# Quadro

**University Of Florida**
**Department of Electrical and Computer Engineering**
**Intelligent Machines Design Laboratory**

Jeffrey Van Anda
4/28/97
Dr. Keith L. Doty

## TABLE OF CONTENTS

ABSTRACT .... 3
EXECUTIVE SUMMARY .... 3
INTRODUCTION .... 3
ELECTRONIC SYSTEM .... 4
STRUCTURE .... 4
ACTUATION .... 5
BEHAVIORS .... 6
SLIPPAGE .... 9
CONCLUSION .... 10
CREDIT .... 11
APPENDEX .... 12

## ABSTRACT

Quadro is a four legged robot with eight degrees of freedom.  It runs off two 68HC11 processors with 32K of extended memory.  Quadro can walk in four directions (forward, backward, right, and left) without turning.

## EXECUTIVE SUMMARY

Quadro is a four legged walking robot.  It has two degrees of freedom on each leg.  This is accomplished using two high torque servos.  Of the two servos, one creates lift and the other propels the structure forward.  Quadro is run by two Motorola 68HC11 processors communicating through the SPI.  The main processor is mounted on a Motorola EVBU evaluation board and is in control of the arbitration and sensors.  The second is mounted on a MCC11 single chip board and controls the servo positions.  The main processor keeps track of the walking position and tells the second processor what to send to the servos.  The second processor controls the servos through a servo controller board.  The servo controller takes the five output compare pulse lines from the processor along with three select lines.  It then demultiplexes the time shared output compare lines and sends specific pulses to each of the servos .  Quadro walks by cycling through set leg positions.  Three legs drive Quadro forward while one leg is in the air resetting its position.  These leg positions are the same for walking forward and backward, Quadro just cycles through them in opposite directions.  Quadro has another set of positions for walking right and left.  The same relation hold between these positions.

## INTRODUCTION

Walking robots can be the most versatile in our world since most creatures living in it walk.  However, riding along with versatility is complexity.  To make a robot walk, one must explore the extensive mechanical design including things such as balance, footing, and structure. Also, one must explore the intricate electrical system such as controlling the servos to create walking patterns and extracting data from sensors for obstacle avoidance. The objective of Quadro was to create a four legged walker that could walk with only two servos per leg and required no balance shifting. Quadro was also to walk in all four directions without turning and avoid obstacles. This paper will cover Quadro up to the walking stages.

## ELECTRONIC SYSTEM

Quadro runs off two Motorola 68HC11 processors with 32K of extended memory. One processor is mounted on a Motorola EVBU evaluation board and is in control of the sensors and arbitration. The second processor is mounted on a MCC11 single chip board and sends output compare pulses to the servo controller board. The processors communicate together through the SPI. The servo controller board time shares the five output compare pulses and sends the correct pulses to the eight servos.

## STRUCTURE

Quadro's structure is made of two plates of the shape shown in Figure 1. The electronic are mounded in between the plates. The design is horizontally and vertically symmetric. With this design Quadro can walk forward, backward, left, or right equally without turning.

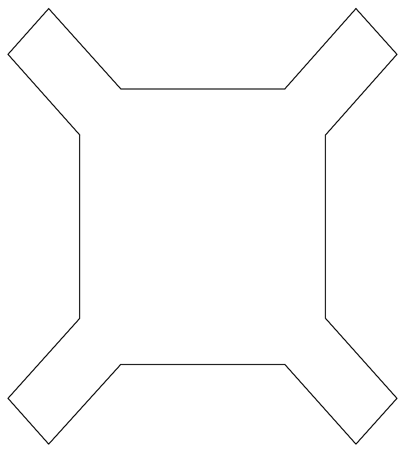


Figure 1 - Quadro Platform

## ACTUATION

Quadro has a leg assembly on each on the four outlets of its structure. The leg assembly is shown here in Figure 2.

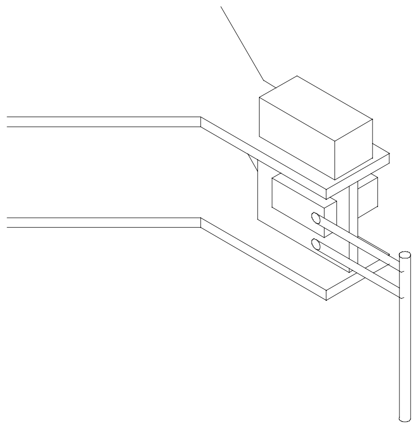


Figure 2 - Leg Assembly

The assembly consists of two servos. These servos need their high torque rating of 133oz.-in. since they will support all Quadro's weight or propel Quadro forward. Quadro creates lift with the servo that is mounted inside shown in figure 3.

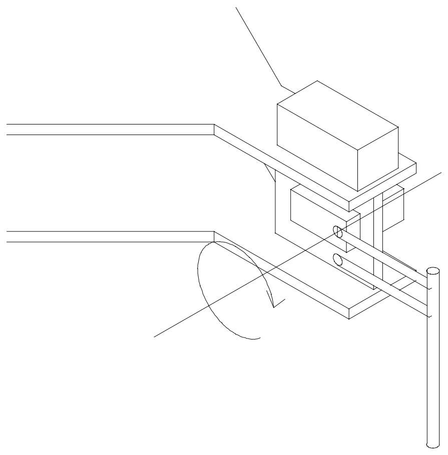


Figure 3 - Creating Lift

Through a double strut leg design as the servo rotates the leg will move straight up or straight down.  The servo mounted on top propels Quadro forward as shown in figure 4.

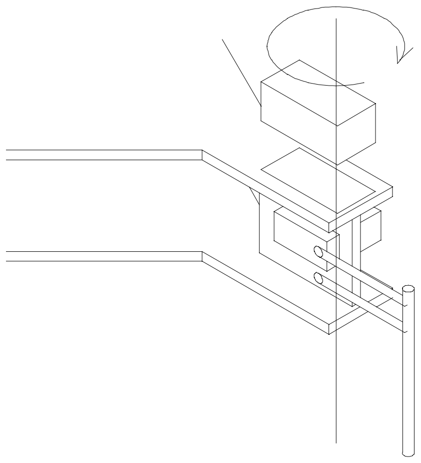


Figure 4 - Rotating Leg

The lift servo is mounted to a plate with the leg. The rotation servo rotates this entire setup to propel Quadro forward.  The actual leg must be rigid.  I made these from wooden dowels and two broke during demo time.  I then reconstructed one from a metal pipe, and the other from a carbon fiber arrow.  Both of these techniques worked fine.  The bottom strut must also be fastened to the plate firmly.  Any give with create
flexibility in the leg allowing it to rotate and break instead of propelling Quadro forward.

## BEHAVIORS

By controlling these servos effectively Quadro can walk forward, backward, left, and right.  In Quadro's walking scheme the leg plate has four positions as shown in figure 5.

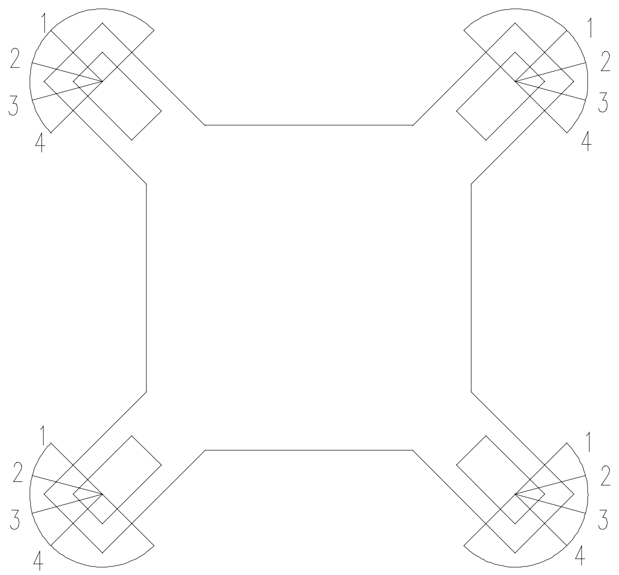


Figure 5 - Forward and Backward Plate Positions

At any given moment each leg assembly is in a different plate position. As Quadro walks the legs cycle through the positions. As the plate moves from position one to position two, for instance, that leg will be propelling Quadro forward. One leg will be in the air at all time also. This is when the leg is returning from position four to position one again. An example of Quadro's forward walk is shown in figure 6.

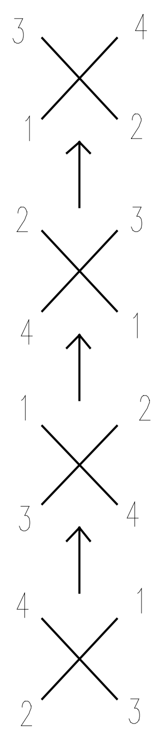


Figure 6 - Walking Example

This example starts from the bottom and it is as if Quadro is walking up the page. The font left leg starts in the number four plate position. This means the leg is raised. The idea was to put the other three legs in a position so that they would create a tripod formation about the center of gravity. Once Quadro was demonstrated we realized this is not exactly what was happening. What happened was the other legs were higher taking most of Quadro's weight, but the raised leg was still touching the ground. The leg could slide itself across the ground to it's new position. Thus, Quadro would still walk, but the legs never really left the ground. Quadro can walk backward in the same manner with the same plate positions only cycling through them backward. The leg has to be raised when moving from the one plate position to the four plate position. The same relation hold between the right and left movements, but the plate positions are different. These plate positions are in figure 7.

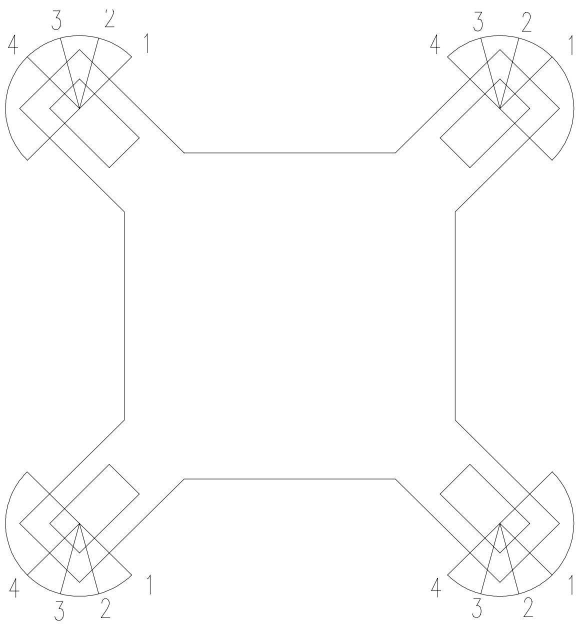


Figure 7 - Right and Left Plate Positions

## SLIPPAGE

There is another problem with this walking structure and scheme. There is an amount of slippage as the legs rotate to propel Quadro forward. This is shown if figure 8.

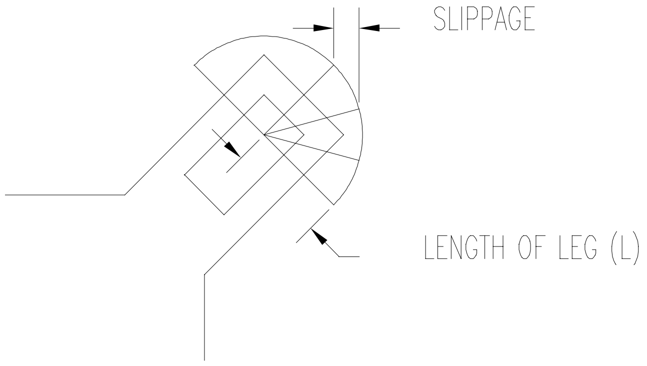
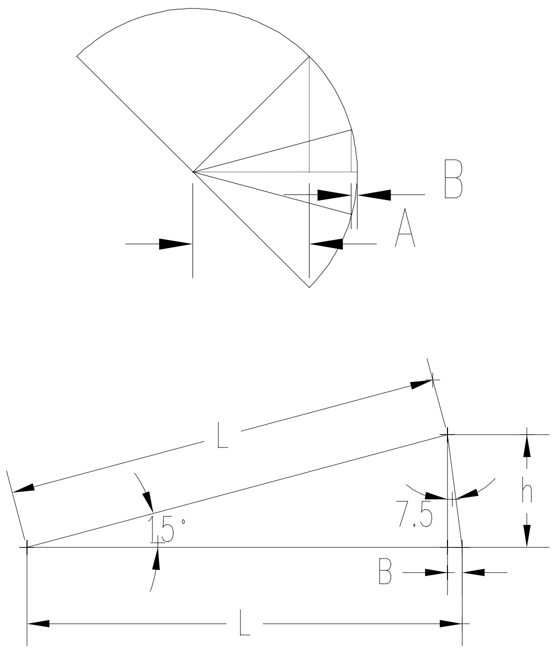


Figure 8 - Slippage

The calculations for this slippage are as follows in figure 9:



$$\text{Slippage} = L - A - B$$

$$\cos(45) = A / L$$

$$A = L * \cos(45)$$

$$\sin(15) = h / L$$

$$h = L * \sin(15)$$

$$\tan(7.5) = B / L * \sin(15)$$

$$B = L * \sin(15) * \tan(7.5)$$

$$L - A - B = 0.26 * L$$

Figure 9 - Slippage Calculation

Since I set the length of the leg at 2 inches, the slippage will be 0.52 inches.

## CONCLUSION

Quadro is constructed up to the scope of this paper. It is walking in four directions, but has no obstacle avoidance. The mechanics of this project exceeded my knowledge since I am an electrical engineer, and need improving. I would like to adjust Quadro to fully lift its legs off the ground when walking. I would also like to implement my obstacle avoidance scheme. If I were to start this project over, I would rethink the entire mechanical setup. I would explore some balance shifting techniques.

## CREDIT

I used some mechanical ideas from the robot “Thing” by Jessy Nightingale at University of Massachusetts.

Please refer to his web site at http://piglet.cs.umass.edu:4321/thing/thing.html

I want to thank Purvesh Thakker for his design of the Servo Controller Board.

## APPENDEX

```
/********  Servo Numbers ********/
int front_left_rot = 1;
int front_left_lift = 2;
int front_right_rot = 3;
int front_right_lift = 4;
int back_right_rot = 5;
int back_right_lift = 6;
int back_left_rot = 7;
int back_left_lift = 8;

/********  Front Left (1,2) ********/
int flr_pos_array[9] = {3400,3967,4533,5100,          /*First 4 for For/Back  */
                        1600,2200,2800,3400};          /*  second 4 for Right/Left  */
int flr_stride_array[9] = {4,1,2,3,2,3,4,1};

int fll_pos_array[3] = {3200,3700};
int fll_stride_array[17] = {2,1,1,1,1,2,1,1,          /* First 4 for For/Right */
                            1,1,2,1,1,1,1,2};          /* second 4 for Back/Left  */

/******** Front Right (3,4) ********/
int frr_pos_array[9] = {3300,2700,2100,1500,
                            3300,3900,4500,5100};
int frr_stride_array[9] = {1,2,3,4,4,1,2,3};

int frl_pos_array[3] = {3400,2900};
int frl_stride_array[17] = {1,1,1,2,2,1,1,1,
                            2,1,1,1,1,2,1,1};

/******** Back Right (5,6) ********/
int brr_pos_array[9] = {4900,4333,3767,3200,
                            3200,2634,2068,1500};
int brr_stride_array[9] = {3,4,1,2,1,2,3,4};

int brl_pos_array[3] = {3400,3900};
int brl_stride_array[17] = {1,2,1,1,1,1,2,1,
                            1,1,1,2,2,1,1,1};

/********  Back Left (7.8) ********/
int blr_pos_array[9] = {1500,2067,2633,3200,
                            4800,4267,3734,3200};
int blr_stride_array[9] = {2,3,4,1,3,4,1,2};
int bll_pos_array[3] = {3500,3000};
int bll_stride_array[17] = {1,1,2,1,1,1,1,2,
                            1,2,1,1,1,1,2,1};

/********  Other Variables ********/
int vel = 100;
int acc = 10;
int dec = 10;

/********  Position Functions 1-4  ********/

void flr_pos(int x)
{
                set_servo_pul(front_left_rot,flr_pos_array[x-1],vel,acc,dec);
}
void frr_pos(int x)
{
                set_servo_pul(front_right_rot,frr_pos_array[x-1],vel,acc,dec);
}
void brr_pos(int x)
{
                set_servo_pul(back_right_rot,brr_pos_array[x-1],vel,acc,dec);
}
void blr_pos(int x)
{
                set_servo_pul(back_left_rot,blr_pos_array[x-1],vel,acc,dec);
}
/********  lift 1=mid 2=top ********/
void fll_pos(int x)
{
                set_servo_pul(front_left_lift,fll_pos_array[x-1],vel,acc,dec);
}
void frl_pos(int x)
{
                set_servo_pul(front_right_lift,frl_pos_array[x-1],vel,acc,dec);
}
void brl_pos(int x)
{
                set_servo_pul(back_right_lift,brl_pos_array[x-1],vel,acc,dec);
}
void bll_pos(int x)
{
                set_servo_pul(back_left_lift,bll_pos_array[x-1],vel,acc,dec);
}
```

```
/********  Wait Function  ********/
void wait(int milli_seconds)
{
                long timer_a;

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

/********  Step Functions  ********/

/**** step forward, backward, right, or left  ****/
/*  Forward:  stride_modifier =  1, rot_modifier = 0, lift_modifier = 0  */
/*  Backward: stride_modifier = -1, rot_modifier = 0, lift_modifier = 4  */
/*  Right:    stride_modifier =  1, rot_modifier = 4, lift_modifier = 8  */
/*  Left:     stride_modifier = -1, rot_modifier = 4, lift_modifier = 12 */
int step_fb(int stride, int stride_modifier, int rot_modifier, int lift_modifier)
{
                wait(100);
                flr_pos(flr_stride_array[stride-1+rot_modifier]+rot_modifier);
                frr_pos(frr_stride_array[stride-1+rot_modifier]+rot_modifier);
                brr_pos(brr_stride_array[stride-1+rot_modifier]+rot_modifier);
                blr_pos(blr_stride_array[stride-1+rot_modifier]+rot_modifier);
                wait(600);

                fll_pos(fll_stride_array[stride-1+lift_modifier]);
                frl_pos(frl_stride_array[stride-1+lift_modifier]);
                brl_pos(brl_stride_array[stride-1+lift_modifier]);
                bll_pos(bll_stride_array[stride-1+lift_modifier]);
                wait(300);

                stride = stride + stride_modifier;
                if (stride>4)
                        stride = 1;
                if (stride<1)
                        stride = 4;

                return stride;
}
int step_forward(int stride)
{
                int new_stride;
```

```
                         new_stride = step_fb(stride,1,0,0);
                         return new_stride;
}
int step_backward(int stride)
{
                         int new_stride;
                         new_stride = step_fb(stride,-1,0,4);
                         return new_stride;
}
int step_right(int stride)
{
                         int new_stride;
                         new_stride = step_fb(stride,1,4,8);
                         return new_stride;
}
int step_left(int stride)
{
                         int new_stride;
                         new_stride = step_fb(stride,-1,4,12);
                         return new_stride;
}

/*****  Walk Functions  *****/
int walk_forward(int temp)
{
int count = 0;                   /* count build in just to create delay for demonstration */
while (count<40){
                         temp = step_forward(temp);
                         count++;
                         }
return temp;
}
int walk_backward(int temp)
{
int count = 0;
while (count<40){
                         temp = step_backward(temp);
                         count++;
                         }
return temp;
}
int walk_right(int temp)
{
int count = 0;
while (count<40){
```

```
                        temp = step_right(temp);
                        count++;
                        }
return temp;
}
int walk_left(int temp)
{
int count = 0;
while (count<40){
                        temp = step_left(temp);
                        count++;
                        }
return temp;
}

void main()
{
                        servos_on();
                        walk();
}
void walk()
{
int track = 1;

while (1){

                        track = walk_forward(track);
                        track = walk_backward(track);
                        track = walk_right(track);
                        track = walk_left(track);
                        }
}
```