EN1093

LABORATORY PRACTICE - I

LAB PROJECT

LINE FOLLOWING AND OBSTACLE AVOIDING ROBOT

GROUP 12

180379R	MAHEEKUMARA K.A.G.D.
180391V	MAYOORAN T.
180398A	MENDIS N.P.A.
180402J	MISHANTH P.

CONTENT

Introduction

Atmel Atmega32 was used to control the robot in order to perform line following and obstacle avoiding. Custom made 7 TCRT5000 (IR sensors) was used to construct the sensor panel. L298N was used to drive 2 DC gear motors of 150rpm which draws 12V. The robot was powered by Li Po battery of 2200 mAh 25c. The program written in C (programming language) was uploaded to the Atmega32 by Atmel Studio and burned by Extreme Burner. The chassis of the robot is made of Acrylic boards. Screws and nuts were used in assembling processes.

When it comes to the electronic part, there are 6 printed circuit boards in order to perform specific tasks.

1) Motor Controller Circuit
2) Sensor panel circuit
3) Comparator circuit
4) Power supply circuit
    a) Output 5V
    b) Output 12V
5) Micro controller circuit

Inputs and outputs of each circuits were passed to the relevant circuit using connectors. LEDs were used to indicate certain logical and power outputs. Each PCBs were established using iron and etch method. Almost all the components (except 2 inductors which were surface mounted) were through hole mounted and soldered.

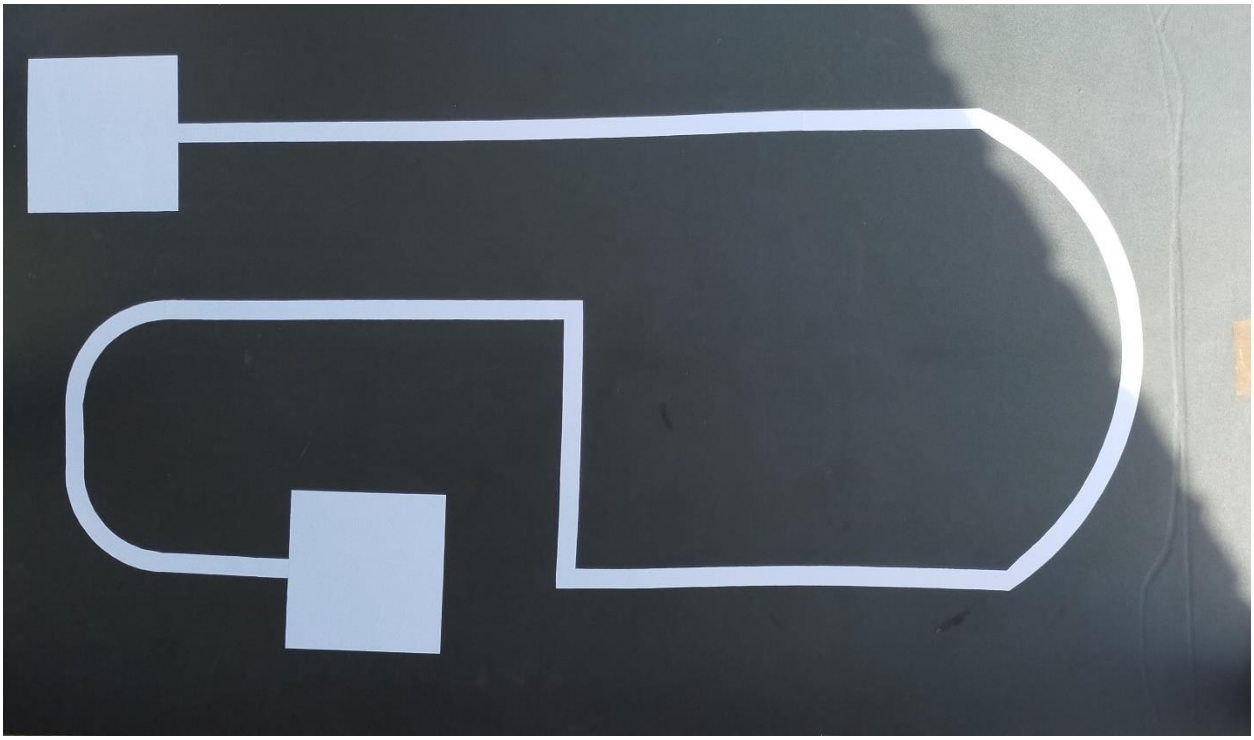
Sharp IR sensor is used for the obstacle avoiding purposes. A caster wheel is placed at the center of the front part of the chassis.

Task

The robot is supposed to perform line following along with obstacle avoiding. The arena for perform this task was made of wooden plate of 2m × 1.2m in dimension and was covered by black sticker which has same dimensions. The start and end for the task was designed of 25cm × 25cm white squares. The lines were of 3 cm width. The arena consists of 90° sharp bends and consists of smooth curvy bends. The bends were designed in the way not to make the robot cut two lines at any same instant. The lines and the squares are placed with an offset from the edges of the wooden board so that the robot would not fall outside the arena.

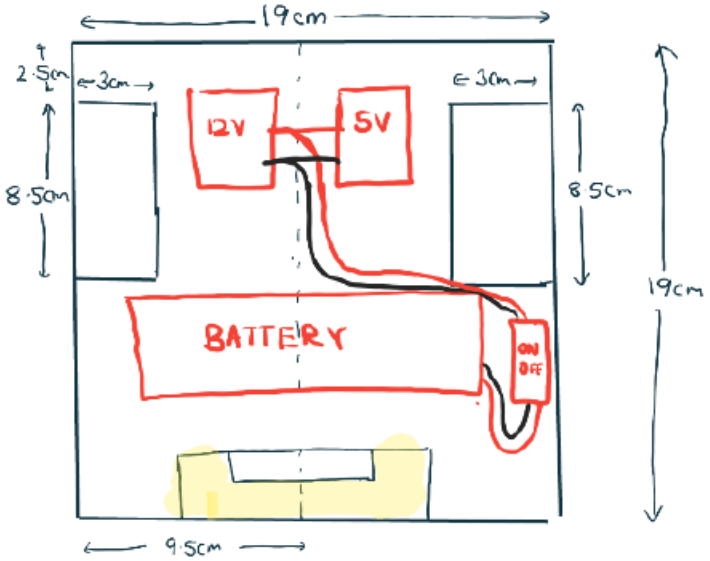
The robot is supposed to avoid an obstacle which is placed on any straight line and get back to the line in the shortest possible distance. The obstacle is a cylinder made of cardboard which has a diameter of 10cm and height of 25cm. The obstacle is placed on the longest line which enables the robot to perform a convenient turn and get back to the line.

The robot is supposed to start and find the white line when it placed on the white square and to stop the wheels as it reaches the white square.



Robot specifications

The robot was supposed not to exceed 25 cm × 25 cm in length and width. The robot that we designed was 21cm × 20 cm. The sensor panel is fixed in a way that the middle 3 sensors could be aligned inside the line so that it could be programmed to maintain the optimum line following. The robot was designed using 2 acrylic boards aligned one top of the other, because it went impossible to get down all the circuits to a single storey robot. The components which weigh much were assembled in the bottom acrylic board so that the gravitation center is maintained at the lower most place of the robot in order to stabilize the structure.



Bottom acrylic board

Hardware

1. Motor controller Circuit

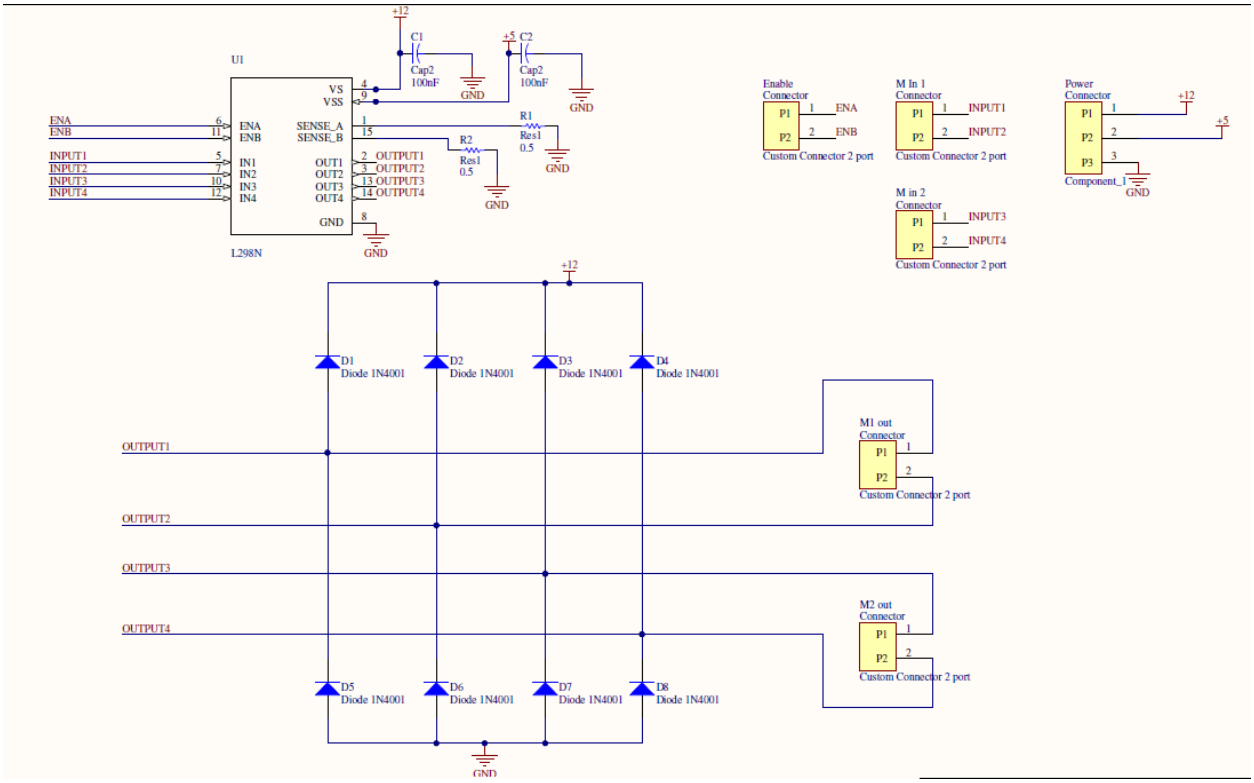
Motor controller is a current enhancing device. It can also be act as Switching Device. Thus, after inserting motor driver among the motor and microcontroller. Motor driver taking the input signals from microcontroller and generate corresponding output for motor. L298N is the IC we used to perform the above task.

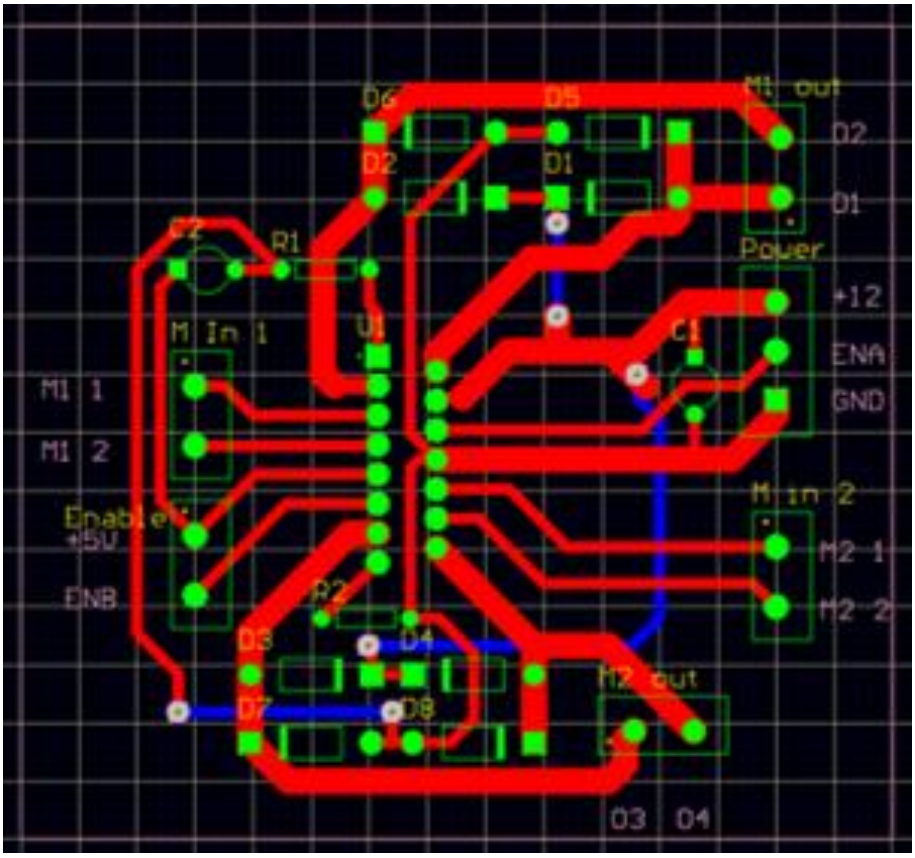
L298N specifications

- Double H bridge drive
- Chip L298N (ST NEW)
- Logical voltage 5V
- Drive voltage 5V-35V
- Logical current 0mA-36mA
- Drive current 2A (MAX single bridge)
- Storage temperature -20 to +135
- Max power 25W

The above-mentioned IC which uses the H-bridge technology to change the direction of the motor is used to control the DC gear motors.

Diodes were used to resist back EMF from the motors. The biasing voltage of 5V and a supply voltage of 12V were provided to the IC. IN1, IN2, IN3 and IN4 were passed from the microcontroller in order to give the directions to the motors. ENA and ENB were passed from the microcontroller in order to determine the speed of the gear motors.


U1
L298N
C1
Cap2
100nF
C2
Cap2
100nF
R1
Res1
0.5
R2
Res1
0.5
+12
+5
GND
ENA
ENB
INPUT1
INPUT2
INPUT3
INPUT4
SENSE_A
SENSE_B
OUTPUT1
OUTPUT2
OUTPUT3
OUTPUT4
Enable Connector
M In 1 Connector
M in 2 Connector
Power Connector
Component_1
Custom Connector 2 port
D1 Diode 1N4001
D2 Diode 1N4001
D3 Diode 1N4001
D4 Diode 1N4001
D5 Diode 1N4001
D6 Diode 1N4001
D7 Diode 1N4001
D8 Diode 1N4001
M1 out Connector
M2 out Connector

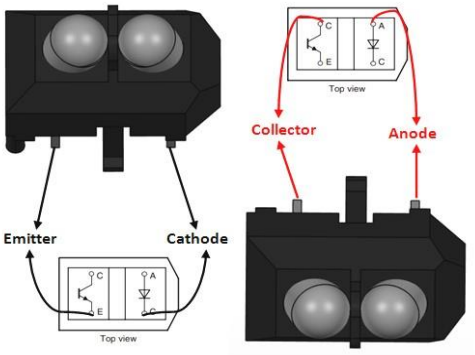

D6
D5
M1 out
D2
D1
C2
R1
Power
+12
M In 1
U1
C1
ENA
GND
M1 1
M1 2
M in 2
M2 1
M2 2
ENB
R2
D3
D4
M2 out
D7
D8
03
04

2. Sensor Panel

Components

i. TCRT5000 *7
ii. 330ohm *7

TCRT5000



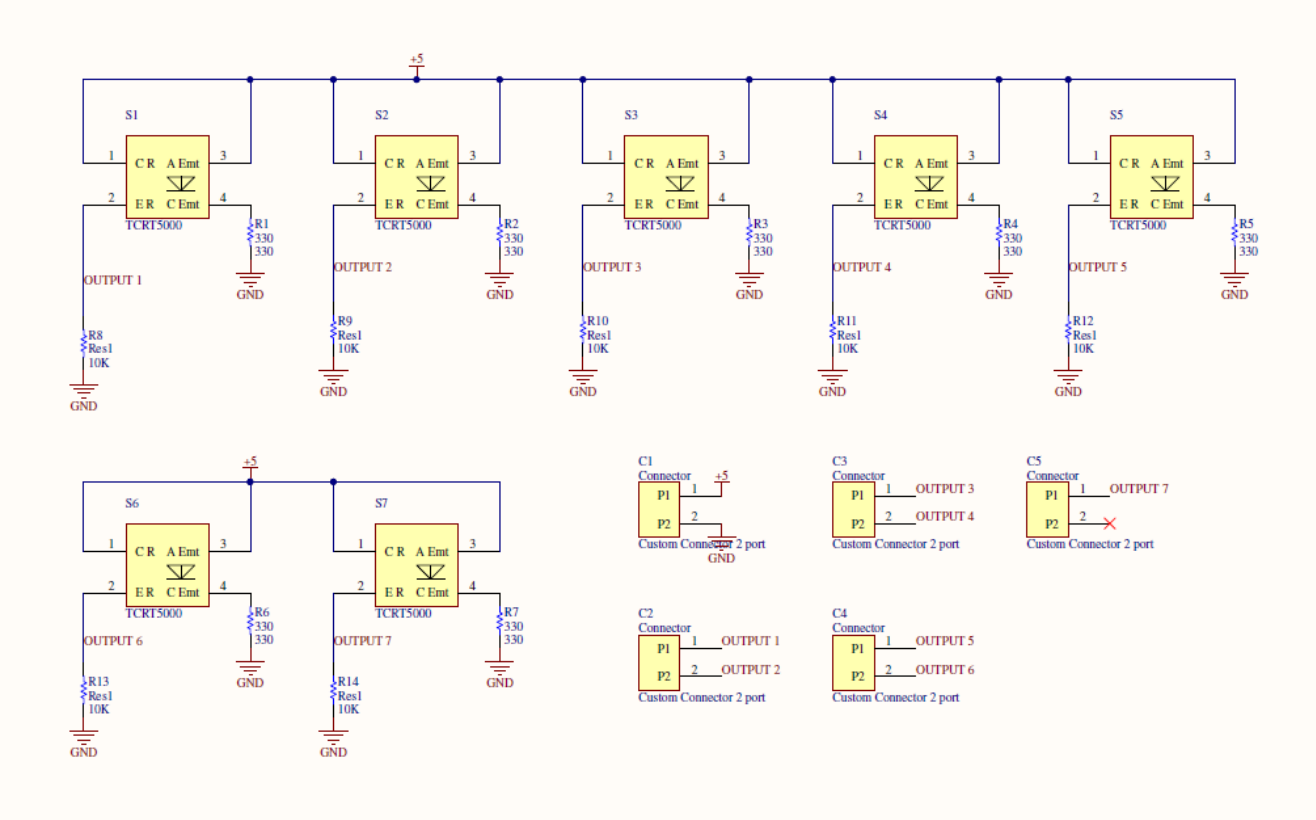
TCRT5000 is a reflective sensor which include an infrared emitter and phototransistor. It can be used for identifying the difference between white and black.

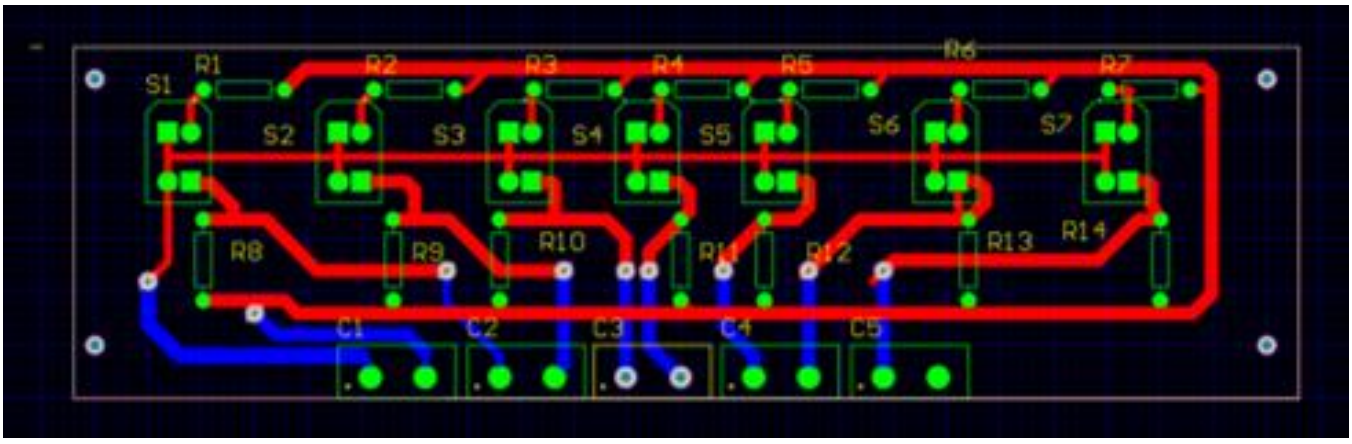
Features

- Sensing Distance: 0.2 mm to 15 mm
- Dimensions (L x W x H in mm): 10.2 x 5.8 x 7
- Output Device: Phototransistor
- Collector- Emitter Voltage VCEO Max: 70 V
- Maximum Collector Current: 100 mA
- Forward Voltage: 1.25 V
- Reverse Voltage: 5 V
- Operating Temperature Range: - 25 C to +85 C
- Sensing Method: Reflective
- Wavelength: 950 nm

Advantage

- Constant distance between transmitter and receiver
- Noise protected
- Operating temperature range is suitable for SriLankan climate


+5
S1
S2
S3
S4
S5
S6
S7
C R A Emt
E R C Emt
TCRT5000
R1 330 330
R2 330 330
R3 330 330
R4 330 330
R5 330 330
R6 330 330
R7 330 330
OUTPUT 1
OUTPUT 2
OUTPUT 3
OUTPUT 4
OUTPUT 5
OUTPUT 6
OUTPUT 7
R8 Res1 10K
R9 Res1 10K
R10 Res1 10K
R11 Res1 10K
R12 Res1 10K
R13 Res1 10K
R14 Res1 10K
GND
C1 Connector
C2 Connector
C3 Connector
C4 Connector
C5 Connector
P1
P2
Custom Connector 2 port

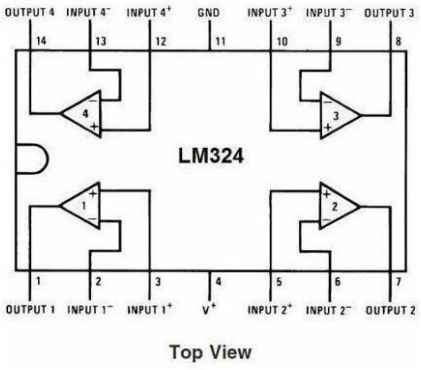

R1
R2
R3
R4
R5
R6
R7
S1
S2
S3
S4
S5
S6
S7
R8
R9
R10
R11
R12
R13
R14
C1
C2
C3
C4
C5

3. Comparator circuit

Components

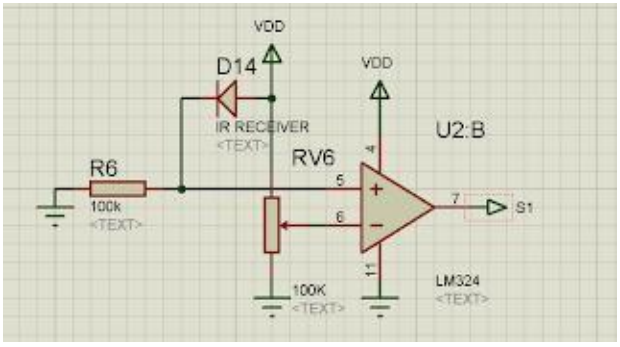
i. 10k *7
ii. 10k Variable Resistors *7
iii. LM324N Operational Amplifier *2

LM324N Operational Amplifier



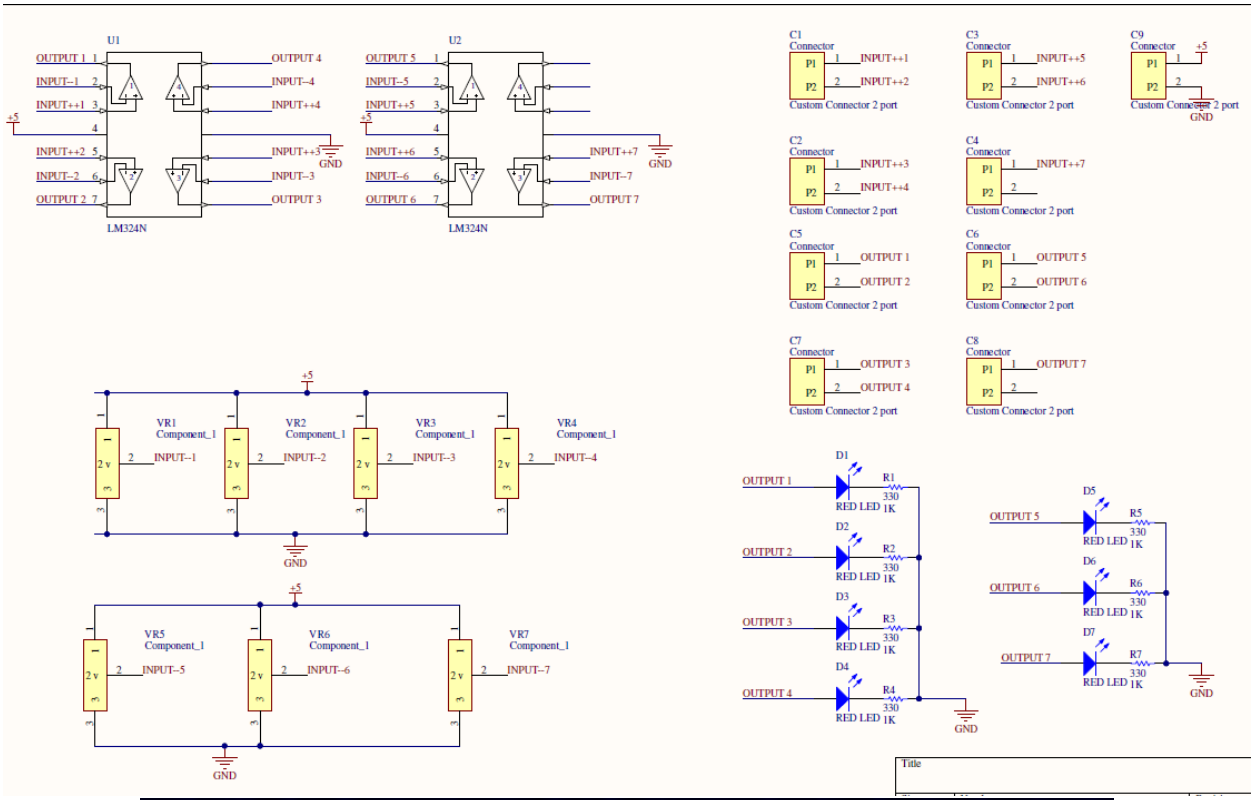
It is used to compare the output voltage of the phototransistors against a fixed voltage and output 5V or 0V corresponding to the digital 1 and 0.

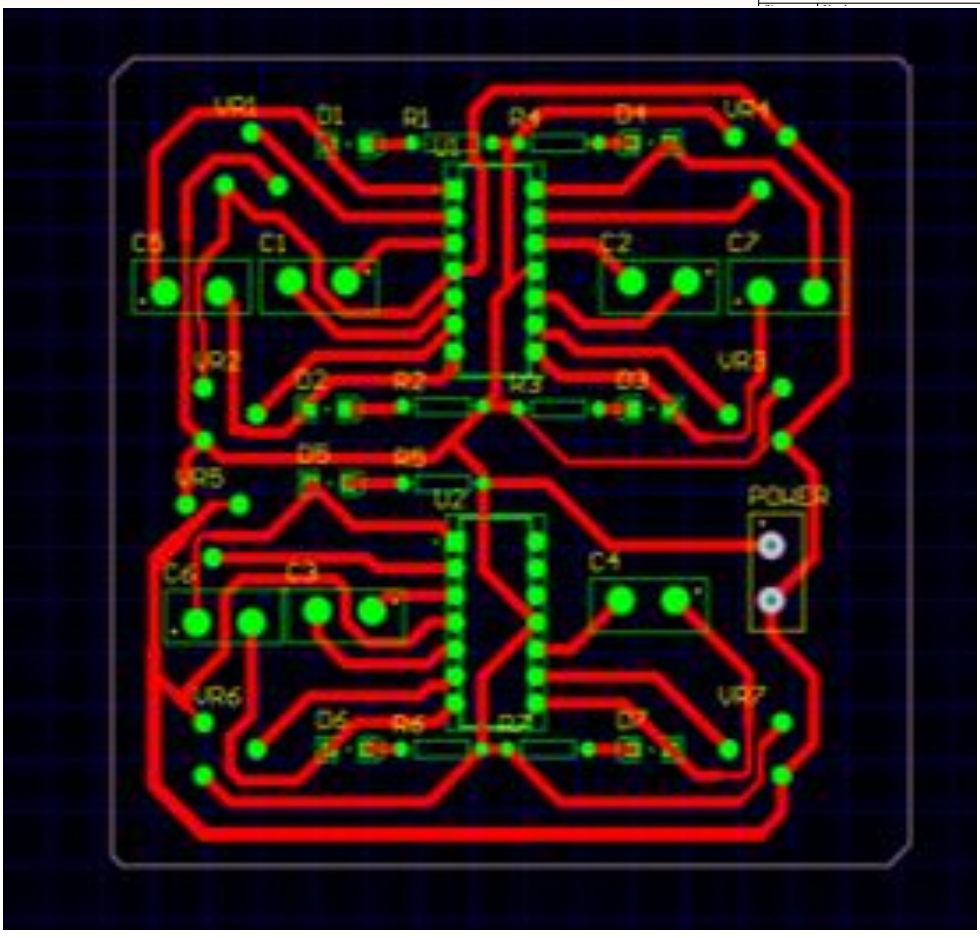
Tentative circuit plans



Purpose of variable resistors

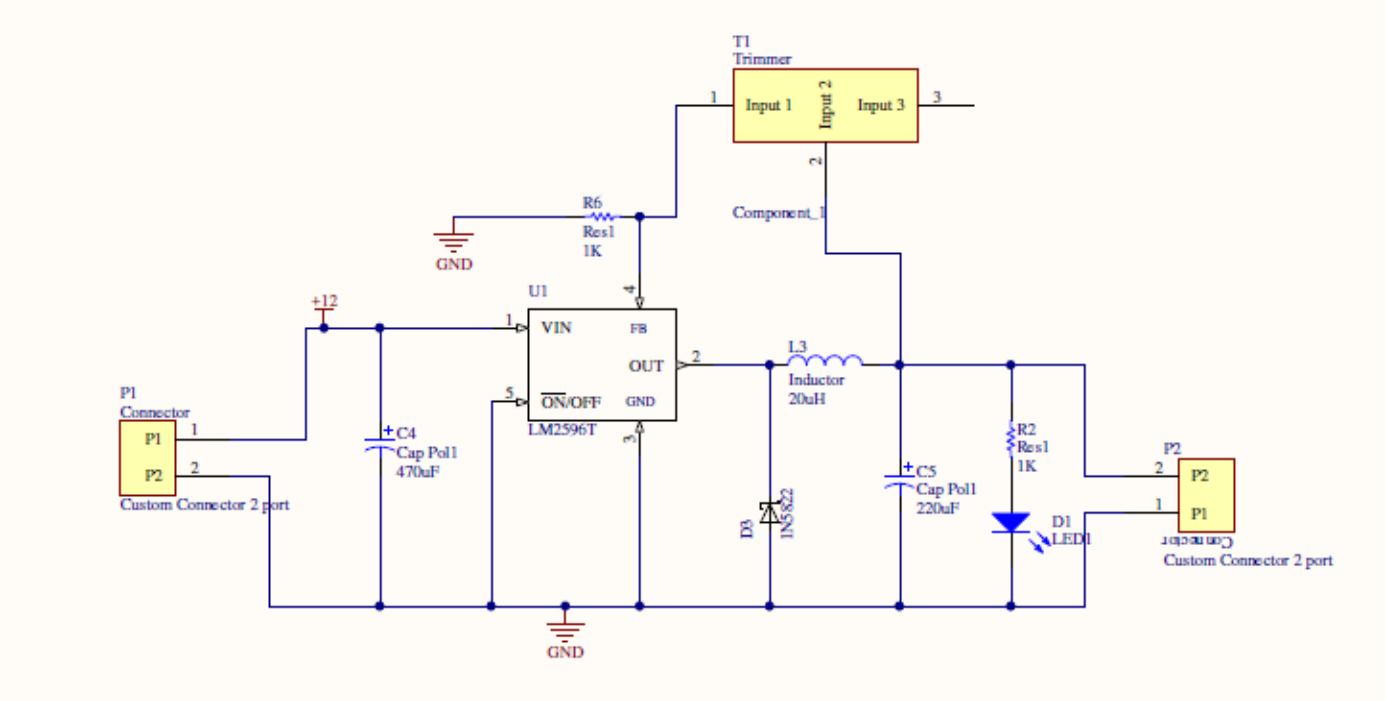
To calibrate in different lightning conditions.

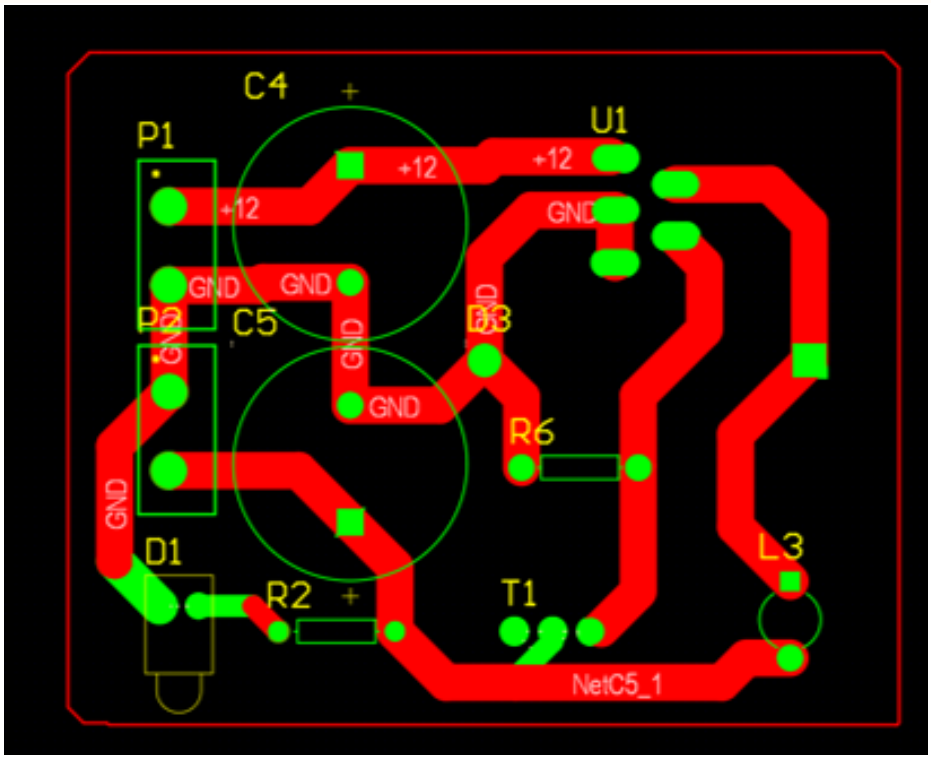

U1
OUTPUT 1
INPUT--1
INPUT++1
INPUT++2
INPUT--2
OUTPUT 2
OUTPUT 4
INPUT--4
INPUT++4
INPUT++3
INPUT--3
OUTPUT 3
LM324N
U2
OUTPUT 5
INPUT--5
INPUT++5
INPUT++6
INPUT--6
OUTPUT 6
INPUT++7
INPUT--7
OUTPUT 7
LM324N
GND
+5
C1 Connector
P1
P2
INPUT++1
INPUT++2
Custom Connector 2 port
C3 Connector
INPUT++5
INPUT++6
C9 Connector
C2 Connector
INPUT++3
INPUT++4
C4 Connector
INPUT++7
C5 Connector
OUTPUT 1
OUTPUT 2
C6 Connector
OUTPUT 5
OUTPUT 6
C7 Connector
OUTPUT 3
OUTPUT 4
C8 Connector
OUTPUT 7
VR1 Component_1
VR2 Component_1
VR3 Component_1
VR4 Component_1
INPUT--1
INPUT--2
INPUT--3
INPUT--4
VR5 Component_1
VR6 Component_1
VR7 Component_1
INPUT--5
INPUT--6
INPUT--7
D1
D2
D3
D4
D5
D6
D7
R1
R2
R3
R4
R5
R6
R7
330
1K
RED LED
Title


VR1
D1
R1
R4
D4
VR4
C5
C1
C2
C7
VR2
D2
R2
R3
D3
VR3
D5
R5
VR5
U2
POWER
C6
C3
C4
VR6
D6
R6
D7
VR7

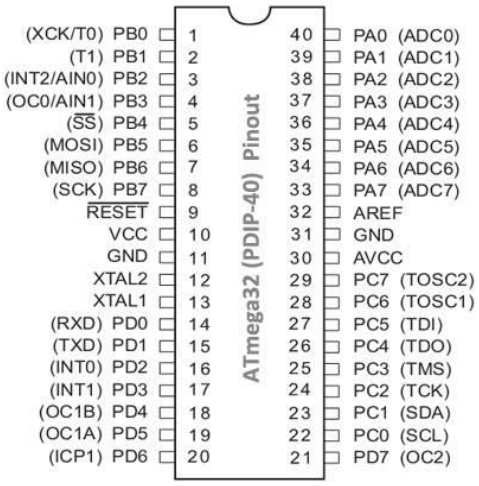
4. Power supply Circuit

Two power supply circuits were used in order to generate smoothened 5V and 12V. Buck converter technique is used in this circuits. Inductors were surfcae mounted in this circuit. Variable resister (trimmer) is used to fine tune the output voltage of these circuits.





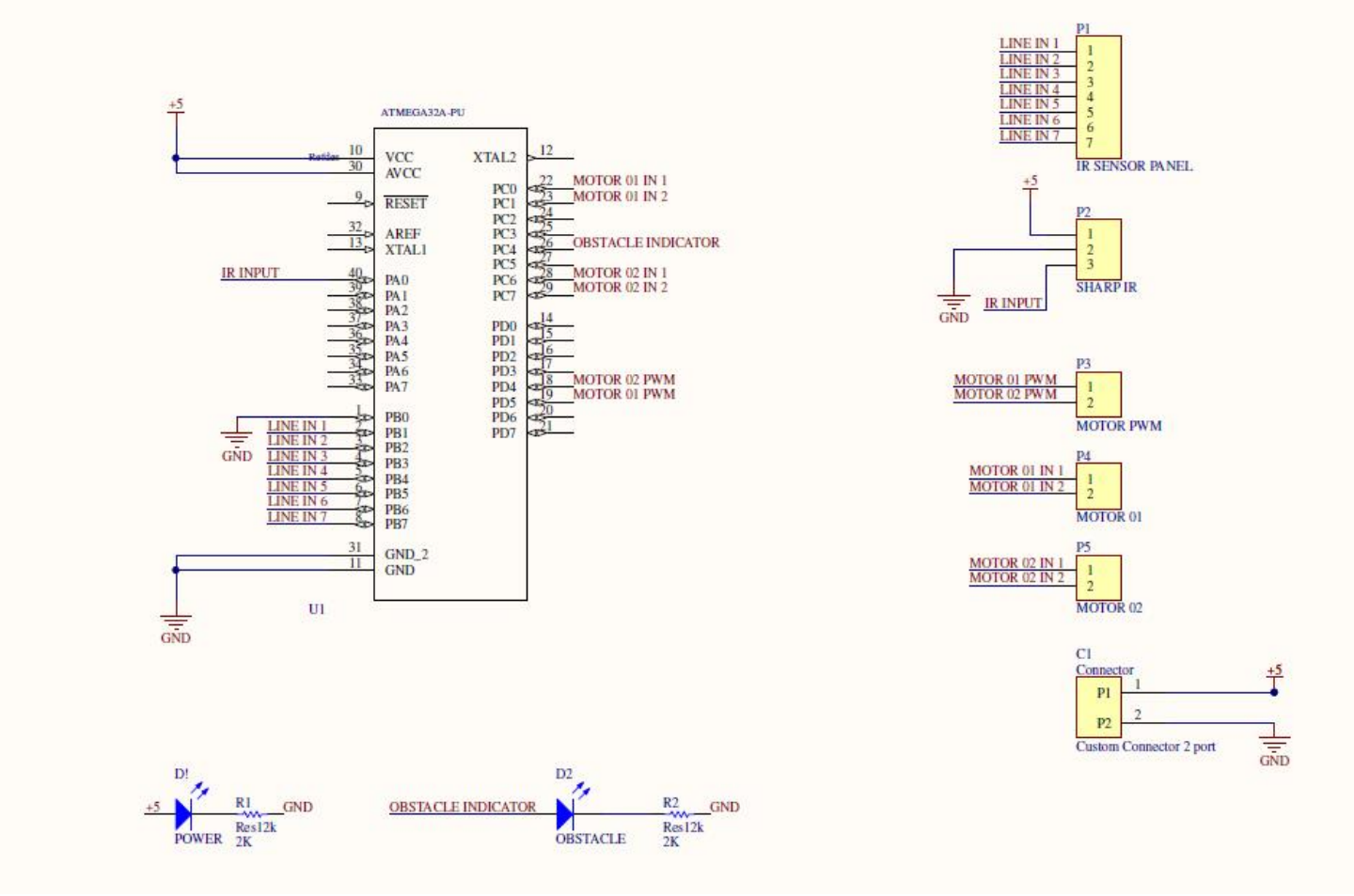
5. Microcontroller circuit - ATMEGA 32L

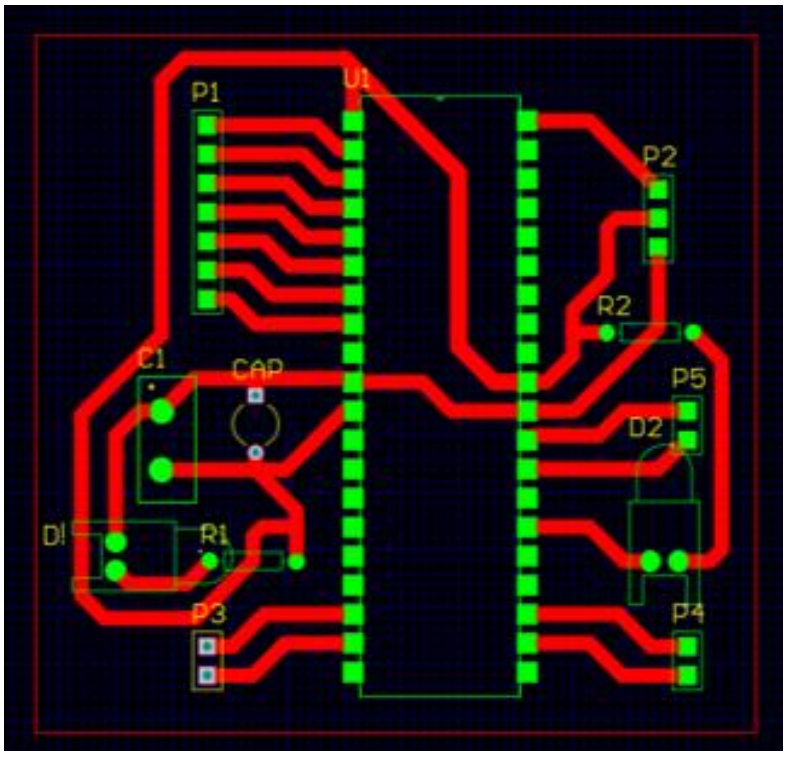


The AVR microcontrollers are based on the advanced RISC architecture. ATmega32 is a low power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. AVR can execute 1 million instructions per second if cycle frequency is 1MHz.

Key Features:

- 32 x 8 general working purpose registers.
- 32K bytes of in system self-programmable flash program memory
- 2K bytes of internal SRAM
- 1024 bytes EEPROM
- Available in 40 pin DIP, 44 lead QTFP, 44-pad QFN/MLF
- 32 programmable I/O lines
- 8 Channel, 10-bit ADC
- Two 8-bit timers/counters with separate pre scalers and compare modes
- One 16-bit timer/counter with separate pre scaler, compare mode and capture mode.
- 4 PWM channels
- In system programming by on-chip boot program
- Programmable watch dog timer with separate on-chip oscillator.
- Programmable serial USART
- Master/slave SPI serial interface


ATMEGA32A-PU
VCC
AVCC
RESET
AREF
XTAL1
XTAL2
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PD0
PD1
PD2
PD3
PD4
PD5
PD6
PD7
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
GND_2
GND
U1
+5
GND
IR INPUT
LINE IN 1
LINE IN 2
LINE IN 3
LINE IN 4
LINE IN 5
LINE IN 6
LINE IN 7
MOTOR 01 IN 1
MOTOR 01 IN 2
OBSTACLE INDICATOR
MOTOR 02 IN 1
MOTOR 02 IN 2
MOTOR 02 PWM
MOTOR 01 PWM
P1
IR SENSOR PANEL
P2
SHARP IR
P3
MOTOR PWM
P4
MOTOR 01
P5
MOTOR 02
C1
Connector
Custom Connector 2 port
D!
R1
Res12k
2K
POWER
D2
R2
OBSTACLE


P1
U1
P2
R2
C1
CAP
P5
D2
D!
R1
P3
P4

Design specifications

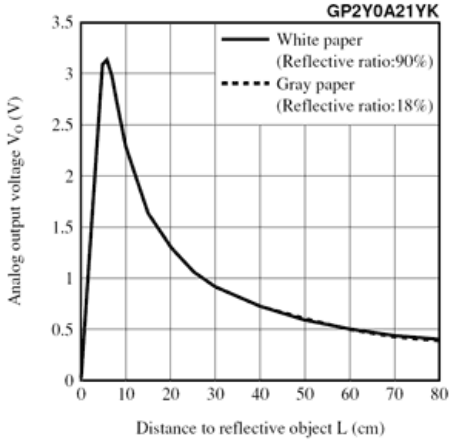
1. Sharp IR sensor

Sharp GP2Y0A21YK0F Analog Distance Sensor 10-80cm

The GP2Y0A21 Sharp distance sensor is a great way to add obstacle avoidance or motion sensing to your robot or any other project. With a detection range of 4″ to 32″ (10 cm to 80 cm) and an analog voltage indicating the distance, this sensor is very easy to use.

Feature summary

- Operating voltage: 4.5 V to 5.5 V
- Average current consumption: 30 mA (note: this sensor draws current in large, short bursts, and the manufacturer recommends putting a 10 µF capacitor or larger across power and ground close to the sensor to stabilize the power supply line)
- Distance measuring range: 10 cm to 80 cm (4″ to 32″)
- Output type: analog voltage
- Output voltage differential over distance range: 1.9 V (typical)
- Update period: 38 ± 10 ms
- Size: 44.5 mm × 18.9 mm × 13.5 mm (1.75″ × 0.75″ × 0.53″)
- Weight: 3.5 g (0.12 oz)



2. Chassis

Acrylic boards were used to build up the chassis. Six PCBs were mounted on two layers of Acrylic boards using screws and nuts. The structure of acrylic is cut and filleted and drilled in order to mount screws.

3. Motors

12V DC gear motors of 150 rpm were used in this robot.

4. Caster wheel

A caster wheel which was able to spin inside the metal casing is mounted at the bottom edge of the chassis. Lubricant is used to decrease the friction inside the wheel.

5. Holders

Acrylic boards were heated and bent in order to make curved parts to hold the battery and the caster wheel to the chassis. Battery was firmly fixed inside the two layers of chassis using this holder. A holder lifted to a relevant height was used to attach the caster wheel with the chassis.

Observations and results

- ALTIUM was used to design schematic and PCB layouts. Even though there were many software which provides the exact same facilities, using ALTIUM and finding footprints were more convenient. It was also noticed that many tutorials were present online regarding ALTIUM and footprint availabilities. Finding and adding libraries, downloading footprints, designing PCB layouts manually and handling with Gerber files were some skills which were developed associated with ALTIUM.

- Then we tried to make PCBs one by one. At first, we tried to make a PCB using screen printing method. But it doesn't meet the standards. Sometimes there was not all tracks have printed on the copper board. Then we used ironing method to construct PCBs. It was very easy and cheaper method. At the beginning we didn't consider much about the thickness of the tracks. With time, when the motor drew certain current from the tracks, it is noticed that the tracks were damaged at some places. The PCBs were made again with thicker tracks where it is required.

- The next issue we encountered was, the temperature sensitive nature of some ICs. Some ICs got damaged quickly even when a proper soldering technique was used. We practiced handling those with extreme care.

- The power circuits required some inductors which were not available in the market for through hole mounting. We used surface mounting inductors in order to mount through holes. We were managed to mount them on the copper plates using surface mounting techniques.

- We connected 12V LIPPO battery for give power to motors directly. But after some time, Robot did not go according to code. Then we found the battery power has drained under a certain amount. So, as a solution we got power through another buck convertor so finally we used 2 buck convertors for get 12V and 5V.

- When we fix PCBs to Robot structure, first we used gum tapes to get space between structure and PCB. But after few days circuit had corroded. As a solution we fixed the PCBs to the chassis using screws and nuts.

- As the motor controller IC dissipates a considerable amount of heat, we were supposed to connect a heat sink for it. On the other hand, we notice that the circuit was short connected through the thermal paste we applied between the IC and heat sink. The paste was removed as a solution.

- After making the robot we code it and tried to run it. But it could not get the turns correctly. Because first we use 360-degree front wheel for turning purpose. Then we used a caster wheel and the robot took turns correctly.

Outcomes achieved

1. Designing schematic diagrams and PCB layouts
2. Coding microcontroller using C (programming language)
3. Printing PCBs using iron and etch method
4. Mechanical skills related to making the chassis
5. Handing data sheets of specific components
6. Knowledge of leading electronic industries which manufactures certain products
7. Simulation of electronic circuits
8. Knowledge about Basic electrical limitations

Results

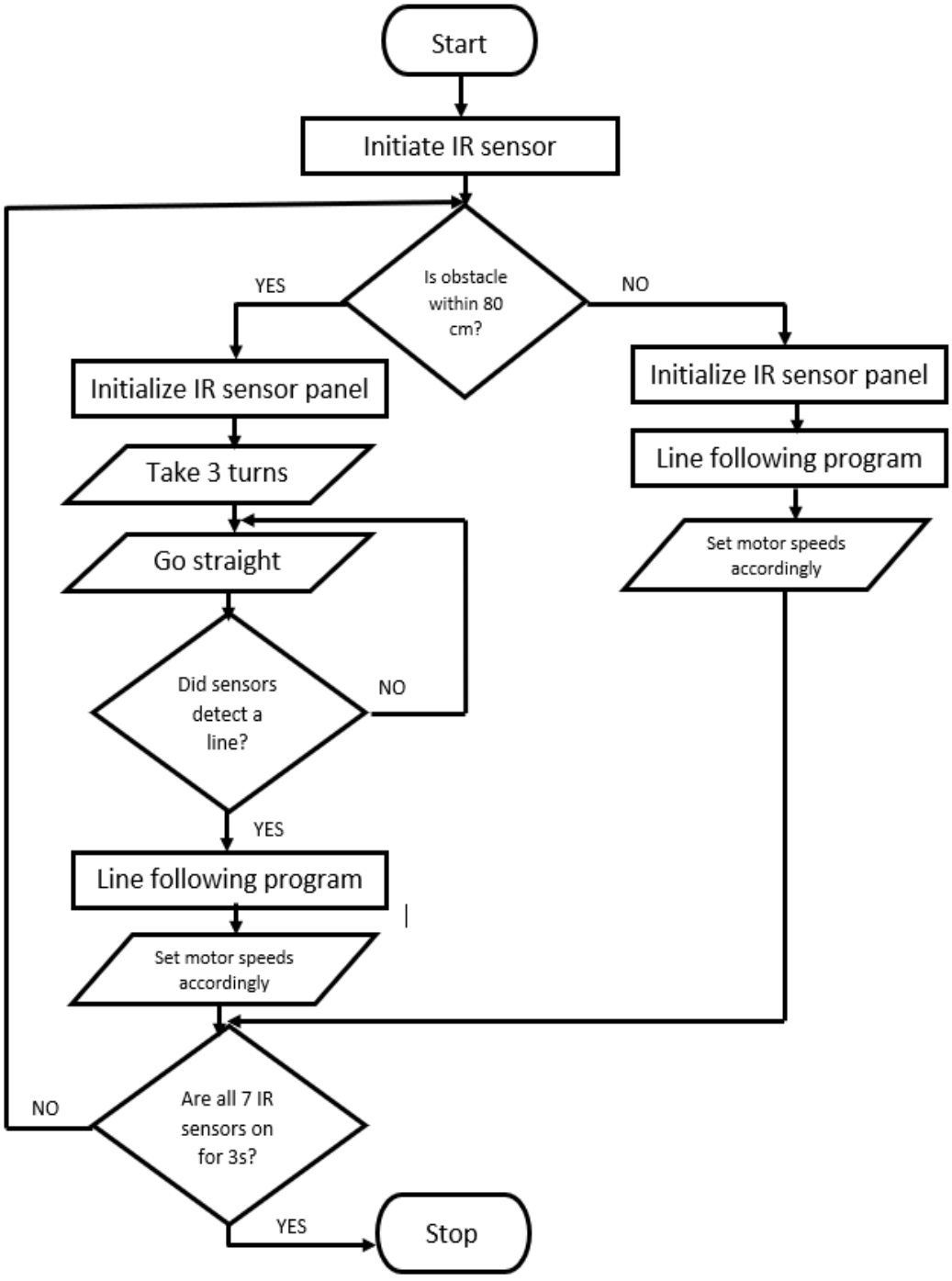
Fortunately, the demonstration of the robot was successful. The robot started well at the starting point and followed the line. The obstacle avoiding part was extremely successful. The curvy and 90-degree turns were performed well by the robot. The only minor issue was the robot had not stopped at the end square. But for the second try it performed the end task also. The reason might be the curvy bend at the end point. The curvy line ended just before square, so that the square was not aligned at the required manner to the robot as the robot did not reached the straight orientation. Also, it is notable that the end program is supposed to run after a certain second as it detects all 7 LEDs lights up. This algorithm is developed because the robot detected all 7 LED light ups even when it takes 90 degree turns.

Finally, I would like to mention that the robot represented our group finished the task in a single try and which was the best performance out of the other demonstrations.

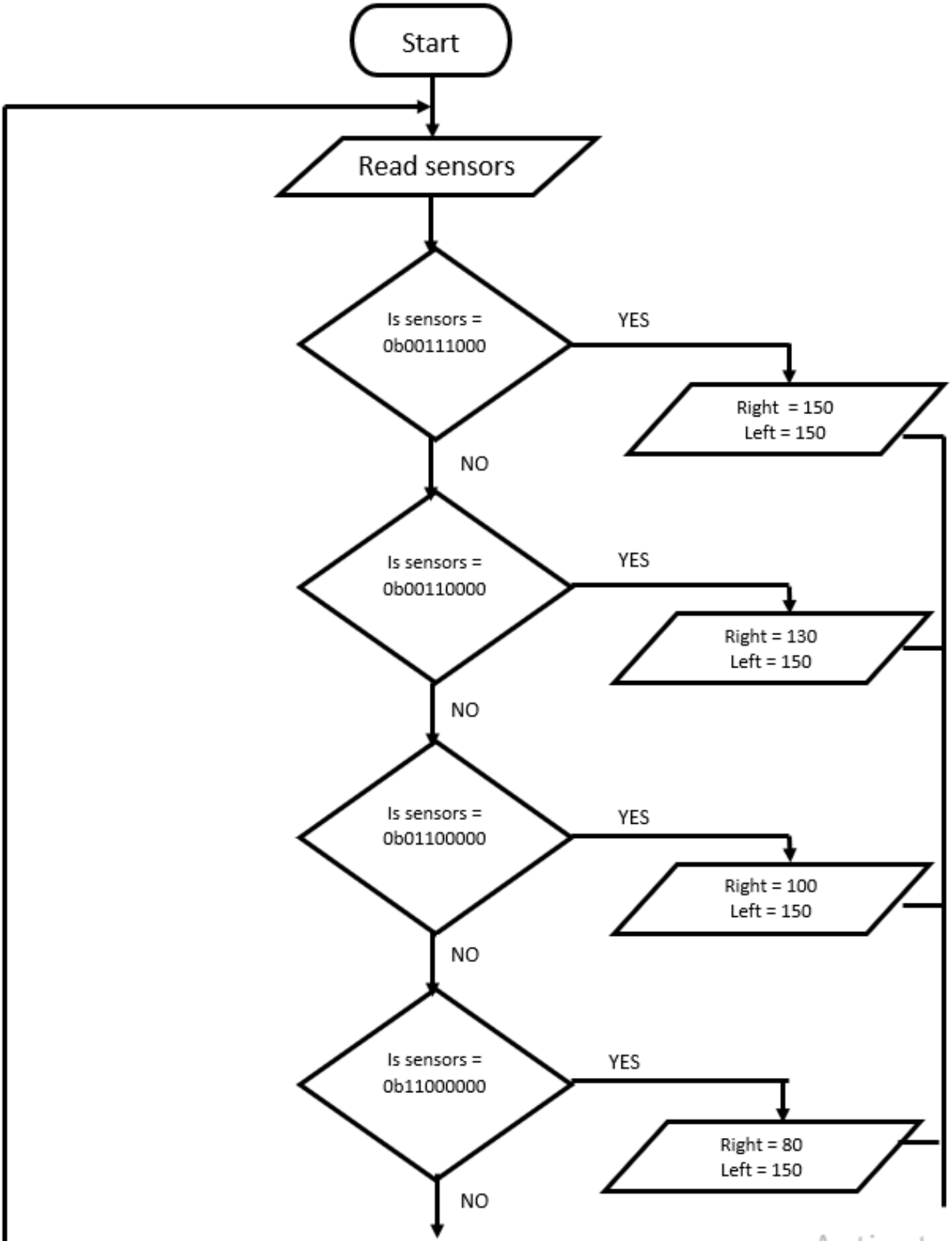
Programming algorithm

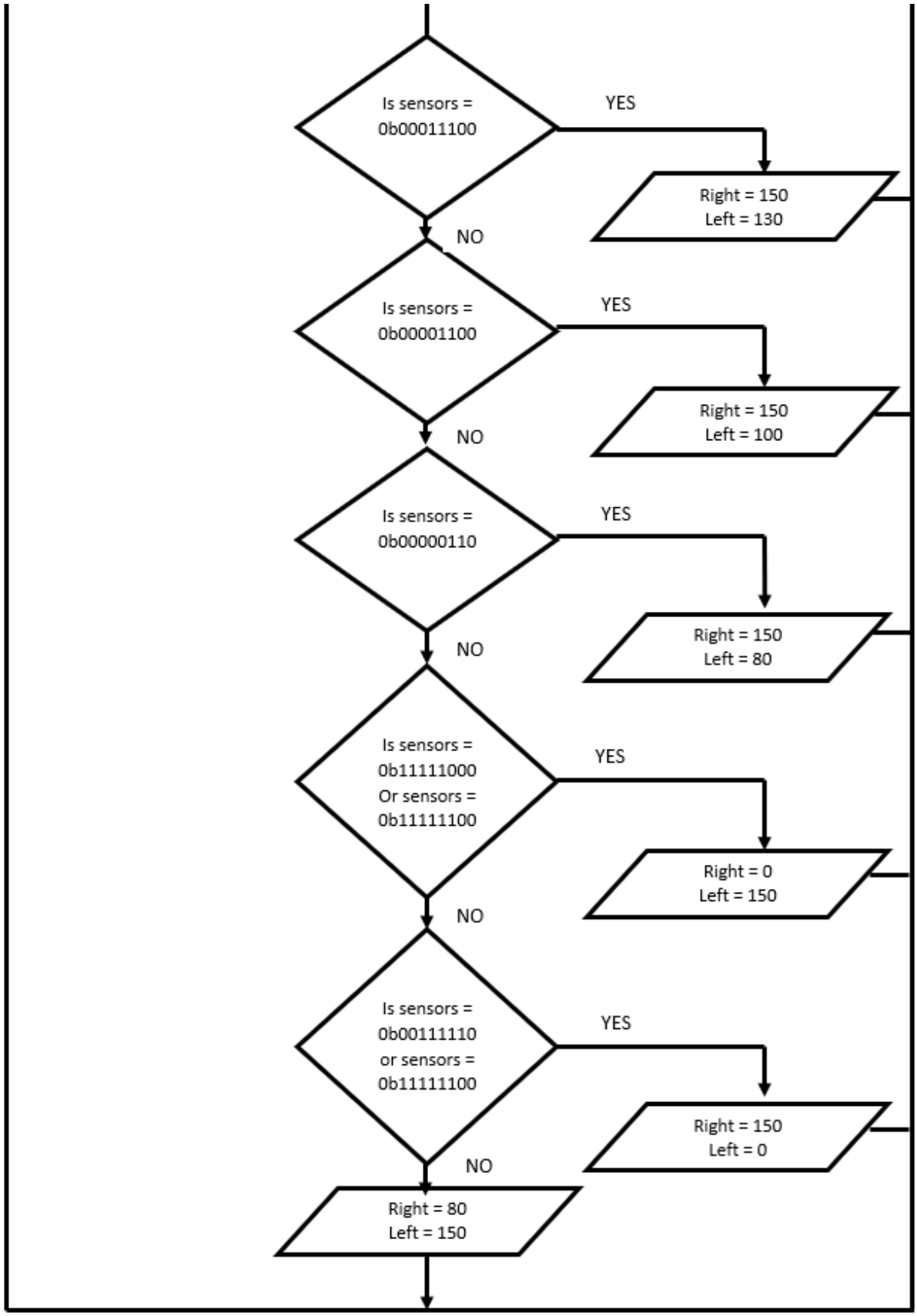
The coding for this robot can be easily understood by flow charts. Main program portraits the overall loop and the line following program which comes across the main program performs the respective task.

Main program



Line following program




Is sensors = 0b00011100
YES
Right = 150
Left = 130
NO
Is sensors = 0b00001100
YES
Right = 150
Left = 100
NO
Is sensors = 0b00000110
YES
Right = 150
Left = 80
NO
Is sensors = 0b11111000 Or sensors = 0b11111100
YES
Right = 0
Left = 150
NO
Is sensors = 0b00111110 or sensors = 0b11111100
YES
Right = 150
Left = 0
NO
Right = 80
Left = 150

Code (Language - C)

```
/*
* line_following_robot_with_phase_correct_PWM.c
*
* Created: 1/22/2020 12:29:32 AM
* Author : pahan
*/
#define F_CPU 1000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#define leftMotorPWMPin OCR1B
#define rightMotorPWMPin OCR1A
#define In1 PC0
#define In2 PC1
#define In3 PC6
#define In4 PC7
#define LEDPIN PINC4
#define IRSensor PINA0
#define MAXSPEED 180
#define ROTATESPEED 150
/* Initializes the OC1A and OC1B pins for 8bit
phase correct PWM.
* with noninverting
mode. OC1A/OC1B is cleared on compare match
* when upcounting
and is set on compare match when downcounting.
*/
void initialize()
{
//Configure PWM pins OC1B and OC1A to output mode
DDRD |= (1<<PIND4) | (1<<PIND5) | (1<<LEDPIN);
//Configure motor direction control pins to output mode
DDRC |= (1<<In1) | (1<<In2) | (1<<In3) | (1<<In4) | (1<<LEDPIN);
ADMUX |= (1<<REFS0); // reference voltage
ADCSRA |= (1<<ADPS1) | (1<<ADPS0); // prescaler 8
ADCSRA |= (1<<ADEN); // start ADC
ADMUX |= (1 << ADLAR); // Left adjust ADC result to allow easy 8 bit
reading
//Clear OC1A/OC1B on compare match when upcounting
and set OC1A/OC1B
//on compare match when downcounting (sets PWM to noninverting
mode)
TCCR1A |= (1<<COM1A1) | (1<<COM1B1);
//Selects prescalar value 8
TCCR1B |= (1<<CS11|(1<<CS10));
//Phase Correct PWM mode is selected
TCCR1A |= (1<<WGM11);

TCCR1B |= (1<<WGM13);
//Sets TOP value to be 250. Frequency is 500Hz
ICR1 = 250;
}
```

```
/* Sets the speed and direction of the two motors. The value of each
* of the arguments can range from 250
to +250. A negative sign
* is used when the direction of the motor is to be reversed.
*/
void set_motors(int leftMotorSpeed, int rightMotorSpeed)
{
if(leftMotorSpeed >= 0)
{
leftMotorPWMPin = leftMotorSpeed;
PORTC |= (1<<In1);
PORTC &= ~(1<<In2);
}
else
{
leftMotorPWMPin = MAXSPEED + leftMotorSpeed;
PORTC |= (1<<In2);
PORTC &= ~(1<<In1);
}
if(rightMotorSpeed >= 0)
{
rightMotorPWMPin = rightMotorSpeed;
PORTC |= (1<<In4);
PORTC &= ~(1<<In3);
}
else
{
rightMotorPWMPin = MAXSPEED + rightMotorSpeed;
PORTC |= (1<<In3);
PORTC &= ~(1<<In4);
}
}
int main(void)
{
DDRB = 0X00;
initialize();
char sensors = 0;
uint16_t distance;
int obstacles = 0;
int end_count=0;

_delay_ms(300);
set_motors(80,75);
_delay_ms(50);
set_motors(150,150);
_delay_ms(250);
set_motors(0,0);
_delay_ms(50);
while (1)
{
sensors = PINB;
if(!(sensors==0b11111110))
{
end_count=0;
}
ADCSRA |= (1<<ADSC);
```

```
loop_until_bit_is_clear(ADCSRA,ADSC);
distance = ADCH;
if ((distance > 56)&&(distance < 78) ) //2nd obstacle
{
set_motors(0,0);
PORTC |= (1<<LEDPIN);
_delay_ms(1000);
PORTC &= !(1<<LEDPIN);
_delay_ms(1000);
set_motors(180,50);
_delay_ms(400);
set_motors(0,0);
_delay_ms(100);
set_motors(170,170);
_delay_ms(600);
set_motors(0,0);
_delay_ms(100);
set_motors(5,ROTATESPEED); //2nd
_delay_ms(180);
set_motors(160,160);
_delay_ms(450);
set_motors(0,0);
_delay_ms(100);
set_motors(0,100); //3rd
_delay_ms(180);
obstacles=2;
set_motors(0,0);
_delay_ms(100);
while(1)
{
set_motors(102,100);
_delay_ms(1);

set_motors(0,0);
_delay_ms(1);
sensors = PINB;
if (sensors & (1 << 1))
{
break;
}
}
}
if ((distance > 78) && (distance < 120)) // 1st obstacle
{
set_motors(0,0);
PORTC |= (1<<LEDPIN);
_delay_ms(1000);
PORTC &= !(1<<LEDPIN);
_delay_ms(1000);
set_motors(200,10);
_delay_ms(150);
set_motors(0,0);
_delay_ms(100);
set_motors(170,170);
_delay_ms(600);
set_motors(0,0);
```

```
_delay_ms(100);
set_motors(5,
ROTATESPEED); //2nd
_delay_ms(300);
set_motors(160,160);
_delay_ms(300);
set_motors(0,0);
_delay_ms(100);
set_motors(0,120); //3rd
_delay_ms(100);
set_motors(0,0);
_delay_ms(100);
while(1)
{
set_motors(100,100);
_delay_ms(1);
set_motors(0,0);
_delay_ms(1);
sensors = PINB;
if (sensors & (1 << 1))
{
break;
}
}
}
else
{
/* line following module */
//straight
if((sensors == 0b00111000))
{
set_motors(0,0);
set_motors((int)0.2*MAXSPEED,(int)0.2*MAXSPEED);
set_motors((int)0.5*MAXSPEED,(int)0.5*MAXSPEED);
set_motors((int)0.7*MAXSPEED,(int)0.7*MAXSPEED);
set_motors(MAXSPEED,MAXSPEED);
}
//right drift
else if((sensors == 0b00110000))
{
set_motors(0,0);
set_motors((int)(0.1*MAXSPEED),(int)0.3*MAXSPEED);
set_motors((int)(0.5*MAXSPEED),MAXSPEED);
}
else if((sensors == 0b01110000))
{
set_motors(0,0);
set_motors((int)(0.1*MAXSPEED),(int)0.35*MAXSPEED);
set_motors((int)(0.3*MAXSPEED),(int)0.7*MAXSPEED);
set_motors((int)(0.5*MAXSPEED),MAXSPEED);
}
else if((sensors == 0b01100000))
{
set_motors(0,0);
set_motors((int)(0.2*MAXSPEED),(int)0.4*MAXSPEED);
```

```
set_motors((int)(0.4*MAXSPEED),MAXSPEED);
}
else if((sensors == 0b11100000))
{
set_motors(0,0);
set_motors((int)(0.1*MAXSPEED),(int)0.3*MAXSPEED);
set_motors((int)(0.2*MAXSPEED),(int)0.6*MAXSPEED);
set_motors((int)(0.3*MAXSPEED),MAXSPEED);
}
else if((sensors == 0b11000000))

{
set_motors(0,0);
set_motors((int)(0.1*MAXSPEED),(int)0.3*MAXSPEED);
set_motors((int)(0.1*MAXSPEED),(int)0.6*MAXSPEED);
set_motors((int)(0.2*MAXSPEED),MAXSPEED);
}
else if (sensors == 0b10000000)
{
set_motors(0,0);
set_motors((int)(0.02*MAXSPEED),(int)0.3*MAXSPEED);
set_motors((int)(0.04*MAXSPEED),(int)0.6*MAXSPEED);
set_motors((int)(0.05*MAXSPEED),MAXSPEED);
}
// left drift
else if((sensors == 0b00011000))
{
set_motors(0,0);
set_motors((int)(0.2*MAXSPEED),(int)0.1*MAXSPEED);
set_motors((int)(0.5*MAXSPEED),(int)0.3*MAXSPEED);
set_motors((int)(0.7*MAXSPEED),(int)0.4*MAXSPEED);
set_motors((int)0.9*MAXSPEED,((int)0.5*MAXSPEED));
}
else if((sensors == 0b00111000))
{
set_motors(0,0);
set_motors((int)(0.5*MAXSPEED),(int)0.2*MAXSPEED);
set_motors((int)(0.7*MAXSPEED),(int)0.3*MAXSPEED);
set_motors(MAXSPEED,((int)0.45*MAXSPEED));
}
else if((sensors == 0b00001100))
{
set_motors(0,0);
set_motors((int)(0.5*MAXSPEED),(int)0.2*MAXSPEED);
set_motors((int)(0.7*MAXSPEED),(int)0.25*MAXSPEED);
set_motors(MAXSPEED,((int)0.4*MAXSPEED));
}
else if((sensors == 0b00001110))
{
set_motors(0,0);
set_motors((int)(0.4*MAXSPEED),(int)0.15*MAXSPEED);
set_motors((int)(0.7*MAXSPEED),(int)0.25*MAXSPEED);
set_motors(MAXSPEED,(int)0.3*(MAXSPEED));
}
else if((sensors == 0b00000110))
{
```

```
set_motors(0,0);
set_motors((int)(0.5*MAXSPEED),(int)0.05*MAXSPEED);

set_motors((int)(0.7*MAXSPEED),(int)0.15*MAXSPEED);
set_motors(MAXSPEED,((int)0.2*(MAXSPEED)));
}
else if (sensors == 0b00000010)
{
set_motors(0,0);
set_motors((int)(0.3*MAXSPEED),(int)0.02*MAXSPEED);
set_motors((int)(0.7*MAXSPEED),(int)0.04*MAXSPEED);
set_motors(MAXSPEED,(int)(0.05*MAXSPEED));
}
//right angle turns
else if (sensors == 0b11111000 || sensors == 0b11111100 || sensors
== 0b11110000)
{
set_motors(0,ROTATESPEED*0.1);
set_motors(0,ROTATESPEED*0.3);
set_motors(0,ROTATESPEED*0.65);
_delay_ms(300);
set_motors(0,0);
_delay_ms(100);
}
else if (sensors == 0b00111110 || sensors == 0b01111110 || sensors
== 0b00011110)
{
set_motors(ROTATESPEED*0.1,0);
set_motors(ROTATESPEED*0.2,0);
set_motors(ROTATESPEED*0.6,0);
_delay_ms(300);
set_motors(0,0);
_delay_ms(100);
}
else if (sensors == 0b11111110 )
{
set_motors((int)(MAXSPEED*0.3),(int)(MAXSPEED*0.23));
_delay_ms(50);
end_count++;
if (end_count>300)
{
while(1)
{
set_motors(0,0);
}
}
}
else
{
set_motors((0),(0));
set_motors((int)(MAXSPEED*0.1),(int)(MAXSPEED*0.9));
set_motors((int)(MAXSPEED*0.3),(int)(MAXSPEED*0.28));
}
}
```