**University of Florida**

**Department of Electrical Engineering**

**EEL5934**

**Intelligent Machines Design Laboratory**

# Final Report

# Mojo

**Kevin J Hakala**

**12/3/95**

## TABLE OF CONTENTS

## ABSTRACT

This paper gives a complete description of the work and accomplishments that have been made with the autonomous mobile robot, Mojo. The paper starts with an overview of Mojo and then continues on to look in more detail at Mojo's platform and actuation and then each of his sensors and behaviors.

## EXECUTIVE SUMMARY

This paper serves as a complete description of the project that was undertaken as part of Intelligent Machines Design Laboratory, EEL 5934, a class taught by Professor Keith Doty in the Department of Electrical Engineering at the University of Florida. The project was to build an autonomous mobile robot that exhibited some interesting behaviors. Initially Mojo was designed to be a cat and exhibit those behaviors associated with a cat. As the course developed it became clear that those were lofty goals.

Mojo has IR sensors, CdS photoresistors, and a pyroelectric detector. These sensors allow Mojo to accomplish obstacle avoidance, light detection, and heat source detection. A Motorola microprocessor board serves as the control unit and it is programmed using Interactive C.

## INTRODUCTION

The project of building an autonomous mobile robot, Mojo, requires the use of several components. Mojo requires a controlling architecture which is provided by a Motorola HC11 EVBU board which is expanded with 32K of non-volatile memory. This controller is programmed using Interactive C.

Mojo's platform is constructed of LEGOs and mobility is provided by two swiss motors. To be autonomous, Mojo must be aware of his surroundings and react to them as necessary. Three different kinds of sensors provide this ability. Two IR sensors provide proximity detection for obstacle avoidance, eight CdS photoresistors provide light readings, and a pyroelectric detector detects heat sources.

The behaviors are implemented using a subsumption architecture. All three behaviors are running at the same time and an arbitration unit decides which behavior is granted control.

## ROBOT SYSTEM

The robot system is the sum of several different parts to obtain the desired features. A Motorola HC11 EVBU provides control over the system. This board has been expanded to contain 32K of memory using a Dallas non-volatile RAM. The robot senses the environment using several sensors. Mojo has Infrared LED's and detectors to provide proximity detection. Cadmium Sulfide cells detect light levels. A pyroelectric detector detects heat sources. Mojo also has a piezo speaker element to play music and aid in debugging.  The Motorola board continuously makes readings from these sensors and reacts as appropriate.

## LEGO PLATFORM

Mojo's platform is the base model of a LEGO Technic Hovercraft.  Its dimensions are approximately 5" wide by 10" long.  The LEGO structure has been strengthened with hot glue at certain points.  The LEGO structure has proven to be less than ideal.  The hot glue was needed to keep the structure together but even with that it is not very strong.  The inherently weak structure of the LEGOs also limited the size of the structure to something that was smaller than desired for holding all the equipment.

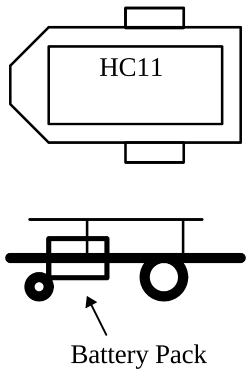

Figure 1. Platform Design

## ACTUATION

Two swiss motors drive Mojo.  They have sufficient torque and speed so that no gearing is necessary.  Driving the wheels directly from the swiss motors has not caused any problems.  This is likely due to the relatively light weight of Mojo's structure.

A L293 motor driver controls the motors and with these motors no diodes are necessary. Power is supplied by battery pack consisting of 8AA batteries. This same supply powers the Motorola board with 5 volts through a voltage regulator. The motors are mounted with a full size LEGO piece on the bottom and a base plate piece on the top. A dremel ground down the bottom piece so that the finished motor unit size was two LEGO blocks high. Epoxy attached the motor to the LEGO pieces, and this has proven to be a very sturdy assembly.

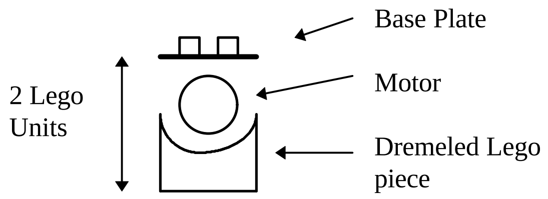

Figure 2. LEGO Motor Construction

Mojo’s wheels are two - 3” diameter model aircraft wheels. The wheels are directly mounted on the motor shafts with the aid of the plastic tubing from the inside of a pen and superglue. A 1.25” castor wheel on the back of the platform provides balance.

## IR PROXIMITY DETECTORS

Two IR detectors are used for collision avoidance. They are mounted in a cross eyed configuration. Each detector has one IR LED that is mounted on top of it. The LEDs output a 40kHz IR signal. This 40kHz signal is generated using a 74HC390 decade counter to divide down the HC11’s E clock. When this signal bounces off an obstacle it is read by a modified Sharp GP1U52X sensor which provides an analog output that is directly correlated to the amount of IR that it is detecting. The collision avoidance algorithm makes use of differential readings of the two sensors to make adjustments in speed and direction as in the code dynam11 which was discussed in class.

**CDS PHOTORESISTORS**

The main components of Mojo's light detecting sensor are CdS cells which are photoresistors. CdS cells vary in resistance depending upon the amount of light hitting their surface. These CdS cells have a resistance ranging from 100Ω in bright light to 500kΩ in the dark.

The CdS cells are collimated to focus them in one direction. They are collimated with ~1/4 inch of black heat shrink tubing. To make readings from these CdS cells, they are incorporated into a voltage divider with a 1kΩ resistor as shown in Figure 3. The analog signal from the voltage divider is then connected to an A/D line of the microprocessor.

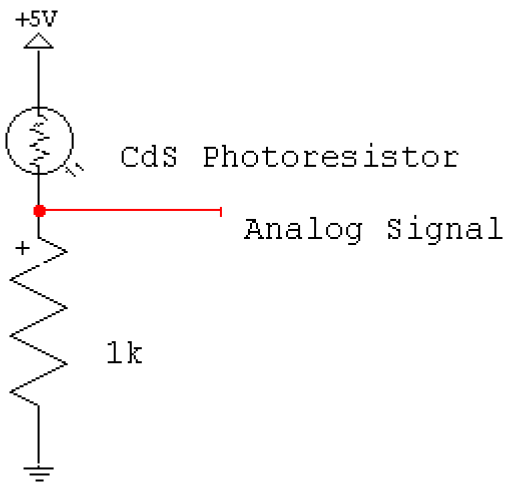

Figure 3. CdS Sensor in Voltage Divider.

Mojo's light detector sensor uses eight CdS cells so there are eight analog signals that need to be connected to A/D lines, but there are only a total of eight A/D lines on the 68HC11. Because of these limitations it was necessary to multiplex all eight analog signals to one A/D line. This is done through the use of memory mapped I/O, a 573 octal latch, and a 4051 analog multiplexer as shown in Figure 4. The latch was mapped to memory location $6000. The lower three data lines were then connected through the latch to the control lines of the multiplexer. These control lines select which CdS cell is attached to the A/D line. By poking address $6000 with a value from $00 to $07 one of the CdS signal can be selected and then read with IC's analog function.

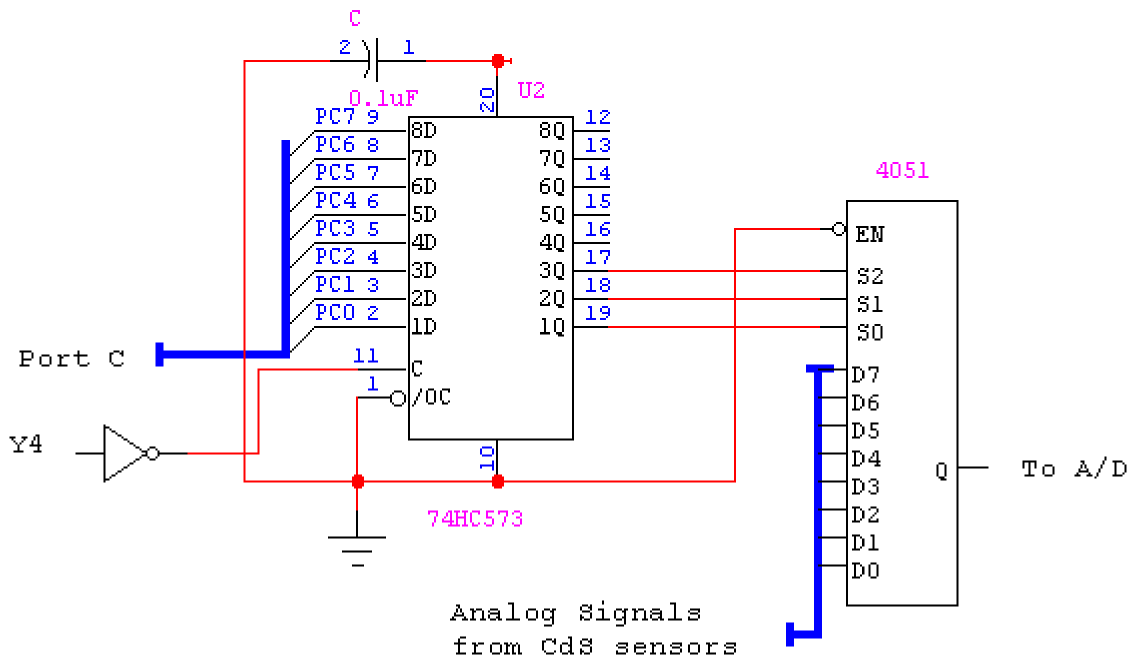

Figure 4. Memory Mapped I/O and Multiplexer.

As shown in Figure 5. the design of the light detector has been changed from the original design. Previously all CdS cells had their own resistor that composed their voltage divider, now there is only one resistor on the output side of the analog multiplexer. This simplified the hardware and allows for easier future modification of the resistor to change sensivity.

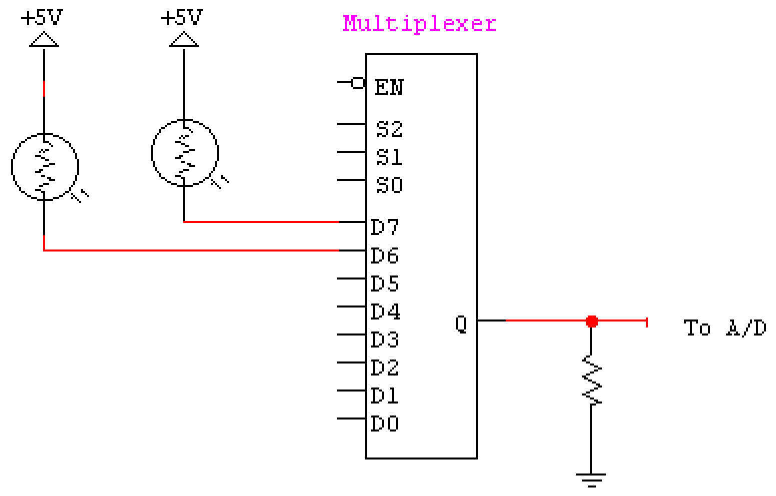

Figure 5. Modification to Original Light Detector Circuit.

Mojo's light detection sensor makes use of eight of the CdS cells to detect the brightest area in the room (a local maximum). These CdS cells are arranged in a petal like design as suggested to me by Scott. The seven CdS cells around the outside allow Mojo to see the brightness of his surroundings There is one CdS cell pointed straight up to give Mojo a light reading of where he is. The CdS cell that is pointed up will receive more light than any of the other cells so its sensitivity had to be reduced. Originally this was going to be done by using a different value resistor in its voltage divider but then it was easier to use the same hardware setup and to adjust its weight in the algorithm that determines where the brightest light is. To adjust it a number was subtracted from it before it is compared to the other sensor values.  This value was experimented with until the behavior was as desired.

## ELTEC PYROELECTRIC DETECTOR

A pyroelectric sensor is used to detect heat sources, in particular body heat. This sensor (Model 442-3, $35.00) and a fresnel lens (Model 6800408, $2.50) were obtained from Eltec Instruments (1-800-874-7780). It outputs an analog signal that is approximately 2.5 volts and then as a heat source passes through its field of vision this output signal rises up to ~3 volts, and then falls to ~2 volts before settling back to 2.5 volts. Depending on the direction that the heat source is moving it may fall and then rise. This sensor is provided with +5 volts and ground and the output is connected to an A/D line.

The pyroelectric sensor is mounted on the front of Mojo. The fresnel lens has a 1" focal length and is mounted at the end of a cone attached to the pyroelectric sensor. The cone is made of black construction paper and is wrapped in aluminum foil.

## BEHAVIORS

Mojo has three behaviors. All processes are running at the same time, and the subsumption architecture arbitrates between them. The following behaviors have been implemented with Mojo:

- Obstacle Avoidance
- Light Detection
- Heat Source Detection

Obstacle avoidance is obviously the highest priority behavior and will always receive priority over the other behaviors to avoid colliding with an obstacle. Mojo initially will search for the brightest area in a room. Upon finding it or becoming bored with searching he will scan for a heat source. If Mojo finds a heat source he will head in the heat source's general direction and then scan for it again. If Mojo is unable to reaquire the heat source he will become bored and switch to the light detection behavior. In each behavior if Mojo becomes bored that behavior will lose priority and another behavior will take over.

## CONCLUSION AND FUTURE WORK

There are several areas in which Mojo could be upgraded. The LEGO body is less than ideal and if he was to be redesigned that would be the first thing to change. The body would also be greatly improved if it was larger so that it would more easily accommodate more components.

The light sensors sensitivity could be changed by adjusting the one resistor that is used in a voltage divider with each of the CdS cells. This adjustment could be made by Mojo as he read the sensors and from the average reading he can know if his sensors are too sensitive or not sensitive enough and adjust the resistor accordingly.

The pyroelectric sensor could be mounted on a servo so that it could pan around the surroundings to detect heat sources. This would provide for much more accurate heat source tracking than what can be done now.

The obstacle avoidance program could be modified to include more than the two IR sensors. Previously Mojo had four IR sensors but as I changed the obstacle avoidance to use dynam11 I was unable to use all four.

Not much time was spent in the programming of Mojo. Each behavior was developed and tested independently and then were put together at the end. So there is not always a smooth working between the different behaviors. Next semester in the one credit class further programming in areas such as fuzzy logic and learning algorithms will be done which will greatly improve Mojo's behaviors.

## REFERENCES

[1] Anita Flynn, *Olympic Robot Building Manual*, MIT AI Memo, 1988.

[2] Joseph Jones and Anita Flynn, *Mobile Robots: Inspiration to Implementation*, A.K. Peters, 1993.

[3] Pattie Maes, *Designing Autonomous Agents*, MIT Press, 1990.

[4] Fred Martin, *The 6.270 Robot Builders Guide*, MIT Media Lab, 1992.

[5] Motorola, *M68HC11 Reference Manual*, Motorola, 1991.

# APPENDIX

```
/* Kevin Hakala*/
/* ic file for Mojo */
/* also need music.c */



/*sensors*/
/*sharps*/
int ir_left, ir_right;
/*CdS cell array*/
int eye[8];
/* turn processes on(1) or off(0) */
int dolight,doheat;
/* pyro detector reading */
int pyro;

int rand;
void main()
{
    /* start w/ light behavior */
    doheat=0;
    dolight=1;
    ao();
    start_process( sense() );
    start_process( avoid() );
    start_process( light() );
    start_process( detectheat() );
}

void sense() {
    while(1) {
    pyro=analog(6);
    /* scott's random number generator */
    rand=( ( ((int)mseconds()%2)* ((int)mseconds()))%2 )* (-2) +1;
    }
}

void light() {
    int max;
    sleep(5.0);
    while(1) {
    while(dolight==1) {
/* read light sensors  */
        /*front*/
        poke(0x6000, 0x00);
        eye[0]=analog(7)+2;
        /*up*/
        poke(0x6000, 0x01);
        eye[1]=analog(7)-75;
        poke(0x6000, 0x02);
```

```
          eye[2]=analog(7);
          poke(0x6000, 0x03);
          eye[3]=analog(7);
          poke(0x6000, 0x04);
          eye[4]=analog(7);
          poke(0x6000, 0x05);
          eye[5]=analog(7);
          poke(0x6000, 0x06);
          eye[6]=analog(7);
          poke(0x6000, 0x07);
          eye[7]=analog(7);

          max=maxarray(8,eye);
          /* go in direction of light */
          if(eye[0]==max)
               full_ahead();
          if(eye[2]==max||eye[3]==max||eye[4]==max)
               full_right();
          if(eye[5]==max||eye[6]==max||eye[7]==max)
               full_left();
          /* max light above then switch to heat behavior */
          if(eye[1]==max) {
               ao();
               /*looney_tune(); */
               tone(500.0,0.3);
               tone(600.0,0.2);
               tone(700.0,0.1);

               dolight=0;
               doheat=1;
               defer();
          }

     } }
}


void detectheat() {
int readpyro;
int hipyro=138;
int lopyro=118;
int next=0;
int toright=2;
int toleft=1;
int clear=0;

while(1) {



        next=clear;
        while(doheat==1) {
          readpyro=pyro;
        /*something going left to right*/
          if(readpyro>hipyro) {
               ao();
               charge();
               next=toright;
               slow_ahead();
               wait(1500);
               }
          if(readpyro<lopyro) {
               ao();
               charge();
               next=toleft;
               slow_ahead();
               wait(1500);
               }
          if(next==toright) slow_right();
          if(next==toleft) slow_left();

        }
}
}


void avoid()
{
float stime=0.001;
int get_clear=1;
int obstacle=105;


     while(1) {
          /*read sharps*/
          poke(0x7000,0x02);
          sleep(stime);
          ir_left=analog(1);
          poke(0x7000,0x01);
          sleep(stime);
          ir_right=analog(0);
          /* turn ir's off */
          poke(0x7000,0x00);
          /* check for obstacle*/
          if(ir_left>obstacle||ir_right>obstacle) {
               hog_processor();
               if(rand==1)
               full_right();
               else full_left();
               beep();
               beep();
               get_clear=0;
               doheat=0;
               dolight=0;
               }
          /* if you have gotten clear
               return to light behavior */
          if(ir_left<obstacle&&ir_right<obstacle&&get_clear==0) {
               ao();
```

```
               sleep(1.0);
               dolight=1;
               doheat=0;
               get_clear=1;
               }

}

}

/* returns the maximum value in a given array of given size */
int maxarray(int size, int array[]) {
     int max=0;
     int i=0;
     int temp;
     while(i<size) {
            temp=array[i];
            if (temp>max)
               max=temp;
            i++;
            }
     return max;
}

/* return absolute value */
float abs(float x) {
     if(x<0.0) x=-x;
     return x;
}

/*motor controls */
void full_ahead() {
        motor(1,90);
        motor(0,90);
}
void slow_ahead() {
     motor(1,70);
     motor(0,70);
}
void full_right() {
        motor(1,100);
        motor(0,-100);
}
void slow_right() {
     motor(1,70);
     motor(0,-70);
}
void full_left() {
        motor(1,-100);
        motor(0,100);
}
void slow_left() {
     motor(1,-70);
     motor(0,70);
```

```
}



void wait(int milli_seconds)
{
    long timer_a;
    timer_a = mseconds() + (long) milli_seconds;
    while(timer_a > mseconds()) {
        defer();
    }
}
```