Faculty of Information Technology

IN – 1900 ICT Project

CAN CRUSHER

Group No: 10

Index Number	Name
204092P	**KANAKKAHEWA Y.B.**
204140M	NISHATH M.N.H.A.
204186H	SANCHITHA S.P.Y.P.
204020V	BASNAYAKE S.B.
204098N	KAVINDA K.A.R.

Supervisor's Name: Mr. B.H. Sudantha

Dean/Senior Lecturer

Faculty of Information Technology

University of Moratuwa

Signature of Supervisor: Date of Submission: 08/06/2022

Table of Contents

List of Tables

1.0 Introduction

In the last few decades, technology has progressed at a staggering rate. Smartphones, the internet, cloud computing, and hundreds of other inventions are changing every face of our lives. Communication, business, day to day life, government, travel, fundraising, and even agriculture have been affected. As because of this evolution of technology the human kind has achieved so many things as well as they have caused a lot of problems and disasters too. **Environmental Pollution** is one of those major disasters. Due to the improper disposal ways of non-biodegradable garbage such as plastic and metal materials have caused many problems when it comes to environmental pollution. In order to prevent this situation, the 3R concepts of Reduce, Recycle and Reuse has been introduced. This method is being mainly used worldwide for the plastic components such as plastic bottles, cups, containers etc. But when considering the metallic materials specially the **Alluminum** materials which people use in their day-to-day life, such as beer cans, energy drink cans and containers this 3R concept is not widely used and applied. As a result of that the aluminum disposal has been done in an improper way for ages for many reasons. So, the environmental pollution has now become a severe issue that the people have been tempted to rethink about it.

2.0 Literature Survey

This chapter gives a brief description of similar projects that are designed to help recycle Alluminium cans. And in common tongue these types of machines are known as Reverse Vending Machines.

(1) Pepsi Reverse Vending Machine

This machine has created through a partnership between Pepsi, Waste Management and Keep America Beautiful and manufactured by GreenOps. In order to use the machine people, have to scan the barcode on their PET plastic bottle or Aluminium can. Then the machine calculates how many points to be awarded for each item and a user can redeem his/her points for a prize on the host venue or at onGreenopolis.com

(2) Tomra Reverse Vending Machines

Tomra is a Norwegian Multinational Cooperation specializing themselves in manufacturing collection and sorting products. They have created wide range of Reverse Vending Machines depending on the scale.

Mini Line Reverse Vending Machines- This is their suggested reverse vending machines for small stores with small spaces.

Eg :- H10 & H11

Standalone Line Reverse Vending Machines – These are their Reverse Vending Machines with low to medium storage capacity. And much suitable for small and medium stores with much space.

Eg:- Tomra S1 Rugged, Tomra T70 Single & Dual ,Tomra T70 TriSort, Tomra T90

Flexible Line Reverse Vending Machines – There machines are tailor made to fit the customers' requirements such as collecting items, storage capacity and the store's layout.

Eg:- Tomra T9 with EasyPac & Tomra T9 with MultiPac

Revolution Line Reverse Vending Machines– In these systems' customers can insert all of their containers at once.

Eg:- Tomra R1 & T9 with MultiPac Air

Expert Line Reverse Vending Machines– This is their solution for industrial level collecting and sorting. They suggest this system for beverage wholesalers, Logistic centers, System Operators, Bottle depots, Redemption Centers, Industrial facilities.

Eg:- TOMRA E1 Rücknahmezentrum

Even though, they manufacture different types of Reverse Vending Machines their technology is same in every machine. In these machines they differentiate containers by their barcode and the rewarding facility normally do depends on the owner. But mostly owners give coupons to be used in their stores.

(3) RVM System's Reverse Vending Machines

RVM System is a Sweden based company specialized in manufacturing Reverse Vending Machines.

Stand Alone Reverse Vending Machines – These machines are designed to fit the needs of small stores or supermarkets. According to the manufactures these machines takes less space, easy to install, clean and maintain. They can scan between 50 to 60 containers per minute. And only sorts Aluminium cans, PET plastic bottles and glass containers.

Eg:- RVMX2,RVMX3, RVMX10, RVMX20 ,RVMX30, RVMX200

Large Capacity Modular Reverse Vending Machines – These Machines are designed to fit the needs of stores and supermarkets with high volumes of recyclable containers. These machines can sort maximum of 60 containers for minute and only takes Alluminium cans, PET plastic containers and glass bottles.

Eg:- RVM X Proline, RVM X Proline DUO, RVM X Proline Slim

Mega/Bulk Reverse Vending Machines

RVM Mega Proline Plus- This machine can process up to 100 containers a minute. And sorts PET plastic bottles, Aluminium cans and glass bottles. And according to the manufacturer this machine is also accessible for wheelchair and mobility scooter users.

In these machines the manufacturer sorts PET plastic bottles through a optical PET sensor and glass and Alluminium cans through scanning their barcode. And for the rewarding methods coupons or loyalty cards were used.

3.0 Aim & Objectives

3.1 Aim

- Develop an automated machine for dispose cans (ex: beer cans, soft drink cans) in an environmentally friendly method.

3.2 Objectives

- Protecting ecosystems and wildlife
- Influence people, not to throw cans everywhere by using a rewarding can collecting method
- Saving energy

  (Producing aluminium from old products (old cans) uses less energy than making it from the scratch)
- Conserving natural resources by reducing risky, expensive and damaging mining and extraction of new metal ores
- Maintaining the beauty of the environment

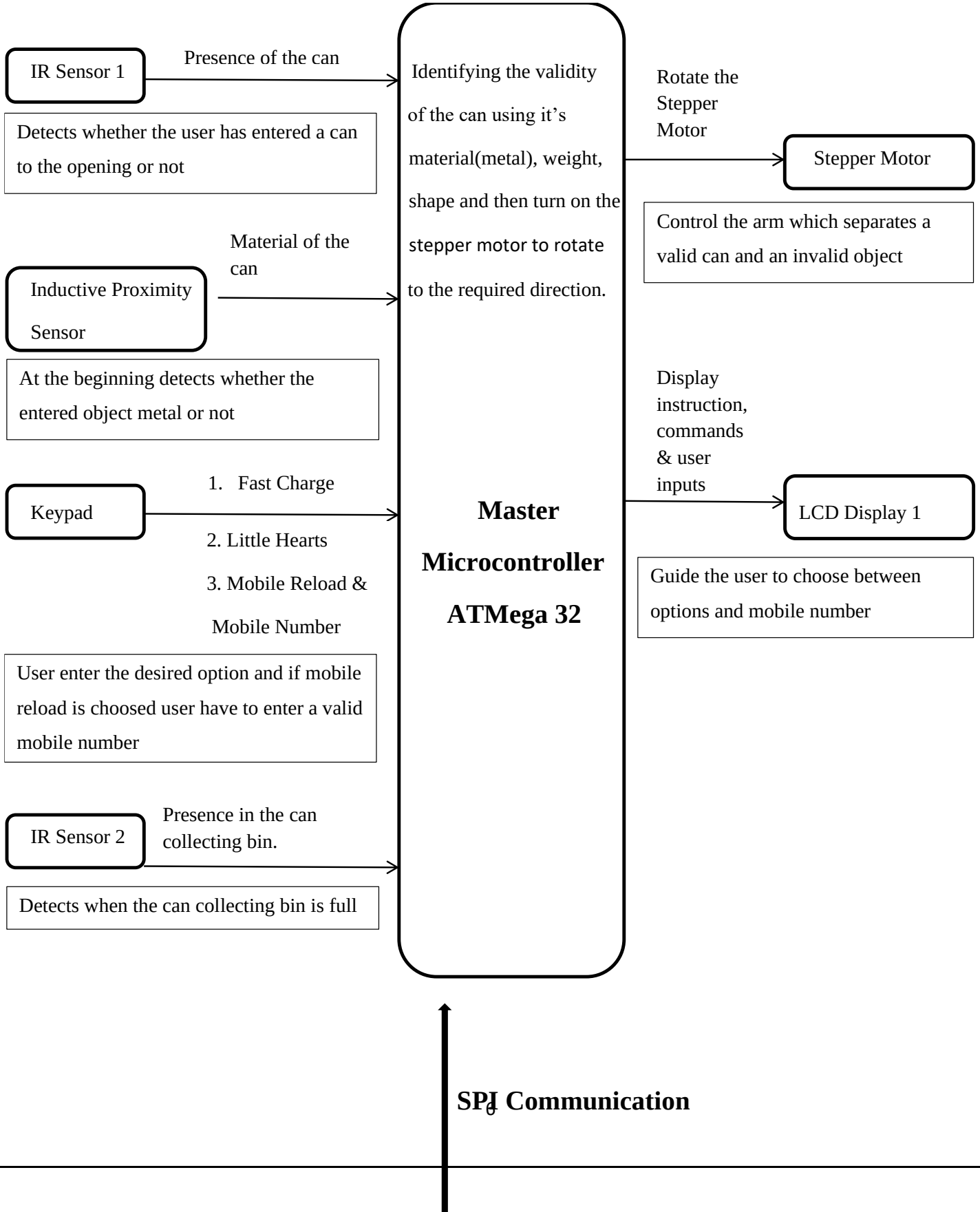
4.0 Analysis and Design

So, In order to encourage recycling of Aluminium cans we propose an automated machine.

This invention collects used Aluminium cans and stores them in a proper way. At the first place we have used a servo motor to control the amount of cans entering at a time. Here the machine is designed and programmed in a manner to identify whether the user's input is a proper Aluminium can or not. The validity of the Can will be checked with the help of the Inductive Proximity Sensor, weight sensor and the Ultrasonic sensor. We have used the HX711 load cell amplifier to control the weight sensor and HC-SR04 Ultrasonic sensor to determine the shape. If user's input is a proper Can, the arm connected to the stepper motor will direct the Can to the crushing unit then the machine will crush and store it using the induction motor in the designated area. If not, the machine will return it back to the user. We have used L293d Driver motor driver in order to control the stepper motor.

The user will be rewarded according to his/her order of preference. The machine will give an option to the user through the 16*2 LCD display to choose between a mobile reload or a donate the value to Little Hearts foundation using SIM 900A gsm module or get a timely fast charging facility according to user's preference.

Overall Block (Input – Output) Diagram



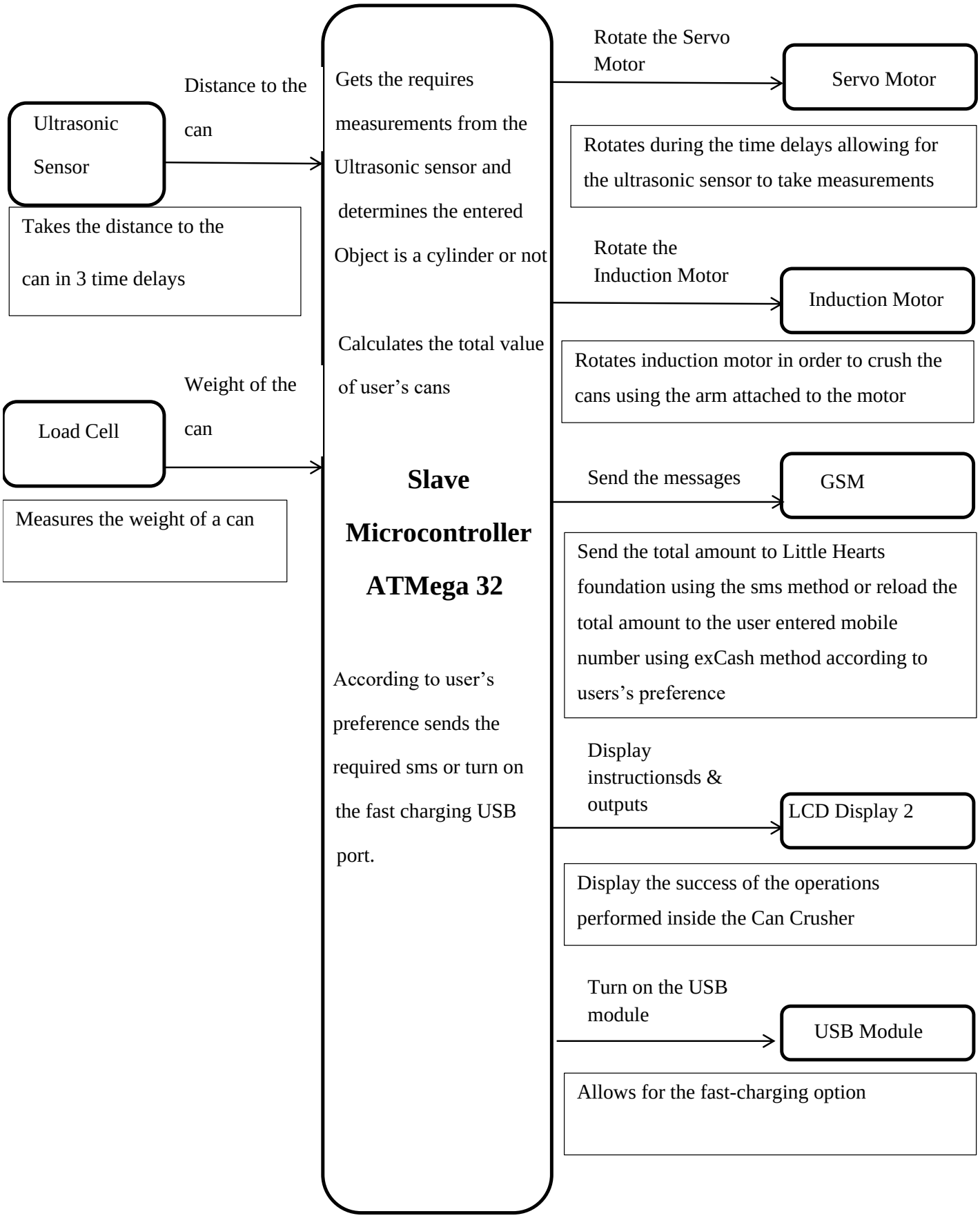

Ultrasonic Sensor
Distance to the can
Takes the distance to the can in 3 time delays
Load Cell
Weight of the can
Measures the weight of a can
Gets the requires measurements from the Ultrasonic sensor and determines the entered Object is a cylinder or not
Calculates the total value of user's cans
Slave Microcontroller ATMega 32
According to user's preference sends the required sms or turn on the fast charging USB port.
Rotate the Servo Motor
Servo Motor
Rotates during the time delays allowing for the ultrasonic sensor to take measurements
Rotate the Induction Motor
Induction Motor
Rotates induction motor in order to crush the cans using the arm attached to the motor
Send the messages
GSM
Send the total amount to Little Hearts foundation using the sms method or reload the total amount to the user entered mobile number using exCash method according to users's preference
Display instructionsds & outputs
LCD Display 2
Display the success of the operations performed inside the Can Crusher
Turn on the USB module
USB Module
Allows for the fast-charging option

Figure 1

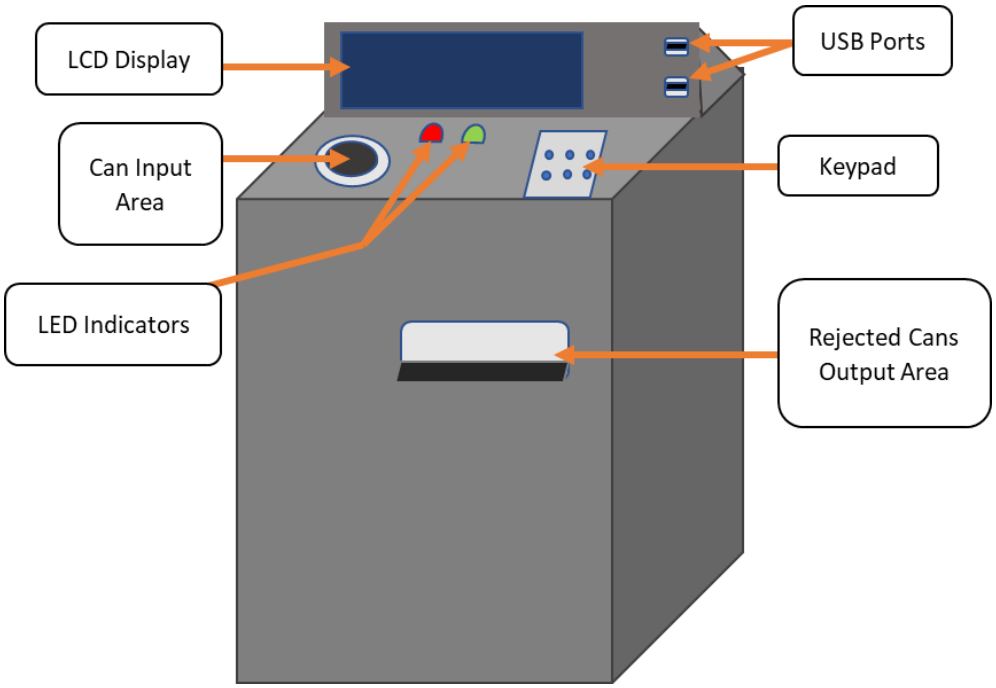


Figure 2

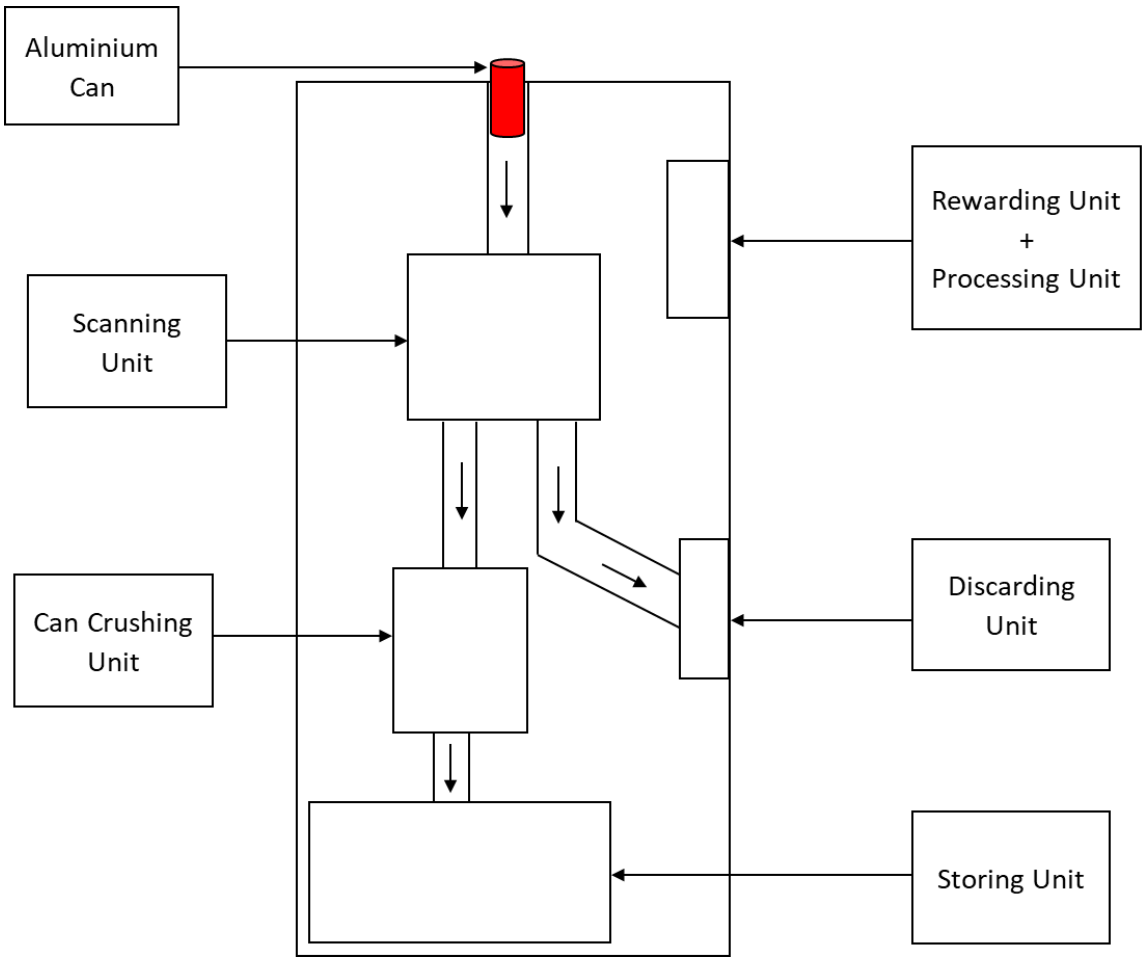


Figure 3

5.0 Testing and Implementation

We have finished the coding parts for all the components. We used Atmel Studio to write programs and all completed programs were tested using Proteus Software and used Keycad to design the PCB's.

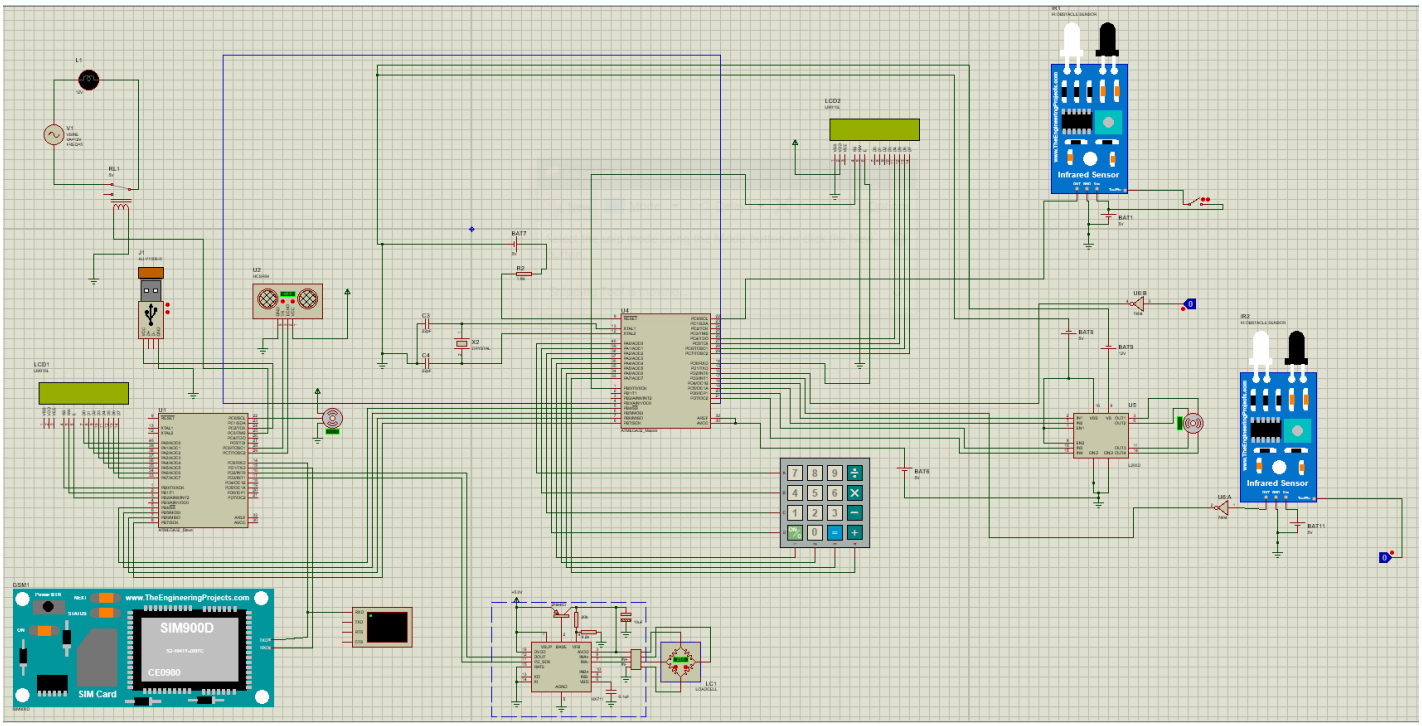
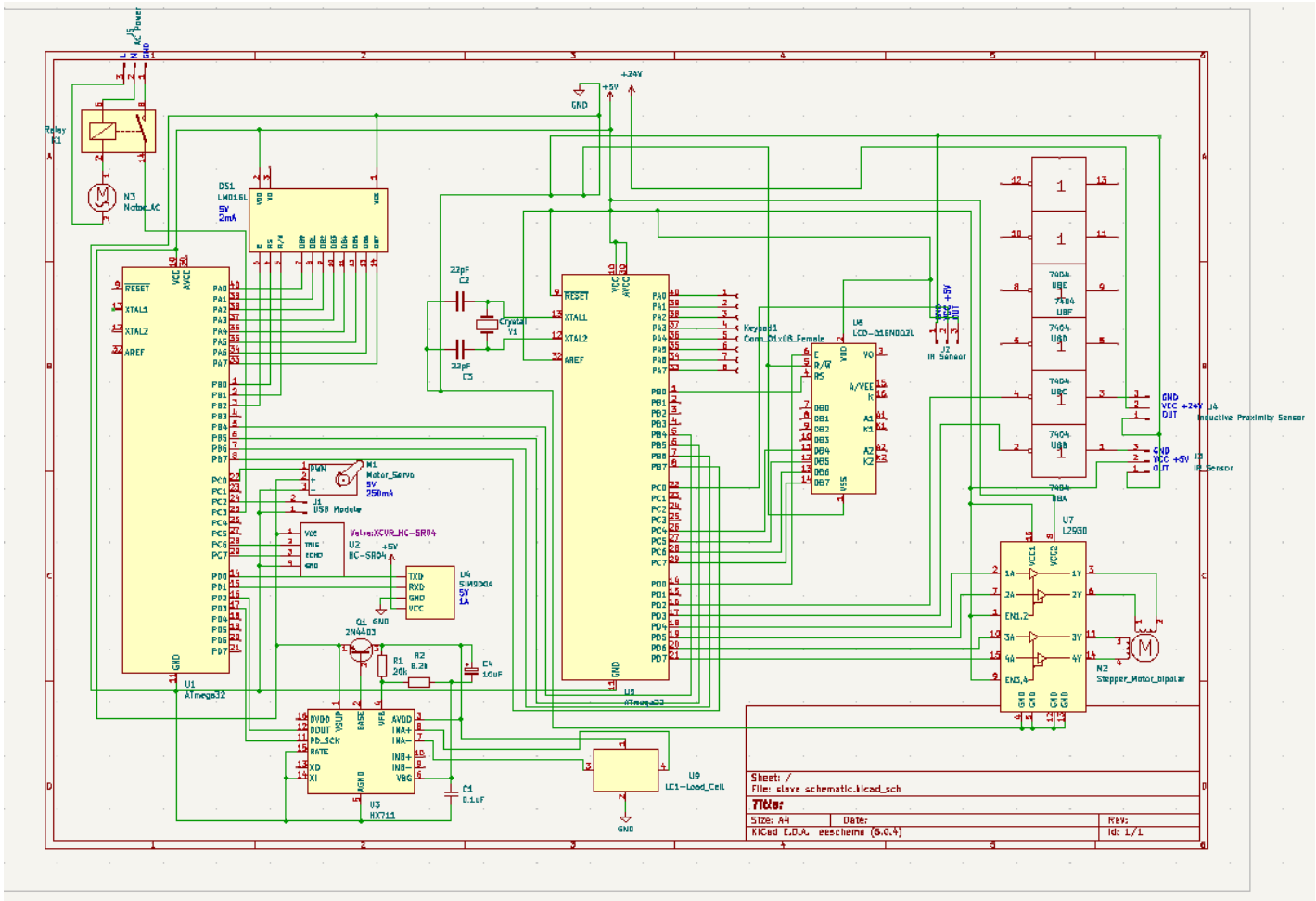
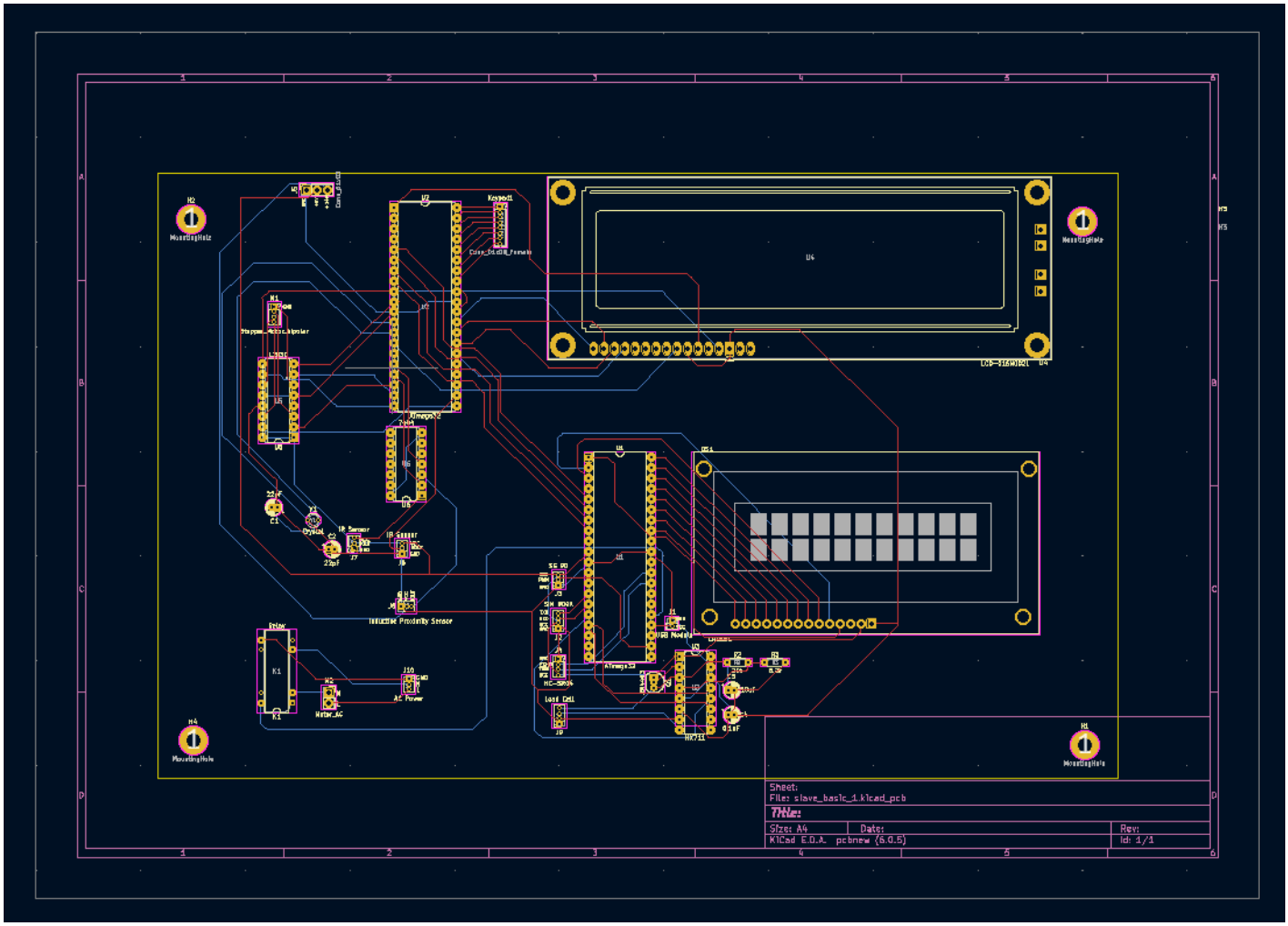


Figure 4 Proteus full circuit Diagram

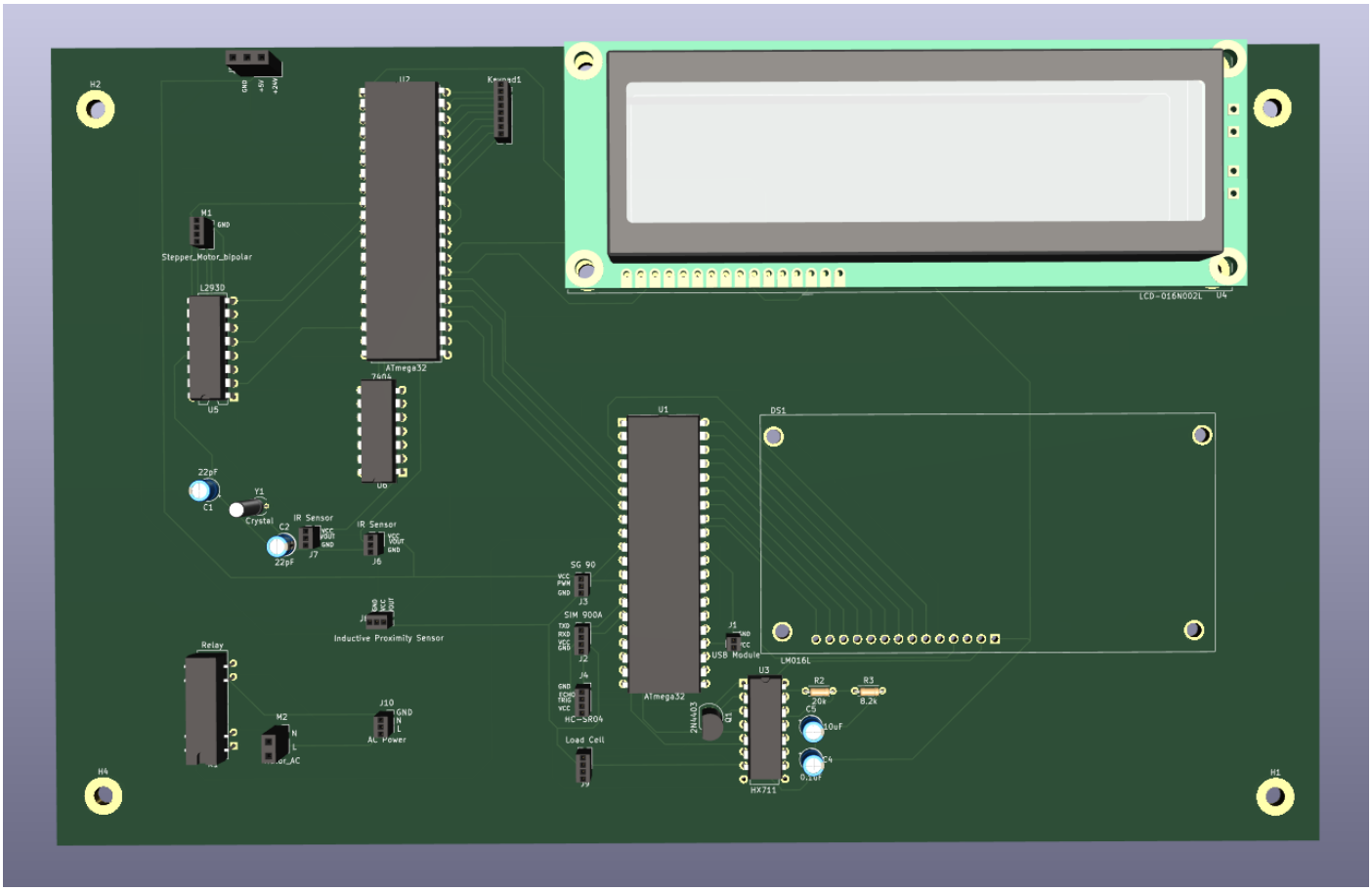
Schematic Diagram



PCB editor


Sheet:
File: slave_basic_1.kicad_pcb
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. pcbnew (6.0.5)
Id: 1/1

3D viewer


H2
H4
H1
M1
Stepper_Motor_bipolar
L2930
U5
ATmega32
74HC
U6
U7
Keypad1
LCD-016N002L
U4
22pF
C1
Y1
Crystal
C2
22pF
IR Sensor
J7
IR Sensor
J6
SG 90
J3
SIM 900A
J2
Inductive Proximity Sensor
J4
HC-SR04
Load Cell
J9
Relay
M2
J10
AC Power
U1
ATmega32
J1
USB Module
DS1
LM016L
U3
HX711
2N4403
Q1
R2
20k
R3
8.2k
C3
10uF
C4
0.1uF

6.0 Estimated Cost & Expenditure

Table 1

Name	Unit Price (Rs.)	Quantity	Total Price (Rs.)
1602 16x2 Blue Backlight LCD Display	800	2	1600
4x4 16-Key Membrane Switch Keypad Module	200	1	200
Load Cell	610.00	1	610
Load cell 10g + hx711	1,110.00	1	1,110
Analog Servo Motor Metal Wheel ES08MA-II Full Set	1440	2	2880
A6 GSM GPRS Quad-band Module with Antenna	2380	1	2380
3V Mini Buzzer	60	1	60
Atmega 32 microcontroller	2020	2	2020
Line Hunting Sensor Module with Preset 3-pin TCRT5000 (IR Sensor)	280	2	560
LJ18A3-8-Z/AX 8mm NPN NC DC Inductive Proximity Sensor	1944	1	1944
Nema 17 1.8° 42mm 0.44N.M 1.7A Stepper Motor	4432	1	4432
HC-SR04 Ultrasonic Sensor	400	1	400
Induction Motor	15000	1	15000
TOTAL			33196

7.0 References

[1] [Online]. Available: https://www.electronicwings.com.

[2] [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-weight-measurement-using-load-cell.

[3] [Online]. Available: https://circuitdigest.com/microcontroller-projects/arduino-weight-measurement-using-load-cell.

[4] [Online]. Available: https://dronebotworkshop.com/servo-motors-with-arduino/.

[5] [Online]. Available: https://www.electroniclinic.com/proximity-sensors-inductive-and-capacitive-proximity-sensors-with-arduino/.

[6] [Online]. Available: https://www.circuitstoday.com/interface-gsm-module-with-arduino.

[7] [Online]. Available: https://www.circuitstoday.com/interface-gsm-module-with-arduino.

[8] [Online]. Available: https://robu.in/product/ir-infrared-obstacle-avoidance-sensor-module/.

[9] [Online]. Available: http://davidegironi.blogspot.com/2019/04/hx711-load-cell-amplifier-driver-for.html.

[10] [Online]. Available: https://inhabitat.com/new-reverse-vending-machine-pays-you-to-recycle/.

[11] [Online]. Available: https://www.electronicwings.com/avr-atmega/atmega1632-usart.

[12] [Online]. Available: https://www.electronicwings.com/avr-atmega/servo-motor-interfacing-with-atmega16.

[13] [Online]. Available: https://www.electronicwings.com/avr-atmega/sim900a-gsm-module-interfacing-with-atmega1632-.

Further Work

If we are further developing this machine we can,

- Develop this machine to store plastic & glass cans/bottles.
- Offer a supermarket coupon or a voucher to the user as the reward.

Appendix A – Individual Contribution

1. Student name-: Kanakkahewa Y.B – 204092P

Learning & interfacing the,

- Induction motor
- Inductive Proximity Sensor
- USB Ports
- SPI communication

With and between the atmega32 microcontroller.

Inductive Proximity Sensor (LJ18A3-8-Z/AX)

In our project we use an Inductive Proximity Sensor to see whether the users can (Input) is metal or not as we are checking the physical attributes.

Proximity Sensors are used to detect objects (Ferrous and Non-Ferrous)

There are 3 types of proximity sensors

Inductive (Detect Metallic objects)

Capacitive (Detect objects)

IR (Read distances)

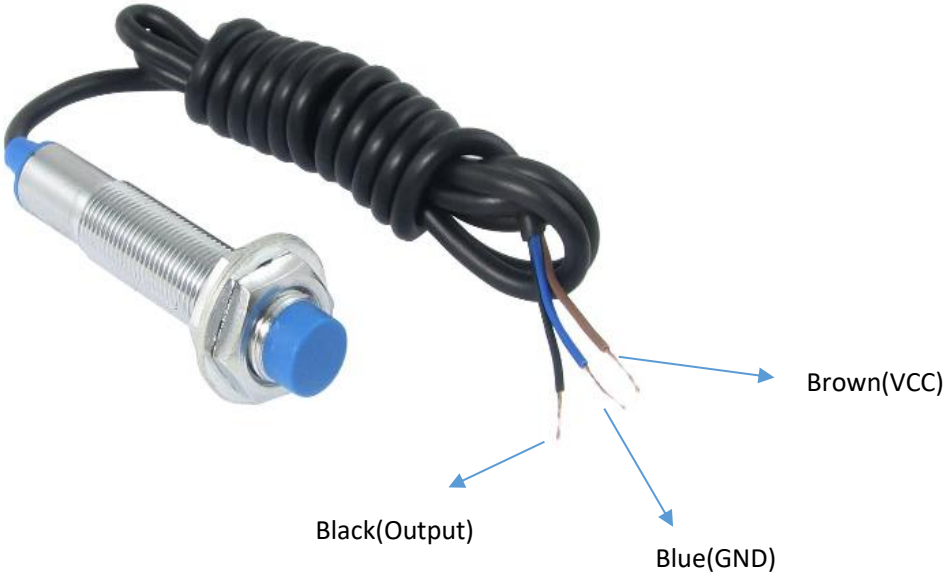
Here we use the Inductive Proximity Sensor to check the users input whether it is metal or not.

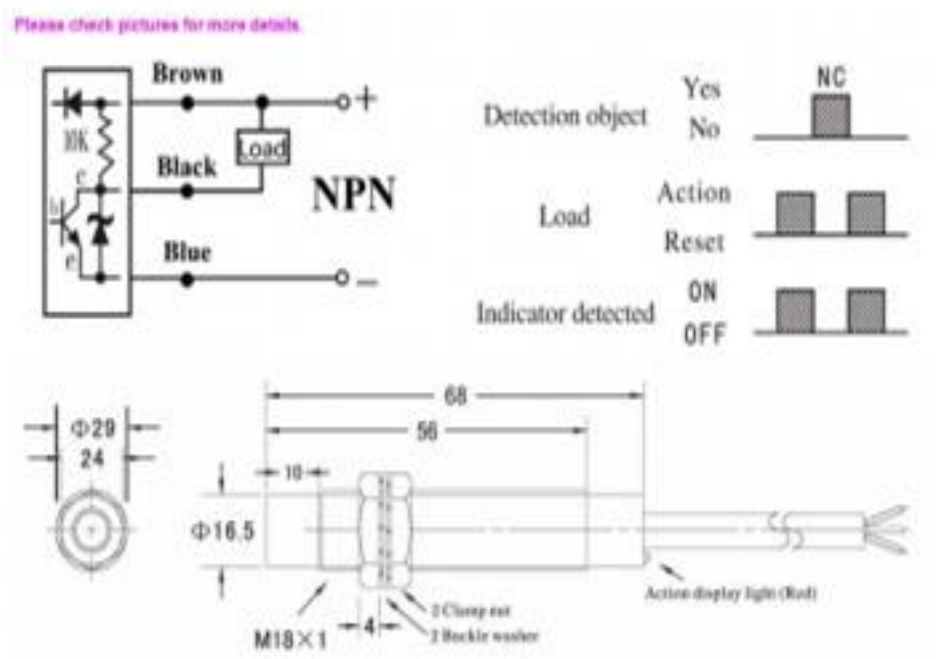
There are also two types of Inductive Proximity sensors as,

PNP

NPN

Here we use the NPN Inductive Proximity Sensor, and it has three wires to connect within.



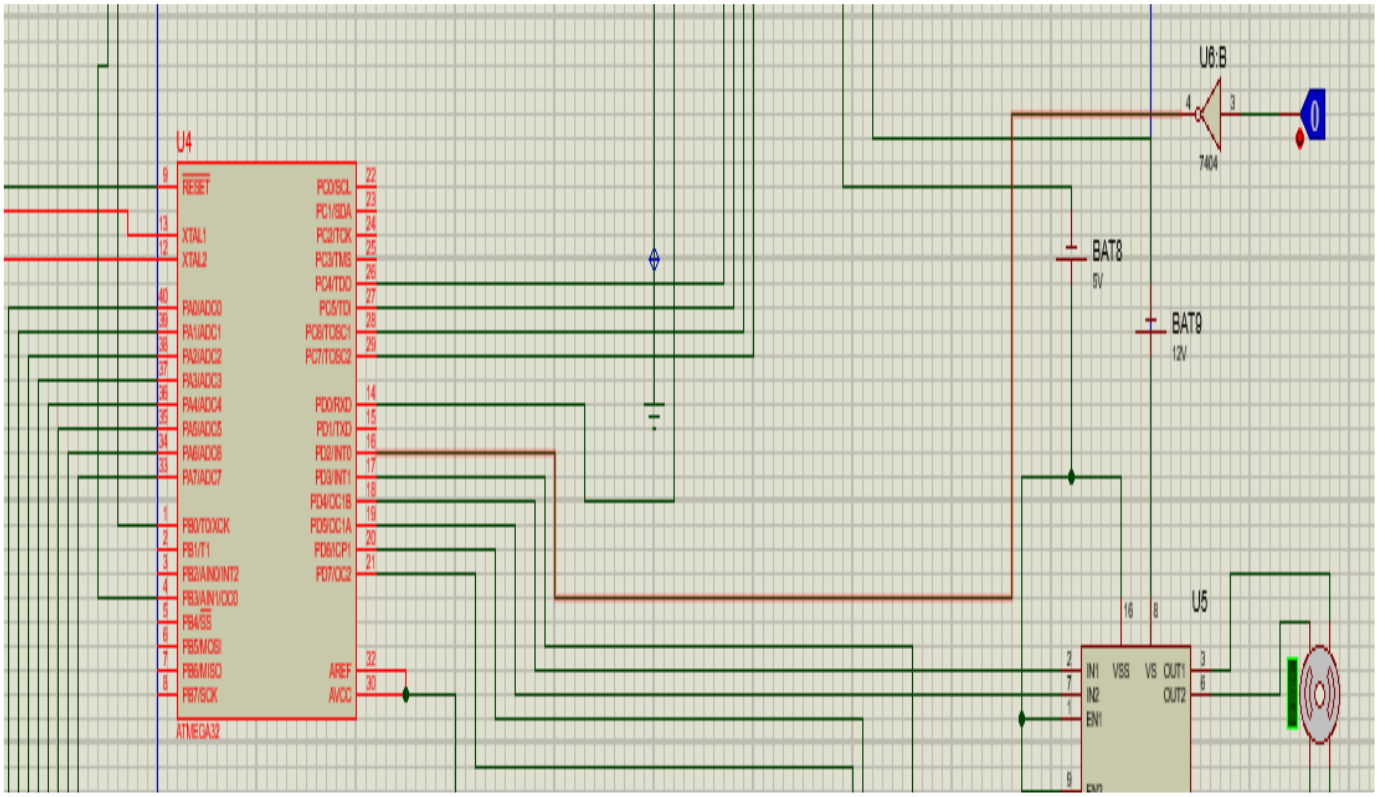


Data Sheet: https://www.daraz.lk/products/nickel-plated-brass-lj18a3-8-zax-npn-nc-dc-inductive-proximity-sensor-switch-detection-8mm-i108690427.html

As this sensor is not available in Proteus, we decided to hardcode the relevant pin (PD2) which the sensors output pin will be connected. Using a **Logic Toggle**.

The output pin of the Inductive Proximity Sensor is connected to pin **PD2**.

Inductive Proximity Sensor Circuit Diagram



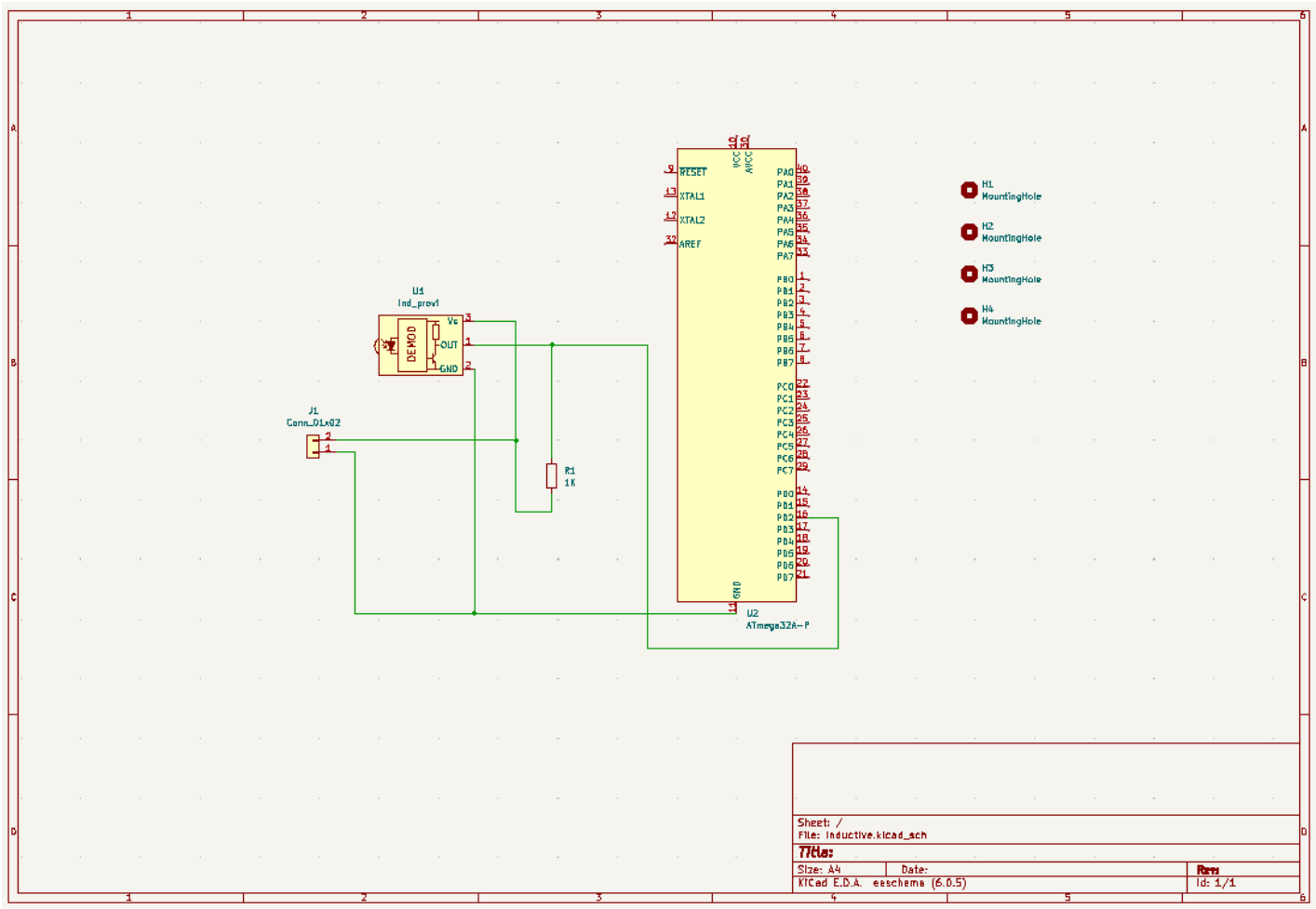
Code

```
int Stepper_motor(){

        //Set a1, a2, a3, a4 as output

        DDRD |= a1 | a2 | a3| a4;

        //Set input & pull-up resistor

        DDRD &= ~SW;

        PORTD |= SW;          //input pin

        DDRD &= ~SW2;         //pullup resistor high

        PORTD |= SW2;

        while(1){

                if((PIND & SW) && (PIND & SW2)){

                        while(1){

                                if((!(PIND & SW2))){

                                        while(1){
                                                if((!(PIND & SW))){
                                                        validcan++;

                                                        PORTD |= a1;

                                                        PORTD &= ~a2;

                                                        PORTD &= ~a3;
```

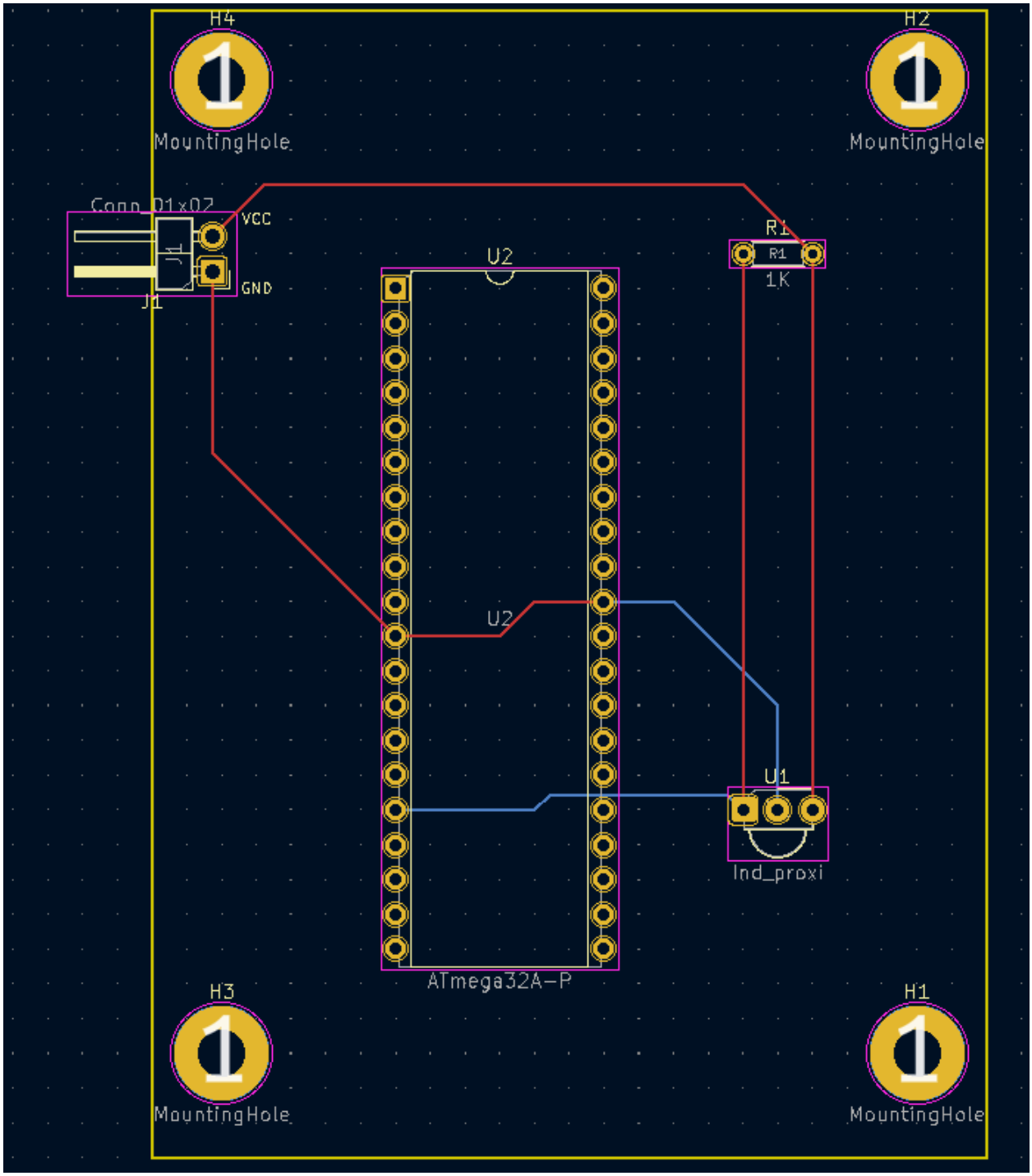
```
PORTD &= ~a4;

_delay_ms(20);


PORTD |= a1;

PORTD &= ~a2;

PORTD  &= ~ a3;
```

Here the code of the sensor is inside the stepper motor code as both functions together.

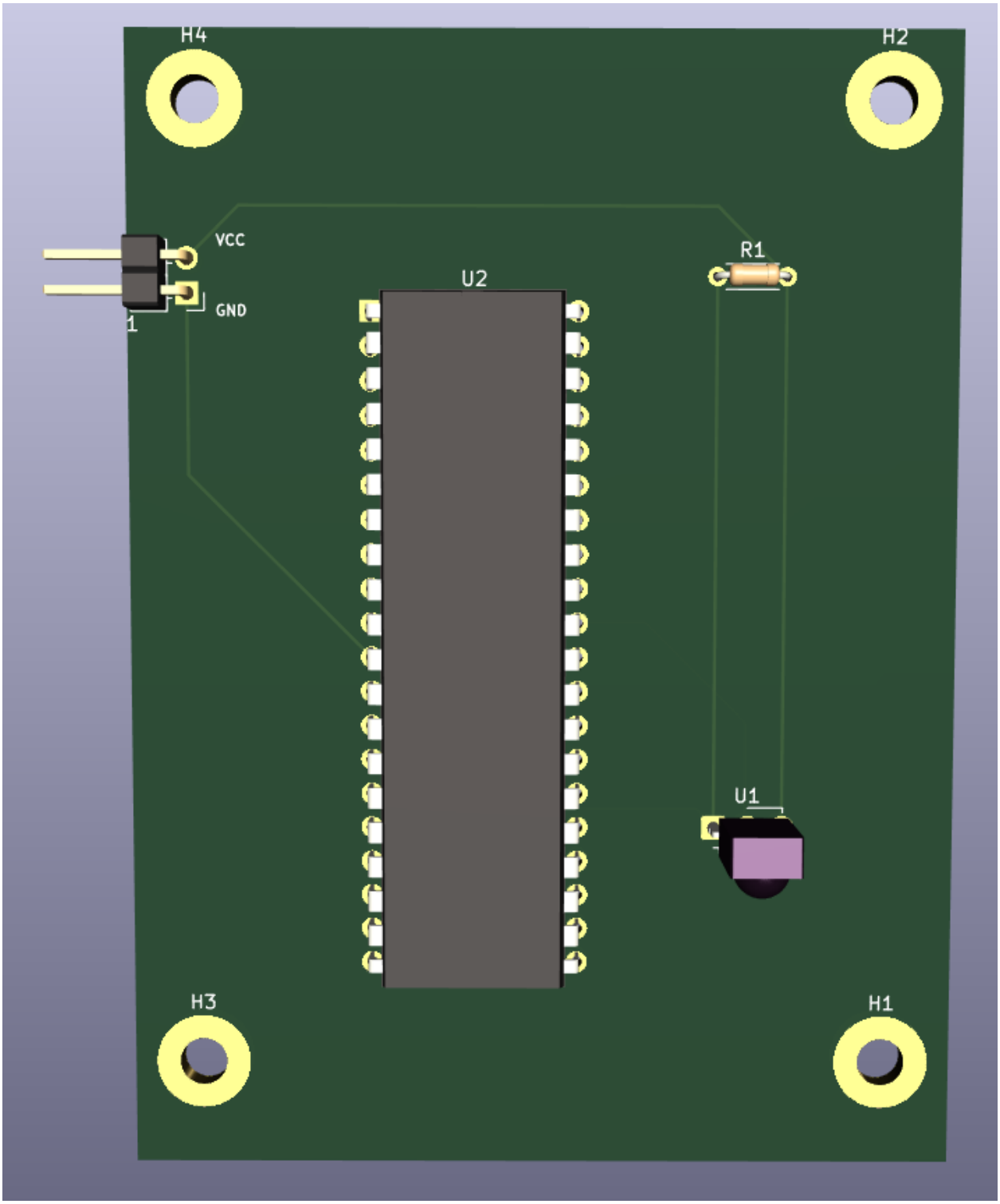
Schematic editor



PCB editor


H4
MountingHole
H2
MountingHole
Conn_01x02
VCC
GND
J1
U2
R1
R1
1K
U2
U1
Ind_proxi
ATmega32A-P
H3
MountingHole
H1
MountingHole

3D viewer

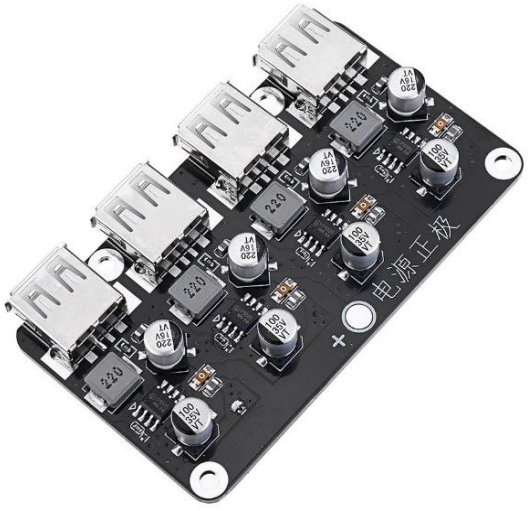


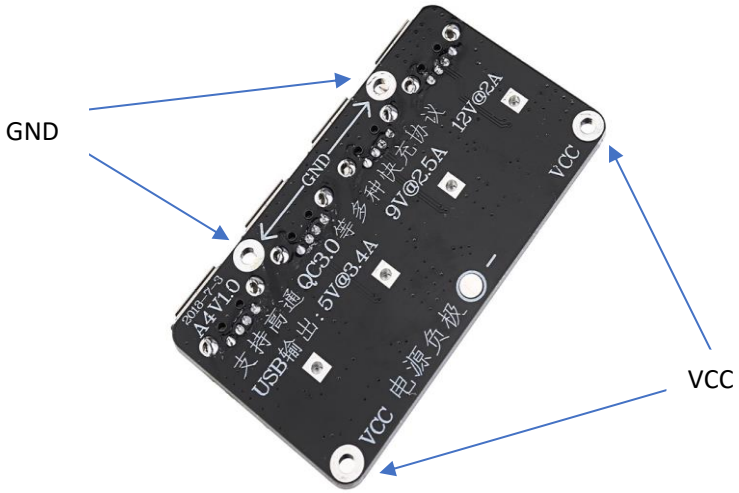
USB Module (SKU: 766344)

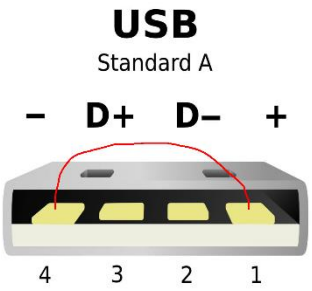
Here we have used fast charging USB ports to facilitate the fast charging.

Instead of type C we have decided to use USB type A as they are more common.

Also, we have used the 1x4 channel USB fast charging module to save space and power consumption.







Here the D- and D+ ports inside the USB is neglected as they are for data transferring.

All we have to do is to power up the module using the supply.

Here the microcontroller acts as a switch. So the USB outlets will be powered on.

Specifications

Power : 2.5V – 5V

Standby current : 0.013A

Output maximum : 24W

Switching Frequency :220 kHZ

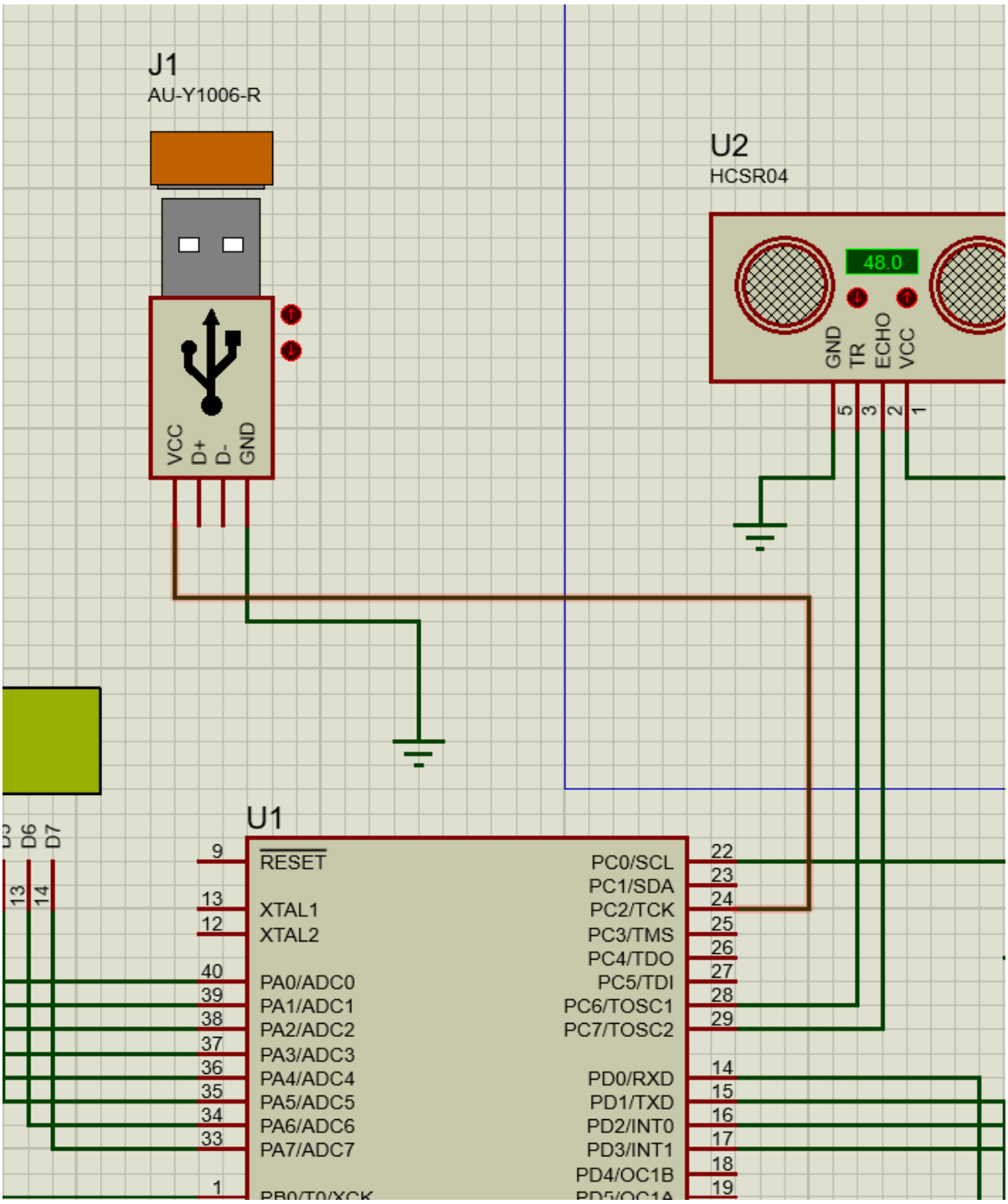
Datasheet: https://www.banggood.com/4-USB-Fast-Charging-Module-Board-12V24V-to-QC2_0-QC3_0-Step-Down-Power-Supply-Module-12V-24V-p-1455615.html?cur_warehouse=CN

Code

```
//amount=cans*price;
amount=1000;

if(r_choice==1) /* fast charging */
{
        PORTC=0b00000010;                  //Powerup USB s
}

else if(r_choice==2) /* Ez-Cash */
{
        GSM_Dialling();
        lcd_string("Reload send",11);
}

else if(r_choice==3) /* Little Hearts */
{
        sprintf(lh,"LH %d",amount);
        GSM_Send_Msg("77110",lh);
        memset(lh,0,strlen(lh));    /* clear lh array */

        lcd_string(".....Sending....",16);
        _delay_ms(70); //original-7000
        lcd_clear();

        lcd_string("Sent to Little",14);
        _delay_ms(30); // //original-500
        lcd_clear();
        lcd_string("Hearts",6);
        _delay_ms(30); //original-500
        lcd_clear();
```

This is the code of the slave atmega microcontroller where USB powering up is included.

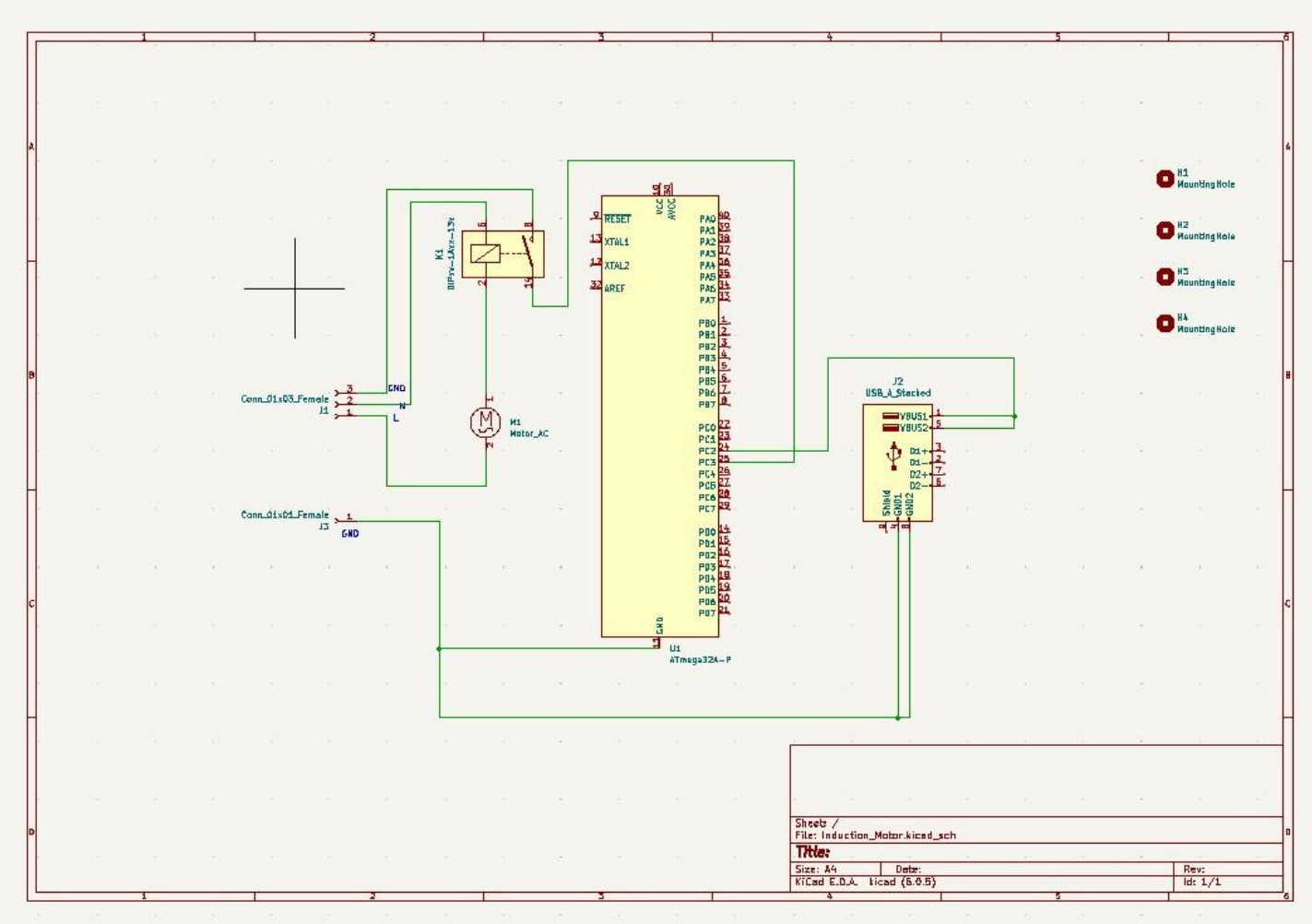
USB Module Circuit Diagram



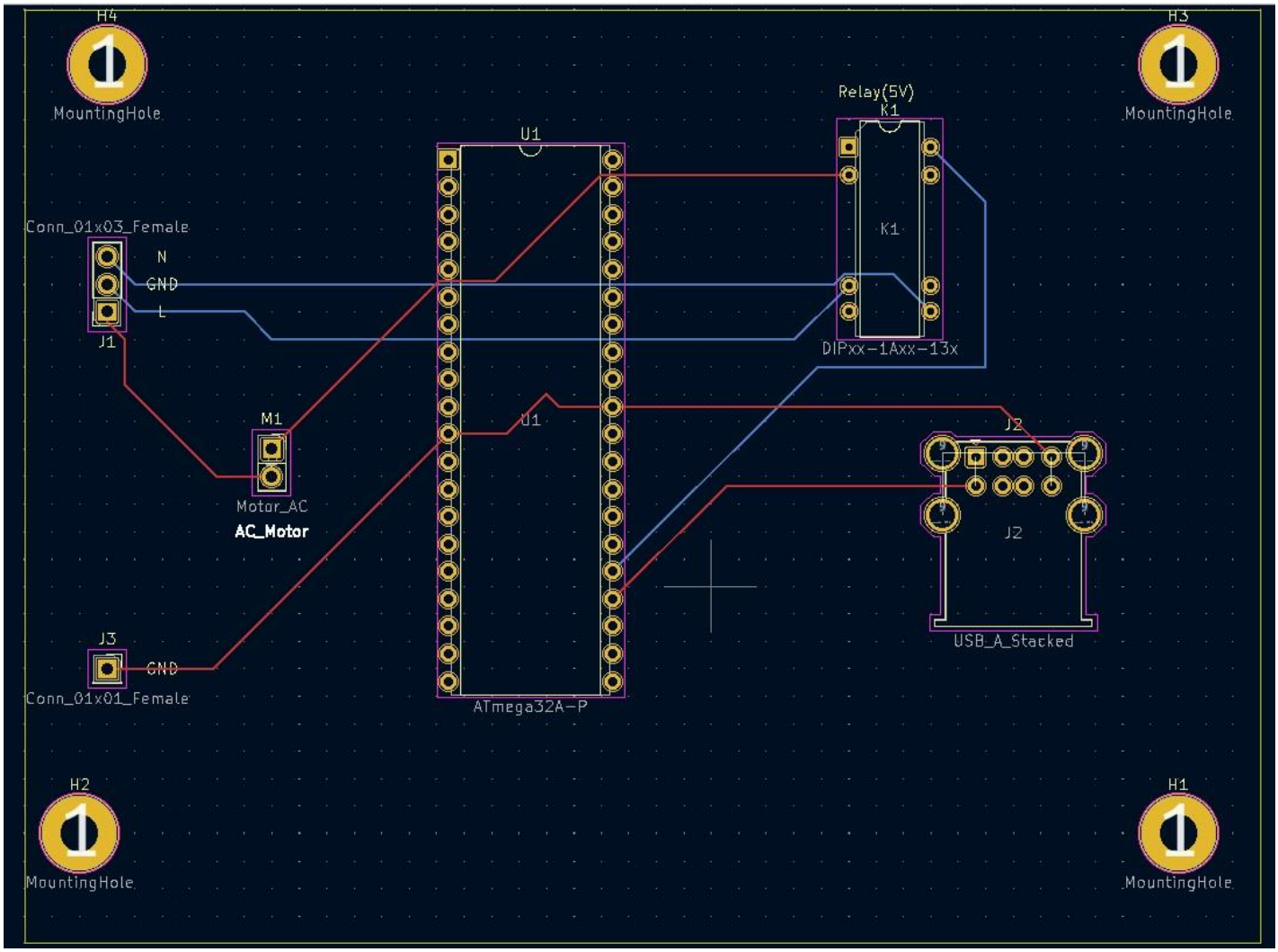
The above diagram is only for demonstration purposes as it cannot be simulated.

Schematic editor

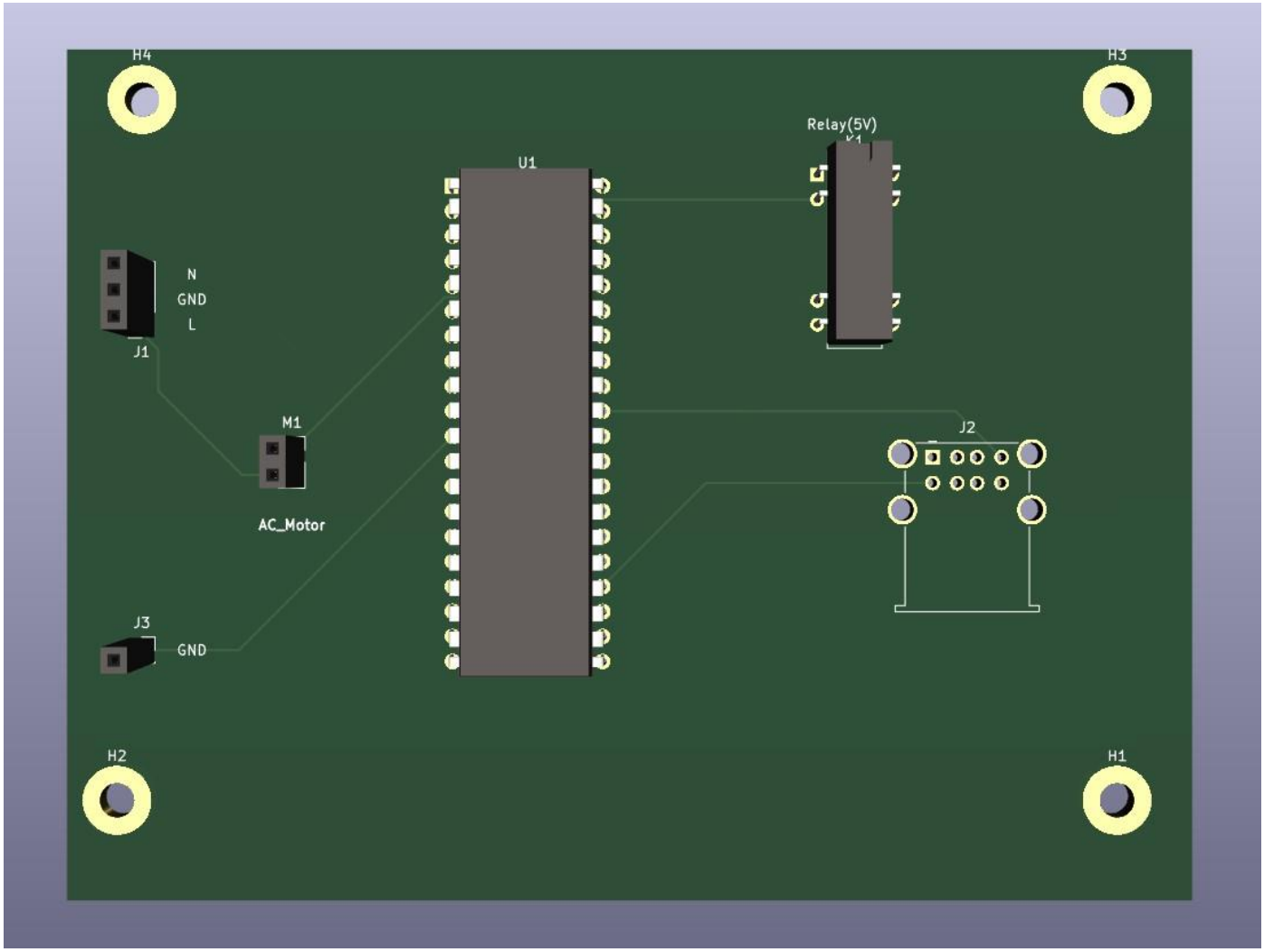
(USB + Induction Motor)



PCB editor

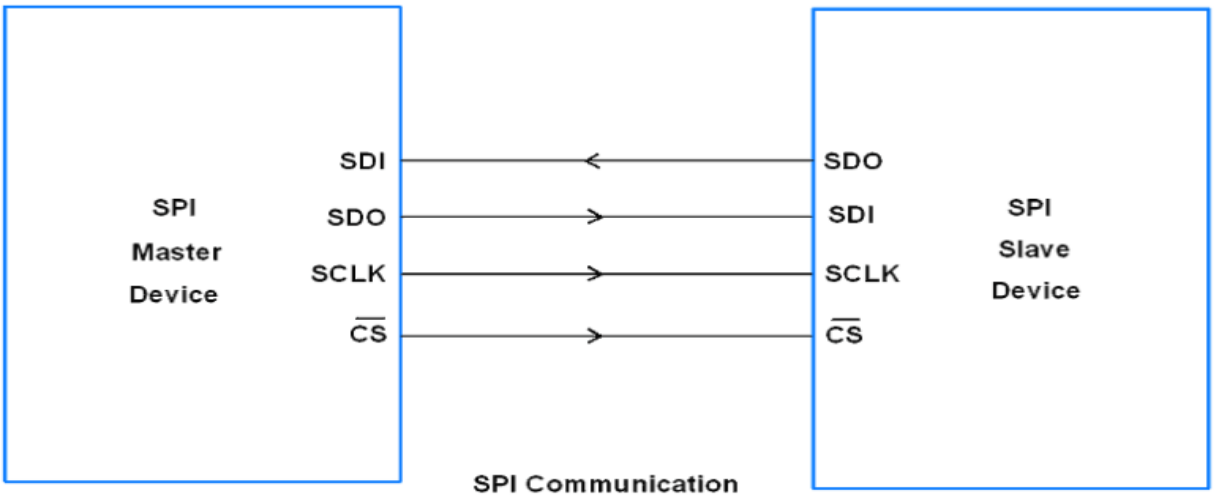

H4
MountingHole
H3
MountingHole
Relay(5V)
K1
U1
Conn_01x03_Female
N
GND
L
J1
K1
DIPxx-1Axx-13x
M1
U1
J2
Motor_AC
AC_Motor
J2
J3
GND
Conn_01x01_Female
USB_A_Stacked
ATmega32A-P
H2
MountingHole
H1
MountingHole

3D viewer



SPI (Serial Peripheral Interface) Communication

Here we use a SPI communication method to communicate between the two atmega microcontrollers.



This is how we tested the code before implementing to the main system. Below is the code of the *master atmega*.

```
#define F_CPU 8000000UL                                    /* Define CPU Frequency e.g.
here its 8MHz */
#include <avr/io.h>                                        /* Include AVR std. library
file */
#include <util/delay.h>                                    /* Include Delay header file */
#include <stdio.h>
#include <string.h>
//#include "LCD_16x2_H_file.h"                    /* Include LCD header file */
#include "SPI_Master.c"                    /* Include SPI master header file */
```

```
int main(void)
{
	unsigned char count[4]="kin";
	//char buffer[5];

	//LCD_Init();

	SPI_Init();
	SS_Enable;

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

{
		SPI_Write(count[i]);


	}

}
```

This is the code of *slave atmega*.

```
#define F_CPU 8000000UL                                    /* Define CPU Frequency e.g.
here its 8MHz */
#include <avr/io.h>                                        /* Include AVR std. library
file */
#include <util/delay.h>                                    /* Include Delay header file */
#include <stdio.h>
#include <string.h>                                        /* Include string header file
*/
//#include "LCD_16x2_H_file.h"                    /* Include LCD header file */
#include "SPI_Slave.c"                    /* Include SPI slave header file */
```

```
int main(void)
{
	unsigned char a;
	//char buffer[5];

	DDRA=0xff;


	//LCD_Init();
	SPI_Init();

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

		if(a=='k')
		{
			PORTA=0b00000100;
		}
		else if(a=='i')
		{
			PORTA=0b00000010;
			_delay_ms(10);
		}
		else if(a=='n')
		{
			PORTA=0b00000100;
			_delay_ms(10);
		}

	}
```

The codes in the final code cannot be demonstrated individually as they are all over the finished code.

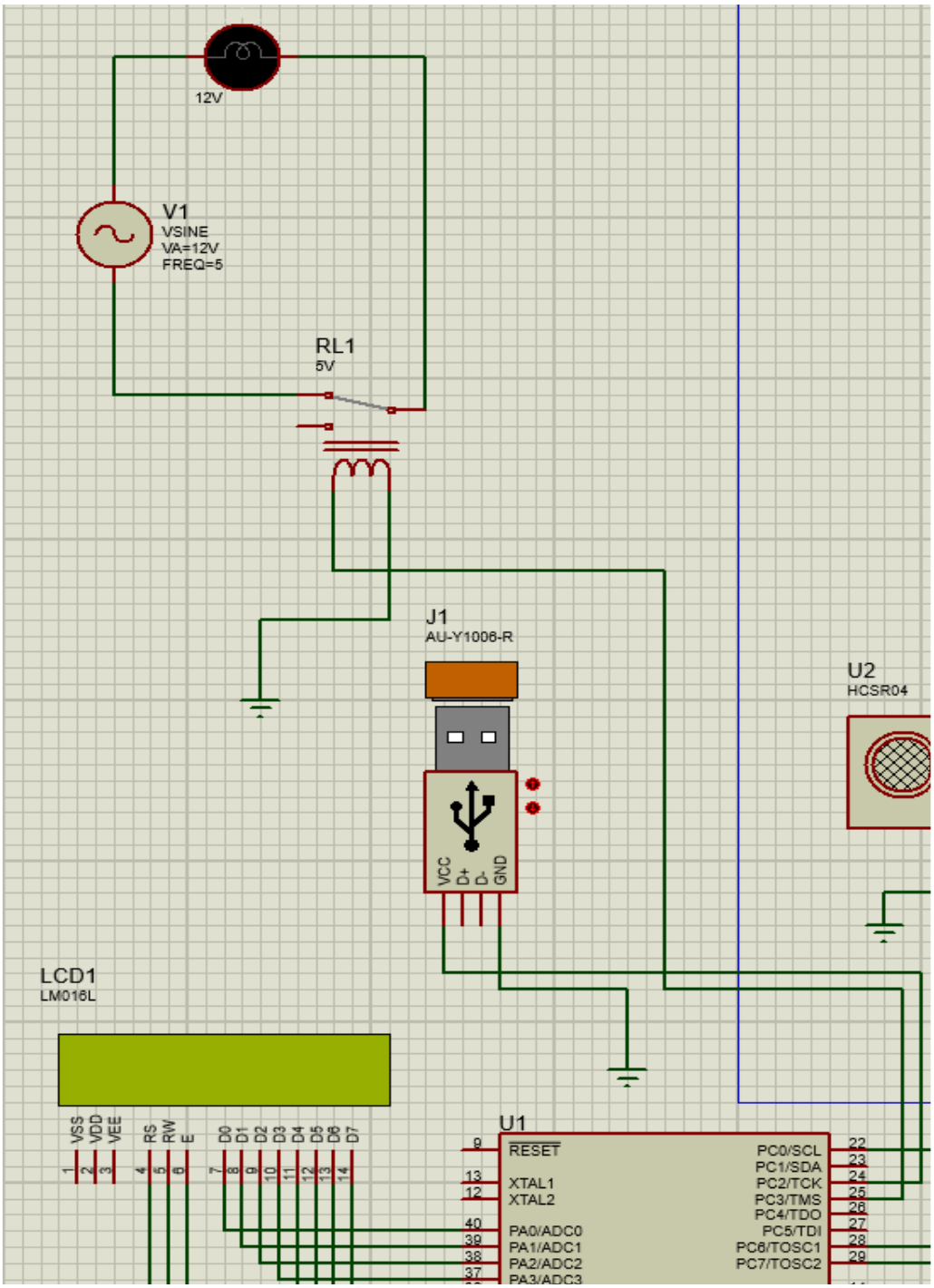
Induction Motor (3Pahse)

We use an Induction motor to crush the cans.

In order to crush the the HP should be at least **0.1**

Type of Motor	Specifications				
	Power		Voltage	Speed	Ratio:1
	(KW)	*(HP)*	*(V)*	*(RPM)*	
	0.09	0.12		1300	1:30,1:20
	0.12	0.16		1330	1:30,1:20
	0.18	0.24		1400	1:10, 1:20 /1 Phase
	0.18	0.24		1400	1:10, 1:20 /3Phase

Induction Motor Circuit Diagram



Here a bulb is used as the motor because AC motors aren't available in proteus.

Here we have use a 5V Relay so it can be triggered with 5V so the AC motor will get 240V from the AC current.

Code

```
            int can_number= atoi(can_no);

            r_slave=SPI_Read();

            if(r_slave)
            {
                  for(int k=0;k<can_number;k++){
                  ir2_func();
                  Stepper_motor();
                        if (validcan>0)      //induction motor
                        {
                              PORTC=0b00000100;
                        }
                  }

            }

            SPI_Write(validcan);
      }

      return 0;
}
```

2. Student name-: Nishath M.N.H.A. – 204140M

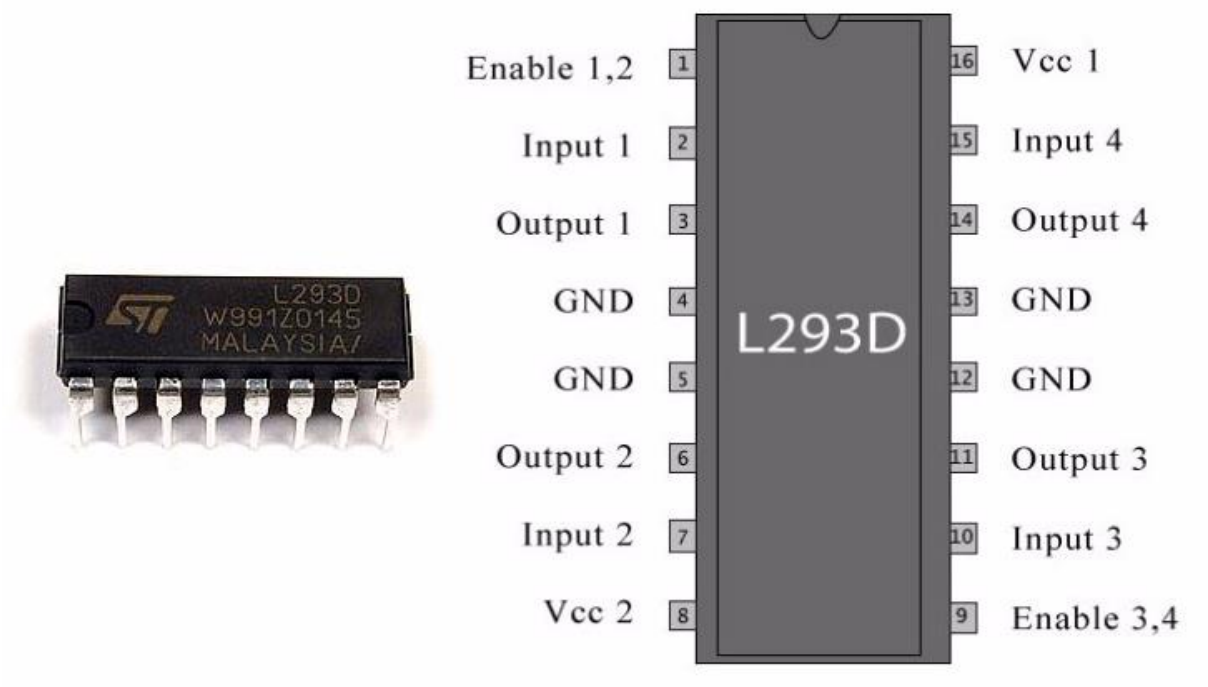
Learning & Interfacing the ,

- Stepper motor
- IR Sensor
- 4x4 Keypad

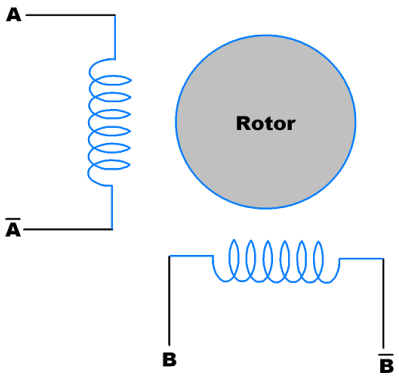
with the atmega32 microcontroller.

Stepper motor

For our project we use NEMA 17 bipolar stepper motor. To control the bipolar stepper motor two H-bridge circuits are required because stepper motor requires sufficient and controlled energy for phases in a precise sequence. So we are going to use L293d driver IC which can work as dual H-bridge driver to drive the stepper motor.



L293D Motor driver IC

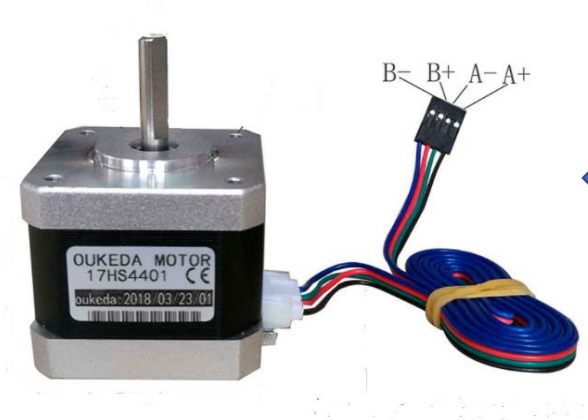


Sequence of pulses that should send to motor in order to rotate the motor in 45° angles

A+	A-	B-	B+
1	0	0	0
1	0	1	0
0	0	1	0
0	1	1	0
0	1	0	0
0	1	0	1
0	0	0	1
1	0	1	0

Electrical Specification

- NEMA17 have a recommended driving voltage of 12-24V.
- Motor Type: Bipolar
- Step Angle: is 1.8 deg



Nema17 bipolar stepper

Function of the code

After the scanning process happened .If the can identified as an aluminium can then can would be send into crushing unit and if the can is rejected then it would be send into Discarding unit .We are using an arm that can rotate by a stepper motor to do the Process of moving the can into crushing unit or discarding unit.

Code

```
#ifndef F_CPU

#define F_CPU 16000000UL //freaquency of external crystal

#endif

#include <avr/io.h>

#include <util/delay.h>



//define interfacing pins between L293D and ATmega32

#define a1 (1<<PD4)

#define a2 (1<<PD5)

#define a3 (1<<PD6)

#define a4  (1<<PD7)



//For switch

#define SW (1<<PD3)    //Ir

#define SW2 (1<<PD2)   //ALL validation
```

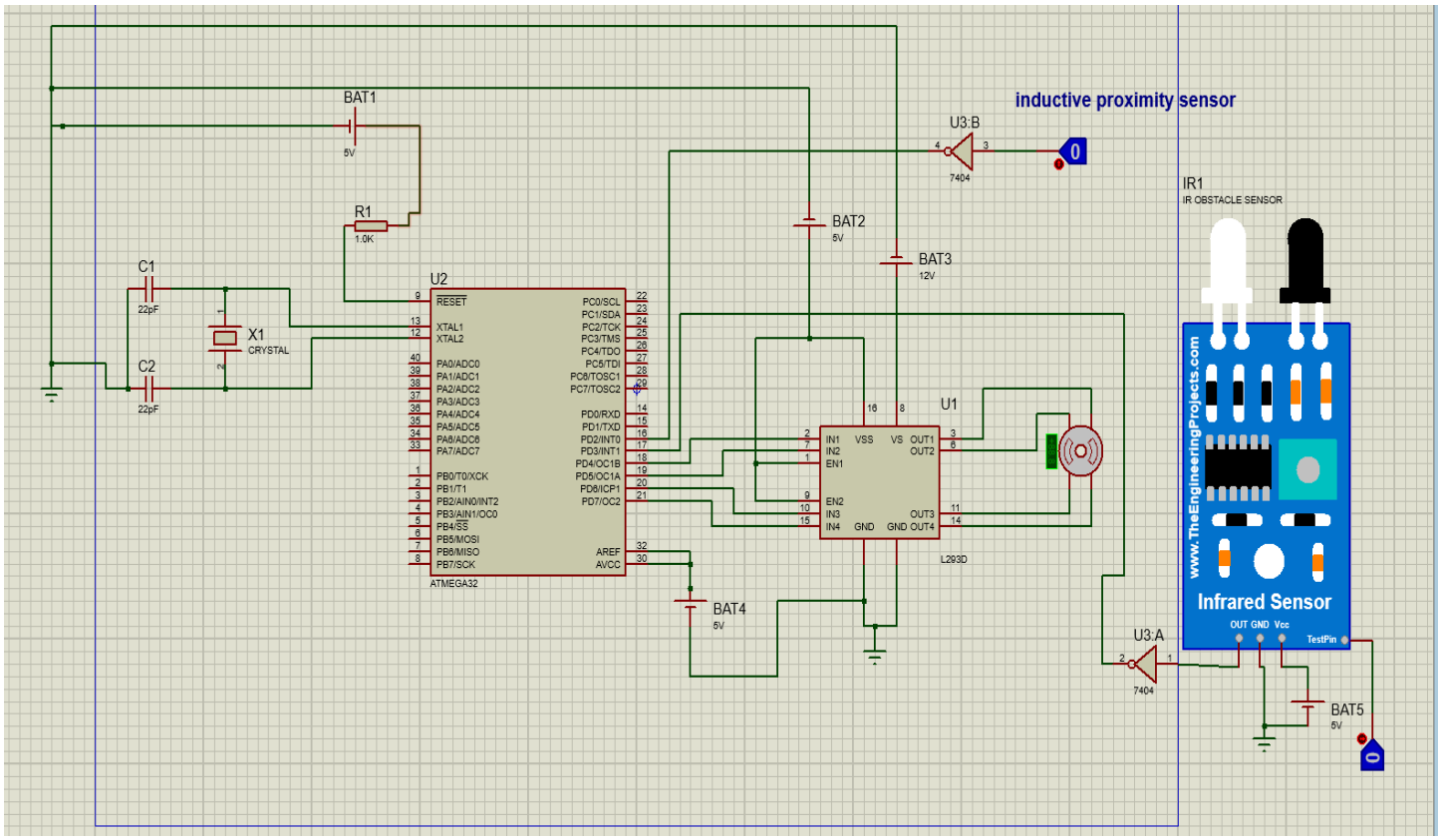
```
int main(void){

        Stepper_motor();

        return(0);

}

void Stepper_motor(){

        //Set a1, a2, a3, a4 as output

        DDRD |= a1 | a2 | a3| a4;

        //Set input & pull-up resistor

        DDRD &= ~SW;

        PORTD |= SW;

        DDRD &= ~SW2;

        PORTD |= SW2;

        while(1){
                if((PIND & SW) && (PIND & SW2)){
                        while(1){
                                if((!(PIND & SW2))){
```

```
while(1){
        if((!(PIND & SW))){
                PORTD |= a1;
                PORTD &= ~a2;
                PORTD &= ~a3;
                PORTD &= ~a4;
                _delay_ms(50);

                PORTD |= a1;
                PORTD &= ~a2;
                PORTD  &= ~ a3;
                PORTD |= a4;
                _delay_ms(20);


                PORTD &= ~a1;
                PORTD &= ~a2;
                PORTD &= ~a3;
                PORTD |= a4;
                _delay_ms(50);

                PORTD |= a1;
                PORTD &= ~a2;
                PORTD &= ~ a3;
                PORTD |= a4;
                _delay_ms(20);

                PORTD |= a1;
                PORTD &= ~a2;
                PORTD &= ~a3;
                PORTD &= ~a4;
```

```
                    _delay_ms(50);
                    break;
                }
            }
        }

        else if(!(PIND & SW)){


            PORTD |= a1;
            PORTD &= ~a2;
            PORTD &= ~a3;
            PORTD &= ~a4;
            _delay_ms(50);

            PORTD |= a1;
            PORTD &= ~a2;
            PORTD |= a3;
            PORTD &= ~a4;
            _delay_ms(20);

            PORTD &= ~a1;
            PORTD &= ~a2;
            PORTD |= a3;
            PORTD &= ~a4;
            _delay_ms(50);
            PORTD |= a1;
            PORTD &= ~a2;
            PORTD |= a3;
            PORTD &= ~a4;
            _delay_ms(20);
```

```
                    PORTD |= a1;
                    PORTD &= ~a2;
                    PORTD &= ~a3;
                    PORTD &= ~a4;
                    _delay_ms(50);
                    break;
                }
            }
        }
    }
```

Stepper Motor Circuit Diagram

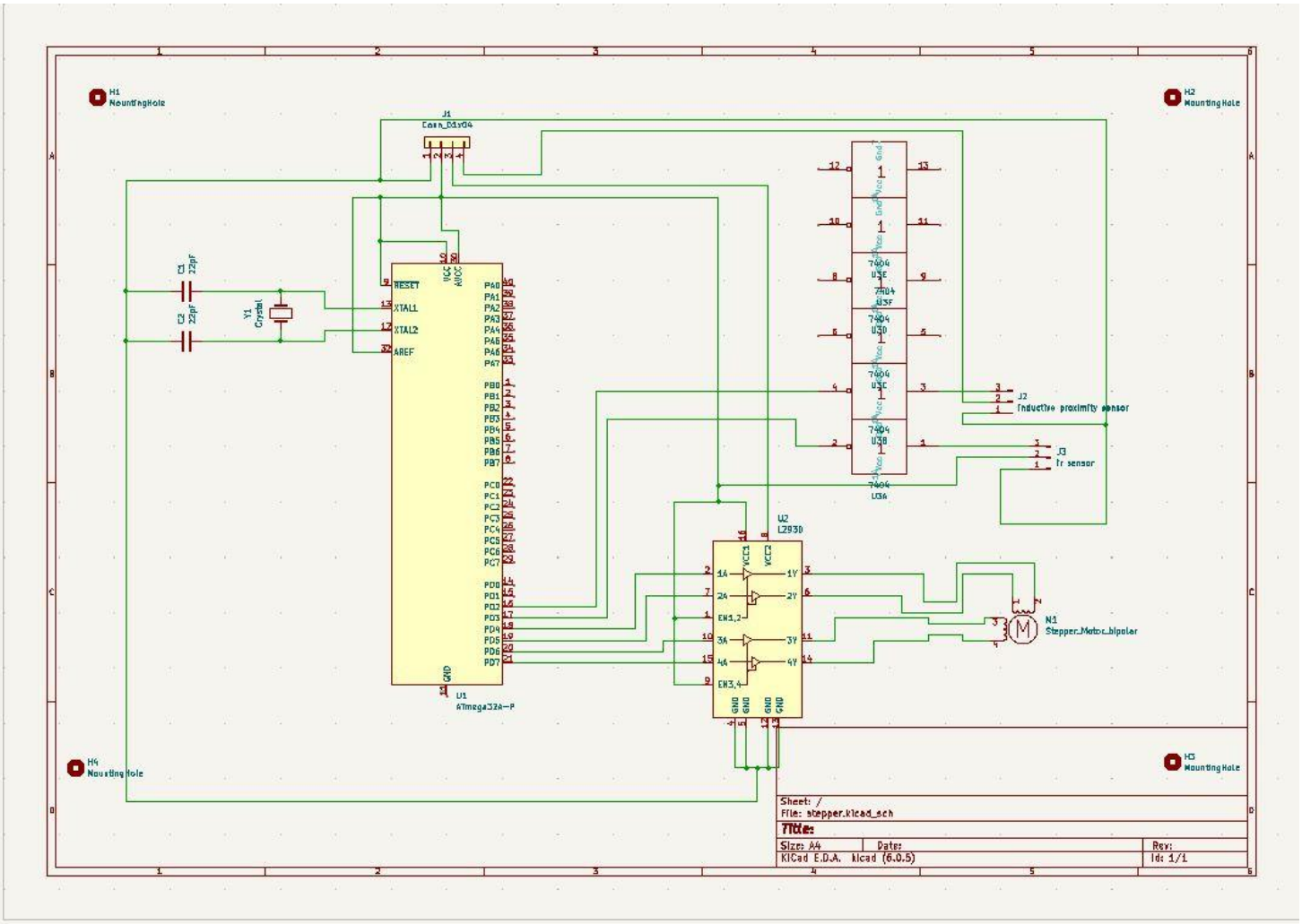


Data sheet of Stepper motor-https://www.omc-stepperonline.com/download/17HS15-1504S1.pdf

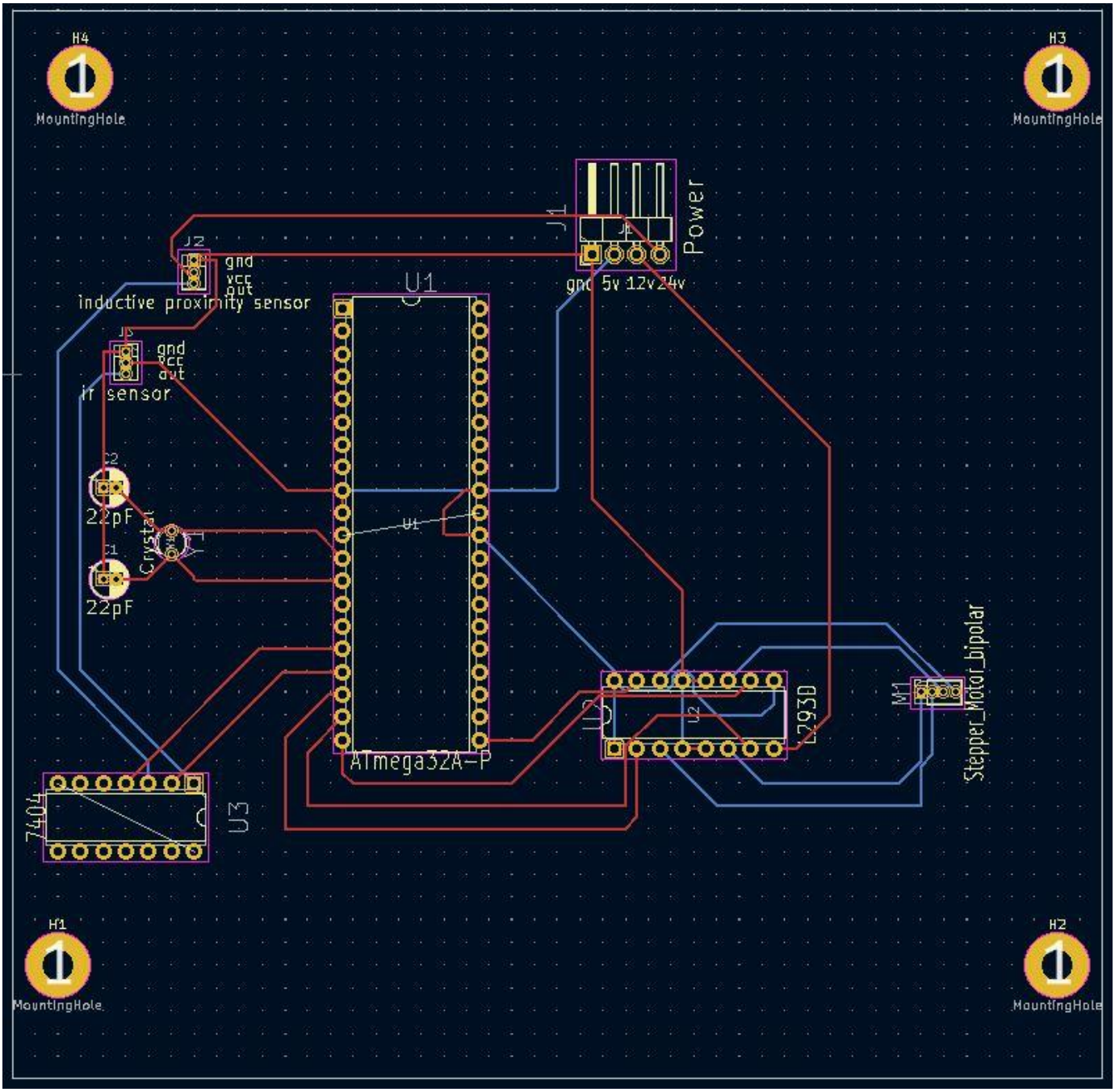
Data sheet of L293D Motor driver IC-

https://components101.com/sites/default/files/component_datasheet/L293D%20Datasheet.pdf

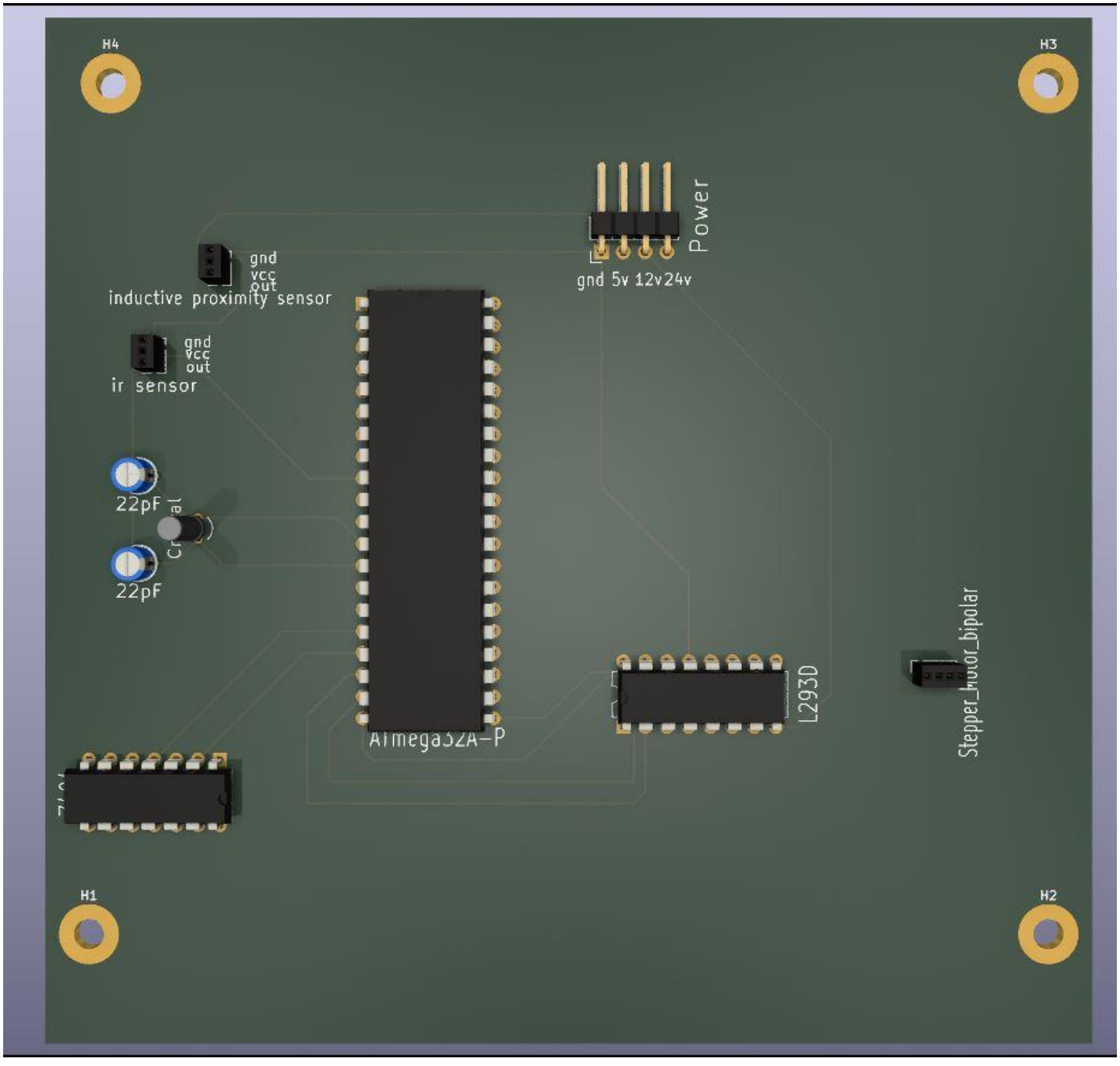
Schematic editor



PCB editor

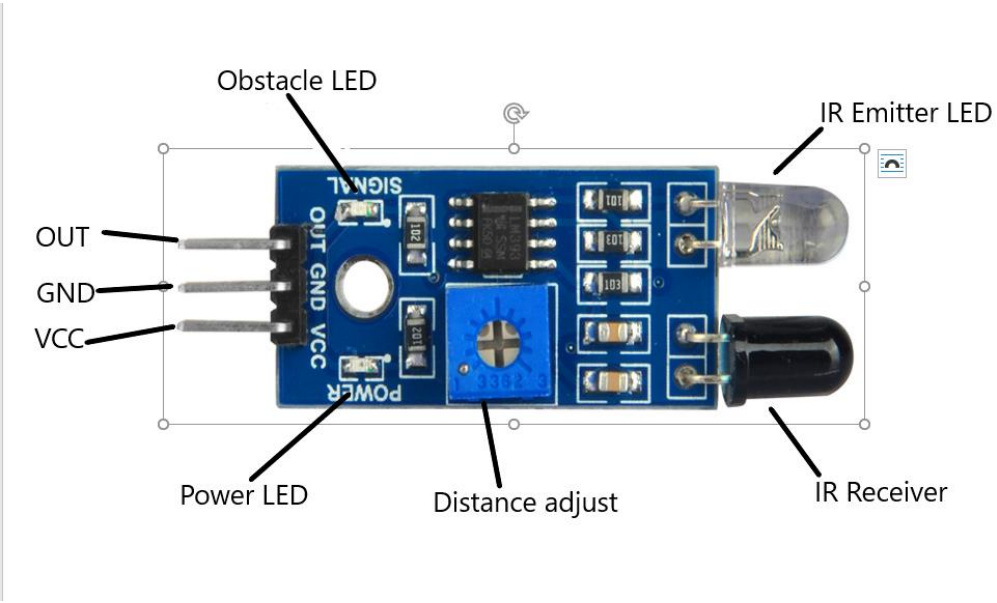

H4
MountingHole
H3
MountingHole
J1
Power
gnd 5v 12v24v
J2
gnd
vcc
out
inductive proximity sensor
J5
gnd
vcc
out
ir sensor
U1
ATmega32A-P
C2
22pF
Crystal
C1
22pF
U2
L293D
M1
Stepper_Motor_bipolar
U3
7404
H1
MountingHole
H2
MountingHole

3D viewer


H4
H3
Power
gnd 5v 12v24v
gnd
vcc
out
inductive proximity sensor
gnd
vcc
out
ir sensor
22pF
22pF
ATmega32A-P
L293D
Stepper_motor_bipolar
H1
H2

IR Sensor

- IR obstacle sensor module has 3 wire interface with VCC , GND and an OUT pin on its tail .
- Operating voltage of the IR sensor is 3.6v to 5v and average current consumption is 0.06mA.
- IR sensor has Effective distance range of (2~30)cm



VCC : 3.3V-5V external voltage

GND: connects to the ground

OUT: connects to one of the pins that is declared as a input in the microcontroller

Function of the code

1.Detect whether discarding unit is full or not

2.Identifying the presence of the can to activate the stepper motor

Code(Detect whether discarding unit is full or not)

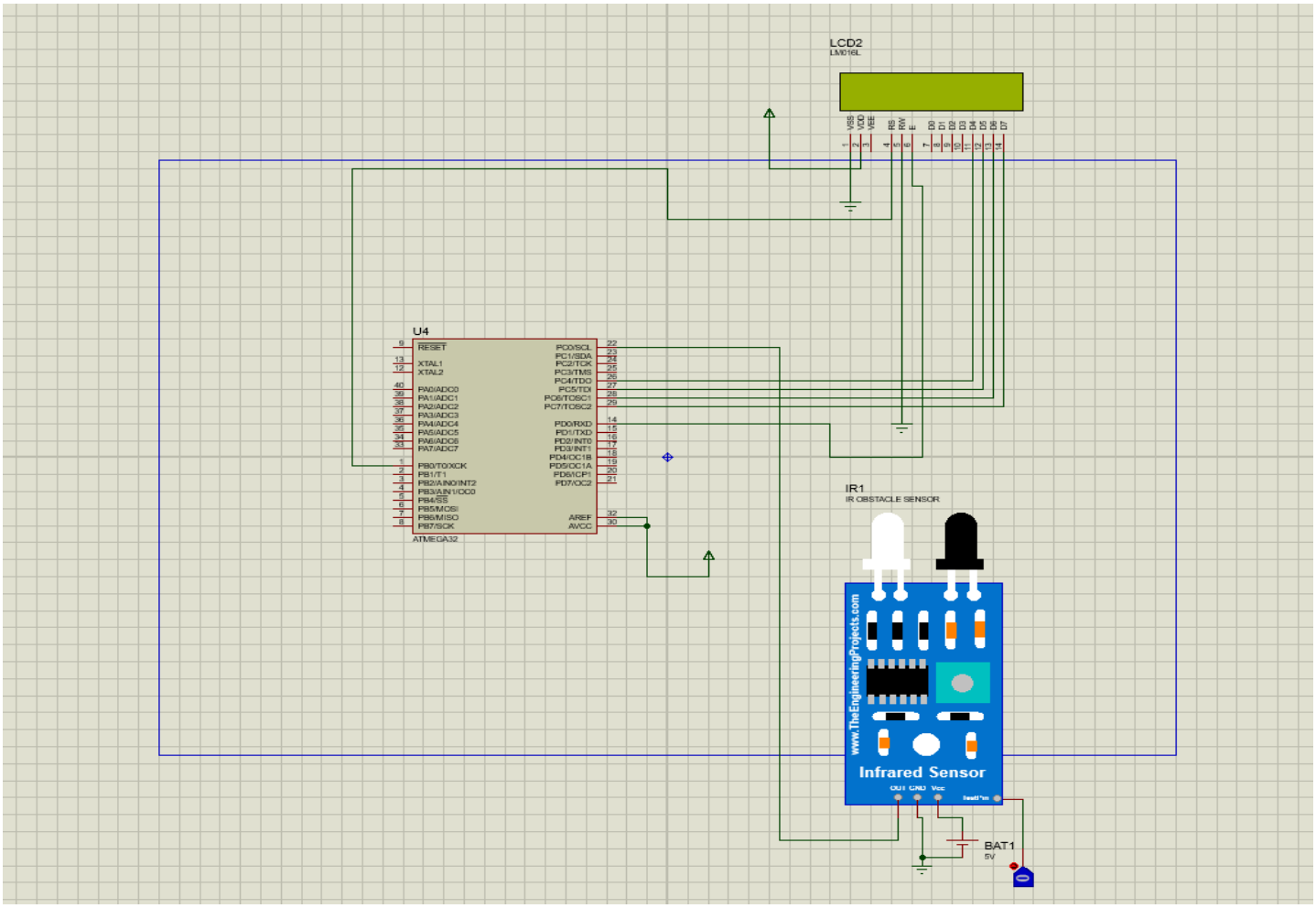
```
#include <avr/io.h>

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

#define LCD PORTC
#define EN 0
#define RS 0
#define RW 1



//int check_discard_bin=0;

void lcdcmd(unsigned char cmd){

        PORTB &=~(1<<RS); //RS=0 for command (1111 1110)
        PORTB &=~(1<<RW); //RW=0 for write (1111 1101)
        LCD= cmd & 0xF0;  //Send upper nibble
        PORTD|=(1<<EN); //EN=1 for H to L pulse
        _delay_ms(1);
        PORTD &= ~(1<<EN); // EN=0 for H to L pulse

        LCD = cmd<<4; // send low nibble;
        PORTD |=(1<<EN); //EN=0for H to L pulse
        _delay_ms(1);
```

```
        PORTD &= ~(1<<EN);
}

void lcd_Print (char*str){
        int i;
        for(i=0;str[i]!=0;i++)                  /* Send each char of string till the NULL */
        {
                lcddata(str[i]);
        }
}


void lcddata(unsigned char data){
        PORTB |= (1<<RS); //RS=1 for data
        PORTB &= ~(1<<RW); //RW=0 for write
        LCD= data & 0xF0;  //Send upper nibble
        PORTD|= (1<<EN); //EN=1 for H to L pulse
        _delay_ms(5);
        PORTD &= ~(1<<EN); // EN=0 for H to L pulse

        LCD = data<<4; // send low nibble;
        PORTD |=(1<<EN); //EN=1 for H to L pulse
        _delay_ms(5);
        PORTD &= ~(1<<EN);

}




void lcd_init(){
```

```
        DDRC= 0xF0; // Define  output LCD port
        DDRD |=(1<<EN); //Define EN pin  as output
        DDRB=0xFF; // define RS and  RW pin as output
        PORTD&= ~(1<<EN); //initialize EN=0
        lcdcmd(0x33);
        lcdcmd(0x32);
        lcdcmd(0x28); // LCD in 4 bit mode
        lcdcmd(0x0E); //display on cursor on
        /*lcddata(72);
        lcddata(69);
        lcddata(76);
        lcddata(76);
        lcddata(79);*/
        lcd_Print("WELCOME");
        _delay_ms(40);
        lcdcmd(0x01); //clear LCD
        _delay_ms(2);
}
int ir2_func(){
        while(1){
                if(PINC & 0x01){
                        lcdcmd(0x01); //clear LCD
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print("Discard bin is");
                        lcdcmd(0xc0);//set the curser to second line
                        lcd_Print("full");
                        lcdcmd(0x80);//set the curser to first line
                        _delay_ms(50);
                        lcdcmd(0x01); //clear LCD
                        lcd_Print("Clear the");
                        lcdcmd(0xc0);//set the curser to second line
```

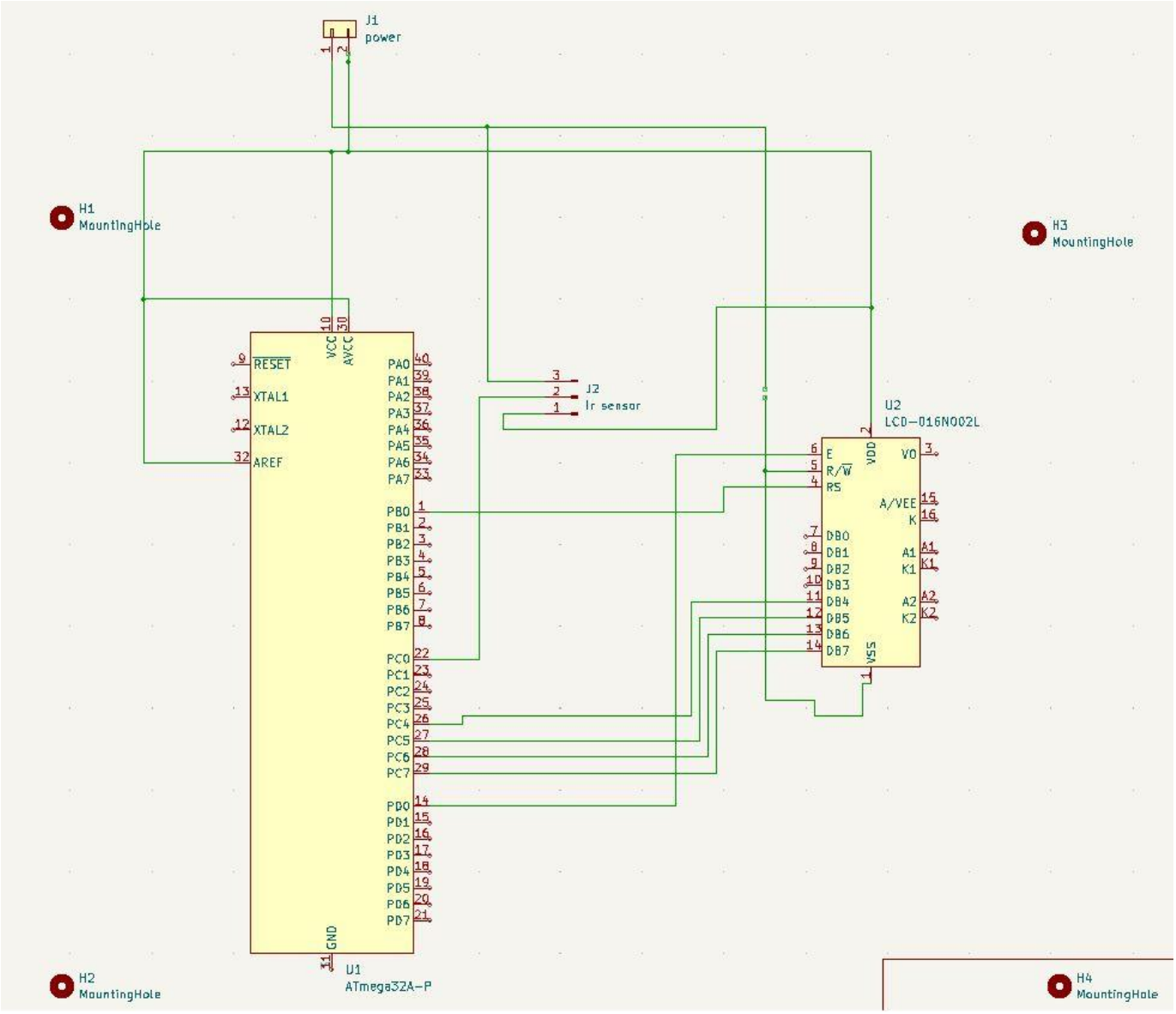
```
                    lcd_Print("discard bin");
                    lcdcmd(0x80);//set the curser to first line
                }
                else{
                    return 0;
                }

        }
}

int main(void)
{
        lcd_init();
        while(1){
                ir2_func();

        }
}
```

IR Sensor Circuit Diagram

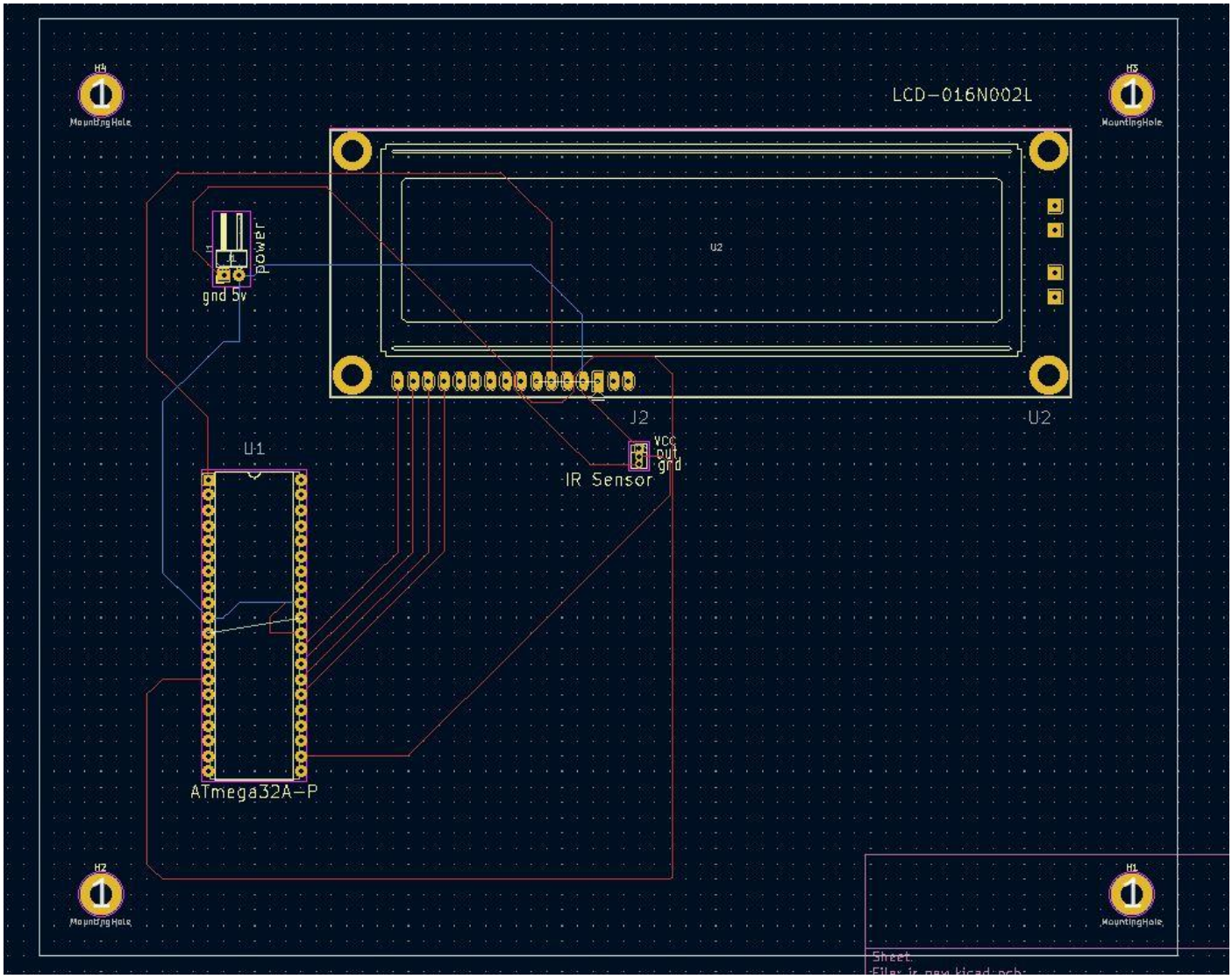


Data sheet of IR obstacle sensor-**file:///C:/Users/user/Downloads/arduino-ir-infrared-obstacle-avoidance-sensor-module%20(1).pdf**

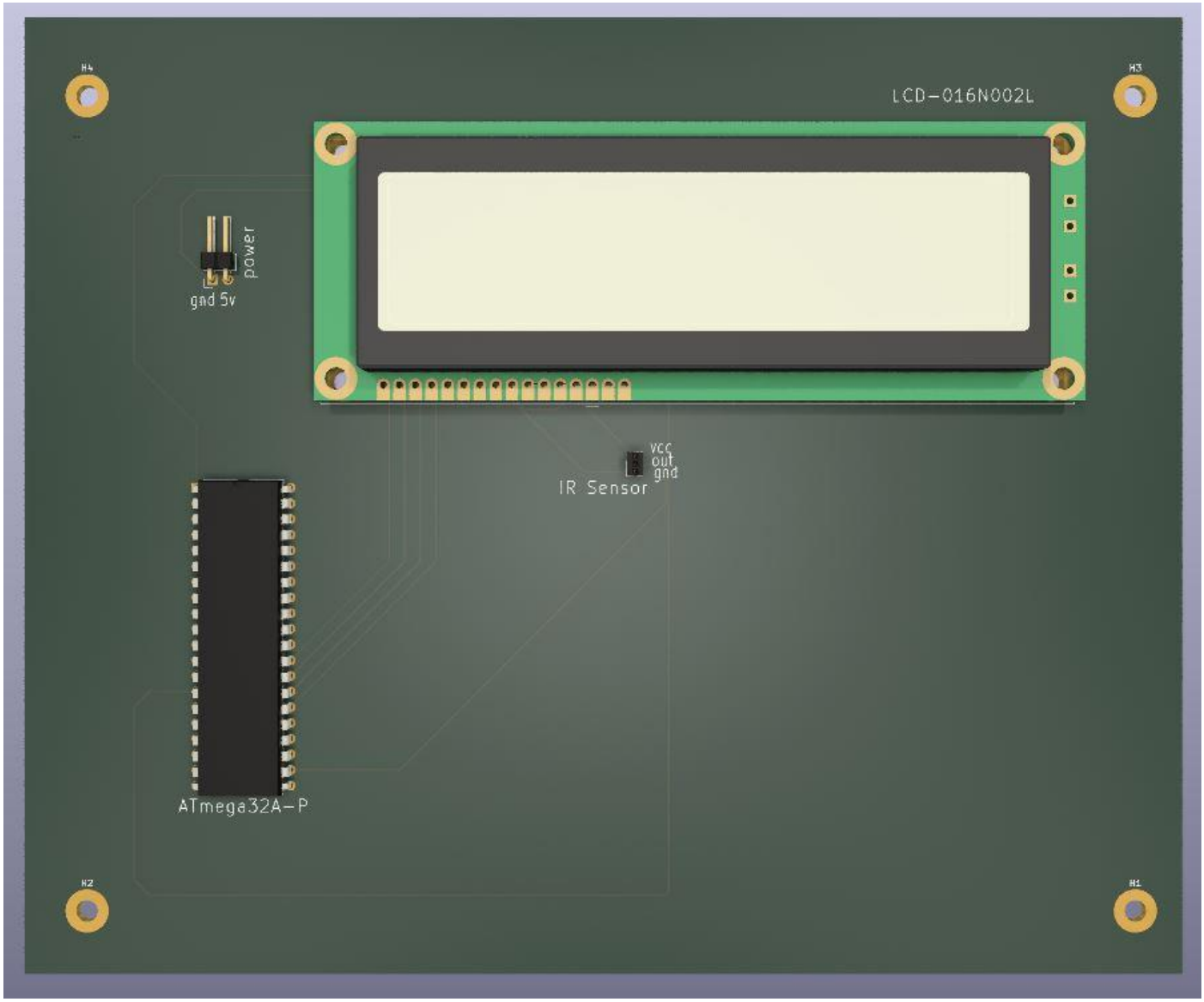
Schematic editor



PCB editor


LCD-016N002L
power
gnd 5v
U1
ATmega32A-P
J2
vcc
out
grd
IR Sensor
U2
MountingHole

3D viewer


H4
H3
LCD-016N002L
power
gnd 5v
vcc
out
gnd
IR Sensor
ATmega32A-P
H2
H1

Key pad

For our project we use 4x4 matrix keypad with 16 buttons .Eight pins from atmega32 microcontroller are needed to connect the 4x4 keypad

Electrical Specification

- 8-pin access to 4×4 matrix
- Dimensions: Keypad: 2.7 x 3.0 in (6.9 x 7.6 cm)
- Operating temp range: 32 to 122 °F

function of the code

1.Get the number of cans that user is going to enter.

2.After entering the cans select which reward option user wants(donate , fast charge or reload)

3.Get the phone number to process the reload option.

code

```
#define F_CPU 16000000UL //freaquency of external crystal


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



//lcd and key board
#define LCD PORTC
#define EN 0
#define RS 0
#define RW 1

 int get_pn;
 char can_no[10];



unsigned char keypad();
int keypad_func(){
        unsigned char x;
        char no[10];
        int i=0;
        int i1=0;
        DDRA =0x0F; // make PA0 to PA3 = output and PA4 to PA7 = Input
        lcd_init();
```

```
while(1){
        PORTA=0xF0; // make all 4 column 1 and all rows 0
        if(PINA!= 0xF0){
                _delay_ms(5);
                x=keypad();
                if (x=='*')
                {
                        lcdcmd(0x01);//clear lcd
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print("1-Fast Charge");
                        lcdcmd(0xc0);//set the curser to second line
                        lcd_Print("2-Ez Cash");
                        _delay_ms(100);
                        lcdcmd(0x01);
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print("3-Donate");

                        while(1){
                                if(PINA!= 0xF0){
                                        x=keypad();
                                        //charge
                                        if(x=='1'){
                                                _delay_ms(5);
                                                lcdcmd(0x80);//set the curser to first line
                                                lcd_Print(" ");
                                                lcdcmd(0x01);//clear lcd
                                                break;
                                        }
                                        //easy cash
```

```
                else if(x=='2'){
                        get_pn=1;//get phone no
                        _delay_ms(5);
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print(" ");
                        lcdcmd(0x01);//clear lcd
                        lcd_Print("Enter the Phone");
                        lcdcmd(0xc0);
                        lcd_Print("Number");
                        lcdcmd(0x80);
                        _delay_ms(100);
                        lcdcmd(0x01);
                        lcd_Print("Press = to");
                        lcdcmd(0xc0);
                        lcd_Print("Finish");
                        lcdcmd(0x80);
                        _delay_ms(100);
                        lcdcmd(0x01);
                        break;
                }
                //donate
                else if(x=='3'){

                }
            }

        }

}
else if (x=='=')
{
```

```
                    get_pn=0;
                    lcdcmd(0x01);
                    lcd_Print("Entered phone NO.:");
                    lcdcmd(0xc0);
                    lcd_Print(&no);//print the no
                    lcdcmd(0x80);
                    //lcd_Print(&can_no);//print the no
                    int phone_no= atoi(no);//phone nimber in int
                    _delay_ms(100);
                    lcdcmd(0x01);
                    _delay_ms(100);
                    lcdcmd(0x01);
                    i=0;
                    return 0;
                }
                else
                {
                    lcddata(x);
                    if(get_pn==1){
                        no[i]=x;
                        i++;
                    }
                    else{
                        can_no[i1]=x;
                        i1++;
                    }
                }
            }
        }
```

```
}

void lcdcmd(unsigned char cmd){
        PORTB &=~(1<<RS); //RS=0 for command (1111 1110)
        PORTB &=~(1<<RW); //RW=0 for write (1111 1101)
        LCD= cmd & 0xF0;  //Send upper nibble
        PORTD|=(1<<EN); //EN=1 for H to L pulse
        _delay_ms(1);
        PORTD &= ~(1<<EN); // EN=0 for H to L pulse

        LCD = cmd<<4; // send low nibble;
        PORTD |=(1<<EN); //EN=0for H to L pulse
        _delay_ms(1);
        PORTD &= ~(1<<EN);
}

void lcd_Print (char*str){
        int i;
        for(i=0;str[i]!=0;i++)                  /* Send each char of string till the NULL */
        {
                lcddata(str[i]);
        }
}


void lcddata(unsigned char data){
        PORTB |= (1<<RS); //RS=1 for data
        PORTB &= ~(1<<RW); //RW=0 for write
        LCD= data & 0xF0;  //Send upper nibble
        PORTD|= (1<<EN); //EN=1 for H to L pulse
        _delay_ms(5);
```

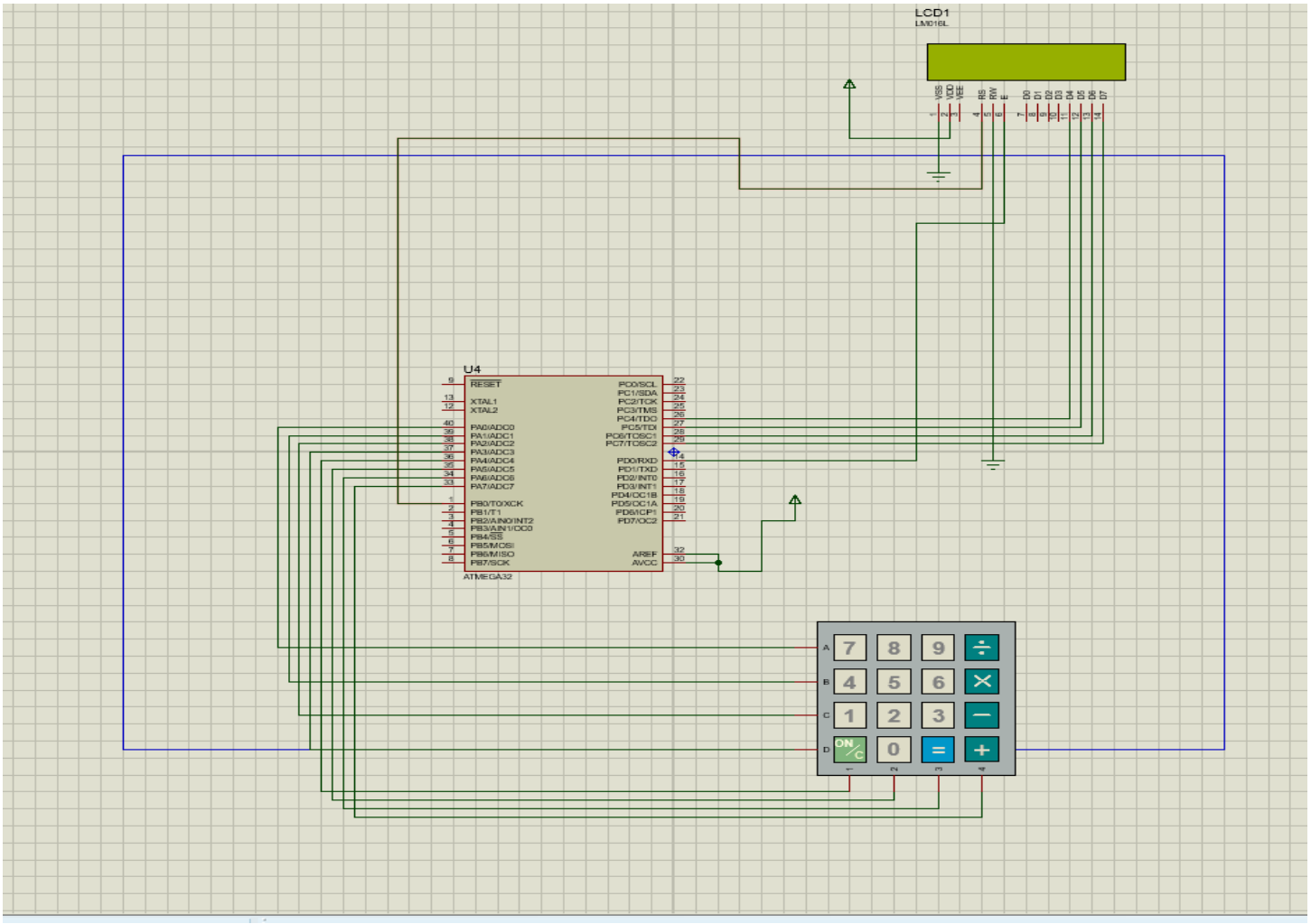
```
        PORTD &= ~(1<<EN); // EN=0 for H to L pulse


        LCD = data<<4; // send low nibble;
        PORTD |=(1<<EN); //EN=1 for H to L pulse
        _delay_ms(5);
        PORTD &= ~(1<<EN);

}

void lcd_init(){
        //DDRC= 0xFF;
        DDRC= 0xF0; // Define  output LCD port(ALL c ports are outputs)
        DDRD |=(1<<EN); //Define EN pin  as output
        DDRB=0xFF; // define RS and  RW pin as output
        PORTD&= ~(1<<EN); //initialize EN=0
        lcdcmd(0x33);
        lcdcmd(0x32);
        lcdcmd(0x28); // LCD in 4 bit mode
        lcdcmd(0x0E); //display on cursor on
        /*lcddata(72);
        lcddata(69);
        lcddata(76);
        lcddata(76);
        lcddata(79);*/
        lcd_Print("WELCOME");
        _delay_ms(40);
        lcdcmd(0x01); //clear LCD
        lcd_Print("Enter the Number");
        lcdcmd(0xc0);//set the curser to second line
        lcd_Print("of cans and");
        _delay_ms(40);
```

```
        lcdcmd(0x01); //clear LCD
        lcdcmd(0x80);//set the curser to first line
        lcd_Print("Press + to start");
        _delay_ms(40);
         lcdcmd(0xc0);//set the curser to second line
         lcd_Print("the process : ");
          //lcdcmd(0x80);

}
unsigned char keypad(){
        PORTA=0b11111110; // make 1st row 0
        if((PINA & (1<<PINA4))==0){
                _delay_ms(3);
                return '7';
                }else if((PINA & (1<<PINA5))==0){
                _delay_ms(3);
                return '8';
                }else if((PINA & (1<<PINA6))==0){
                _delay_ms(3);
                return '9';
                }else if((PINA & (1<<PINA7))==0){
                _delay_ms(3);
                return '/';
        }
        PORTA=0b11111101;
        if((PINA & (1<<PINA4))==0){
                _delay_ms(3);
                return '4';
                }else if((PINA & (1<<PINA5))==0){
                _delay_ms(3);
                return '5';
```

```
            }else if((PINA & (1<<PINA6))==0){
            _delay_ms(3);
            return '6';
            }else if((PINA & (1<<PINA7))==0){
            _delay_ms(3);
            return '+';
    }
    PORTA=0b11111011;
    if((PINA & (1<<PINA4))==0){
            _delay_ms(3);
            return '1';
            }else if((PINA & (1<<PINA5))==0){
            _delay_ms(3);
            return '2';
            }else if((PINA & (1<<PINA6))==0){
            _delay_ms(3);
            return '3';
            }else if((PINA & (1<<PINA7))==0){
            _delay_ms(3);
            return '-';
    }
    PORTA=0b11110111;
    if((PINA & (1<<PINA4))==0){
            _delay_ms(3);
            return 'C';
            }else if((PINA & (1<<PINA5))==0){
            _delay_ms(3);
            return '0';
            }else if((PINA & (1<<PINA6))==0){
            _delay_ms(3);
            return '=';
```

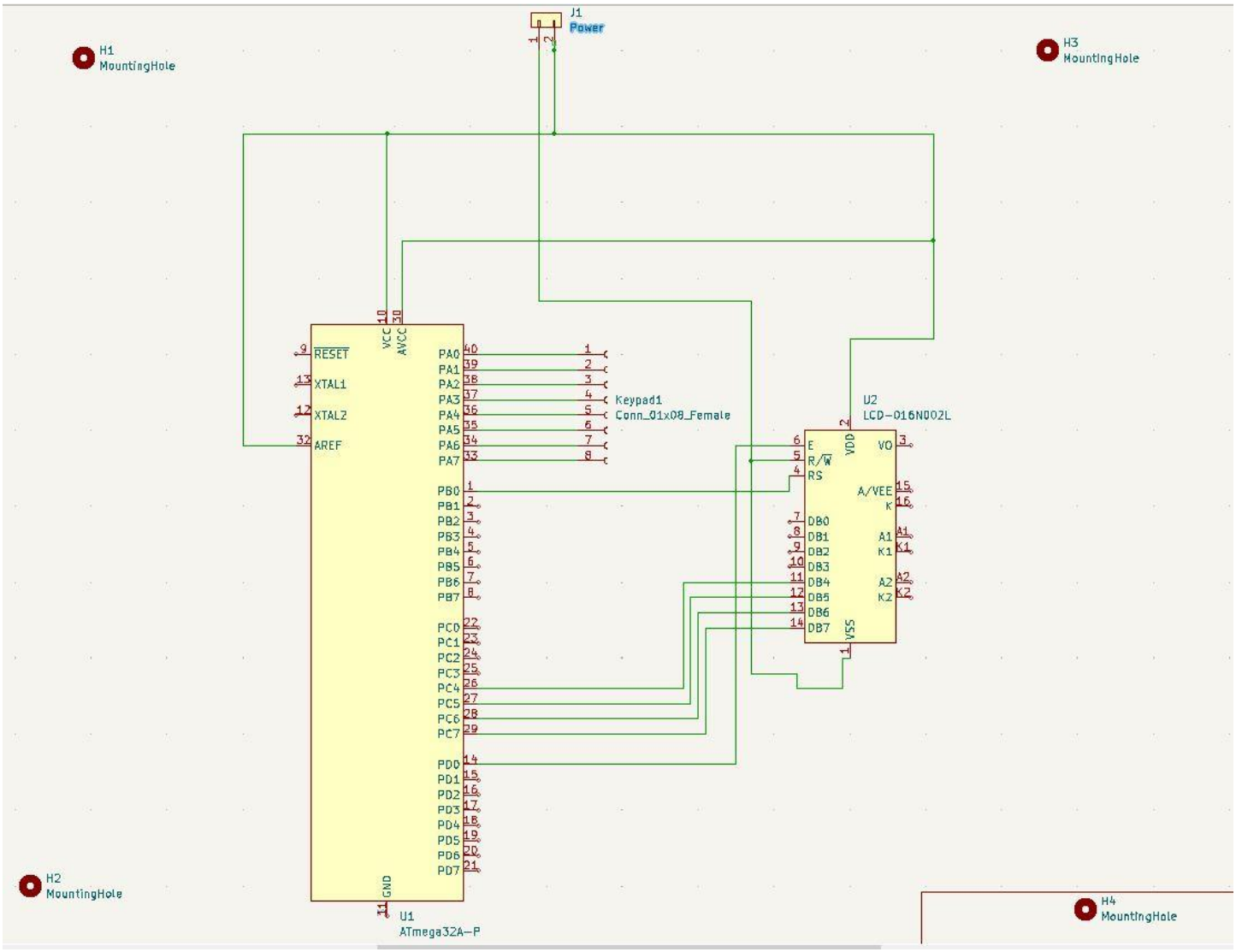
```
                }else if((PINA & (1<<PINA7))==0){
                _delay_ms(3);
                return '*';
        }
}

int main(void){
   while(1){
                keypad_func();

        }

        return 0;
}
```

Keypad Circuit Diagram

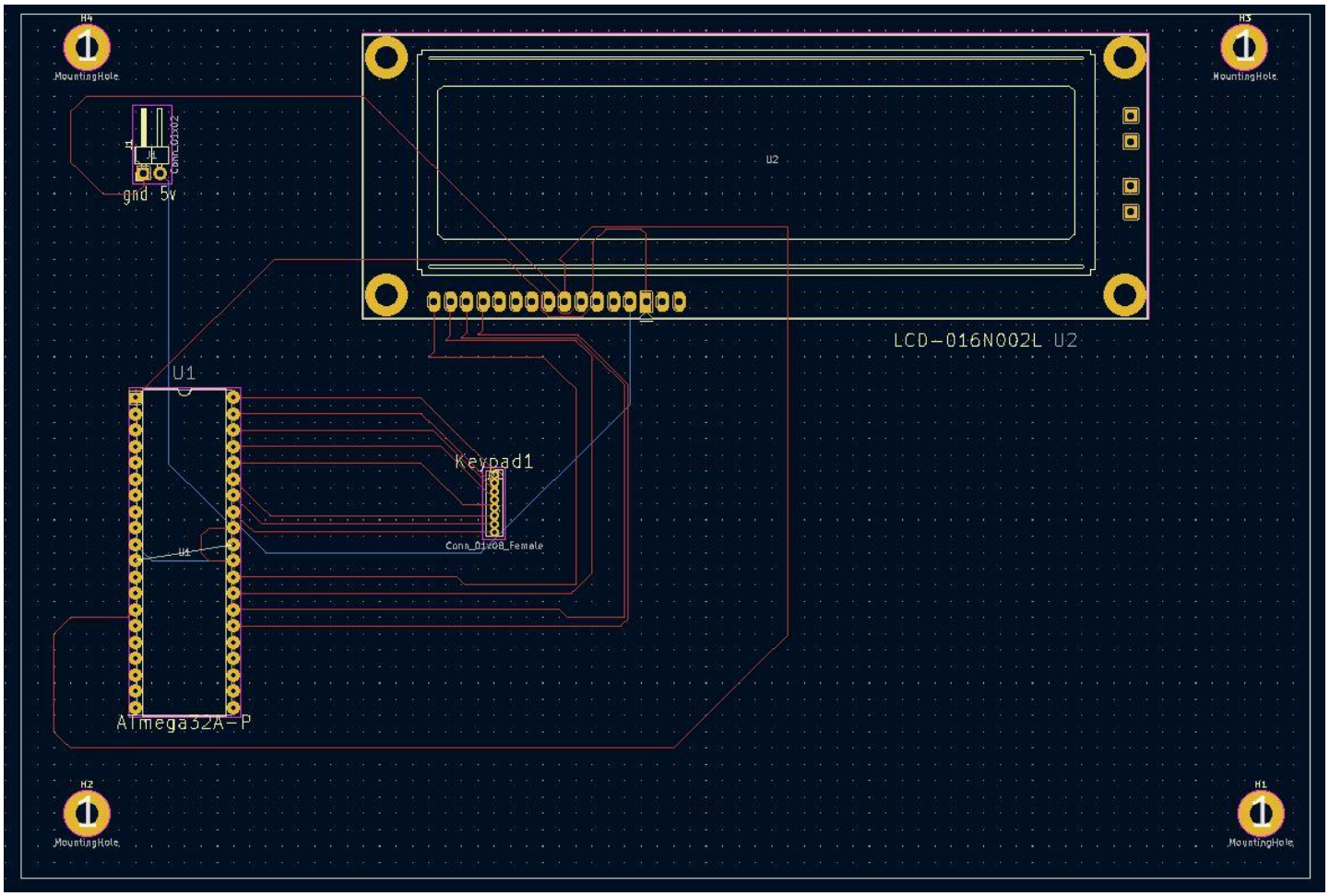


Data sheet of 4x4 keypad-https://cdn.sparkfun.com/assets/f/f/a/5/0/DS-16038.pdf

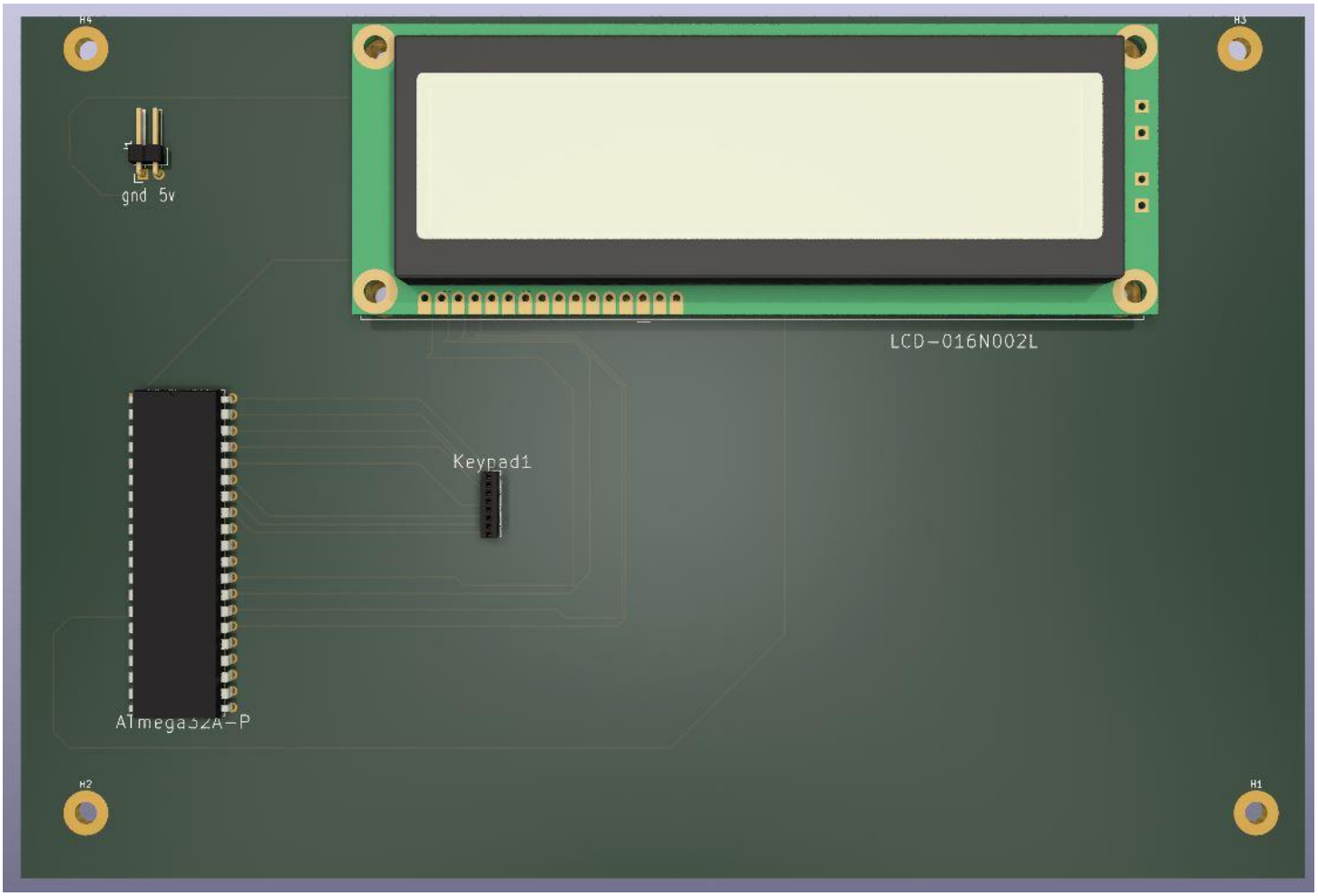
Schematic editor



PCB editor


H4
MountingHole
H3
MountingHole
J1
Conn_01x02
gnd 5v
U2
LCD-016N002L U2
U1
Keypad1
Conn_01x08_Female
U4
ATmega32A-P
H2
MountingHole
H1
MountingHole

3D viewer



Code for all my components combined (code for Master atmega32)

```
#ifndef F_CPU

#define F_CPU 16000000UL //freaquency of external crystal

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



//define interfacing pins between L293D and ATmega32

#define a1 (1<<PD4)

#define a2 (1<<PD5)

#define a3 (1<<PD6)

#define a4  (1<<PD7)



//For switch

#define SW (1<<PD3)    //Ir
#define SW2 (1<<PD2)   //ALL validation
```

```
//lcd and key board
#define LCD PORTC
#define EN 0
#define RS 0
#define RW 1

 int get_pn;
 //int get_cn;//number of cans
 char can_no[10];



unsigned char keypad();
int keypad_func(){
        unsigned char x;
        char no[10];
        int i=0;
        int i1=0;
        DDRA =0x0F; // make PA0 to PA3 = output and PA4 to PA7 = Input
        lcd_init();
        while(1){
                PORTA=0xF0; // make all 4 column 1 and all rows 0
                if(PINA!= 0xF0){
                        _delay_ms(5);
                        x=keypad();
                        if (x=='*')
                        {
                                lcdcmd(0x01);//clear lcd
                                lcdcmd(0x80);//set the curser to first line
                                lcd_Print("1-Fast Charge");
                                lcdcmd(0xc0);//set the curser to second line
                                lcd_Print("2-Ez Cash");
```

```
_delay_ms(100);
lcdcmd(0x01);
lcdcmd(0x80);//set the curser to first line
lcd_Print("3-Donate");

while(1){
        if(PINA!= 0xF0){
                x=keypad();
                //charge
                if(x=='1'){
                        _delay_ms(5);
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print(" ");
                        lcdcmd(0x01);//clear lcd
                        break;
                }
                //easy cash
                else if(x=='2'){
                        get_pn=1;//get phone no
                        _delay_ms(5);
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print(" ");
                        lcdcmd(0x01);//clear lcd
                        lcd_Print("Enter the Phone");
                        lcdcmd(0xc0);
                        lcd_Print("Number");
                        lcdcmd(0x80);
                        _delay_ms(100);
                        lcdcmd(0x01);
                        lcd_Print("Press = to");
                        lcdcmd(0xc0);
```

```
                    lcd_Print("Finish");
                    lcdcmd(0x80);
                    _delay_ms(100);
                    lcdcmd(0x01);
                    break;
                }
                //donate
                else if(x=='3'){

                }

            }
        }
}
else if (x=='=')
{
        get_pn=0;
        lcdcmd(0x01);
        lcd_Print("Entered phone NO.:");
        lcdcmd(0xc0);
        lcd_Print(&no);//print the no
        lcdcmd(0x80);
        //lcd_Print(&can_no);//print the no
        int phone_no= atoi(no);//phone nimber in int
        _delay_ms(100);
        lcdcmd(0x01);
        _delay_ms(100);
        lcdcmd(0x01);
        i=0;
        return 0;
```

```
                    }
                    else
                    {

                        lcddata(x);
                        if(get_pn==1){
                            no[i]=x;
                            i++;
                        }
                        else{
                            can_no[i1]=x;
                            i1++;
                        }

                    }
                }
            }

}
void lcdcmd(unsigned char cmd){
        PORTB &=~(1<<RS); //RS=0 for command (1111 1110)
        PORTB &=~(1<<RW); //RW=0 for write (1111 1101)
        LCD= cmd & 0xF0;  //Send upper nibble
        PORTD|=(1<<EN); //EN=1 for H to L pulse
        _delay_ms(1);
        PORTD &= ~(1<<EN); // EN=0 for H to L pulse

        LCD = cmd<<4; // send low nibble;
        PORTD |=(1<<EN); //EN=0for H to L pulse
        _delay_ms(1);
```

```
        PORTD &= ~(1<<EN);
}

void lcd_Print (char*str){
        int i;
        for(i=0;str[i]!=0;i++)                  /* Send each char of string till the NULL */
        {
                lcddata(str[i]);
        }
}



void lcddata(unsigned char data){
        PORTB |= (1<<RS); //RS=1 for data
        PORTB &= ~(1<<RW); //RW=0 for write
        LCD= data & 0xF0;  //Send upper nibble
        PORTD|= (1<<EN); //EN=1 for H to L pulse
        _delay_ms(2);
        PORTD &= ~(1<<EN); // EN=0 for H to L pulse

        LCD = data<<4; // send low nibble;
        PORTD |=(1<<EN); //EN=1 for H to L pulse
        _delay_ms(2);
        PORTD &= ~(1<<EN);

}

void lcd_init(){
        DDRC= 0xFF;
        DDRD |=(1<<EN); //Define EN pin  as output
        DDRB=0b11110111; //0xff define RS and  RW pin as output
```

```
        PORTD&= ~(1<<EN); //initialize EN=0
        lcdcmd(0x33);
        lcdcmd(0x32);
        lcdcmd(0x28); // LCD in 4 bit mode
        lcdcmd(0x0E); //display on cursor on
        /*lcddata(72);
        lcddata(69);
        lcddata(76);
        lcddata(76);
        lcddata(79);*/
        lcd_Print("WELCOME");
        _delay_ms(40);
        lcdcmd(0x01); //clear LCD
        lcd_Print("Enter the Number");
   lcdcmd(0xc0);//set the curser to second line
        lcd_Print("of cans and");
        _delay_ms(40);
        lcdcmd(0x01); //clear LCD
        lcdcmd(0x80);//set the curser to first line
        lcd_Print("Press + to start");
        _delay_ms(40);
         lcdcmd(0xc0);//set the curser to second line
         lcd_Print("the process : ");
          //lcdcmd(0x80);

}

unsigned char keypad(){
        PORTA=0b11111110; // make 1st row 0
        if((PINA & (1<<PINA4))==0){
                _delay_ms(3);
```

```
            return '7';
            }else if((PINA & (1<<PINA5))==0){
            _delay_ms(3);
            return '8';
            }else if((PINA & (1<<PINA6))==0){
            _delay_ms(3);
            return '9';
            }else if((PINA & (1<<PINA7))==0){
            _delay_ms(3);
            return '/';
}
PORTA=0b11111101;
if((PINA & (1<<PINA4))==0){
            _delay_ms(3);
            return '4';
            }else if((PINA & (1<<PINA5))==0){
            _delay_ms(3);
            return '5';
            }else if((PINA & (1<<PINA6))==0){
            _delay_ms(3);
            return '6';
            }else if((PINA & (1<<PINA7))==0){
            _delay_ms(3);
            return '+';
}
PORTA=0b11111011;
if((PINA & (1<<PINA4))==0){
            _delay_ms(3);
            return '1';
            }else if((PINA & (1<<PINA5))==0){
            _delay_ms(3);
```

```
                return '2';
                }else if((PINA & (1<<PINA6))==0){
                _delay_ms(3);
                return '3';
                }else if((PINA & (1<<PINA7))==0){
                _delay_ms(3);
                return '-';
        }
        PORTA=0b11110111;
        if((PINA & (1<<PINA4))==0){
                _delay_ms(3);
                return 'C';
                }else if((PINA & (1<<PINA5))==0){
                _delay_ms(3);
                return '0';
                }else if((PINA & (1<<PINA6))==0){
                _delay_ms(3);
                return '=';
                }else if((PINA & (1<<PINA7))==0){
                _delay_ms(3);
                return '*';
        }
}
```

```
int Stepper_motor(){
        //Set a1, a2, a3, a4 as output

        DDRD |= a1 | a2 | a3| a4;

        //Set input & pull-up resistor

        DDRD &= ~SW;

        PORTD |= SW;

        DDRD &= ~SW2;

        PORTD |= SW2;

        while(1){

                if((PIND & SW) && (PIND & SW2)){

                        while(1){

                                if((!(PIND & SW2))){
                                        while(1){
                                                if((!(PIND & SW))){
                                                        PORTD |= a1;
                                                        PORTD &= ~a2;
                                                        PORTD &= ~a3;
                                                        PORTD &= ~a4;
                                                        _delay_ms(20);
```

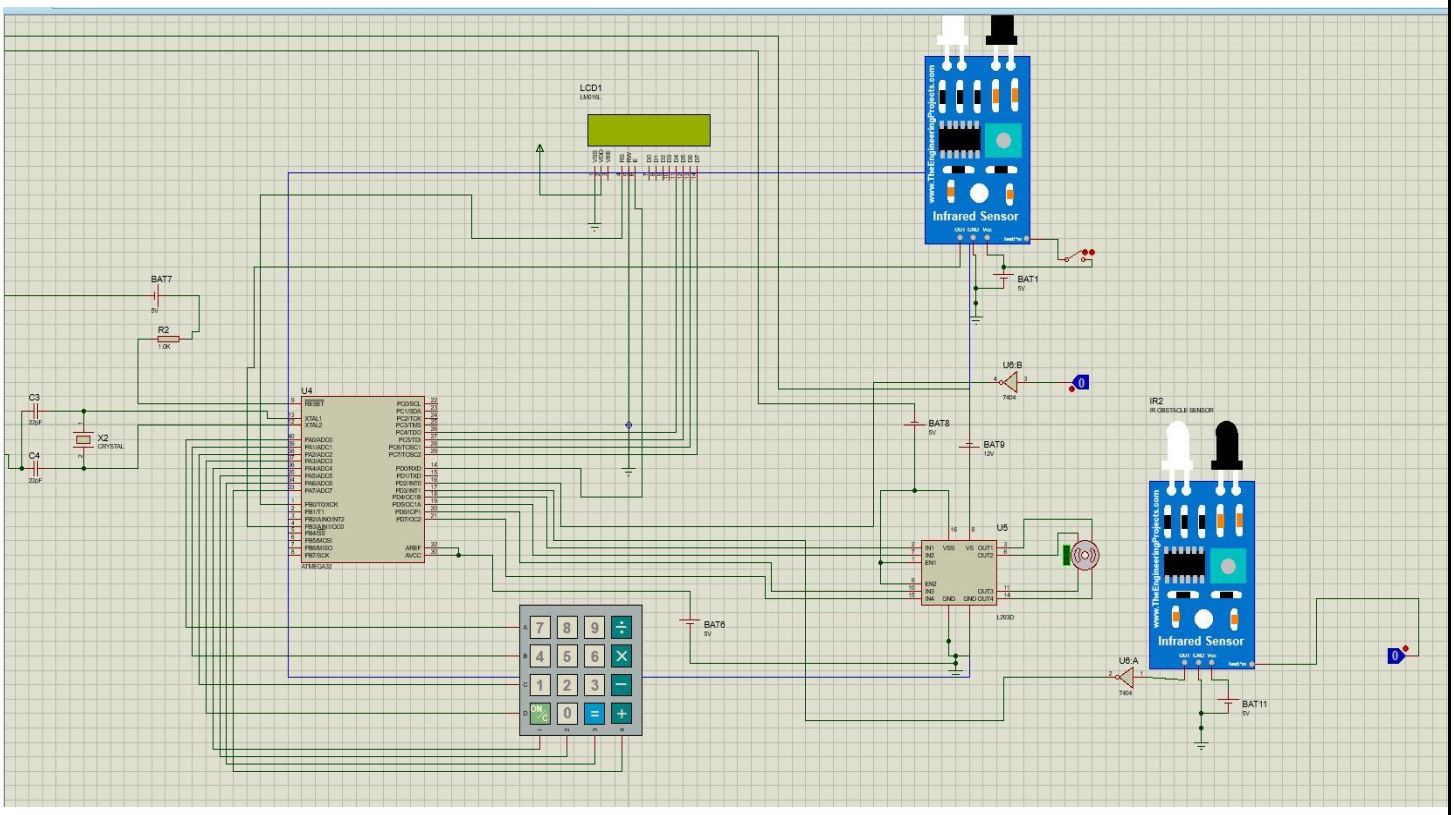
```
                PORTD |= a1;
                PORTD &= ~a2;
                PORTD  &= ~ a3;
                PORTD |= a4;
                _delay_ms(4);

                PORTD &= ~a1;
                PORTD &= ~a2;
                PORTD &= ~a3;
                PORTD |= a4;
                _delay_ms(20);

                PORTD |= a1;
                PORTD &= ~a2;
                PORTD &= ~ a3;
                PORTD |= a4;
                _delay_ms(4);

                PORTD |= a1;
                PORTD &= ~a2;
                PORTD &= ~a3;
                PORTD &= ~a4;
                _delay_ms(20);
                return 0;
            }
        }

    }
```

```
else if(!(PIND & SW)){

        PORTD |= a1;
        PORTD &= ~a2;
        PORTD &= ~a3;
        PORTD &= ~a4;
        _delay_ms(20);

        PORTD |= a1;
        PORTD &= ~a2;
        PORTD |= a3;
        PORTD &= ~a4;
        _delay_ms(4);


        PORTD &= ~a1;
        PORTD &= ~a2;
        PORTD |= a3;
        PORTD &= ~a4;
        _delay_ms(20);


        PORTD |= a1;
        PORTD &= ~a2;
        PORTD |= a3;
        PORTD &= ~a4;
        _delay_ms(4);
```

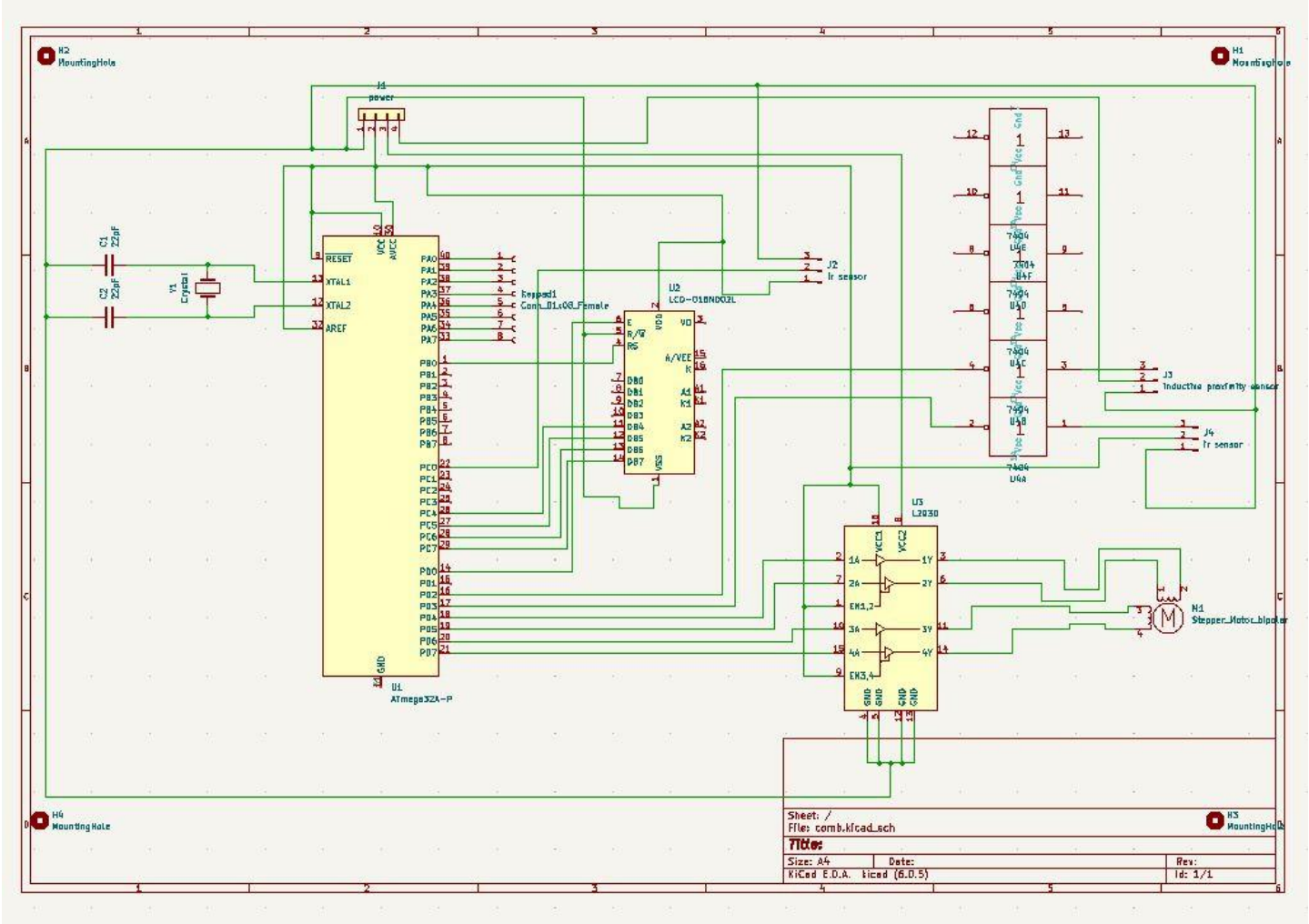
```
                                        PORTD |= a1;
                                        PORTD &= ~a2;
                                        PORTD &= ~a3;
                                        PORTD &= ~a4;
                                        _delay_ms(20);
                                        return 0;
                                    }

                                }

                        }
                }

}
int ir2_func(){
        while(1){

                if(PINB & 0b00001000){
                        lcdcmd(0x01); //clear LCD
                        lcdcmd(0x80);//set the curser to first line
                        lcd_Print("Discard bin is");
                        lcdcmd(0xc0);//set the curser to second line
                        lcd_Print("full");
                        lcdcmd(0x80);//set the curser to first line
                        _delay_ms(50);
                        lcdcmd(0x01); //clear LCD
                        lcd_Print("Clear the");
                        lcdcmd(0xc0);//set the curser to second line
                        lcd_Print("discard bin");
                        lcdcmd(0x80);//set the curser to first line
                }
```

```
                else{
                        return 0;
                }

        }

}
int main(void){
   while(1){
                keypad_func();
                int can_number= atoi(can_no);
                for(int k=0;k<can_number;k++){
                        ir2_func();
                        Stepper_motor();
                }

        }

        return 0;
}
```

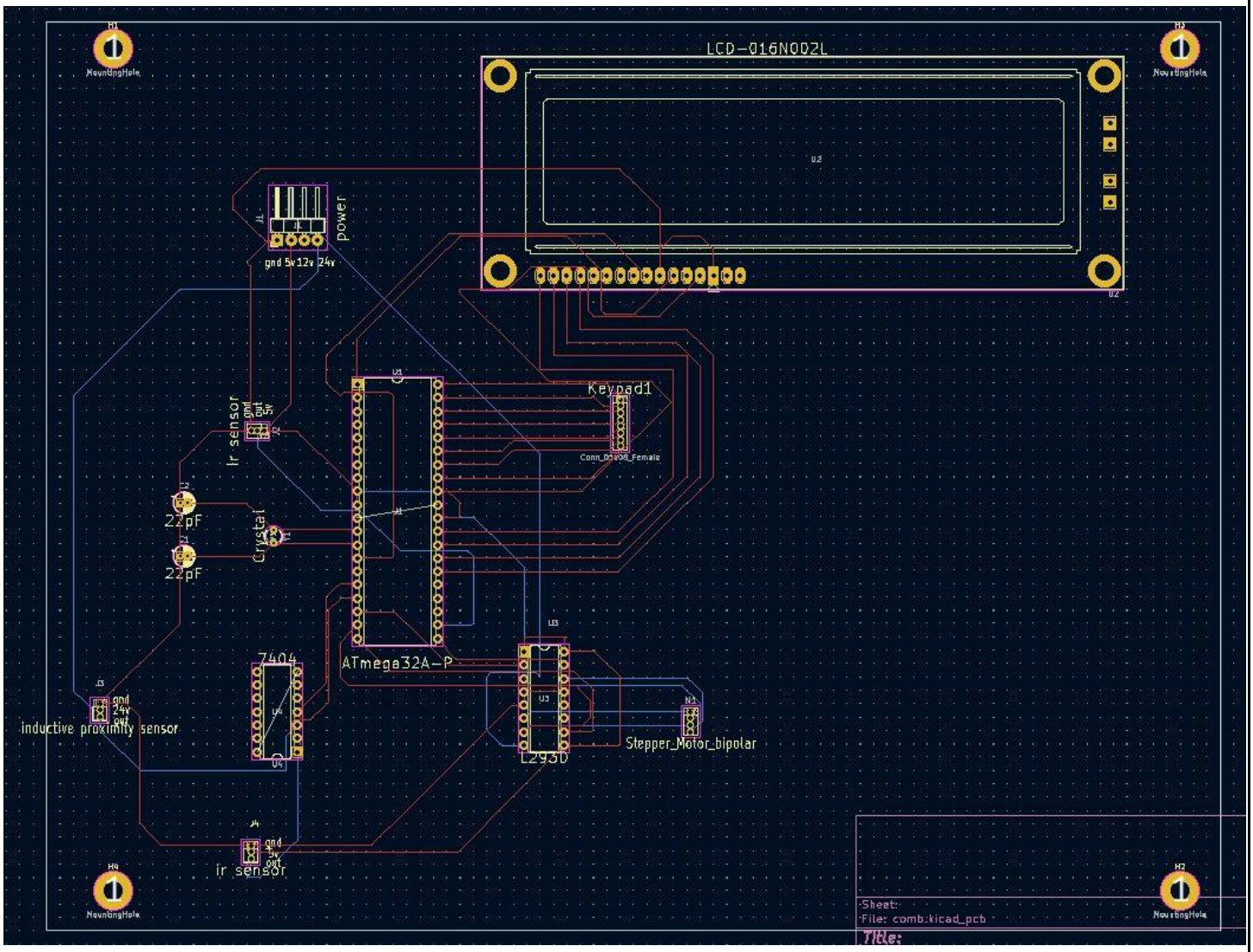
Circuit diagram for all my components combined (code for Master atmega32)



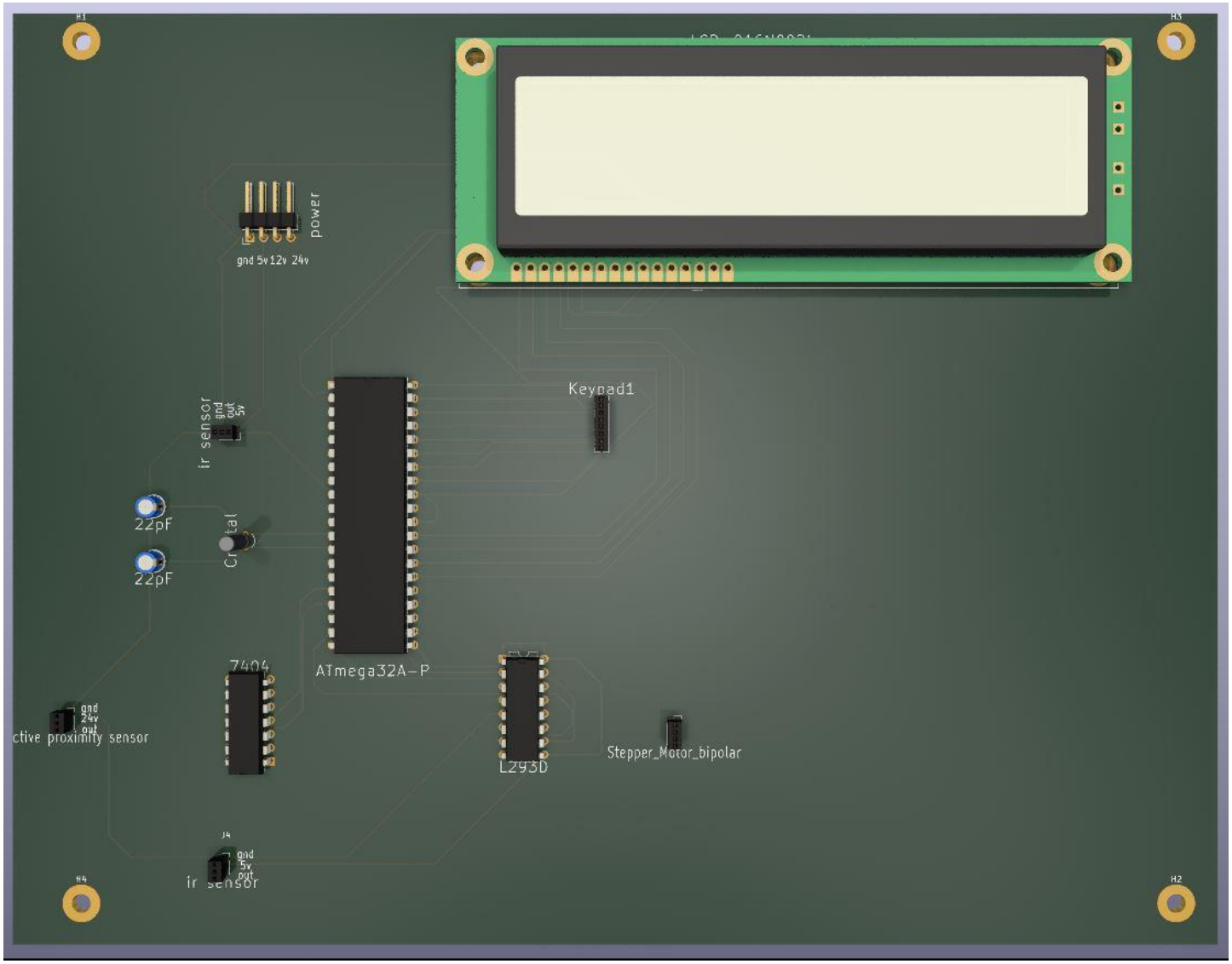
Schematic editor



PCB editor


LCD-016N002L
power
gnd 5v 12v 24v
ir sensor
gnd
out
5v
22pF
22pF
Crystal
Keypad1
ATmega32A-P
7404
L293D
Stepper_Motor_bipolar
inductive proximity sensor
gnd
24v
out
ir sensor
gnd
5v
out
Sheet:
File: comb.kicad_pcb
Title:

3D viewer


power
gnd 5v 12v 24v
ir sensor
gnd
out
5v
Keypad1
22pF
22pF
Crystal
ATmega32A-P
7404
L293D
Stepper_Motor_bipolar
gnd
24v
out
ctive proximity sensor
J4
gnd
5v
out
ir sensor

3.Student name-: Basnayake S.B. – 204020V

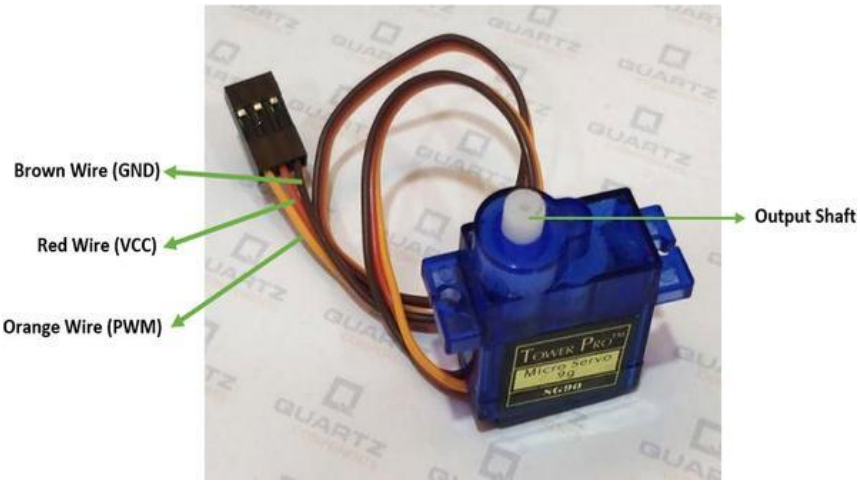
Servo motor (SG90)

Servo Motor is used to rotate the can in order to detect whether the can is valid or not. The shape of the object is detected with the combination of Ultrasonic sensors and the rotation of the servo motor. When the microcontroller signals some type of object have been detected I programmed the servo motor to rotate 180 degrees.

Servo motors are used to perform precise rotations.

Servo motors usually have a rotation angle that varies between 0 and 180 degrees. The rotation angle of the servo motor is controlled by applying a PWM signal to it.

Usually there are 3 pins in a servo.



VCC (Red) – connects +VCC supply to the pin.

GROUND (Brown) – connects to the ground.

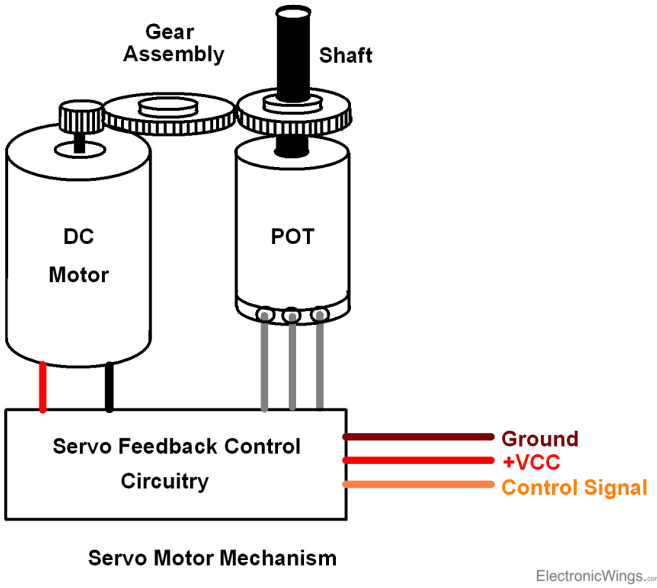
Control Signal (Orange) – connects PWM of 20ms(50Hz) period to this pin.

Power requirement

SG90 servo motor requires a voltage supply of 4.8V (~5V). And it can handle a maximum voltage of 6V and a maximum current of 250mA

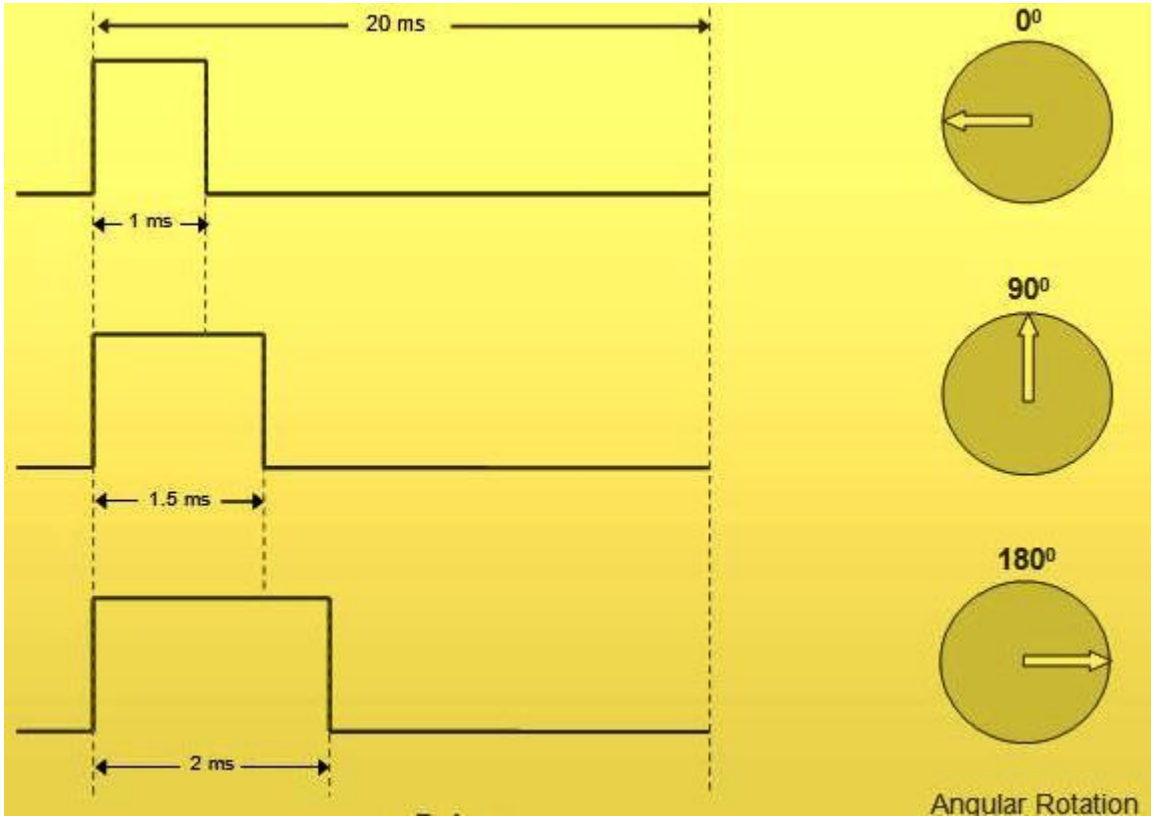
Other Specifications

- Torque :2.5kg/cm
- Operating speed :0.1s/60
- Rotation : 0-180 degrees
- Weight : 9g

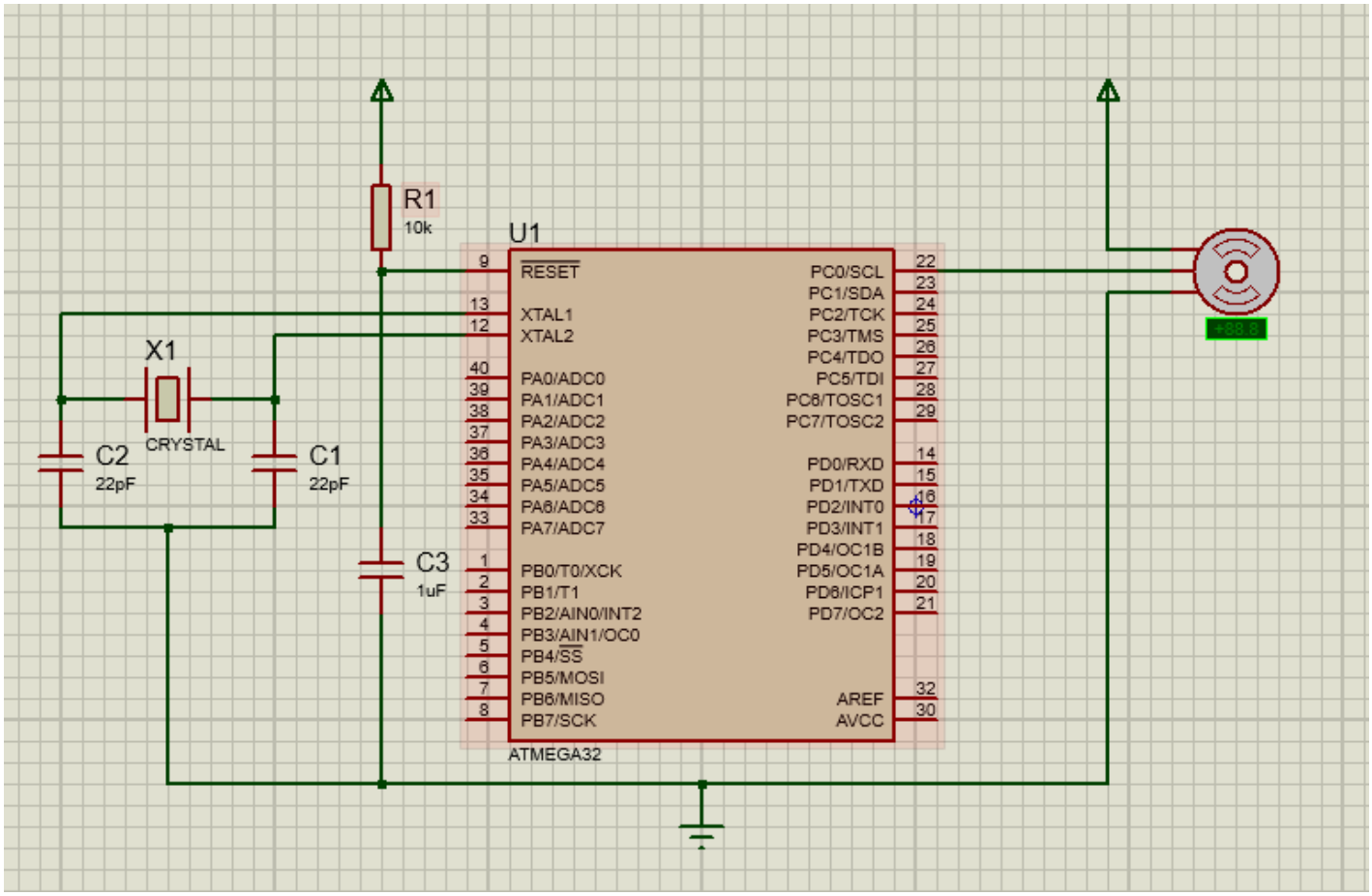


Servo Motor Mechanism

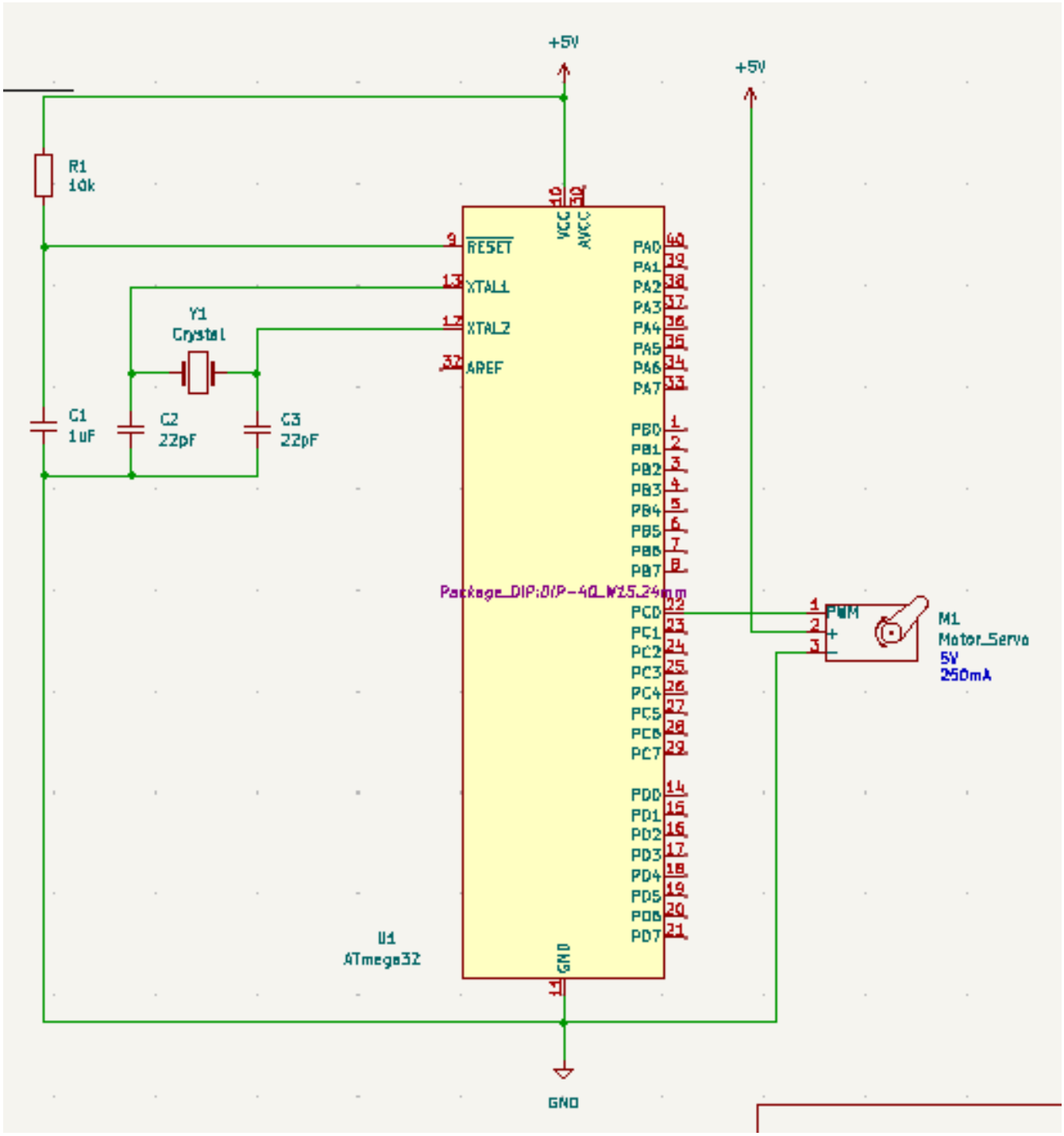
The user can changed the rotation angle of the servo motor by changing the pulse width of the PWM signal. The below image shows the approximate duty cycle and it's related rotation of the shaft.



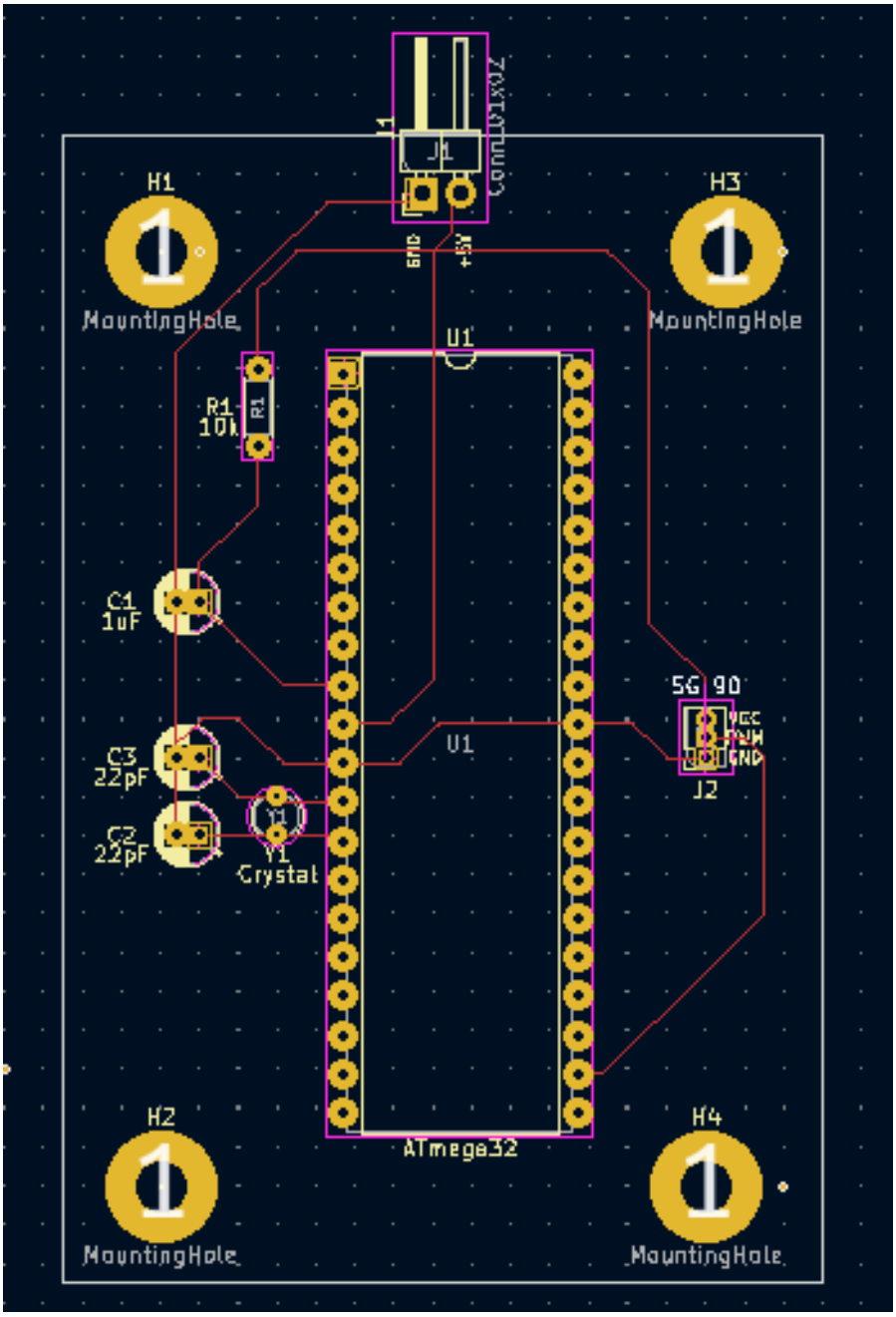
Circuit Diagram for the Servo Motor

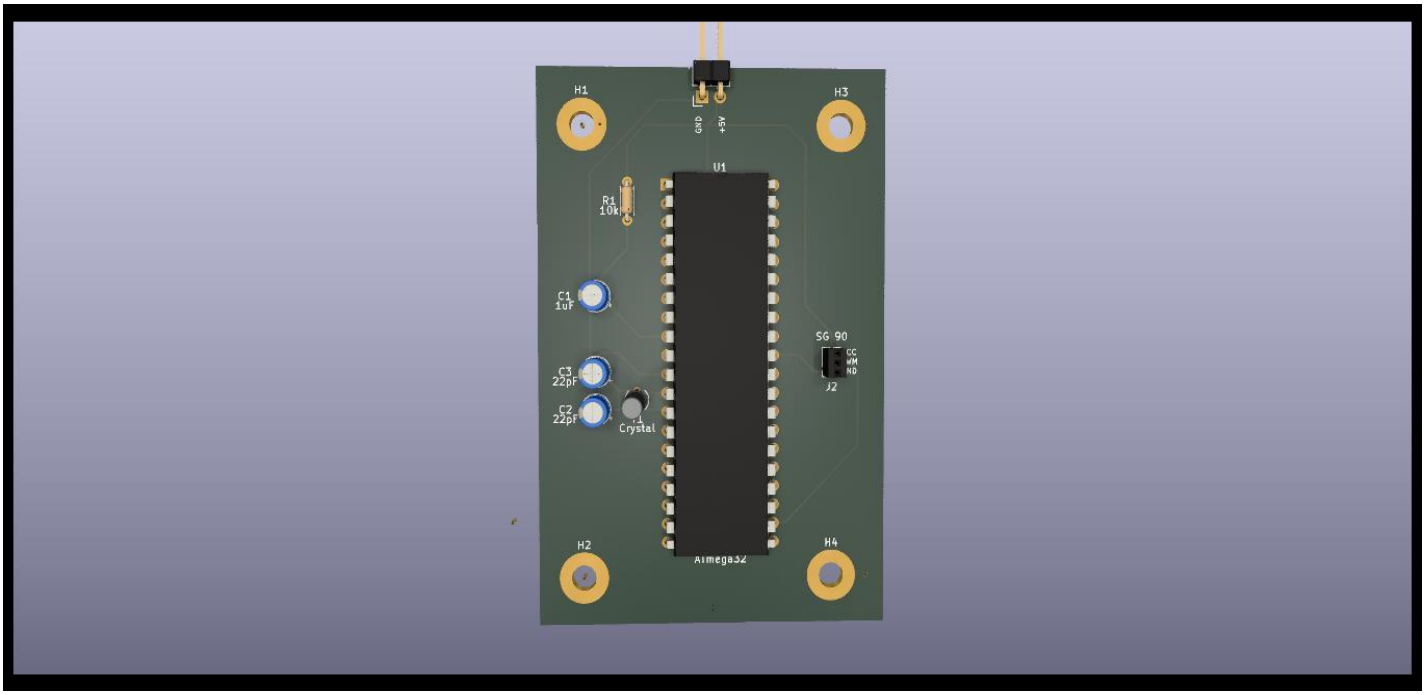


Schematic Diagram



PCB Design


J1
Conn_01x02
GND
+5V
H1
MountingHole
H3
MountingHole
U1
R1
10k
C1
1uF
C3
22pF
C2
22pF
Y1
Crystal
U1
ATmega32
SG 90
VCC
PWM
GND
J2
H2
MountingHole
H4
MountingHole


H1
H3
GND
+5V
U1
R1
10k
C1
1uF
SG 90
CC
WM
GD
J2
C3
22pF
C2
22pF
Crystal
H2
H4
Atmega32

Servo motor code

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

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

void servo()
{
    DDRC = 0x01; //Makes PC0 output pin
       PORTC = 0x00;
       for (int i=0;i<4;i++)
       {
              //Rotate Motor to 0 degree
              PORTC = 0x01;
              _delay_us(1000);
              PORTC = 0x00;

              _delay_ms(1000);

              //Rotate Motor to 90 degree
              PORTC = 0x01;
              _delay_us(1500);
              PORTC = 0x00;

              _delay_ms(1000);

              //Rotate Motor to 180 degree
              PORTC = 0x01;
              _delay_us(2000);
              PORTC = 0x00;

              _delay_ms(1000);
       }
}

int main(void)
{
       servo();
}
```

SG90 datasheet- http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

GSM module (SIM900A)

- There are many GSM modules are in the market
    - Ex: - SIM900, SIM700, SIM800, SIM808, SIM5320
- SIM 900A can work with any GSM Network Operator.
- But in order to connect to a cellular network the modem requires a SIM card provided by a network provider.
- SIM 900A is a plug and play modem with RS232 serial communication support.

Power Requirement

- SIM900A requires an external supply of ~5V and can draw up to ~2A at its peak. And the minimum current requirement is 1A.

Indicators

- It has two LED indicators as,

ON – It indicates that the module is powered and switched on.

Network LED – When module is powered up, network LED blink every second and after registration it will start to blink after every 3 seconds. And this shows that the modem is registered with the network.

GSM Module is used for the rewarding process. I programmed the GSM module to

1. Give money according to the value of the cans
2. Donate entire money to the Little Hearts Foundation

The user has to choose between these 3 options including a fast charging option.

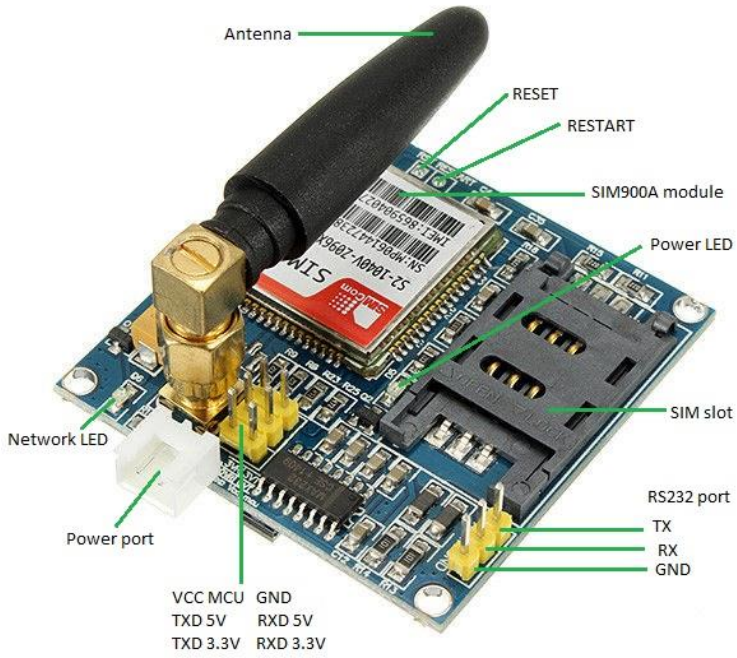
If the user selects the option to get the money he/she have to enter a mobile phone number. And according to the phone number entered he/she will receive the money using the **ezCash** method.

Picture of the GSM module and all pins

AT commands are used to work with supported features of SIM 900A.

After sending the command a respond will be send by the GSM module. And every time user will receive the response in the form of

<CL><LF>response<CR><LF>



Given below are the AT commands used when building our project.(<CR><LF> are removed intentionally.

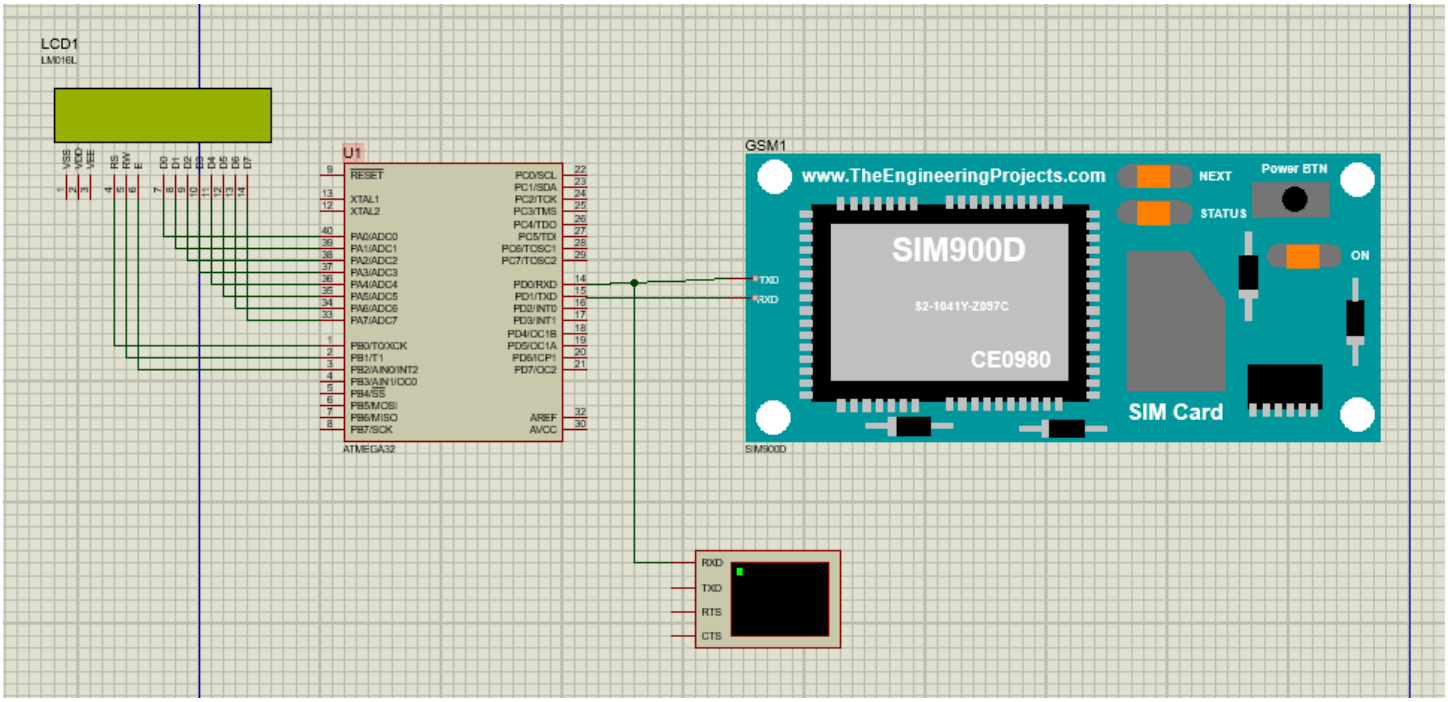
Command	Function	Expected Response
ATE0	Set echo mode	OK
AT+CMGF=<index> Index - 0: PDU 1: Text	Select message format	OK
AT+CMGS=”Mobile Number”	Send message	> ”Type message here” Press ‘Ctrl+z’ to end message or ‘ESC’ to exit without sending OK
ATD**MobileNumber**;	Calling	OK

The GSM module uses USART communication to communicate with a microcontroller.

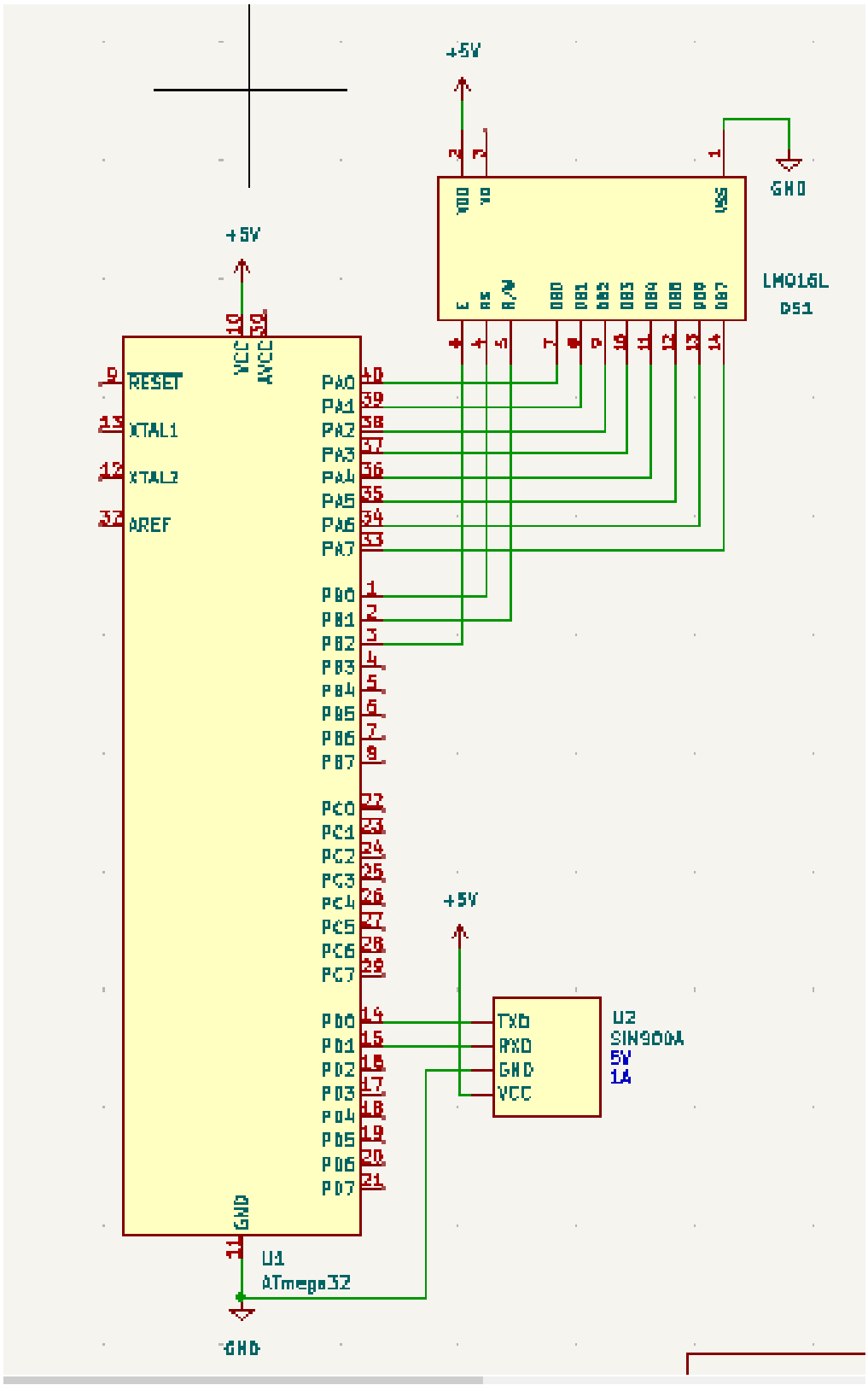
TXD 5V pin of the GSM module connects to the pin 14(RXD) of the ATMega 32.

RXD 5V pin of the GSM module connects to the pin 15(TXD) of the ATMega 32.

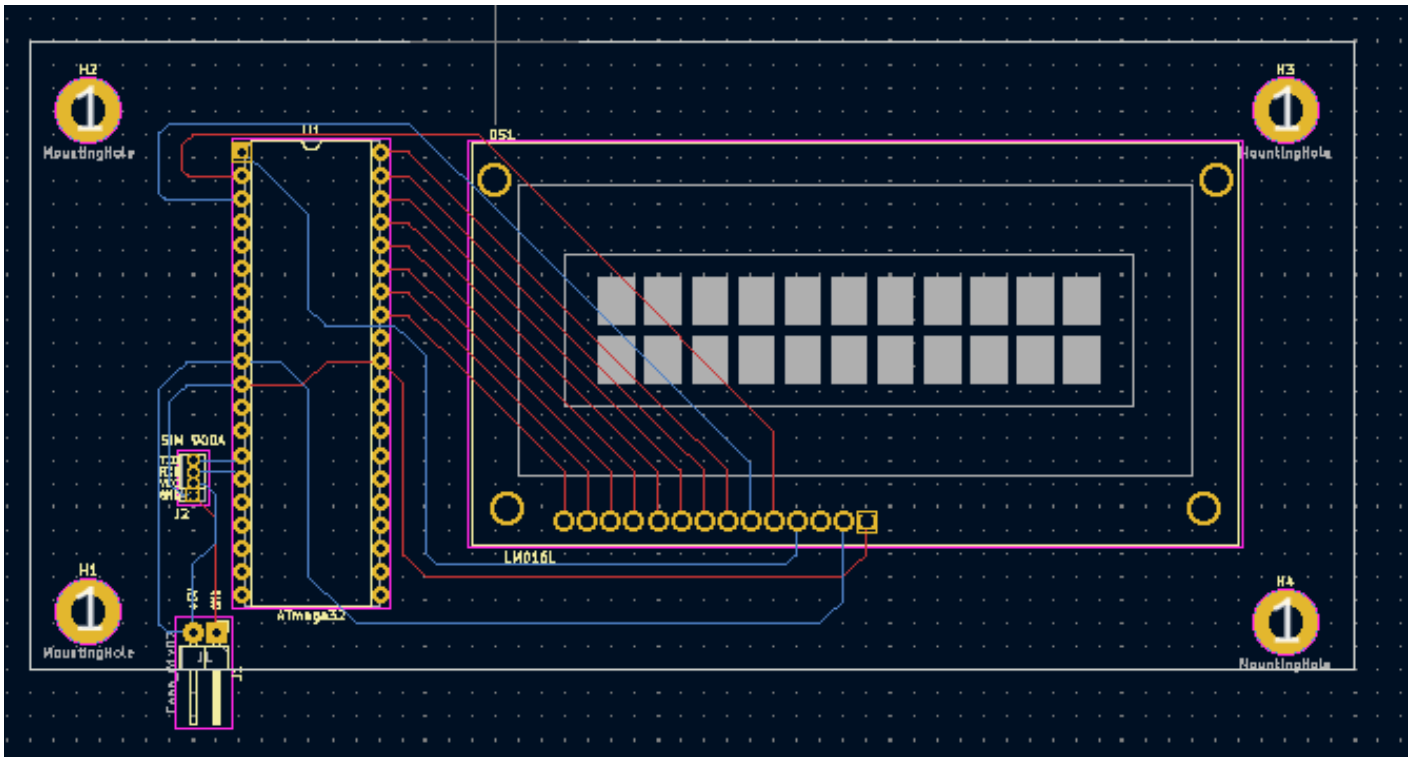
Circuit Diagram for the GSM

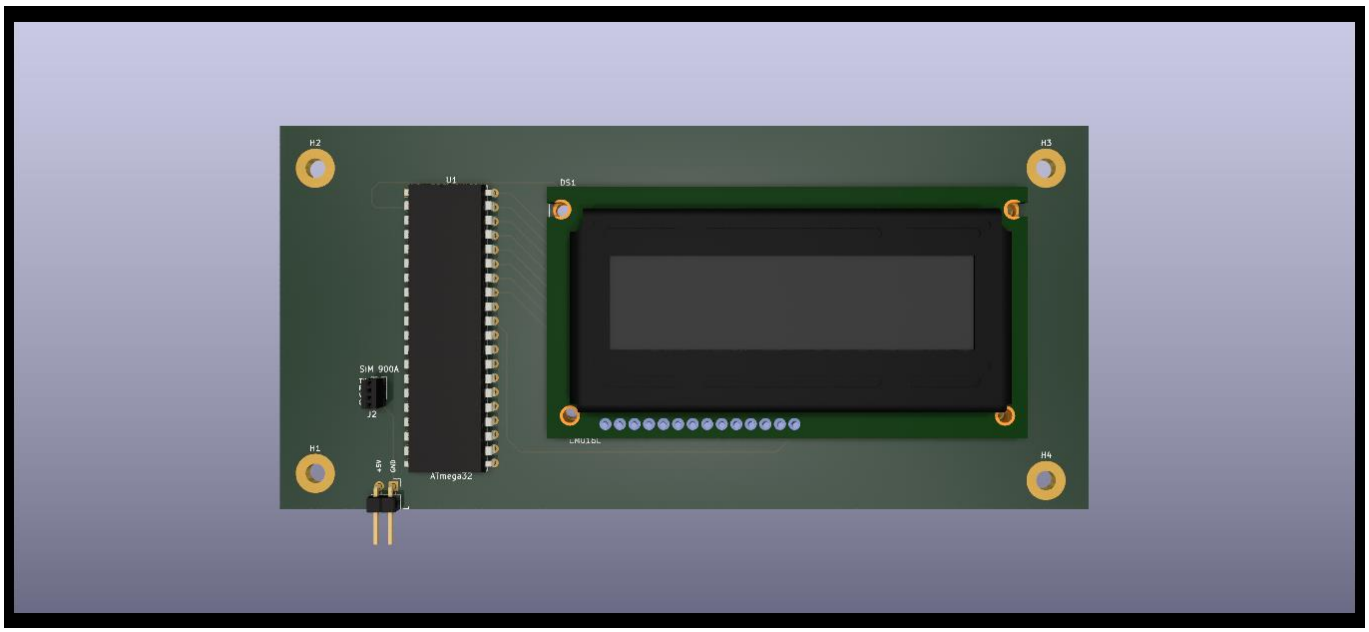


Schematic Diagram



PCB Design


H2
MountingHole
H3
MountingHole
U1
DS1
SIM 900A
J2
LM016L
H1
MountingHole
ATmega32
J1
H4
MountingHole


H2
H3
U1
DS1
SIM 900A
J2
H1
H4
ATmega32

Code

USART_Interrupt c file

```
include "USART_Interrupt.h"


void USART_Init(unsigned long 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 prescale value */
        UBRRH = (BAUD_PRESCALE >> 8);                                  /* Load
UBRRH with upper 8 bit of prescale value */
}


char USART_RxChar()                                                 /*
Data receiving function */
{
        while (!(UCSRA & (1 << RXC)));                          /* Wait until new
data receive */
        return(UDR);                                                   /* Get and
return received data */
}


void USART_TxChar(char data)                                    /* Data
transmitting function */
{
        UDR = data;                                                         /*
Write data to be transmitting in UDR */
```

```
        while (!(UCSRA & (1<<UDRE)));                    /* Wait until data
transmit and buffer get empty */
}

void USART_SendString(char *str)                         /* Send string of USART
data function */
{
        int i=0;
        while (str[i]!=0)
        {
                USART_TxChar(str[i]);                    /* Send
each char of string till the NULL */
                i++;
        }
}
```

USART_INTERRUPT file

```
#ifndef USART_INTERRUPT_H_
#define USART_INTERRUPT_H_
#define F_CPU 1000000UL
#include <avr/io.h>                                          /* Include AVR std.
library file */
#include <avr/interrupt.h>
#include "USART_Interrupt.h"

#define BAUD_PRESCALE (((F_CPU / (BAUDRATE * 2UL))) - 1)      /* Define prescale value
*/

void USART_Init(unsigned long);                             /* USART initialize function */
char USART_RxChar();                                        /* Data receiving function */
void USART_TxChar(char);                                    /* Data transmitting function
*/
void USART_SendString(char*);                               /* Send string of USART data
function */



#endif /* USART_INTERRUPT_H_ */
```

Lcd.c file

```
#include <stdio.h>
#include <math.h>

#include <avr/io.h>

#define lcdport PORTA

#define _signal PORTB

#define en PB2

#define rw PB1

#define rs PB0



#define LCD_Data_Dir DDRA                    /* Define LCD data port direction
*/
#define LCD_Command_Dir DDRB                 /* Define LCD command port direction
register */
#define LCD_Data_Port PORTA                  /* Define LCD data port */
#define LCD_Command_Port PORTB
#define EN PB2                                       /* Define Enable signal
pin */
#define RW PB1                                       /* Define Read/Write
signal pin */
#define RS PB0

char key;
```

```
void lcdcmd(unsigned char cmd);//lcd commands

void lcdint();//initialize lcd

void lcddata(unsigned char data);

void lcd_string(const unsigned char *str, unsigned char length);//display in lcd

int charToInt(char c);//character to integer

int displayKey();//save input

void lcd_clear();//lcd clear

void lcd_line_one();//lcd line one

void lcd_line_two();//lcd line two



void lcdint()
{
        lcdcmd(0x38);

        _delay_ms(1);

        lcdcmd(0x01);

        _delay_ms(1);
```

```
        lcdcmd(0x0E);

        _delay_ms(1);
}


void lcdcmd(unsigned char x)
{
        lcdport=x;

        _signal=(0<<rs)|(0<<rw)|(1<<en);

        _delay_ms(1);

        _signal=(0<<rs)|(0<<rw)|(0<<en);

        _delay_ms(10); /* original-50 */
}

void lcddata(unsigned char data)
{
        lcdport= data;

        _signal= (1<<rs)|(0<<rw)|(1<<en);

        _delay_ms(1);

        _signal= (1<<rs)|(0<<rw)|(0<<en);

        _delay_ms(10); /* original-50 */
```

```
}


void lcd_string(const unsigned char *str, unsigned char length)
{
	for(int i=0; i<length; i++)
	{
		lcddata(str[i]);
	}
}

int charToInt(char c){
	int num = 0;
	num = c - '0';

	return num;
}


void lcd_clear()
{
	lcdcmd(0x01);
}

void lcd_line_one()
{
	lcdcmd(0x80);
}

void lcd_line_two()
```

```
{
        lcdcmd(0xC0);
}


void LCD_Char (char char_data)                          /* LCD data write
function */
{
        LCD_Data_Port = char_data;                      /* Write data to
LCD data port */
        LCD_Command_Port &= ~(1<<RW);                   /* Make
RW LOW (Write) */
        LCD_Command_Port |= (1<<EN)|(1<<RS);            /* Make RS HIGH (data
reg.) and High to Low transition on EN (Enable) */
        _delay_us(1);
        LCD_Command_Port &= ~(1<<EN);
        _delay_ms(1);                                   /* Wait
little bit */
}
```

Main.c file

```
#define F_CPU 16000000UL                                    /* define Clock
Frequency */
#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <util/delay.h>
#include "lcd.c"                          /* include 16x2 LCD Header file */
#include "USART_Interrupt.c"                        /* include USART Header file */

#define SREG   _SFR_IO8(0x3F)

void GSM_Begin();
void GSM_Dialling();
void GSM_Response();
void GSM_Response_Display();
void GSM_Send_Msg(char* ,char*);



char buff[160];                                                 /* buffer to store
responses and messages */
char status_flag = 0;                                   /* for checking any new message
*/
volatile int buffer_pointer;

int price=2;                              /*price given to a single can*/
int cans;                                       /* amount of cans inserted by the user */
```

```
int amount;                                          /* total value user will receive */
char mob_no[11]="0775499920";      /*user's phone number */




int main(void)
{
        char lh[10];
        DDRA=0xff;


        DDRB=0x07;



        buffer_pointer = 0;


        lcdint();
        USART_Init(9600);                                    /* initialize USART
communication */
        sei();
        lcd_line_one();
        lcd_string("GSM Initializing",16);
        _delay_ms(90); //original-3000
        lcd_clear();
        lcd_line_one();
        lcd_string("AT",2);
        GSM_Begin();                                                  /* check
GSM responses and initialize GSM */
        lcd_clear();
```

```
        //amount=cans*price;
        amount=1000;
        sprintf(lh,"LH %d",amount);
        GSM_Send_Msg("77110",lh);
        memset(lh,0,strlen(lh));          /* clear lh array */

        lcd_string(".....Sending....",16);  /* lcd print function */
        _delay_ms(70); //original-7000
        lcd_clear();   /* function to clear lcd */

        lcd_string("Sent to Little",14);
        _delay_ms(30); // //original-500
        lcd_clear();
        lcd_string("Hearts",6);
        _delay_ms(30); //original-500
        lcd_clear();

        GSM_Dialling();
        lcd_string("Reload send",11);
}

void GSM_Begin()
{

        while(1)
        {
                lcdcmd(0xc0);
                USART_SendString("ATE0\r");                    /* send ATE0 to check
module is ready or not */
                _delay_ms(30); //original-500
```

```
            if(strstr(buff,"OK"))
            {
                GSM_Response();             /* get Response */
                memset(buff,0,160);
                break;
            }
            else
            {
                lcd_line_one();
                lcd_string("Error",5);
            }
        }
        _delay_ms(50);  // original-1000

        lcd_clear();
        lcd_line_one();
        lcd_string("Text Mode",9);
        lcdcmd(0xc0);
        USART_SendString("AT+CMGF=1\r");/* select message format as text */
        GSM_Response();
        _delay_ms(40);  // original-1000
}



/* ISR routine to save responses/new message */
ISR(USART_RXC_vect)
{
        buff[buffer_pointer] = UDR;                                     /* copy
UDR(received value) to buffer */
        buffer_pointer++;
```

```
        status_flag = 1;                                        /* flag for new
message arrival */
}



void GSM_Send_Msg(char* num,char *sms)
{
        char sms_buffer[35];
        buffer_pointer=0;
        sprintf(sms_buffer,"AT+CMGS=\"%s\"\r",num);
        USART_SendString(sms_buffer);           /*send command
AT+CMGS="Mobile No."\r */
        _delay_ms(20); // original-200
        while(1)
        {
                if(buff[buffer_pointer]==0x3e)          /* wait for '>' character*/
                {
                        buffer_pointer = 0;
                        memset(buff,0,strlen(buff));                    /* emptying "buff"
*/
                        USART_SendString(sms);          /* send msg to 77110 */
                        USART_TxChar(0x1a);             /* send Ctrl+Z then only
message will transmit*/
                        break;
                }
                buffer_pointer++;
        }
        _delay_ms(30);  // original-300
        buffer_pointer = 0;
        memset(buff,0,strlen(buff));                    /* emptying "buff" */
        memset(sms_buffer,0,strlen(sms_buffer));    /* emptying "sms_buffer" */
}
```

```
void GSM_Dialling()
{
	char call[35];
	sprintf(call,"ATD#111#2#%s#%d#1234#;\r",mob_no,amount);  /* build command */
	USART_SendString(call);                                  /* send command ATD<Dial_code> */

}

void GSM_Response()
{
	unsigned int timeout=0;
	int CRLF_Found=0;  /* character elements found CL & LF */
	char CRLF_buff[2];  /* store the CL & LF) */
	int Response_Length=0;
	while(1)
	{
		if(timeout>=60000)                                   /*if timeout occur then return */
		return;
		Response_Length = strlen(buff);                      /* getting the length of the response */
		if(Response_Length)                                  /* checking the availability of the response */
		{
			_delay_ms(2);
			timeout++;
			if(Response_Length==strlen(buff))
			{
				for(int i=0;i<Response_Length;i++)
```

```
                                {
                                        memmove(CRLF_buff,CRLF_buff+1,1);       /*
changing the memory location of the response */
                                        CRLF_buff[1]=buff[i];
                                        if(strncmp(CRLF_buff,"\r\n",2))
                                        {
                                                if(CRLF_Found++==2)
        /* search for \r\n in string */
                                                {
                                                        GSM_Response_Display();         /*
display response */
                                                        return;
                                                }
                                        }

                                }
                                CRLF_Found = 0;

                        }

                }
                _delay_ms(1);
                timeout++;
        }
        status_flag=0;
}

void GSM_Response_Display()
{
        buffer_pointer = 0;
        int lcd_pointer = 0;
```

```
        while(1)
        {
                if(buff[buffer_pointer]== '\r' || buff[buffer_pointer]== '\n')  /*search for \r\n in
string */
                {
                        buffer_pointer++;
                }
                else
                break;
        }



        lcdcmd(0xc0);
        while(buff[buffer_pointer]!='\r')
          /* display response till "\r" */
        {
                LCD_Char(buff[buffer_pointer]);
                buffer_pointer++;
                lcd_pointer++;
                if(lcd_pointer==15)
/* check for end of LCD line */
                lcdcmd(0x80);
        }
        buffer_pointer=0;
        memset(buff,0,strlen(buff));  /* emptying memory block */
}
```

GSM command manual - https://www.espruino.com/datasheets/SIM900_AT.pdf

GSM datasheet - https://researchdesignlab.com/projects/GPRSGSM%20SIM900A%20MODEM.pdf

ATMEGA 32 Full Data sheet - https://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf

ATMEGA 32 Summery Data sheet - https://ww1.microchip.com/downloads/en/DeviceDoc/2503S.pdf

4.Name of Student: Sanchitha S.P.Y.P. - 204186H

Load Cell

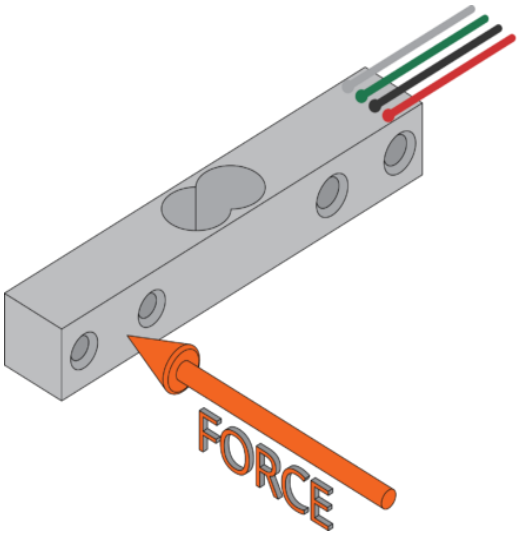
First, I choose the Strain Gauge load cells as the most suitable load cell type for this project. The reason is these type load cells are high accuracy, Cost-effective and very sturdy. As the load cell amplifier, I choose HX711 amplifier.

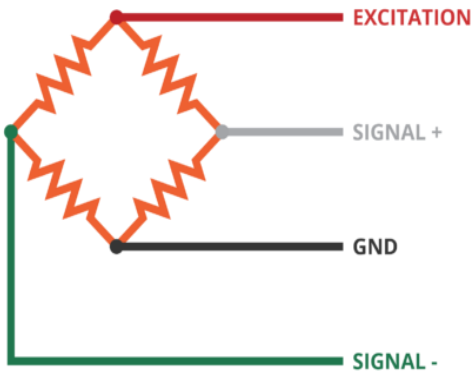
Strain Gauge Load Cell

Load cell is a device which use to measure weight. Main part of the load cell is Strain Gauge.

Strain Gauge is a device which used to measure strain of an object. The resistance of a strain gauge changes when force is applied and this change will give a different electrical output.

So in the load cell, there are 4 strain gauges connected like a Wheatstone Bridge. When the bridge is in balanced position there is no voltage difference, but when bridge is not balanced there is voltage difference. By using this method, we can find the weight of an object.





Specifications -:

Housing Material	-	Aluminum Alloy
Load Cell Type Strain Gauge Capacity	-	20g
Cable - no. of leads	-	4
Excitation Voltage	-	5 VDC
Operating Temperature Range	-	-10 to ~+40°C
Safe Overload	-	120% Capacity
Ultimate Overload	-	150% Capacity

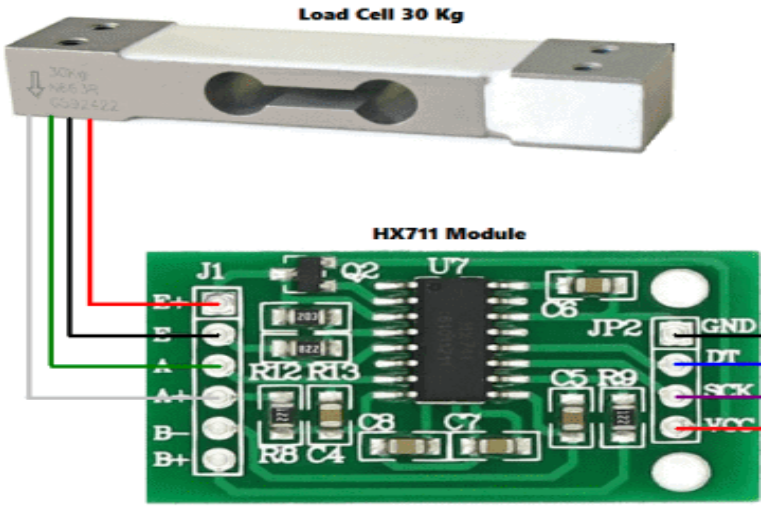
HX-711 load cell amplifier

HX711 is a precision 24-bit analog- to-digital converter (ADC) designed for weigh scales and industrial control applications to interface directly with a bridge sensor

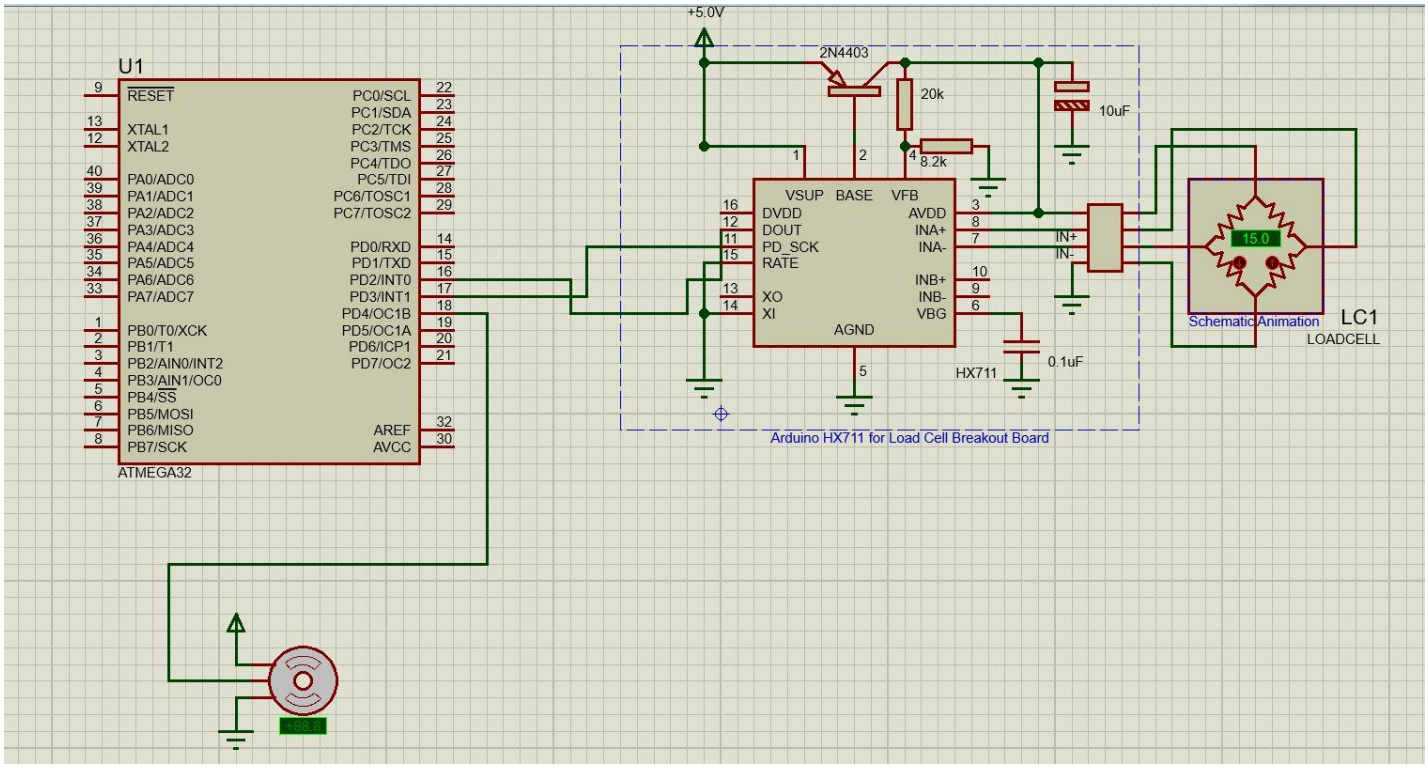
Specifications -:

Operating voltage (v)	-	2.7V to 10VDC
Operating Current	-	<10 mA

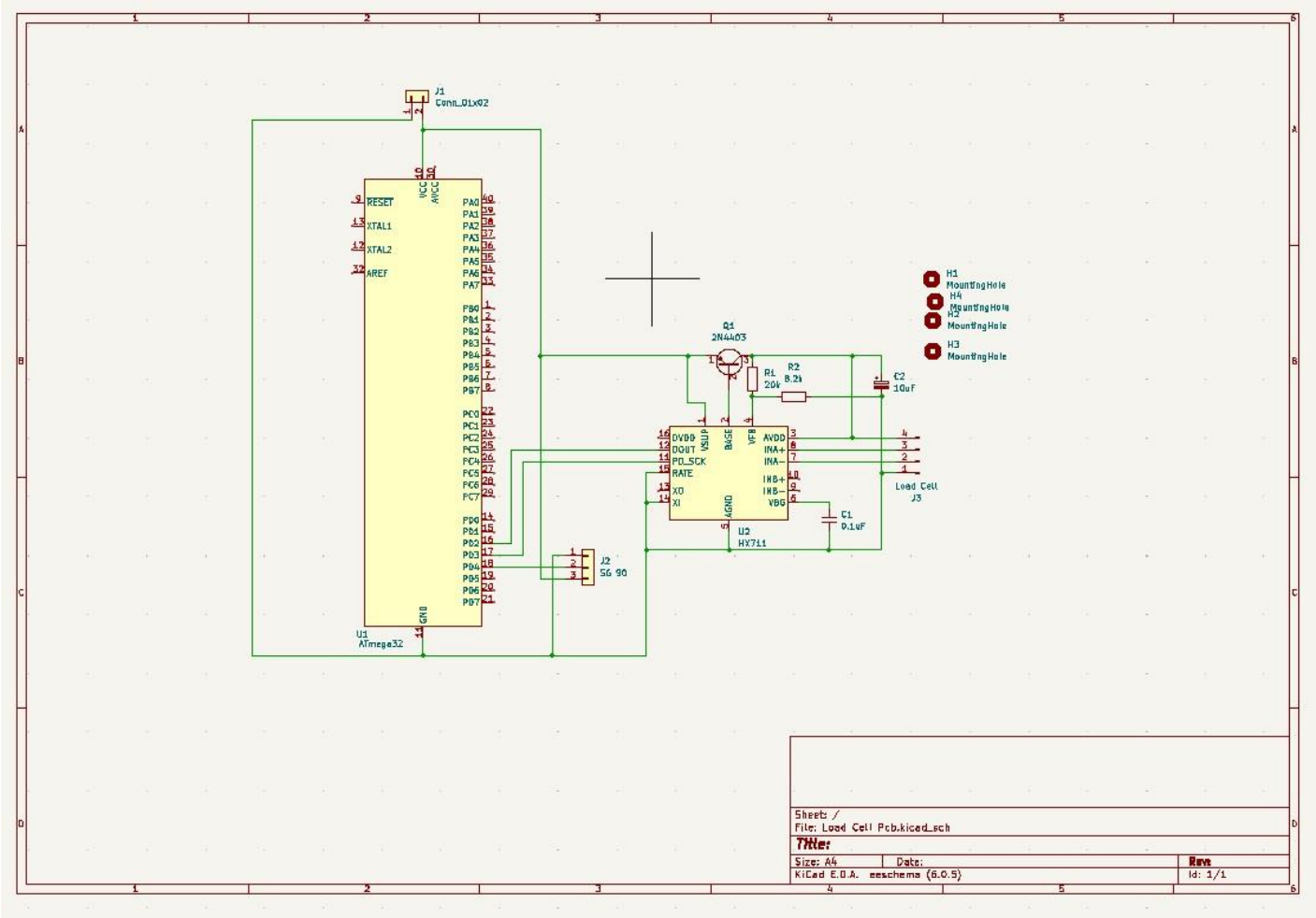
Da



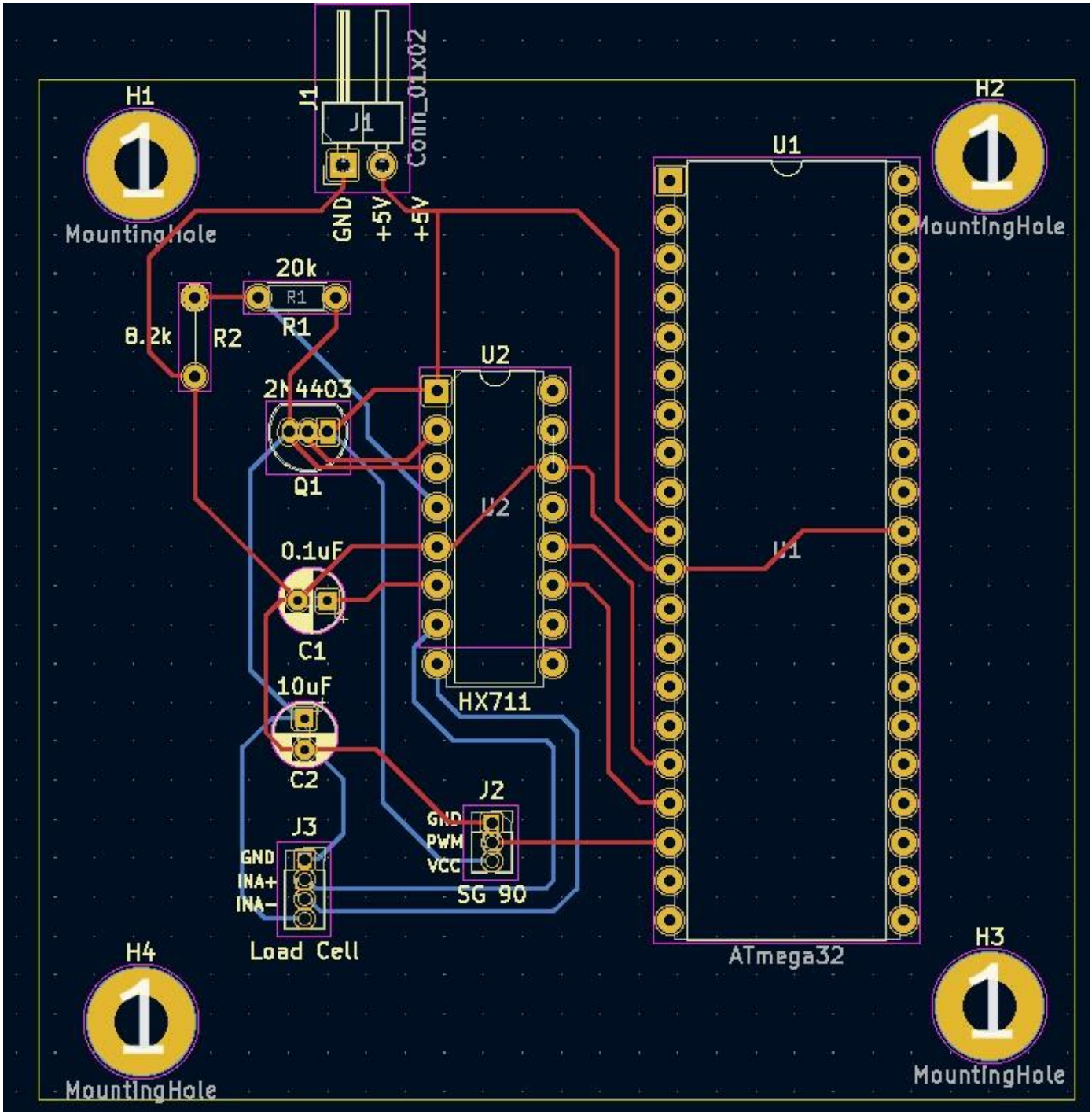
Circuit Diagram



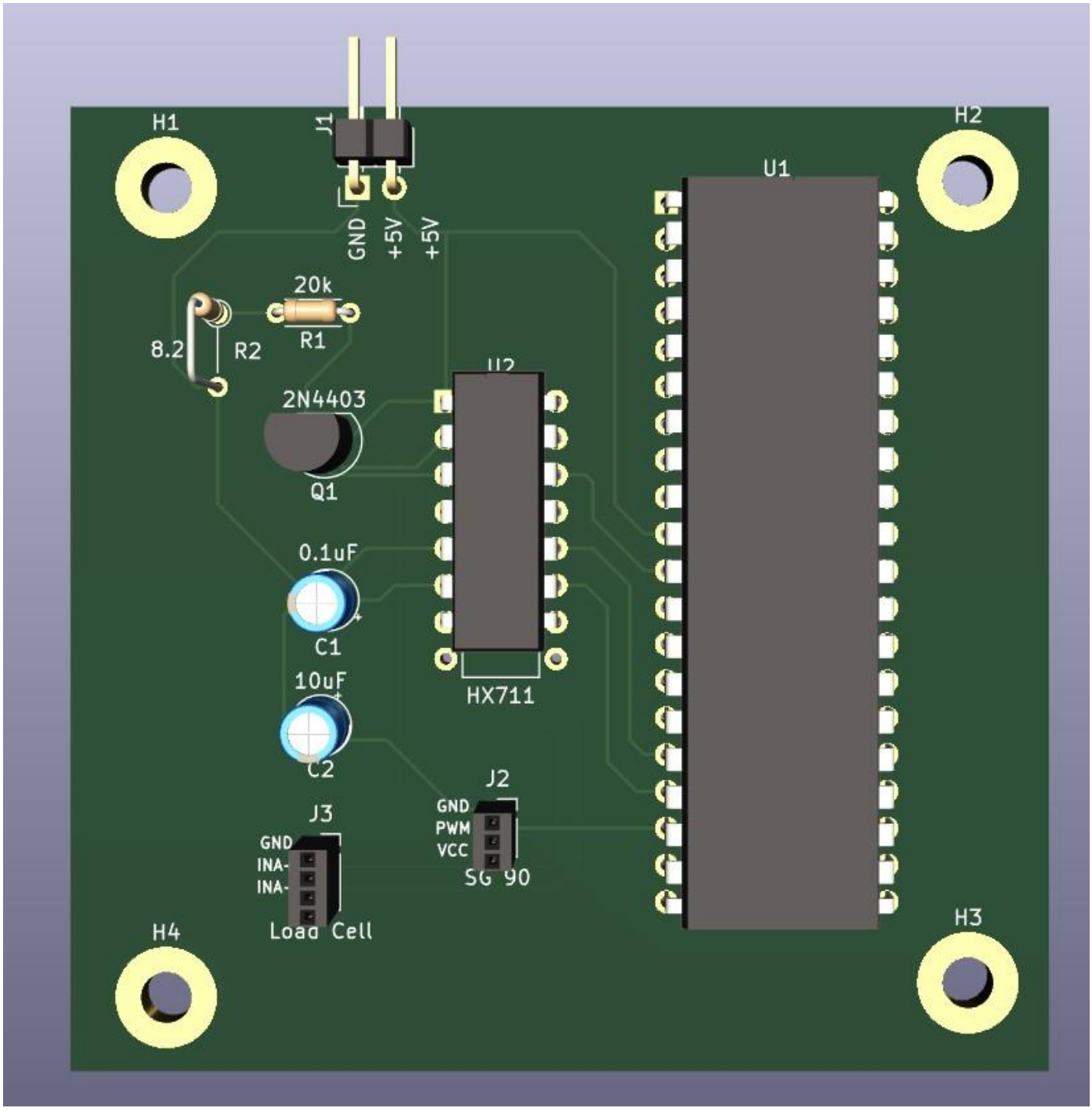
Schematic Diagram



PCB Design


H1
MountingHole
J1
Conn_01x02
GND
+5V
+5V
H2
MountingHole
U1
20k
R1
8.2k
R2
U2
2N4403
Q1
U2
0.1uF
C1
U1
10uF
C2
HX711
J2
GND
PWM
VCC
SG 90
J3
GND
INA+
INA-
Load Cell
H4
MountingHole
ATmega32
H3
MountingHole

3d view


H1
J1
H2
U1
GND
+5V
+5V
20k
8.2
R2
R1
2N4403
Q1
0.1uF
C1
10uF
C2
HX711
J2
GND
PWM
VCC
SG 90
J3
GND
INA-
INA-
Load Cell
H4
H3

Code

```
#define F_CPU 8000000
#include <avr/io.h>
#include <util/delay.h>

uint8_t hx711H=0; //Load Scale High Bits
uint16_t hx711L=0;//Load Scale Low Bits
float loadCellRead();
#define Load_data 2
#define Load_clk 3


float loadCellRead();
float hx=0;
void servoangle(uint8_t pin);

int main(void)
{
	DDRD|=(1<<Load_clk); //Load cell clock pin
	PORTD&=~(1<<Load_clk);//Clock pin low
	DDRD|=(1<<4);

	while (1)
	{
		hx=loadCellRead();
		if (hx>15)
		{servoangle(0);
		}
		else
		{servoangle(1);
		}
		_delay_ms(100);
	}
}


float loadCellRead(){
	hx711H=0;hx711L=0;  //clear variables
	for(uint8_t i=0;i<8;i++){  // Load cell data high 8 bits
		PORTD|=(1<<Load_clk); //Clock pin high
		_delay_us(10);
		if ((PIND&(1<<Load_data))>>Load_data)  //read data pin
		{hx711H|=(1<<(7-i));//set hx 711 varible
		}
		else
		{hx711H&=~(1<<(7-i));
		}
		PORTD&=~(1<<Load_clk); //Clock pin low
		_delay_us(5);
	}


	for(uint8_t i=0;i<16;i++){ // Load cell data low 16 bits
		PORTD|=(1<<Load_clk); //Clock pin high
		_delay_us(10);
		if ((PIND&(1<<Load_data))>>Load_data) //read data pin
		{hx711L|=(1<<(15-i));
```

```
                }
                else
                {hx711L&=~(1<<(15-i));
                }
                PORTD&=~(1<<Load_clk); //Clock pin low
                _delay_us(5);
        }

        hx711L=hx711L>>1; //shift bits

        if (hx711H&1)  //bit setup
        {hx711L|=(1<<15);
        }
        else
        {hx711L&=~(1<<15);
        }
        hx711H=hx711H>>1;

        return (hx711H*(65536/18029.6))+hx711L/18029.6; //load cell calibration
}

void servoangle(uint8_t pin){

        if (pin)
        {
                for(uint8_t j=0;j<100;j++){
                        PORTD|=(1<<4);
                        for(uint8_t i=0;i<10;i++){
                                _delay_us(100);
                        }
                        PORTD&=~(1<<4);

                        uint8_t ser=200-10;

                        for(uint8_t i=0;i<ser;i++){
                                _delay_us(100);
                        }
                }
        }
        else
        {for(uint8_t j=0;j<100;j++){

                PORTD|=(1<<4);
                for(uint8_t i=0;i<15;i++){
                        _delay_us(100);
                }
                PORTD&=~(1<<4);

                uint8_t ser=200-15;

                for(uint8_t i=0;i<15;i++){
                        _delay_us(100);
                }
        }
}



}
```

5.Name of Student: Kavinda K.A.R. – 204098N

ULTRASONIC SENSOR

Ultrasonic Sensor (HC-SR04)

The ultrasonic sensor works on the principle of SONAR and RADAR system which is used to determine the distance to an object.

An ultrasonic sensor generates the high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo. By using that we can measure the distance to the object

Distance measuring

Distance = speed * time

The speed of sound waves is 340 m/s. (34000 cm/s)

So,

Total Distance = $(Tx34000 \times 10^{-6})/2$

= (17/1000)

= T/58

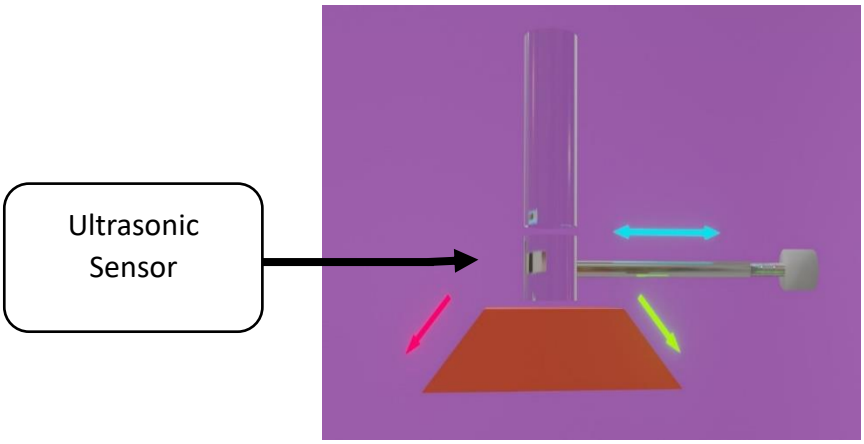
HC-SR04 ultrasonic sensor has 4 pins which are VCC, Trigger, Echo, GND

VCC	- Connects to the VCC supply
Trigger	-Creates 8 pulses of 10µsec or more
Echo	-Starts a single clock when trigger activated
GND	-Connects to the ground

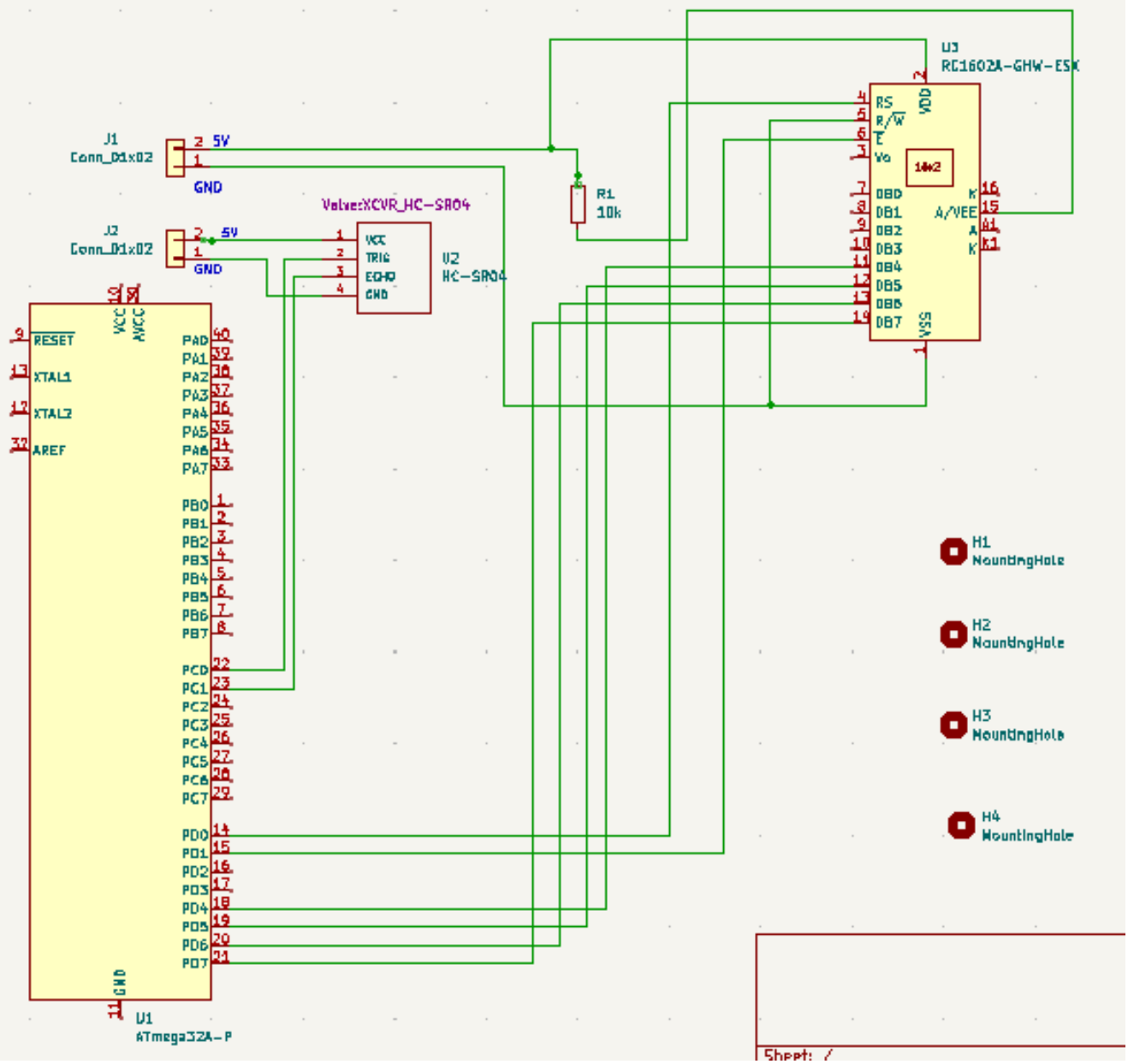
Specifications -:

Model	: HC-SR04
Power	: 5V
Working Current	: 15mA
Ranging distance	: 0.2- 400 cm
Measuring Angle	: 30 ^{o}
Trigger input pulse width	: 10 µS
Weight	: 10g
Modulation frequency	:40Hz

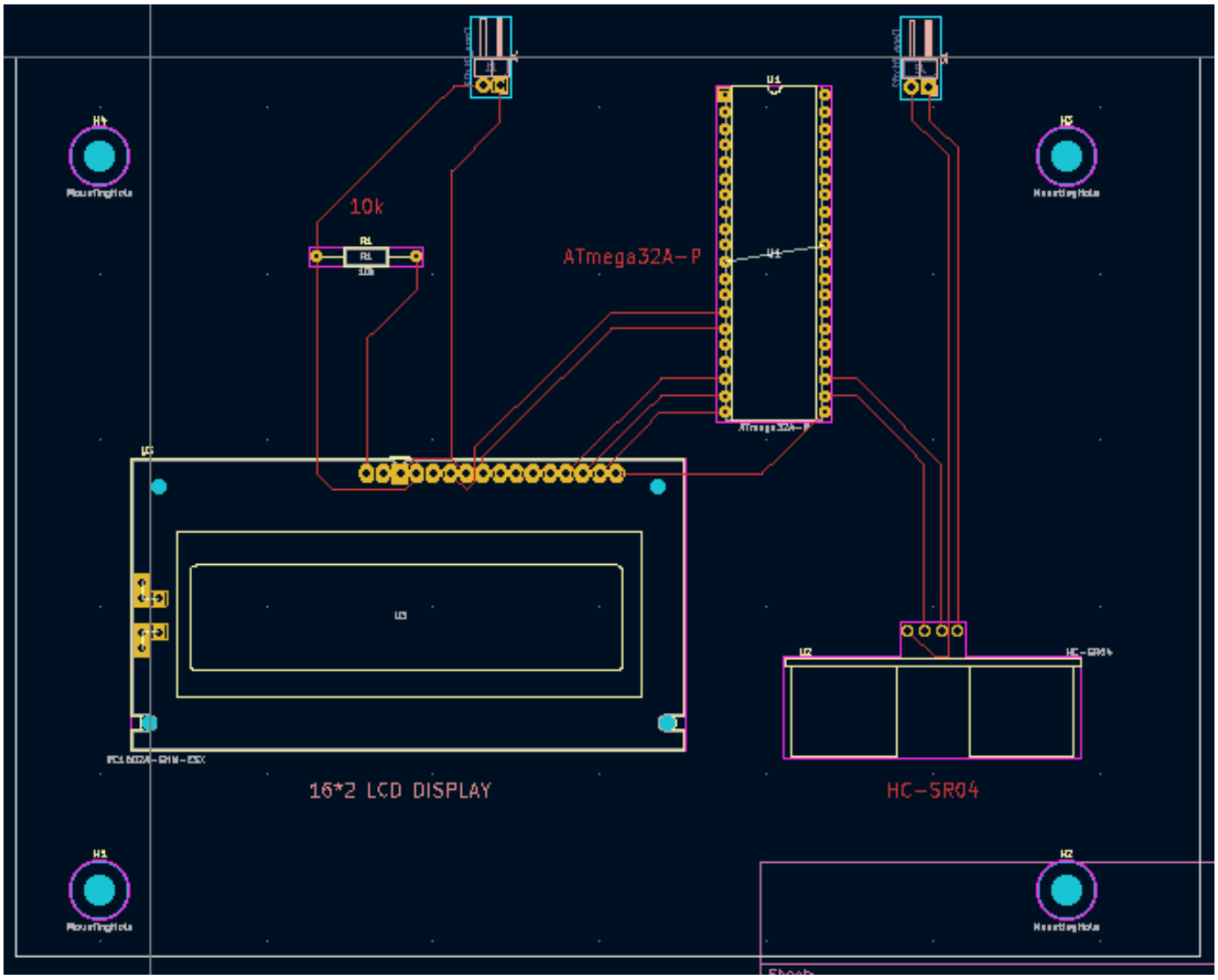
physically location



Schematic diagram of ultrasonic sensor



PCB design



Code for ultrasonic sensor

```
#define F_CPU 16000000UL
#include <avr/io.h>            // Include AVR std. library file
#include <util/delay.h>
#include <style>            //convert integer to character array

#define lcd_port PORTD          // connected lcd on to the port D
#define lcd_data_dir  DDRD      // difinding the direction of the pins- input/out
#define rs PD0
#define en PD1

#define US_PORT PORTC           // connected ultrasonic to port C
#defineUS_PIN          PINC           // initialization of pin register
#define US_DDR       DDRC           // data direction register to set the data direction flow

#define US_TRIG_POS         PC0
#define US_ECHO_POS       PC1


#define US_ERROR              -1     //difinding variable to  working or not
#defineUS_NO_OBSTACLE-2

int distance, previous_distance;
```

```
void HCSR04Init();
void HCSR04Trigger();

void lcd_command( unsigned char );

void HCSR04Init()
{


        US_DDR|=(1<<US_TRIG_POS);
}

void HCSR04Trigger()
{

        US_PORT|=(1<<US_TRIG_POS);

        _delay_us(15);

        US_PORT&=~(1<<US_TRIG_POS);
}

uint16_t GetPulseWidth()
{


        uint32_t i,result;


        for(i=0;i<600000;i++)
```

```
{
        if(!(US_PIN & (1<<US_ECHO_POS)))
        continue;
        else
        break;
}

if(i==600000)
return US_ERROR;

TCCR1A=0X00;
TCCR1B=(1<<CS11);
TCNT1=0x00;


for(i=0;i<600000;i++)
{
        if(US_PIN & (1<<US_ECHO_POS))
        {
                if(TCNT1 > 60000) break; else continue;
        }
        else
        break;
}

if(i==600000)
return US_NO_OBSTACLE;
```

```
        result=TCNT1;



        TCCR1B=0x00;

        if(result > 60000)
        return US_NO_OBSTACLE;
        else
        return (result>>1);
}



void initialize (void)
{
        lcd_data_dir = 0xFF;     //lcd pins that connected to micro-controller as out puts
        _delay_ms(15);
        lcd_command(0x02);
        lcd_command(0x28);
        lcd_command(0x0c);
        lcd_command(0x06);
        lcd_command(0x01);
        _delay_ms(2);
}

void lcd_command( unsigned char cmnd )
{

        lcd_port = (lcd_port & 0x0F) | (cmnd & 0xF0);
        lcd_port &= ~ (1<<rs);
```

```
	lcd_port |= (1<<en);
	_delay_us(1);
	lcd_port &= ~ (1<<en);


	_delay_us(200);


	lcd_port = (lcd_port & 0x0F) | (cmnd << 4);
	lcd_port |= (1<<en);
	_delay_us(1);
	lcd_port &= ~ (1<<en);
	_delay_ms(2);
}

void lcd_clear()
{
	lcd_command (0x01);
	_delay_ms(2);
	lcd_command (0x80);
}


void lcd_print (char *str)
{

	int i;
	for(i=0; str[i]!=0; i++)
	{
```

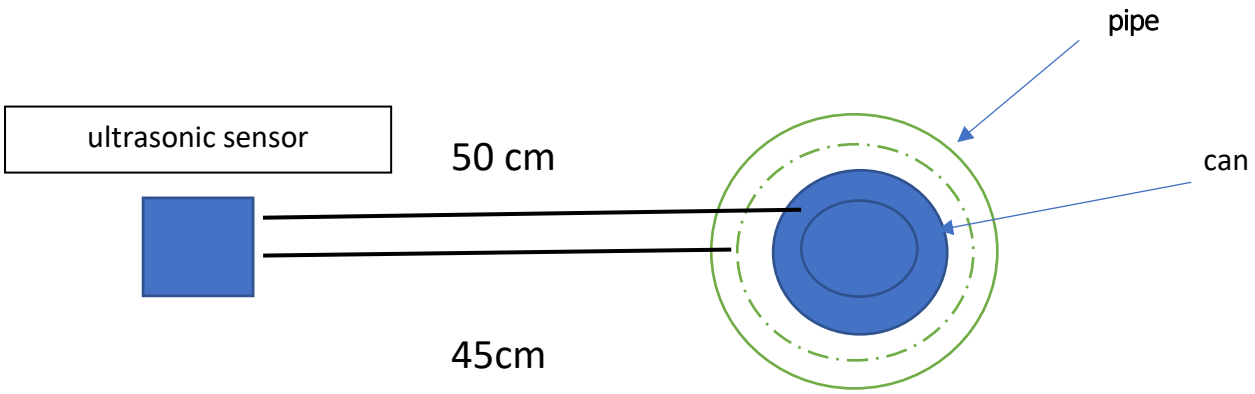
```
            lcd_port = (lcd_port & 0x0F) | (str[i] & 0xF0);
            lcd_port |= (1<<rs);
            lcd_port|= (1<<en);
            _delay_us(1);
            lcd_port &= ~ (1<<en);
            _delay_us(200);
            lcd_port = (lcd_port & 0x0F) | (str[i] << 4);
            lcd_port |= (1<<en);
            _delay_us(1);
            lcd_port &= ~ (1<<en);
            _delay_ms(2);
        }
}



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

int main()
{
        initialize();

        char numberString[4];

        while(1) {
```

```
uint16_t r;

_delay_ms(100);



HCSR04Init();  // set io port direction of sensor



lcd_print("hello customer !");
_delay_ms(10);
lcd_setCursor(1, 2);
lcd_print("enter your can");
_delay_ms(1000);
lcd_clear();
while(1)
{

        HCSR04Trigger();     // calling the ultrasonic sound wave generator
function

        r=GetPulseWidth();      // getting the duration of the ultrasound took
echo back

    // handle errors
        if(r==US_ERROR)
        {
            lcd_setCursor(1, 1);
            lcd_print("Error!");
```

```
}
else
{

        distance=(r*0.034/2.0);


        if (distance != previous_distance)
        {
                lcd_clear();
        }

        lcd_setCursor(1, 1);


        if((distance>45)&&(distance<50))
        {
                lcd_setCursor(1, 1);
                lcd_print("can  detected");

        }
        else
        {
                lcd_setCursor(3, 1);
                lcd_print("can not detected");
        }


        itoa(distance, numberString, 10);   // converting
        lcd_setCursor(8, 2);
```

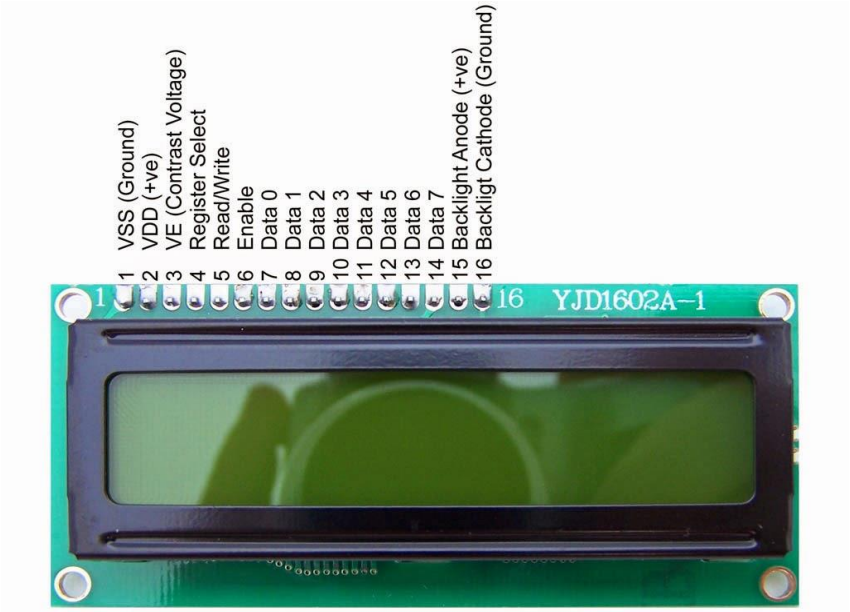
```
                    lcd_print(numberString);
                    lcd_setCursor(10, 2);
                    lcd_print(" cm");




                    previous_distance = distance;
                    _delay_ms(30);
                }
            }

        }
}
```

Working process



- Using an ultrasonic sensor, we measure the distance between the can from the sensor. If the distance is between 50cm and 45cm we can determine that customer entered a valid can. But to ensure that it is a can, we also must consider weight sensor output and proximity sensor output.

Ultrasonic Sensor Data Sheet -: https://datasheetspdf.com/pdf/1380136/ETC/HC-SR04/1



Lcd 16*2 display

This display includes 2 lines, and each line can display 16 characters. Each character displays in 5*7-pixel matrix. Most of the 16x2 LCDs use a Hitachi HD44780 or a compatible controller.

Using this display, we can get many benefits.it is inexpensive, simply programmable and there are no limitations for displaying custom characters. Hence, we use lcd 16*2 display for our hardware project.

Features

Operating voltage : 4.7V-5.3V

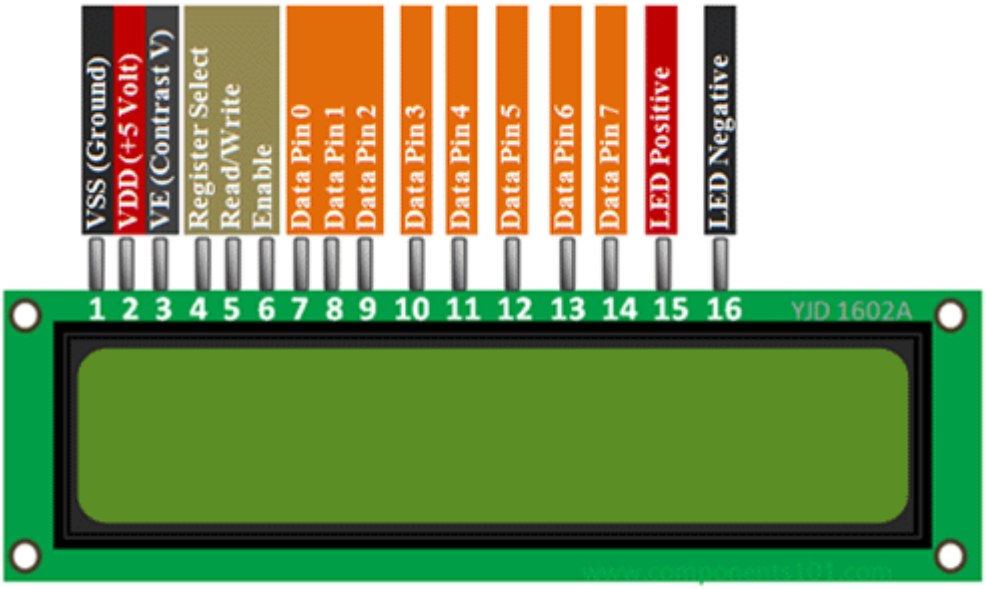
The utilization of current : 1mA (with no backlight)

Working modes : 4bit mode & 8bit mode

Display characters : 32 characters

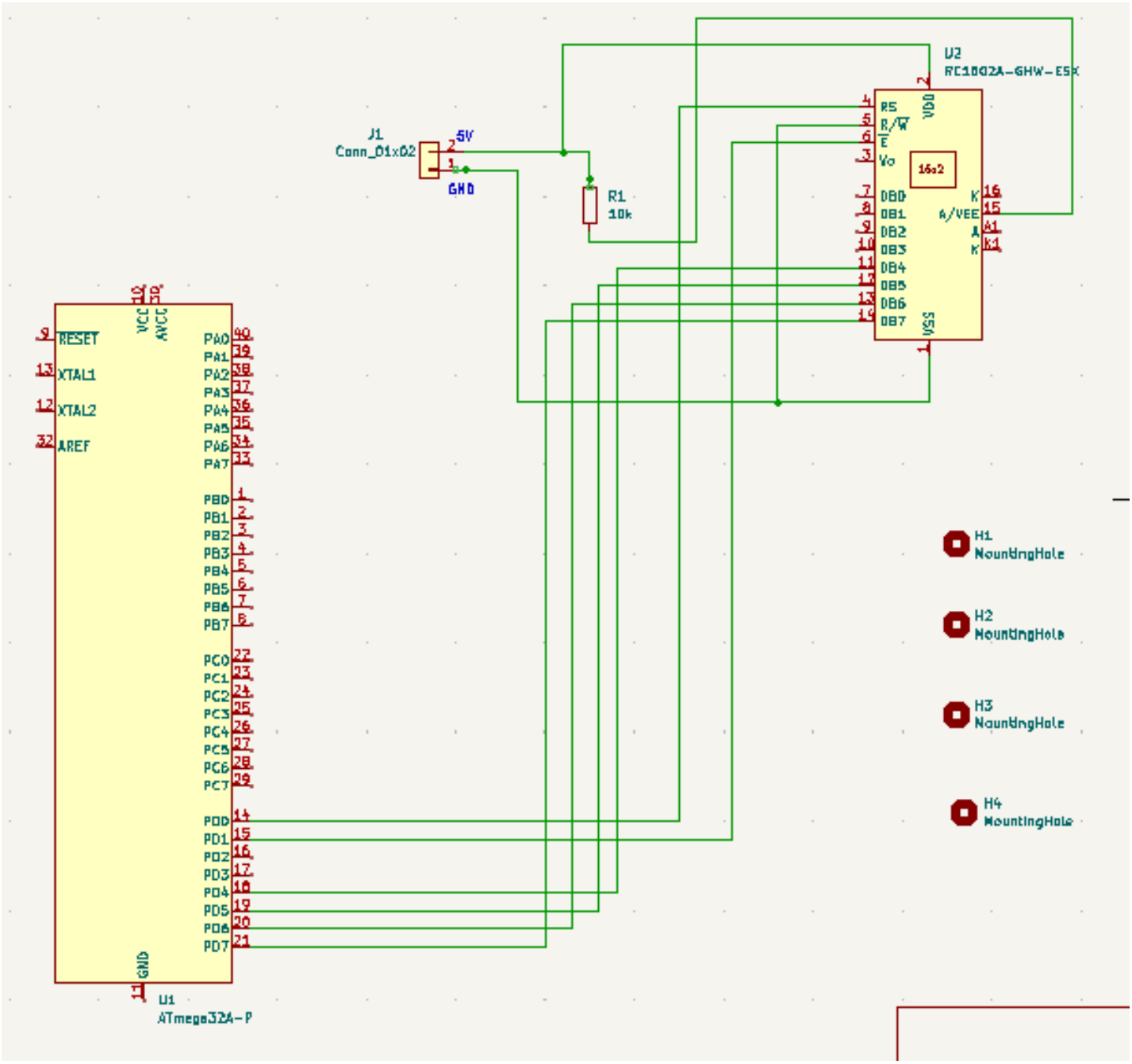
There are green & blue blacklight displays in the market

Pin configuration of 16*2 display

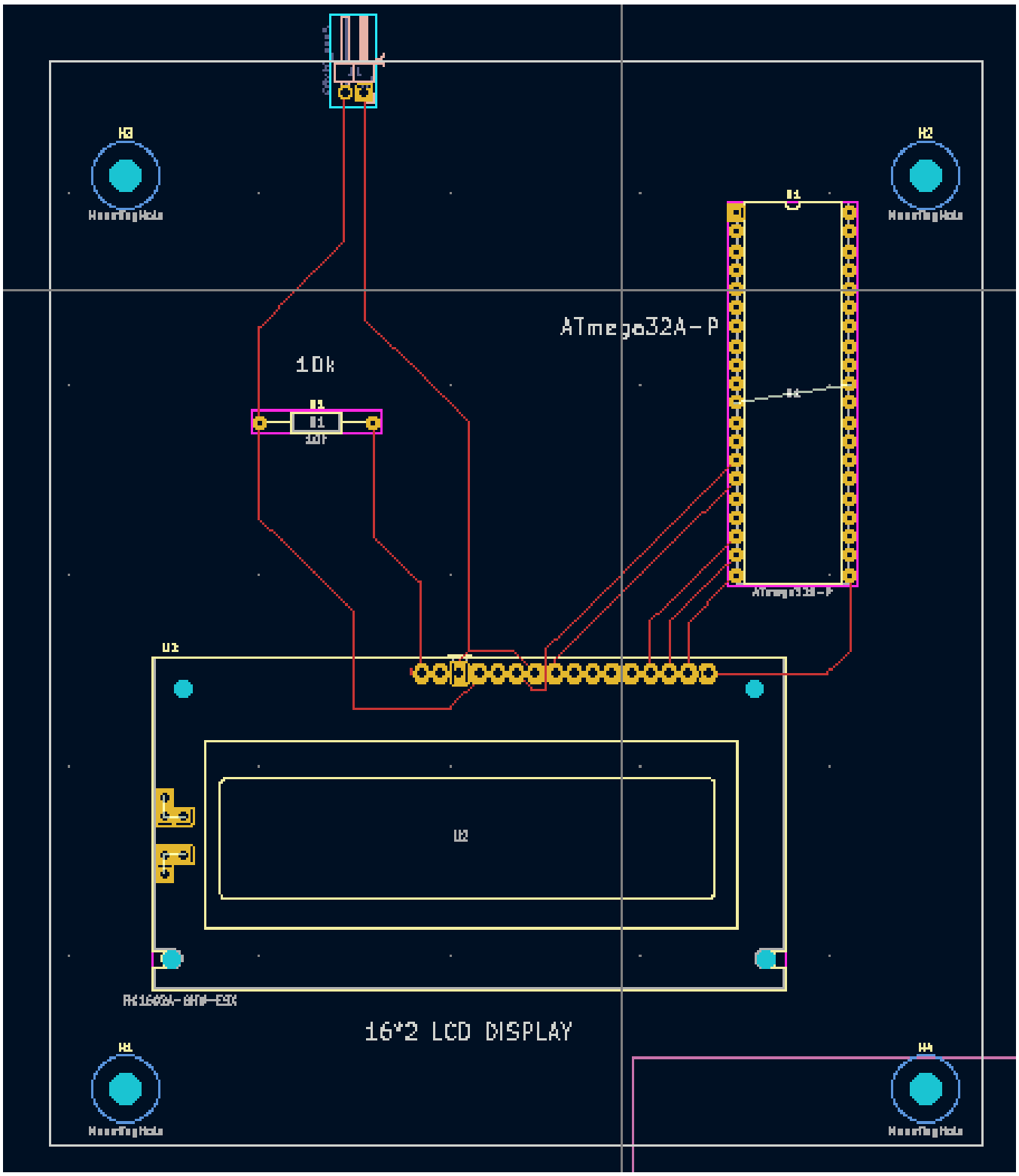


physically location

Schematic diagram of lcd display



PCB design



Code for lcd display

```
#define F_CPU 16000000UL
#include <avr/io.h>        // Include AVR std. library file
#include <util/delay.h>

#define lcd_port PORTD    // connected lcd on to the port D
#define lcd_data_dir  DDRD   // difinding the direction of the pins- input/out
#define rs PD0
#define en PD1

void lcd_command( unsigned char );

void initialize (void)
{
        lcd_data_dir = 0xFF;
        _delay_ms(15);
        lcd_command(0x02); // 4-bit initialization of lcd
        lcd_command(0x28);   // 2 line,5*7 matrix in 4-bit mode
        lcd_command(0x0c);  //display on cursor off
        lcd_command(0x06);  //increment cursor
        lcd_command(0x01);  // clear display screen
        _delay_ms(2);
}

void lcd_command( unsigned char cmnd )
{
        // send higher 4 bit of data
        lcd_port = (lcd_port & 0x0F) | (cmnd & 0xF0);
        lcd_port &= ~ (1<<rs);
        lcd_port |= (1<<en);
```

```
        _delay_us(1);
        lcd_port &= ~ (1<<en);


        _delay_us(200);

        // lower 4 bit of data
        lcd_port = (lcd_port & 0x0F) | (cmnd << 4);
        lcd_port |= (1<<en);
        _delay_us(1);
        lcd_port &= ~ (1<<en);
        _delay_ms(2);
}

void lcd_clear()
{
        lcd_command (0x01);
        _delay_ms(2);
        lcd_command (0x80); // cursor at home
}



void lcd_print (char *str)
{
        int i;
        for(i=0; str[i]!=0; i++)
        {

                lcd_port = (lcd_port & 0x0F) | (str[i] & 0xF0);
                lcd_port |= (1<<rs);
```

```
                lcd_port|= (1<<en);
                _delay_us(1);
                lcd_port &= ~ (1<<en);
                _delay_us(200);
                lcd_port = (lcd_port & 0x0F) | (str[i] << 4);
                lcd_port |= (1<<en);
                _delay_us(1);
                lcd_port &= ~ (1<<en);
                _delay_ms(2);
        }
}



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

int main()
{
        initialize();

        lcd_setCursor(2, 1);
        lcd_print("hello customer");
        lcd_setCursor(2,2 );
        lcd_print("Enter your can");

}
```

Working process

Using lcd display we are giving necessary outputs to the customer according to the measurements of the sensors.

Lcd display data sheet - https://www.vishay.com/docs/37299/37299.pdf