Faculty of Information Technology

IN 1900 – ICT Project

Office Trashbot

Group No: 18

204071C - Hathnagoda H.W.E.M.J.N (Leader)

204084T - Jayasekara J.P.I.A

204118E - Madhushika B.A.E

204172L - Rambukkamage R.D.S.T

204184B - Samararathna B.S

Supervisor's Name: Mr. B.H. Sudantha
Dean / Senior Lecturer
Faculty of Information Technology
University of Moratuwa

Ms. K.M.S.J. Kumarasinghe
Lecturer
Faculty of Information Technology
University of Moratuwa

Date of Submission: 08.06.2022

Signature of Supervisor:

01. ……...

02. ……...

Table of Contents

1.0 Introduction 1

2.0 Literature Survey 2

3.0 Aim and Objectives 6

3.1 Aim 6

3.2 Objectives 6

4.0 System Description 7

4.1 Block Diagram 7

4.2 Functionality 7

4.3 Appearance (3D View) 8

4.4 Circuit Diagram 10

4.5 PCB Design 10

5.0 Testing and Implementation 12

6.0 Further Work / Improvements 16

7.0 References 17

Appendix A 18

Cost Estimation 18

Appendix B 19

Individual Contribution 19

1.0 Introduction

Trash management is an integral part of industrial section. Just by doing simple tasks in companies (IT companies), we generate a substantial amount of waste. The trash that humans create isn't the most pleasant part of life, but regardless, it needs to be handled professionally. While having pounds of garbage as a constant co-worker doesn't sound like the most glamorous working environment.

Without proper monitoring, waste can stagnate in our trash cans. A possible solution is to take out the trash frequently, but that may also be wasteful if the trash can is mostly empty. Likewise, different amounts of waste are generated day by day, so predicting exactly when to take out the trash becomes more difficult. This becomes even more difficult considering the number of trash cans in a workplace.

2.0 Literature Survey

Here we have studied about what are the types of smart trash bins, automations available in market and what are the flaws in them. Below shows the products and ideas which are available.

i. iTouchless 13-Gallon Sensor Trash Can [12]

- This stainless steel is instantly eye-catching with its quick-opening butterfly lid and comes in sizes 13 and 21 gallons. But it can't move around automatically to collect garbage. It opens the lid only by sensing the hand or finger six inches away using the IR sensor. And the cost of this trash can is high so it's difficult to afford this price.

FIGURE 2.1: ITOUCHLESS 13-GALLON SENSOR TRASH CAN

- Similar Products: hOmeLabs 13-Gallon Automatic Trash Can
  Glad Stainless Steel Motion Sensor Can
  Ninestars DZT-20-1R Automatic Touchless Trash

- Comparison:
  - ✓ Lid opening feature is same.
  - ✓ This trashcan can't move but our trashbot can move.
  - ✓ The cost of this trashcan is high.

ii. TrashBot™ Slim [16]

- Clean Robotics has built an autonomous system that uses robotics, computer vision and artificial intelligence to detect and separate landfill from recyclables. This product identifies the types of waste. But it can't move automatically from one place to another. So people have to go to near to the bin and put the trash. Cost of this trashbot is difficult to afford by Mid-level IT companies.

FIGURE 2.2: TRASHBOTTM SLIM

- Comparison:
  - ✓ This trashbot is using advanced technologies like robotics, computer vision but we are not using such advanced technologies.
  - ✓ This one can't move but our trashbot can move.
  - ✓ The cost of this trashcan is high.

iii. Garbage collector robot [8]

- This is a semi-autonomous garbage collector robot which can do multiple functions. This was designed to collect outdoor trash and this is semiautonomous robot. Cost is low but this can't function fully autonomously. They use autonomous line following method and also manual method by using Arduino and Bluetooth module. This can be controlled by a software. This robot also has robotic arm to pick the garbage and dispense it in main basket attached to the robot and camera to administrate the robot.

iv. Autonomous Refuse Robot [14]

- Created by Canada's pioneering robotics and artificial intelligence research company. The new AI-enhanced robotic system introduces a device that can automatically navigate to the curbside and wait for the pickup truck to arrive at a pre-scheduled time. The scheduling is done with the APP and the status can be checked in real time. Company uses SLAM Technology combined with in-depth learning to be a pioneer for the new generation of robots. Automated trash refuse robot is an application for company's Versatile Self Localizing Autonomous Platforms (VSLAP). This robot uses software called the Quantum Slam Operating System (Q-OS).

FIGURE 2.3: AUTONOMOUS REFUSE ROBOT

- Comparison:
  - ✓ This autonomous refuse robot is used outdoors and our trashbot is used for office purpose.
  - ✓ This robot uses advanced AI technologies but trashbot navigate using programmed wheel encoder and dc motor.
  - ✓ This robot can be controlled by an app but we aren't using an mobile app.
  - ✓ Both robots working with pre-scheduled time.

v. Robotic Trash Can [2]

- This trash-can has programmed robotic wheels that are compatible with any municipal-issued trash receptacle. It can travel from one docking station in a person's accommodation to a second docking station on the border. This one is also synced to an app that can schedule your garbage collection time and date.

FIGURE 2.4: ROBOTIC TRASH CAN

- Comparison:
  - ✓ As previous one this trash can is also used outdoors and our trashbot is used for office purpose.
  - ✓ This robot uses programmed wheels and our trashbot navigate using programmed wheel encoder and dc motor.
  - ✓ This robot can be controlled by an app but we aren't using an mobile app.
  - ✓ Both robots working with pre-scheduled time.

3.0 Aim and Objectives

3.1 Aim

- Design and develop a robot to improve the trash management process in IT companies.

3.2 Objectives

- To collect trash in workplaces by going predefined route.
- To collect polythene and papers separately.
- To collect trash inside office rooms by sending a message to the person inside the room.
- To measure the trash level.
- To notify janitor when trash level is high. (90%)

4.0 System Description

4.1 Block Diagram

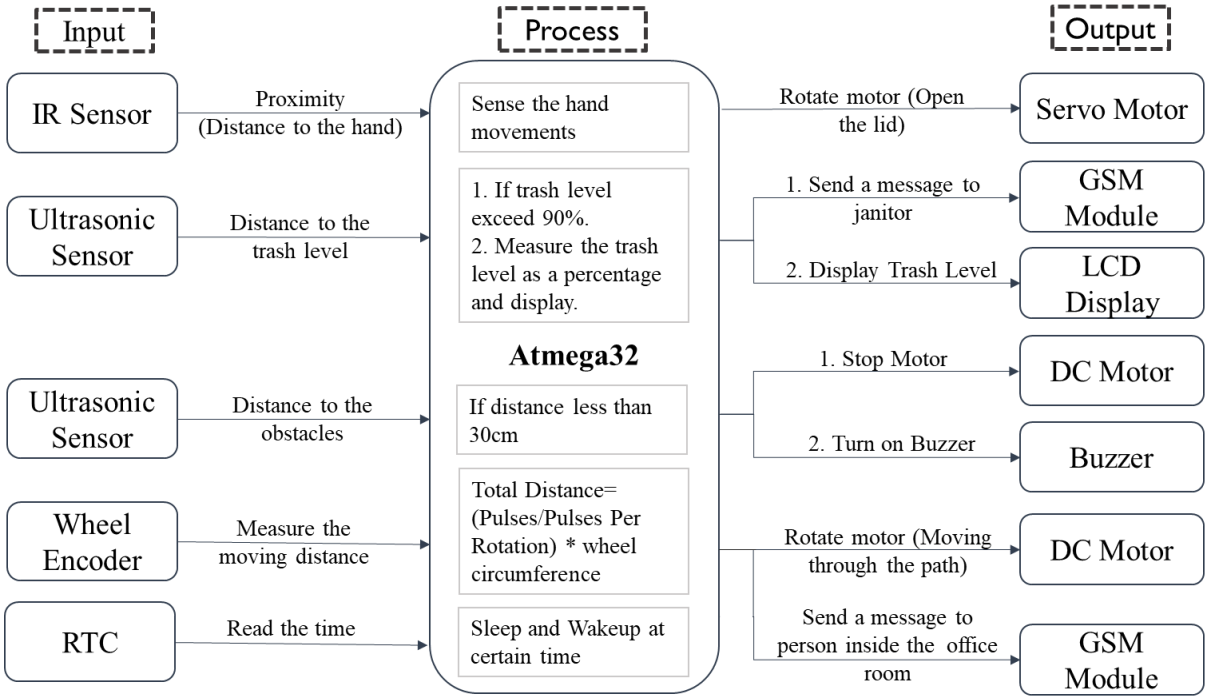


FIGURE 4.1.1: INPUT -OUTPUT DIAGRAM OF THE PROPOSED SYSTEM

4.2 Functionality

- We will integrate a microcontroller, sensors, as well as motors with wheels to create the robot.
- There are two spaces and lids for polyethene and paper separately. Lids open automatically using PIR sensor and servo motor when human hand is detected.
- Using ultrasonic sensors placed inside the robot, we can detect the level of waste. Trash level displays on LCD. Polyethene and paper level displays separately. If any type of trash level passes a certain threshold, microcontroller send a signal to send a message to janitor's mobile via GSM module.
- The robot move itself to the doors and through the workplace using a predefined route. Wheel encoders and DC motors are used to navigate the robot. Robot moves according to a time schedule.
- When robot encounters an unknown obstacle like human, it will stop and tun on a small buzzer.
- When robot comes near to a door, person inside the room will be notified via a message using GSM module.

4.3 Appearance (3D View)

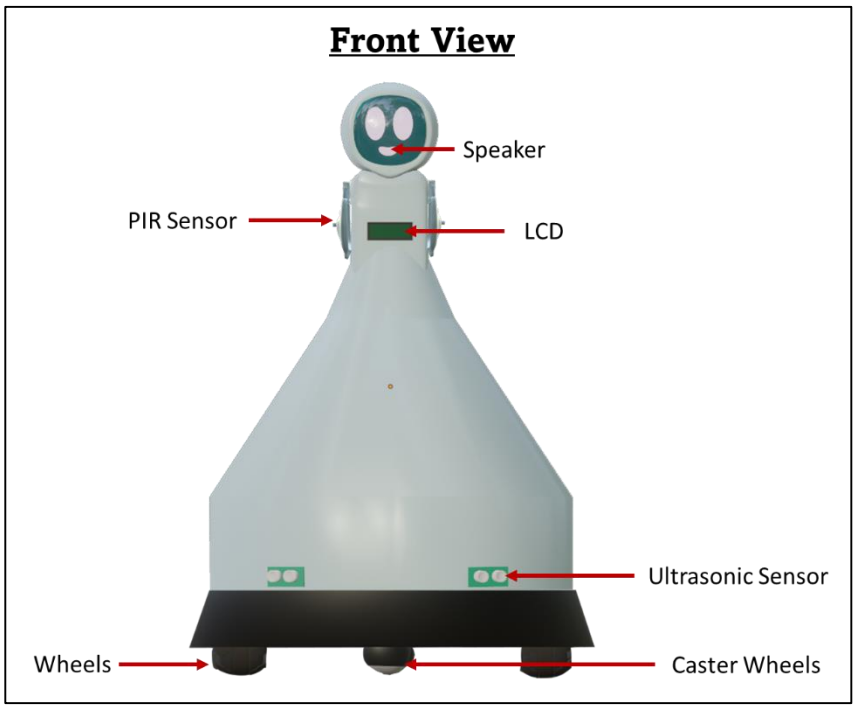


FIGURE 4.3.1: FRONT VIEW

FIGURE 4.3.2: SIDE VIEW

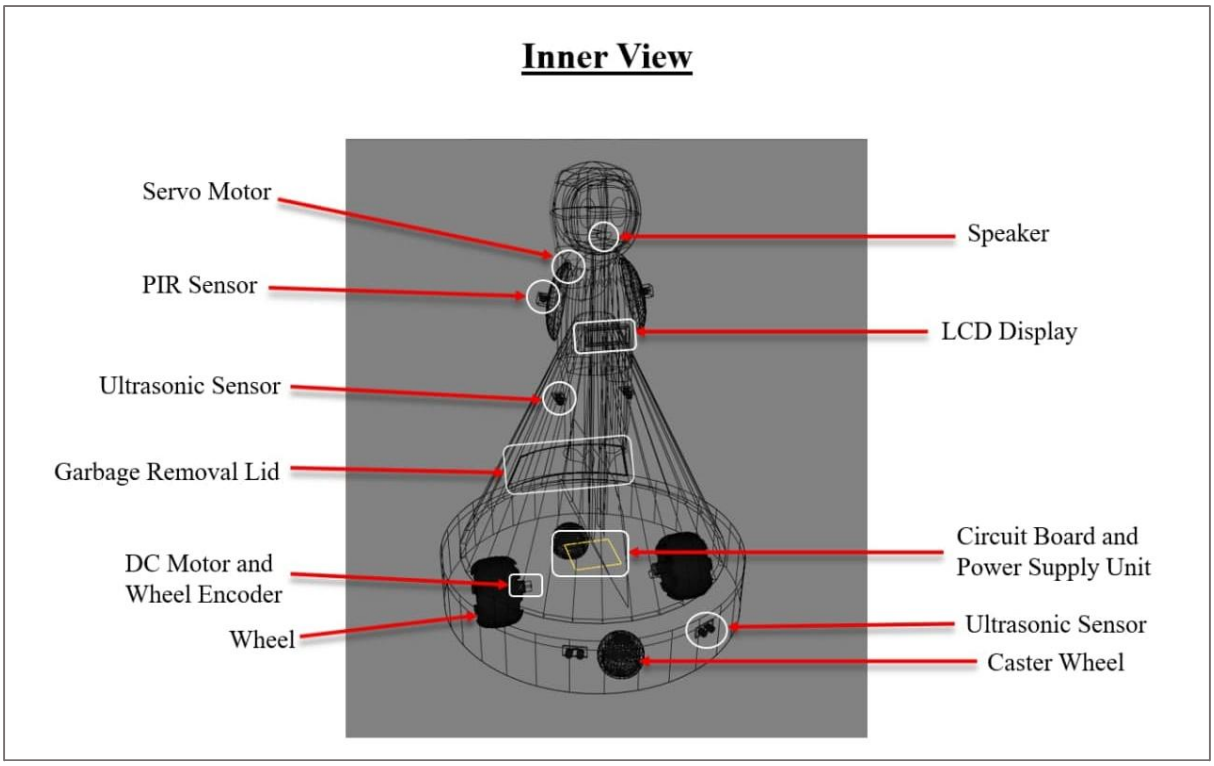


FIGURE 4.3.3: INNER VIEW

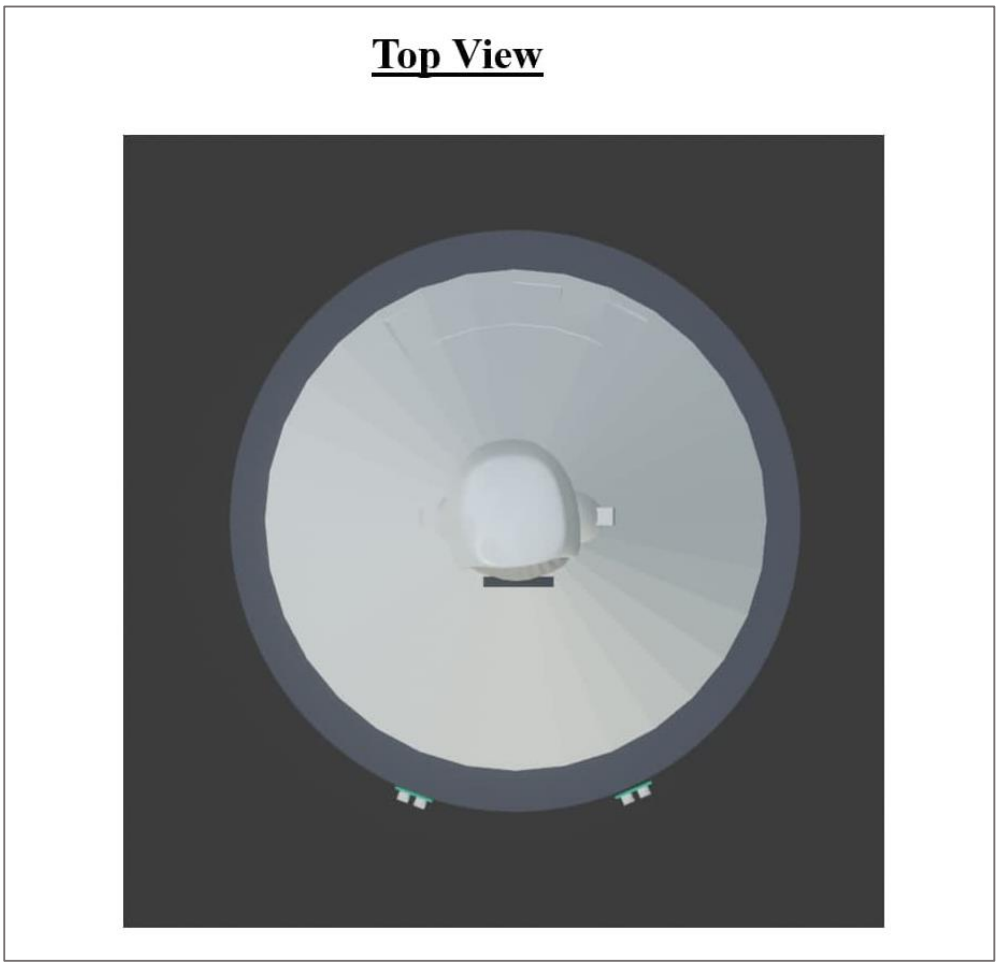


FIGURE 4.3.4: TOP VIEW

4.4 Circuit Diagram

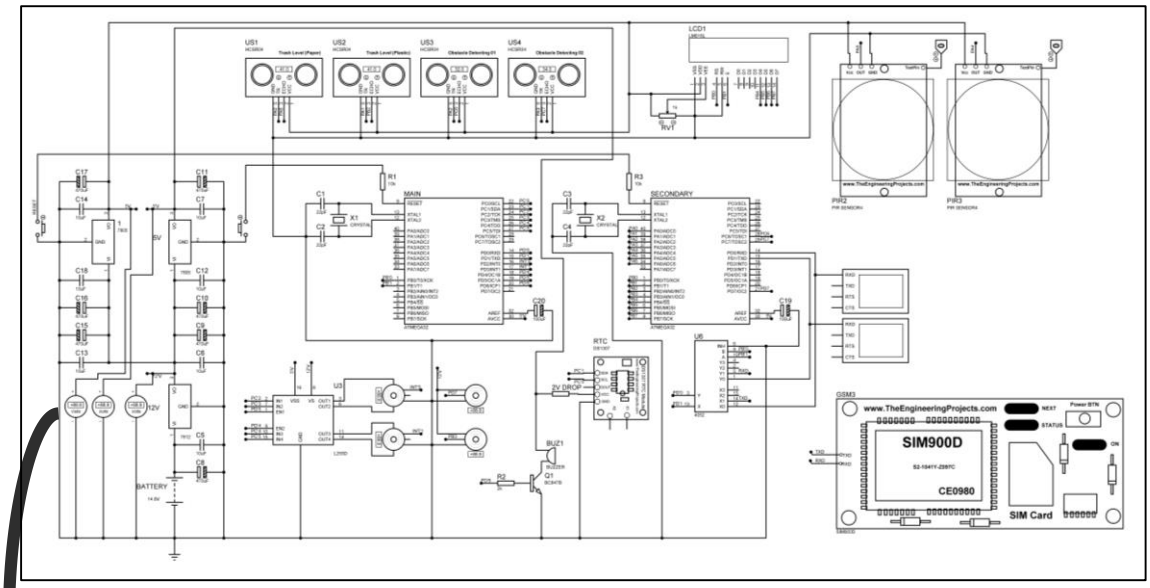


FIGURE 4.4.1: CIRCUIT DIAGRAM

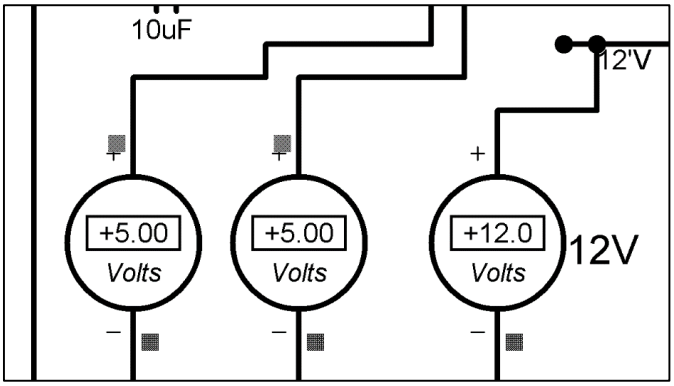


FIGURE 4.4.2: VOLTAGES IN SIMULATION MODE

4.5 PCB Design

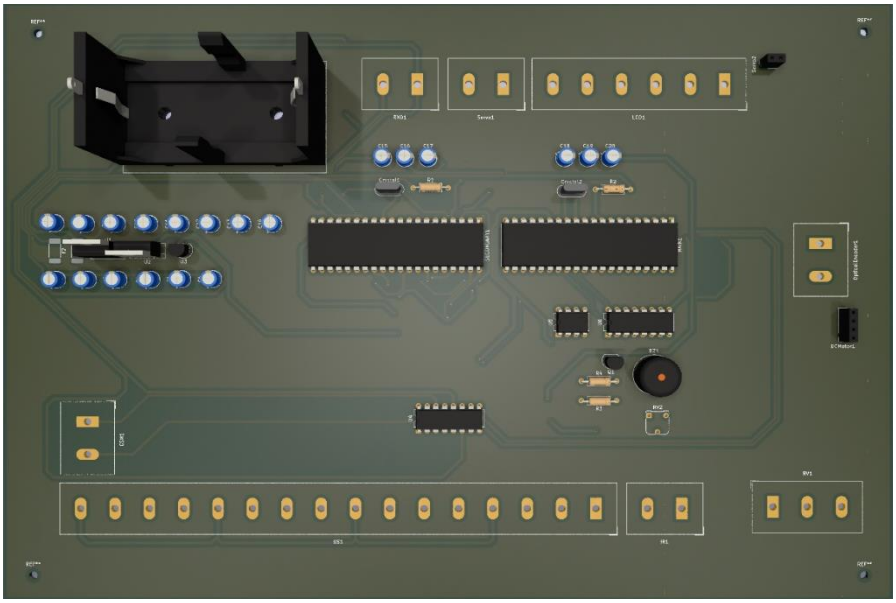


FIGURE 4.5.1: FRONT VIEW

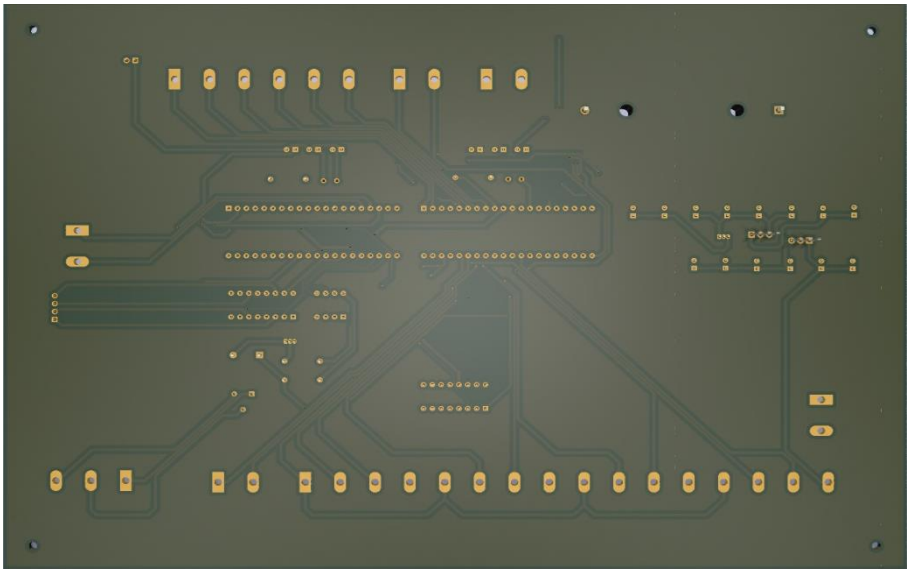


FIGURE 4.5.2: BACK VIEW

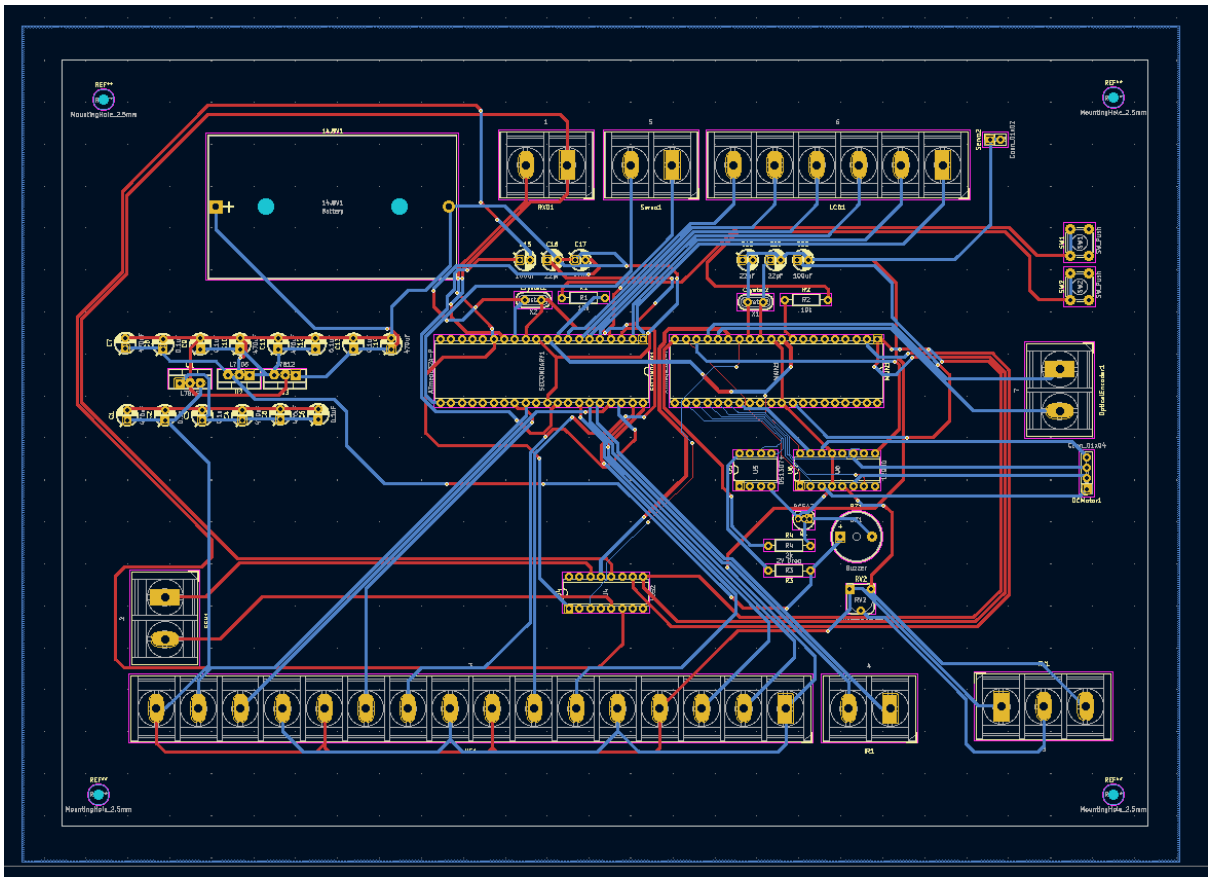


FIGURE 4.5.3: COMPONENT LAYER

5.0 Testing and Implementation

- First, we studied basic information about the components individually as we distributed before. Then we discussed about features and selected Atmega32 as the microcontroller.
- Then we continued studying about components, like pin configurations and basic coding parts. We tried to simulate them using microchip studio and proteus software.
- Then we started 3D modeling part and animation part.
- After completing individual parts, we started to integrate the code and design the full circuit on proteus.
- Meanwhile members started design the PCB using KiCAD software.

<u>Problems we got while testing:</u>

- We had to use 2 Atmega microcontrollers because timers were not enough for the components. Both ultrasonic sensors and dc motors use timer1 register. And encoders also use timers
- Previously decided to give path by using A* algorithm. But this robot needs to come back its initial position. By using A* algorithm it difficult to give path. Because it finds shortest path to the destination. Then we found this path can manually give by using motor encoder.
- Because of connecting two microcontrollers GSM and ultrasonic sensor can’t work at the same time. Hence, we can not send message to refuse collector when trash level over 90%.
- AT commands are working on GSM module while simulating but sending message function is not working properly.

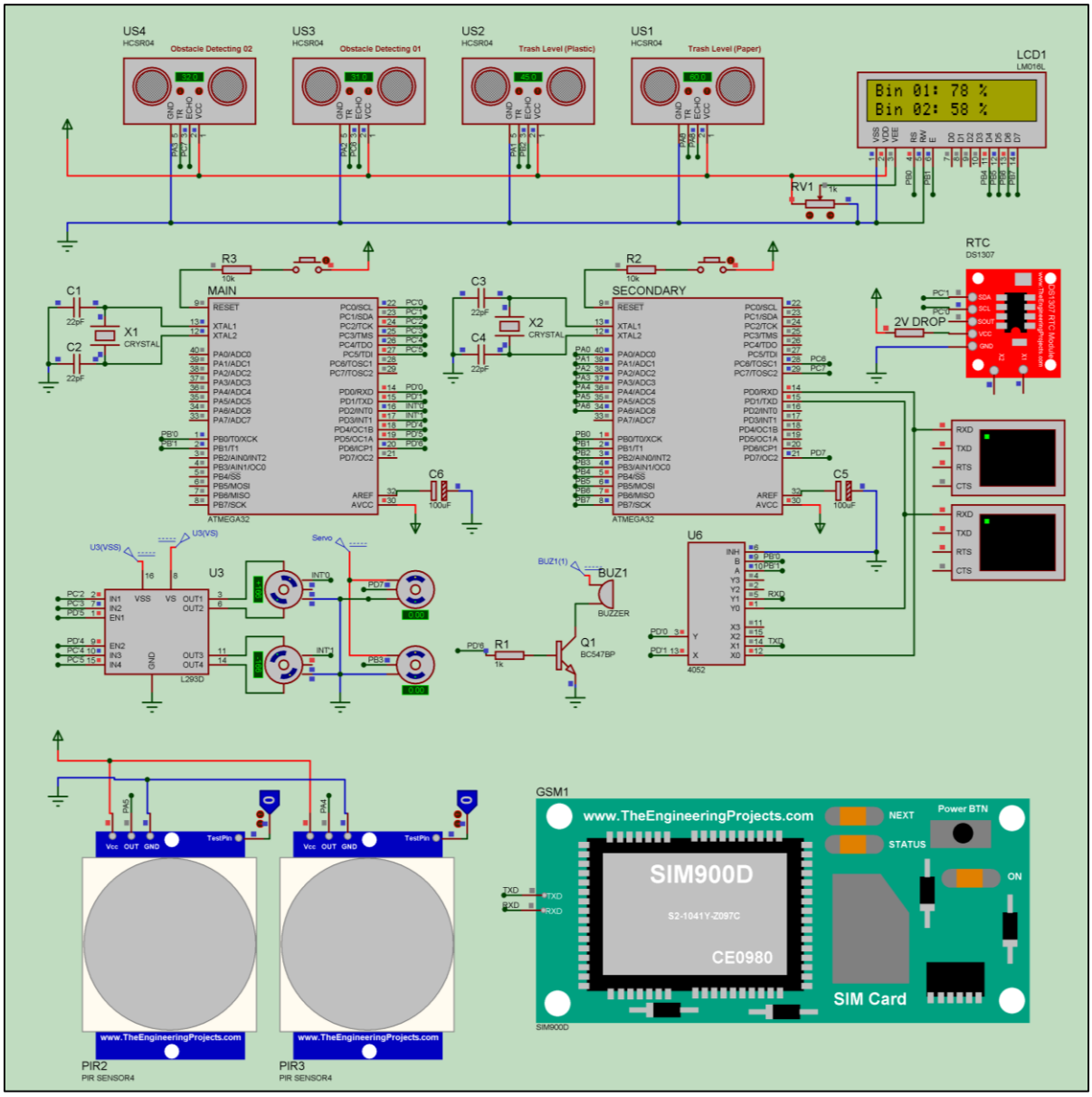


FIGURE 5.0.1: SIMULATION 01

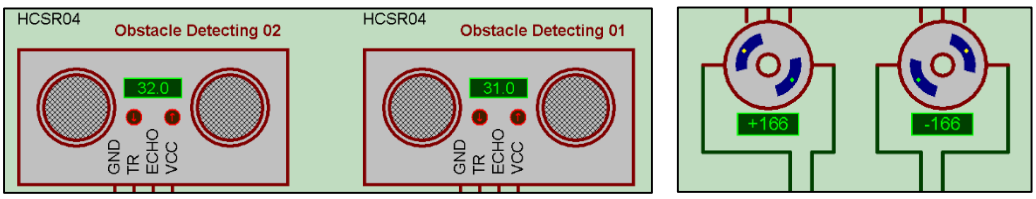


FIGURE 5.0.2: MOTOR OPERATION IN SIMULATION 01

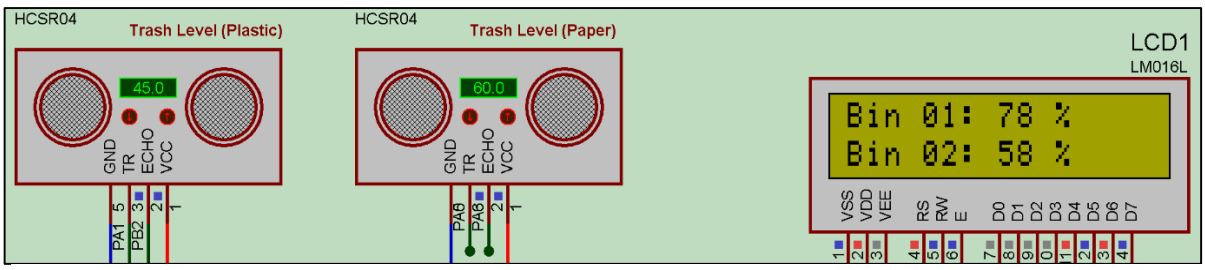


FIGURE 5.0.3: TRASH LEVEL DISPLAY IN SIMULATION 01

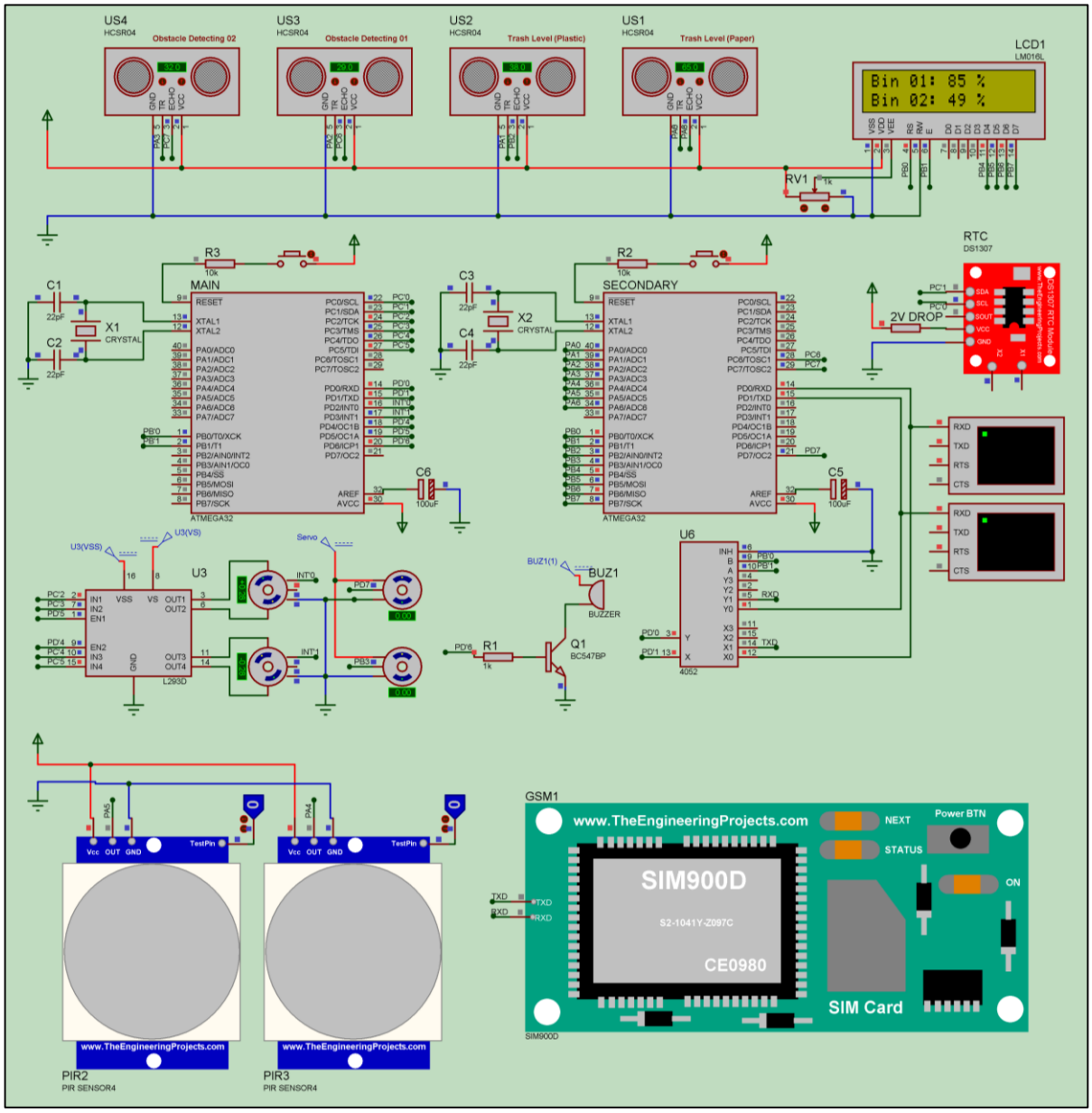


FIGURE 5.0.4: SIMULATION 02

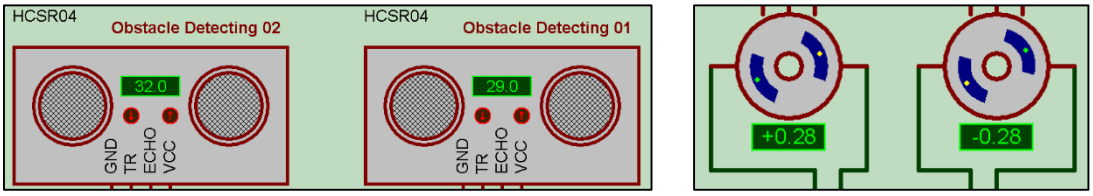


FIGURE 5.0.5: TRASH LEVEL DISPLAY IN SIMULATION 02

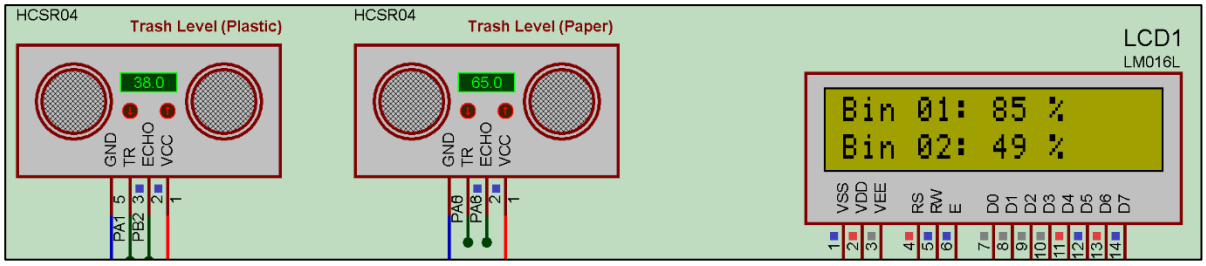


FIGURE 5.0.6: TRASH LEVEL DISPLAY IN SIMULATION 02

FIGURE 5.0.7: 3D DIAGRAM DESIGNING

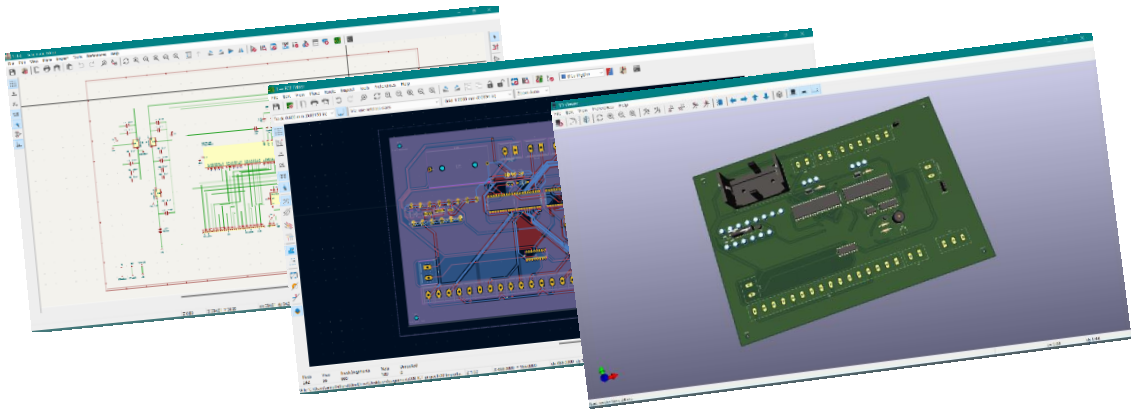


FIGURE 5.0.8: PCB DESIGNING

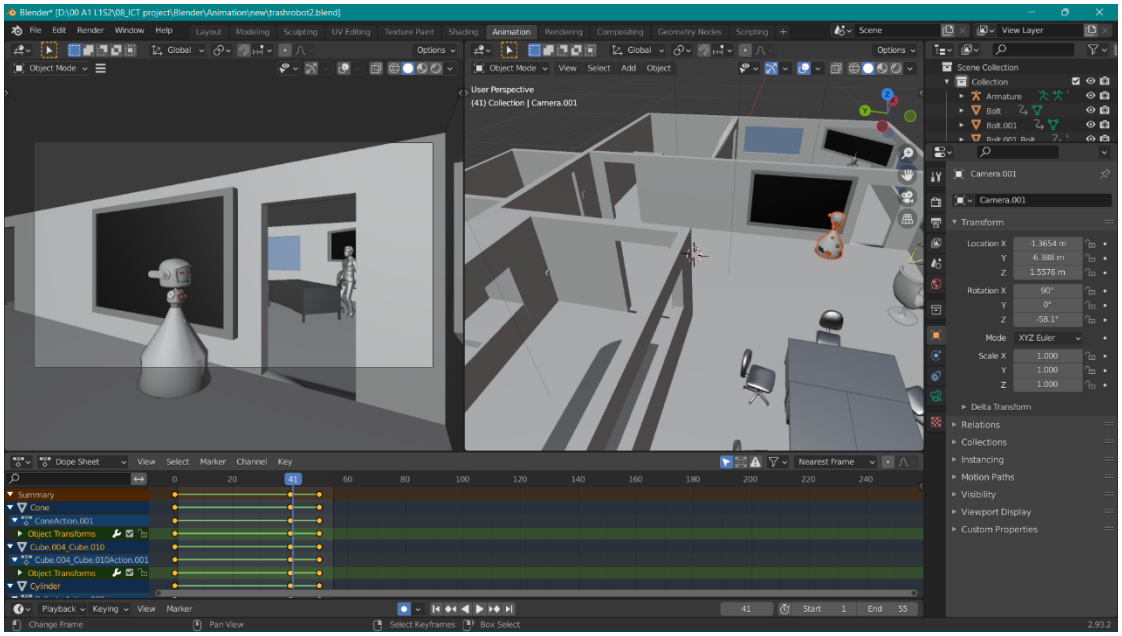


FIGURE 5.0.9: ANIMATION DESIGNING

6.0 Further Work / Improvements

- Navigation system can be improved by using advanced technics like image processing or route giving by touch screen.
- Unknown obstacle detecting can be improved without an alarm. It can be improved to avoid obstacles, turn and take another path.
- We got a problem of sending message to refuse collector when trash level exceeds 90%. It should be solved in any other way.
- It will give more accuracy by using an encoder disk with higher no of opaque lines.
- RFID can be used to locate the robot.

7.0 References

[1] “AliExpress - Online Shopping for Popular Electronics, Fashion, Home & Garden, Toys & Sports, Automobiles and More products,” *AliExpress*. https://www.aliexpress.com/?spm=a2g0o.home.1000002.1.15b92145hJo6nD

[2] “Autonomous Trash Can Takes Itself Out to the Curb,” *Nerdist*. https://nerdist.com/article/autonomous-trash-can/

[3] “Circuit Digest,” *Circuitdigest.com*, 2019. https://circuitdigest.com/.

[4] “Duinolk | The Biggest Arduino Online Store in Sri Lanka,” *www.duino.lk*. https://www.duino.lk/.

[5] “Electronic Store in Sri Lanka,” *Nilambara Electronics*. https://nilambaraelectronics.com/

[6] “Electronic circuits, tutorials and projects,” *Gadgetronicx*. https://www.gadgetronicx.com/.

[7] “ElectronicWings - Hardware Developers Community,” *www.electronicwings.com*. https://www.electronicwings.com/

[8] M. A. Khan, “Garbage collector robot,” *Indian Journal of Science and Technology*, vol. 13, no. 20, pp. 2065–2070, May 2020, doi: 10.17485/ijst/v13i20.212.

[9] GitHub, “GitHub,” *GitHub*, 2018. https://github.com/.

[10] “LK-Tronics || Most Affordable Electronic Components In Sri Lanka.” https://lk-tronics.com/

[11] “Microchip.lk – one stop for all your electronics….” https://microchip.lk/

[12] J. Kanter and J. Kanter, “RS Recommends: The Best Smart Trash Cans for Any Kitchen,” *Rolling Stone*, Sep. 24, 2021. https://www.rollingstone.com/product-recommendations/electronics/best-smart-trash-cans-1229152/.

[13] “Scion Electronics – The Most Trusted Name in Electronics.” https://scionelectronics.com/

[14] A. I. Incorporated, “Self Driving Car Technology on a Trash Can!,” *www.prnewswire.com*. https://www.prnewswire.com/news-releases/self-driving-car-technology-on-a-trash-can-300621889.html

[15] “TRONIC.LK,” *tronic.lk*. https://tronic.lk/

[16] “TrashBot™ a smart waste bin by CleanRobotics,” *CleanRobotics*. https://cleanrobotics.com/trashbot/.

[17] “Zenix Store - ZENIX Store,” *store.zenix.lk*. https://store.zenix.lk/

[18] “electroSome - Electronics Tutorials, Projects and Products,” *electroSome*. https://electrosome.com/

Appendix A

Cost Estimation

[1] [4] [5] [10] [11] [13] [15] [17]

TABLE 1: COST ESTIMATION

Component	Cost per Unit (Rs.)	Quantity	Cost (Rs.)
PIR Sensor HC-SR501	360	2	720
Ultrasonic Sensor Module HC-SR04	400	4	1600
HC-020K Double Speed Measuring Sensor Module with Photoelectric Encoders Kit	500	2	1000
SIM900A Module (GSM GPRS)	2800	1	2800
DS1307 Real time clock	250	1	250
Atmega32A-PU - PDIP	1370	2	2740
MG995 Tower Pro Digital Servo Motor	1500	2	3000
12VDC 200RPM ZGA25RP High Torque Gear Motor	1490	2	2980
L293D Motor Driver Mini Expansion Board	390	1	390
1602 LCD Display 16*2 (Blue)	885	1	885
5V Mini Buzzer (SP0001)	80	1	80
65mm Rubber Tire with Sponge Liner	480	2	960
Steel Ball Swivel Caster Wheel	150	2	300
12V 3000mAh Lithium Rechargeable Battery	5500	1	5500
Battery Charger	720	1	720
7805 5V Voltage Regulator	40	1	40
7812 12V Voltage Regulator	35	1	35
10µF Capacitor	25	2	50
470µF Capacitor	100	2	200
Resistors	20	10	200
Jumper Wires	175	2	350
Circuit Board	200	1	200
Demultiplexer	50	1	50
Total Cost			**25050**

Appendix B

Individual Contribution

Name of student: Hathnagoda H.W.E.M.J.N (204071C) [3] [6] [7] [18]

- As the leader of the group, I divided the work among group members. And I checked the progress they have reached in each area that they were assigned to.
- According to that below are the hardware components that were assigned to me.

    1. DC Motor
    2. L293D Motor Driver
    3. Wheel Encoders
    4. Buzzer

- Apart from above work I designed complete circuit diagram by assigning relevant pins to my group members.
- I helped my friends to understand some coding parts and also helped to make the literature survey.
- I helped for integrating the full code and as the leader I'm coordinating the whole project.
- I learnt and did the PCB design for my individual components.

1. DC Motor

- As first my task was to choose a suitable DC motor to move the robot. So, I calculated the power needed for moving the robot by taking assumptions. So, by using the calculations I selected a suitable Dc motor with required power and torque.

Specifications:

- No-load Speed: 200RPM at 12 Volts
- Load speed: 158RPM
- Power: 7.2W
- No-load Current: 0.15A
- Load Current: 0.6A
- Reduction Ratio: 1: 22
- Input Torque = 2.2 kg.cm (0.215754Nm)
- Output Torque = 0.215754Nm * 22
  = 4.7465Nm

Working Principle:

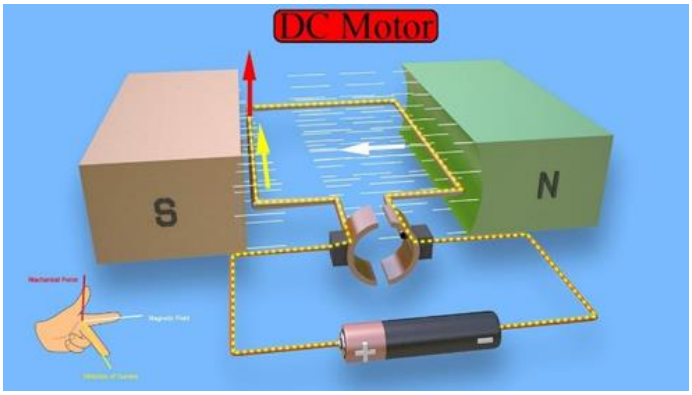


FIGURE B1.1: WORKING PRINCIPLE OF DC MOTOR

- When kept in a magnetic field, current carrying conductor gains torque and develops tendency to move. In short when electric field and magnetic fields interact, a mechanical force arises.
- Next challenge was to rotate the robot. For that I stopped one DC motor and moved another forward. To rotate it by 90 degrees I took stopped wheel as the center of the circle and arc length as the moved distance by other motor. Radius of the circle is the length between two wheels.

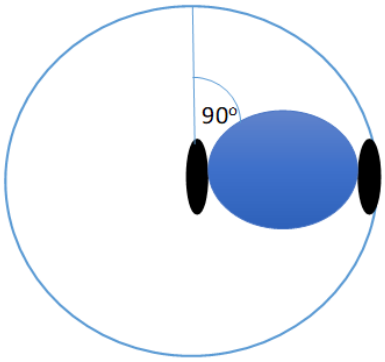


FIGURE B1.2: MECHANISM OF ROTATING

2. L293D Motor Driver

- The voltage comes from Atmega32 microcontroller was not enough to drive 2 dc motors. So, I used a motor driver called “L293D” motor driver for driving 2 motors. It uses H bridge concept.

Specifications:

- Max Voltages:

  V_s - 36V V_{ss} - 36V Vin – 7V V_{en} - 7V

- Can supply 600mA current per channel continuous and 1.2A peak

H Bridge Concept:

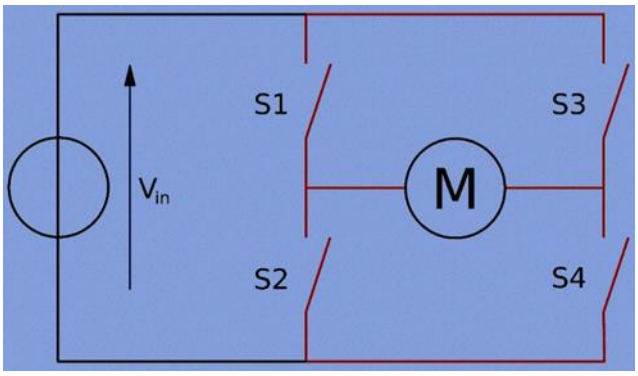


FIGURE B1.3: H BRIDGE CONCEPT

- A H-bridge is fabricated with four switches like S1, S2, S3 and S4. When the S1 and S4 switches are closed, then a $+V_e$ voltage will be applied across the motor. By opening the switches S1 and S4 and closing the switches S2 and S3, this voltage is inverted, allowing invert operation of the motor.

3. Optical Encoders

- After that my main challenge was to calculate the distance moved by the robot. After researching for weeks, I used optical encoder for calculating the moved distance. There were other encoders too which were way expensive. For accuracy we can use them.

Working Principle:

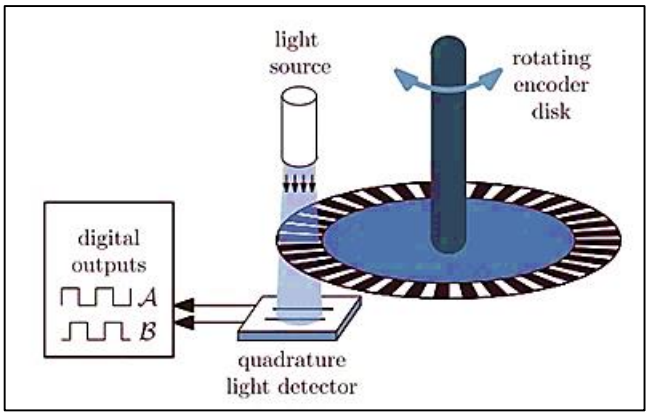


FIGURE B1.4: WORKING PRINCIPLE OF ENCODER

- ❖ A beam of light emitted from an LED pass through a Code Disk, a transparent disk patterned with opaque lines (much like the spokes on a bike wheel). The light beam is picked up by a Photodetector Assembly, also called a photodiode array or a photo sensor.
- ❖ The Photodetector Assembly responds to the light beam, which is transformed into a square wave or pulse train and sends to the microcontroller.

DISTANCE = PULSE COUNT / PULSES PER ROTATION

Specifications:

- ❖ Module Working Voltage: 4.5-5.5V
- ❖ Launch tube current: < 20mA

- Finally, I checked the moved distance in the proteus software by using a LCD display. In here there was motor encoders which includes encoders inside the motor in proteus software.

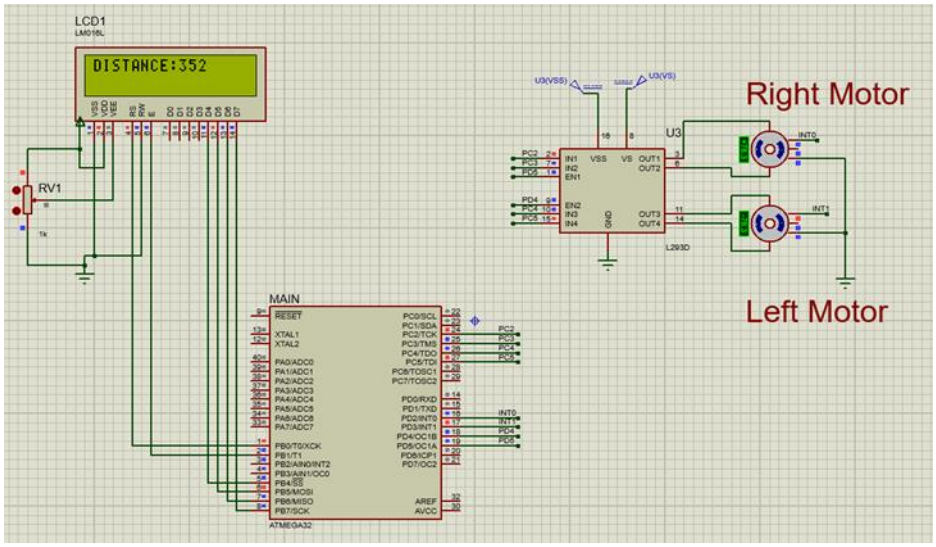


FIGURE B1.5: SIMULATION

4. Buzzer

- After that my main challenge was to calculate the distance moved by the robot. After researching for weeks, I used optical encoder for calculating the moved distance. There were other encoders too which were way expensive. For accuracy we can use them.

Working Principle:

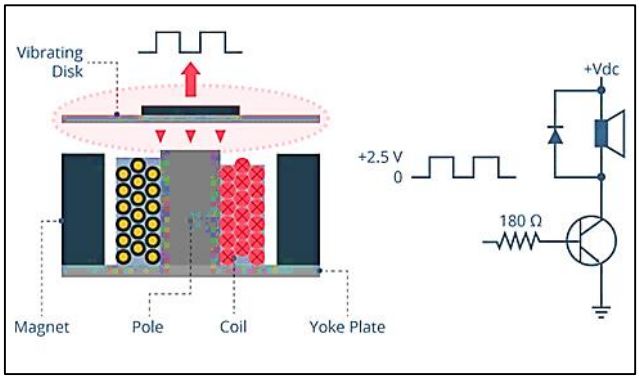


FIGURE B1.6: WORKING PRINCIPLE OF BUZZER

- When power is applied, current runs through the coil of wire inside the buzzer, which produces a magnetic field. The flexible Vibrating disk is attracted to the coil when the magnetic field is activated, then returns to rest when the magnetic field is off. The active buzzer can sound continuously by directly connecting the rated power supply.

Schematic Diagram:

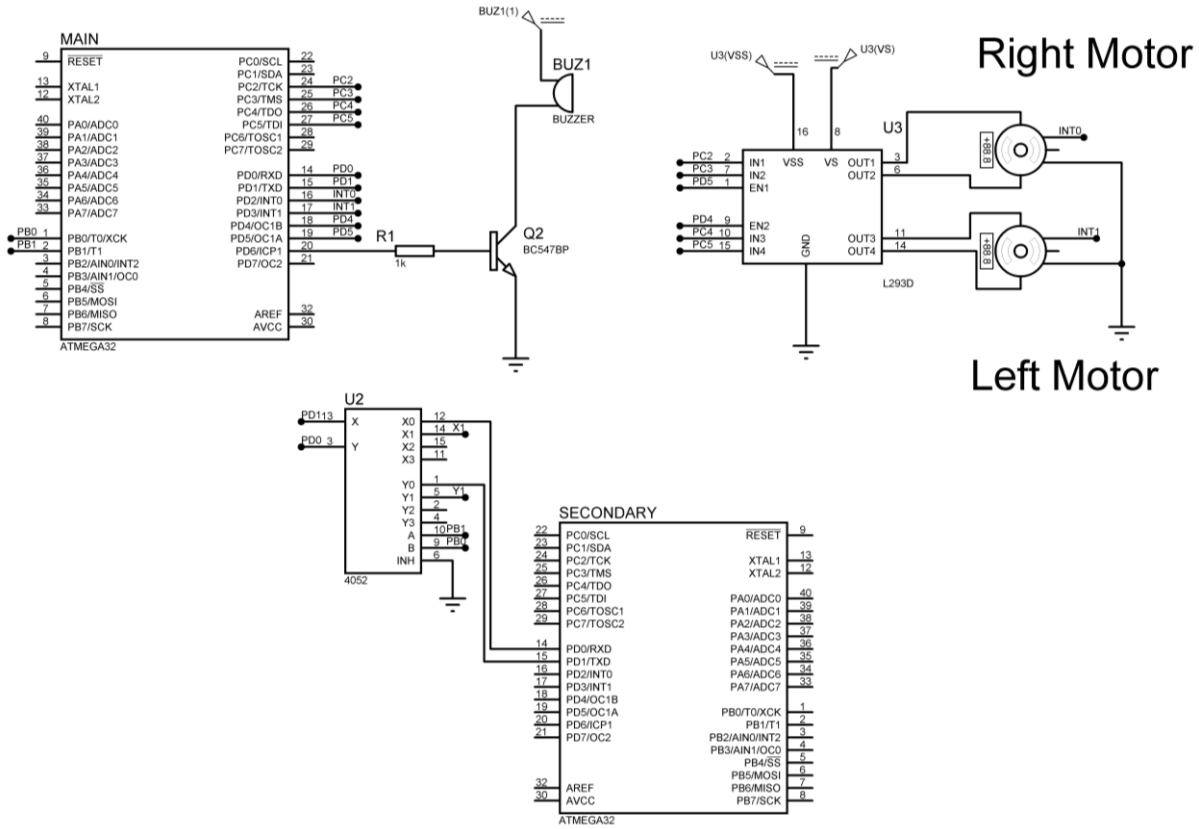


FIGURE B1.7: SCHEMATIC DIAGRAM

PCB Design:

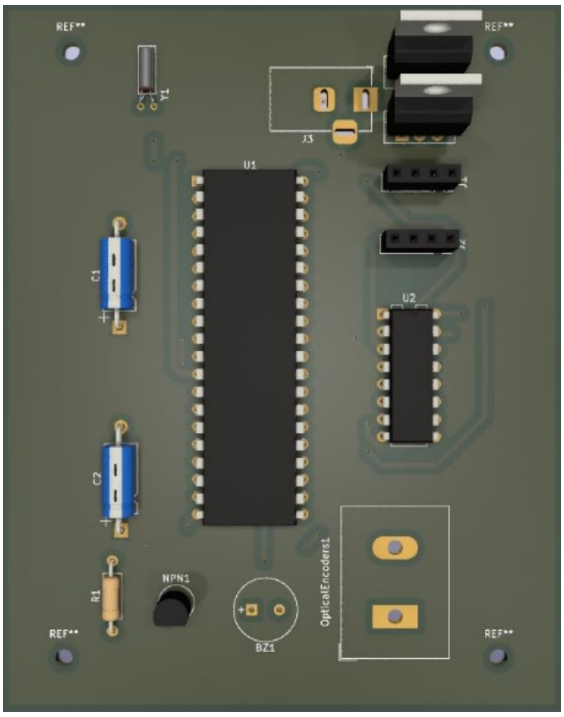


FIGURE B1.8: FRONT VIEW

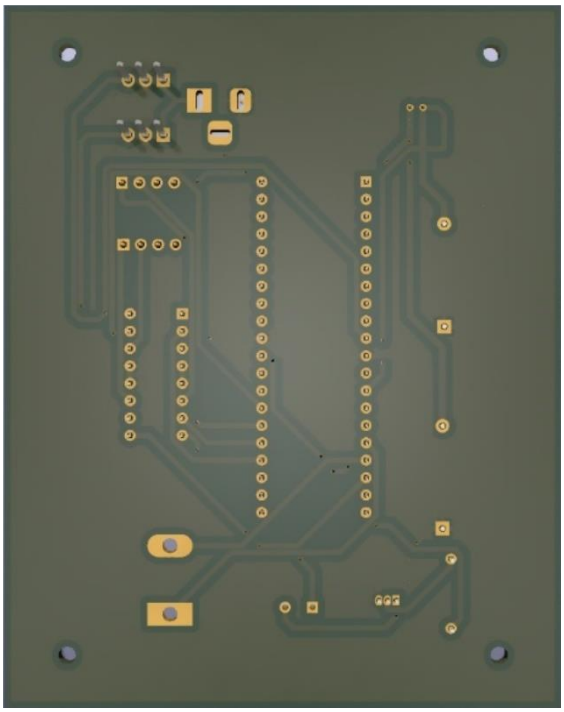


FIGURE B1.9: BACK VIEW

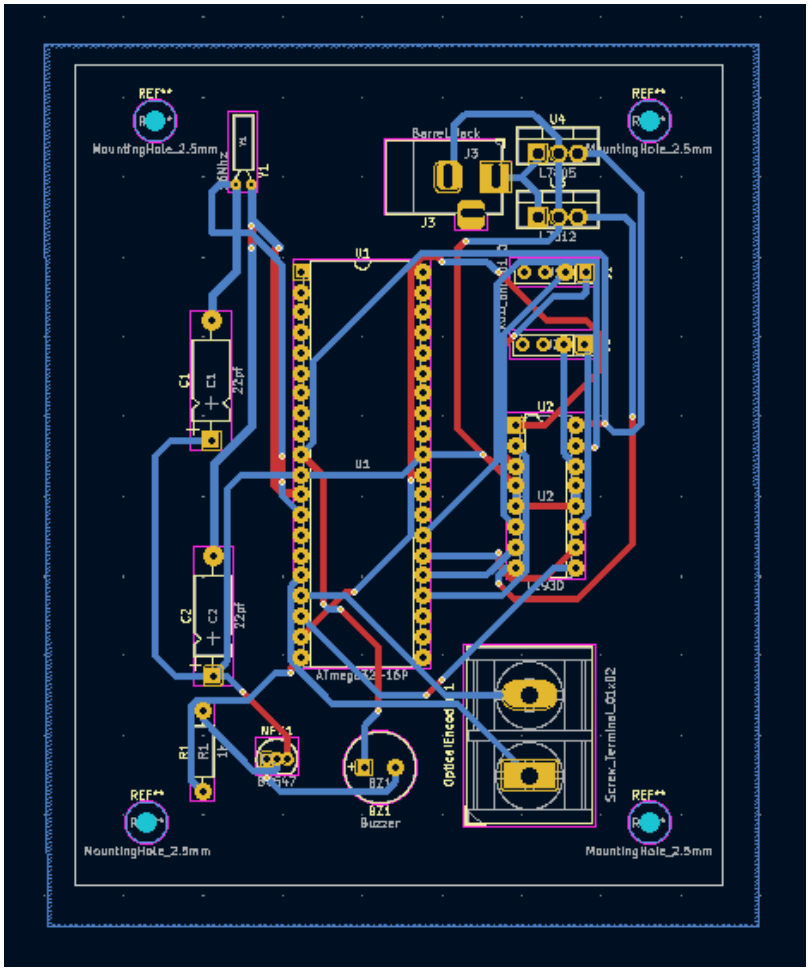


FIGURE B1.10: COMPONENT LAYER

Codes:

Motor.h

```
#include <avr/io.h>

#ifndef MOTOR_H
#define MOTOR_H

#define OC1A_PORT D
#define OC1A_PIN PD5

#define OC1B_PORT D
#define OC1B_PIN PD4


#define DDR(a) __CONCAT(DDR,a)
#define PORT(a) __CONCAT(PORT,a)
#define PIN(a) __CONCAT(PIN,a)


#define MOTOR_STOP      0
#define MOTOR_CW        1
#define MOTOR_CCW       2

void MotorInit();
void MotorA(uint8_t dir,uint8_t speed);
void MotorB(uint8_t dir,uint8_t speed);

#endif
```

Motor.c

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include "motor.h"

void MotorInit(){
        TCCR1A=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM10);
        TCCR1B=(1<<CS11)|(1<<CS10);
        DDR(OC1A_PORT)|=(1<<OC1A_PIN);
        DDR(OC1B_PORT)|=(1<<OC1B_PIN);
        DDRC|=0X3C;
}

void MotorA(uint8_t dir,uint8_t speed){
        //Direction
        if(dir == MOTOR_STOP){
                PORTC&=(~(1<<PC2));
                PORTC&=(~(1<<PC3));
        }
        else if(dir == MOTOR_CCW){
                PORTC&=(~(1<<PC2));
                PORTC|=(1<<PC3);
        }
        else if(dir == MOTOR_CW){
                PORTC&=(~(1<<PC3));
                PORTC|=(1<<PC2);
        }

        //Speed
        uint8_t sreg=SREG;
        cli();
        OCR1A=speed;
        SREG=sreg;
}

void MotorB(uint8_t dir,uint8_t speed){
        //Direction
        if(dir == MOTOR_STOP){
                PORTC&=(~(1<<PC4));
                PORTC&=(~(1<<PC5));
        }
        else if(dir == MOTOR_CCW){
                PORTC&=(~(1<<PC4));
                PORTC|=(1<<PC5);
        }
        else if(dir == MOTOR_CW){
                PORTC&=(~(1<<PC5));
                PORTC|=(1<<PC4);
        }
        //Speed
        uint8_t sreg=SREG;
        cli();
        OCR1B=speed;
        SREG=sreg;
}
```

Main.c

```
#define F_CPU 16000000UL
#include <stdint.h>
#include <avr/delay.h>
#include <stdlib.h>
#include <avr/interrupt.h>
#include <avr/io.h>
```

```
//Encoder Codes
void EncoderInit(void){
        //set pins as input
        DDRD &= ~((1<<PD2)|(1<<PD3));
        //enable interrnal pullups;
        PORTD |= (1<<PD2)|(1<<PD3);
```

```
#include <util/delay.h>
#include "motor.h"
#include "buzzer.h"
#include "LCD.h"
volatile uint16_t countLeft = 0;
volatile uint16_t countRight = 0;
uint32_t disLeft = 0;
uint32_t disRight = 0;
uint32_t speedRight;
uint32_t speedLeft;
#define Init_ticks 100
#define Time_Interval 0.01
#define PPR 20.0
#define R 6.5
#define WHEELCIRCUMFERENCE 40.84
#define GearRatio 73.6
void EncoderInit(void);
void init_system();

int main(){
	init_system();
	LCD_Init();
	while(1){
		char numberString[4];
		LCD_SetCursor(1, 1);
		LCD_Print("DISTANCE(cm):");
		LCD_SetCursor(14, 1);
		itoa(disLeft, numberString, 10);
		LCD_Print(numberString);
		if(disLeft < 500){
			//Start Moving Forward
			MotorA(MOTOR_CW,120);
			MotorB(MOTOR_CCW,120);
			uint8_t wheelspeed;
			if(disLeft > 380){
				wheelspeed = 10 * ((500 -
disLeft)/10);
				if(wheelspeed < 5){
					wheelspeed = 0;
				}
	 MotorA(MOTOR_CW,wheelspeed);
			 MotorB(MOTOR_CCW,wheelspeed);
			}
		}
		else{
			MotorA(MOTOR_STOP,0);
			MotorB(MOTOR_STOP,0);
			buzzer();
		}
	}
	return 0;
}
```

```
}
void CheckEncoders(void){
	disRight += (countRight/PPR) * WHEELCIRCUMFERENCE;
	disLeft += (countLeft/PPR) * WHEELCIRCUMFERENCE;
	speedRight =
((countRight/(Time_Interval*GearRatio))/PPR)*60.0;  // Speed
equation
	speedLeft=
((countLeft/(Time_Interval*GearRatio))/PPR)*60.0;
	countRight = 0;
	countLeft = 0;
	TCNT0 = Init_ticks;
}
void Timer0_Start(void){
	TCNT0 = Init_ticks;			/* Load TCNT0, count
for 10ms*/
	TCCR0 = (1<<CS02) | (1<<CS00); /* Start timer0 with
/1024 prescaler*/
	sei();
	TIMSK=(1<<TOIE0);		/* Enable Timer0 overflow
interrupts */
}
ISR(TIMER0_OVF_vect){
	//reading Encoders
	CheckEncoders();
}

/* Interrupt Service Routine for INT0 to count each pulse from
encoder */
ISR(INT0_vect){
	countRight++;
}

/* EX PIN ISR to count each pulse from encoder */
ISR(INT1_vect){
	countLeft++;

}
void init_system(){
	//Initialize motor subsystem
	MotorInit();
	EncoderInit();
	GICR = 1<<INT0;		/* Enable INT0*/
	MCUCR = 1<<ISC01 | 1<<ISC00;

	GICR = 1<<INT1;		/* Enable INT1*/
	MCUCR = 1<<ISC11 | 1<<ISC10;

	sei();			/* Enable Global Interrupt */
	Timer0_Start();

}
```

Buzzer.h

```
#ifndef BUZZER_H_
#define BUZZER_H_
void buzzer(){
	DDRD |= (1 << PIND6);  //make b0 output pin
	while (1){
		PORTD |= (1 << PIND6); //b0 1
		_delay_ms(10);
		PORTD &= ~(1 << PIND6); //b0 0
		_delay_ms(10);
	}
}
```

Name of student: Jayasekara J.P.I.A (204084T) [3] [6] [7] [18]

1. PIR Sensor
    - Detect hand movements for lids opening process.
2. Robot Navigation Part.
3. 3D Modeling.
    - Created the 3D diagram of the Trash Robot by using the Blender Software.
4. PCB Design.
    - Design full schematic and PCB of project by using KiCAD software.

- Helped team members to design the individual PCB parts.
- Helped to integrate the full code and finalized the circuit diagram.
- Helped to prepare the project proposal and presentation.

1. PIR Sensor

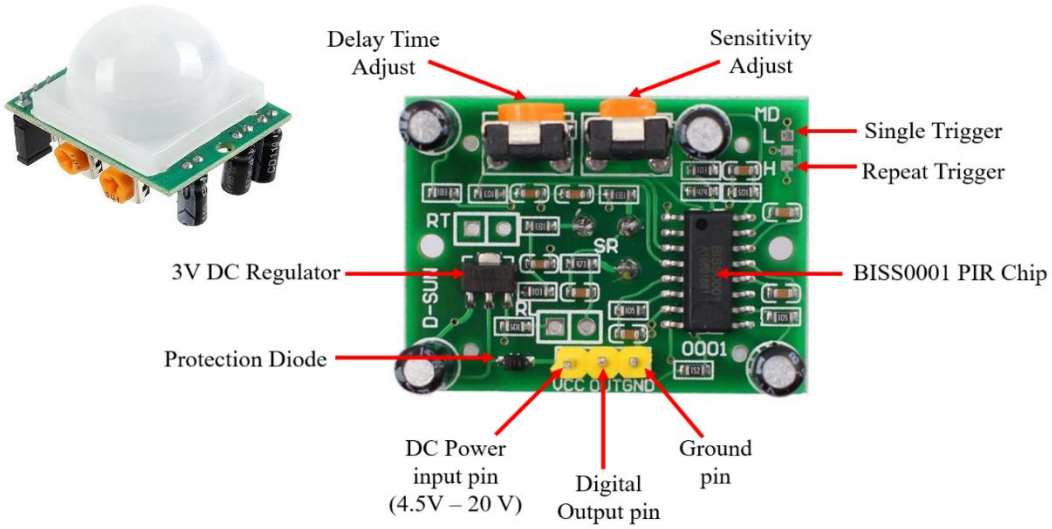


FIGURE B2.1: PIR SENSOR AND LABELED DIAGRAM OF PIR SENSOR

Specifications:

- ❖ Maximum Voltage: 20 V (Operating voltage range-DC: 4.5V-20V)
- ❖ Maximum Current: 65 mA
- ❖ Maximum Sensing Distance: 7m
- ❖ Angle Sensor: 110º
- ❖ Adjustable Delay Time: 3s to 5min
- ❖ TTL Output: High - 3.3V / Low - 0V
- ❖ Trigger Methods: L-Disable repeat trigger, H-Enable repeat trigger

- HC SR501 Motion Detector/ Body Sensor Module is based on infrared technology, and it is highly sensitive, high reliable, low power consumption. Basically, made from a pyroelectric sensor which can detect levels of IR. It allows to sense movement of human hand in or out of the sensing range. So, it detects the infrared/ thermal radiation emitted or reflected from human hand and does not generate or radiate energy by itself. Therefore, in this project PIR sensor is more reliable motion detection sensor than Ultrasonic Sensor or Active IR Sensor.

FIGURE B2.2: PYROELECTRIC SENSOR

FIGURE B2.3: GRID EYE

- There are two potentiometers. By adjusting the distance potentiometer clockwise increases the sensing range and vice versa. By adjusting the delay potentiometer clockwise increases the delay time and vice versa. This project used distance potentiometer as well as cover the sensor for restrict the detection range.

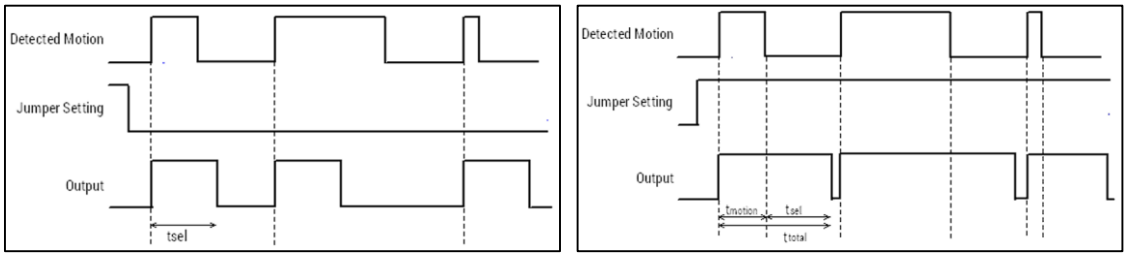


FIGURE B2.4: SINGLE TRIGGER AND REPEAT TRIGGER MODE TIMING DIAGRAM

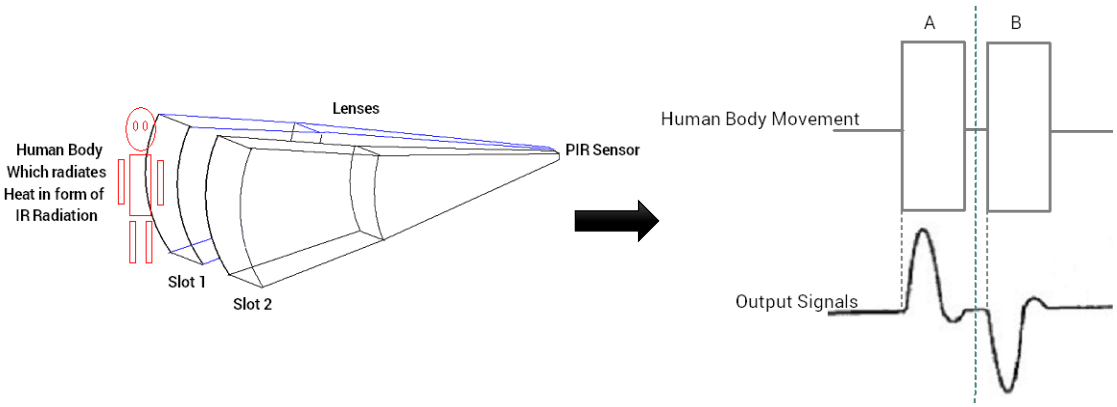


FIGURE B2.5: DIFFERENTIAL CHANGES BETWEEN TWO SLOTS

- In this project used repeat trigger mode because when object stops or disappears from sensing range, PIR continues its high state up to certain specific delay.
- Detected temperature is calculated based on Stefan Boltzmann Law

$$T_c = \sqrt{T_s^4 + \frac{\Phi}{A \sigma \varepsilon_s}}, \quad [T_s = \text{sensor's surface temperature } T_c = \text{object's temperature in Kelvin}]$$

$$[\Phi = \text{magnitude of net thermal radiation flux } \Phi, \varepsilon = \text{emissivity of the object}]$$

Code:

```
#include <avr/io.h>
#define F_CPU 16000000UL

int main(void) {
  DDRB = DDRB | (1<<3);            //Make PB3 as output pin
  DDRD = DDRD | (1<<7);            //Make PD7 as output pin
  DDRA = DDRA & (~(1<<4));                  //Make PA4 as input pin
  DDRA = DDRA & (~(1<<5));                  //Make PA5 as input pin

   while (1) {
                    if (PINA & (1<<4)){
                              PORTB = PORTB | (1<<3);
                    }
                    else{
                              PORTB = PORTB & (~(1<<3));
                    }

                    if (PINA & (1<<5)){
                              PORTD = PORTD | (1<<7);
                    }
                    else{
                              PORTD = PORTD & (~(1<<7));}
          }
}
```

Schematic Diagram:

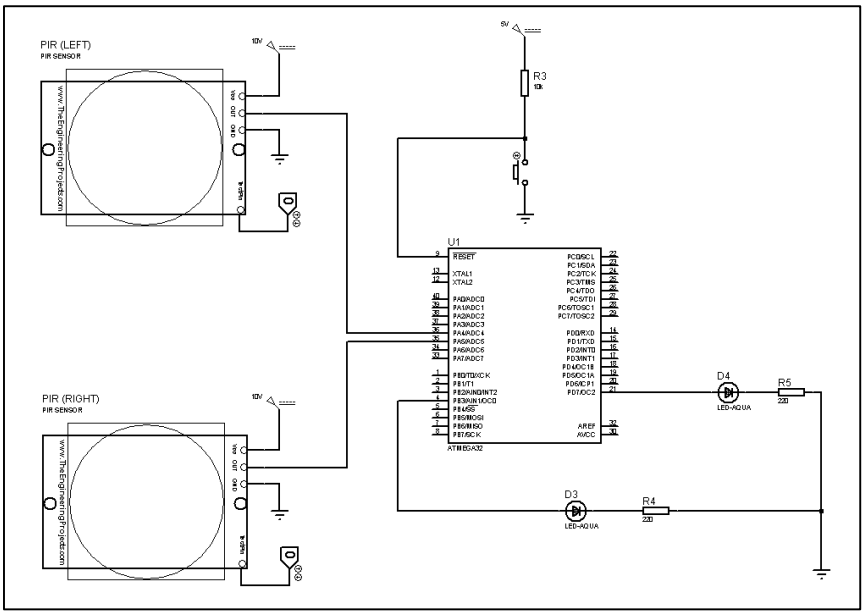


FIGURE B2.6: SCHEMATIC DIAGRAM

PCB Design:

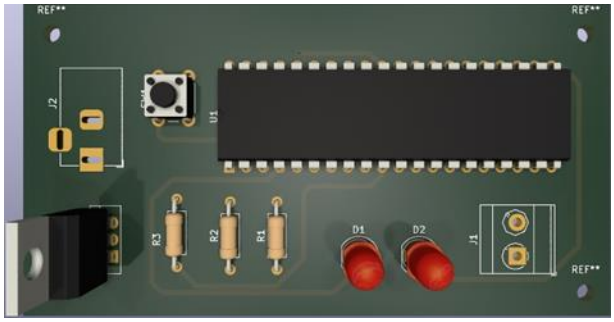


FIGURE B2.7: FRONT VIEW

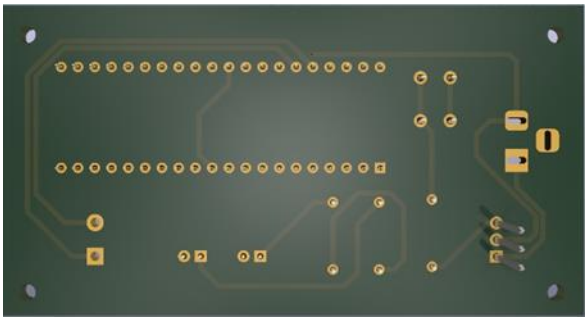


FIGURE B2.8: BACK VIEW

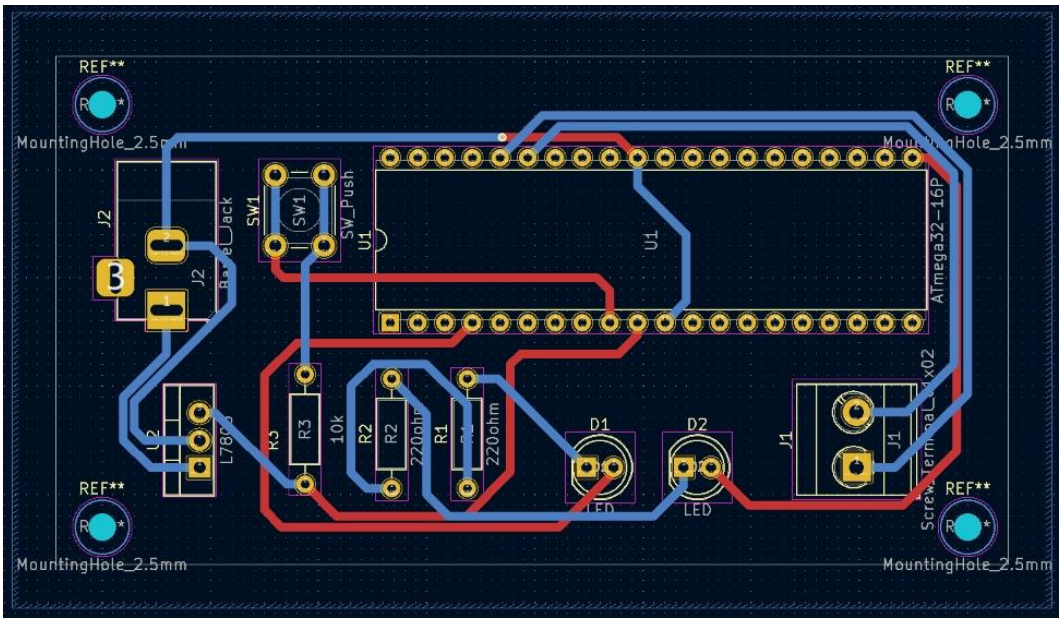


FIGURE B2.9: COMPONENT LAYER

2. Robot Navigation Part

- Previously decided to give path by using A* algorithm. But this robot needs to come back its initial position. By using A* algorithm it difficult to give path. Because it finds shortest path to the destination.
- Lately, we found this path can manually give by using motor encoder. Therefore, we left A* algorithm and go with this new method.

Code:

```
if (disLeft <= 1580){
        if(disLeft < 500){
                MotorA(MOTOR_CW,255);
                MotorB(MOTOR_CCW,255);
                uint8_t wheelspeed;
                if(disLeft > 380){
                        wheelspeed = 10 * ((500 - disLeft)/10);
        if(wheelspeed < 5){
                        wheelspeed = 0;
        }
                MotorA(MOTOR_CW,wheelspeed);
                MotorB(MOTOR_CCW,wheelspeed);
            }
    }
      else if (500 <= disLeft && disLeft < 580){
                MotorA(MOTOR_STOP,0);
                MotorB(MOTOR_CCW,150);
        }
        else if(disLeft == 580){
                MotorA(MOTOR_STOP,0);
                MotorB(MOTOR_STOP,0);
      //Start Moving Forward
                MotorA(MOTOR_CW,255);
                MotorB(MOTOR_CCW,255);
        }
        else{     //Start Moving Forward
                MotorA(MOTOR_CW,255);
                MotorB(MOTOR_CCW,255);
        }
}
else{
        MotorA(MOTOR_STOP,0);
        MotorB(MOTOR_STOP,0);
}
```

3. 3D Modeling

- Created 3D Model of the robot by using the Blender software.

FIGURE B2.10: 3D MODELING

4. PCB Design

- Design full schematic and PCB of project by using KiCAD software.

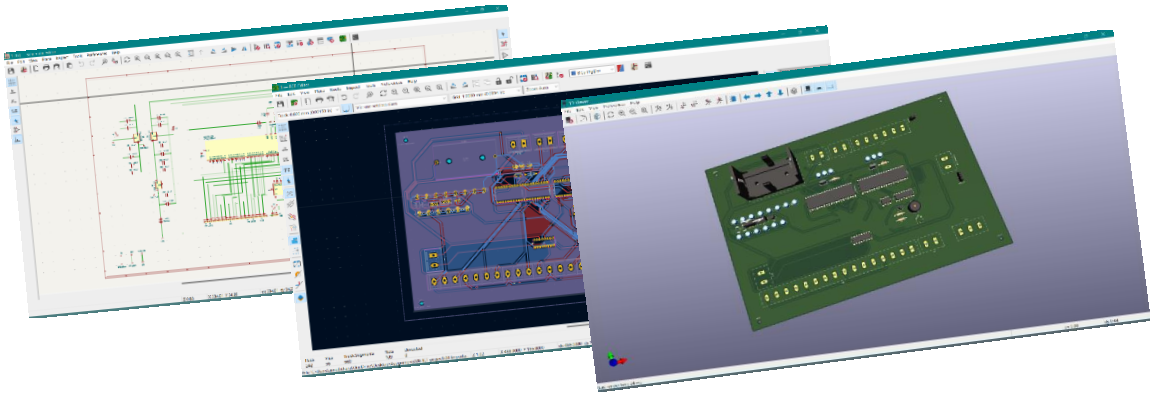


FIGURE B2.11: PCB DESIGNING

Name of student: Madhushika B.A.E (204118E) [3] [6] [7] [18]

1. Ultrasonic Sensor
    - Measure the trash level of separate bins
    - Obstacle avoidance (Detect the obstacles)
2. LCD Display
    - Display the trash level of each bin.

- Helped to integrate the full code.
- Made the project proposal, presentation and other documents with the help of team members.
- Finalized the circuit diagram with team members.

1. Ultrasonic Sensor

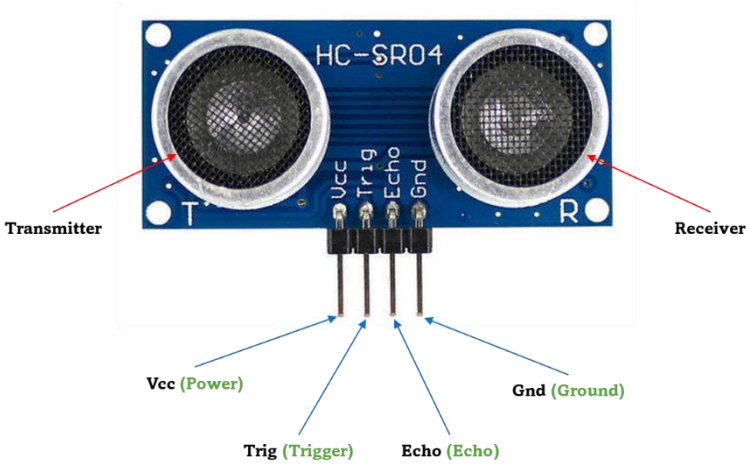


FIGURE B3.1: PIN DIAGRAM

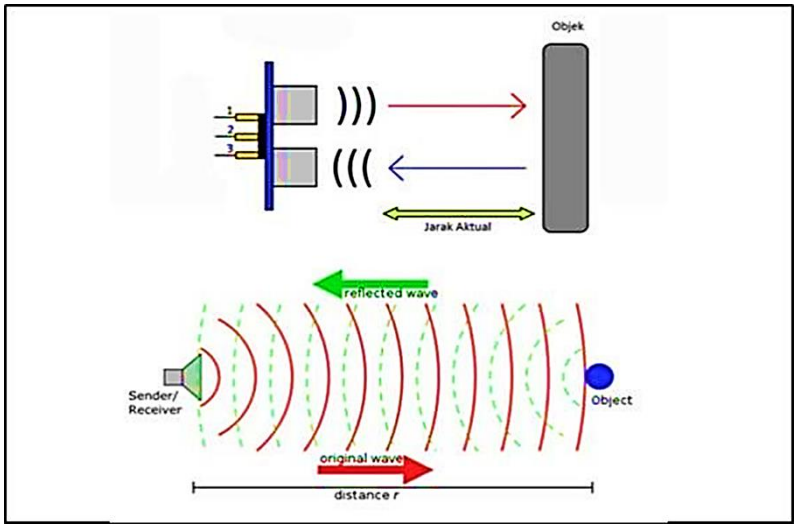


FIGURE B3.2: WORKING PRINCIPLE

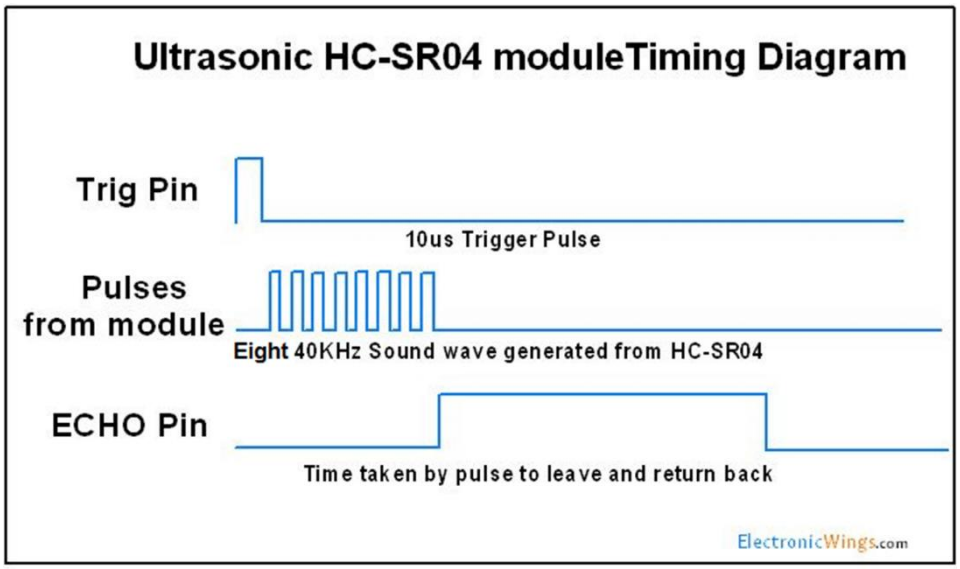


FIGURE B3.3: TIMING DIAGRAM

Calculate Distance:

Distance = Speed x Time

- Speed of sound waves is approximately 343 ms^{-1}

$$\textbf{Total Distance} = \frac{\textbf{343 ms}^{-1}\textbf{ x Time of High(Echo) Pulse}}{\textbf{2}}$$

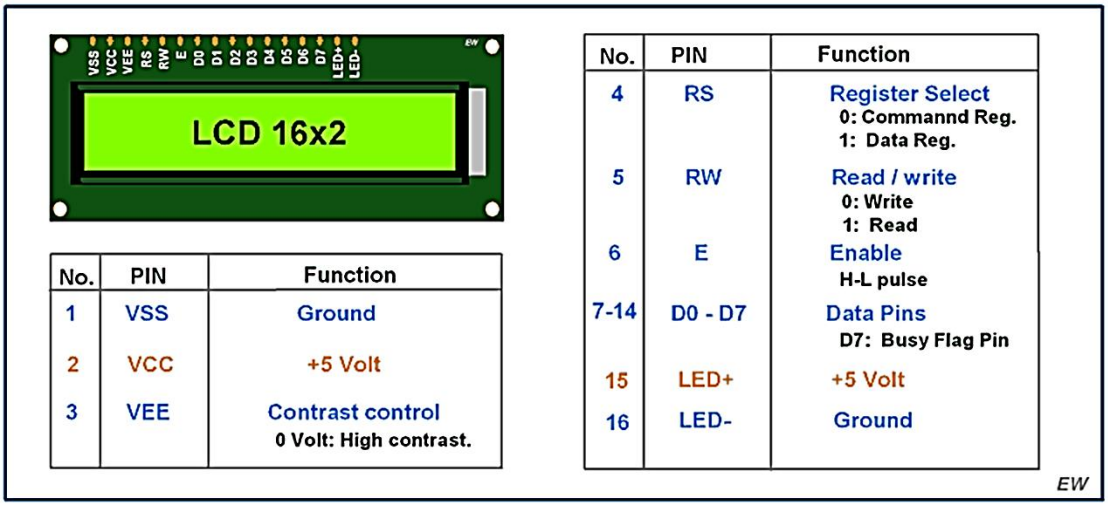
- We divide this by 2, because we get the time taken by pulse to leave and return.

Specifications:

- Maximum Voltage: 5 V
- Maximum Current: 15 mA
- Working Frequency: 40 Hz
- Effectual Angle: 15º
- Max Range: 4 m
- Min Range: 2 cm

- Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR system.
- An ultrasonic sensor generates the high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver.
- By measuring the time required for the echo to reach to the receiver, we can calculate the distance.

2. LCD



No.	PIN	Function
1	VSS	Ground
2	VCC	+5 Volt
3	VEE	Contrast control 0 Volt: High contrast.

No.	PIN	Function
4	RS	Register Select 0: Commannd Reg. 1: Data Reg.
5	RW	Read / write 0: Write 1: Read
6	E	Enable H-L pulse
7-14	D0 - D7	Data Pins D7: Busy Flag Pin
15	LED+	+5 Volt
16	LED-	Ground

EW

FIGURE B3.4: PIN DESCRIPTION

Specifications:

- Maximum Voltage: 5.3 V
- Maximum Current: 3.3 mA
- Number of characters: 32
- Number of columns: 16
- Number of rows: 2
- Working in 4-bit or 8-bit mode

Schematic Diagram:

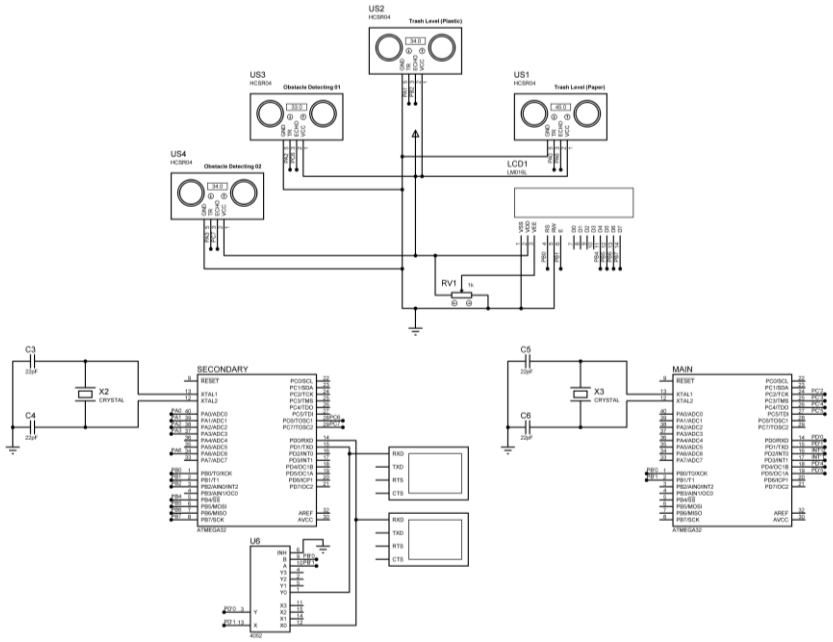


FIGURE B3.5: SCHEMATIC DIAGRAM

PCB Design:

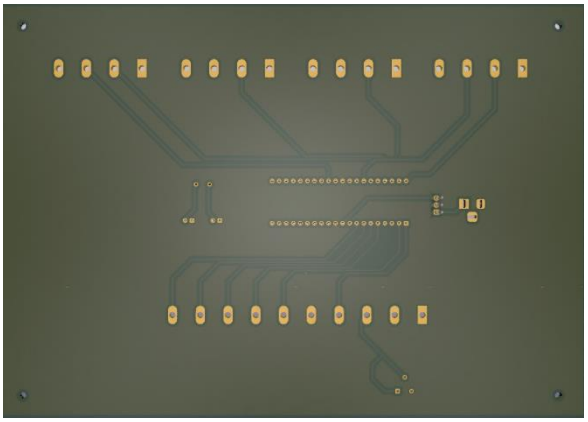


FIGURE B3.6: FRONT VIEW

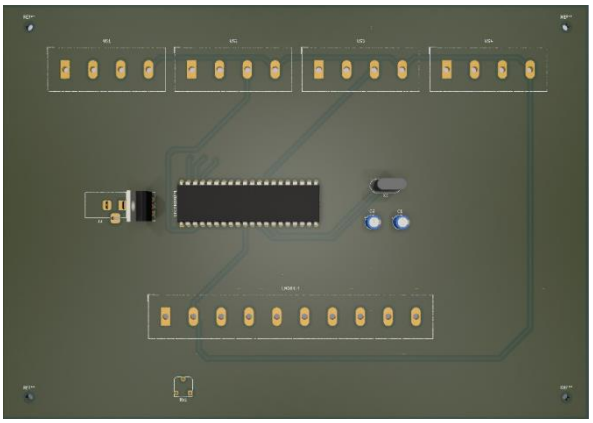


FIGURE B3.7: BACK VIEW

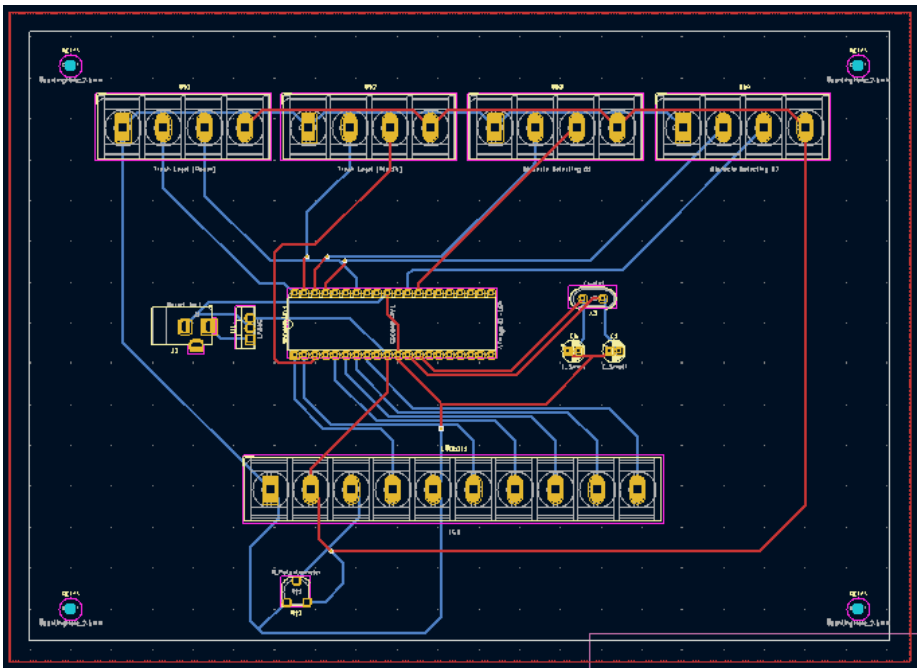


FIGURE B3.8: COMPONENT LAYER

Codes:

Utrasonic.h

```
#ifndef ULTRASONIC_H_
#define ULTRASONIC_H_

#define US1_TRIG_POS PA0      // Trigger pin on PA0
#define US1_ECHO_POS PA6      // Echo pin on PA6
#define US3_TRIG_POS PA2      // Trigger pin on PA2
#define US3_ECHO_POS PC6      // Echo pin on PA6
#define US2_TRIG_POS PA1      // Trigger pin on PA1
#define US2_ECHO_POS PB2      // Echo pin on PB2
#define US4_TRIG_POS PA3      // Trigger pin on PA3
#define US4_ECHO_POS PC7      // Echo pin on PC7
#define  US_NO_OBSTACLE       -2

// Function Prototypes
void HCSR04Init();
```

```
        if(i==600000)
        return US_NO_OBSTACLE;    //Time out

        result=TCNT1;             //Falling edge found
        TCCR1B=0x00;              //Stop Timer

        if(result > 60000)
        return US_NO_OBSTACLE;    //No obstacle
        else
        return (result>>1);
}

void HCSR04Trigger3(){
        PORTA |= (1<<US3_TRIG_POS);     //high
        _delay_us(15);                  //wait 15uS
```

```
void HCSR04Trigger1();
void HCSR04Trigger2();
uint16_t GetPulseWidth1();
uint16_t GetPulseWidth2();
void HCSR04Trigger3();
void HCSR04Trigger4();
uint16_t GetPulseWidth3();
uint16_t GetPulseWidth4();
void readDistances();

int distance1, previous_distance1, percentage1, distance2,
previous_distance2, percentage2, distance3=31, distance4=31;

char numberString1[4], numberString2[4];

int distances[2] = {31,31};   //stores the distances in an array

void HCSR04Init(){
        DDRA|=(1<<US1_TRIG_POS);   // Trigger pin as output
        DDRA|=(1<<US2_TRIG_POS);
        DDRA|=(1<<US3_TRIG_POS);   // Trigger pin as output
        DDRA|=(1<<US4_TRIG_POS);
}

void HCSR04Trigger1(){
        PORTA |= (1<<US1_TRIG_POS);     //high
        _delay_us(15);                  //wait 15uS
        PORTA &= ~(1<<US1_TRIG_POS);    //low
}

void HCSR04Trigger2(){
        PORTA |= (1<<US2_TRIG_POS);     //high
        _delay_us(15);                  //wait 15uS
        PORTA &= ~(1<<US2_TRIG_POS);    //low
}

uint16_t GetPulseWidth1() {
        uint32_t i,result;
        for(i=0;i<600000;i++){
                if(!(PINA & (1<<US1_ECHO_POS)))
                continue;           //Line is still low
                else
                break;              //High edge detected
        }
        TCCR1A=0X00;
        TCCR1B=(1<<CS11);           // Resolution of the timer
        TCNT1=0x00;                 // Start counting time

        for(i=0;i<600000;i++){
                if(PINA & (1<<US1_ECHO_POS))
                {
                        if(TCNT1 > 60000)
                        break;    // No object in the range
                        else
                        continue;
                }
                else
                break;
        }
        if(i==600000)
        return US_NO_OBSTACLE;   //Time out

        result=TCNT1;               //Falling edge found

        TCCR1B=0x00;                //Stop Timer

        if(result > 60000)
        return US_NO_OBSTACLE;    //No obstacle
        else
        return (result>>1);
}
        PORTA &= ~(1<<US3_TRIG_POS);    //low
}

void HCSR04Trigger4(){
        PORTA |= (1<<US4_TRIG_POS);     //high
        _delay_us(15);                  //wait 15uS
        PORTA &= ~(1<<US4_TRIG_POS);    //low
}

uint16_t GetPulseWidth3(){
        uint32_t i,result;

        for(i=0;i<600000;i++){
                if(!(PINC & (1<<US3_ECHO_POS)))
                continue;           //Line is still low
                else
                break;              //High edge detected
        }
        TCCR1A=0X00;
        TCCR1B=(1<<CS11);           // Resolution of the timer
        TCNT1=0x00;                 // Start counting time

        for(i=0;i<600000;i++)       // Almost 40 milliseconds
        {
                if(PINC & (1<<US3_ECHO_POS))
                {
                        if(TCNT1 > 60000)
                        break;    // No object in the range
                        else
                        continue;
                }
                else
                break;
        }

        if(i==600000)
        return US_NO_OBSTACLE;   //Time out
        result=TCNT1;               //Falling edge found
        TCCR1B=0x00;                //Stop Timer

        if(result > 60000)
        return US_NO_OBSTACLE;    //No obstacle
        else
        return (result>>1);
}

uint16_t GetPulseWidth4(){
        uint32_t i,result;

        for(i=0;i<600000;i++){
                if(!(PINC & (1<<US4_ECHO_POS)))
                continue;           //Line is still low
                else
                break;              //High edge detected
        }

        TCCR1A=0X00;
        TCCR1B=(1<<CS11);           // Resolution of the timer
        TCNT1=0x00;                 // Start counting time

        for(i=0;i<600000;i++){
                if(PINC & (1<<US4_ECHO_POS)){
                        if(TCNT1 > 60000)
                        break;    // No object in the range
                        else
                        continue;
                }
                else
                break;
        }
```

```
uint16_t GetPulseWidth2(){
        uint32_t i,result;

        for(i=0;i<600000;i++){
                if(!(PINB & (1<<US2_ECHO_POS)))
                continue;          //Line is still low
                else
                break;             //High edge detected
        }

        TCCR1A=0X00;
        TCCR1B=(1<<CS11);          // Resolution of the timer
        TCNT1=0x00;                // Start counting time
        for(i=0;i<600000;i++){
                if(PINB & (1<<US2_ECHO_POS)){
                        if(TCNT1 > 60000)
                        break;   // No object in the range
                        else
                        continue;
                }
                else
                break;
        }
        if(i==600000)
        return US_NO_OBSTACLE;          //Time out
        result=TCNT1;                   //Falling edge found
        TCCR1B=0x00;                    //Stop Timer

        if(result > 60000)
        return US_NO_OBSTACLE;          //No obstacle
        else
        return (result>>1);
}

#endif /* ULTRASONIC_H_ */
```

LCD.h

```
#ifndef LCD_H_INCLUDED
#define LCD_H_INCLUDED

#define RS PB0              // Register Select pin on PB0
#define EN PB1              // Enable pin on PB0

// Function Prototypes
void LCD_Init (void);
void LCD_Command(unsigned char);
void LCD_Print (char *);
void LCD_Clear();
void LCD_SetCursor(unsigned char, unsigned char);

void LCD_Init (void){
        DDRB = 0b11111111;
        _delay_ms(15);
        LCD_Command(0x02);        // 4 Bit Mode
        LCD_Command(0x28);        // 2 lines, 5x8 matrix,4-bit
mode
        LCD_Command(0x0c);        // Display on, cursor off
        LCD_Command(0x06);        // Shift the cursor to right
        LCD_Command(0x01);        // Clear the display
        _delay_ms(2);
}

void LCD_Command(unsigned char cmnd){
        PORTB = (PORTB & 0x0F) | (cmnd & 0xF0);
         // Send upper nibble
        PORTB &= ~(1<<RS);        // RS=0, command reg.
        PORTB |= (1<<EN);         // Enable=1
        _delay_us(1);
        PORTB &= ~(1<<EN);        // Enable=0
        _delay_us(200);
        PORTB = (PORTB & 0x0F) | (cmnd << 4);
        // Send lower nibble
        PORTB |= (1<<EN);
        _delay_us(1);
        PORTB &= ~ (1<<EN);
        _delay_ms(2);
}
```

```
void LCD_Print (char *str)
{
        int i;
        for(i=0; str[i]!=0; i++)
        {
                PORTB = (PORTB & 0x0F) | (str[i] & 0xF0);
                // Send upper nibble
                PORTB |= (1<<RS);         // RS=1, data reg.
                PORTB|= (1<<EN);          // Enable=1
                _delay_us(1);
                PORTB &= ~(1<<EN);        // Enable=0

                _delay_us(200);

                PORTB = (PORTB & 0x0F) | (str[i] << 4);
                // Send lower nibble
                PORTB |= (1<<EN);
                _delay_us(1);
                PORTB &= ~(1<<EN);
                _delay_ms(2);
        }
}

void LCD_Clear()
{
        LCD_Command (0x01);       // Clear the display
        _delay_ms(2);
        LCD_Command (0x80);
       // Force the cursor to the beginning of the 1st line
}

void LCD_SetCursor(unsigned char x, unsigned char y
{
        unsigned char adr[] = {0x80, 0xC0};
        LCD_Command(adr[y-1] + x-1);
        _delay_us(100);
}

#endif // LCD_H_INCLUDED
```

Main.c

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include "LCD.h"
#include "Ultrasonic.h"

int distance1, previous_distance1, percentage1, distance2,
previous_distance2, percentage2, distance3, previous_distance3,
distance4, previous_distance4;

int main()
{
	LCD_Init();		// Initializing the LCD

	char numberString1[4], numberString2[4];

	DDRD = 0xFF;

	while(1)
	{
		uint16_t r1,r2,r3,r4;

		_delay_ms(100);

		HCSR04Init();	// Initializing

		while(1)
		{
			HCSR04Trigger1();
			r1 = GetPulseWidth1();

			HCSR04Trigger2();
			r2 = GetPulseWidth2();

			distance1=(r1*0.034/2.0);
			percentage1= distance1*100/75;

			if (distance1 != previous_distance1)
			{
				LCD_Clear();
			}

			LCD_SetCursor(1, 1);
			LCD_Print("Bin 01:");
			LCD_SetCursor(9, 1);
			itoa(percentage1, numberString1, 10);
			LCD_Print(numberString1);
			LCD_SetCursor(12, 1);
			LCD_Print("%");

			previous_distance1 = distance1;
			_delay_ms(30);

			distance2=(r2*0.034/2.0);
			percentage2= distance2*100/75;

			if (distance2 != previous_distance2)
			{
				LCD_Clear();
			}
			LCD_SetCursor(1, 2);
			LCD_Print("Bin 02:");
			LCD_SetCursor(9, 2);
			itoa(percentage2, numberString2, 10);
			LCD_Print(numberString2);
			LCD_SetCursor(12, 2);
			LCD_Print("%");

			previous_distance2 = distance2;
			_delay_ms(30);

			HCSR04Trigger3();
			r3 = GetPulseWidth3();

			HCSR04Trigger4();
			r4 = GetPulseWidth4();

			distance3=(r3*0.035/2.0);

			previous_distance3 = distance3;
			_delay_ms(30);

			distance4=(r4*0.035/2.0);

			previous_distance4 = distance4;
			_delay_ms(30);
			}
		}
	}
}
```

Name of student: Rambukkamage R.D.S.T (204172L) [3] [6] [7] [18]

1. Servo motor
    - To rotate to open the lid.
2. Real-time clock
    - Track time

- Helped to create the PCB design.
- Help to create the animation using blender software.
- Helped to finalize the circuit diagram.

1. Servo motor

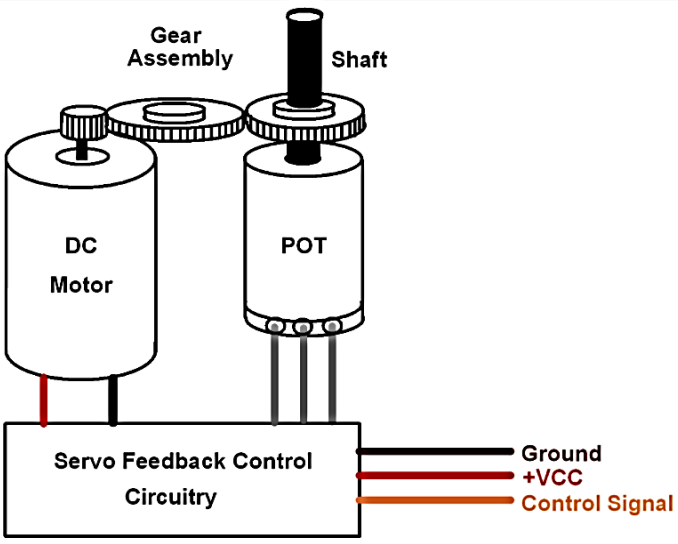


FIGURE B4.1: MECHANISM

- I guessed our lid weight will be 200-400g and the distance(r) will be 6 cm.
- I decided to choose a 3kg. cm standard servo motor. Because I have to think about the friction.
- They are small, powerful, easily programmable, and accurate. Most importantly, though, they allow for near-perfect repeatability of motion
- It has a rotation angle that varies from 0° to 180°.
- Torque = force * r
    - 3 kg → 1 cm
    - r → 6 cm
    - w = 500g

Specifications:

- Current: 1A
- Torque: 3kg.cm/41.74oz.in (4.8V)
- Voltage: 4.8V
- Dimensions: 40.8 × 20.1 × 38 mm
- Control system: Analog

Schematic Diagram:

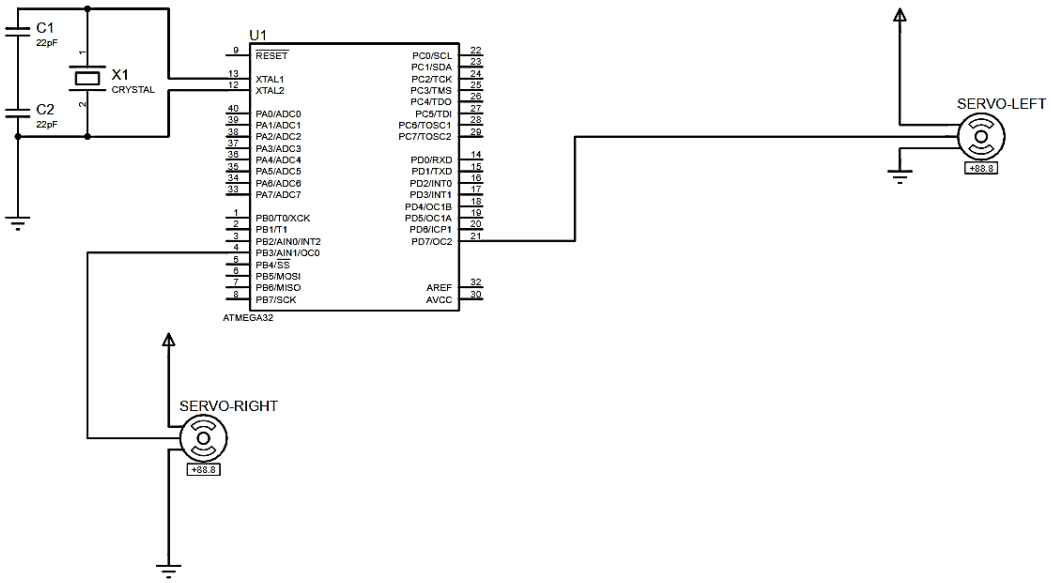


FIGURE B4.2: SCHEMATIC DIAGRAM

PCB Design:

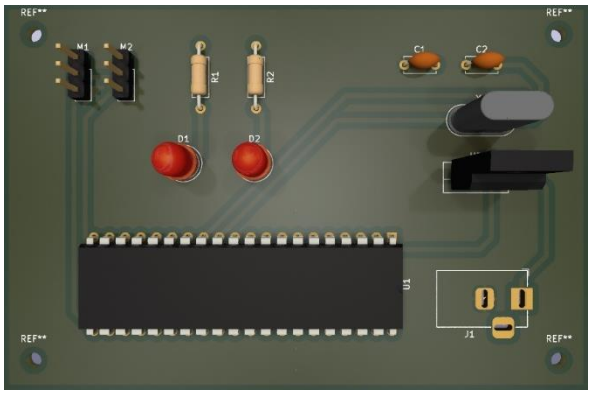


FIGURE B4.3: FRONT VIEW

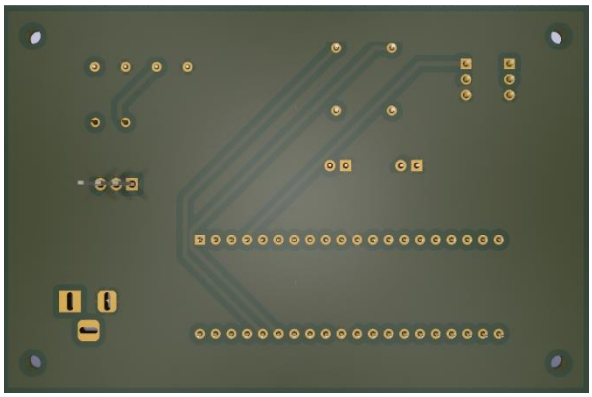


FIGURE B4.4: BACK VIEW

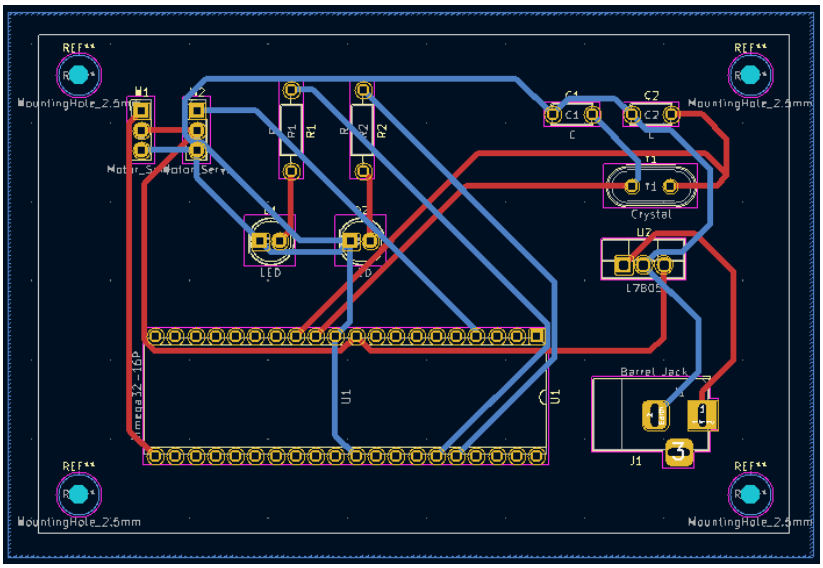


FIGURE B4.5: COMPONENT LAYER

Code:

```
#ifndef F_CPU
#define F_CPU 16000000UL
#endif

#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
        DDRB = 0xff;
        PORTB= 0x00;
        DDRD = 0xff;
        PORTD= 0x00;

  while (1)
  {
                PORTB = 0x08;
                _delay_us(1000);
                PORTB = 0x00;
                _delay_ms(2000);
                PORTB = 0x08;
                _delay_us(2500);
                PORTB = 0x00;
                _delay_ms(2000);

                PORTD= 0x80;
                _delay_us(1000);
                PORTD= 0x00;
                _delay_ms(2000);

                PORTD  = 0x80;
                _delay_us(2500);
                PORTD  = 0x00;
                _delay_ms(2000);
  }
}
```

2. Real-time clock

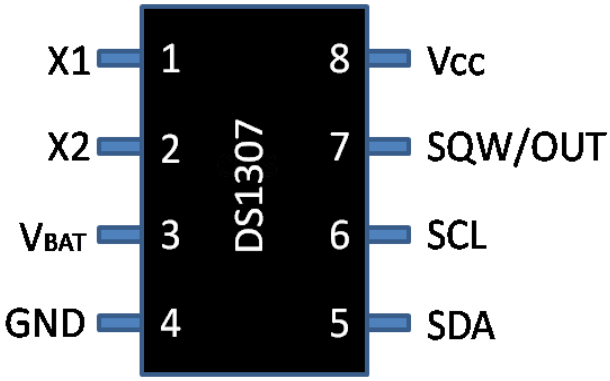


FIGURE B4.6: PIN DIAGRAM OF RTC

Specifications:

- RTC module: DS1307
- Battery model: CR1220 button cell
- Operating Voltage: 5V
- Unit information: Second, Minute, Date, Week, Month and Year
- Operating temperature: -10°C ~ +85°C

Schematic Diagram:

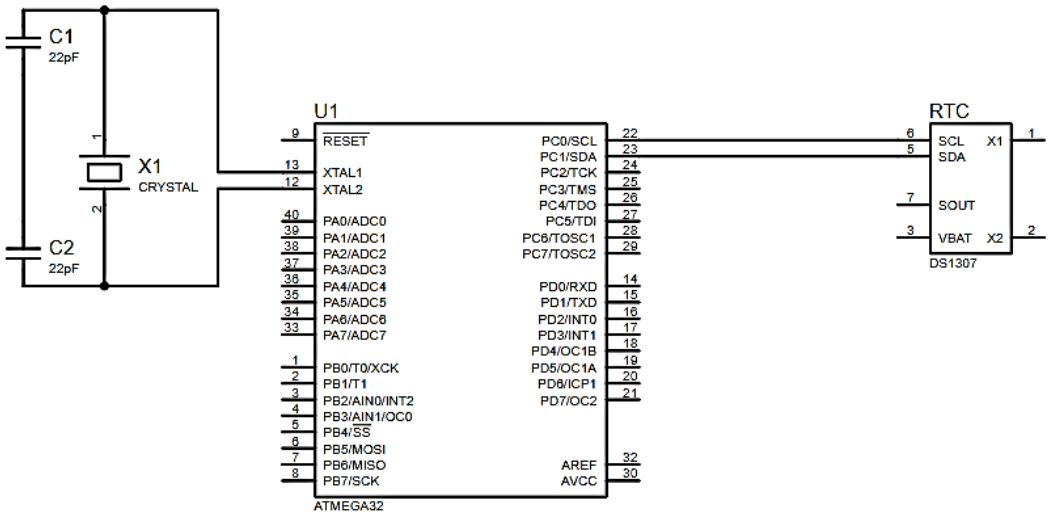


FIGURE B4.7: SCHEMATIC DIAGRAM

Codes:

I2C Master.h [RTC]

```
#ifndef I2C_MASTER_H_FILE_H_

#define I2C_MASTER_H_FILE_H_

#define F_CPU 16000000UL

#include <avr/io.h>

#include <util/delay.h>

#include <math.h>

#define SCL_CLK 100000L

#define BITRATE(TWSR)    ((F_CPU/SCL_CLK)-
16)/(2*pow(4,(TWSR&((1<<TWPS0)|(1<<TWPS1)))))

void I2C_Init();

uint8_t  I2C_Start(char write_address);
uint8_t  I2C_Repeated_Start(char read_address);
void I2C_Stop();

void I2C_Start_Wait(char write_address);
uint8_t  I2C_Write(char data);

int I2C_Read_Ack();
```

I2C Master.c [RTC]

```
#include <avr/io.h>
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include "I2C_Master_H_file.h"
#include <avr/sleep.h>
#define Device_Write_address        0xD0
#define Device_Read_address         0xD1
#define TimeFormat12                0x40
#define AMPM                        0x20
int second,minute,hour,day,date,month,year;
bool IsItPM(char hour_){
        if(hour_ & (AMPM))
        return 1;
        else
        return 0;
}
void RTC_Read_Clock(char read_clock_address){
        I2C_Start(Device_Write_address);
        I2C_Write(read_clock_address);
        I2C_Repeated_Start(Device_Read_address);
        second = I2C_Read_Ack();
        minute = I2C_Read_Ack();
        hour = I2C_Read_Nack();
        I2C_Stop();
}
void RTC_Read_Calendar(char read_calendar_address){
        I2C_Start(Device_Write_address);
```

```
int I2C_Read_Nack();

#endif
```

```
        I2C_Write(read_calendar_address);
        I2C_Repeated_Start(Device_Read_address);
        day = I2C_Read_Ack();
        date = I2C_Read_Ack();
        month = I2C_Read_Ack();
        year = I2C_Read_Nack();
        I2C_Stop();
}
int main(void){
        I2C_Init();
  while(1) {
                RTC_Read_Clock(0);
                if (hour & TimeFormat12) {
                        if(hour=9 & AMPM){
                        sleep_disable();
                        }if(hour=12 & AMPM){
                        sleep_disable();
                        }
                        if (hour=10 & AMPM){
                                sleep_enable();
                        }
                }
  }
}
```

PCB Design:

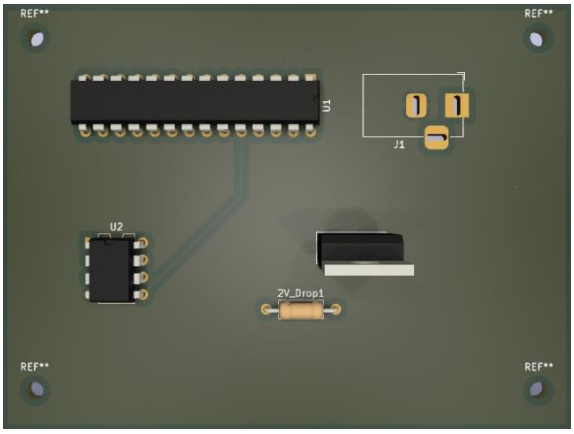


FIGURE B4.8: FRONT VIEW

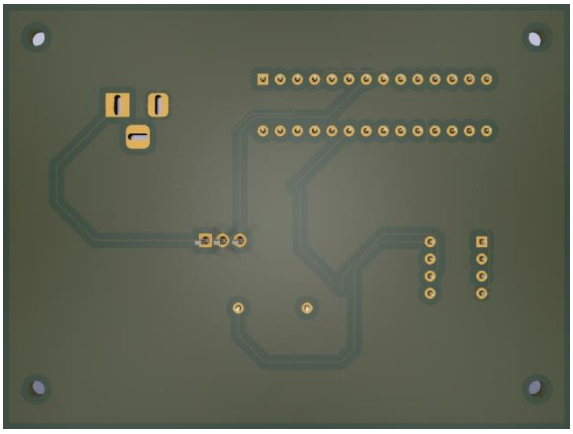


FIGURE B4.9: BACK VIEW

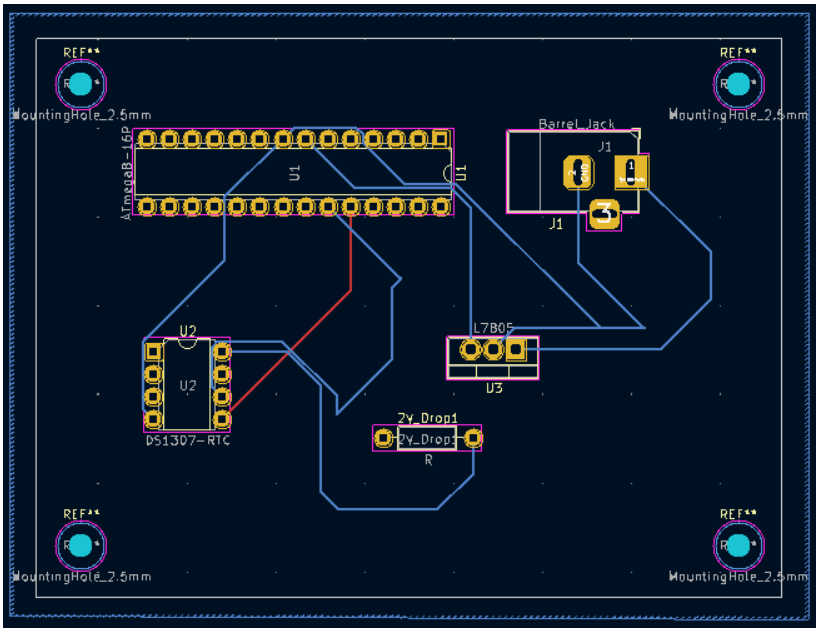


FIGURE B4.10: COMPONENT LAYER

Name of Student: Samararathna B.S (204184B) [3] [6] [7] [18]

1. GSM module
    - Send a SMS to Receiver.
2. Power Supply Unit
    - Distributing power to all components.
3. Communication between 2 atmega32 microcontrollers

1. GSM Module

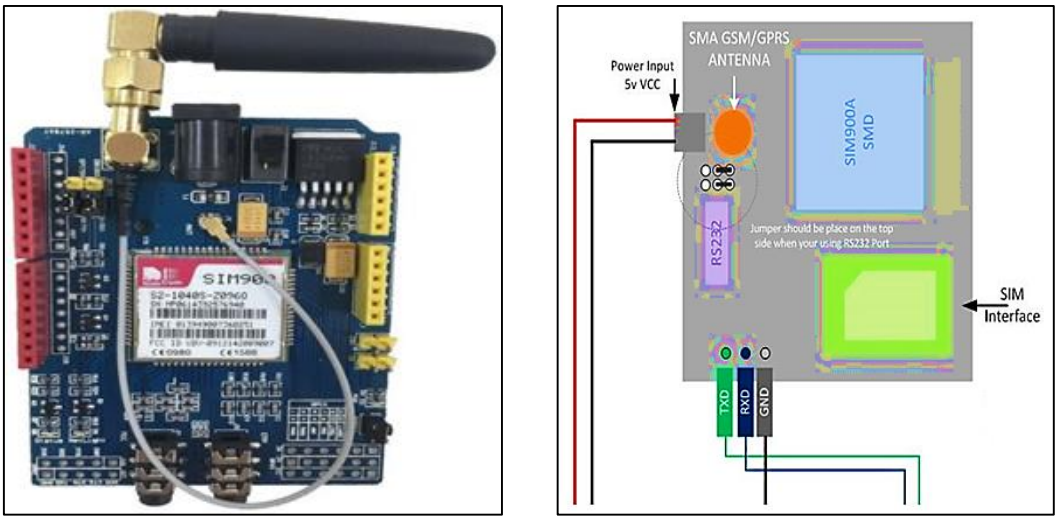


FIGURE B5.1: GSM MODULE

Specifications:

- ❖ Single supply voltage: 3.4V – 4.5V
- ❖ Uses a 900 and 1800MHz frequency band
- ❖ Use USART for integration with Atmega32
- ❖ Supports UART interface
- ❖ Search the two frequency bands automatically

- The SIM900A is a complete Dual-band GSM / GPRS solution. Baud rate is configurable from 9600-115200 through AT command. “Number of bits per second (bps)” also known as Baud rate in Binary systems.
- A USART, means a Universal Synchronous/Asynchronous Receiver/Transmitter - is a microcontroller peripheral that converts incoming and outgoing bytes of data into a serial bit stream.

- AVR atmega has flexible USART, which can be used for serial communication with serial GSM 900A. To establish USART interface you have to connect RXD of module to TXD of Atmega32 and TXD is connected to RXD of Atmega32.
- Using this USART, In Serial data framing While sending and receiving data, some bits are added for the purpose of knowing the beginning/ending of data

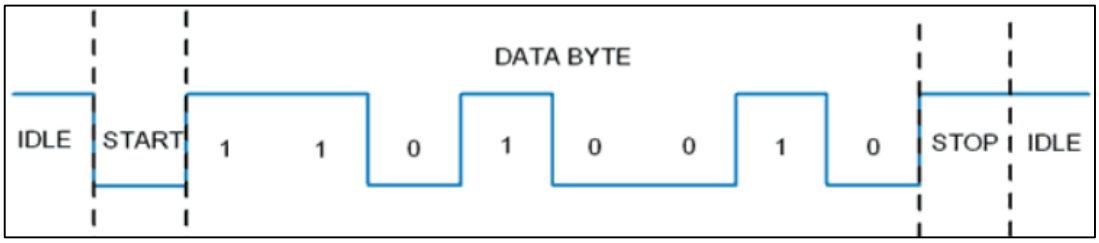


FIGURE B5.2: BASIC FRAME STRUCTURE

Code:

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdlib.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include <string.h>
#include <stdbool.h>
#include "USART.h"
void USART_Transmit(char data );
void senddata(char string[16]);
void USART_Init();
void delay_ms(unsigned int de);
void sendSMS();

int main(){
DDRC = 0x01;
sei();

while (1)
{
sendSMS();

}
}
void sendSMS()
{
senddata("AT+CMGD=1");
USART_Transmit(13);
USART_Transmit(10);
delay_ms(1000);

senddata("AT+CMGF=1");
USART_Transmit(13);
USART_Transmit(10);
delay_ms(1000);

senddata("AT+CMGW=");
USART_Transmit(34);
senddata("+919812345678");  //Enter Mobile number
USART_Transmit(34);
USART_Transmit(13);
USART_Transmit(10);
delay_ms(1000);

senddata("Robot is here");
USART_Transmit(13);
USART_Transmit(10);
delay_ms(1000);

USART_Transmit(26); //Cntrl+Z
delay_ms(1000);
delay_ms(1000);

senddata("AT+CMSS=1");
USART_Transmit(13);
USART_Transmit(10);
delay_ms(1000);
}
```

Schematic Diagram:

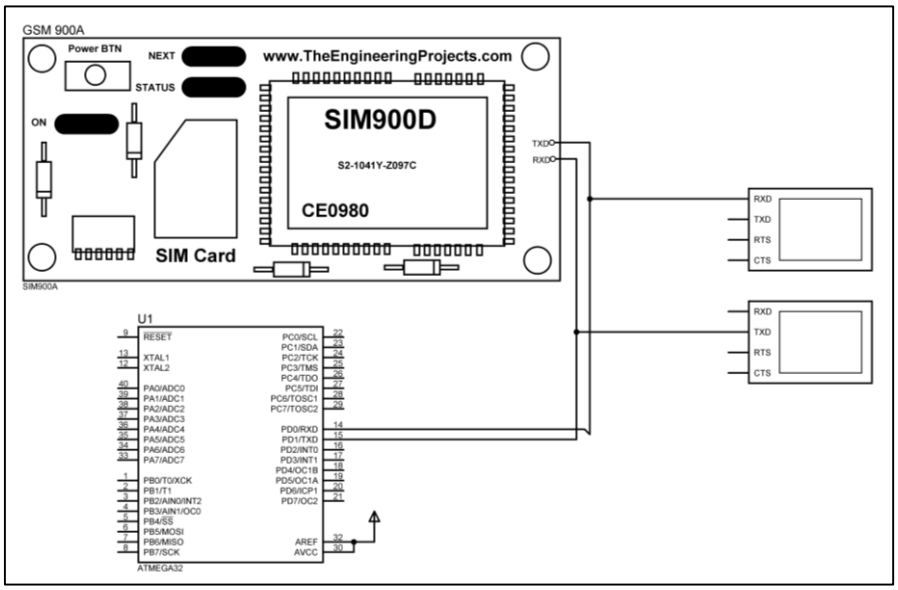


FIGURE B5.3: SCHEMATIC DIAGRAM

2. Power Supply Unit

Specifications:

- Battery Supply Voltage is 14.8 V
- Used 2 x 5V voltage down regulators and 12V regulator

Schematic Diagram:

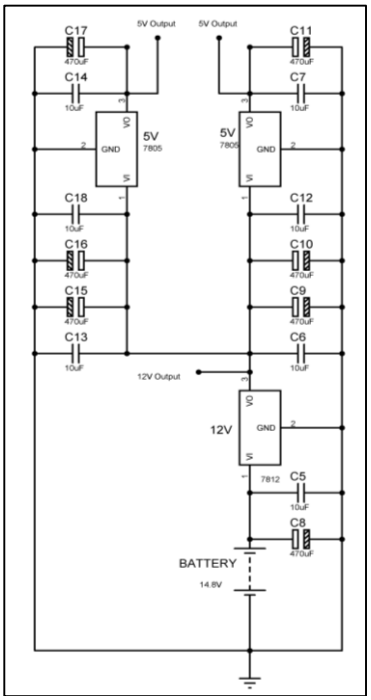


FIGURE B5.4: SCHEMATIC DIAGRAM

3. Communication between 2 microcontrollers

- Used CD4052 4 - Channel Multiplexer

Schematic Diagram:

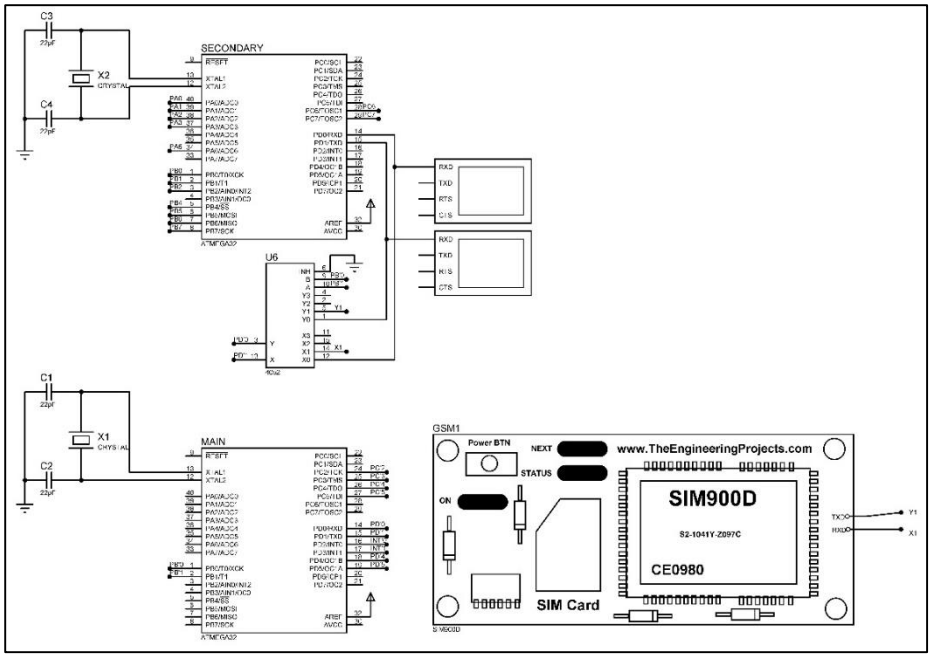


FIGURE B5.5: SCHEMATIC DIAGRAM

Code:

```
#define F_CPU 16000000UL
#ifndef USART_H_
#define USART_H_

#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

void USART_init(unsigned long USART_BAUDRATE)          /* USART initialize function */
{
UCSRB |= (1 << RXEN) | (1 << TXEN) | (1 << RXCIE);     /* Enable USART transmitter and receiver */
UCSRC |= (1 << URSEL)| (1 << UCSZ0) | (1 << UCSZ1);    /* Write USCRC for 8 bit data and 1 stop bit */
UBRRL = BAUD_PRESCALE;                                 /* Load UBRRL with lower 8 bit of pre scale value */
UBRRH = (BAUD_PRESCALE >> 8);                          /* Load UBRRH with upper 8 bit of pre scale value */
}

unsigned char USART_RxChar()
{
while ((UCSRA & (1 << RXC)) == 0);                     /*Do nothing until data have been received*/
return(UDR);                                           /* return the byte*/
}

void USART_TxChar(char ch)
{
while (! (UCSRA & (1<<UDRE)));                         /*Wait for empty transmit buffer*/
UDR = ch ;
}

void USART_SendString(char *str)
{
unsigned char j=0;
```

```
while (str[j]!=0)                                   /*send string up to null */
{
USART_TxChar(str[j]);
j++;
}
}

#endif /* USART_H_ */
```

PCB Design:

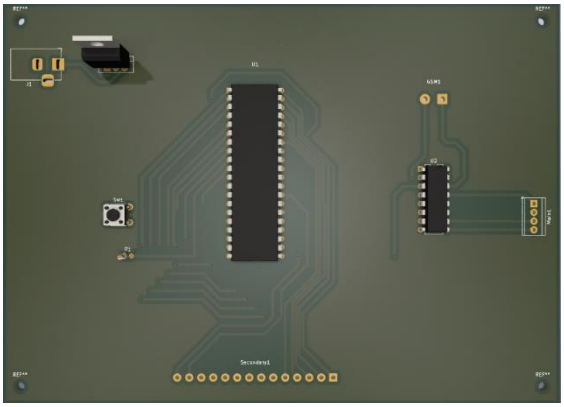


FIGURE B5.6: FRONT VIEW

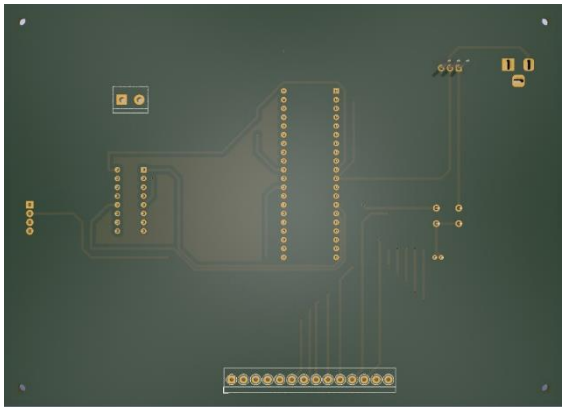


FIGURE B5.7: BACK VIEW

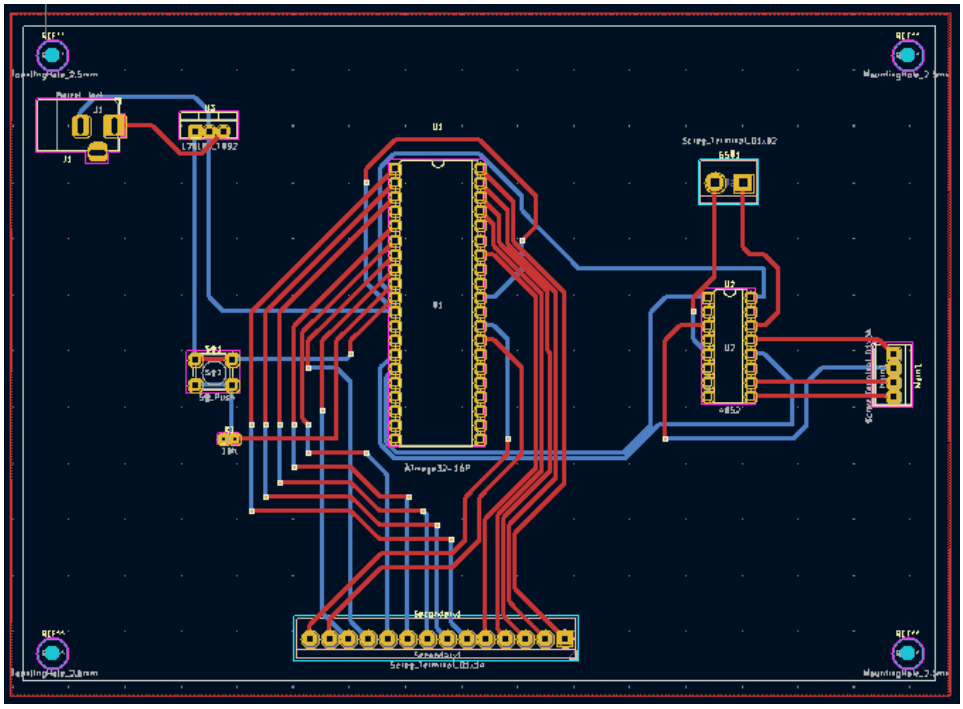


FIGURE B5.8: COMPONENT LAYER