

EGR 345 Team Project

Final Report

December 7, 2005

Team Members:

Jonathan Kolb

Ryan Parent

Nicole Vander Velden

Luke Koops

GVSU School of Engineering
EGR 345 – Dynamic Systems Modeling
Prof. H Jack

TABLE OF CONTENTS

Project Objective.....	3
Design Constraints	3
Design Breakdown.....	4
Base Frame.....	4
Barrel	5
Turret	5
Motor & Gearbox	6
Air Valve & Air Lines	6
Hopper	6
Loading Chamber	6
Bill of Materials	7
Test Results.....	8
Mechanical Test 1.....	8
Mechanical Test 2.....	8
Control Test 1.....	8
Control Test 2.....	8
Firing Chamber Collar Motion	9
Barrel and Turret Rotational Motion.....	10
Electrical Schematic.....	10
Design Problems and Solutions	11
Conclusions	12

Appendix A – Project Budget

Appendix B – Control Program

Appendix C – Pictures of Actual Robot

Appendix D – Pro-Engineer Drawings

The objective of this project was to develop an automated device to fire wiffle golf balls at four randomly activated targets. At separate times each target sent out an electrical signal indicating when the target was active. Once the device recognized the active target it had to aim, fire, and reload. The device needed to be lightweight, cost effective, and structurally rigid while maximizing the number of targets hit.

Design Constraints

Some design constraints had to be followed when designing the target acquisition and firing system. The total weight of the machine must be less than 3 [Kg]. The entire machine must fit inside a one foot cube. The machine must be capable of firing 1.5 [in] practice golf balls. The machine can be controlled by a combination of pneumatic and electrical components. Once complete the only thing that may pass into the team area, during competition, is wires and airlines. The device must operate in a safe manner for the protection of the people observing the competition, and the equipment that controls the competition. The judges will make the final safety decision before the competition. Any unsafe entries will be disqualified. There is maximum budget of \$200.00 that may be spent during the production of the device. Records of all purchases must be kept for budget verification. There were incentives placed into the final score to encourage the minimization of some if the design requirements, such as: total cost, longest dimension, and total weight.

Design Breakdown

This section describes the main features of the device. Each feature is described based on prevalent design factors and contributing function to the automated system. The number on Figure 1 corresponds to the numbered item.

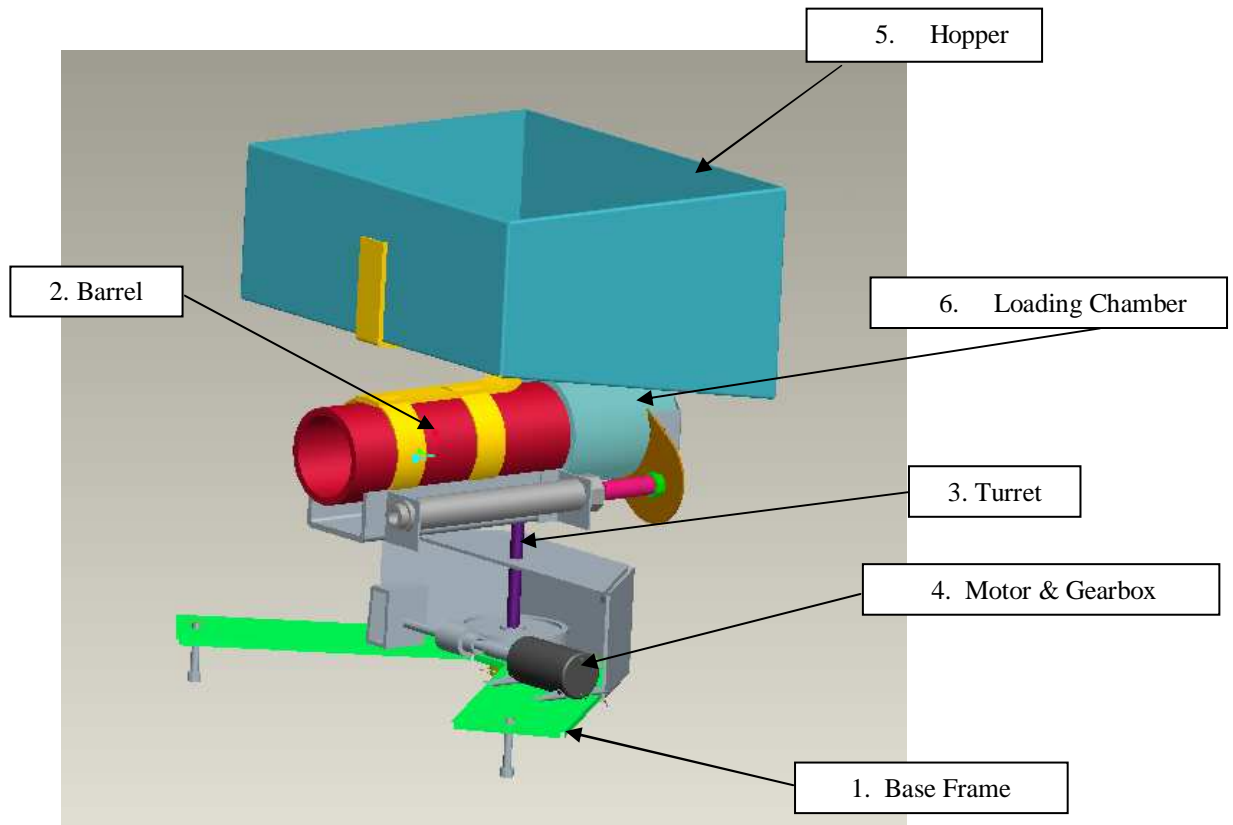


Figure 1 – Isometric view showing the main features of the automated wiffle golf ball device.

1. Base Frame

The base frame supports the device and holds all acting features together. The base frame was designed as a V shape. This shape enabled the gun to have stability from side to side and front to back. The vertex of the ‘V’ was an ideal position for the turret axle and gearbox to be mounted while the motor and control boards could fasten directly to the legs of the ‘V’. The base frame was made of

aluminum and was composed of three separate pieces fastened together to form the 'V' shape. Mechanically fastening pieces together rather than welding them gave flexibility for later design changes.

2. Barrel

The barrel guides the ball as it is pushed by high pressure air. The barrel of the device was made of PVC pipe. Pipe is purchased by inside dimensions (rather than outside dimensions). The inside diameter of the barrel was originally 1.56" I.D. (slightly bigger than a golf ball diameter) but was later bored out to allow for oversized wiffle golf balls. The final inside diameter of the barrel was 1.64". PVC material is rigid and lightweight yet convenient to purchase. Using pre-manufactured parts saved cost and time when developing the device. A large hole (1.64+ inch diameter) was drilled through the wall of the barrel to allow only one golf ball to load at a time. The barrel was fastened to an aluminum C-channel to allow other parts of the device to be securely attached. For example, the loading chamber air cylinder.

3. Turret

The turret mechanically rotates the barrel on a horizontal plane of motion. The turret design consists of an axle, worm, worm gear and potentiometer. The vertical axle, driven by the worm and worm gear setup, was attached to the barrel C-channel. When the motor was activated, the vertical axle was rotated and the barrel would move left or right. The rotation of the vertical axle was tracked by a

potentiometer which gave feedback to the microprocessor. Mounting the potentiometer directly to the rotating vertical axle eliminated the need for more gears or other hardware. The microprocessor in turn directed the turret to the correct position for firing. The worm and worm gear were made of nylon material whereas the vertical axle was made of 0.25” diameter aluminum rod.

4. Motor & Gearbox

The motor and gearbox turned the turret. The 5V motor received a signal from the microprocessor and was controlled by the potentiometer mounted on the turret. The gearbox allowed for high torque and slow RPM necessary for turning the worm and worm gear. The gearbox was made of plastic, ideal for a mass critical design.

5. Hopper

The hopper was used to hold and feed wiffle golf balls into the loading chamber. The hopper design needed to prevent golf balls from bridging and had to centralize the exiting point above the loading chamber. The hopper was fastened to the barrel to allow the hopper to rotate as the barrel did. The maximum number of balls allowed during competition was 30 balls. The hopper was made of aluminum sheet at 0.011” thickness, a common thickness for printing plates used by industrial printers. The abrupt motion of the air cylinder prevented golf balls from bridging inside the hopper.

6. Loading Chamber

The loading chamber was used to implement the loading of wiffle golf balls. The loading chamber consisted of two basic parts: a collar and an air cylinder. The air

cylinder was rigidly mounted to the barrel and the air cylinder was fastened to the free sliding collar. The inside diameter of the collar was slightly larger than the outside diameter of the barrel; this allowed the collar to translate along the length of the barrel with minimal friction. When the collar covered the large hole in the barrel the balls from the hopper could not enter the barrel. But when the 9/16” bore air cylinder was activated, the collar would reveal the hole and one ball would drop inside. The cylinder could only be activated for less than 1.5 seconds; else golf balls inside the hopper would jam between the collar and the bottom edge of the hopper. The air cylinder had a maximum stroke length of 1.75”, just enough stroke for clearance between the balls and the collar. An air adapter was connected to the collar as well. This quick connect adapter was where the given air line from the target range could be connected.

Bill of Materials

The bill of materials and exploded assembly of the device are shown in Table 1 & Figure 2.

ITEM	PART NUMBER	DESCRIPTION	MATERIAL	QTY
1	PF-012	BARREL PIPE, 1-1/2"	PVC	1
2	P-016	MOTOR, ELECTRIC	-	1
3	P-007	COUPLING, 1-1/2" SC	PVC	1
4	P-001	ADAPTER, AIR	STEEL	1
5	FS-019	FRAME, BASE	ALUMINUM	1
6	P-025	POTENTIOMETER	-	1
7	PS-015	HOPPER	ALUMINUM	1
8	P-023	CYLINDER, AIR	STEEL	1
9	P-016	GEAR	PLASTIC	1
10	PF-018	WORM	PLASTIC	1
11	PF-021	SUPPORT, GUN BARREL	ALUMINUM	1
12	PF-022	AXLE, GUN BARREL	ALUMINUM	1
13	PF-025	BRACKET, COLLAR	ALUMINUM	1
14	PF-009	STRAP, 3/4" WIDTH H	COPPER	3
15	PF-020	FRAME, BACK SUPPORT	ALUMINUM	3
16	PF-024	BRACKET, COLLAR	ALUMINUM	1
17	P-026	SCREW, FOOT	-	4

Table 1 – Bill of materials

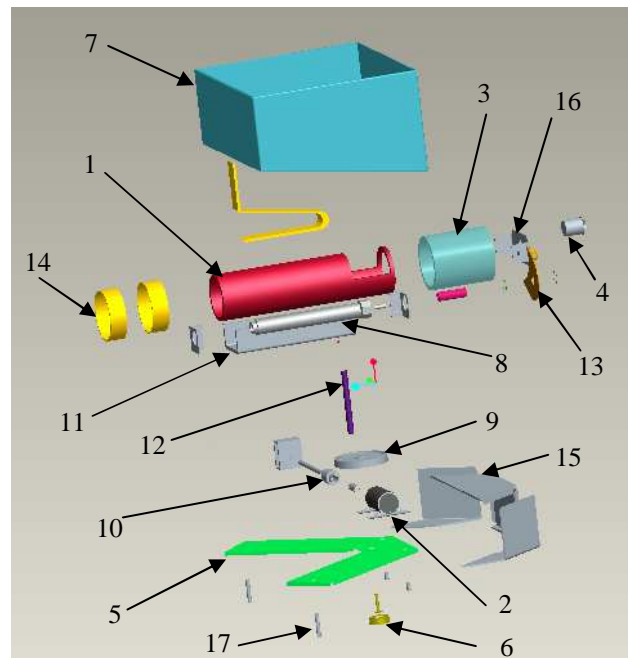


Figure 2 – Exploded assembly view

Test Results

Mechanical Test 1:

An air reservoir was built with a 2" diameter and 2" length. The volume of air was found to be excessive – the ball was fired with an extremely high velocity. Therefore, the size of the tank could be reduced to slow the projectiles velocity.

Mechanical Test 2:

The tank size and weight was reduced to a 0.625" ID hose with a length of 3". The ball still fired quickly with a more acceptable velocity. The new reservoir design was accepted.

Control Test 1:

The control program was modified to test the fire and reload functions by depressing buttons on the keyboard. Prior to attaching to the competition PLC the fire and reload functions operated correctly. However, the PLC did not receive enough voltage to activate the valves.

Control Test 2:

The device was inspected by judges and was given an approximate score value. The judges recommended changes to the hopper to remove duct tape and hammer imprints. The device was tracking targets but not firing golf balls at the time.

Firing Chamber Collar Motion

Equations of motion were derived to analyze the motion of the firing chamber when it is controlled by an air cylinder. The cylinder chosen for controlling the motion was a spring return nose-mounted air cylinder with a push force of 15 [lbs] and a pull force of 12 [lb].

$$\begin{aligned}M_{\text{collar}} &= 1/3 \text{ [lb]} \\F_{\text{cylinder (push)}} &= 15 \text{ [lb]} \\F_{\text{cylinder (pull)}} &= 12 \text{ [lb]} \\\mu_{\text{pvc}} &= 0.5\end{aligned}$$

The equation representing the forward (pushed by cylinder) motion of the collar is (1). The equation representing the reverse (pulled by cylinder) motion is (2). NOTE: (2) was adjusted to fit the profile of curve in Figure 3.

$$x_{\text{push}}(t) = 2.417 \cdot t^2 \tag{1}$$

$$x_{\text{pull}}(t) = -2.083 \cdot (t - 2)^2 + 2.5 \tag{2}$$

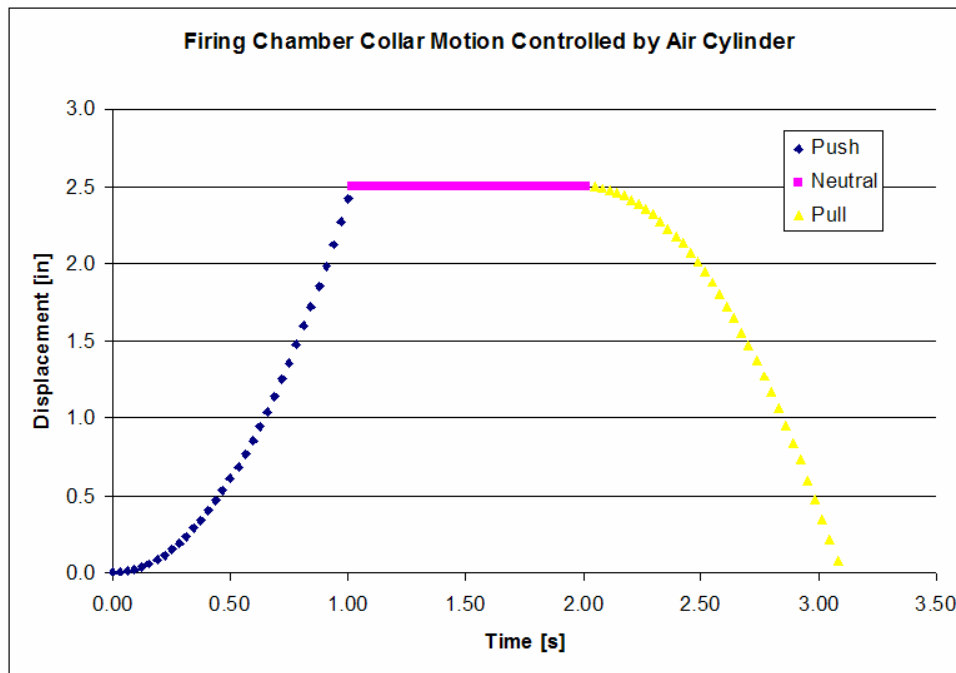
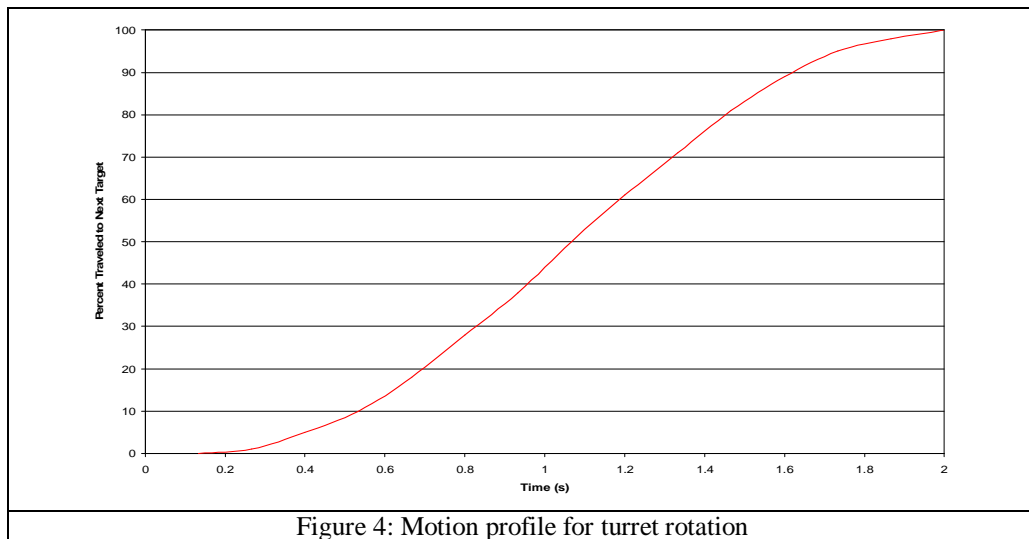


Figure 3 – Graphing showing the motion of the firing chamber collar as it is pushed and pulled by an air cylinder. The neutral position of the collar allows one wiffle golf ball to load.

Barrel and Turret Rotational Motion

Figure 4 shows the rotational motion of the barrel and turret of the firing device. If the turret rotates too fast the ball will climb up the barrel. Therefore, the turret rotation has to be smooth and controlled. A proportional and integral motion controller was used to control the movement of the turret.



Electrical Schematic

Figure 5 below is an electrical schematic of the firing device. The schematic shows the operating devices as they are connected to ATmega 32 thumb board. The pins used in the design structure are proposed.

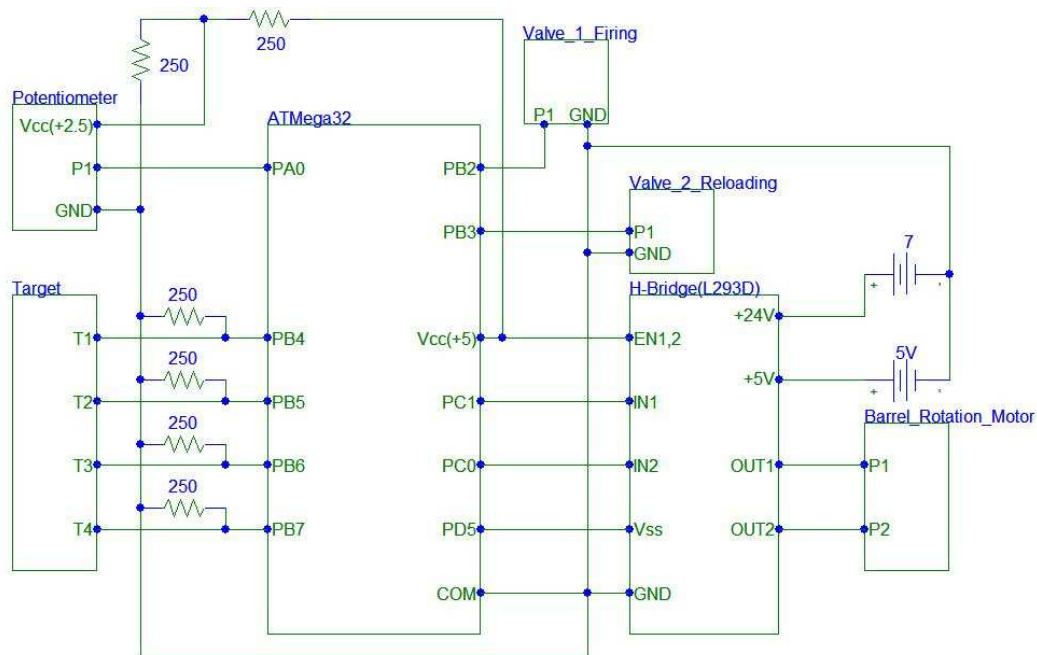


Figure 5 – Electrical schematic of firing device

Design Problems and Solutions

This section describes key problems with the proposed design that arose during the production of the device.

Problem: The proposed mounting location and method for the air valve used for firing did allow the barrel to rotate far enough to reach the end targets.

Solution: The valve was mounted to the turret allowing it to rotate with the barrel, which eliminated the extra torque caused by the airline.

Problem: The balls used in the competition are not all the exact same size. The barrel was designed around one of the smaller balls meaning that the larger balls get jammed in the barrel.

Solution: The barrel had to be capable of firing any one of the balls without a problem, so measurements were taken of thirty competition balls. The largest ball found had a

diameter of 1.62". Therefore, the barrel was bored out to 1.64in. This allows for a ball that is 0.02" larger than the largest ball found to be fired through the gun.

Problem: The encoder to be used for positioning the barrel did not operate correctly. It would travel 4 times farther in the one direction than in the other when set to an equal displacement.

Solution: The encoder was mounted to the end of the motor shaft. The encoder was changed to a potentiometer but could not travel far enough when attached to the motor. The potentiometer was moved underneath the device and attached to the turret so that it only had to move as far as the barrel. Moving the potentiometer also reduced the amount of power used from the motor to turn it. This is because the encoder was attached directly to the motor shaft. Since the potentiometer is attached to the turret there is a gear reduction in place between the motor and potentiometer.

Problem: The air valve and air lines from the original design were causing problems with turret rotation. In addition, the velocity at which the air valve projected the golf ball was too high for the given target setup. That is, there was a possibility that wiffle golf balls could bounce out of the targets.

Solution: The air valve and the large diameter air lines were removed from the design and replaced by a single 1/4" diameter air line attached directly to the back of the barrel. This reduced the overall mass of the device and was a satisfactory means for projecting the golf ball to the targets.

Score Calculation

The scoring is based on six different parameters. The target value for each parameter was determined by maximizing the other parameters and analyzing the impact it had on the score.

The parameters as well as the target values and actual values are shown below in Table 2.

	Target Value	Actual Value
H - # of targets hit:	20	10
C - Total Cost:	75	107.51
L - Largest Dimension (in):	6	15
B - Build Quality (0 - 1):	1	0.7
T - Theory quality (0 -1):	1	0.7
M - Mass (kg):	1	1.395
Score:	0.000366802	0.310842347

Table 2 - Score calculation comparison of target vs. actual parameter values.

The actual score is based on an estimated 10 targets hit. The theory quality is base on the rough draft estimate and the build quality is based on a previous score. The actual value should increase once the parameters are finalized.

Conclusions

1. The design of the automated golf ball firing device was successful in meeting the restrictive design specifications.
2. The device was below both the maximum price level and maximum mass and was contained within a 12” cube.

Appendix A – Project Budget

2005 BUDGET - TEAM #5 EGR 345 PROJECT					As of:	7-Dec-05		
Part Numbering Code:								
	P - (P)urchased Part							
	F - Part Requires (F)abrication							
	S - Part Requires (S)ubassembly							
Part No.	Part Description	Material	Qty	Mass [lb] @ Qty	Cost @ Qty	Fabrication?	Purchased?	Arrived?
P-001	ADAPTER, AIR	STEEL	1	0.1	\$ 0.50	NO	YES	YES
P-002	CLAMP, WORM GEAR	S.S.	4	0.1	\$ 2.86	NO	YES	YES
PF-007	COUPLING, 1-1/2" SCH 40	PVC	1	0.135	\$ 0.65	FAB'D	YES	YES
PF-009	STRAP, 3/4" WIDTH HANGER	COPPER	4FT	0.29	\$ 1.20	YES	YES	YES
PF-012	PIPE, 1-1/2" ID PVC (BARREL)	PVC	8in	0.1	\$ 0.12	FAB'D	YES	YES
P-013	BOARD, ATMEGA 32 CONTROLLER	PLASTIC	1	0.05	\$ 30.00	NO	YES	YES
PF-014	BOARD,L298 4 CHNL DRIVER	PLASTIC	1	0.06	\$ 18.00	NO	YES	YES
PS-015	HOPPER	ALUMINUM	1	0.1	\$ 2.00	YES	YES	YES
P-016	MOTOR, ELECTRIC	STEEL	1	0.155	\$ 10.00	NO	NO	YES
P-017	GEAR, WORM	NYLON	1	0.05	\$ 2.29	NO	YES	YES
PF-018	WORM	NYLON	1	0.05	\$ 6.16	NO	YES	YES
FS-019	FRAME, BASE	ALUMINUM	1	0.176	\$ 4.00	YES	YES	YES
PF-020	FRAME, BACK SUPPORT	ALUMINUM	1	0.4	\$ 2.00	YES	YES	YES
PF-021	SUPPORT, GUN BARREL	ALUMINUM	1	0.4	\$ 2.00	YES	NO	YES
PF-022	AXLE, GUN BARREL	ALUMINUM	1	0.3	\$ 1.00	FAB'D	YES	YES
P-023	CYLINDER, AIR (for chamber)	STEEL	1	0.4	\$ 20.00	NO	YES	YES
PF-024	BRACKET, COLLAR	ALUMINUM	1	0.1	\$ 1.00	YES	YES	YES
P-025	POTENTIOMETER	-	1	0.05	\$ 3.61	YES	YES	YES
P-026	SCREW, FOOT	-	4	0.02	\$ 0.12	NO	NO	YES
			lbs	3.036	\$ 107.51	TOTALS		
			Kg	1.38				

Appendix B – Control Program

```
#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 0x82 /* Static friction */
#define c_kinetic_neg 0x90 /* Make the value positive */
#define c_static_pos 0xa5 /* Kinetic friction */
#define c_static_neg 0xd8 /* 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 2 /* Controller proportional gain constant */
#define Ki 0 /* Controller integral gain constant */

int P1 = 0x4f; /* Position of target 1 */
int P2 = 0x47; /* Position of target 2 */
int P3 = 0x40; /* Position of target 3 */
int P4 = 0x31; /* Position of target 4 */
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 off by default */
int count = 0x4f; /* 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 */

// Function Initalizations
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 */
}

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() - 128;
    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(array_position < 200){ /* Populate the array for 1 second */
        pot_position[array_position] = Cf;
        array_position++;
    }else{
        count += 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 = 0xC0;          /* Set the input to channel 0 */
    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);
    TIMSK |= (1<<TOIE0);
    TIMSK &= ~(0<<OCIE0);
    sei(); /* Enable interrupts flag */
}

void fire()
{
    PORTB = PINB | 0x04; /* opens firing valve (on PA3) */
    delay(115); /* firing valve open for 250ms delay(ms)*0.461538 */
    PORTB = PINB & ~0x04; /* closes firing valve */
    delay(500);
}

void reload()
{
    PORTB = PINB | 0x08; /* opens reloading door (on PA2) */
    delay(100); /* holds door open for 215ms delay(ms)*0.461538 */
    PORTB = PINB & ~0x08; /* closes reloading door */
    delay(100);
}

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

    DDRB = 0x0F;
    IO_setup();

    outln("Golfball Launching Program");
    outln("Press s to stop the motor\n");

    for(;;){
/*      while((c = input()) == -1){} /* wait for a keypress
        if(c == '=' || c == '+'){

```

```

        array_position = 0;                // Initialize the array position variable so
                                           // interrupts start taking data
        count += 1;                        // Set PWM value
        desired_speed = count;            // Store desired pwm value
        outint(count); outln(" = is the count.");
    } else if(c == '-' || c == '_'){
        array_position = 0;                // Initialize the array position variable so
                                           // interrupts start taking data
        count -= 1;                        // Set PWM value
        desired_speed = count;            // Store desired pwm value
        outint(count); outln(" = is the count.");
    } else if(c == 's' || c == 'S'){
        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 == 'p' || c == 'P'){
        outint(Cf); outln(" = Barrel Position");
    } else if(c == '1'){
        P1 = count;
        outln("Target 1 Position is Locked");
    } else if(c == 'r' || c == 'R'){
        reload();
        outln("reloaded");
    } else if(c == 'f' || c == 'F'){
        fire();
        outln("fired");
    } else if(c == '2'){
        P2 = count;
        outln("Target 2 Position is Locked");
    } else if(c == '3'){
        P3 = count;
        outln("Target 3 Position is Locked");
    } else if(c == '4'){
        P4 = count;
        outln("Target 4 Position is Locked");
    } else if(c == 'c' || c == 'C'){
        outln("***COMPETITION MODE ACTIVATED***\n");
        while((c != 'q') && (c != 'Q'))
        {
*/
        while(((PINB & 0x10) == 0) && // PB0 = P1 (Target 1)
              ((PINB & 0x20) == 0) && // PB1 = P2 (Target 2)
              ((PINB & 0x40) == 0) && // PB2 = P3 (Target 3)
              ((PINB & 0x80) == 0) && // PB3 = P4 (Target 4)
              (c = input()) == -1) { // Key Press
            if(((PINB & 0x10))) // Target 1 activated
            {
                array_position = 0;
                count = P1;
                desired_speed = count;
                while((Cf < (P1 - 0x02)) || (Cf > (P1 + 0x02))){outint(Cf);}
                reload();
                fire();
            }
        }
        else if(((PINB & 0x20))) // Target 2 activated

```

```

    {
        array_position = 0;
        count = P2;
        desired_speed = count;
        while((Cf < (P2 - 0x02)) || (Cf > (P2 + 0x02))){outint(Cf);}
        reload();
        fire();
    }
else if((PINB & 0x40)) // Target 3 activated
{
    array_position = 0;
    count = P3;
    desired_speed = count;
    while((Cf < (P3 - 0x02)) || (Cf > (P3 + 0x02))){outint(Cf);}
    reload();
    fire();
}
else if((PINB & 0x80)) // Target 4 activated
{
    array_position = 0;
    count = P4;
    desired_speed = count;
    while((Cf < (P4 - 0x02)) || (Cf > (P4 + 0x02))){outint(Cf);}
    reload();
    fire();
}
else if(c == 'q' || c == 'Q')
{
    break;
    outln("quit");
}
}
return 1;
}

```

Appendix C – Pictures of Actual Robot

