

Grand Valley State University

School of Engineering
College of Engineering and Computing

EGR 345 Dynamic Systems Modeling and Control



Final Project Report
Team 2

Members:
Todd Alexander
Dave Bouwkamp
Rick Lehman
Kevin Littlepage
Derek Maxim

December 7, 2005

Table of Contents

Table of Contents	2
1.0 Executive Summary	3
2.0 Design Description	4
2.1 Assembly Drawing	
2.2 System Block Diagrams	
2.4 Firing System Wiring Schematic	
2.5 Calculations	
2.6 Project Budget, BOM, Weight Inventory	
3.0 Test Results	13
3.1 Simulation Results	
3.2 Final Competition Results	
3.3 Overall Score Estimates	
4.0 Conclusions	14
5.0 Recommendations	14
Appendices	15
Appendix A: Drawings	
Appendix B: C Program	
Appendix C: MatLab Simulation	
Appendix D: Receipts and Cost Evidence	

1.0 Executive Summary

The Target Acquisition and Firing System is a circular base with a rotating platform connected by a gear head motor. The motor will rotate the platform a maximum of 22 degrees in both the positive and negative direction, and will align the barrel with each target. The gear head motor will be controlled using a C program involving a position feedback loop. This loop will allow for the precise rotation of the motor and therefore, the aiming of the firing mechanism. A potentiometer will be used for feedback to the motor. The feasibility of using a potentiometer will allow for the firing mechanism to be aimed into the center of the four degree range of each target. The active target will be determined by four signals connected to different ports on the controller board. The board will read the inputs every time the interrupt is run, and the current target will be the target corresponding to the logical high input.

The balls will be fired through a tube, using two spinning wheels with high radial velocities, similar to a pitching machine. These wheels will be powered by two separate motors spinning at the same velocity. The speed of each motor will be controlled using velocity feedback similar to that used in lab 9. Equal speeds for the firing motors are necessary to ensure straight flight of the ball. These speeds will be set in the program and held constant, while the motor uses feedback to position the barrel throughout the entire test. There will be one feedback control loop that will control the angular positioning of the platform.

2.0 Design Description

2.1 Assembly Drawing

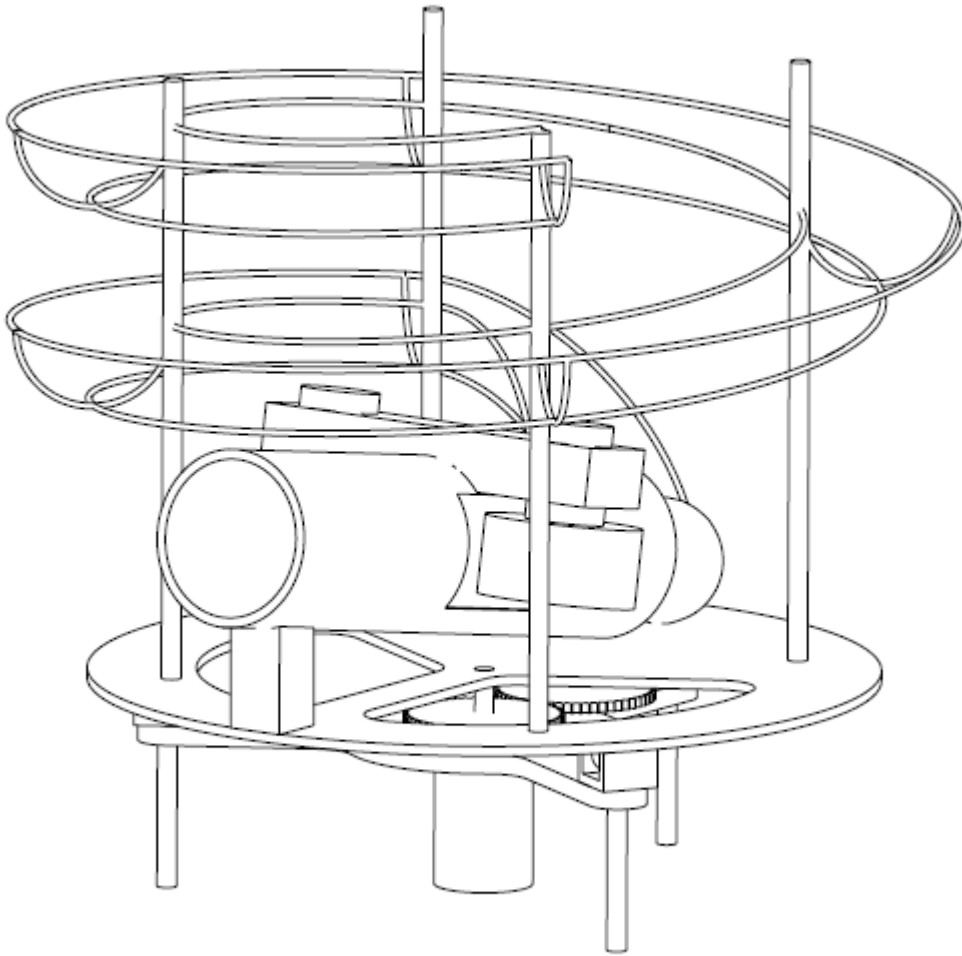


Figure 2.1: Assembly Drawing of Target Acquisition and Firing System

2.2 System Block Diagrams

Figures 2.2 through 2.6 shows various wiring and system schematics for the Target Acquisition and Firing System.

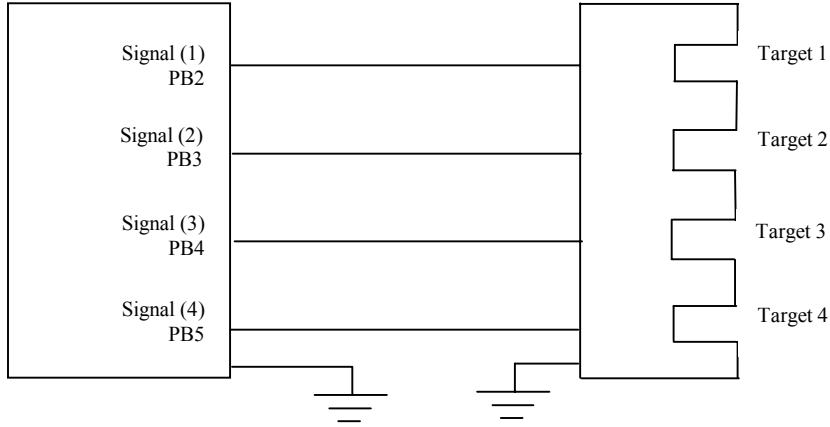


Figure 2.2: Target signal wiring schematic

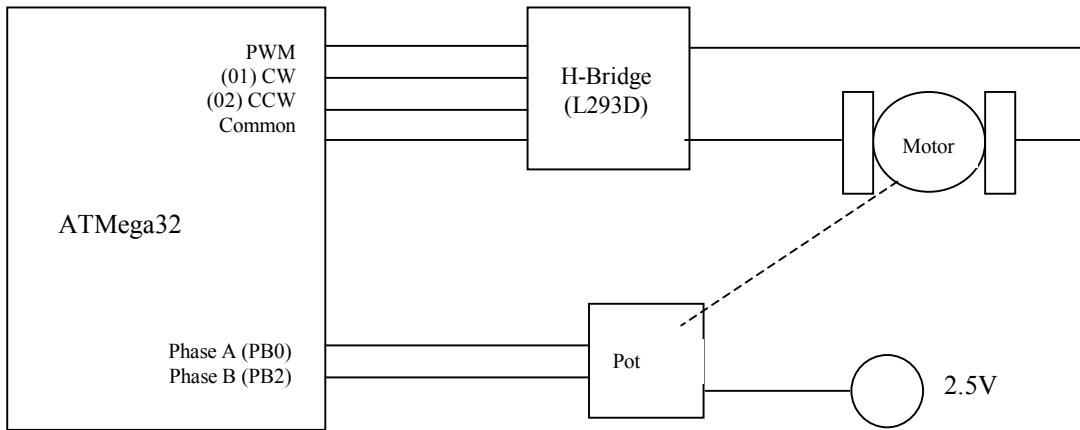


Figure 2.3: Target positioning wiring system

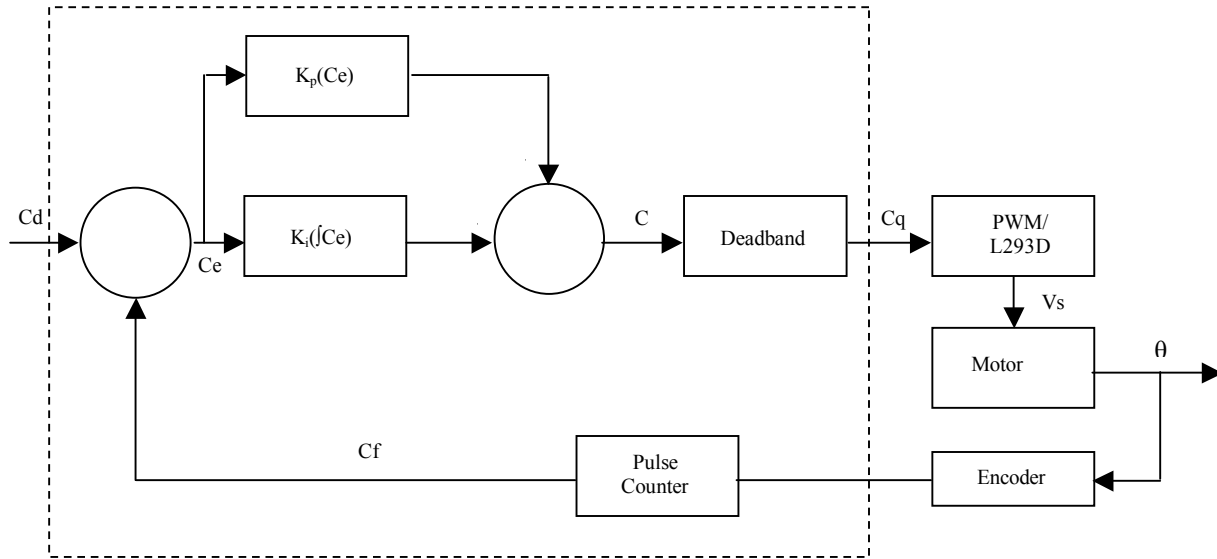


Figure 2.4: Target positioning program feedback loop



Figure 2.5: Firing motor wiring schematic

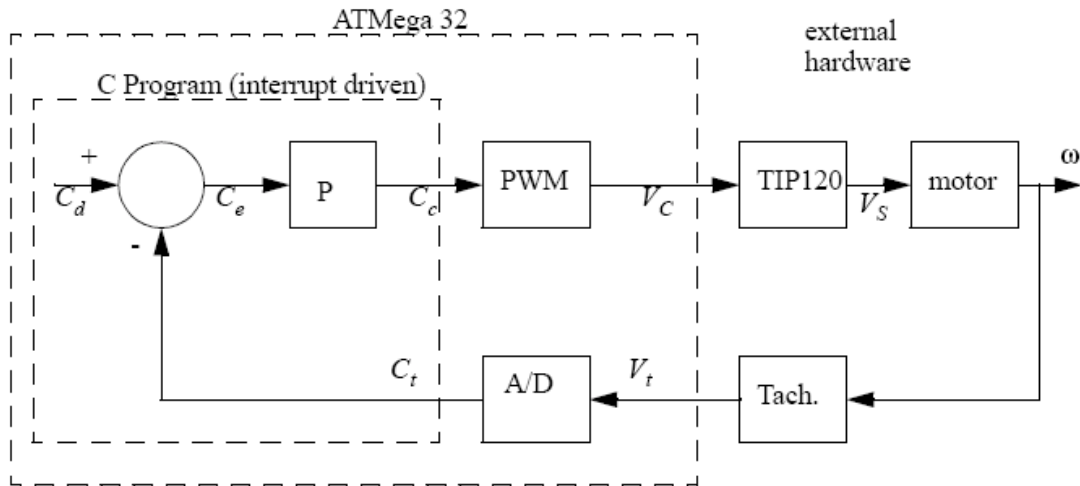


Figure 2.6: Ball firing program feedback loop

- ** C_d = Set motor speed (left)
- C_t = Actual Motor Speed (left)
- $C_e = C_d - C_t$ = system error (left)
- C_c = Control volume to output (left)
- P = Controller Gain (left)

** Right motor will use PD6 and PA1 instead of PA7 and PA0.

2.4 Firing System Wiring Schematic

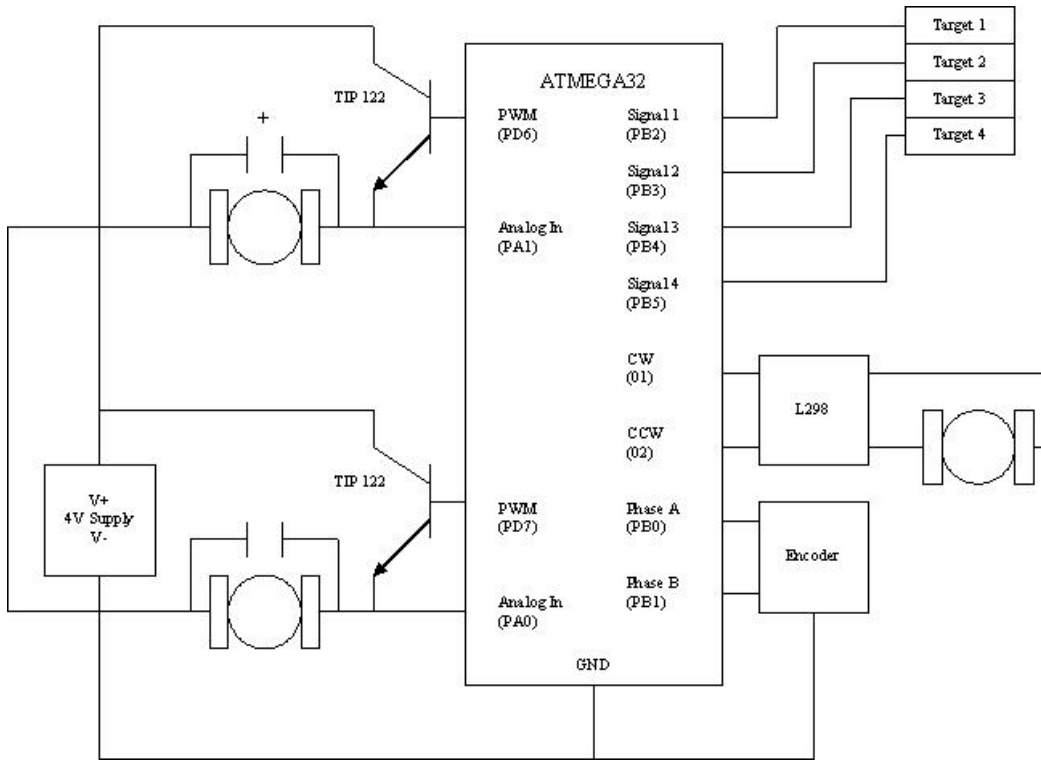


Figure 2.7 Firing system wiring hematic

2.5 Calculations

The full angle of rotation required for the base to turn was calculated using Figure 2.8.

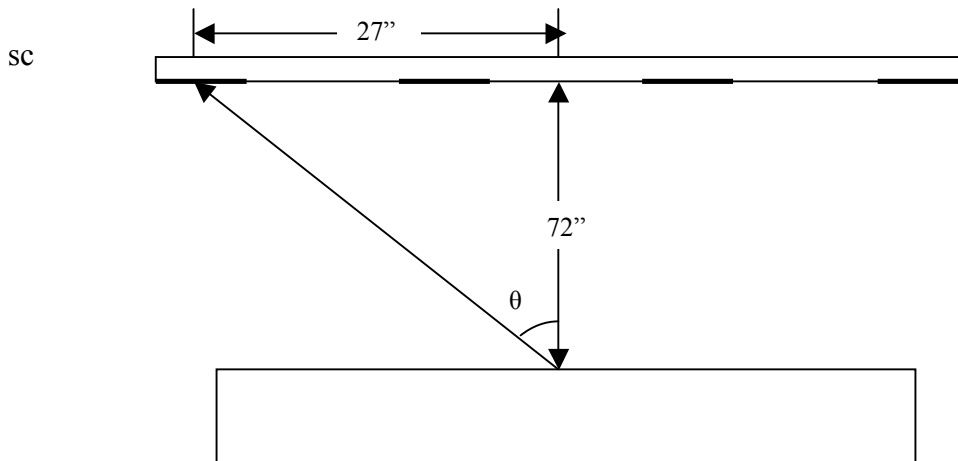


Figure 2.8: Angle of rotation for apparatus

Basic trigonometry yields that the angle of rotation to reach the far target when the firing mechanism is located in the center of the team area is 20.6°, as shown in Equation 2.1.

$$\theta = \arcsin\left(\frac{27}{72}\right) = 20.556 \quad (2.1)$$

Therefore, the base must rotate a total of 41° to hit both far targets.

The wheels being used for the apparatus are soft inflated rubber wheels attached to a plastic rim. As the velocity of the wheels increases, the diameter increases and centrifugal force increases. Placement of the wheels relative to each other is critical, because the gap between the two wheels must be just enough for the ball to be grabbed by the wheel and pass through. For various voltages, the wheel diameter was carefully measured with a pair of calipers. Results from the test are shown in Table 2.1.

Table 2.1: Wheel Diameter Vs. Voltage Calibration Results

Voltage	Wheel Dia. (in)
1.479	2.354
2.531	2.354
3.503	2.405
4.406	2.46
5.41	2.541
6.16	2.592
7.13	2.662

The collected data was plotted (as shown in Figure 2.10) and used as a calibration curve for computing tangential velocity.

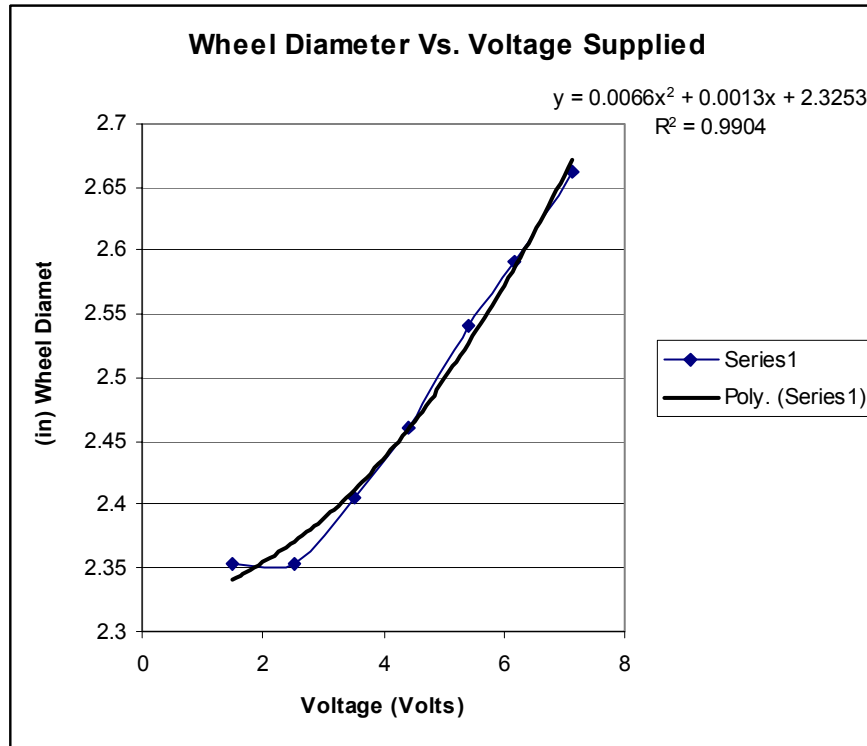


Figure 2.9 – Wheel Diameter Vs. Supplied Voltage

A polynomial regression fit the data with a correlation of 99%. For various voltages the angular velocity of one of the wheels was measured with a tachometer. The equation from the regression line in Figure 2.9 was used to determine the approximate diameter of the wheel at the voltages tested. With these pieces of information the tangential velocity of the wheels was determined as shown in Table 2.2.

Table 2.2: Determination of Tangential Velocity Using Voltage, Velocity, and Wheel Diameter

Voltage (V)	RPM	Wheel Dia. (in)	Circumference (ft)	Approximate Tangential Velocity (ft/sec)
1.866	765.1	2.351	0.615	7.848
2.942	1605	2.386	0.625	16.711
3.344	1867	2.403	0.629	19.579
4.103	2510	2.442	0.639	26.742
5.03	3287	2.499	0.654	35.839
6.02	4028	2.572	0.673	45.210

A plot of the data shown in Table 2 is shown in Figure 2.10.

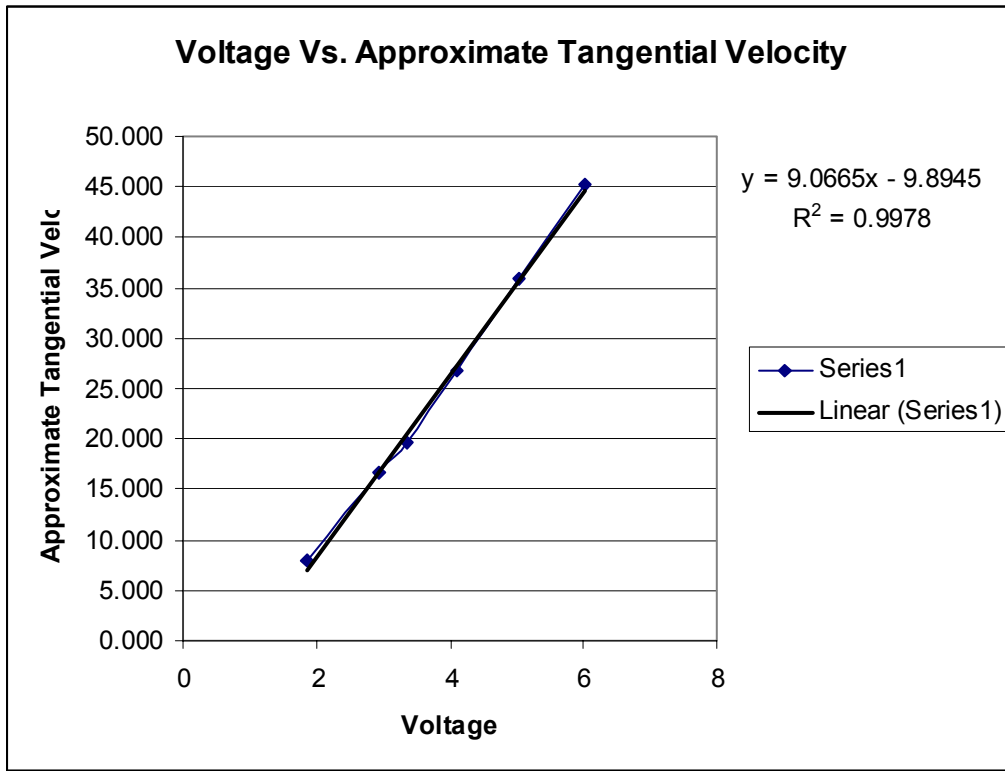


Figure 2.10 – Calibration Curve for Determining Tangential Velocity from Voltage Supplied to the Firing Motors.

2.6 Project Budget, BOM, Weight Inventory

Table 2.3 shows the material list and budget of the Target Acquisition and Firing System. The total cost of the project was \$101.04 (receipts for cost verification are found in Appendix D). Additionally, each part of the assembly was weighed individually to determine the weight inventory. The total weight of the project was 1.141 kg. The weight and cost are figured into the Score estimate found in Section 3.3.

Table 2.3: Project budget, materials, and weight inventory

Qty	Description	Price	Ext. Price	Mass (Kg)	Supplier	Status
1	Wire for Ball Feed	\$3.50	\$3.50	0.08	Gemmens	Purchased
2	Small High RPM Motors (firing wheels)	\$2.99	\$5.98	0.15	Riders	Purchased
1	Rubber Wheels (firing wheels)	\$13.99	\$13.99	0.25	Riders	Purchased
1	ATMega32 Circuit Board	\$30.00	\$30.00	0	GVSU	Acquired
1	Breakout Board	\$20.00	\$20.00	0	GVSU	Acquired
1	Potentiometer	\$2.58	\$2.58	0.005	GVSU	Acquired
1	Linear Actuator (1" stroke)	\$14.99	\$14.99	0.135	McMaster	Purchased
1	AC Gear Motor	\$10.00	\$10.00	0.19	GVSU	Acquired
1	Misc. Brackets/Components	\$0.00	\$0.00	0.326	GVSU	Purchased
3	Rollers	\$0.00	\$0.00	0.005	Made	Acquired
Total:		\$98.05	\$101.04	1.141		

The ATMega32 Circuit Board and the Breakout board are listed as 0 kilograms because their masses do not count for the overall apparatus mass used in determining the performance score.

3.0 Test Results

3.1 Simulation Results

The targets are approximately 6 inches wide by 10 inches tall. Aiming at the center of the target, it is necessary to determine the parameters necessary for the angle of the barrel and voltage supplied to the firing wheel motors. MatLab was used to simulate the projection of the ball toward a target 6 feet away. From testing the prototype it was found that the firing wheels performed best at a velocity of approximately 6 volts. This corresponded to a tangential velocity of 45.21 ft/sec. Using the mass and diameter of the ball, a constant drag coefficient, and a barrel angle of 15 degrees the simulation was executed and a plot of height versus distance of the ball was generated, as shown in Figure 3.1.

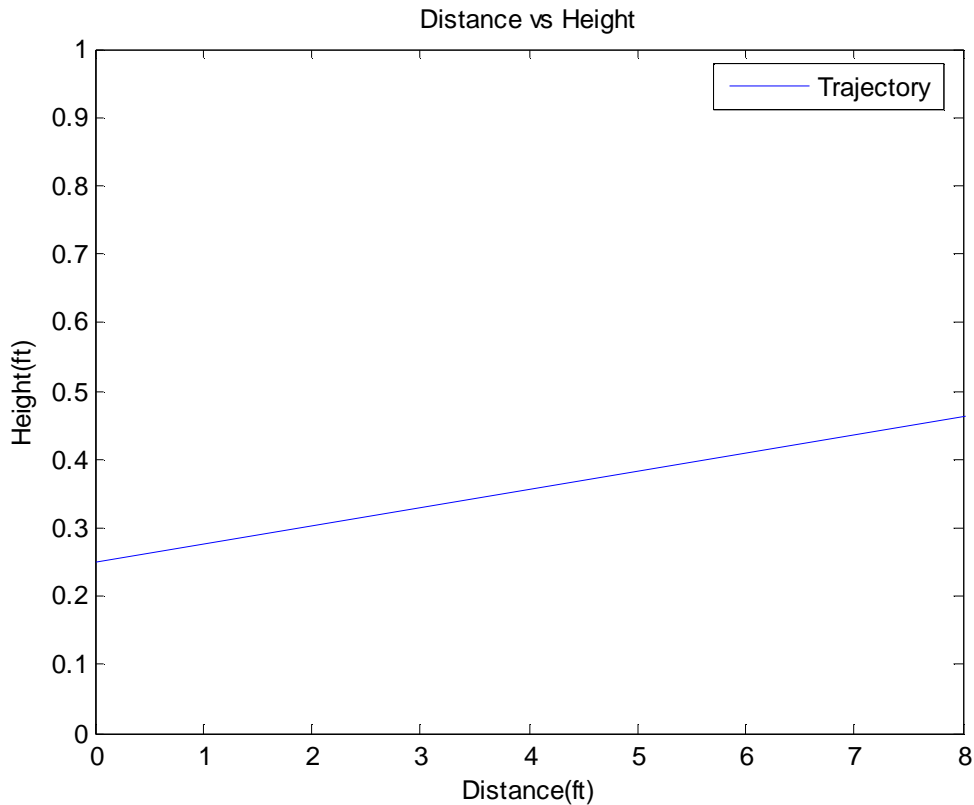


Figure 3.1 – Height of Ball Versus Horizontal Distance from Barrel.

It was found that the motion of the ball over a 6 foot range is approximately linear. From the simulated graph shown in Figure 2.11, at 6 feet the ball is approximately 5 inches off the ground, which will be the center of the target. The MatLab simulation program is shown in Appendix C.

3.2 Final Competition Results

Final competition results are yet to be determined.

3.3 Overall Score Estimates

The final score is determined using Equation 3.1,

$$score = H^{-2} (4)_{80}^C (10)^{-B} (10)^{-T} (2)_{3}^L (M)^4 \quad (3.1)$$

where: H is the number of targets hit in the two minute interval, C is the total cost of the parts (in dollars) , L is the largest dimension of the apparatus (in inches), B is the build quality assigned by judges, T is the theory quality assigned by judges, and M is the mass of the apparatus (in kilograms).

An initial estimate of the project score (as of November 16) is shown below in Table 3.1. Two estimates were calculated. The estimate assumes good scores (1.0) for the build and theory quality and a high amount of targets hit during the two minute test. The second estimate assumes poor quality scores (0.1) and a low number of targets hit.

Table 3.1: Performance estimates (using Equation 3.1)

Scoring Estimate (Performance)	Variable s	Value(good)	Value (bad)
	H	30	2
	C	\$101.01	\$101.04
	B	1	0.1
	T	1	0.1
	L	9	12
	M	1.141	1.141
Estimates (using score equation)		0.0008677	24.63793

Possible scores range from 0.00087 to 24.638. This is a large range due to the unknown quality scores that will be assigned by the judges, as well as the actual number of targets hit during the competition.

4.0 Conclusions

The final apparatus was finished and tested numerous times. The design performs the intended task quickly and efficiently. The calculations performed in Section 2 and the Simulation in Section 3 found that the firing wheel motors should be given a voltage of 6 volts. If the voltage was any lower than that the ball would land short of the target. During testing this theory was confirmed. The machine must be given a voltage, no less than 6 volts, if the ball is to land in the target above ground. All the major components are in proper working order and will be ready for the final competition on Dec. 8th, 2005. The potentiometer, base motor, and gears work very well, with the code written, to move the base toward the intended targets. The most important part of getting the motor to work correctly was adjusting the gain in the program. During the design phase the load on the motor would change. Adjusting the gain in the program would compensate for this changing load. The actuator feeds the balls into the firing wheels when the machine reaches the intended target. The feeding mechanism is a winding track of wire that works better than a hopper that can get jammed easily. At the first competition the design didn't look as appealing to the eye as it should have been. The machine was taken apart and given a major face lift to improve the build quality. This design is expected to perform at a very high level and will be very tough to beat at the final competition.

5.0 Recommendations

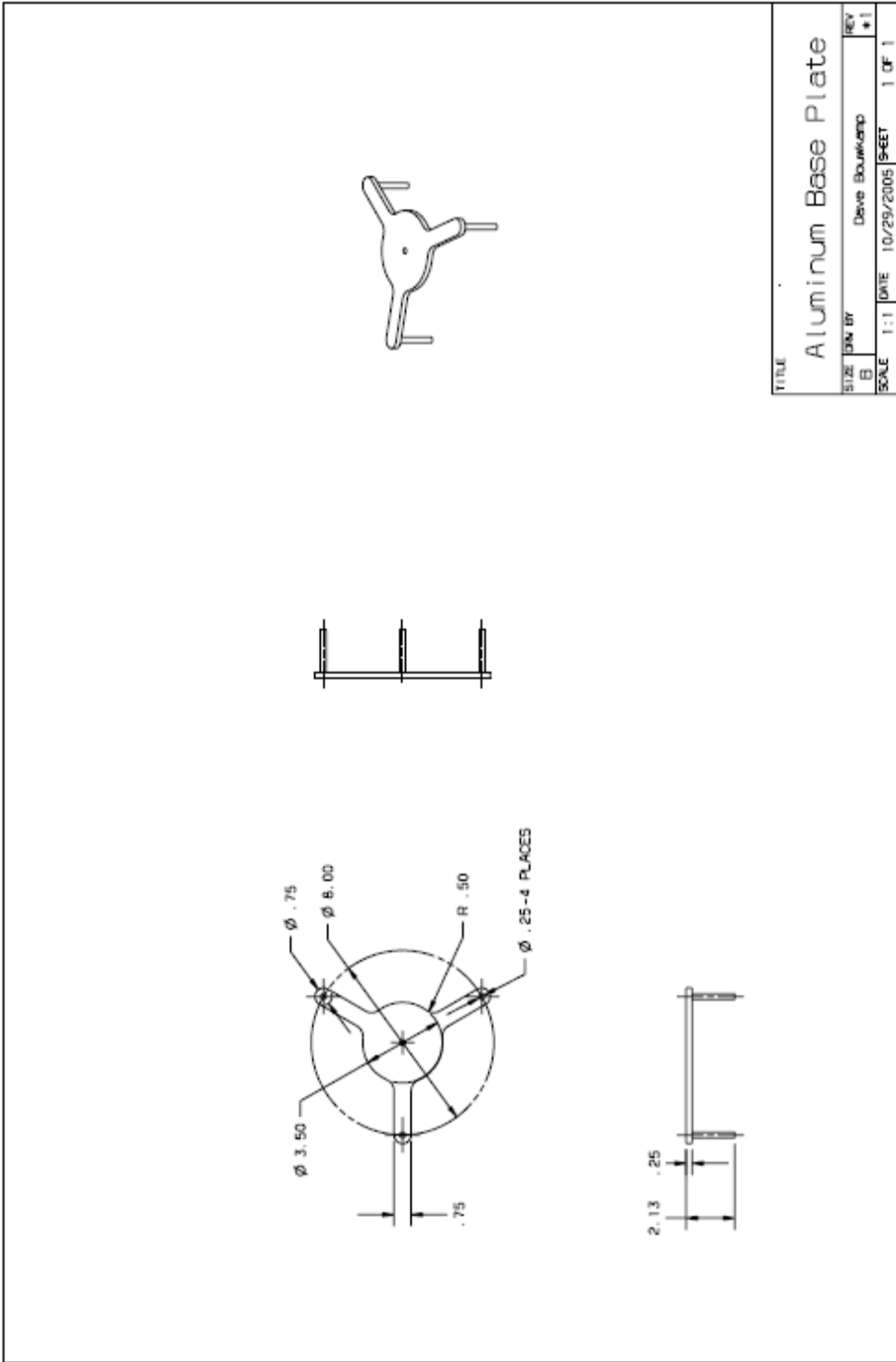
The results shown in the calculations section will be used to modify the design slightly to optimize the firing mechanism and hopefully reduce the voltage supplied to the firing motors. This will hopefully reduce vibration and increase accuracy. The one advantage of vibration is it helps the feeding mechanism if it gets jammed. The feeding mechanism was built using glue and fastening wire to keep it together. Sometimes the balls get caught up on the wire that is wrapped around the track. One recommendation would be to use different materials to build the track, so this can't happen. Another problem that comes with the current design is weight. Since the entire machine relies on heavy electrical motors the weight of the design can increase rapidly. This forced every other component to be extremely light, which means they may get weak over time. The designed machine didn't utilize air compression at all. This meant that many power supplies were needed to have a fully functional machine. Wiring could get very messy at times and if air was utilized, this problem may be

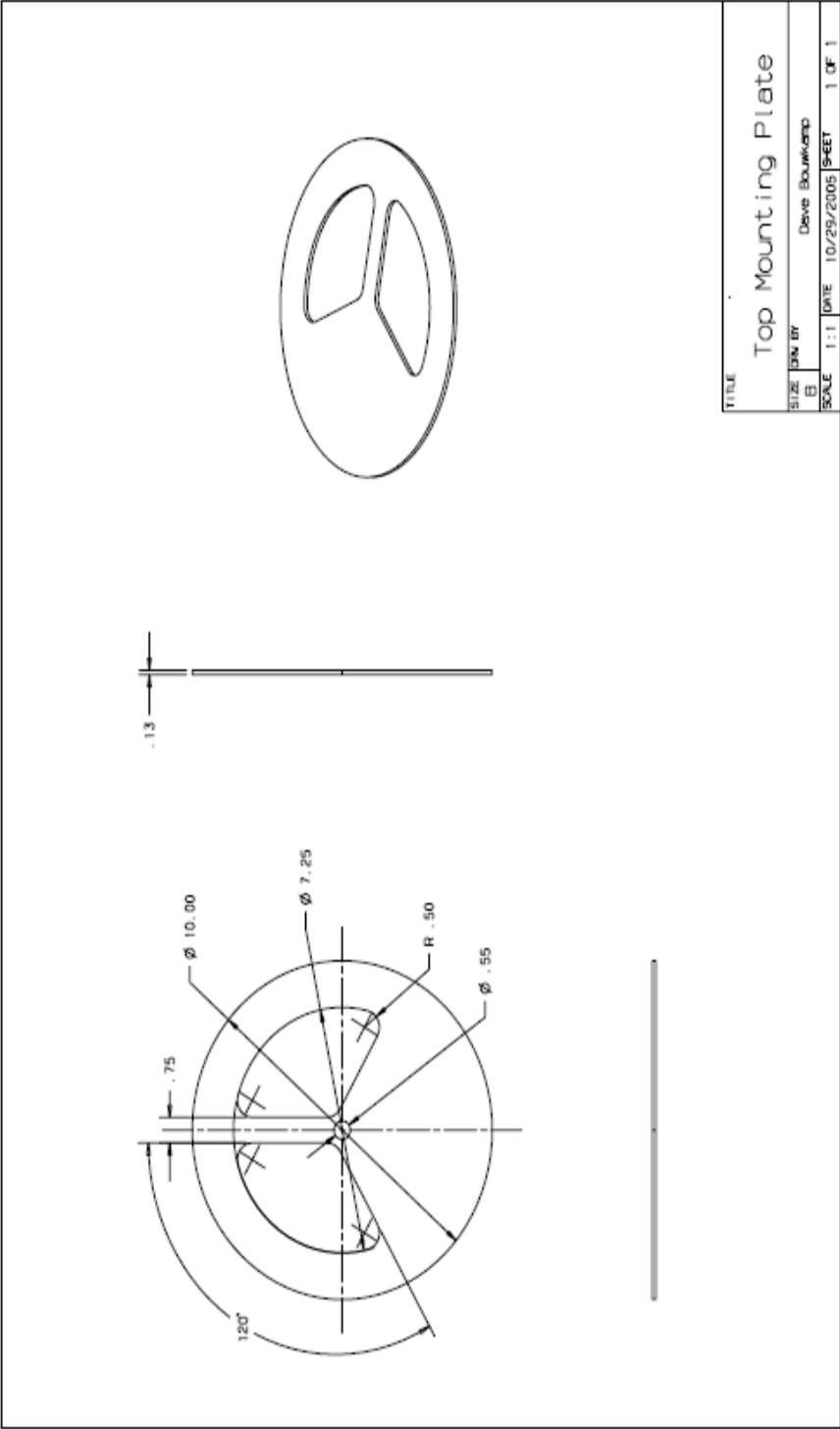
eliminated. These recommendations should be very helpful for any one designing a machine similar to the one used in this project.

Appendices

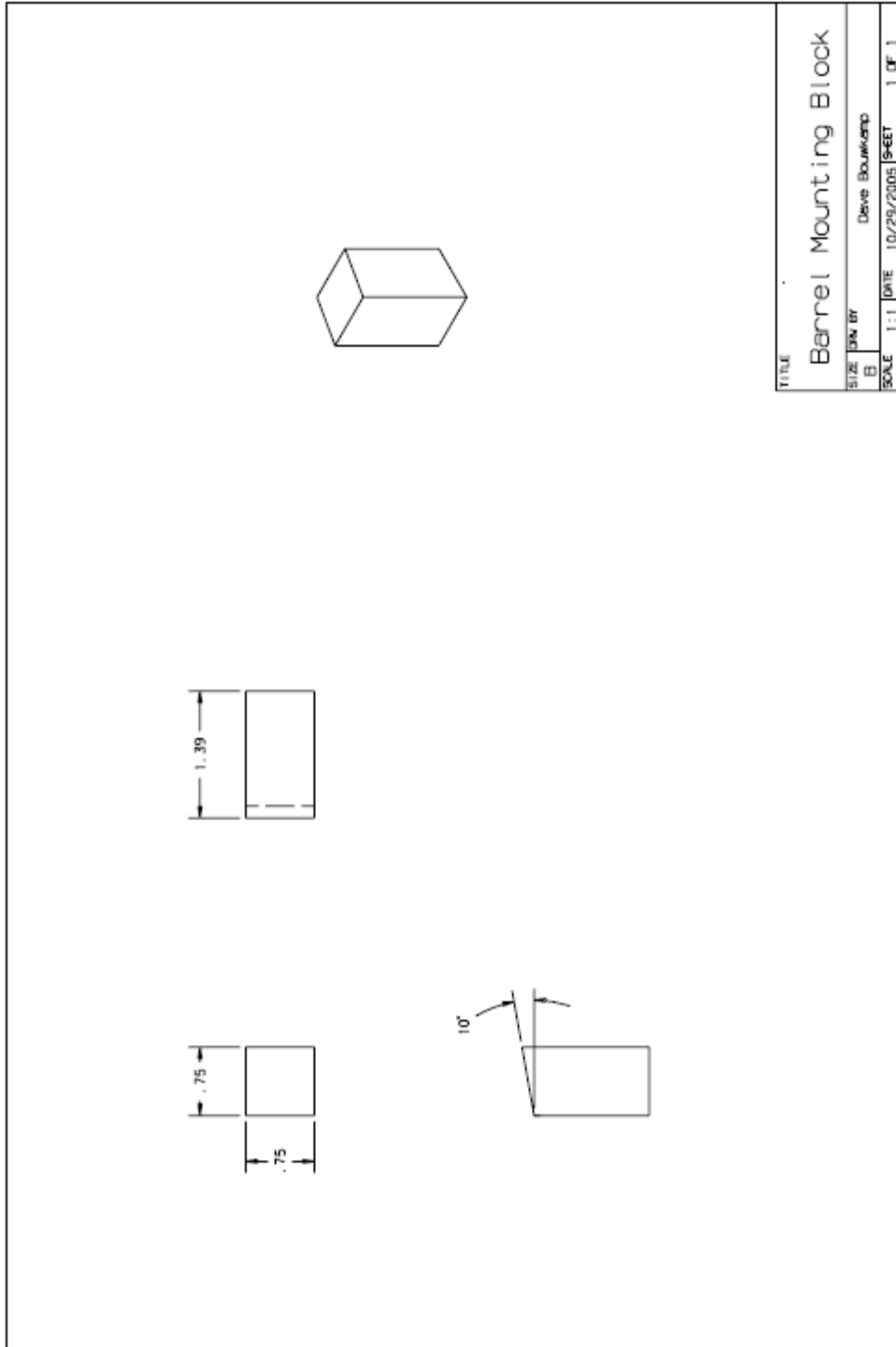
Appendix A: Drawings

Dimensioned drawings for individual components of the assembly are shown on the following pages.

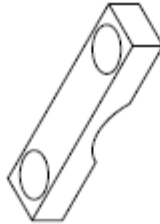
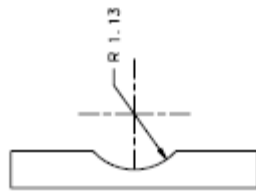
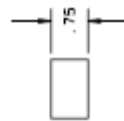
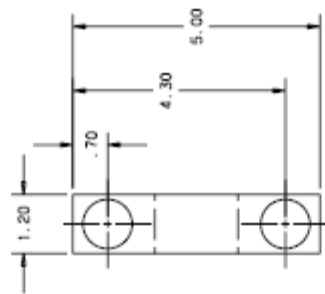




TITLE		Top Mounting Plate	
SIZE	DATE BY	Dave Bouwkamp	
E	DATE	10/29/2005	SHEET 1 OF 1
SCALE	1:1		



TITLE		Barrel Mounting Block	
SIZE	DATE BY	Deve Boukamp	
E			
SCALE	1:1	DATE	10/29/2005
SHEET		1 OF 1	



TITLE	Motor Mount For Barrel		
SIZE	3x4 BY	Drawn By	Deve Bouwkamp
SCALE	1:1	DATE	10/29/2005
		SHEET	1 OF 1

Appendix B: C Program

The C program used to control the Target Acquisition and Firing System during the competition is found on the following pages.

```

#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#include "sio.c"

#define CLK_ms 10 /* Set the updates for every 10ms */
#define c_kinetic_pos 0xa0 /* Static friction */
#define c_kinetic_neg 0xa0 /* Make the value positive */
#define c_static_pos 0xa0 /* Kinetic friction */
#define c_static_neg 0xa0 /* Make the value positive */
#define c_max 255
#define c_min 255 /* Make the value positive */
#define TABLE_SIZE 200 /* Size of the potentiometer position array */
#define Kp 5 /* Controller proportional gain constant */
#define Ki 0 /* Controller integral gain constant */

int pot_position[TABLE_SIZE] = {0}; /* Define and initialize tach_speed */
int array_position; /* Variable to define array position */
int db_correct = 1; /* Deadband correction is on by default */
int count = 128; /* A count value to be output on port B */
int moving = 0; /* Assume it starts without moving */
unsigned int CNT_timer1; /* The delay time */
volatile unsigned int CLK_ticks = 0; /* The current number of ms */
volatile unsigned int CLK_seconds = 0; /* The current number of seconds */
int Cf; /* Feedback position from potentiometer */
int test;

int deadband(int c_wanted);
void PWM_init();
void PWM_update(int value);
int v_output(int v_adjusted);
void IO_setup();
void IO_update();
void delay(int ticks);
void AD_setup(void);
int AD_read();
SIGNAL(SIG_OVERFLOW0);
void CLK_setup();

int deadband(int c_wanted)
{ /* Call this routine when updating */
    int c_pos;
    int c_neg;
    int c_adjusted;
    if(c_wanted > c_max) c_wanted = c_max;
    if(c_wanted < -c_min) c_wanted = -c_min;
    if(moving == 1){
        c_pos = c_kinetic_pos;
        c_neg = c_kinetic_neg;
    } else {
        c_pos = c_static_pos;
        c_neg = c_static_neg;
    }
    if(c_wanted == 0){ /* Turn off the output */
        c_adjusted = 0;
    } else if(c_wanted > 0){ /* A positive output */
        c_adjusted = c_pos + (unsigned)(c_max - c_pos) * c_wanted / c_max;
    } else { /* The output must be negative */
        c_adjusted = -c_neg - (unsigned)(c_min - c_neg) * -c_wanted / c_min;
    }
    return c_adjusted;
}

```

```

void PWM_init()
{
    /* Call this routine once when the program starts */
    DDRD |= (1 << PD5);          /* Set PWM outputs */
    DDRC |= (1 << PC0) | (1 << PC1);
    /* Set motor direction outputs on port C using OCR1 */
    TCCR1A = _BV(COM1A1) | _BV(COM1B1) | _BV(WGM10);
    /* Turn on both PWM outputs on counter 1 */
    TCCR1B = _BV(CS11);          /* Set the internal clock */
    DDRA = 0;                    //Set port to inputs
}

void PWM_update(int value)
{
    /* To update the PWM output */
    if(value > 255) value = 255;
    if(value < 0) value = 0;
    OCR1A = value;                /* Duty cycle */
}

int v_output(int v_adjusted)
{
    /* Call from the interrupt loop */
    int RefSignal;                /* The value to be returned */
    if(v_adjusted >= 0){
        /* Set the direction bits to CW on, CCW off */
        PORTC = (PINC & 0xFC) | 0x02;          /* Bit 1 on, 0 off */
        if(v_adjusted > 255){                  /* Clip output over maximum */
            RefSignal = 255;
        } else {
            RefSignal = v_adjusted;
        }
    } else {
        /* Need to reverse output sign */
        /* Set the direction bits to CW off,
        CCW on */
        PORTC = (PINC & 0xFC) | 0x01;          /* Bit 0 on, 1 off */
        if(v_adjusted < -255){                 /* Clip output below minimum */
            RefSignal = 255;
        } else {
            RefSignal = -v_adjusted;          /* Flip sign */
        }
    }
    return RefSignal;
}

int controller(int Cd, int Cf)
{
    int Ce;
    int Cw;

    Ce = Cd - Cf;
    Cw = Kp * Ce;

    return Cw;
}

void IO_setup()
{
    sio_init();
    PWM_init();
    AD_setup();
    CLK_setup();
}

```

```

void IO_update()
{
    /* This routine will run once per interrupt for updates */

    Cf = AD_read();

    outln("");

    outint(AD_read());

    if((PINA & _BV(PA2)) >= 1 ) //waits for voltage on target 1 to go high
    {
        count = 0x67;
    }
    else if((PINA & _BV(PA3)) >= 1 ) //waits for voltage on target 2 to go high
    {
        count = 0x77;
    }
    else if((PINA & _BV(PA4)) >= 1 ) //waits for voltage on target 3 to go high
    {
        count = 0x87;
    }
    else if((PINA & _BV(PA5)) >= 1 ) //waits for voltage on target 4 to go high
    {
        count = 0x97;
    }

    if(db_correct == 0){
        PWM_update(v_output(controller(count,Cf)));
    } else {
        PWM_update(v_output(deadband(controller(count,Cf))));
    }

    moving = (Cf != 0); // determines if motor is moving or not

    if(Cf == 0)
    {

}

void delay(int ticks)
{
    /* Ticks are approximately 1ms */
    volatile int i, j;
    for(i = 0; i < ticks; i++){
        for(j = 0; j < 1000; j++){
        }
    }
}

void AD_setup(void)
{
    /* Setup A/D to read tach input from channel 0 */
    ADMUX = 0xC1;          /* Set the input to channel 1 */
    ADCSRA = 0xE0;        /* Turn on ADC and set to free running */
}

int AD_read()
{
    return (ADCW >> 2);
}

SIGNAL(SIG_OVERFLOW0)
{
    /* The interrupt calls this function */
    CLK_ticks += CLK_ms;
}

```

```

        IO_update();
        if(CLK_ticks >= 1000){          /* The number of interrupts between output
changes */
            CLK_ticks = CLK_ticks - 1000;
            CLK_seconds++;
        }
        TCNT0 = CNT_timer1;
    }

void CLK_setup()
{
    /* Start the interrupt service routine */
    TCCR0 = (0<<FOC0) | (0<<WGM01) | (0<<WGM00) | (0<<COM00)
            | (0<<COM01) | (1<<CS02) | (0<<CS01) | (1<<CS00);
    /* Use CLK/1024 prescale value */
    /* Disable PWM and Compare Output Modes */
    CNT_timer1 = 0xFFFF - CLK_ms * 8; /* 8 = 1ms, 80 = 10ms */
    TCNT0 = CNT_timer1;          /* Start at the right point */
    TIFR |= (1<<TOV0);
    TIFR &= ~(0<<OCF0);
    /* TIFR = (0<<OCF2) | (0<<TOV2) | (0<<ICF1) | (0<<OCF1A) | (0<<OCF1B)
    | (0<<TOV1) | (0<<OCF0) | (1<<TOV0); // set to use overflow interrupts
*/
    TIMSK |= (1<<TOIE0);
    TIMSK &= ~(0<<OCIE0);
    /* TIMSK = (0<<OCIE2) | (0<<TOIE2) | (0<<TICIE1) | (0<<OCIE1A) |
(0<<OCIE1B)
    | (0<< TOIE1 ) | (0<<OCIE0) | (1<<TOIE0); // enable TCNT1 overflow */
    sei(); /* Enable interrupts flag */
}

int main()
{
    int c, i;
    int desired_speed = 0; /* desired motor speed (for table output) */

    IO_setup();

    outln("Tachometer Speed Data Acquisition");
    for(;;){
        while((c = input()) == -1){} /* Wait for a keypress */

        if(PA2 == 1 ){ /* Increment the output on
port B */
            count = 0x67; /* Set PWM value */
            desired_speed = count; /* Store desired pwm value */

            outint(count); outln(" = is the PWM.");
        }else if(PA3 == 1 ){ /* Increment the
output on port B */
            array_position = 0; /* Initialize the array
position variable so
data */
            count = 0x77; /* Set PWM value */
            desired_speed = count; /* Store desired pwm value */

            outint(count); outln(" = is the PWM.");
        }else if(PA4 == 1 ){ /* Increment the
output on port B */
            array_position = 0; /* Initialize the array
position variable so
data */
            count = 0x87; /* Set PWM value */

```

```

        desired_speed = count;          /* Store desired pwm value */
        outint(count); outln(" = is the PWM.");
    }else if(PA5 ==1){                  /* Increment the
output on port B */                    array_position = 0;          /* Initialize the array
position variable so                    interrupts start taking
data */                                  count = 0x97;          /* Set PWM value */
        desired_speed = count;          /* Store desired pwm value */
        outint(count); outln(" = is the PWM.");
    }else if(c == '0'){                 /* Increment the
output on port B */                    array_position = 0;          /* Initialize the array
position variable so                    interrupts start taking
data */                                  count = 0x67;          /* Set PWM value */
        desired_speed = count;          /* Store desired pwm value */
        outint(count); outln(" = is the PWM.");
    } else if(c == '1'){                /* Increment the output on
port B */                               array_position = 0;          /* Initialize the array
position variable so                    interrupts start taking
data */                                  count = 0x67;          /* Set PWM value */
        desired_speed = count;
        outint(count); outln(" = is the PWM.");
    }else if(c == '2'){                /* Increment the output on
port B */                               array_position = 0;          /* Initialize the array
position variable so                    interrupts start taking
data */                                  count = 0x77;          /* Set PWM value */
        desired_speed = count;
        outint(count); outln(" = is the PWM.");
    }else if(c == '3'){                /* Increment the output on
port B */                               array_position = 0;          /* Initialize the array
position variable so                    interrupts start taking
data */                                  count = 0x97;          /* Set PWM value */
        desired_speed = count;
        outint(count); outln(" = is the PWM.");
    }else if(c == '4'){                /* Increment the output on
port B */                               array_position = 0;          /* Initialize the array
position variable so                    interrupts start taking
data */                                  count = 0xa7;          /* Set PWM value */
        desired_speed = count;
        outint(count); outln(" = is the PWM.");

```

```

        }else if(c == '5'){
            /* Increment the output on
port B */
            array_position = 0;
            /* Initialize the array
position variable so
data */
            count = 176;
            desired_speed = count;
            /* Set PWM */
            outint(count); outln(" = is the PWM.");
        }else if(c == '6'){
            /* Increment the output on
port B */
            array_position = 0;
            /* Initialize the array
position variable so
data */
            count = 224;
            desired_speed =count;
            outint(count); outln(" = is the PWM.");
        }else if(c == '7'){
            /* Increment the output on
port B */
            array_position = 0;
            /* Initialize the array
position variable so
data */
            count = 240;
            desired_speed = count;
            outint(count); outln(" = is the PWM.");
        }else if(c == '8'){
            /* Increment the output on
port B */
            array_position = 0;
            /* Initialize the array
position variable so
data */
            count = 96;
            desired_speed = count;
            outint(count); outln(" = is the PWM.");
        }else if(c == '9'){
            /* Increment the output on
port B */
            array_position = 0;
            /* Initialize the array
position variable so
data */
            count = 127;
            desired_speed = count;
            outint(count); outln(" = is the PWM.");
        } else if(c == 'c'){
            if(db_correct == 0){
                db_correct = 1;
                outln("deadband correction ON");
            } else {
                db_correct = 0;
                outln("deadband correction OFF");
            }
        }
        IO_update();
    }else if (c == 'd'){
        outln("Desired Motor Speed (Hex) Potentiometer Position Volts
(Hex)\n");
        for(i = 0; i < 200; i++){
            outint(pot_position[i]); outln("");
        }
    }
}

```

```

    }
}else if(c == 'r'){
    count = 0;
    moving = 0;
    array_position = 1000;          /* Set large array position
to endure                          no data being taken */
    outint(count); outln(" = count, motor stopped");
}else if(c == 'f'){
    outint(Cf); outln(" = Pot Position");
} else if(c == 'h'){
    outln(" 1 - 9 PWM value between -128 and 127");
    outln(" d: dumps data in array to screen");
    outln(" c: enable or disable (default) deadband comp.");
    outln(" r: stop the motor");
    outln(" f: display pot position");
    outln(" q: quit");
} else if(c == 'q'){
    count = 128;
    moving = 0;
    outint(count); outln(" = count, Quitting!");
    break;
}
}

sio_cleanup();
return 1;
}

```

Appendix C: MatLab Simulation

The analysis of the projectile motion assumes a two-dimensional, x-y coordinate system, where x is the horizontal distance of the motion and y is the vertical distance of the motion. The motion is modeled by four differential equations shown as equations 1 through 4. The derivations of these equations are shown in the Appendix.

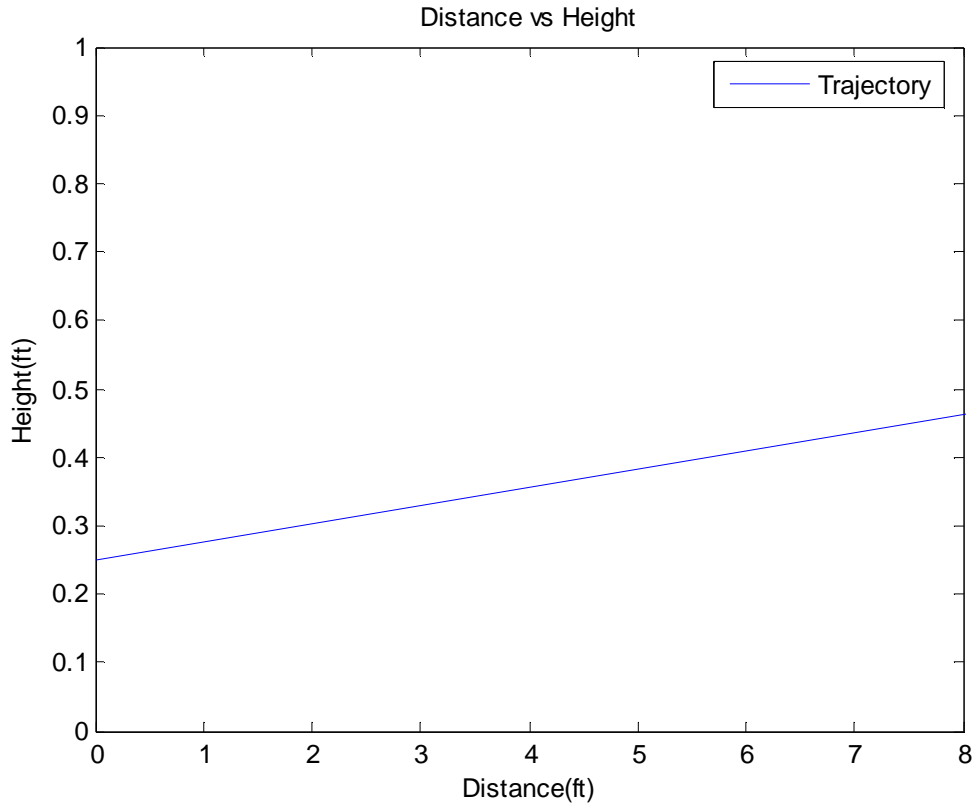
$$u = \frac{dx}{dt} \tag{1}$$

$$v = \frac{dy}{dt} \tag{2}$$

$$\frac{du}{dt} = -\frac{C_D \text{ Re}}{24\tau} u \tag{3}$$

$$\frac{dv}{dt} = -\frac{C_D \text{ Re}}{24\tau} v - g \tag{4}$$

Max Distance 20.76 ft
Max Height 0.79 ft
Max X-dir Velocity 7.76 ft/sec



DERIVATION OF DIFFERENTIAL EQUATIONS FOR PROJECTILE MOTION SIMULATION

First we will solve for an explicit solution for distance in the x direction.

Rewriting (A14) gives us:

$$U' = \frac{-C_D \cdot \text{Re}}{24 \cdot \tau} \cdot U \quad (\text{A1})$$

Putting this equation in a form organized for solving:

$$U' + \frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot U = 0 \quad (\text{A2})$$

This is a first order homogeneous equation. The general form for this type of differential equation is:

$$U_h(t) = A \cdot e^{B \cdot t} \quad (\text{A3})$$

Differentiate this equation:

$$U'_h(t) = A \cdot B \cdot e^{B \cdot t} \quad (\text{A4})$$

Plug these back into (A23) and set time equal to 0:

$$B + \frac{C_D \cdot \text{Re}}{24 \cdot \tau} = 0 \quad (\text{A5})$$

Solve for B:

$$B = -\frac{C_D \cdot \text{Re}}{24 \cdot \tau} \quad (\text{A6})$$

This is inserted back into (A23) to find the homogeneous solution.

$$\therefore U_h(t) = C_1 \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot t} \quad (\text{A7})$$

We can solve for the constant in (A28) by plugging in an initial condition. Using equation (A19) in the x direction:

$$U(0) = x(0) = V_0 \cdot \cos(\theta) \quad (\text{A8})$$

Inserting this information into equation (A28) yields:

$$V_0 \cdot \cos(\theta) = C_1 \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot t} \quad (\text{A9})$$

This can be solved for C_1 .

$$C_1 = V_0 \cdot \cos(\theta) \quad (\text{A10})$$

C_1 is now added to (A28) and yields an explicit solution:

$$\boxed{\therefore U(t) = V_0 \cdot \cos(\theta) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot t}} \quad (\text{A11})$$

We know that:

$$U = \frac{dx}{dt} \quad (\text{A12})$$

Plug this into equation (A32):

$$\frac{dx}{dt} = V_0 \cdot \cos(\theta) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} \quad (\text{A13})$$

Multiplying through by dt yields:

$$dx = V_0 \cdot \cos(\theta) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} \cdot dt \quad (\text{A14})$$

This can be integrated to find x as a function of time.

$$\int dx = x(t) = \int_0^t V_0 \cdot \cos(\theta) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} \cdot dt \quad (\text{A15})$$

Integrating the right side (A36) we get the explicit solution for distance as a function of time in the x direction:

$$\therefore x(t) = \left(\frac{-V_0 \cdot \cos(\theta) \cdot 24 \cdot \tau}{C_D \cdot \text{Re}} \right) \cdot e^{\left(\frac{-C_D \cdot \text{Re}}{24 \cdot \tau} \right) t} + \left(\frac{V_0 \cdot \cos(\theta) \cdot 24 \cdot \tau}{C_D \cdot \text{Re}} \right) \quad (\text{A16})$$

Next, we will solve for an explicit solution for distance in the y direction.

$$V' = \frac{-C_D \cdot \text{Re}}{24 \cdot \tau} \cdot V - g \quad (\text{A17})$$

Rewriting the equation for solving a differential equation:

$$V' + \frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot V = -g \quad (\text{A18})$$

We need to solve for the homogenous solution so we set the equation equal to 0:

Homogeneous Solution $V_h(t)$:

$$V_h(t) = A \cdot e^{Bt} \quad (\text{A19})$$

Differentiate (A40)

$$V_h'(t) = A \cdot B \cdot e^{Bt} \quad (\text{A20})$$

Plug these back into (A39) and set time equal to 0:

$$B + \frac{C_D \cdot \text{Re}}{24 \cdot \tau} = 0 \quad (\text{A21})$$

Solving for B yields:

$$B = -\frac{C_D \cdot \text{Re}}{24 \cdot \tau} \quad (\text{A22})$$

This is inserted into (A40):

$$\therefore V_h(t) = C_1 \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} + C_2 \quad (\text{A23})$$

Next we need a particular solution to solve for C_2

$$V_p(t) = C_2 \quad (\text{A24})$$

Differentiating (A45) yields:

$$V_p'(t) = 0 \quad (\text{A25})$$

Inserting (A45) and (A46) into (A44):

$$0 + \frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot C_2 = -g \quad (\text{A26})$$

Solving for C_2 yields:

$$C_2 = \frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} \quad (\text{A27})$$

Inserting C_2 into (A44) gives us:

$$\therefore V(t) = C_1 \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} + \frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} \quad (\text{A28})$$

We can now solve for C_1 using an initial condition. Using equation (A21) in the y direction gives us:

$$V(0) = y(0) = V_0 \cdot \sin(\theta) - g \cdot t \quad (\text{A29})$$

Inserting this information into (A49):

$$V_0 \cdot \sin(\theta) - 3 \cdot g \cdot 0 = C_1 \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} \cdot 0} + \frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} \quad (\text{A30})$$

Solve for C_1 .

$$C_1 = V_0 \cdot \sin(\theta) + \frac{24 \cdot \tau \cdot g}{C_D \cdot \text{Re}} \quad (\text{A31})$$

This yields the explicit solution in terms of velocity:

$$\therefore V(t) = \left(V_0 \cdot \sin(\theta) + \frac{24 \cdot \tau \cdot g}{C_D \cdot \text{Re}} \right) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} + \left(\frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} \right) \quad (\text{A32})$$

Next, we can solve for the explicit solution for position in the y direction. We know that:

$$V = \frac{dy}{dt} \quad (\text{A33})$$

Inserting (A54) into equation (A53):

$$\frac{dy}{dt} = \left(V_0 \cdot \sin(\theta) + \frac{24 \cdot \tau \cdot g}{C_D \cdot \text{Re}} \right) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} + \left(\frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} \right) \quad (\text{A34})$$

This can be integrated to find the height as a function of time.

$$\int dy = \int_0^t \left[\left(V_0 \cdot \sin(\theta) + \frac{24 \cdot \tau \cdot g}{C_D \cdot \text{Re}} \right) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} + \left(\frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} \right) \right] dt \quad (\text{A35})$$

Separating the right side gives us:

$$y(t) = \int_0^t \left(V_0 \cdot \sin(\theta) + \frac{24 \cdot \tau \cdot g}{C_D \cdot \text{Re}} \right) \cdot e^{-\frac{C_D \cdot \text{Re}}{24 \cdot \tau} t} dt + \int_0^t \frac{-C_D \cdot \text{Re} \cdot g}{24 \cdot \tau} dt \quad (\text{A36})$$

Integrate the function:

$$y(t) = \left[\frac{-24 \cdot \tau}{C_D \cdot \text{Re}} \cdot \left(V_0 \cdot \sin(\theta) + \left(\frac{24 \cdot g \cdot \tau}{C_D \cdot \text{Re}} \right) \right) \right] + \left[\frac{-24 \cdot \tau}{C_D \cdot \text{Re}} \cdot \left(V_0 \cdot \sin(\theta) + \left(\frac{24 \cdot g \cdot \tau}{C_D \cdot \text{Re}} \right) \right) \cdot e^{\left(\frac{-C_D \cdot \text{Re}}{24 \cdot \tau} \right) t} \right] - \left(\frac{24 \cdot g \cdot \tau}{C_D \cdot \text{Re}} \right) \cdot t \quad (\text{A37})$$

We have an explicit solution for distance as a function of time in the y direction.

MATLAB PROGRAM USED FOR SIMULATION

%CONSTANT FRICTION

```

% Test Golf ball with Constant friction - Part 3
% Initializing Vectors and Specifying Constants
i = 1;
X = [];
Y = [];
U = [];
V = [];
X(i) = 0;
Y(i) = .25;
U(i) = 0;
V(i) = 0;
d = 1.75/12;
g = 32.2;
m = 0.000310559;
u = 0.000000375;
dt = 0.04;
t = 1;
Vo = 45.21;           // initial velocity of ball
Theta = pi/12;
CD = 0.25;
Re = 90000;
Xmax = 0;
Ymax = 0;
Umax = 0;
Vmax = 0;

%Calculate tao
tao = (m/(3*pi*d*u));

%Provide Time Sp n of Integration
while Y >= eps      % Before Ball hits the ground
    i = i + 1;
    t = t + dt;

    %X direction Position
    X(i) = (-Vo*cos(Theta)*24*tao)/(CD*Re)*exp((-CD*Re)/(24*tao)*t)-Vo*cos(Theta)*24*tao/(CD*Re);

    %X direction Velocity
    U(i) = Vo*cos(Theta)*exp((-CD*Re)/(24*tao)*t);

    if U(i) > U(i-1)
        Umax = U(i);
    end

    %Y direction Position
    Y(i) = (-24*tao*(Vo*sin(Theta)+(24*g*tao)/(CD*Re)))/(CD*Re)*exp((-CD*Re)/(24*tao)*t)-...
    (-24*tao*(Vo*sin(Theta)+(24*g*tao)/(CD*Re)))/(CD*Re)-(24*g*tao)/(CD*Re)*t;

    if Y(i) > Y(i-1)
        Ymax = Y(i);
    end
end

```

```

%Y direction Velocity
V(i) = (Vo*sin(Theta)+(24*g*tao)/(CD*Re))*exp((-CD*Re)/(24*tao)*t)-(24*g*tao)/(CD*Re);

if V(i) > V(i-1)
    Vmax = V(i);
end

end

fprintf('Results of Part 3: \n');
fprintf('Max Distance %-5.2f ft \n', X(i));
fprintf('Max Height %-5.2f ft \n', Ymax);
fprintf('Max X-dir Velocity %-5.2f ft/sec \n', Umax);
fprintf('Max Y-dir Velocity %-5.2f ft/sec \n', Vmax);

%*****Plot the Results*****

plot(X,Y)
Title('Distance vs Height')
axis([0,8,0,1])
xlabel('Distance(ft)')
ylabel('Height(ft)')
legend('Trajectory')

```

