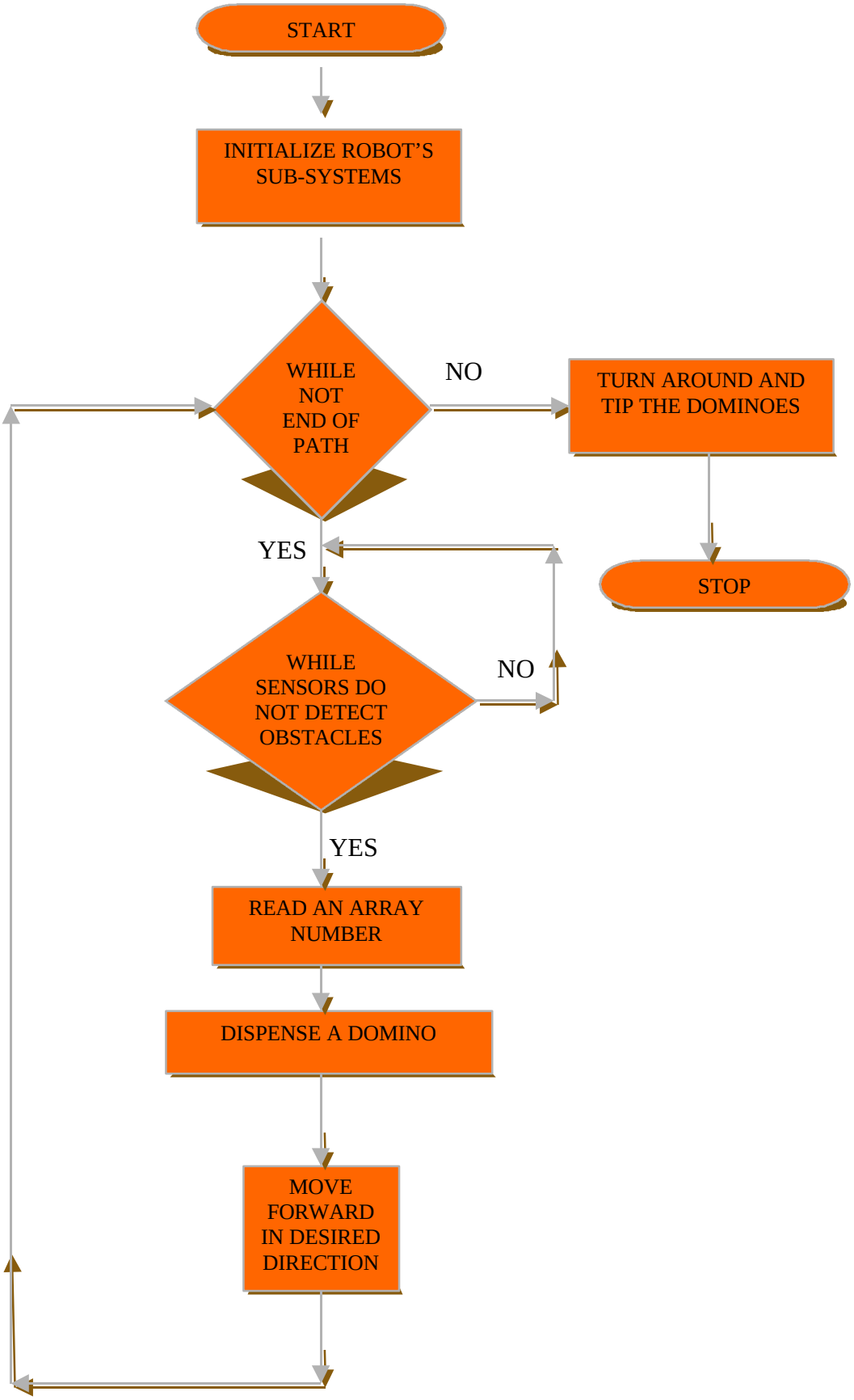

START
INITIALIZE ROBOT'S SUB-SYSTEMS
WHILE NOT END OF PATH
NO
TURN AROUND AND TIP THE DOMINOES
STOP
YES
WHILE SENSORS DO NOT DETECT OBSTACLES
NO
YES
READ AN ARRAY NUMBER
DISPENSE A DOMINO
MOVE FORWARD IN DESIRED DIRECTION

Fig. 1

MOBILE PLATFORM

The platform for the robot is built out of wood. The size of the platform is 10 inshes by 10 inches and it is supported by a pair of wheels 3 inches in diameter on either side about 6 inches from the front end of the robot. The front and rear of the robot are supported by angle plates in order to prevent it from rocking back and forth. Some clearance is given to allow a slight rocking motion to compensate for any snag that may occur in the dispenser. This is a safety feature that would prevent the servos that control the dispenser's slider from burning out should the dominoes get snagged in the dispenser. In the event of a snag, the servos would work on lifting the robot platform instead of pushing the dominoes out. This is made possible only by providing the above mentioned clearance.

The wheels of the robot are supported by a double axle suspension as shown in fig. 2 in order to give them enough stability and to prevent buckling under the wieght of the robot which is fairly heavy. An extra feature was added to the construction of the wheel assembly in the form of slots as shown in fig. 3 this enables easy adjustment of the wheel angle in terms of toe-in or, toe-out or just pointing straight ahead. I found this to be a useful feature as the wheels can be adjusted to compensate, to an extent, the errors caused by other flaws in the assembly such as poorly distributed weight on the platform etc.

Further, everything that has been assembled on the platform has been done so in a fully breakdown manner. That is, every part on the robot can be disassembled and put back together very easily. This also was a great help towards the end when I had fine adjustments to make. If parts are assembled permanently onto the platform, it becomes a problem when something has to

be changed or removed. In order to achieve this kind of an assembly, machine screws of size 6 and 8 along with angle plates were used extensively to prop up componenets like motors, servos, supports etc. on the platform.

The dispenser was placed at the rear of the robot and more or less along the central axis of the robot. It contains a tray and an enclosure into which the dominoes fall before being pushed out into the open by the sliding mechanism. It also contains a stack placed vertically over the tray that is about 12 inches high and is capable of holging about 50 dominoes. The entire layout of the platform is as shown in fig 2.

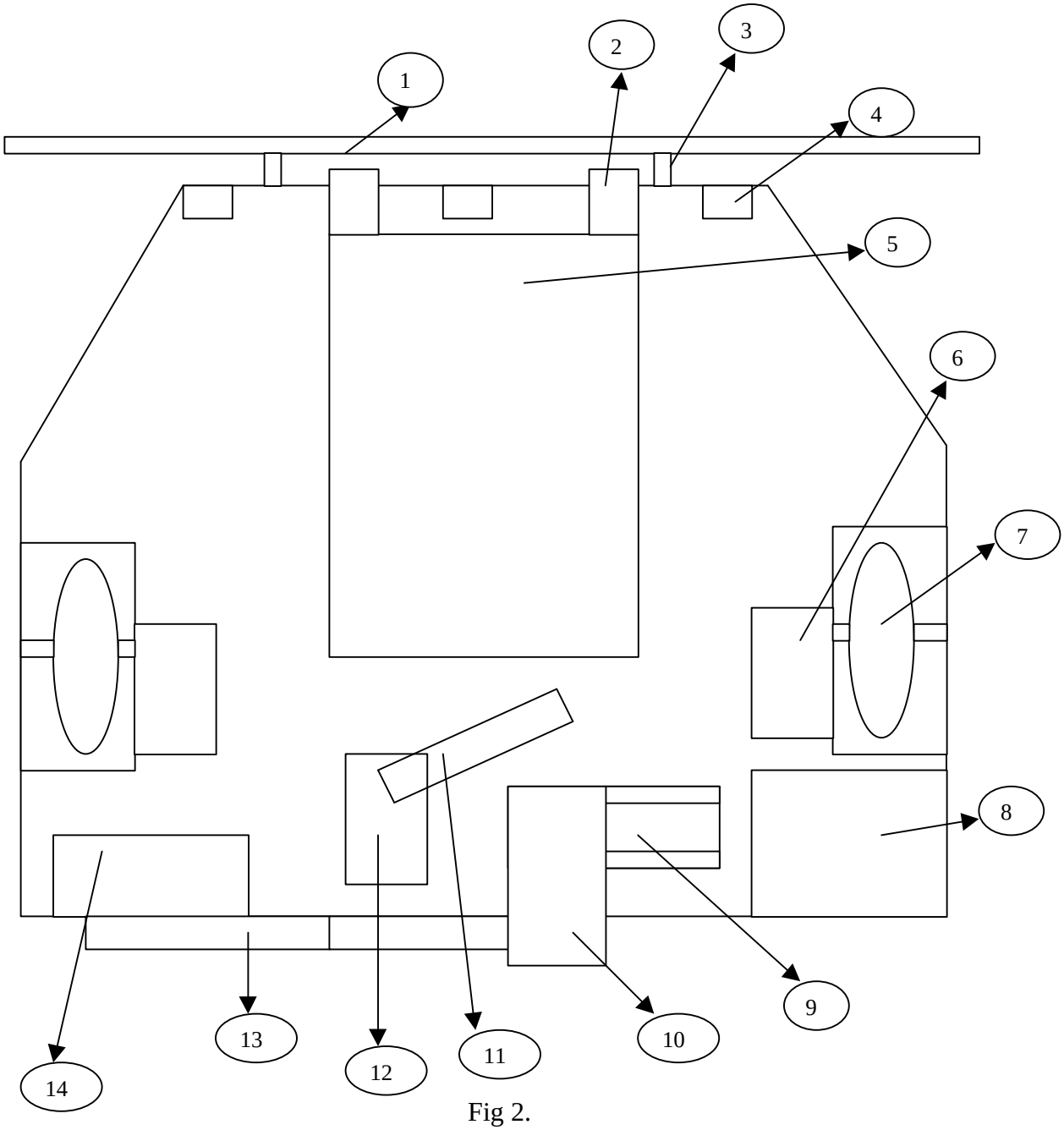


Fig 2.

For fig 2.

1. Bumber (wooden strip held in place with a hinge)
2. Front support plates (1 inch aluminium right angle plates)
3. Bump switch
4. IR emitter-detector pair

5. 68HC11 board
6. Drive motor (42 ounce inch fully hacked servo)
7. 3 inch diamenter wheel with double axle suspension
8. Battery pack (6 Ni-Cd cells)
9. Stack for storing dominoes (height is 12 inches and it holds about 50 dominoes)
10. Dispenser tray (placed below stack)
11. Pusher (connected directly to the servo head)
12. Servo to activate the pusher (42 ounce inch torque)
13. Slider mechanism to push dominoes out into the open
14. Servo to actuate the slider (42 ounce inch torque)

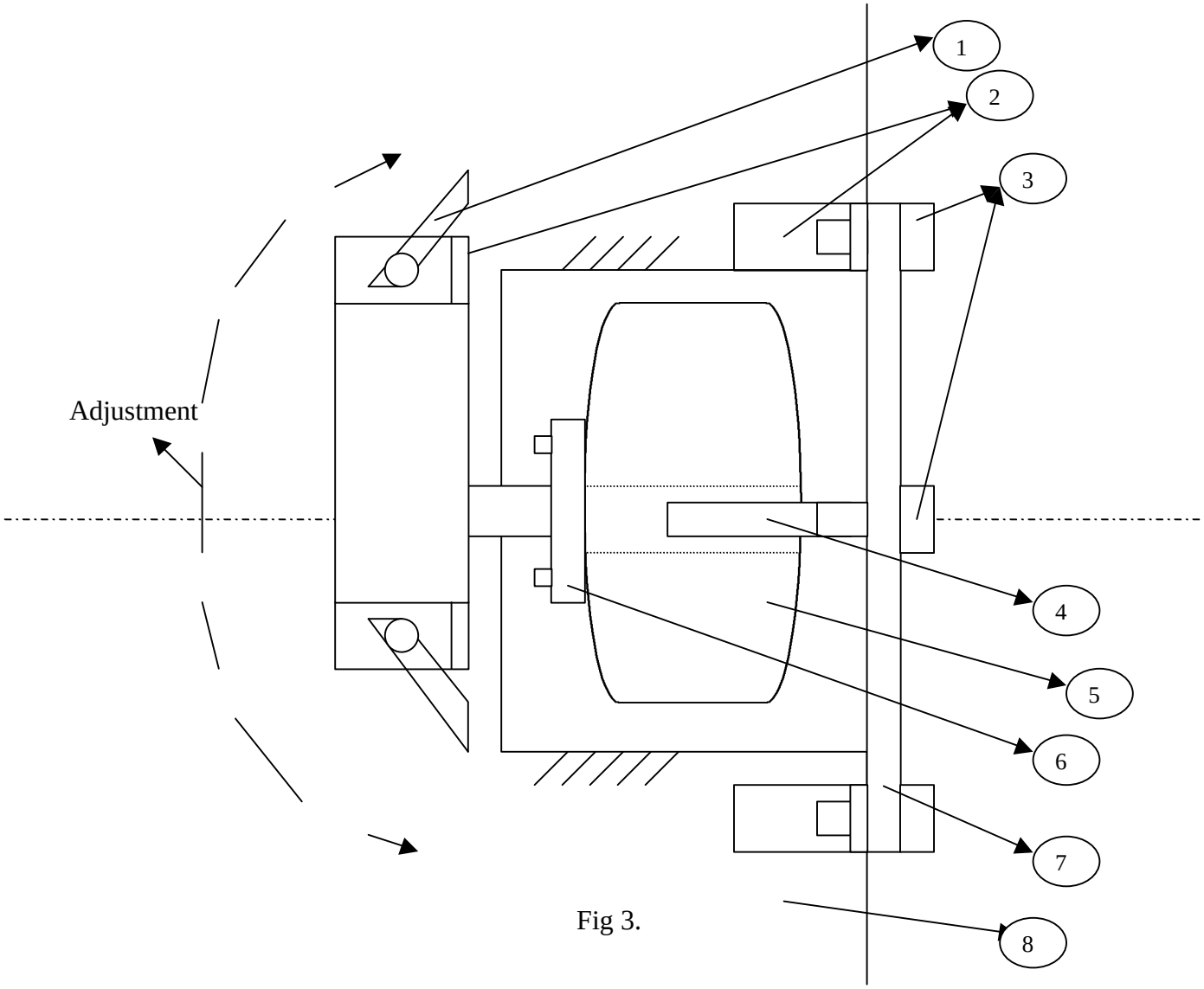


Fig 3.

For fig 3.

1. Slot made for adjusting the angle of the wheel
2. 1 inch aluminium angle plates to used to prop up motors, supports etc
3. Machine screws of size 6 and length one inch
4. Hollow copper rod with outer diameter 1/8th inch. Fits snuggly inside wheel hub and the screw fits inside the copper rod.
5. 3 inch dia wheel
6. Circular coupling that comes with the servos. Coupled with machine screws of size 3
7. Wooden plate with dimensions 1 inch by 5 inches spanning the side of the wheel
8. Platform

ACTUATION

Three separate actuators were used on DomiGnome and are discussed below in detail.

Drives:

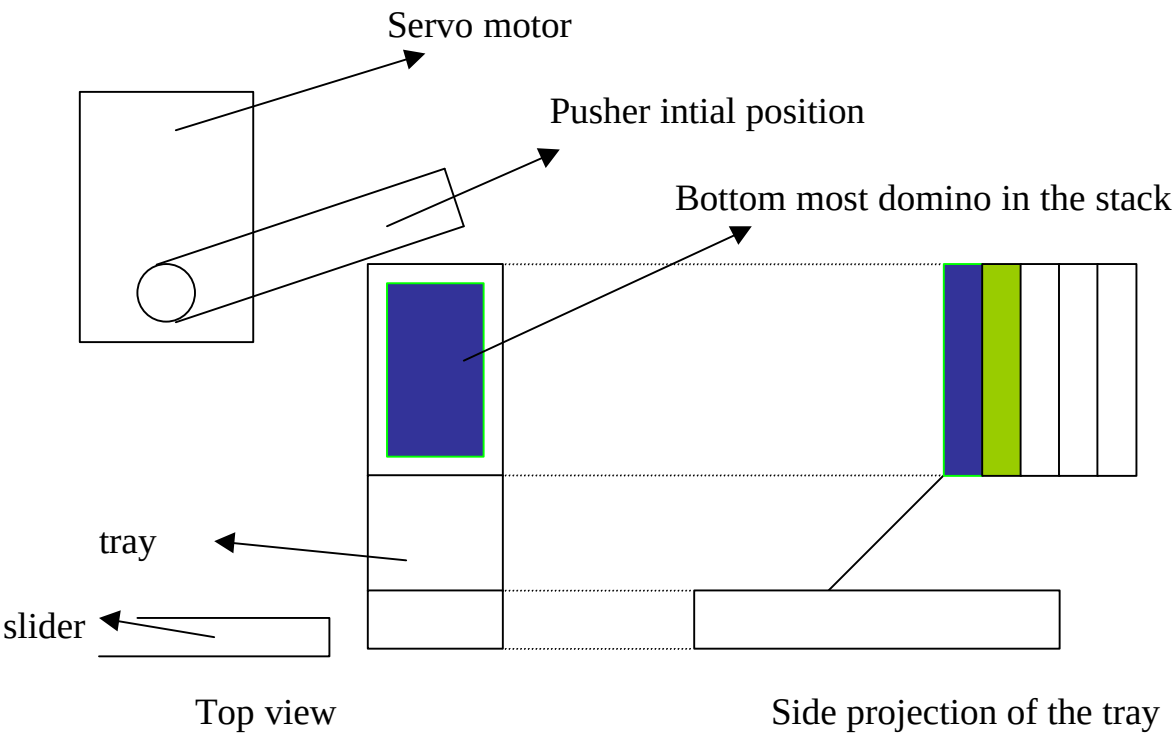
The robot is driven through a single differential axle powered by two fully hacked servos with a torque rating of 42 ounce-inches. The motors were made to slow down in the program that I wrote as the wheels would tend to skid on full power. This was done by looping through the drive command and a sleep command for a number of steps.

The motion of the robot was intermittant, that is, it would move about ½ an inch forward and wait a second for the dispenser to set a domino before it moved another ½ an inch. This can be seen in the function move_me( ).

This drive system lacked any kind of precision and could not be used to follow an absolutely straight path. A better option for a drive system for a characteristically slow robot with intermittant motion such as DomiGnome would be one with stepper motors with a suitable gear reduction to compensate for the lack of driving torque in stepper motors.

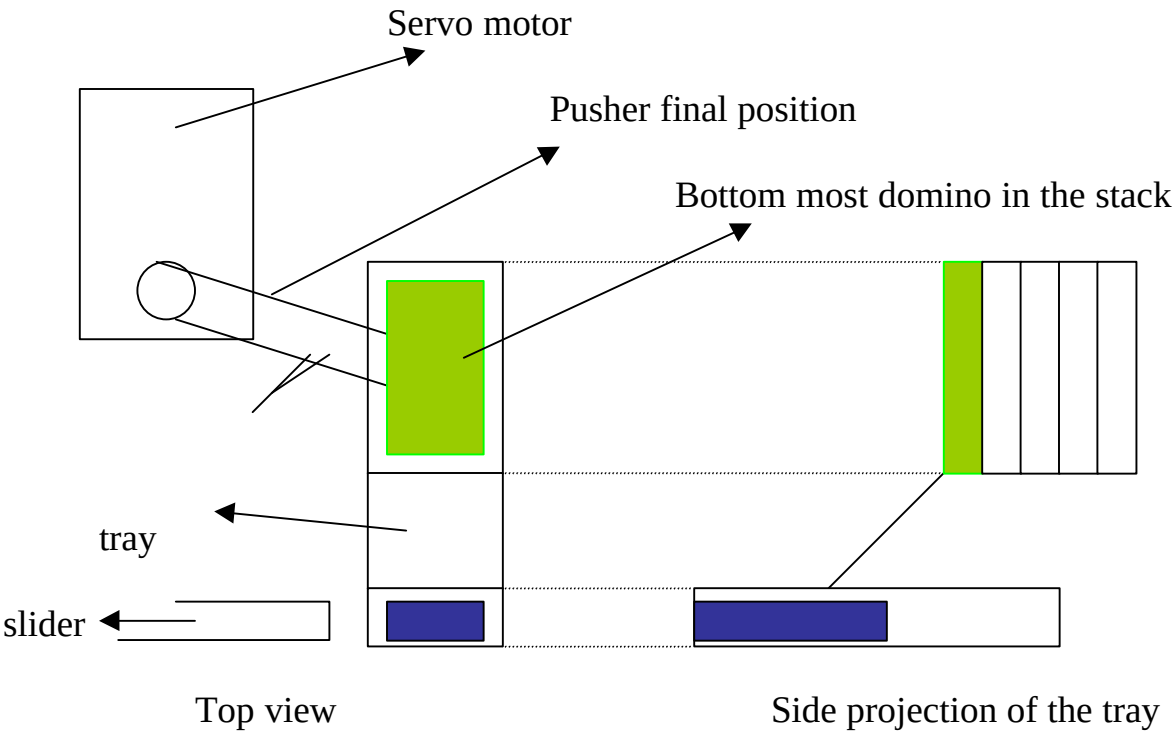
Dispenser:

The dispenser uses two servos, one to push the dominoes out of the stack from the bottom of the stack, and the other to slide the domino out into the open. Both servos had a rating of 42 ounce-inches and worked in synchronisation. The actuation of the two servos is shown in fig4 & 5.



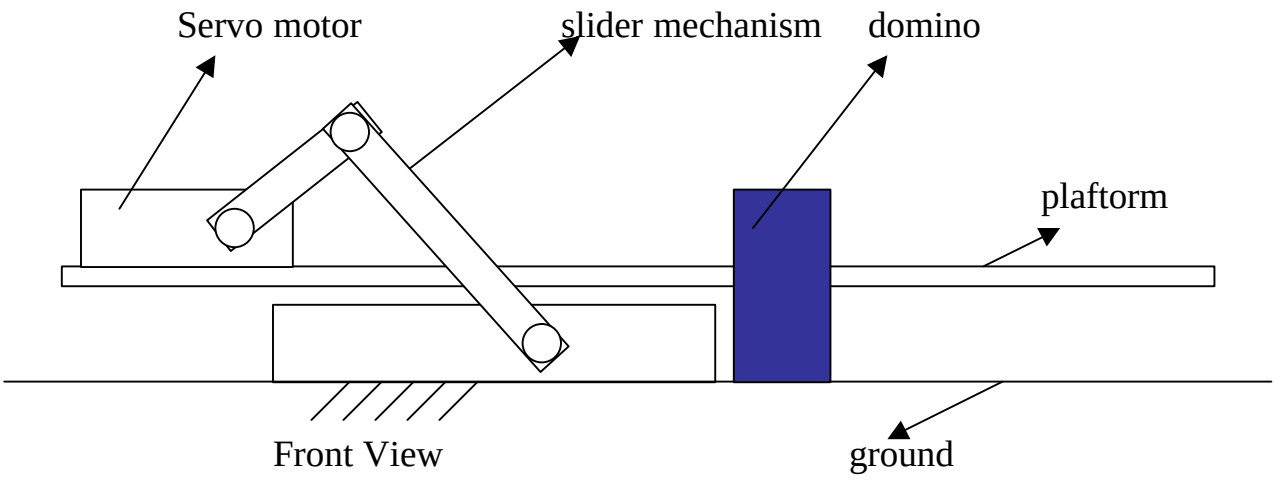
STEP 1

Fig. 4a.



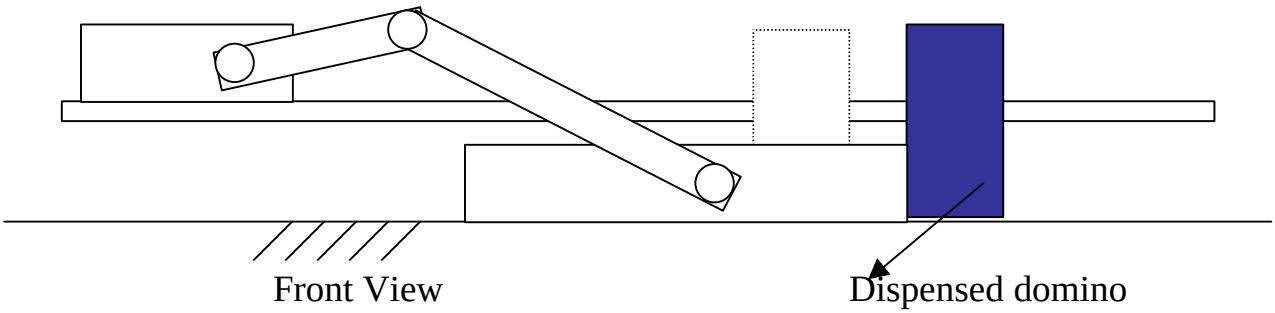
STEP 2

Fig. 4b



STEP 1

Fig. 5a



STEP 2

Fig. 5b

As seen in the above figures, the dominoes are first pushed oout of the stack from the bottom and then slid out into the open with the help of the slider mechanism. The synchronization of these actuators is as shown below

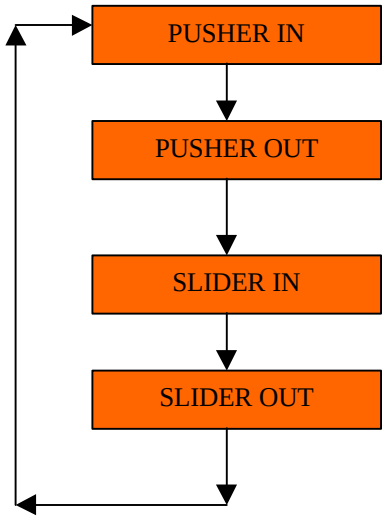


Fig. 6

The dimension of the dominoes were 1 3/8th “ by 5/8th “ by 1/4th “ . the weight of each domino is under an ounce and therefore the mechanism had to be carefully caliberated in oder to apply the right amount of force so that the dominoes don’t

get thrown out or bounce out of the dispenser. For this, the servos had to be slowed down in a similar manner by which thewheels were slowed down. This routine used to slow down the servos can be seem in the functions move_slider_fwd( ), move_pusher_fwd( ) in the C program at the end of this paper.

SENSORS

Infrared Emitter/Detector Pairs:

Infrared Light Emitting Diodes (LEDs) emit infrared light to illuminate objects nearby (modulated at 40KHz). The detector used is a Sharp GP1U58Y, which only detects infrared light modulated at a frequency of 40Khz. This sensor contains an infrared sensor, amplifier, 40Khz filrt, and a comparator. Once the sensor has been modified, it outputs an analog signal corresponding to the intensity of infrared light emitted. DomiGnome is equipped with three of these sensors in the front of the platform and one at the rear. The sensors in front are used for obstacle detection and the one at the back is hooked up to the stack and is used to detect if the stack has run out of dominoes.

Since the robot moves only in the forward direction while dispensing dominoes, the sensors are not required on the sides or the rear of the vehicle for obstacle detection.

Bump Sensors:

Two bump sensors have been used in front of the vehicle to detect obstacle collision. If the vehicle bumps int anything, the bump switches will be shorted and a value of zero is sent to the corresponding analog port which the sensor is hooked onto. Again, these are only required in the front of the vehicle as the vehicle is not likely to bump into anything from the side or the rear.

BEHAVIORS

DomiGnome has been programmed to have four distinct behaviors. They are,

1. Dispensing of dominoes as it moves along a pre-defined path
2. Obstacle detection
3. Detection of the completion of dominoes in the stack
4. Setting off the chain reaction at the end of the pre-programmed path.

All these behaviors are discussed in detail blow.

DomiGnome has been programmed to dispense dominoes in a path that can be set by the user. The robot understands the path in terms of an array of numbers that it reads one after the other. The numbres maybe either a 0, 1, 2, 3 or 5. If the program reads the value 0, the robot will advance 4 steps in the forward direction. If it reads the value 1, it advances 4 steps while turning right at the same time. The same happens when it reads a value of 2 but it turns left. When the robot reads a value of three, it is programmed to leave a gap of about 3 inches. Intially, this was incorporated as a safety feature but later I found that there was no need of this feature as a safety feature as the robot would never set off the chain reaction while it was dispensing dominoes. I decided to keep the feature though as some people set interesting objects like ladders and toy cars etc in the domino chain to enhance the chain reaction and to make it look more fun. The robot robot recognises the end of the path when it reads the value 5.

The path below is a sample path that DomiGnome followed while it read the array below.

Path_array{0,1,1,0,2,5}

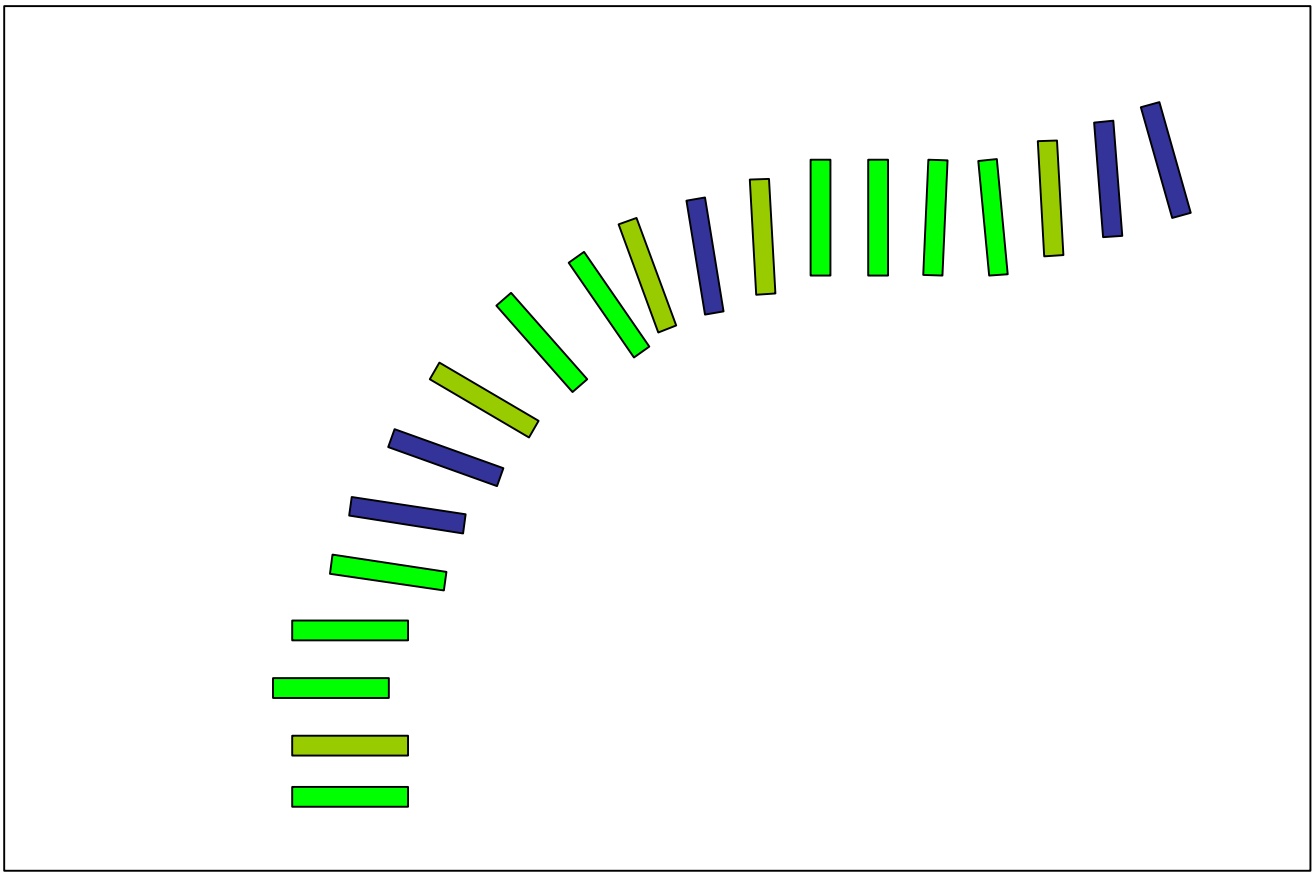


Fig. 7

The above fig. Shows the sample path from the array path_array.

The robot has been programmed to detect obstacles in its path and to stop when it senses one. It resumes its task once the obstacle has been removed. No affort has been made to avoid or circumnavigate the obstacle as this would defeat the purpose of having a pre-programmed path. If domiGnome senses that it has run out of dominoes in the stack, it stops for about 4 minutes which is the reloading timet hat I have given it. It resumes its task after 4 minutes if the stack has been filled. If not, it waits another 4 minutes.

At the end of the path, the robot moves a little forward, turns around and tips the last domino that it set in order to start the chain reaction. this feature was added to give the robot a an autonomous look where it can be seen as playing with dominoes on its own.

EXPERIMENTAL LAYOUT AND RESULTS

Dispenser Calibration:

The dispenser had to be adjusted for dimensional toleracne and applied force. There were many dimensions that played an important role in the dispensing of only one domino at a time and to get the dominoes to stand upright. These tolerances were initially incorporated in the design but still needed adjustments at the end to compensate for errors in manufacturing the parts. The adjustments were done with a trial and error technique till the dominoes were being dispensed in a flawless manner. The breakdown design mentioned earlier helped enormously in making the fine adjustments. Whereever a fine adjustment was percieved, I put slots instead of holes for the machine screws so that they could be moved to the right posistion and then fastened without much trouble. The use of aluminium for the tray helped in getting the bend angle that helps set the dominoes upright. Since aluminium is flexible, I could easily adjust this angle till I got the optimum angle where the dominoes would slide down the tray without any hitches.
The force with which the pusher and the slider mechanism push the dominoes out had to be controled and this was done in the C program by slowing down the servos.

Other experimants that were done on the robot include wheel alignment that was explained in the mobile platform and choosing of the ratio of the length of the two slider links. The ratio was chosen such that the torque required to push the slider forward was minimum. Again this was done by trial and error and was gauged by the amount the slider would push a domino out when the servo was operated at full torque.

OTHER EXPERIMENTAL WORK

After I designed the dispenser that I used on DomiGnome, I designed another type of dispenser that would work on continuous motion rather than the intermittant motion that I described earlier. On some manual tests, I found this design to be quite satisfactory but later found it was fairly difficult to implement as the critical parts- the helical coils as shown below had to be self made to the right dimensions and this wasn't easy. This design could be taken up as future work by anyone for a similar kind of a dispenser. The design is as exaplained below.

This design consists of a spring type dispenser which is powered by a 40 ounce-inch hacked servomotor. The dispenser consists of two helical coils, one turning clockwise and the other anti-clockwise in a synchronized manner. The rotary motion of the helical coils is used to translate the dominoes that are placed between the two helical coils. The assembly of this sensor is as shown in the figure in Appendix B. Support rods are provided on either side to provide side support for the dominoes and to ensure proper directionality. A single motor through the gearing shown drives both springs. This enables proper synchronization of the two coils.

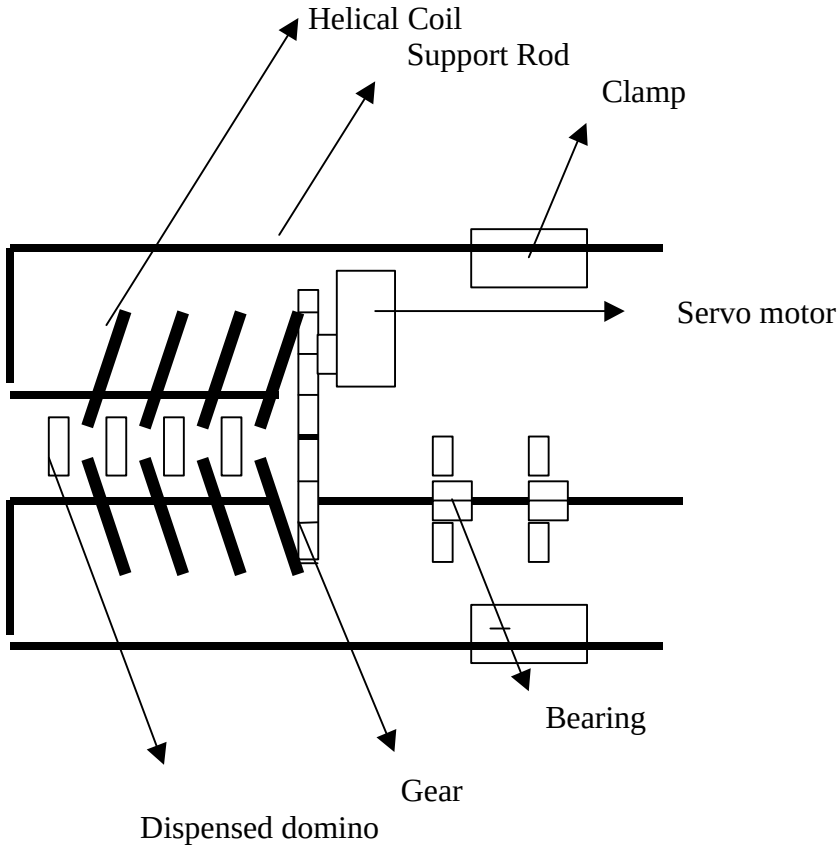

Helical Coil
Support Rod
Clamp
Servo motor
Bearing
Gear
Dispensed domino

Fig. 8

CONCLUSION

Summary of work:

The following objectives were accomplished during the development of GomiGnome-

1) Autonomously dispense dominoes according to a pre-programmed path
2) Interaction with the environment through various sesnosrs, actuators and behaviors

The only shortcoming in DomiGnome is the lct of accuracy in following a path. The use of stepper motors to drive the wheels would have eliminted this problem. as a further improvement, the helical coil dispenser could be implemented.

Technical caveats:

Since its conception, DoniGnome has changed considerably. The original design was that of a four wheeled platform with front wheel steering and rear wheel drive. This would have been a good idea from the point of view of path tracking using regular hacked servos for drives, but, due to the lack of availability of parts for a four wheeled platform, I had to abandon the idea to keep up with the time constraint that this project was bound by.

Future Work:

If I were to start again, I would choose a 4 wheeled platform with front wheel steering and rear wheel drive powered by hacked 42 ounce-inch servos. This would eliminate the limitations of stepper motors and would also give enough directionality for the purpose of dispensing dominoes. I found the dispenser that I made extremely efficient and would stick by it unless I needed a better dispensing rate in which case, I would implement the coil type dispenser.

REFERENCES

Jantz, Scott, Sharp Sensor Modification, Intelligent Machines Design Lab notes, Department of Electrical Engineering, University of Florida. (1996)

Martin Fred, the 6.270 Robot Builder's guide, The Media Lab, MIT, (1999)

Brain Kernighan and Ritchie Dennis, The C Programming Language, (1997)

Bevan Thomas, Theory of Machines, (1992)

APPENDIX

Loaded Libraries:

LIB_RW11.C

TWOSERVO.ICB

TWOSERVO.C

DOMIGNOME.C

Domignome.c

```
/*********************************************************/
/*   Programmer:         Arfath Pasha                    */
/*   Class:              EEL-5666 - Mobile Robots        */
/*   Date:               April 15, 1999                  */
/*   Purpose:       To drive DomiGnome                   */
/*********************************************************/

/*  DomiGnome's Parameters */
float SLIDER_OUT= 151.0, SLIDER_IN= 95.0;
float PUSHER_IN= 50.5, PUSHER_OUT=130.0;

/* DomiGnome's Path */
int PATH[] = {0,0,1,1,1,0,0,0,2,2,2,0,0,0,0,5,5,5};


void main()
{
   int i = 0, j = 0, k = 1;

   /* Initialization */
   sleep(0.5);
   servo_on();
   servo_deg1(SLIDER_OUT);
   servo_deg2(PUSHER_OUT);
   poke(0x7000, 0xff);

   while(k == 1)
   {
      check_sensor();
```

```
check_stack();

if( PATH[i] == 0)   /* move straight for 4 steps */
{
   for(j=0; j<4; ++j)
   {
      check_sensor();
      check_stack();

      move_pusher_fwd();
      wait(100);
      move_pusher_bkwd();
      wait(250);
      move_slider_fwd();
      wait(100);
      move_slider_bkwd();
      wait(100);
      move_me();
      stop_me();
      wait(100);
   }
}
else if( PATH[i]== 1)   /* move right for 4 steps */
{
   for(j=0; j<4; ++j)
   {
      check_sensor();
      check_stack();

      move_pusher_fwd();
      wait(100);
      move_pusher_bkwd();
      wait(250);
      move_slider_fwd();
      wait(100);
      move_slider_bkwd();
      wait(100);
      move_me_left();
      stop_me();
      wait(100);
   }
}

else if( PATH[i]== 2)   /* move left for 4 steps*/
{
   for(j=0; j<4; ++j)
   {
      check_sensor();
      check_stack();
```

```
            move_pusher_fwd();
            wait(100);
            move_pusher_bkwd();
            wait(250);
            move_slider_fwd();
            wait(100);
            move_slider_bkwd();
            wait(100);
            move_me_right();
            stop_me();
            wait(100);
         }
      }
      else if( PATH[i]== 3)      /* leave a gap of two inches */
      {
         for(j=0; j<4; ++j)
         {
            check_sensor();
            check_stack();

            move_pusher_fwd();
            wait(100);
            move_pusher_bkwd();
            wait(2000);
            move_slider_fwd();
            wait(100);
            move_slider_bkwd();
            wait(100);
            move_gap();
            stop_me();
            wait(100);
         }
      }
      else
      {
         dance();
         k = 0;
      }

      ++i;
   }

   stop_me();
   servo_deg1(SLIDER_OUT);
   servo_deg2(PUSHER_OUT);
}
/* end of main */
```

```
void wait(int milli_seconds)
{
   long timer_a;

   timer_a= mseconds() + (long) milli_seconds;
   while( timer_a > mseconds() )
   {
      defer();
   }
}


void check_sensor()
{
   int k = 1;
   int B, C, D, E, F, G;

   while(k == 1)
   {
      /* IR sensors */
      C = analog(2);
      D = analog(4);
      E = analog(5);

      /* Bump sensors */
      F = analog(3);
      G = analog(7);

      if( (C > 100) || (D > 100) || (E > 100) || (F < 100) || (G
< 100))
         {
           stop_me();
      }
      else
      {
        k=0;
      }

   }
}


void check_stack()
{
   int k = 1;
   int A;
```

```
   A = analog(0);  /* stack sensor */

   while (k == 1)
   {
      if( A > 128)
      {
        stop_me();
      }
      else
      {
        k = 0;
      }
   }
}


void move_me()
{
   int i;

   while( i < 6)
   {
      motor(0,100.0);
      motor(1,100.0);
      wait(15);          /* speed control (directly
proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }
}

void move_gap()
{
   int i;

   while( i < 100)
   {
      motor(0,20.0);
      motor(1,20.0);
      wait(5);      /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }
}

void move_me_left()
{
```

```
   int i;

   while( i < 30)
   {
      motor(0,100.0);
      motor(1,100.0);
      wait(5);      /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }

   while( i < 30)
   {
      motor(0,0.0);
      motor(1,100.0);
      wait(5);      /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }
}

void move_me_right()
{
   int i;
   while( i < 30)
   {
      motor(1,100.0);
      motor(0,100.0);
      wait(8);      /* speed control (directly proportional) */
      motor(1,0.0);
      motor(0,0.0);
      ++i;
   }


   while( i < 30)
   {
      motor(1,100.0);
      wait(8);      /* speed control (directly proportional) */
      motor(1,0.0);

      ++i;
   }
}


void dance()
```

```
{
   int i=0;
   while( i < 30)
   {
      motor(0,100.0);
      motor(1,100.0);
      wait(20);     /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }

   wait(100);
   i=0;

   while( i < 30)
   {
      motor(0,-20.0);
      motor(1,20.0);
      wait(20);     /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }

   wait(100);
   i=0;

   while( i < 30)
   {
      motor(0,20.0);
      motor(1,-20.0);
      wait(20);     /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
   }

   wait(100);
   i=0;

   while( i < 70)
   {
      motor(0,-20.0);
      motor(1, 20.0);
      wait(20);     /* speed control (directly proportional) */
      motor(0,0.0);
      motor(1,0.0);
      ++i;
```

```
   }
}
void stop_me()
{
   motor(0,0.0);
   motor(1,0.0);
   wait(1000);
}


void move_slider_fwd()
{
   float slider_in = SLIDER_IN, slider_out = SLIDER_OUT;

   while(slider_out > slider_in)
   {
      slider_out = slider_out - 0.5;
      servo_deg1(slider_out);
      wait(15);
   }
}


void move_slider_bkwd()
{
   float slider_out = SLIDER_OUT;

   servo_deg1(slider_out);
}


void move_pusher_fwd()
{
   float pusher_out = PUSHER_OUT, pusher_in = PUSHER_IN;

   while(pusher_out > pusher_in)
   {
      pusher_out= pusher_out - 5.0;
      servo_deg2(pusher_out);
      wait(15);
   }
}

void move_pusher_bkwd()
{
   float pusher_out = PUSHER_OUT;

   servo_deg2(pusher_out);
}
```

Vendor Information:

IR emitters/detectors:

LED emitters	Mekatronics 316 NW 17th St. , Suite A Gainesville, FL 32603 (407) 672 6780 http://www.mekatronics.com (purchased from Scott Jantz) $0.75 each
Detectors	Sharp GPIU58Y via Mekatronics (Purchased from Scott Jantz) $ 3.00 each

Bump switches:

Mekatronics
(Purchased from Scott Jantz)
$0.75 each

Dispensing unit:

Servo motor	Mekatronics (Purchased from Scott Jantz) 42 ounce inch torque $11.0
Brackets, fasteners	Zell's Ace Hardware 3727 W University Avenue Gainesville FL. 352 378 4650
Dominoes	www.ebay.com Game: Domino Rally $20 (including shipping)