FIREBUG

Final Report

Marc Milks
University of Florida
EEL5666
Intelligent Machine Design Lab

Instructors: Dr. Arroyo
Dr. Schwartz

TA's: Mike Pridgen
Ryan Stevens
Thomas Vermeer
Tim Martin
Devin Hughes

Table of Contents

Abstract

FIREBUG is an autonomous liquid dispersal agent. It is capable of traversing a room and avoiding obstacles in search of a flame source. Once the flame source is found, FIREBUG will disperse the contents of its reservoir to perform the chosen task, either extinguishing, enflaming, or both.

Executive Summary

FIREBUG began in thought as a firefighting robot. It was initially envisioned as a flame detecting robot that could put out a flame. After a rousing speech by Dr. Arroyo focusing on challenging ourselves to go beyond the "same old robot", I decided to move past the traditional models and implement firefighting with a twist.

Most of the first month was spent looking at robotics supply websites and picking up basic skills. I learned to solder properly and also to make connector wire. On receiving my PVR board I began learning machine interface programming and started thinking about the manner in which I would program my final robot.

As my sensors began arriving, I interfaced my sonar sensor with my board along with my LCD. This was the first sensor that I was able to draw raw data from. I determined distances based on the resultant numerical data and incorporated that in my later code. I also manufactured bump sensors and demonstrated them to my TAs.

My platform was originally going to be manufactured using the available wood in lab. I decided that the dimensions and design of my robot would be better served by a thicker board and so moved on to compressed board due to affordability. The two levels are joined by six 1 x ½ inch spacers. My motive power is proved by two motors which are mounted beneath the bottom level and powered by a 12V source separate from my PVR board. These along with a 360 degree caster wheel provide rotation and stability for the rest of the platform.

The motor system involved learning how to interface the PVR with my serial controller motor controller. This proved a challenging task as I'd had no history with Atmel documentation or interfacing electronics boards. I created a set of software functions which I shared with fellow IMDLers to control the motor functions using simple calls. This allowed me to move on to obstacle avoidance behavior.

I placed the sonar sensor on the underside of the lower level to give ground coverage and, using the data from my initial tests, set limits on the distance a wall could be avoided at. I then mounted bump sensors on either side of the front two spacers with the intent of later putting on extensions to assure clearance for the robot as it moved forward. I then placed the Hamamatsu sensor on the board. I had to modify the stock implementation by removing the 5V regulator n its driving circuit and directly attaching the power to the PVR's 5V bus. This turned out to be a relatively painless operation and worked first time.

This point had me putting on Aluminum in preparation for the pump assembly. I initially put unfinished aluminum on the robot and later painted it with red enamel. The pump assembly ended up taking me a little under a month. I went through an initial attempt with a 4V pump but later had to change to a 12V pump with a pressure regulator controlled by a relay set off by a mosfet in a convoluted, though necessary, setup.

Testing occurred and final software programming occurred over the course of two weeks and resulted in integrated behavior that allowed me to meet all my goals for this project. It was extremely rewarding learning all the bits and pieces and turning them in to a finished product.

Introduction

While this design originated in my thoughts, after some brief research I found that it is a well worn project. My original idea was to extinguish flames, but seeing how many robots had pursued that same end I decided to broaden the scope of my project and allow delivery of various liquids. There are many tasks that can be performed by a robot with a liquid reservoir and some guidance principle: Flame extinguishing to fight small fires, pyrotechnic exhibitions for entertainment, or even ink marking of some structural flaw or substance not visible to unaided human perception. FIREBUG will focus on the first two of these possibilities for the planned demo. In this paper I will detail the construction and materials that will go into constructing FIREBUG and make projections about its later function and purpose in so far as I am able.

Integrated System

FIREBUG is built around a Pridgen Vermeer Robotics Xmega128 board powered by a 6 Eneloop NiMH AA battery pack. Attached to that board will be a variety of sensors and actuators. For collision detection and avoidance a series of bumpers and a LV-MaxSonar®-EZ1 sonar sensor is used. For Actuation, two Easy Roller robot wheels and are used. These motors will be controlled with a qik 2s9v1 dual motor controller from Pololu which will have its own 12V power supply from a tenergy 2000 mAh battery pack from allbattery.com. For flame detection, a Hamamatsu IR R2868 flame sensor and C3704 Drive Circuit will be used to detect the presence of signature radiation from the flame. Lastly the generic 12V RC fuel pump was controlled by a power relay attached to the Xmega and dispersed liquids from a model airplane fuel tank through a small atomizing nozzle from Home Depot. The pump is powered through the motor power cell.

Firebug will begin by doing an angular scan using its wheel rotation to point the flame detector and will receive a signal from the A/D converter. If the flame is not detected FIREBUG will perform some movement based search criteria and scan again. If the flame is detected, however, the signal strength and obstacle sensor will determine whether it is close enough for liquid dispersal. If it is not close enough, Firebug will move toward the flame and reassess. When it determines it is in range of the flame, FIREBUG will vent a controlled burst of its liquid reservoir. While all this is occurring, the collision avoidance sensors will be monitoring to prevent damage to the robot.

Fig. 1: Design of FIREBUG

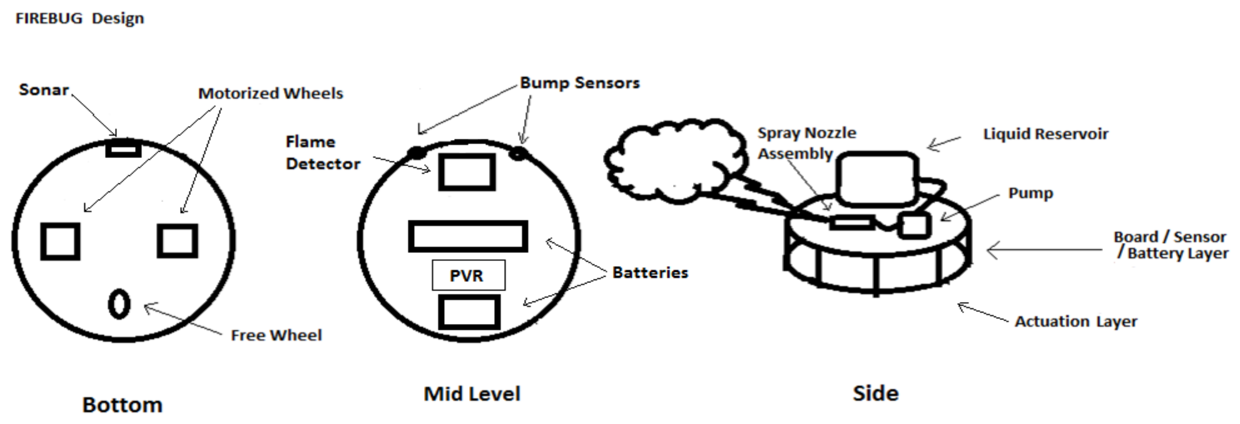


Fig. 2: Block Diagram of FIREBUG electronic components

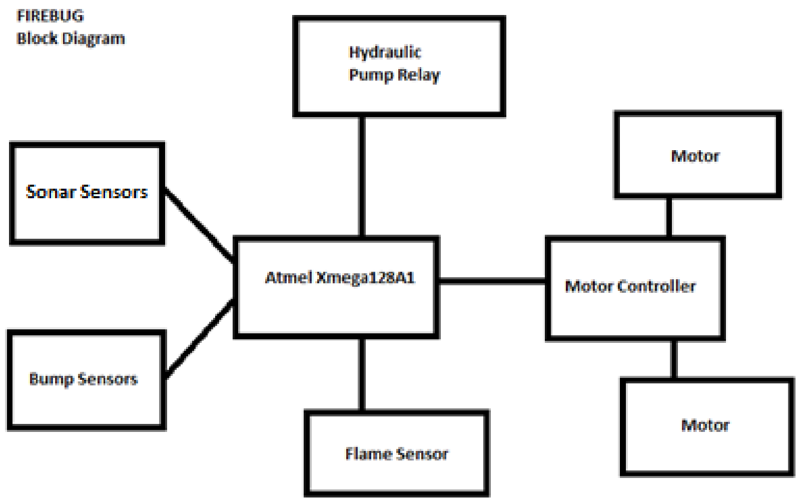


Fig. 3: Pump Diagram

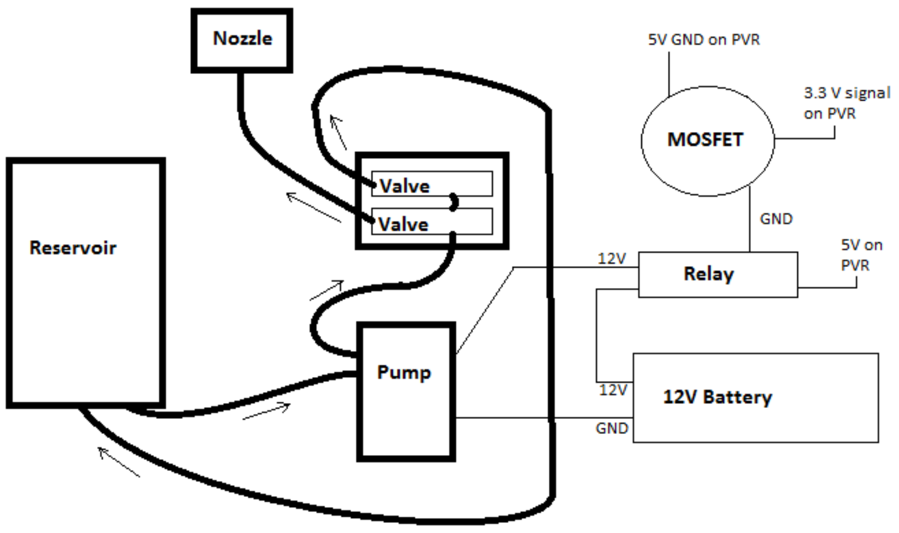
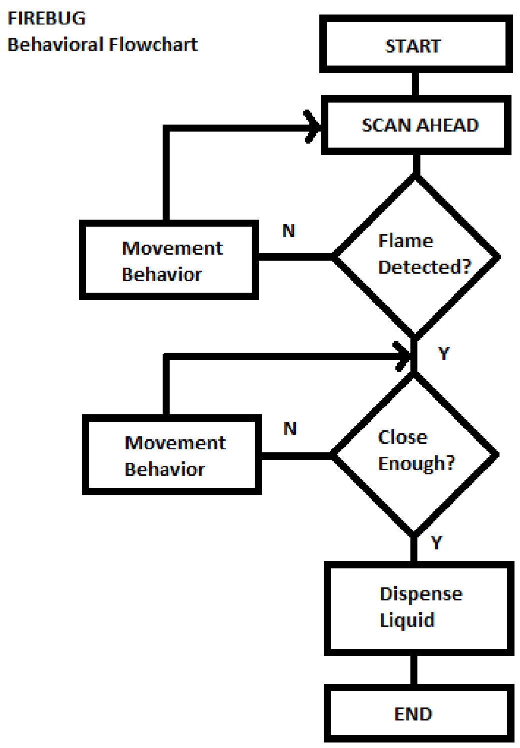


Fig.4: Flowchart of FIREBUG Behavior



Mobile Platform (Fig 1 above)

FIREBUG's Mobile platform will consists of 2 identically semi-round cuts of wood with 6 wood spacers between levels. The lower level accommodates the motors and wheels (mounted beneath) as well as the Xmega board and also the sensor arrays for collision avoidance and flame detection and serves as a mount point for the bumper switches. The lower layer is also where both battery packs are mounted. The top layer contains the liquid apparatus including the pump, reservoir, and spray nozzle assembly. This is coated in aluminum painted with enaml for water resistance to prevent accidental water from travelling between the levels and thereby damaging the electronics. The platform was the easiest portion of this project. It involved using the band saw and drill press in lab and had no troublesome points

Actuation

FIREBUG is using two Easy Roller robot motors with attached wheels for actuation. Along with a 3rd free wheel this allows the circular robot the ability to rotate in one spot and also to adjust its movement easily to stay on course when approaching the flame. The torque from the data sheet lists there motors as having 3.6 KG CM torque and 200 RPM max. These motors will be controlled with a qik 2s9v1 dual motor controller from Pololu. There were initial issues getting the motor controller to communicate with the PVR board, but in the end it was attached to port F, pins 2&3 and now works completely within expectations. Putting together the serial control together was the hardest part of actuation. I didn't know how to read the Atmel documentation nor did I know what the PVR board used for TTL serial communication.

The actuation algorithm involves searching between periods of obstacle avoidance movement. Once a flame is detected the robot moves toward the flame in small increments correcting direction as it goes. Once the sonar detects that the distance is optimal, there is a side to side movement to accompany the pump initiation. (fig 4)

The Pump actuation is controlled by a relay. This could not be controlled directly and so was connected to a mosfet on its ground line and placed on the 5V power bus. This setup can be seen in fig 3.

Sensors (Fig 2 above)

FIREBUG contains a variety of sensors. The simplest of the sensors are 2 bump sensors. These sensors detect impact through an open/close circuit. Next, Firebug has a LV-MaxSonar®-EZ1 sonar sensor to determine the proximity of obstacles. This sensor has a stated range of approximately 250 inches, though Firebug will not need that level of distance detection. It uses this sonar sensor to detect between 8 to 12 inches of clearance. Lastly it has a Hamamatsu IR R2868 flame sensor and C3704 Drive Circuit sensor to detect the presence of flame directly in front of itself. Interfacing the sensors was relatively painless, the only problem occurred when the LCD was used and interfered with the necessary data sampling from the sensors. A small time delay fixed this problem.

Behaviors (Fig 4 above)

FIREBUG exhibits multiple behaviors. Using its flame sensor in conjunction with a movement/scan rotation as a search, FIREBUG quickly locates a source of flame and performs a predetermined action of liquid dispersal. FIREBUG also displays collision avoidance behavior. Random search behaviors are programmed for operation until the flame source is detected.

Experimental Layout

The experimental layout is simple. Firebug will be placed randomly in a closed area containing one or more open flames and some random obstacles. FIREBUG will find and extinguish all open flames without becoming obstructed using a liquid delivery which enflames to extinguish.

Conclusion

Firebug, as described, is a novel and interesting spin on the firefighting robot. I have accomplished roughly exactly what I sent out to do with Firebug. I had, however, initially expected to use 2 pumps and reservoirs and settled for just 1 due to space and cost. My work was limited by the time constraints of the semester. I had to spend a month working on the pump assembly and had to shorten my software revisions. It was also limited by the availability of components locally. Each time a component had to be sent from overseas a delay occurred in development.

My pump assembly exceeded my expectation. It allowed for much more flexibility that I had originally intended and was the highlight of my demonstrations. It also gave me the opportunity to make a functional electronics assembly with the mosfet and relay. I feel I could spend more time improving my software. I had to do things much faster than I had originally intended and feel the use of occasional interrupts for obstacle avoidance may have benefited the finished robot.

My recommendations to students who follow in IMDL are simple. First, begin on your platform immediately. It is nice to use the precision cut wood in IMDL, but it is quicker to just self assemble without learning the design software. You'll need time later for the more complicated electronics integration and programming. Secondly, I would recommend that if a student doesn't know how to program they learn basic C as soon as possible in the semester. Lastly, I would recommend that all students familiarize themselves with the way their board uses serial ports and other interfaces available and how to program them.

Starting the project over, I would have done almost everything the same. I would, however, have added 4 IR sensors in the periphery of the robot to aid in obstacle detection. This would have allowed for better obstacle avoidance behavior and give FIREBUG options for sensing its environment. I may have also added a second reservoir with an electronically controlled switch so that two separate sources could be utilized to make dual colored flames.

Appendix

- Final Code

```
#include "usart_driver.h"
#include "PVR.h"

#define USART USARTF0

void M0_ForwardSlow(int a);
void M0_ForwardFast(int a);
void M0_ReverseSlow(int a);
void M0_ReverseFast(int a);
void M1_ForwardSlow(int a);
void M1_ForwardFast(int a);
void M1_ReverseSlow(int a);
void M1_ReverseFast(int a);

uint8_t sendData [4];
uint8_t receivedData;

int main(void)
{
		xmegaInit();
		delayInit();
		ADCAInit();
		lcdInit();


//Begin Motor Controller Setup

		PORTF.DIRSET = PIN3_bm;

		PORTF.DIRCLR = PIN2_bm;

		USART_Format_Set(&USART, USART_CHSIZE_8BIT_gc, USART_PMODE_DISABLED_gc, false);

		USART_Baudrate_Set(&USART, 3317,-4);

		USART_Rx_Enable(&USART);
		USART_Tx_Enable(&USART);

//End Motor Controller Setup

		PORTH_DIR = 0b11111110;				//sets up flame detector on port H pin 0
		PORTJ_DIR = 0b11111100;				//sets up bump sensors on Port J pins 0,1
		PORTB_DIR = 0b11111111;				//sets up switch signal for pump mosfet,LEDs (1 green 2 yellow 3 red)
		PORTB_OUT = 0b00001110;				//turn on all but LEDs
		delay_ms(5000);					//delay for optional reprogramming

		bool dontStop = true;				//stop condition for whole behavior
		bool flameNotDetected = true;			//variable to flag flame being found
		bool rotationStop = true;			//stop flag for flame detection movement
		bool collisionStop = true;			//stop flag for obstacle avoidance
		bool beginDemo = true;
		int bump;					//bump sensor variable
		int sonar = 1000;				//sonar sensor variable
		int detector = 0;				//flame detector variable
		int timeout = 0;				//timeout rotation and movement phases
		int rotationConst = 0;				//loops it takes for 1 revolution using delay_ms(5)

		M0_ForwardSlow(30);
		M1_ReverseSlow(30);

		PORTB_OUT = 0b00001000;				//set to 0 to turn off all but Red LED

		while((rotationConst<5000)&&rotationStop){
				delay_ms(7);
				rotationConst++;
				detector = PORTH_IN&0x01;
```

```
        if(rotationConst>500){
                if(detector>0){
                        rotationStop = false;
                        PORTB_OUT = 0b00000010;
                }

        }
}

M0_ForwardSlow(0);
M1_ReverseSlow(0);
delay_ms(2000);

int testcase = 4;
while(beginDemo){
        testcase =  1 + (PORTJ_IN & 0b00000011);
        if(testcase == 2){
                beginDemo = false;
        }
        if(testcase == 3){
                PORTB_OUT = 0b00001111;
                delay_ms(15000);
                PORTB_OUT = 0b00000010;
                testcase = 4;
        }

        delay_ms(200);
}

while(dontStop){

        //BEGIN BEHAVIORS

        do{
                //DETECT FLAME


                //rotate
                M0_ForwardSlow(30);
                M1_ReverseSlow(30);

                //check for flame every 5ms
                rotationStop = true;
                timeout = 0;                            //(3200 for one rotation)

                while(rotationStop&&(timeout < rotationConst)){

                        timeout++;
                        delay_ms(7);

                        detector = PORTH_IN&0x01;

                        if(detector > 0){

                                PORTB_OUT = 0b00001010;

                                M0_ForwardSlow(0);
                                M1_ForwardSlow(0);
                                rotationStop = false;
                                flameNotDetected = false;

                        }

                }

                //MOVEMENT/COLLISION AVOIDANCE
                //add in movement and avoidance behavior here

                if(flameNotDetected){

                        timeout = 0;
                        collisionStop = true;

                        M0_ForwardSlow(0);
                        M1_ForwardSlow(0);
                        delay_ms(2000);
                        M0_ForwardSlow(30);
                        M1_ForwardSlow(29);
                        //****RESET DIRECTION PRIOR TO MOVEMENT???****

                        while(collisionStop&&(timeout < 75)){

                                bump = 1 + (PORTJ_IN & 0b00000011);
                                sonar = ADCA0();

                                if((bump < 4)||(sonar < 350)){

                                //collision avoidance behavior goes here

                                        PORTB_OUT = 0b00000110;

                                        M0_ForwardSlow(0);
                                        M1_ForwardSlow(0);
                                        delay_ms(1000);
                                        M0_ReverseSlow(30);
                                        M1_ReverseSlow(30);
                                        delay_ms(600);
                                        M0_ForwardSlow(0);
                                        M1_ForwardSlow(0);
                                        delay_ms(1000);
                                        M0_ReverseSlow(30);
                                        M1_ForwardSlow(29);
                                        delay_ms(2000);
                                        M0_ForwardSlow(0);
                                        M1_ForwardSlow(0);
                                        delay_ms(2000);

                                        collisionStop = false;

                                }

                                timeout++;
                                delay_ms(200);
                                PORTB_OUT = 0b00000010;

                        }
                }

        }while(flameNotDetected);

        //BEHAVIOR FOR PUMP ACTIVATION
        //-check distance and correct if necessary
        //-turn on pump and perform movement back and forth
        //-check for flame
        //-if out then stop else repeat behavior (with mods?)

                for(int i = 0;i<20;i++){
                                sonar += ADCA0();
                                delay_ms(50);
                        }
                        sonar = sonar/20;

                int direction = 0;

                while(sonar > 350){

                        M0_ForwardSlow(30);
                        M1_ForwardSlow(30);

                        if(sonar>600){
                        delay_ms(2000);
                        }else{
                        delay_ms(1000);
                        }

                        M0_ForwardSlow(0);
                        M1_ForwardSlow(0);
                        delay_ms(1000);

                        if(direction == 0){
                        M0_ForwardSlow(30);
                        M1_ReverseSlow(30);
                        direction = 1;
                        }else{
                        M0_ReverseSlow(30);
                        M1_ForwardSlow(30);
                        direction = 0;
                        }

                        detector = 0;
                        rotationStop = true;


                        while(rotationStop){

                            delay_ms(7);

                            detector = PORTH_IN&0x01;

                            if(detector > 0){

                                M0_ForwardSlow(0);
                                M1_ForwardSlow(0);
                                rotationStop = false;

                            }

                        }

                            sonar = 0;
                            for(int i = 0;i<20;i++){
                                sonar += ADCA0();
                                delay_ms(50);
                            }
                            sonar = sonar/20;
                    }

                    PORTB_OUT = 0b00001011;

                    M0_ForwardSlow(30);
                    M1_ReverseSlow(30);
                    delay_ms(200);
                    M0_ForwardSlow(0);
                    M1_ReverseSlow(0);
                    delay_ms(125);

                    for(int i = 0;i<10;i++){

                        M0_ReverseSlow(30);
                        M1_ForwardSlow(30);
                        delay_ms(400);
                        M0_ForwardSlow(0);
                        M1_ReverseSlow(0);
                        delay_ms(125);
                        M0_ForwardSlow(30);
                        M1_ReverseSlow(30);
                        delay_ms(400);
                        M0_ForwardSlow(0);
                        M1_ReverseSlow(0);
                        delay_ms(125);
                    }


                    M0_ReverseSlow(30);
                    M1_ForwardSlow(30);
                    delay_ms(200);
                    M0_ForwardSlow(0);
                    M1_ReverseSlow(0);


                    PORTB_OUT = 0b00001110;



                dontStop = false;
            }

            //stop robot
            M0_ForwardSlow(0);
            M1_ForwardSlow(0);


        }

        void M0_ReverseSlow(int a){

            sendData[0] = 170;
```

```
		sendData[1] = 9;
		sendData[2] = 8;
		sendData[3] = a;

		for(int i = 0; i<4;i++){

			do{

			}while(!USART_IsTXDataRegisterEmpty(&USART));
			USART_PutChar(&USART, sendData[i]);

		}


}

void M0_ReverseFast(int a){

		sendData[0] = 170;
		sendData[1] = 9;
		sendData[2] = 9;
		sendData[3] = a;

		for(int i = 0; i<4;i++){

			do{

			}while(!USART_IsTXDataRegisterEmpty(&USART));
			USART_PutChar(&USART, sendData[i]);

		}


}

void M0_ForwardSlow(int a){

		sendData[0] = 170;
		sendData[1] = 9;
		sendData[2] = 10;
		sendData[3] = a;

		for(int i = 0; i<4;i++){

			do{

			}while(!USART_IsTXDataRegisterEmpty(&USART));
			USART_PutChar(&USART, sendData[i]);

		}


}

void M0_ForwardFast(int a){

		sendData[0] = 170;
		sendData[1] = 9;
		sendData[2] = 11;
		sendData[3] = a;

		for(int i = 0; i<4;i++){

			do{

			}while(!USART_IsTXDataRegisterEmpty(&USART));
			USART_PutChar(&USART, sendData[i]);

		}


}

void M1_ForwardSlow(int a){

		sendData[0] = 170;
		sendData[1] = 9;
		sendData[2] = 12;
		sendData[3] = a;
```

```
        for(int i = 0; i<4;i++){

                do{

                }while(!USART_IsTXDataRegisterEmpty(&USART));
                USART_PutChar(&USART, sendData[i]);

        }


}

void M1_ForwardFast(int a){

        sendData[0] = 170;
        sendData[1] = 9;
        sendData[2] = 13;
        sendData[3] = a;

        for(int i = 0; i<4;i++){

                do{

                }while(!USART_IsTXDataRegisterEmpty(&USART));
                USART_PutChar(&USART, sendData[i]);

        }


}

void M1_ReverseSlow(int a){

        sendData[0] = 170;
        sendData[1] = 9;
        sendData[2] = 14;
        sendData[3] = a;

        for(int i = 0; i<4;i++){

                do{

                }while(!USART_IsTXDataRegisterEmpty(&USART));
                USART_PutChar(&USART, sendData[i]);

        }


}

void M1_ReverseFast(int a){

        sendData[0] = 170;
        sendData[1] = 9;
        sendData[2] = 15;
        sendData[3] = a;

        for(int i = 0; i<4;i++){

                do{

                }while(!USART_IsTXDataRegisterEmpty(&USART));
                USART_PutChar(&USART, sendData[i]);

        }


}
```