University of Florida
Department of Electrical and Computer Engineering
EEL 5666
Intelligent Machines Design Laboratory

MOCTAR

Final Design Report

Justin McCollum
12/12/97

Instructor: Keith L. Doty
TA: Scott Jantz

Table of Contents

Abstract

MOCTAR is an autonomous robot. The robot uses a four independent wheel drive system to propel itself around in search of items to collect. The steering of the platform is controlled by a servo located under the collection bin. The location of the collection bin is in the rear. Therefore, the MOCTAR also has rear wheel steering. It consists of eight servo/motors and uses twenty-two AA size rechargeable batteries. The sensor array includes several mechanical type sensors. These include: bump switches, IR detectors (both a 32KHz and a 40KHz), a mechanical lift, and a mechanical grip. Behaviors which are included in its programming are obstacle avoidance, object detection, object gathering, object storage. Fundamental experiments done for MOCTAR have ranged form testing spring constants to finding the right resistor and capacitor values for creating a 32KHz signal. MOCTAR is an autonomous robot who purpose is to collect objects.

Executive Summary

MOCTAR, Mobile Object Collector and Transport Autonomous Robot, was designed using a Cad utility and is cut out of wood. It was created, constructed, wired, and programmed in three and a half months. The idea for this robot came form a general want for the robot to be useful. It was then drafted and cut out of wood. Specialty parts were then ordered. Once enough of the parts had been cut or arrived, construction began on MOCTAR. After finishing the structural aspect of the robot , the electrical components were added. After adding some of the electrical components, teasing on certain parts of the robot were constructed. This included the spring forces on the clamp, the steering mechanism, and the lowering mechanism.

These test proved that the overall concept of MOCTAR could be accomplished. However, some of the original ideas had to be changed due to unaccounted for forces, structural limitations, and the amount of time and money to spend. The final prototype is lined with a large number of wires. Future mockups of this robot would have fewer wires and look more appealing to the eye.

Introduction

MOCTAR, Mobile Object Collector and Transport Autonomous Robot, collects and stores reasonable sized objects. This Autonomous robot completes its objective using sensors to interact with its environment. The way in which MOCTAR reacts to a situation is determined by programmed behavior routines which will guide, but not direct (control), the robot in the completion of its goal. The robot's design enables it to grab, lift, and place the objects in an on-board bin. This report describes the integrated systems, mobile platform, methods of actuation, sensors, and behaviors of this robot.

Integrated System

The MC68HC11 micro-controller with 32k of SRAM is the housing and processing center for all of MOCTAR'S functions. These functions are programmed using Interactive C. This allows for easy debug of both sensory and behavior routines. Two micro-controllers are needed to meet the requirements of this robot. One is mounted on the EVBU board, the micro-controller which houses and executes the main functions, and the other on a MSCC11 board. The one affixed on the MSCC11 will control the robot's motors and servos. The two controllers communicate through a SPI (Serial Peripheral Interface) connection. These boards are attached to the top platform of the mobile platform for easy maintenance. MOCTAR uses several senses to navigate in its environment. This feed back is then run into the micro-controller, where it is processed and then gives a response.

Mobile Platform

The robot's is designed as a front open belly truck. The structure consists of two main static platforms relative to the its wheels. The top one holds the controller boards. Unlike the upper platform, the lower is cut in the shape of a "U". This shape allows the vehicle to roll over its detected object while enabling the back to be dedicated for a steering. Between the two platforms is the object acquisition mechanics. The platform also has extended front wheel to match the steering ones in the back. This also gave an addition benefit; more room. The open pockets in the sides and top allow for easy maintenance and repair. The platform houses the eight servo/motors used to run MOCTAR, which brings us to our next topic.

Actuation

Four motors and four servos (three of which are partially hacked) are constructed into MOCTAR. Two of the four motors are use as the front two drive motors. The other two can be located it the claw mechanism. One of the hacked servos is used with a gear-head that acts like the rack in a rack and pinion system. The other two hacked servos control object movement form the back making MOCTAR four wheel drive. The two motors that drive the claw mechanism use string instead of other devices for two reasons. One, it is cheap, and the second is all of the forces are in tension, not compression. The actuation of the lift by ball screw was a bad concept. There was too much force do to moments for this to work in the configuration I needed. This concludes the description of the mechanical actuation.

Sensors

The robot uses different four sensors. MOCTAR'S IR sensors detect obstacles for avoidance and target acquisition. The IR for the object detection are 32KHz while the obstacle avoidance are 40KHz. The 32KHz frequency signal was created using a 555 timer. Bump triggers are attached to the front only. The reason for only two bump sensors is that there are two more IR detectors placed on the back side to detect obstacle behind MOCTAR. Cadmium sulfide cells were scraped from my project do to the lack of time. Wheel encoders were also scrapped since they were not necessary on this robot. The circuit used for the 555 timer can be found in an engineering note book sold by Radio Shack. The "new", fourth sensor was my mechanical claw. The spring constant graphs and structure can be seen under the Experimental layout and results category.

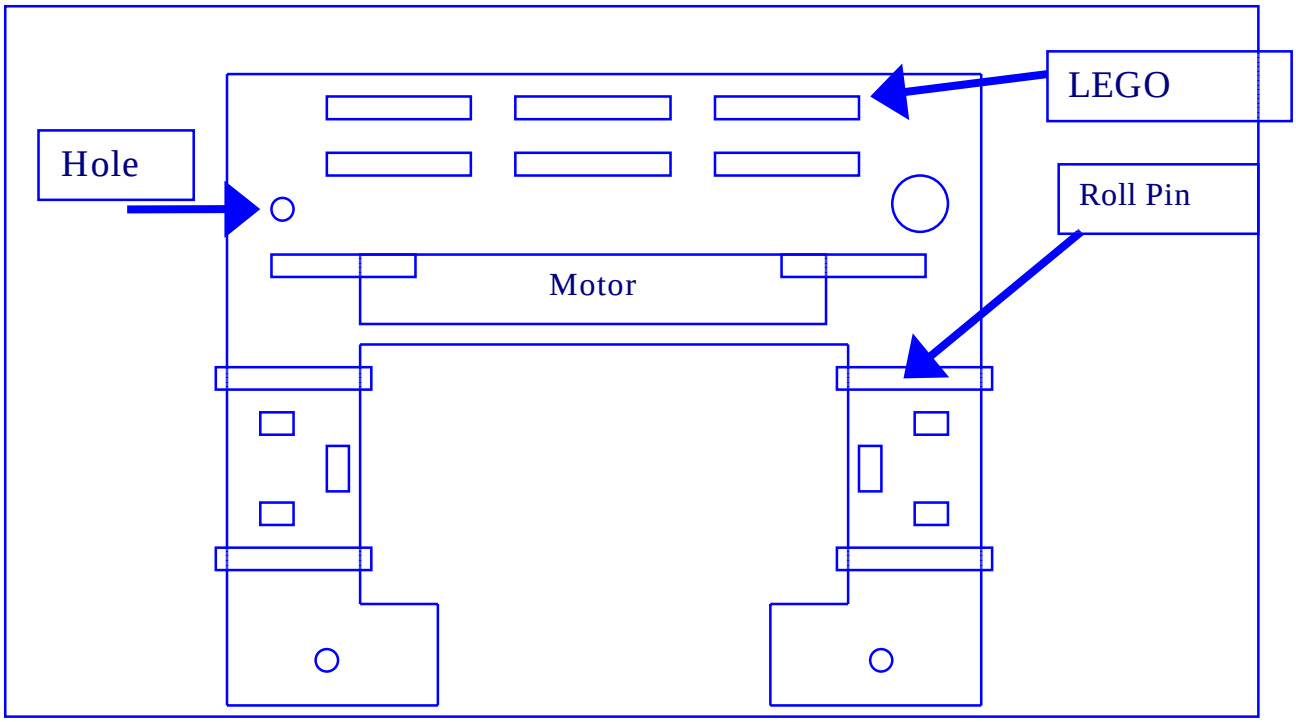
Behaviors

Each sensor has at least one corresponding behavior. The main behavior designates a search for objects to be collected. This process is ongoing until a certain number of objects have been collected. However, while the search behavior is running, other smaller behaviors can arise, such as MOCTAR'S obstacle avoidance routine. The actual avoidance behavior is mixed in with the object detect behavior. Other behaviors include the clamp and storage behavior. This behavior reacts with the internal environment of MOCTAR. All of the code was written in IC. The SPI routines were written by Purvesh Thakker. The behavioral code will be put in the appendix. All of these behaviors rely on how well the robot senses its environment both internally and externally.

Experimental Layout and Results

There were several experiments need to determine how much MOCTAR'S clamp could hold. To determine this, I used a specified amount of weight to pull against the force

springs. This allowed me to find the spring constant, and thus the pressure exerted on an object at any distance. Figure 1. below shows a rough sketch of the claw and the determined graph of the spring constant. The other major experiment was to determine the resistor values for the 32Khz frequency generator. This was done by using a potentiometer and a known value resistance.



F= -kx ; known: F= 0.3875 lb. , x= .125 in.
So, k= F / x = 3.1

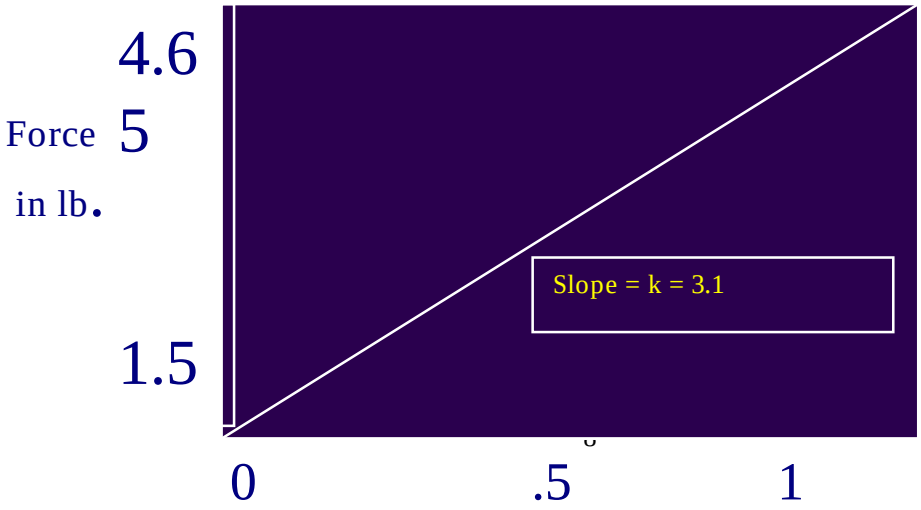


Figure 1.

Conclusion

Throughout this project there are many experiences that caused me to learn and grow as an engineer. Not just the electrical side of the project was challenging, but the mechanical as well. The overall design of MOCTAR is sound, but the complexity of it made it very difficult to do all the things I wanted. I had to sacrifice a lot of the electrical additions due to the time constraint. However, during the course of the semester, I learned AutoCAD, the 555 timer, and IC. All of these experiences made up an interesting robot: MOCTAR.

Appendix

MOCTAR'S IC CODE:

```
float angle;
int direction, i, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10;
int left_front_ir, right_front_ir, left_rear_ir, right_rear_ir;
int object_left_ir, object_right_ir;
float steering_angle, fs, fm, ms, mm, ss, sm;
int bumped1, bumped2,z;

void main()
      {
      init_spi();
        zero_servo();
        inital_all();
        start_process(obs_locate());
        }

void irsense()
{
poke(0x7000,0xff);
poke(0x6000,0xff);
left_front_ir = analog(3);
right_front_ir = analog(2);
left_rear_ir = analog(6);
right_rear_ir = analog(7);
object_left_ir = analog(1);
object_right_ir = analog(0);
}


void obs_locate()
{
while (1)
        {
        straight();
        forward(fs,fm);

        if ((peek(0x5000) & 0b00000010) > 0)
        bumped1 = 1;
        else bumped1 =0;
        if ((peek(0x5000) & 0b10000000) > 0)
        bumped2 = 1;
        else bumped2 =0;

        reset_coms();
        z = 0;
        if (bumped1 == 0)
                {
```

```
                stop_moctar();
                reverse(fs,fm);
                while (z < 3000)
                        {
                        z=z+1;
                        if ((analog(6)>109) || (analog(7) >109))
                        stop_moctar();

                        }
                        left_turn(3);
                        forward(fs,fm);

                }

        else if (bumped2 == 0)
                        {
                        stop_moctar();
                        reverse(fs,fm);
                        while (z < 3000)
                        {
                        z=z+1;

                        if ((analog(6)>109) || (analog(7) >109))
                        stop_moctar();

                        }
                        right_turn(3);
                        forward(fs,fm);

                }
sleep(3.0);
straight();
forward(fs,fm);

irsense();

if ((left_front_ir > 100) && (right_front_ir > 100))
{
z=0;
stop_moctar();
reverse(fs,fm);
while (z < 3000)
{
z=z+1;
if ((analog(6)>109) || (analog(7) >109))
stop_moctar();
}
right_turn(3);
forward(fs,fm);
}
sleep(5.0);
straight();

if (left_front_ir > 95)
 {
 stop_moctar();
 right_turn(3);
 forward(fs,fm);
 }
else if (left_front_ir > 95)
        {
```

```
 stop_moctar();
 left_turn(3);
 forward(fs,fm);
    }
if (object_left_ir > 95)
{

while (object_right_ir <92)
{
left_turn(3);
motor(0,0.0);
}
claw_routine();
}
else if (object_right_ir > 95)
{

while (object_left_ir <92)
{
right_turn(3);
motor(0,0.0);
}
claw_routine();
}

straight();
forward(fs,fm);
}
}



void inital_all()
{
fs=175.0;
fm=50.0;
ms=150.0;
mm=65.0;
ss=125.0;
sm=45.0;
reset_coms();
zero_servo();
reset_coms();
zero_motors();
reset_coms();
reset_claw();
reset_coms();
}

void zero_motors()
{
set_motor(7,0,0);
set_motor(6,0,0);
reset_coms();
motor(0,0.0);
motor(1,0.0);
}

void reset_claw()
{
        if ((peek(0x4000) & 0b00010000) > 0)
```

```
                s5 = 1;
        else s5=0;
        while (s5 == 1)
        {
                set_motor(7,150,0);
                reset_coms();
                if ((peek(0x4000) & 0b00010000) > 0)
                        s5 = 1;
                else s5=0;
        }
        set_motor(7,0,0);
        reset_coms();

        if ((peek(0x4000) & 0b00000001) > 0)
                s1 = 1;
        else s1 = 0;
        while (s1 == 1)
        {
        set_servo(4,75.);
        reset_coms();
        if ((peek(0x4000) & 0b00000001) > 0)
                s1 = 1;
        else s1 = 0;
        }
        set_servo(4,59.8);
        reset_coms();

        if ((peek(0x4000) & 0b00000100) > 0)
                s3 = 1;
        else s3 = 0;
        while (s3 == 1)
        {
        set_motor(6,150,0);
        reset_coms();
        if ((peek(0x4000) & 0b00000100) > 0)
                s3 = 1;
        else s3 = 0;
        }
        set_motor(6,0,0);
        reset_coms();
}




void zero_servo()
      {
        set_servo(3,54.);
        set_servo(2,71.2);
        set_servo(5,93.7);
        set_servo(4,59.8);
        steering_angle = 93.7;
      reset_coms();
      defer();
      }

void right_turn(int sharpness)
        {
```

```
        float x;
        reset_coms();
        if (sharpness == 3)         /* note 3 is shapest */
        for (x = steering_angle; x < 125.0; x++)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        else if (sharpness == 1)
                for (x = steering_angle; x < 104.1; x++)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        else if (sharpness == 2)
                for (x = steering_angle; x < 114.6; x++)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        steering_angle = x;
        }

void straight()
        {
        float x;
        reset_coms();
        if (steering_angle < 93.7)         /* note 3 is shapest */
        for (x = steering_angle; x < 93.7; x = x+1.0)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        else if (steering_angle > 93.7)         /* note 3 is shapest */
        for (x = steering_angle; x > 93.7; x=x-1.0)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        steering_angle = x;

        }

void left_turn(int sharpness)
        {
        float x;
        reset_coms();
        if (sharpness == 3)         /* note 3 is shapest */
        for (x = steering_angle; x > 62.4; x=x - 1.0)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        else if (sharpness == 1)
                for (x = steering_angle; x > 83.3; x=x-1.0)
```

```
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        else if (sharpness == 2)
                for (x = steering_angle; x > 72.8;x= x-1.0)
                {
                set_servo(5,x);
                msleep(20L);
                reset_coms();
                }
        steering_angle = x;
        }


void set_servo(int x, float z)
      {
      int y;

      hog_processor();
      y=degree_to_pulse(z);
      out_spi(x);
      out_spi( y>>8 );
      out_spi( y&0xff );
      out_spi(x+(y>>8)+(y&0xff));
      }

void set_motor(int num, int speed, int dir)
      {
      int temp;
      int temp2;

      hog_processor();
      temp=num+40;
      if (dir==1) temp=temp+8;
      out_spi(temp);
      out_spi(256 - speed);
      if (num == 1) temp2=1;
      if (num == 2) temp2=2;
      if (num == 3) temp2=4;
      if (num == 4) temp2=8;
      if (num == 5) temp2=16;
      if (num == 6) temp2=32;
      if (num == 7) temp2=64;
      if (num == 8) temp2=128;
      out_spi(temp2);
      out_spi(temp+256-speed+temp2);
      }

void reset_coms()
      {
      out_spi(0);
      out_spi(0);
      out_spi(0);
      }

void forward(float angle, float mot_speed)
      {
                stop_moctar();
                reset_coms();
```

```
                set_servo(2, 0. + angle);
                set_servo(3, 180. - angle);
                motor(0, -mot_speed);
                motor(1,  mot_speed);
        }

void stop_moctar()
        {
                reset_coms();
                set_servo(3,54.);
                set_servo(2,71.2);
                motor(0,0.0);
                motor(1,0.0);
        }

void reverse(float angle,float mot_speed)
        {
                stop_moctar();
                reset_coms();
                motor(0,  -mot_speed);
                motor(1, -mot_speed);
                set_servo(2, 180. - angle);
                set_servo(3, 0. + angle);
        }

void open_claw()
{
reset_coms();
if ((peek(0x4000) & 0b01000000) > 0)
s7 = 1;
else s7 = 0;
while(s7 == 1)
{
reset_coms();
set_motor(7,250,1);
if ((peek(0x4000) & 0b01000000) > 0)
s7 = 1;
else s7 = 0;
}
set_motor(7,0,1);
reset_coms();
}

void forward_find_claw()
{

int irsensor1_value, irsensor2_value;
poke(0x7000,0x00);
reset_coms();
straight();
while(irsensor1_value < 100)
{
poke(0x7000,0xff);
irsensor1_value = analog(4);
poke(0x7000,0x00);
forward(fs,fm);
}
poke(0x7000,0x00);
while(irsensor2_value < 100)
{
poke(0x7000,0xff);
```

```
irsensor2_value = analog(5);
poke(0x7000,0x00);
forward(fs,fm);
}
stop_moctar();
}

void close_claw()
{
reset_coms();
if ((peek(0x4000) & 0b00010000) > 0 )
s5=1;
else s5=0;
if ((peek(0x4000) & 0b00100000) > 0 )
s6=1;
else s6=0;
while(s5 == 1 || s6 == 1)
{
reset_coms();
set_motor(7,150,0);
if ((peek(0x4000) & 0b00010000) > 0 )
s5=1;
else s5=0;
if ((peek(0x4000) & 0b00100000) > 0 )
s6=1;
else s6=0;
}
set_motor(7,0,0);
reset_coms();
}

void raise_claw()
{
reset_coms();
if ((peek(0x4000) & 0b00001000) > 0 )
s4=1;
else s4=0;
while (s4 == 1)
{
reset_coms();
set_motor(6,100,1);
if ((peek(0x4000) & 0b00001000) > 0 )
s4=1;
else s4=0;
}
set_motor(6,0,1);
reset_coms();
}

void move_claw_back()
{
reset_coms();
if ((peek(0x4000) & 0b00000010) > 0 )
s2 = 1;
else s2 = 0;
while ( s2 == 1)
{
reset_coms();
set_servo(4,42.0);
if ((peek(0x4000) & 0b00000010) > 0 )
s2 = 1;
```

```
else s2 = 0;
}
reset_coms();
set_servo(4,59.8);
reset_coms();
}

void claw_routine()
{
reset_coms();
open_claw();
forward_find_claw();
close_claw();
raise_claw();
move_claw_back();
open_claw();
close_claw();
reset_claw();
}
```