Instructors: Dr. A. Antonio Arroyo
Dr. Eric M. Schwartz

TAs: Ryan Stevens
Josh Weaver
Tim Martin

University of Florida
Department of Electrical and Computer Engineering
EEL 4665/5666
Intelligent Machines Design Laboratory

Formal Report
Batmobot

By: Anthony Incardona

Table of Contents

1. *Abstract*

The robot I designed in the Spring of 2012 is the Batmobot. Batmobot will have an added dimension of travel under its utility belt; a grappling hook will be used aid Batmobot travel vertically. The way it will accomplish this is by shooting a rope over a high up bar and then finding the rope once it's looped around the bar and latching onto it. This will create a pulley for the robot to be able to elevate itself off the ground. This action is very much like how Batman uses his grappling hook to maneuver hence the name Batmobot. It will be able to avoid obstacles using IR proximity sensors, find the bar using a beam break sensor, and be able to know when the metal bar is in front of it by using a sonar range finder as well as bump sensors.

2. *Executive Summary*

Batmobot is a robot that will mimic Batman's ability to utilize a grappling hook when trying to climb vertical obstacles. Batmobot can also be used as a surveillance camera if one was attached to the back of its body. This would enable the user to gain another dimension of viewing to be able to get an idea of the particular situation. To begin the situation would be that Batman needs a spot where he needs a bird's eye view. He then marks the proper area for Batmobot to climb and sends out Batmobot. Batmobot then searches the area for the beacon that was sent out by Batman.

Once the beacon is found by Batmobot. It raises the crossbow and launches the steel bullet up and over the pole that was previously marked by Batman. Then Batmobot searches the area for the bullet. This is achieved by driving up close to where it should be then panning the area for something that will trip the sonar range finder that will tell Batmobot that the bar is straight ahead. Batmobot then turns on his electromagnet and charges the steel brick. By using bump sensors Batmobot will know if the bar has been properly attached to the magnet. Once the bump sensors are tripped it will begin to real in the rope which will allow Batmobot to climb up the vertically so that he it can gain another vantage point for Batman. Once it senses the top it will stop and just hover to allowing for the situation to be assessed.

This robot could also have various other capabilities added to it. The situation could be changed that would allow Batmobot to use the electromagnet to latch onto various metal surfaces or he could use the same pulley system to pulley down a lever that is to high up for a normal person to reach. Mechanically Batmobot is ready for many tasks all that it needs is the proper use of its components and he is ready to begin exceeding expectations

and achieving the various tasks that lay ahead of him.

3. Introduction

Move over robin Batman is going to have a new sidekick named Batmobot. By adding a grappling hook feature the robot will be able to imitate its favorite super hero. The purpose of this robot is to be able to achieve added dimension of mobility to the robot so that it can perform a feature at the apex of its climb. The objective of the Batmobot is to find a suitable area that its rope will be able to safely hook on to and shoot the rope over the bar. Once the rope is over the bar Batmobot will find the metal bullet at the end of the rope so that it can grab the hoop. Once this is completed Batmobot will be able to pull itself up to the top. Using proximity sensors it will be able to stop close to the top safely.

4. Integrated System

The Batmobot will be using the Epiphany DIY board created by Tim Martin. The Epiphany DIY utilizes a ATdmega64A1 for its processing. This chip has all the peripherals that are needed to make Batmobot a functional robot such as two eight-channel 12 bit ADC, eight USARTs, UARTs, eight 16 bit timers, and PWMs. The Epiphany DIY board also supports 4 individual motors that will aid in Batmobot being able to pull itself up the rope as well as guide itself using wheels. It will also support up to 24 different servos which will aid in aiming the hoop and retrieving the hoop. The Epiphany DIY board was chosen for many reasons. The designer of the board is one of the TA's therefore assistance with any broken parts or difficulties will be handled with ease. Also this board has everything that will be needed from a robot built into it from the on board regulators to the servo controller.

5. Mobile Platform

The Batmobot will need to a stylish chassis that will be able to house all the servos, motors, and Epiphany DIY board. A crossbow will be mounted on the top to shoot the magnetic bullet. A magnet will be mounted on the front so that Batmobot can latch on to the strong magnetic bullet and complete the loop. The mobile platform must be strong but also lightweight. This will put less strain on the spool that will be pulling Batmobot upwards. Also Batmobot will need to be able to handle vertical travel. The platform the Batmobot will have to be balanced in such a way that when it starts to pull itself up the

rope that it won't lean any way to distort the vertical movement.

They platform consists of two layers. The top holding the electromagnet and crossbow while the bottom of layer houses the motors and electronics as well. This multi-layered platform allows not only protection for the robot but also will allow for more surface area for all the electronics and actuators to be mounted.

Actuation

The actuation for Batmobot will consist of motors and servos that will be used in order to move the robot and allow for interaction with the environment. Two individual motors for wheels so that it will be able to turn 360 degrees in place with a third rolling wheel. A motor will be needed to supply enough torque to pull wind the spool so it can start to levitate. Servos will be used to aim the crossbow at the correct angle as well as fire the crossbow.

Drive Motors

There will be two motors mounted to the rear of Batmobot that will be the driving motors. A caster wheel is attached on the front of Batmobot so that it will be able to turn in place. This will be taken into account when it is panning around to try to find the bar to attach to it. The motors that were chosen to drive Batmobot where two 172:1 metal gear motors from Pololu.com with 70 cm diameter wheels. This allows for Batmobot to travel at the precise speed that it needs to in order as well as be small enough to not affect the overall weight of Batmobot.

Pulley Motor

In order to pull Batmobot up the rope a really strong motor needed to be chosen. The motor that was chosen in order to perform this task is one that supplies 250 kg/cm of torque. This is more than enough to be able to pull up the weight of the rocket. The only drawback of having so much torque is that it only revolves at about 4rpm. This means that the shaft for the pulley needs to be increased in size that that the most of every spin can be utilized. The robot may not climb up the pole as quick as its counterpart

but it will get there reliably. You know what they always say “The tortoise beat the hare”.

Crossbow Servos

The servos that were chosen in order to control the crossbow aiming system and firing of the crossbow are Hitec HS-645MG. The reason that these servos where chosen where because they could supply 133 oz/in of torque that was required in order to aim the crossbow and well as release the rope. The crossbow is heavy and requires a good bit of torque in order to have proper control over it. Releasing the crossbow string also requires a lot of torque because it the string presses the servo horn against the side wall of the crossbow itself. In order to counter this force the servo needs a lot of force to move it from loaded to fired position.

String Guide Servo

Another servo from HiTec called the HS-485 was chosen to guide the rope once it is on the pulley arm. This is so that the string when it is reeled up the by the shaft is evenly distributed on the entire shaft so that the moment arm of the shaft doesn’t become so big that it increases the torque required by the motor and it fails.

6. Sensors

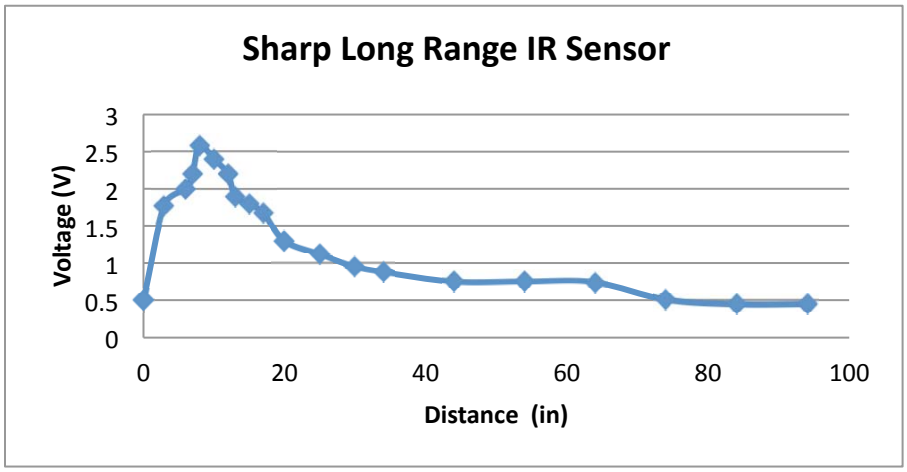
Sensors that Batmobot will be using are infrared sensors for proximity sensing and obstacle avoidance. Bump sensors that will help back up the infrared sensors in obstacle avoidance as a failsafe in case the robot actually runs in to a wall. Also bump sensors will tell Batmobot when the magnet has connected with the steel bullet.

Sonar Range Finder

The sonar range finder that I used was the Maxbotix LV-EZ4. The reason that I chose this sonar range finder was because it has a long narrow beam angle. This allows for precise detection of the steel bullet so that it will always find it and not some other object that is close to it. Having a small beam angle makes it possible to mount it close to the electromagnet so that it can properly match up the magnet with the steel bar.

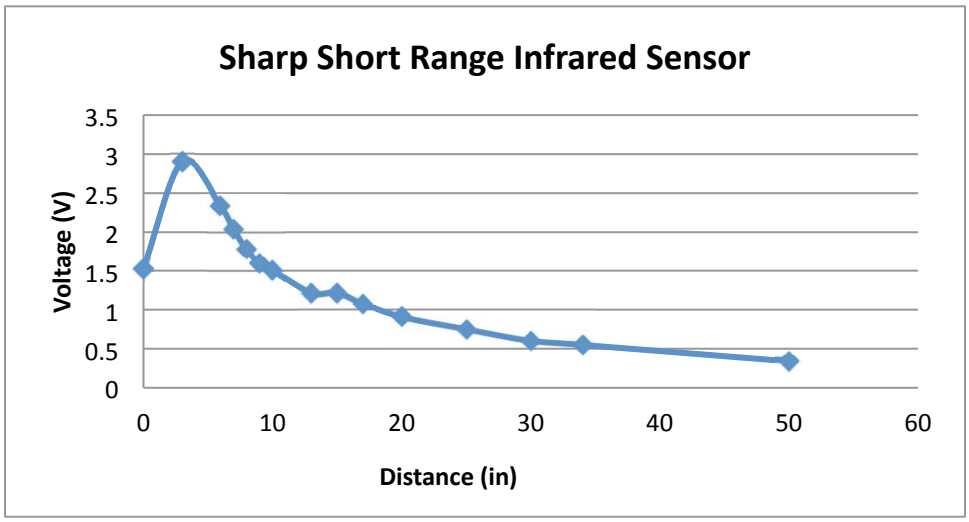
Long Range Infrared Sensor

A Sharp GP2Y0A02YK0F Long range infrared sensor is used as a proximity sensor to warn Batmobot when it gets close to a wall or object right in front of it. A long range sensor was chosen so that it wouldn't get close to the wall and would be able to detect the object early before it was a problem. The long range infrared sensor will be mounted in the center on the front of Batmobot so that centered objects will be seen early and avoided. The only problem that this sensor encounters is that it doesn't sense things very well when they are within 20 cm. Therefore to counter this short range IR Sensors are also used. The graph below in Fig. 3 shows how the voltage changes as the distance of the object changes. This was done by using a large book as the object it was detecting and moving it to certain intervals on a measuring tape. The LCD showed the voltage that was outputted via the Sharp Long Range sensor.



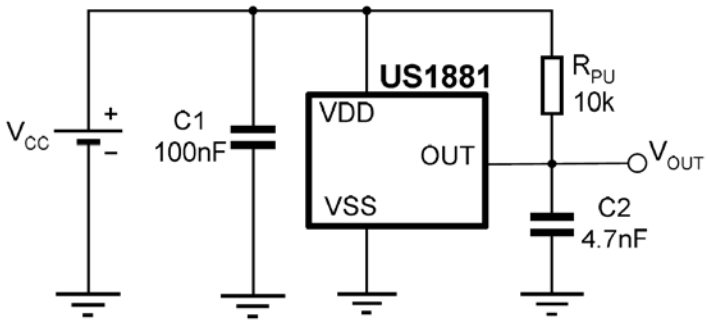
Short Range Infrared Sensor

Similar to the long range IR sensor a short range IR sensor will also be added to Batmobot. Two Sharp GP2D120XJ00F Short Range Proximity Sensors were chosen for this application. They will be angled outward so that they will also be able to detect corners and objects to the side of Batmobot. These are almost identical to the long range sensors only they are applied to smaller measurements. Therefore refer to the long range IR sensor above for more details. The same measurement technique was used for these sensors as the long range sensors which produced the graph shown in Fig. 4 below.



Hall Effect sensor

The hall effect sensor will be used to sense when the magnet bullet is close to Batmobot. This will help Batmobot find the bullet after it has been launched over the pole. The Hall Effect sensor acts as standard logic gate that sends an output true signal when it is in the presence of a magnet. This allows for simple GPIO to be used as the input for this sensor. The sensor that will be used is the US1881 from . Shown below in Fig. 5 is the schematic of how it will be implemented as a circuit.



Special Sensor: Beam Break Sensor

The beam break sensor will be mounted on top of Batmobot and will be facing upwards so that it is level with a surface above it. This sensor will enable Batmobot to search for a suitable launch site so that it can shoot its magnetic bullet over the pole. The pole will already be marked with an IR led to set the beam so that the sensor will be able to detect the pole better. Stable detection is made possible with this sensor due to the wide beam

area and even moving objects can be detected. Batmobot will be able to move around and look for the pole and discover the premarked location. Once this condition is met the robot will be able to launch its bullet at the proper angle to get it over the bar.

7. Behaviors

Search for the pole

When Batmobot is searching for the pole that was marked by Batman it will be running the code for obstacle avoidance so that it doesn't run into and obstacles. It will continually search for the bar until it has finally sensed the IR beacon. Both drive motors will be controlling the robot while all servos will be disabled so that they don't use up any of the battery.

Pole Found

Once the pre-marked pole is discovered Batmobot will stop moving and raise up the crossbow arm to the angle that will shoot it over the bar. Once the conditions are good the crossbow string will be released and the metal bullet will be shot over the pole. After a little bit of time so that the string stops swinging it will drive up a short distance and then pan the robot using the Sonar range finder to find the bar.

Bar Found

Once the bar is found it will drive up to it running full speed into it until the bump sensors are activated telling Batmobot that it has a clan connection with the steel bar. Once it senses a clean connection with the steel bar it will stop and start reeling up the string by using the pulley motor. Once this is completed Batmobot has completed its objective and will now maintain its position to aid Batman.

8. Experimental Layout and Results

The figures shown in the sensor area demonstrate the experiments that where taken in order to verify proper functionality of the sensors. The gyroscope and accelerometer measurements were taken from a rocket launch for the UF rocket team. They demonstrated actual results that were given from the sensor as the rocket moved through space. The various spikes in the acceleration data is explained by the launch, deployment at apogee, and the landing.

9. Conclusion

This was a challenging yet rewarding robot to build. There will be a need to interface the electronics correctly with the mechanical parts so that Batmobot can complete its objective. The limitations of Batmobot are that it can only travel up a high as its rope and launcher can achieve. This is just the beginning of the capabilities of Batmobot and many additions to it will enable it to become a fully functional partner for Batman. Some of the difficulties that where encountered was being able to find the bar after it was launched over the pole. This was achieved by using a sonar range finder and some guide rails to aid Batmobot in being capable of finding the bar so that it can start to reel up the rope. Overall Batmobot is a robot that will continually grow and have many other renditions to come.

10. Documentation

a) Beam Break Sensor
http://pcbheaven.com/circuitpages/Long_Range_IR_Beam_Break_Detector/

b) ATmega64A1 Documents
http://www.atmel.com/devices/atxmega64a1.aspx

11. Appendices

Appendix A: *Code used to test circuits and display results on the LCD panel*

```
#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ATtinyServo.h"
#include "ADC.h"
#include "sonar.h"
```

```
#define DbLedOn()           (PORTR.OUTCLR = 0x02)             //Turns the debug led on.
The led is connected with inverted logic
#define DbLedOff()          (PORTR.OUTSET = 0x02)             //Turns the debug led off.
The led is connected with inverted logic
#define DbLedToggle()       (PORTR.OUTTGL = 0x02)             //Toggles the debug led
off.  The led is connected with inverted logic

int main (void)
{
       board_init(); /*This function originates in the file init.c, and is used to
initialize the Epiphany DIY
                                    motorInit() is declared within because by default you
the user should define what your
                                    motor setup is to prevent hurting the Epiphany.  You
can do this by
                                  */
       RTC_DelayInit();//initializes the Real time clock this seems to actually take an
appreciable amount of time
       DbLedOn();    //I like to do this by default to show the board is no longer
suspended in the bootloader.
       ATtinyServoInit();//this function initializes the servo controller on the Attiny
       uartInit(&USARTC0,115200);//as can be seen in the schematic.  This uart is
connected to the USB port.  This function initializes this uart
       uartInit(&USARTE1,9600);//as can be seen in the schematic.  This uart is connected
to the Xbee port.  This function initializes this uart
       ADCsInits();//this function initializes the ADCs inside the Xmega
       sei();
//     LCDInit();
       uint16_t k;
       int i, j;
       int  close, far, FIRST_F, SECOND_F, THIRD_F, FIRST_C, SECOND_C, THIRD_C;
       float  far_out,close_out, HEXV_F, Digvol_F, HEXV_C, Digvol_C;
       stdout = &lcd_str;

              LCDInit();
              printf("%s", "Far:       V");
              _delay_ms(10);
              LCDCommand(0xC0);
              printf("%s", "Close:     V");
              k=0;

       while(1)
       {

              far_out = analogRead_ADCA(0x0);
              close_out = analogRead_ADCA(0x01);

              LCDCommand(0x87);
              Digvol_F = far_out;
              HEXV_F = ((Digvol_F*5)/4095)-0.1;

              FIRST_F=HEXV_F;
              SECOND_F = (HEXV_F-FIRST_F)*10;
              THIRD_F = (((HEXV_F-FIRST_F)*10)-SECOND_F)*10;
              FIRST_F+=0x30;
              SECOND_F+=0x30;
              THIRD_F+=0x30;
```

```
            printf("%c%c%c%c", FIRST_F,'.',SECOND_F,THIRD_F);

            _delay_ms(10);
            LCDCommand(0xC7);

            Digvol_C = close_out;
            HEXV_C = ((Digvol_C*5)/4095)-0.1;

            FIRST_C=HEXV_C;
            SECOND_C = (HEXV_C-FIRST_C)*10;
            THIRD_C = (((HEXV_C-FIRST_C)*10)-SECOND_C)*10;
            FIRST_C+=0x30;
            SECOND_C+=0x30;
            THIRD_C+=0x30;

            printf("%c%c%c%c", FIRST_C,'.',SECOND_C,THIRD_C);

            _delay_ms(500);

      }

}
```

Appendix B: *Code used to control Batmobot*

```
#include <asf.h>
#include <avr/io.h>
#include <ctype.h>
#include <stdint.h>
#include <stdio.h>
#include <util/delay.h>
#include "motor.h"
#include "lcd.h"
#include "uart.h"
#include "RTC.h"
#include "ATtinyServo.h"
#include "ADC.h"
#include "sonar.h"

#define DbLedOn()          (PORTR.OUTCLR = 0x02)            //Turns the debug led on.
The led is connected with inverted logic
#define DbLedOff()         (PORTR.OUTSET = 0x02)            //Turns the debug led off.
The led is connected with inverted logic
#define DbLedToggle()      (PORTR.OUTTGL = 0x02)            //Toggles the debug led
off.  The led is connected with inverted logic

int main (void)
{
      board_init(); /*This function originates in the file init.c, and is used to
initialize the Epiphany DIY
                        motorInit() is declared within because by default you
the user should define what your
                        motor setup is to prevent hurting the Epiphany.  You
can do this by
                     */
```

```
	RTC_DelayInit();//initializes the Real time clock this seems to actually take an
appreciable amount of time
	DbLedOn();    //I like to do this by default to show the board is no longer
suspended in the bootloader.
	ATtinyServoInit();//this function initializes the servo controller on the Attiny
	uartInit(&USARTC0,115200);//as can be seen in the schematic.  This uart is
connected to the USB port.  This function initializes this uart
	ADCsInits();//this function initializes the ADCs inside the Xmega
	sei();
	LCDInit();
	uint16_t k;
	uint8_t r, l, b, j;
	int  close, far, FIRST_F, SECOND_F, THIRD_F, FIRST_C, SECOND_C, THIRD_C;
	float  far_out,close_out, HEXV_F, Digvol_F, HEXV_C, Digvol_C, i, i_hex, i_dig;
	stdout = &lcd_str;

		int u;


		u = PORTE_DIR;
		u = u | 0x01;
		PORTE_DIR = u;
		PORTE_OUT = 0X00;
		setServoAngle(2,180);

		u = PORTF_DIR;
		u = u & 0x3F;
		u = u | 0x30;
		PORTF_DIR = u;
		PORTF_OUT = 0X00;

		PORTF.PIN7CTRL = PORT_OPC_PULLUP_gc;
		PORTF.PIN6CTRL = PORT_OPC_PULLUP_gc;



		LCDInit();
		printf("%s", "Searching?");
//		printf("%s", "Far:       V");
		_delay_ms(10);
//		LCDCommand(0xC0);
//		printf("%s", "Close:     V");

	while(1)
	{
/*	while(1){
		i = analogRead_ADCA(0x02);                              // use to test range
of IR Beacon in different enviroments
		i_hex = ((i*5)/4095)-0.1;
		if(i_hex<=1.0){PORTF_OUT = 0x10;
		i = analogRead_ADCA(0x02);
		i_hex = ((i*5)/4095)-0.1;}
		else{PORTF_OUT = 0x00;
		i = analogRead_ADCA(0x02);
		i_hex = ((i*5)/4095)-0.1;}
		}
*/
		far_out = analogRead_ADCA(0x00);
```

```
                close_out = analogRead_ADCA(0x01);
                i = analogRead_ADCA(0x02);

                i_hex = ((i*5)/4095)-0.1;

//              LCDCommand(0x87);
                Digvol_F = far_out;
                HEXV_F = ((Digvol_F*5)/4095)-0.1;

/*              FIRST_F=HEXV_F;
                SECOND_F = (HEXV_F-FIRST_F)*10;
                THIRD_F = (((HEXV_F-FIRST_F)*10)-SECOND_F)*10;
                FIRST_F+=0x30;
                SECOND_F+=0x30;
                THIRD_F+=0x30;

                printf("%c%c%c%c", FIRST_F,'.',SECOND_F,THIRD_F);

                _delay_ms(10);
                LCDCommand(0xC7);
*/
                Digvol_C = close_out;
                HEXV_C = ((Digvol_C*5)/4095)-0.1;
/*
                FIRST_C=HEXV_C;
                SECOND_C = (HEXV_C-FIRST_C)*10;
                THIRD_C = (((HEXV_C-FIRST_C)*10)-SECOND_C)*10;
                FIRST_C+=0x30;
                SECOND_C+=0x30;
                THIRD_C+=0x30;

                printf("%c%c%c%c", FIRST_C,'.',SECOND_C,THIRD_C);
                _delay_ms(250);
                _delay_ms(250);
*/

        if(HEXV_C >= 1.2 || HEXV_F >= 2.0)
        {
                //setMotorDuty(4,1500,MOTOR_DIR_BACKWARD_gc);                          // Use
to reset the Robot to lower string

                while(1)
                {
                setServoAngle(2,161);
                _delay_ms(250); _delay_ms(250);
                far_out = analogRead_ADCA(0x0);
                close_out = analogRead_ADCA(0x01);
                i = analogRead_ADCA(0x02);

                i_hex = ((i*5)/4095)-0.1;
                Digvol_F = far_out;
                HEXV_F = ((Digvol_F*5)/4095)-0.1;
                Digvol_C = close_out;
                HEXV_C = ((Digvol_C*5)/4095)-0.1;

                if(HEXV_C >= 1.2 || HEXV_F >= 2.0)
                {
```

```
	while(1)
	{
		_delay_ms(250); _delay_ms(250);
		far_out = analogRead_ADCA(0x0);
		close_out = analogRead_ADCA(0x01);
		i = analogRead_ADCA(0x02);

		i_hex = ((i*5)/4095)-0.1;
		Digvol_F = far_out;
		HEXV_F = ((Digvol_F*5)/4095)-0.1;
		Digvol_C = close_out;
		HEXV_C = ((Digvol_C*5)/4095)-0.1;
		PORTF_OUT = 0X10;


		if(HEXV_C >= 1.2 || HEXV_F >= 2.0)
		{
			setMotorDuty(1,625,MOTOR_DIR_FORWARD_gc);
			setMotorDuty(2,625,MOTOR_DIR_FORWARD_gc);
			_delay_ms(250);
			_delay_ms(250);
			setMotorDuty(1,625,MOTOR_DIR_FORWARD_gc);
			setMotorDuty(2,625,MOTOR_DIR_BACKWARD_gc);
			_delay_ms(250);
			_delay_ms(250);

		}
		else if(i_hex <= 1.0)
		{
			while(1)
			{

				setMotorDuty(1,0,MOTOR_DIR_BACKWARD_gc);
				setMotorDuty(2,0,MOTOR_DIR_BACKWARD_gc);
				LCDCommand(0x80);
				PORTF_OUT = 0X20;
				printf("%s", "Ready to Fire!");
				setServoAngle(1, 117);
				_delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
				_delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
				_delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
				setServoAngle(2,180);
				_delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
				_delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
				//setMotorDuty(4,1500,MOTOR_DIR_FORWARD_gc);
				_delay_ms(250);
				_delay_ms(250);
			_delay_ms(250); _delay_ms(250); _delay_ms(250); _delay_ms(250);

				PORTE_OUT = 0x1;
                                        // Turn on Electromagnet

				while(1)
```

```
                        {
                                far_out = analogRead_ADCA(0x03);

                                Digvol_F = far_out;
                                HEXV_F = ((Digvol_F*5)/4095)-0.1;

                                setMotorDuty(1,600,MOTOR_DIR_BACKWARD_gc);
                                setMotorDuty(2,575,MOTOR_DIR_BACKWARD_gc);

                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
                                _delay_ms(250); _delay_ms(250); _delay_ms(250);
_delay_ms(250);
                                _delay_ms(250); _delay_ms(250);
_delay_ms(250);_delay_ms(250);
                                _delay_ms(250);
_delay_ms(250);_delay_ms(250);_delay_ms(250);
                                _delay_ms(250);_delay_ms(250);
                                setMotorDuty(4,0,MOTOR_DIR_FORWARD_gc);

                                while(1)
                                {setMotorDuty(1,550,MOTOR_DIR_BACKWARD_gc);
                                setMotorDuty(2,550,MOTOR_DIR_FORWARD_gc);

                                far_out = analogRead_ADCA(0x03);
                                Digvol_F = far_out;
                                HEXV_F = ((Digvol_F*5)/4095)-0.1;


                                if(HEXV_F < 0.30)
                                {
                                        while(1)
                                        {
                                                _delay_ms(250); _delay_ms(250);
_delay_ms(250); _delay_ms(200);
                                                 _delay_ms(250);
                                                PORTF_OUT = 0x30;

        setMotorDuty(1,600,MOTOR_DIR_BACKWARD_gc);

        setMotorDuty(2,590,MOTOR_DIR_BACKWARD_gc);
                                                //setMotorDuty(4,700,
MOTOR_DIR_FORWARD_gc);

                                                j = PORTF_IN;
                                                r = j & 0x80;
                                                b = j & 0xC0;
                                                l = j & 0x40;
```

```
                                                if((r == 0x00) || (b == 0X00) || (l ==
0x00))
                                                        {

        setMotorDuty(1,0,MOTOR_DIR_BACKWARD_gc);

        setMotorDuty(2,0,MOTOR_DIR_BACKWARD_gc);
                                                                        while(1)
                                                                        {


        setServoAngle(1,178);
                                                                                disableServo(2);


        setMotorDuty(4,1500,MOTOR_DIR_FORWARD_gc);


                                                                                _delay_ms(250);
_delay_ms(250);
                                                                                far_out =
analogRead_ADCA(0x0);
                                                                                close_out =
analogRead_ADCA(0x01);
                                                                                i =
analogRead_ADCA(0x02);


                                                                                i_hex =
((i*5)/4095)-0.1;
                                                                                Digvol_F = far_out;
                                                                                HEXV_F =
((Digvol_F*5)/4095)-0.1;
                                                                                Digvol_C =
close_out;
                                                                                HEXV_C =
((Digvol_C*5)/4095)-0.1;
                                                                                PORTF_OUT = 0X10;


                                                                                if(HEXV_C >= 1.2 ||
HEXV_F >= 2.0)
                                                                                {
                                                                                        while(1)
                                                                                        {


        setMotorDuty(4,1500,MOTOR_DIR_BACKWARD_gc);


                                                                                        }
                                                                                }


                                                                        }
                                                        }
                                                }

                                        }
                                }
                        }
                }
                }
                else
```

```
                        {
                                setMotorDuty(1,620,MOTOR_DIR_BACKWARD_gc);
                                setMotorDuty(2,620,MOTOR_DIR_BACKWARD_gc);
                        }

                }
                }
        }
}

}
}
```