Autonomous Tank Robot with Light and Metal Detectors

Sami Suteria

IEEE #

Texas Tech University

December 2013

Abstract

This paper describes a system which autonomously tracks and moves toward a red 1 kHz LED beacon while avoiding/neutralizing steel washers on the ground. The overall system is a combination four separate sub systems. The first is a motor system, which is controlled by an L298, a dual-full bridge driver. The second system is the metal detection system, which is a combination of metal proximity sensors and an electromagnet attached to a servo. The third system is the LED detection system, which uses phototransistors and an op-amp based filter to isolate the 1 kHz signal. These three systems are controlled and coordinated by an FPGA, the Xilinx Spartan 3E, which was on the Diligent Basys2 development board.

Table of Contents

List of Figures

List of Tables

1. Introduction

The system was designed was to mimic the following situation: “A tank must traverse through a mine field to reach a pre-set destination. The tank must actively detect the mines then either avoid or neutralize said mines.” A hobby kit called the Rover 5 Robot Platform as seen in Figure 1.1 represents the tank. Steel washers scattered along the floor represent the mines. The destination is a stationary LED that is blinking at 1 kHz. The system we designed has four main components – the motor control system, the mine detection system, the LED detection system, and the control system.

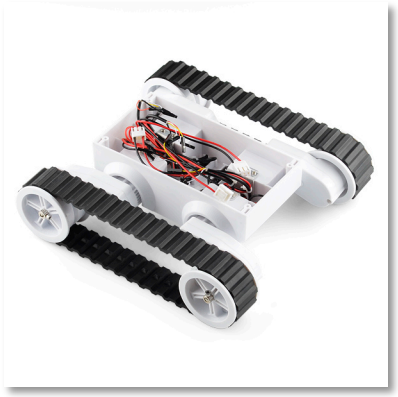

Figure 1.1 – Rover 5 Robot Platform

2. Motor Control System

The motor control system is designed around the L298 Dual Full-Bridge Driver. The datasheet can be found in Appendix A-1. The circuit used to connect the L298 to the motors and power supply can be seen in Figure 2.1. The circuit used in Figure 2.1 was used twice – once for the left motor and once for the right motor. The circuit only uses half of the L298 so two motors can be driven with one chip. The L298 has three control inputs and one output for each motor. The three inputs are: Input A, Input B, and Enable.

The values of Input A and Input B control the direction of the motor and the Enable input controls if the motor is spinning or not. The Enable input had a PWM signal applied to it and the duty cycle affected the speed of the motors. Table 2.1 shows the relationship between the possible values of inputs to motor function.

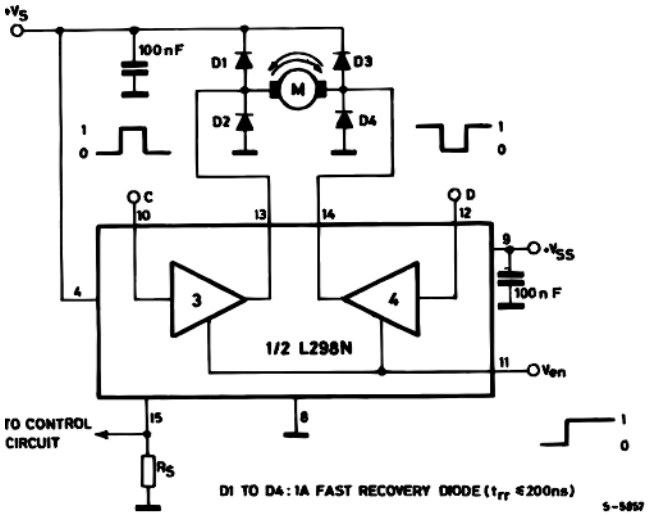

Figure 2.1 – Circuit for driving one motor (L298 Datasheet)

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low H = High X = Don't care

Table 2.1 – Input to Motor function relationship

A PWM signal with a frequency of 100 Hz was applied to the Enable input, which helped in noise reduction in the power supply but caused the motors to move with a jitter. The frequency of the PWM should have been turned up 1kHz or higher once a regulated power supply was created.

As seen in Figure 2.1, there are diodes attached to the input and out of each motor. In order to switch the spinning direction of the motors quickly, Schottky diodes

were used [A-6]. Also seen in Figure 2.1, there is a 100nF capacitor attached near V_s. This is attached in order to reduce noise from the power supply reaching the L298 [A-5]. A similar capacitor is used near the Vss source also. The L298 needs 2 separate power supplies, V_s, which is 7.2V for the motor power supply, and V_{ss}, which is 5V for the internal logic inside the L298. Not labeled on Figure 2.1 is the value of R_s, which is a resistor that was attached to the current/stall sensing output of the L298. R_s was a 1ohm 5W resistor (which was much larger than a typical resistor found in the lab).

3. Mine Detection System

The mine detection system is a combination of a metal detector and an electromagnet. The metal detectors were NPN Style Proximity sensors that were supplied from the lab. As seen in Figure 3.1, the metal detector was hooked with a simple circuit. The output signal was the same as V_{cc} and would produce a high when metal was under the sensor and a low of 0V when no metal was detected.

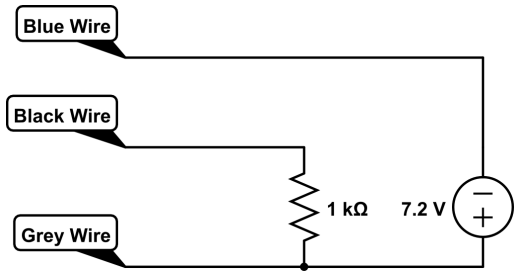

Figure 3.1 – Metal Detector Sensor Wiring Diagram

Four metal detectors were placed evenly spaced on a shelf in front of the tank. In order to compensate for the space in between the metal detectors, weather stripping for doors was used to funnel the mines to one of the four detectors. Weather stripping was also used at the edges of the tank to push any mines close the edge away from the tank.

In addition to the metal detectors, an electromagnet, which was mounted on a servo with a wooden arm, was used to pick up and move the mines from in front of the tank to the side. The state diagram in the control section will explain the logic for how the arm worked when a mine was detected.

The electromagnet was a current limited electromagnet, so it did not try to use as much current as possible. The electromagnet we were using was rated at 12V and has a max current of .33A that it will pull. We ran the magnet at 7.2V and it pulled .18A. The electromagnet was turned on and off by a MOSFET acting as a switch. The circuit can be seen in Figure 3.2. A diode was placed parallel to the electromagnet to prevent voltage from flowing backwards to the FPGA and ruining a pin.

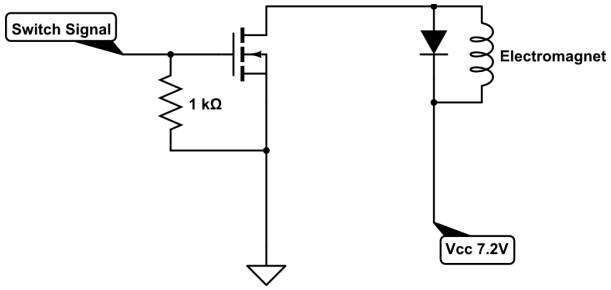

Figure 3.2 – Electromagnet MOSFET Switch

4. LED Detection System

The LED detection system consists of a phototransistor, two amps, a bandpass filter, a half-wave rectifier and a DC smoothing capacitor. As seen in Figure 4.1, the phototransistor is an NPN BJT with the base being used to absorb light. When the phototransistor is receiving a 1kHz signal from a blinking LED point blank, the output from the emitter pin of the transistor is 8V peak to peak with a 4V being the max. In order to increase the range of the photo transistor, a 100:1 inverting op-amp is added.

This increases the range to about 3inches with a 7V peak to peak with a 4V max. Another 100:1 inverting op-amp is added and this increases the range to about 15 feet with 6V peak to peak with a 3V max. The test LED beacon we were using to test the range had two red LEDs on it while the actual beacon that would be used in the demo had 6 red LEDs that were much brighter than ours. So while we could not achieve 20ft with our beacon, we were able to achieve that easily with the real beacon.

After increasing the range of the phototransistor, a band pass filter was added in order to filter out overhead light which works at 60/120 Hz. We followed the band pass filter design from TI's Filter Design in 30 Seconds that most of the class was using. Our filter is centered around 1kHz which is what the beacon will be blinking at. After the band pass filter, the output was a clipped 1kHz signal when the beacon was shown on it and < 1V noise when in ambient light. Finally a half-wave rectifier with a smoothing capacitor were added to the end. The output from this was ~2.7-2.8V when detecting the beacon and <200mV when not detecting the beacon in ambient light.

The phototransistor can detect the beacon from multiple angles so in order to make its detection angle narrower, we used heat shrink to limit the phototransistor to only receive light from directly in front of it. This still left about a 15 degree field of vision to the phototransistor. So in order to make this smaller, we placed the phototransistor inside a 1 in PVC pipe that was colored black by sharpies on the inside.

The tank had two of these LED detection circuits, one on each side of the tank.

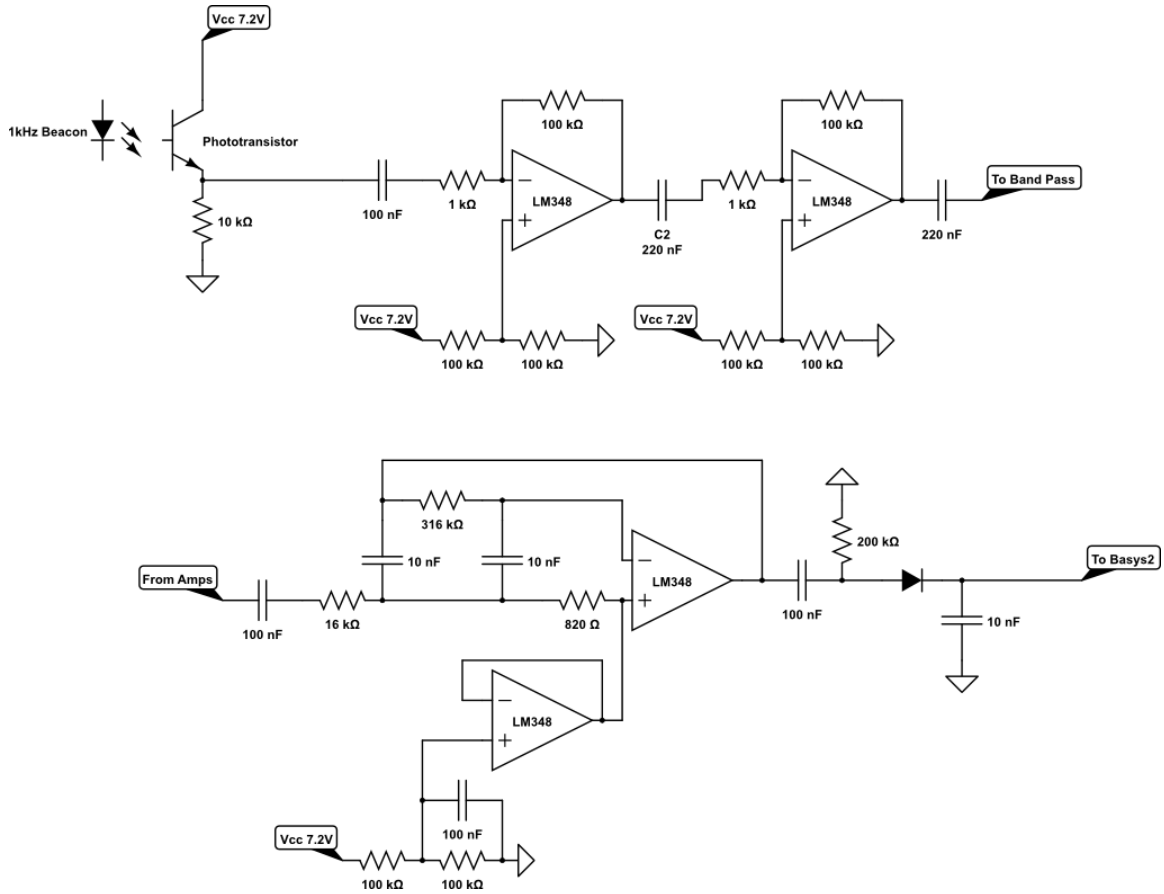

Figure 4.1 – LED Detection Circuit

5. Power System

When the three sensor/output systems were connected together, the PWM from the motors and the PWM that controlled the servo in the metal detector caused an ignorable amount of noise in the LED detection system. So a regulated power supply was needed. The power was supplied by two 7.2V NiMH batteries. The first battery was plugged into the power supply shown in Figure 5.1. This power supply was used for powering the Basys2 Board and the two LED detection circuits. The second battery was plugged directly into the main board and was used to power the motors, servo, and electromagnet. The two grounds from the two batteries were tied together to create a shared ground.

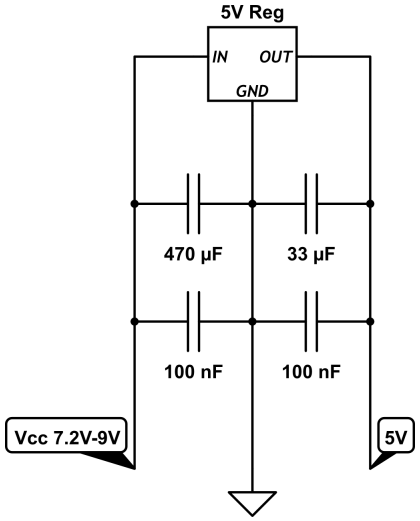

Figure 5.1 – Regulated Power Supply

6. Control System

The control system consists of a Spartan-3E FPGA on the Diligent Basys2 Board. This board has 16 I/O pins in four sets of 4. The first 12 pins on the board were electrically isolated with opticouplers. The motor system uses 6 outputs, the electromagnet uses 1, and the servo that the electromagnet was on uses 1. These 8 are all the outputs from the FPGA. The metal detectors uses 4 inputs to the FPGA. These were electronically isolated because they output 7.2V and instead of doing voltage division to reduce the voltage, isolating via opticoupler seemed simpler since they were already set up on the board. The LED detection system could not be electrically isolated because the output from the circuit did not have enough current to drive the LED in the opticoupler. Since the output from the LED circuit was already in the logic high range of the Basys2, all that was needed was a pull down resistor to be added to the LED detection circuit for when it was plugged into the Basys2.

The control system uses a basic state machine in order to control what the tank does as seen in Figure 6.1. The tank initially starts in the Lost state and will rotate

clockwise till it either both LED detection circuits register a high or a metal detector detects a mine. If the beacon is found, then the system moves to the Found state and drives forward till a mine is detected. The Mine Found state stops the tank and moves the arm on the servo to line up the magnet in front of 1 of the 4 metal detectors that detected the mine. Then the system goes into a Mine Response state where it does the following actions:

1. Backs up the tank. - 1.6 seconds
2. Stop and wait while the Electromagnet turns on. – 1.2 seconds
3. Move forward slowly with the Electromagnet on. – 2 seconds
4. Turn off magnet, move servo arm back to center/default position and go back to the Lost state.

Table 6.1 – Mine Response

The code for the state diagram can be seen in Appendix C-1.

7. Conclusion

The system worked to expectations. The LED detector circuit paired with the bright LEDs on the beacon caused some confusion, as it was able to pick up the signal in the reflection of any shiny metal surface. The metal detectors had a limited range so selecting a different one might be advantages for future projects but it worked for the demo for our tank.

Appendix A

Appendix A is a list of references and data sheets for the materials used in the design of the system.

1. L298 Datasheet:

   https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

2. Rover 5 Robot Platform:

   https://www.sparkfun.com/products/10336

3. Basys2 Board

   http://www.digilentinc.com/Products/Detail.cfm?Prod=BASYS2

4. Metal Detector Circuit

   http://www.thomasathomas.com/Metal_detectors_work.htm

5. Power Supply Noise Reduction

   http://www.designers-guide.org/Design/bypassing.pdf

6. Signal Diode and Switching Diode Characteristics

   http://www.electronics-tutorials.ws/diode/diode_4.html

7. 555 Timer Datasheet

   http://www.ti.com/lit/ds/symlink/ne555.pdf

8. Metal Detector Circuit based on 555 Timer

   http://www.555-timer-circuits.com/metal-detector.html

9. Verilog PWM Example

   http://www.ece301.com/fpga-projects/53-pwm.html

Appendix B

Appendix B is a list of calculations used in the design of the system.

1. Calculations for determining the duty step of the PWM

   50MHz clock on Basys2: C = 50,000,000/second

   Period for Pulse: P = 1ms (picked arbitrarily)

   Cycles per Period: C/P = 50,000 cycles

   8bit resolution: 2^8 = 256

   Duty Step = (C/P)/258 =~ 195

Appendix C

Appendix C is a list of the source code used within this system.

1. Main Module Source Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
/////

module main(
       input [7:0] dipSwitch,
       input [3:0] buttons,
       output [7:0] leds,
       input clock,
       output [1:0] pwm,
       output [3:0] motorPins,
       output clockOut,
       output servoSignal,
       input PT1,
       input PT2,
       output magnetPin,
       input [3:0] metalDetectorPin //left,right,middle right, middle left
);
assign clockOut = clock;

reg [7:0] motorPowerLevel = 0;
wire [1:0] stallswitch = 2'b00;
reg [3:0] motorDirection;
//4'b1010 backwards
//4'b1001 rotate clockwise
//4'b0110 rotate counterclockwise?
//4'b0101 forward

reg [7:0] servoDirection;
//left 1111_0000 240
//right 0011_1100 60
//midrig 0110_0100 100
//midlef 1011_1100 188

reg [3:0] metalDetected;
assign leds[7:4] = metalDetected;
reg magnetOn; //0 off/ 1 on

parameter lost=3'b001, found=3'b010, mineFound=3'b100,
mineResponse=3'b101;//list of states
reg [2:0] state;
reg [31:0] counter;


always@(posedge clock)
begin
if (state == 0) begin
       state <= lost;
end else if(metalDetectorPin && (state==lost || state==found)) begin
       state <= mineFound;
end else begin
              case(state)
                     lost: begin
                            motorDirection <= 4'b1001; //rotate clockwise
                            motorPowerLevel <= 160;
                            servoDirection <= 144;
                            if((PT1 == 1) & (PT2 == 1))
                            begin
```

```
                                state <= found;
                        end
                end
                found: begin
                        motorDirection <= 4'b0101; //forward
                        motorPowerLevel <= 192;
                        servoDirection <= 144;
                        if((PT1 == 0) | (PT2 == 0))
                        begin
                                state <= lost;
                        end
                end
                mineFound: begin
                        motorDirection <= 4'b0000; //stop
                        motorPowerLevel <= 0;
                        case(metalDetectorPin)
                                4'b0001: servoDirection <= 240;
                                4'b0010: servoDirection <= 60;
                                4'b0100: servoDirection <= 100;
                                4'b1000: servoDirection <= 188;
                        endcase
                        counter = 0;
                        state <= mineResponse;
                end
                mineResponse: begin
                        counter = counter + 1;
                        if(counter < 80_000_000) begin //backup
                                motorDirection <= 4'b1010;
                                motorPowerLevel <= 160;
                        end else if (counter > 80_000_000 && counter <
140_000_000) begin //stop and turn on magnet
                                motorDirection <= 4'b0000;
                                motorPowerLevel <= 0;
                                magnetOn <= 1;
                        end else if (counter > 140_000_000 && counter <
240_000_000) begin //move foward slowly with magnet on
                                motorDirection <= 4'b0101;
                                motorPowerLevel <= 160;
                                magnetOn <= 1;
                        end else if (counter > 240_000_000) begin // turn
off magnet and look for beacon
                                magnetOn <= 0;
                                state <= lost;
                                counter = 0;
                        end
                end
                default: state <= lost;
        endcase
end


end

motor leftMotor (
.enable                         (motorDirection[1:0]),
.directionOutput    (leds[1:0]),
.pinOutput                      (motorPins[1:0]),
.stall                          (stallswitch[0])
);

motor rightMotor (
.enable                         (motorDirection[3:2]),
.directionOutput    (leds[3:2]),
.pinOutput                      (motorPins[3:2]),
.stall                          (stallswitch[1])
);
```

```
pwm rightMotorPWM (
.clk                 (clock),
.pwmOutput    (pwm[0]),
.powerLevel (motorPowerLevel)
);

pwm leftMotorPWM (
.clk                 (clock),
.pwmOutput    (pwm[1]),
.powerLevel (motorPowerLevel)
);

magnetControl magnet(
.on                  (magnetOn),
.signal       (magnetPin)
);

servo magnetServo (
.clk                 (clock),
.speed        (servoDirection),
.signal       (servoSignal)
);

endmodule
```

2. Motor Module Source Code

```
module motor(
       input [1:0] enable,
       output [1:0] directionOutput,
       output [1:0] pinOutput,
       input stall
   );

reg[1:0] direction;
reg[1:0] pins;
assign directionOutput = direction;
assign pinOutput = pins;

always@(enable or stall)
       if(stall) begin
              direction = 2'b00;
              pins = 2'b00;
       end else begin
              case(enable)
                     2'b00: direction=2'b00;
                     2'b01: direction=2'b01;
                     2'b10: direction=2'b10;
                     2'b11: direction=2'b00;
                     default: direction=2'b00;
              endcase
              case(enable)
                     2'b00: pins=2'b00;
                     2'b01: pins=2'b01;
                     2'b10: pins=2'b10;
                     2'b11: pins=2'b11;
                     default: pins=2'b00;
              endcase
       end
endmodule
```

3. Motor PWM Module Source Code

```
module pwm(
    input clk,
    output pwmOutput,
        input [7:0] powerLevel
    );

parameter sd = 19530;

reg pwm;
assign pwmOutput = pwm;
reg[31:0] counter = 0;

always@(posedge clk)
begin
       counter = counter + 1;
       if(counter <= powerLevel*sd) pwm=1;
       else pwm=0;
       if(counter >= 5000000) counter=0;
end

endmodule
```

4. Servo and Magnet Control Modules

```
module servo(
       input clk,
   input [7:0] speed,
   output signal
    );

parameter sd = 234;

reg pwm;
assign signal = pwm;
reg[19:0] counter = 0;

always@(posedge clk)
begin
       counter = counter + 1;
       if(counter <= 45000) pwm=1;
       else if (counter <= (45000 + sd*speed)) pwm=1;
       else pwm=0;
       if(counter >= 1000000) counter=0;
end

endmodule

module magnetControl(
    input on,
    output signal
    );

reg outputSig;
assign signal = outputSig;

always@(on)
begin
       outputSig = on;
end

endmodule
```