# Arduino-Based Line Follower with XBee Reporting and Configuration

Sean Hicks, *Student Member, IEEE, and Tanmay Sane, Student Member, IEEE*

***Abstract*—An Arduino-based line follower (ALF) with added wireless XBee communication capabilities used to control and select between several behavioral modes is presented. The ALF exhibits three modes of operation: left-edge following, right-edge following, and manual control. While typical line-following robots cannot be programmed dynamically, the ALF incorporates XBee-based radio communication to allow online reconfigurability and data reporting, while maintaining autonomy when desired.**

***Index Terms*—Line following, Arduino, Xbee, .AT commands**

## I. Introduction

A line follower employs a sensing mechanism to adhere to the boundary between two optically diverse regions during propulsion. The Arduino-based line follower (ALF) presented here uses a discrete infrared (IR) LED and photosensitive transistor to generate a voltage signal proportional to the reflected intensity. Thus, the ALF follows a line by continually comparing the reflected intensity to the midpoint between the two intensity regions and then controlling the vehicle's propulsion to minimize the error from that midpoint. Furthermore, the ALF communicates to a base station wirelessly via an XBee shield. This provides a two-fold advantage over standard line followers without communication capabilities: it allows remote commands to change the operational mode of the ALF between left- and right-edge following and a manual mode where the vehicle is driven using terminal commands at the base station; and it allows reporting from the ALF to the base station of its current operational mode and could in future work report intensity values to enable color-mapping capabilities of remote areas.

The remainder of this paper is divided as follows: Section II offers a brief background of line following implementations; Section III provides detailed descriptions of the design; IV documents the results of this project; and V concludes and anticipates future work.

## II. Background

When IR light is shone on a surface, as with any other wavelength, a portion of the light reflects. In general, white exhibits high reflectance, while black exhibits low reflactance. Thus, by placing an IR emitter (an LED) alongside a receiver and shining on a close surface beneath a vehicle, the intensity in the vicinity of the detector will correlate to the color of the surface.

Sean Hicks and Tanmay Sane are graduate students in the Department of Electrical and Computer Engineering, University of North Carolina Charlotte, Charlotte, North Carolina. (e-mail: tsane@uncc.edu)

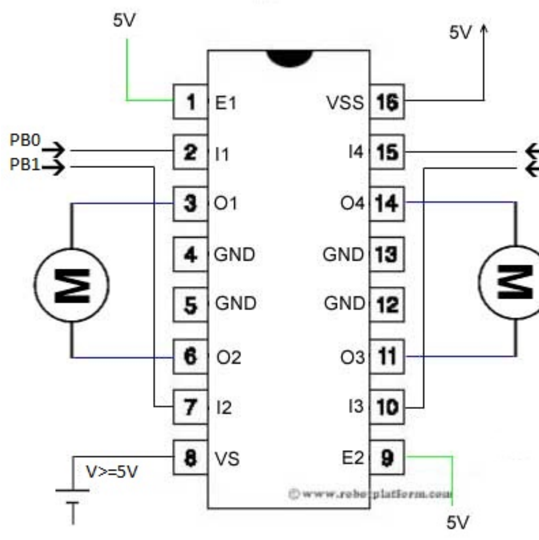

Figure 1. L293D Connections.
Source: robotplatform.com/howto/L293/motor_driver_1.html

## III. Design

The ALF consists of an Arduino RedBoard, XBee shield, chassis, motors, wheels, electrical connectors, and custom sensing board. The design is divided into the following subsystems: power, propulsion, light intensity sensing, line-following, and communication.

### *A. Power*

Power originates from a 9 V battery, which routes into the Arduino RedBoard's power input barrel jack, after which it is converted by an onboard low-dropout voltage regulator to 5 V, which is distributed to the rest of the circuitry.

### *B. Propulsion*

Two wheels are each driven by Solarbotics gear motors (GM9) with gear ratio equal to 143:1. The motors are driven by an L293D H-Driver, with connections shown in Fig. 1.

### *C. Light Intensity Sensing*

By energizing the IR LED and connecting the detector to the analog circuit of Fig. 2 connected to the Arduino's analog-to-digital converter (ADC), then placing this assembly beneath the chassis, the ALF generates a digitized value proportional to the light intensity reflecting off the surface below the vehicle.

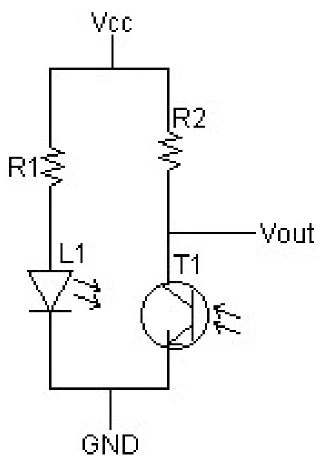

Figure 2. IR LED and detector circuit
Source: societyofrobots.com/schematics_infraredemitdet.shtml

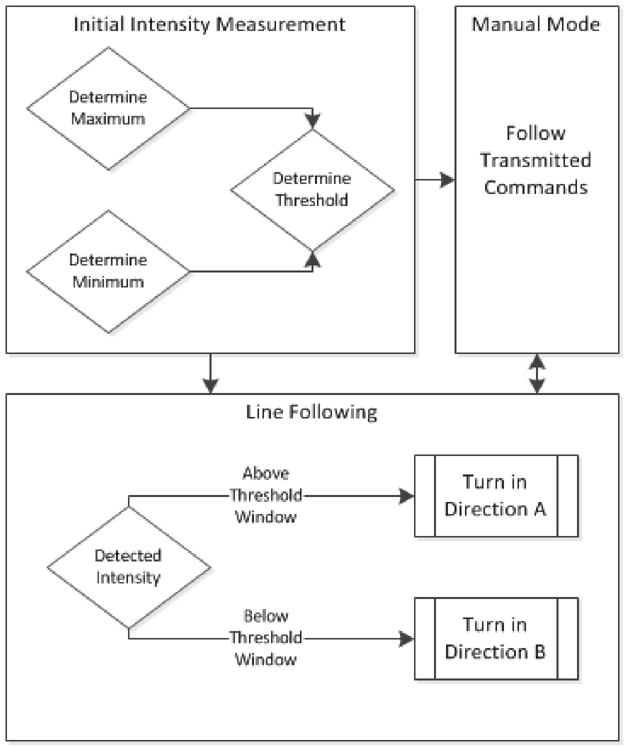

Figure 3. ALF Software Algorithm

### D. Line-Following Algorithm

Fig. 3 shows the algorithm used to establish a midpoint threshold, operate in manual mode, and operate in either left- or right-edge following mode. Upon power-up, the ALF sweeps laterally while recording the highest and lowest intensity values detected so far. After a predetermined period, this collection phase stops, and the Arduino calculates the midpoint between the two extreme values and sets this midpoint as the threshold. The code for this behavior is provided below:

```
// This code runs prior to the while(1) loop
uint8_t adc_min = 0xFF;
uint8_t adc_max = 0;
uint8_t adc_desired;
...
for ( int k = 0; k < 1000; k++ )
{
        adc_in = read_adc_value(1);
        if ( adc_in < adc_min )
                adc_min = adc_in;
        if ( adc_in > adc_max )
                adc_max = adc_in;
        adc_desired = (adc_max + adc_min)/2;
        ...
        PORTB = CIRCLE_LEFT;
        for ( int m = 0; m < 4000; m++ ){}
        PORTB = STOP;
        for ( int m = 0; m < 6000; m++ ){}
}
PORTB = STOP;
```

The mode of operation is determined by receiving ’l’, ’r’, or ’m’ to enter left-edge following, right-edge following, or manual mode, respectively. In manual mode, the ALF begins halted, and will drive forward, backward, circle left, or circle right upon receiving ’w’, ’s’, ’a’, or ’d’, respectively. The code for this behavior is provided below:

```
switch ( ReceivedByte )
{/*   state_var = 0: idle, 2: left-edge, 3:
    right-edge,
      10: man. stop, 11: man. straight, 12:
          man. left, 13: man. back, 14: man.
          right*/
      case 'm':    PORTB = STOP;
                   state_var = 10;
                   UDR0=10;
                   break;
      case 'w':    if ( state_var > 3 )
                   {
                        state_var = 11;
                        PORTB = FORWARD;
                        UDR0=11;
                   }
                   break;
      case 'a':    if ( state_var > 3 )
                   {
                        state_var = 12;
                        PORTB =
                            CIRCLE_LEFT;
                        UDR0=12;
                   }
                   break;
      case 's':    if ( state_var > 3 )
                   {
                        state_var = 13;
                        PORTB =
                            BACKWARD;
                        UDR0=13;
                   }
                   break;
      case 'd':    if ( state_var > 3 )
                   {
                        state_var = 14;
                        PORTB =
                            CIRCLE_RIGHT;
                        UDR0=14;
                   }
                   break;
      case 'l':    state_var = 2;
                   invert_sensing = 1;
                   UDR0=2;
```

```
                        break;
      case 'r':   state_var = 3;
                        invert_sensing = 0;
                        UDR0=3;
                        break;
      default:    ;
}
ReceivedByte = 0;
```

Next, if autonomous mode is requested, in a timed loop, the current detected value is compared to the threshold (with an added tolerance to provide a small window of straight motion). Depending on the result of this comparison, the vehicle corrects in the appropriate direction. This effects left-edge following of a dark line on white floor. To achieve right-edge following, the output of the threshold comparison is simply toggled. The code for this behavior is provided below:

```
// This code runs inside the while(1) loop
if ( state_var == 2 || state+var == 3 )
}
      adc_in = read_adc_value(1);
      if ( (adc_in < adc_desired-5) ^
          invert_sensing )
      {
            PORTB = CIRCLE_RIGHT;
      }
      else if ( (adc_in > adc_desired+5) ^
          invert_sensing )
      {
            PORTB = CIRCLE_LEFT;
      }
}
```

### E. Communication

The ALF Arduino directly communicates with its XBee by placing data in the UDR0 register and serially transmitting it to the XBee, which in turn wirelessly transfers this data across the XBee network to the coordinator. Finally, the coordinator hands this information to a laptop computer for processing. Information is received by the ALF from the base station in a similar manner, with the ALF XBee transmitting to the Arduino's receive pin, resulting in the data residing in UDR0.

The ALF XBee is configured in accordance with Ref. [1] as an end device in AT mode, while the base station XBee is configured as a coordinator in AT mode. The devices communicate at 9600 baud with no parity, no flow control, eight data bits, and one stop bit. Commands are transmitted to the ALF using X-CTU on a laptop computer.

## IV. Result

The ALF properly transitions between all modes of operation and successfully adheres to edges, whether curved, straight, or angled 90°. It responds to all manual-mode commands.

## V. conclusion and Future Work

The ALF successfully integrates wireless communication with an autonomous vehicle and demonstrates the capability to self-navigate using minimal circuitry. Furthermore, it suggests the capability to remotely explore unknown terrain and transmit information about that terrain back to a base station.

## References

[1] Digi International, Inc. *XBee$^{TM}$ ZNet 2.5/XBee-PRO$^{TM}$ ZNet 2.5 OEM RF Modules* 2008.