PROJECT REPORT

On

POTHOLE FILLER

Submitted in partial fulfillment towards the degree of

BACHELOR OF TECHNOLOGY

in

MECHANICAL ENGINEERING

Prepared by

NIRAV D. GAGLANI

MH-27 (I.D. No. 18MHUOS096)

DEPARTMENT OF MECHANICAL ENGINEERING

FACULTY OF TECHNOLOGY

DHARAMSINH DESAI UNIVERSITY, NADIAD

APRIL 2022

BONAFIDE CERTIFICATE

Certified that the project report titled **"POTHOLE FILLER"** is the bonafide work of **NIRAV DIPAKBHAI GAGLANI (Roll No. MH-27, ID No. 18MHUOS096, Bachelor of Technology in Mechanical Engineering, Semester VIII, 2018-22)** who made the project under our supervision.

(Prof (Dr). G. D. Bassan)
HEAD
Mechanical Engineering Department
Faculty of Technology
Dharamsinh Desai University
College Road
Nadiad – 387 001
Gujarat

(Prof. Mahavirsinh M. Chavda)
SUPERVISOR
Assistant Professor
Mechanical Engineering Department
Faculty of Technology
Dharamsinh Desai University
College Road
Nadiad – 387 001, Gujarat

ABSTRACT

Potholes are big concern especially on highways, when vehicles are moving around 100 kmph and suddenly potholes appears. Undeniably potholes create huge discomfort but sometimes it also damages suspensions and if pothole is deeper enough, it may also damage bumpers and lower most parts of car or any other vehicle. In the most unfortunate situation, it may also result in an accident. This project is made with two purpose, first to solve this problem of massive number of unrepaired potholes on the road. The reason for so many unrepaired potholes can be the shortage of people against the massive number of potholes. And second more important motive behind this project is, workers fill potholes in summers, under Sun around 40 to 45 degree centigrade temperature along with the extremely hot Bitumen. With this project, workers will not have to work with hot bitumen nor under the hot weather. Filling potholes will just be a matter of pressing few buttons with this project.

The project's functionality is achieved by programming Arduino MEGA to read the depth measurement data from Ultrasonic Sound Sensor (HCSR04) and giving appropriate commands to five Nema17 Stepper Motors.

ACKNOWLEDGEMENT

The final outcome of this project is the result of lots of efforts, guidance and support from faculties and seniors, encouragement and co-operation from my parents. I feel so privilege to have this all along with the making of this project.

I am highly indebted to **Prof. Mahavirsinh M. Chavda** for providing me an opportunity to do this project. I am thankful for his constant guidance and support. He has been always accessible for any doubt and question, whether be it college hours or his personal time.

I would like to thank honorable **HOD Prof (Dr). G.D. Bassan Sir** for firstly allowing 8th semester students to do project instead of internship and secondly for providing all the available resource of the university.

Special thanks to Prof. P.A. Shah for giving me wonderful idea related to the structure of the project and for providing references of seniors who has been of great help, Mr. Nirav Chhatrola & Mr. Aakash Parmar.

I would like to express my special gratitude and thanks to workshop faculties for giving me their time to help me in welding and carpentry.

Yours Sincerely,

Nirav Gaglani

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

This project aims at solving the existing problem of massive number of unrepaired potholes on roads and to overcome the difficulty of filling it.

1.1 Overview:

The concept of this project is very simple. I have used an ultrasonic sound sensor, which is used to measure linear distance. Now, there is a frame of size 1 meter x 1 meter, which provides a sensing range of about 90 cm x 82 cm. There is a motor drive which provides 2D movement that moves this sensor all over the area of the frame. Now, this sensor measures the depth at almost every point within the frame. After measuring the depth, mapping of pothole is done and the volume of pothole is estimated. All of this is achieved by programming the Arduino MEGA.

Now there is a tank, which stores the filling material, and the amount of filling material to be deposited can be controlled with the stepper motor attached to it.

Once the volume is estimated, a center point of the pothole is estimated where the material can be dispatched. Arduino mega then sends signal to stepper motor attached at storage tank and desired amount of material is dispatched.

Figure 1: Project Picture

Chapter 2

Structure Overview

2.1 STRUCTURAL COMPONENTS:

1. Wooden Base
2. Aluminium Section
3. Acrylic Sheet
4. GT2 Timing Belt
5. GT2 Timing Pulley
6. Plastic Roller (with bearing)
7. Fasteners
8. Wooden Storage Tank
9. MS Frame

2.1.1 WOODEN BASE:

Base is the essential part of the machine. It is the lowest most component on which the whole structure is to be constructed. I have used 2 wooden base of length 1040 mm, width 90 mm and thickness 17 mm. I have also used 4 wooden blocks to elevate the project. The height of this wooden blocks is 40mm.

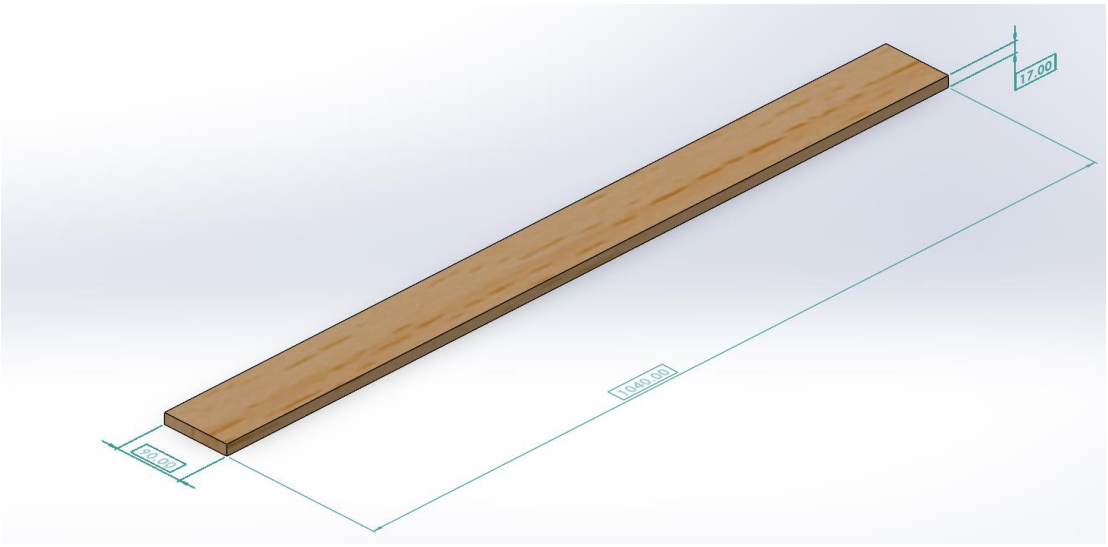

Figure: 2.1.1: Wooden Base

2.1.2 ALUMINIUM SECTION:

The Astro Light Duty Industrial Aluminium Alloy 4040 L European Standard Anodized Profile is made of High Grade tempered Aluminium Alloy 6063 – TS. The dimensions are as per European Standard and the structural cross section thickness is approximately 1.3 mm. The profile can be used for light duty applications like frames. It is not recommended to use this profile as a structural member.

The weight per meter of this section is approx. 0.9kg per meter as compared to the 1.86 kg per meter for the Astro Extra Heavy duty profile. The thin wall section of the 4040 L section makes this suitable for extremely light weight applications.

The profile is anodised in natural aluminium colour to provide a hard layer, to prevent corrosion and wear. The normally available profiles in the market are not anodised and are prone to scratches much easier.

Figure 2.1.2 (a): Al Section Image

Table 2.1.2: Aluminium Section Description

Profile Size	4040 L
Type	European Standard Light Duty Profile
Material	High Strength Tempered Aluminium Alloy 6063 – T5
Diagonal Section Thickness	1.3 mm
Weight	0.9 kg/meter
Dimensions	OD 40 mm x 40 mm
Length	1 meter

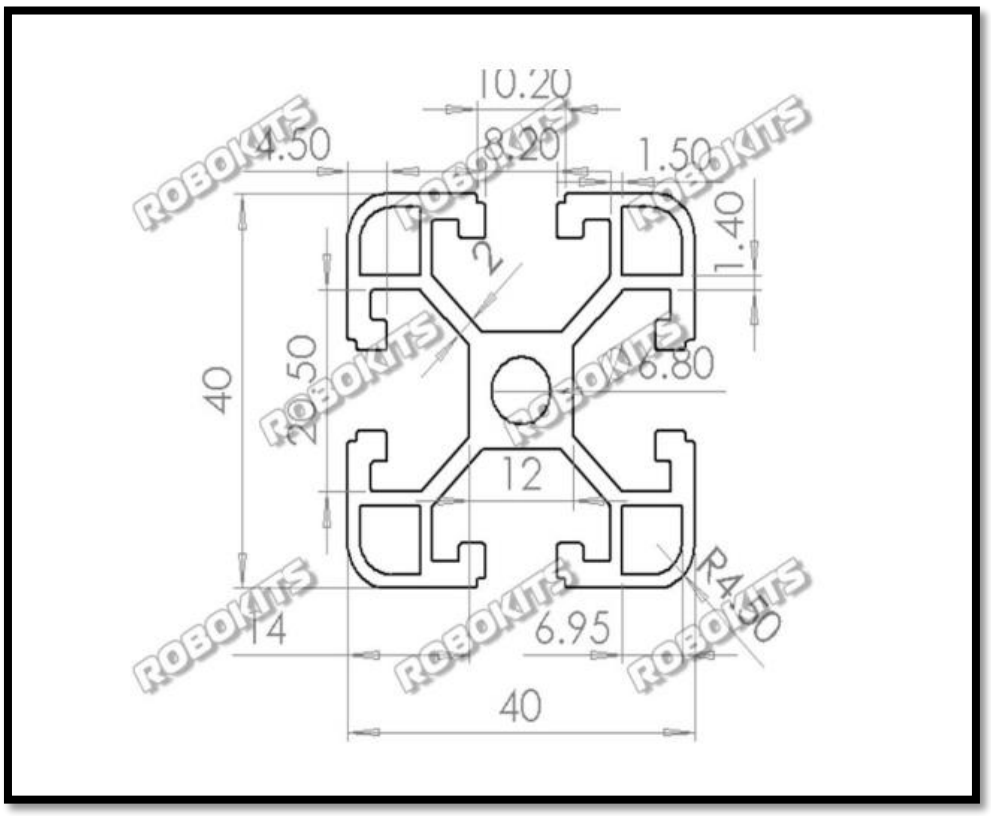

Figure 2.1.2 (b): Al Cross Section

2.1.3 ACRYLIC SHEET:

Acrylic is stronger and lighter than glass, excellent resistance to long-term exposure of outdoor elements. Multipurpose-acrylic sheet uses are glass window, outdoor signs, lighting applications and bending (or manipulation).

Ideal for Arts & Crafts, Educational Model Making, Construction, Window Glass, Home Décor, Designing Products, Tokens, DIY. Acrylic sheets comes with Protection film.

I have used total 3 cut outs of acrylic sheets as shown under. 2 cut outs for Y axis and 1 cut out for X axis.

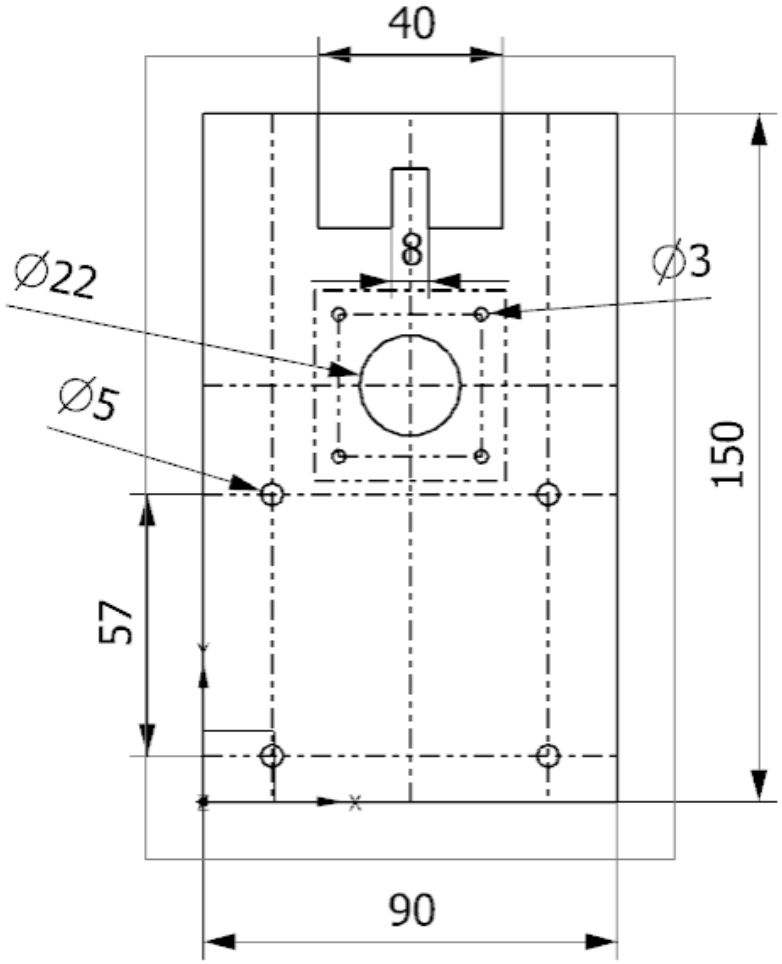

Figure 2.1.3 (a): Cut out for Y axis

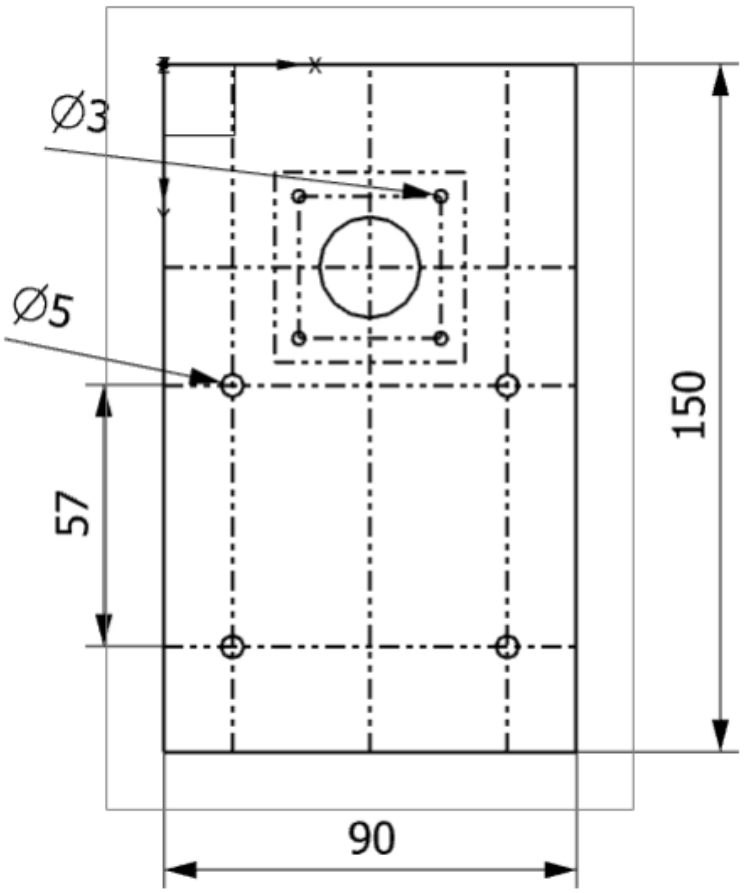

Figure 2.1.3 (b): Cut out for X Axis

Table 2.1.3: Description for Cutouts

Material	Acrylic
No. of cut outs for X-axis	1
No. of cut outs for Y-axis	2

2.1.4 GT-2 TIMING BELT:

GT-2 Timing Belt is also called as synchronous belt and is popularly known for its non-slipping mechanical drive. It is composed of flexible belt which contains row of teeth embedded on the inner surface of the belt. Timing pulley and belt works when toothed parts become compatible with each other.

GT-2 Timing Belt is used in RepRap, 3D Printer, CNC, Robotics and Automation.

I have used total of 3m of GT-2 Timing Belt in this Project.

Figure 2.1.4: GT-2 Timing Belt

Table 2.1.4: Belt Description

Part/Model	6GT2
Colour	Black
Pitch	2 mm
Type	Open Ended
Belt Width	6 mm
Material	Neoprene Rubber with Fibre Glass Reinforcements

2.1.5 TIMING PULLEY:

GT-2 Timing Pulleys are used in a variety of ways to lift loads, apply forces and to transmit power. Pulleys are also assembled as part of belt and chain drives in order to transmit power from one rotating shaft to another. GT-2 Timing Pulleys are used in applications like RepRap, 3D Printer, CNC, Robotics and Automations, etc.

Figure 2.1.5: GT-2 Timing Pulley

Table 2.1.5: Pulley Description

Part/Model	20GT2
Shaft Diameter	5 mm
Pitch	2 mm
Tooth	20
Width	6 mm
Material	Aluminium
No. of pulley used	3

2.1.6 FASTENERS:

A fastener is a hardware device that mechanically joins or affixes two or more objects together. In general, fasteners are used to create non-permanent joints. That is, joints that can be removed or dismantled without damaging the joining components. Welding is an example of creating permanent joints. Steel fasteners are usually made of stainless steel, carbon steel, or alloy steel.

Figure 2.1.6: Fasteners

2.1.7 PLASTIC ROLLER (EITH BEARING):

Generally plastic rollers are used with the slotted Al section in which the tapper part of the roller rests on the slot and freely move on that with the help of the bearing at the center of the wheel. Generally the plastic part of the roller is made by 3D printing technology.

Figure 2.1.7: Plastic Roller with bearing

Table 2.1.7: Roller Description

Roller description :	
Material	POM(polyoxymethylene)
Color	Black + Silver
Bearing type	625
Inner diameter	5mm
Outer diameter	24mm
Whole thickness	10mm
Top thickness	6mm
No. of wheels used	12

2.1.8 WOODEN STORAGE TANK:

Wooden storage tank is used to store the filling material that is to be deposited in the pothole. Four holes of 12 mm diameter are drilled at the bottom of the tank so as to create a way for the material to move down to pothole. This tank is mounted on the MS Frame.

Table 2.1.8: Tank Dimensions

Length	270 mm
Width	270 mm
Height	330 mm

Figure 2.1.8: Wooden Storage Tank

2.1.9 MS FRAME:

MS Frame is used to keep the Wooden Storage Tank at some elevated height. Since the frame need to withstand load of around 10 to 12 kg, MS material is preferred. This frame is prepared using the (1x1) inch MS angles of 3 mm thickness. Following angles are welded together to make the MS frame:

Table 2.1.9: MS Frame Description

Angle length	Quantity
3.25 inch	2
3.5 inch	2
30 inch	4
9 inch	4
43 inch	2

Figure 2.1.9: Tank mounted on MS Frame

2.2 CONSTRUCTION OF STRUCTURE:

- After knowing the kind of desired motion I needed for the project, I thought of several of kinds of motor drives like lead screws, conveyor belts, etc. I searched for 2D movement ideas on internet. Then I asked Prof. P.A. Shah about the same and he gave me a wonderful idea of the slotted extruded Aluminium section.

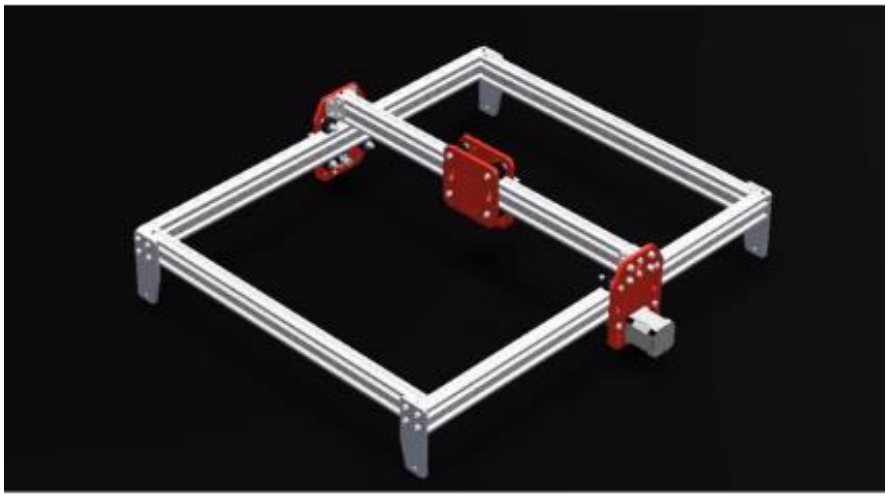

Figure 2.2(a): Aluminium Section Assembly

- Firstly the 4 wooden blocks of required height are set at the required distance below the wooden base.
- Then the two Al sections of 1 meter length are used for Y axis and one Al section is used for X axis.
- Now, the Acrylic cut outs are made as per required dimensions.
- Plastic rollers are set at the required distance and fixed at that location with the help of nuts.

Figure 2.2(b): Motor Assembly

- Acrylic pieces and the rollers are assembled with the section.
- Now the stepper motors are bolted to the acrylic cut outs.
- The pulleys are adjusted on the shat parallel to the slot.

Figure 2.2(c): Acrylic Cut out and Roller assembly

- The timing belt is passed from the teeth of pulley and then tightened at the end of the section with the help nut and bolt.
- Now, finally the ultrasonic sound sensor (HCSR04) is mounted on the X-axis acrylic sheet cut out.

Figure 2.2(d): Belt Tightening

Chapter 3

Electronics

3.1 ELECTRONIC COMPONENTS:

1. Arduino MEGA 2560
2. Ultrasonic Sound Sensor HCSR04
3. Stepper Motor Driver A4988
4. Stepper Motor Nema17
5. SMPS Power Adapter
6. Breadboard
7. Jumper Cable

3.1.1 ARDUINO MEGA:

Overview:

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.

The Mega 2560 is an update to the Arduino Mega, which it replaces.

Pinout Diagram:

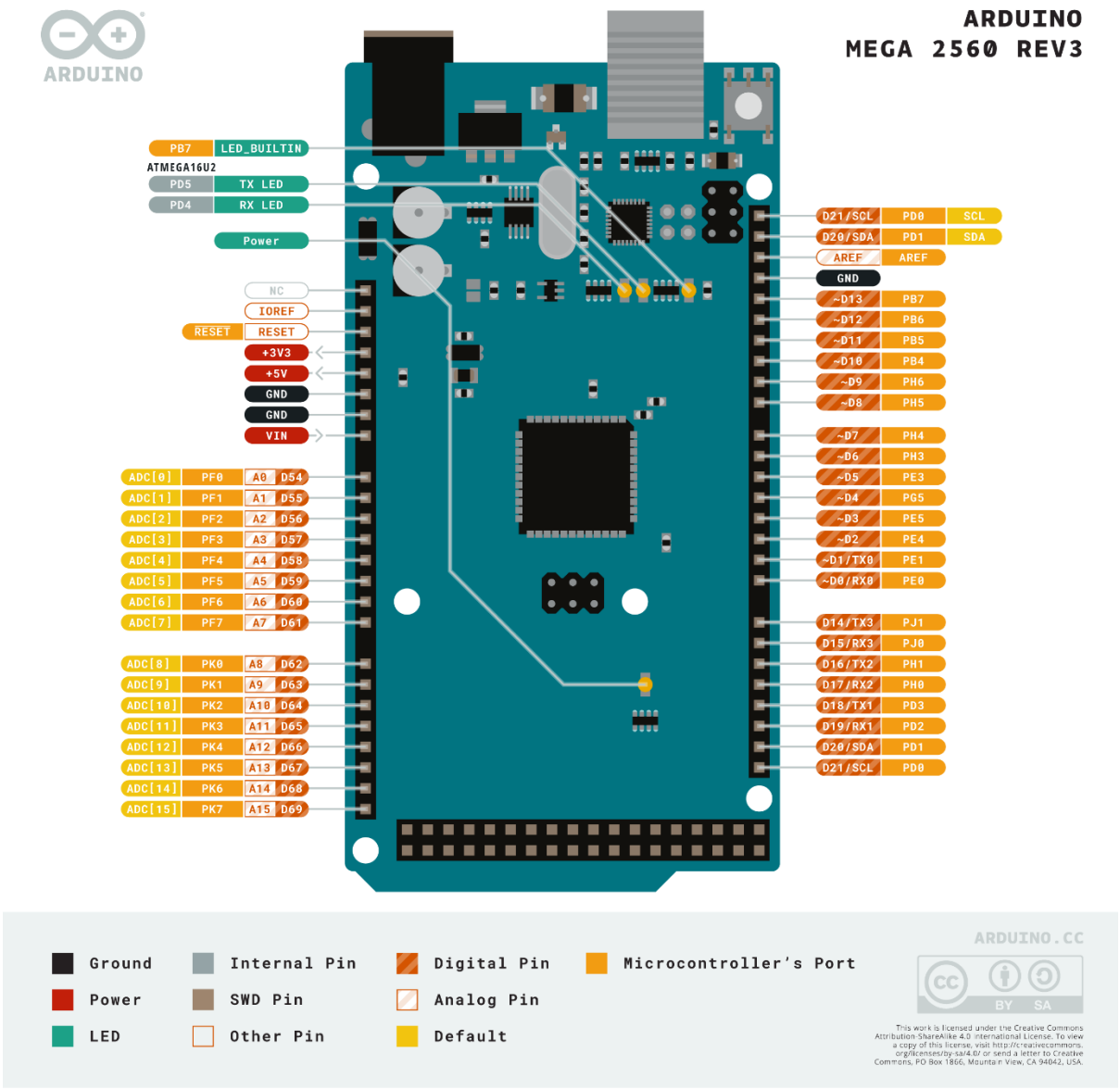

Figure 3.1.1: Arduino Mega Pinout Diagram

Power:

The Mega 2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power

jack. Leads from a battery can be inserted in the GND and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may become unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- Vin. The input voltage to the board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.
- IOREF. This pin on the board provides the voltage reference with which the microcontroller operates. A properly configured shield can read the IOREF pin voltage and select the appropriate power source or enable voltage translators on the outputs for working with the 5V or 3.3V.

Memory:

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output:

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive 20 mA as recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50 k ohm. A maximum of 40mA is the value that must not be exceeded to avoid permanent damage to the microcontroller.

In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega16U2 USB-to-TTL Serial chip.
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low level, a rising or falling edge, or a change in level. See the attachInterrupt() function for details.
- PWM: 2 to 13 and 44 to 46. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Arduino /Genuino Uno and the old Duemilanove and Diecimila Arduino boards.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- TWI: 20 (SDA) and 21 (SCL). Support TWI communication using the Wire library. Note that these pins are not in the same location as the TWI pins on the old Duemilanove or Diecimila Arduino boards.

See also the mapping Arduino Mega 2560 PIN diagram.

The Mega 2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function. There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Table 3.1.1: Arduino Mega Technical Specifications

MICROCONTROLLER	ATmega2560
OPERATING VOLTAGE	5V
INPUT VOLTAGE (RECOMMENDED)	7-12V
INPUT VOLTAGE (LIMIT)	6-20V
DIGITAL I/O PINS	54 (of which 15 provide PWM output)
ANALOG INPUT PINS	16
DC CURRENT PER I/O PIN	20 mA
DC CURRENT FOR 3.3V PIN	50 mA
FLASH MEMORY	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
CLOCK SPEED	16 MHz
LED_BUILTIN	13
LENGTH	101.52 mm
WIDTH	53.3 mm
WEIGHT	37 g

3.1.2 ULTRASONIC SOUND SENSOR HCSR04:

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).

In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emissions of the sound by the transmitter to its contact with the receiver. The formula for this calculation is D = ½ T x C (where D is the distance, T is the time, and C is the speed of sound ~ 343 meters/second).

Ultrasonic Sensor HC-SR04 is a sensor that can measure distance. It emits an ultrasound at 40 000 Hz (40kHz) which travels through the air and if there is an object or obstacle on its path It will bounce back to the module. Considering the travel time and the speed of the sound you can calculate the distance.

The configuration pin of HC-SR04 is VCC (1), TRIG (2), ECHO (3), and GND (4). The supply voltage of VCC is +5V and you can attach TRIG and ECHO pin to any Digital I/O in your Arduino Board.

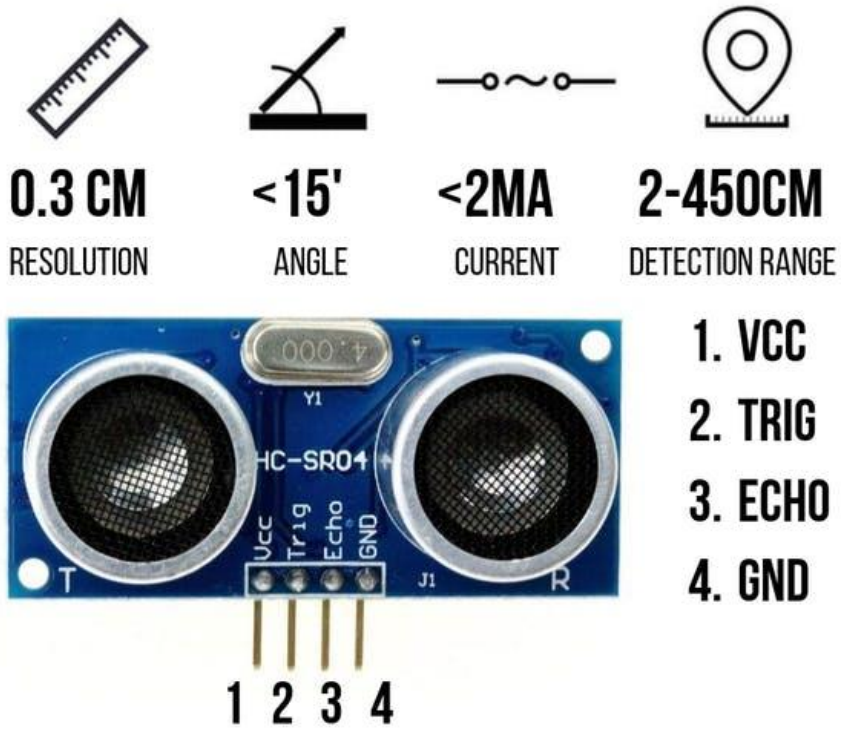

Figure 3.1.2: Ultrasonic Sensor HCSR04

Table 3.1.2: HCSR04 Technical Specifications

Operating Voltage	5V DC
Operating Current	15mA
Operating Frequency	40kHz
Min Range	2cm/1inch
Max Range	400cm/13feet
Accuracy	3 mm
Measuring Angle	<15 degree
Dimensions	45 x 20 x 15 mm

3.1.3 STEPPER MOTOR DRIVER A4988:

The A4988 is a microstepping driver for controlling bipolar stepper motors which has built-in translator for easy operation. This means that we can control the stepper motor with just 2 pins from our controller, or one for controlling the rotation and the other for controlling the steps.

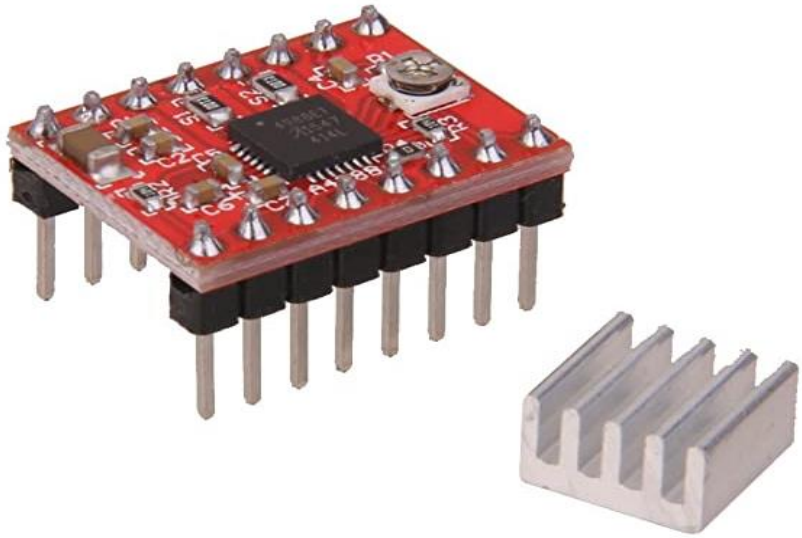

Figure 3.1.3(a): A4988 Stepper Motor Driver

The Driver provides five different step resolutions: full-step, 1/2-step, 1/4-step, 1/8-step and 1/16 step. Also, it has a potentiometer for adjusting the current output, over temperature thermal shutdown and crossover-current protection.

Its logic voltage is from 3 V to 5.5 V and the maximum current per phase is 2A, if enough cooling is provided or 1 A continuous current per phase without heat sink or cooling.

A4988 Pinout:

Now let's closely look at the pinout of the driver and hook it up with the stepper motor and the controller. So we will start with the 2 pins on the bottom right side for powering the driver, the VDD and GND pins that we need to connect with the 3.3 or 5 V and in our case it is the 5 V pin of Arduino Mega. The following 4 pins are for connecting the motor. The 1A and 1B pins will be connected to one coil of the motor and the 2A and 2B pins to the other coil of the motor. For powering the motor we use the next 2 pins, GND and VMOT that we need to connect to

the Power Supply from 8 V to 35 V and we also need to use the decoupling capacitor with at least 47 micro Farad capacity so as to protect the driver board from voltage spikes.

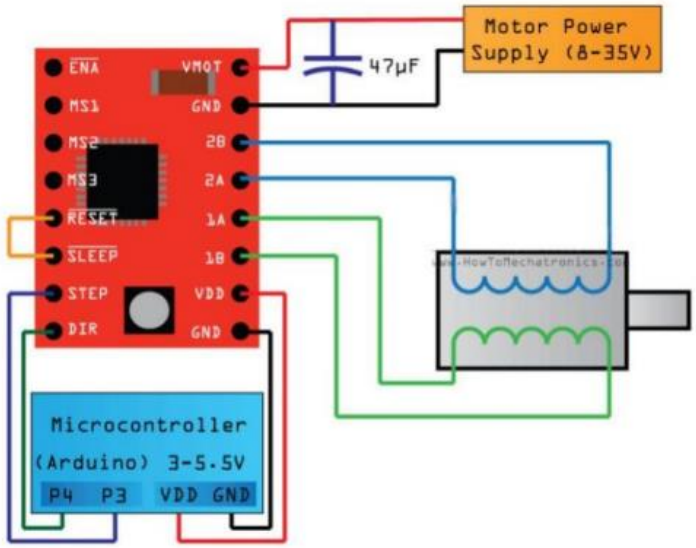

Figure 3.1.3(b): A4988 Driver Pinout

The next two pins, Step and Direction are the pins that we actually use for controlling the motor movements. The direction pin controls the direction of rotation of the motor and we need to connect it to one of the digital pins on our microcontroller board, in our case the Arduino Mega.

With the Step pin, we can control the micro steps of the motor and with each pulse sent to this pin, the motor moves one full step. That means, we don't need any complex programming, phase sequence tables, frequency control lines and so on. Because the built-in translator of the A4988 Driver takes care of everything.

Next is the SLEEP pin. It is an active low pin, low logic puts the board in sleep mode for minimizing power consumption when the motor is not in use.

Next the RESET pin. It sets the translator to a predefined home state. This home state or home microstep position can be seen from A4988 datasheet. If the input state to this pin is logic low then all the STEP inputs will be ignored. The RESET pin is a floating pin, so if we don't have intention of controlling it by the program, then we need to connect it to the SLEEP pin so as to bring it high and enable the board.

Table 3.1.3: Microstepping Pin Configuration

MS1	MS2	MS3	Resolution
LOW	LOW	LOW	Full Step
HIGH	LOW	LOW	Half Step
LOW	HIGH	HIGH	Fourth Step
HIGH	HIGH	LOW	Eighth Step
HIGH	HIGH	HIGH	Sixteenth Step

3.1.4 STEPPER MOTOR NEMA17:

This hybrid bipolar stepping motor has a 1.8° step angle (200 steps/revolution). Each phase draws 1.7 A at 2.8 V, allowing for a holding torque of 4.2 kg-cm. The motor has four color-coded wires terminated with bare leads: black and green connect to one coil; red and blue connect to the other. It can be controlled by a pair of suitable H-bridges (one for each coil), but we recommend using a bipolar stepper motor driver or one of our Tic Stepper Motor Controllers. In particular, the Tics make control easy because they support six different interfaces (USB, TTL serial, I²C, RC, analog voltages, and quadrature encoder) and are configurable over USB with our free configuration utility.

Figure 3.1.4(a): Nema17 Stepper Motor

3.1.4: Nema17 Technical Specifications

Size	42.3 mm square × 38 mm
Weight	285 g (10 oz)
Shaft diameter	5 mm
Steps per revolution	200
Current rating	1.68 A per coil
Voltage rating	2.8 V
Resistance	1.65 Ω per coil
Holding torque	4.2 kg-cm
Inductance	3.2 mH per coil
Lead length	30 cm (12″)

Motor Dimensions:

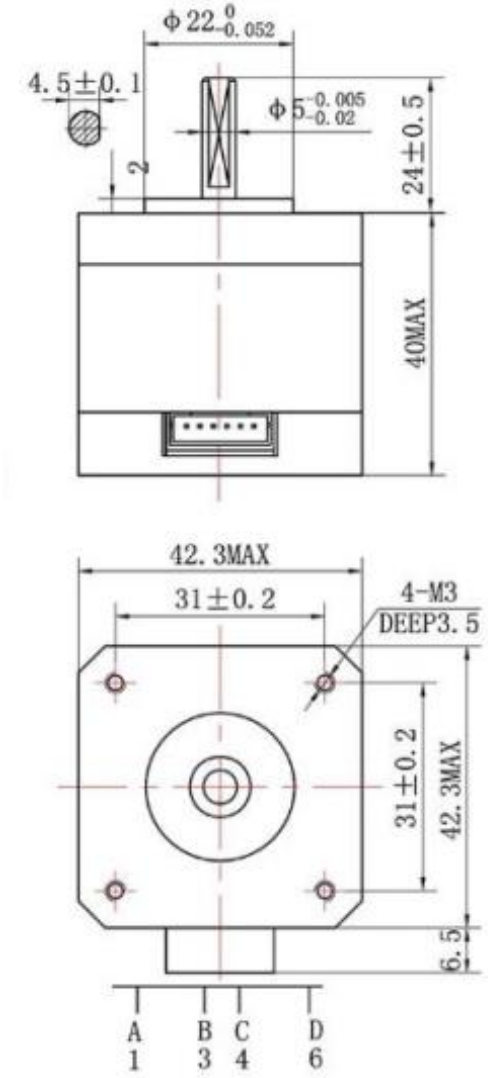

Figure 3.1.4(b): Nema17 Dimensions

3.1.5 SMPS POWER ADAPTER:

Adapter is used to convert the high supply voltage into the desired or operating range of the system. Generally the stepper motors require supply anywhere between 8V to 35V. So I have used 12V adapter and I have selected the ampere rating of the adapter based on the number of stepper motors and their current ratings.

Figure 3.1.5: SMPS (Switched Mode Power Supply)

Table 3.1.5: SMPS Specifications

Input Power	100-240 VAC
Output Power	DC 12V/8Amp

3.1.6 BREAD BOARD:

A breadboard, or protoboard, is a construction base for prototyping of electronics. Originally the word referred to a literal bread board, a polished piece of wood used when slicing bread. In the 1970s the solderless breadboard (a.k.a. plugboard, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.

Because the solderless breadboard does not require soldering, it is reusable. This makes it easy to use for creating temporary prototypes and experimenting with circuit design. For this reason,

solderless breadboards are also popular with students and in technological education. Older breadboard types did not have this property. A stripboard (Veroboard) and similar prototyping printed circuit boards, which are used to build semi-permanent soldered prototypes or one-offs, cannot easily be reused. A variety of electronic systems may be prototyped by using breadboards, from small analog and digital circuits to complete central processing units (CPUs).

Compared to more permanent circuit connection methods, modern breadboards have high parasitic capacitance, relatively high resistance, and less reliable connections, which are subject to jostle and physical degradation. Signaling is limited to about 10 MHz, and not everything works properly even well below that frequency.

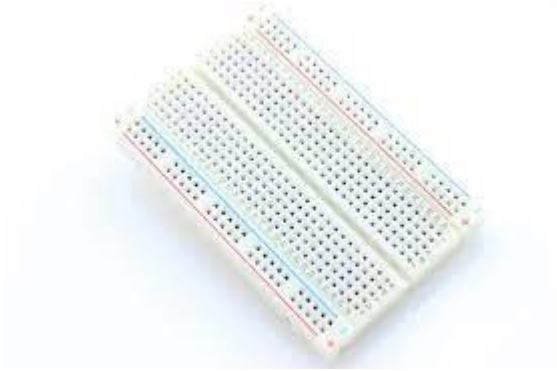

Figure 3.1.6: Bread Board

3.1.7 JUMPER CABLES:

Jumper cables are very flexible and easily detachable cable to the no. of wires according to your requirement. It has 1Pin male to the 1pin male header with both ends. Also, it is compatible with 2.54 mm mil spacing pin headers.

This cable is an electrical wire or group of them in a cable with a connector or pins at each end, which is normally for interconnecting the components of a bread board or other prototype or test circuit, internally or with other equipment or components, without soldering.

Individual Dupont Cables are fitted by inserting their end connectors into the slots provided in a breadboard, the header connect or of a circuit board, or a piece of test equipment.

Mostly it is useful with Banana Pi, Raspberry Pi, Arduino, and other mini pc and development board. It is very useful in the PCB project, pc motherboard, as well as Breadboard connections. Additionally, it allows you to plug and unplug easily for prototyping and can be used over and over again.

3.2 BUILDING UP CIRCUIT:

- Firstly, let us see how to connect a stepper motor with the Arduino Mega. First of all connect the Nema17 connecting wires with the motor driver A4988.
- Secondly, connect the VDD & consecutive GND pin with the power supply and ground row of bread board respectively.
- Now, connect the VMOT and the consecutive GND pin with the other Power supply and ground row of the bread board, where we will connect the 12V 8Amp supply.
- Now, plug in the adapter of SMPS in the regular power supply socket in the switch board and connect the other end (that is the female pin) with the power supply and ground row as mentioned in the above point. Also connect the decoupling capacitor between this two rows.
- Now, connect the Direction Pin and Step Pin of the motor driver with any of the I/O pins of the Arduino Mega.

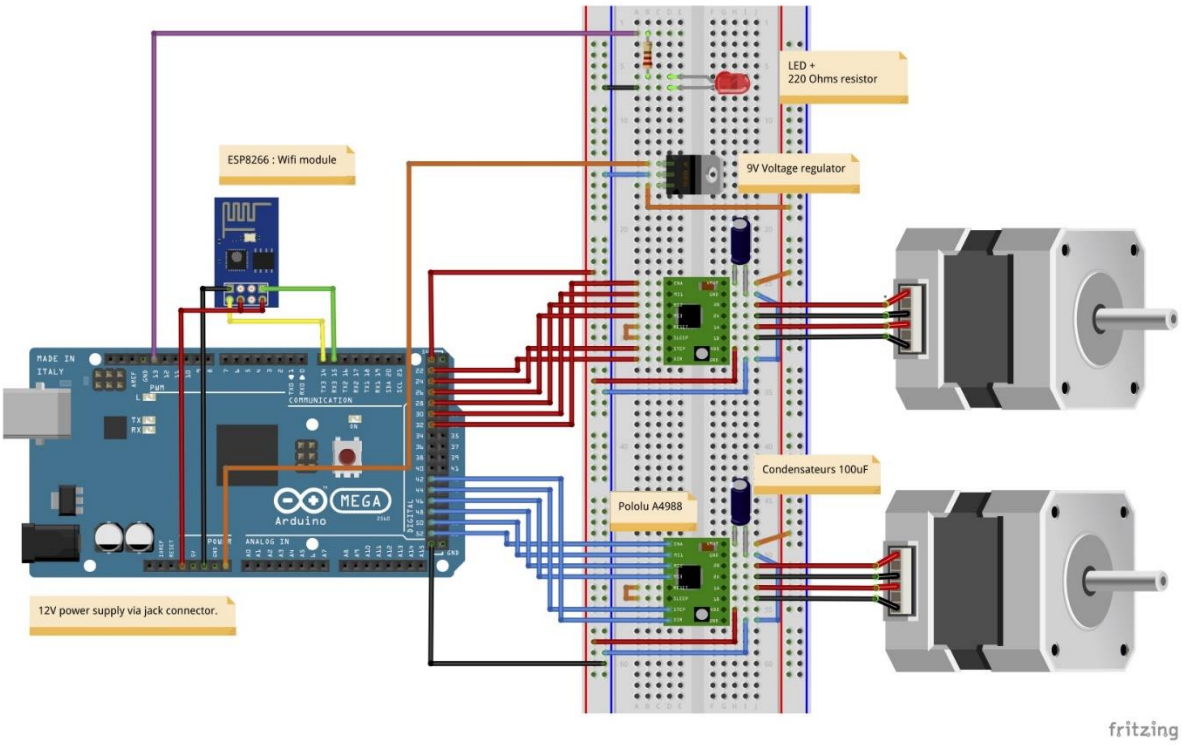

Figure 3.2(a): Motor, Driver & Arduino Mega connection

- Connect the stepper motors of Y axis, X axis and the stepper motor connected at the tank as per the above mentioned steps.
- Now, let us see the connection between ultrasonic sound sensor HCSR04 with the Arduino Mega. There are 4 pins in the HCSR04 sensor. Connect the VCC and GND pin with the power supply row and ground row of the bread board. Now, connect the

TRIG and ECHO pin with any of the I/O pins of the Arduino mega, I have connected with 8th and 9th pin respectively.

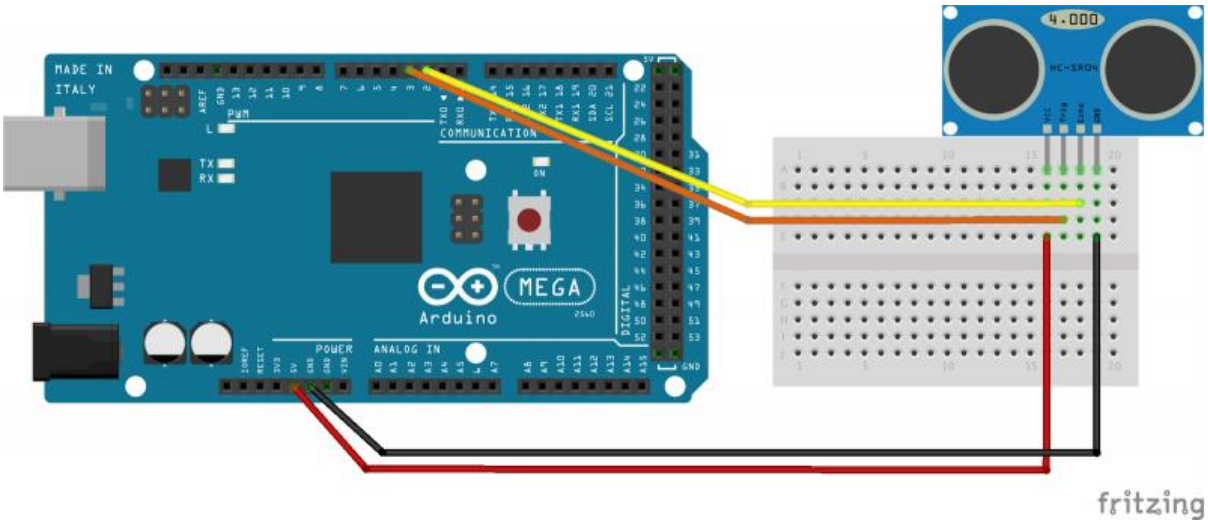

Figure 3.2(b): Ultrasonic Sensor Connection with Arduino Mega

3.3 BILL OF ELECTONIC COMPONENTS:

COMPONENT	QUANTITY
Arduino Mega	1
HCSR04 Sensor	2
A4988 Motor Driver	4
Nema17 Stepper Motor	4
SMPS	1
Breadboard	3
Jumper Cables	Required Quantity

Chapter 4

The Code

Firstly, download the HCSR04 library from internet or from the Arduino IDE itself, then include the library in the code & proceed with the following code:

```
#include <HCSR04.h

HCSR04 hc(8, 9); //initialisation class HCSR04 (trig pin , echo pin)

int rec_mxF=0;

int rec_myF=0;

int rec_mxR=0;

int rec_myR=0;

int pos_mxs=0;

int pos_yfr=0;

int togo_y=0;

int togo_x=0;

int y_till=0;

int x_till=0;

int flag_x_till=0;

int fill_xlow[24]={0};

int fill_xhigh[24]={0};

int ystart=0;

int yend=0;

int xmid=0;

int ymid=0;

int ystartflag=0;
```

```
int xlow_flag=0;
int rec_Fflag=0;
int rec_Rflag=0;
int half_flag=0;
int dir_count=0;
int dcount[24]={0};//it records the number of target datapoints for each strip
float depth=0;
float store_depth[24]={0};
float length_strip=0;
float width_strip=1.6;
float volume=0;
const int dirPin_1 = 2;
const int stepPin_1 = 3;
const int dirPin_2 = 6;
const int stepPin_2 = 7;
const int dirPin_3 = 4;
const int stepPin_3 = 5;
void setup() {
  Serial.begin(9600);
  pinMode(stepPin_1, OUTPUT);
  pinMode(dirPin_1, OUTPUT);
  pinMode(stepPin_2, OUTPUT);
  pinMode(dirPin_2, OUTPUT);
  pinMode(stepPin_3, OUTPUT);
  pinMode(dirPin_3, OUTPUT);
```

```
pos_mxs=0;
 for(int FR=0; FR<2; FR++){
  if(FR%2==0){
  digitalWrite(dirPin_1, LOW);
  digitalWrite(dirPin_2, HIGH);
    }
  else{
  digitalWrite(dirPin_1, HIGH);
  digitalWrite(dirPin_2, LOW);
  rec_Rflag=1;
    }
  for(int my=0; my<12; my++){//----------------global or local ?????----------------
    if(pos_mxs==0){
        digitalWrite(dirPin_3, HIGH);
        Serial.println("Wrote dirPin_3, HIGH");
          }
    if(pos_mxs==4200){
        digitalWrite(dirPin_3, LOW);
        Serial.println("Wrote dirPin_3, LOW");
          }
    if(pos_mxs>0 && pos_mxs<4200){
     digitalWrite(stepPin_3, LOW);
     Serial.println("line *_*_*_*_*_*_*_*_*_*117*_*_*_*_*_*_*_*_*_*_");
     Serial.println("pos_mxs between 0 & 4200, wrote stepPin_3 LOW");
```

```
  Serial.println("Unplug Arduino with in 10 SECONDS");
  Serial.println("Change The Code Please:(");
  delay(10000);
  }
if(rec_Fflag!=1 || rec_Rflag==1){
for(int mx=0; mx<21; mx++){
  for(int mxf=0; mxf<200; mxf++){
     digitalWrite(stepPin_3, HIGH);
     delayMicroseconds(2000);
     digitalWrite(stepPin_3, LOW);
     delayMicroseconds(2000);
     if(my%2==0){
       pos_mxs++;
       }
      else{
         pos_mxs--;
         }
    }
   delay(60);
   depth = hc.dist();
   Serial.println(depth);
   if(rec_Fflag!=1){
    if(depth>22){
      rec_mxF=pos_mxs;
      rec_myF=pos_yfr;
```

```
      Serial.println("Recorded First *FORWARD* Depth Change Point");
      Serial.print(depth);
      Serial.print(" at ");
      Serial.print(rec_mxF);
      Serial.print(" ");
      Serial.println(rec_myF);
      rec_Fflag=1;
      }
     }
    if(rec_Rflag==1){
     if(depth>22){
      rec_mxR=pos_mxs;
      rec_myR=pos_yfr;
      Serial.println("Recorded First *REVERSE* Depth Change Point");
      Serial.print(depth);
      Serial.print(" at ");
      Serial.print(rec_mxR);
      Serial.print(" ");
      Serial.println(rec_myR);
      rec_Rflag=2;
      }
     }
  }
 }
 for(int jm = 0; jm < 200; jm++)
```

```
  {
    digitalWrite(stepPin_1, HIGH);
    digitalWrite(stepPin_2, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin_1, LOW);
    digitalWrite(stepPin_2, LOW);
    delayMicroseconds(2000);
  }
  if(FR%2==0){
  pos_yfr++;
  }
  else{
    pos_yfr--;
    }
}
if(pos_mxs==4200){
 digitalWrite(dirPin_3, LOW);
 Serial.println("MOTOR_3 at 4200 steps, Bringing MOTOR_3 to 0 steps");
  while(pos_mxs!=0){
      digitalWrite(stepPin_3, HIGH);
      delayMicroseconds(2000);
      digitalWrite(stepPin_3, LOW);
      delayMicroseconds(2000);
      pos_mxs--;
    }
```

```
      delay(10);
    }
  else{
    Serial.println("MOTOR_3 already at 0 steps");
    }
   if(pos_mxs>0 && pos_mxs<4200){
    Serial.println("*_*_*_*_*_*_*_*_*_line 190*_*_*_*_*_*_*_*_*_*");
    Serial.println("MOTOR_3 positon between 0 & 4200, NEED TO CHANGE THE CODE");
    Serial.println("Unplug Arduino within 10 SECONDS");
    delay(10000);
    }
}
Serial.println("-----------Current POSITIONS-------------");
Serial.print("Y Position: ");
Serial.println(pos_yfr);
Serial.print("X Position: ");
Serial.println(pos_mxs);
Serial.println("*_*_*_*_*_*_*_*_Going again near the First FORWARD Depth Change Point_*_*_*_*_*_*_*_*");
Serial.println(" ");
//-------------------------Going back to First FORWARD Depth Change Point-----------------------------------------------
//-----here onwards consider yfr-->yhr && mxs-->ms--------
digitalWrite(dirPin_1, LOW);
digitalWrite(dirPin_2, HIGH);
```

```
pos_yfr=0;
if(rec_myF!=0){
  togo_y = ((rec_myF*2)-2);
  Serial.print(togo_y);
  Serial.println(" Half Revolutions to go to FORWARD Depth Change Point");
  }
else{
  togo_y=0;
  Serial.println("Depth change in 0th Strip");
  }
while(pos_yfr!=togo_y){
  for(int a=0; a<100; a++){
        digitalWrite(stepPin_1, HIGH);
        digitalWrite(stepPin_2, HIGH);
        delayMicroseconds(2000);
        digitalWrite(stepPin_1, LOW);
        digitalWrite(stepPin_2, LOW);
        delayMicroseconds(2000);
    }
    pos_yfr++;
  }
if(rec_mxF!=0){
  togo_x = (rec_mxF-200);
  Serial.print(togo_x);
  Serial.println(" x steps to go to FIRST DEPTH CHANGE POINT");
```

```
  }
  else{
    togo_x = 0;
    }
pos_mxs=0;
digitalWrite(dirPin_3, HIGH);
while(pos_mxs!= togo_x){
        digitalWrite(stepPin_3, HIGH);
        delayMicroseconds(2000);
        digitalWrite(stepPin_3, LOW);
        delayMicroseconds(2000);
        pos_mxs++;
  }
y_till = ((rec_myR*2)+2);
if(y_till > 24){
  y_till = 24;
  }
x_till = (rec_mxR+200);
if(x_till > 4200){
  x_till = 4200;
  }
dir_count=0;
for(int ystrip = togo_y; ystrip<=y_till ; ystrip++){
xlow_flag=0;
```

```
if(dir_count%2==0){
  digitalWrite(dirPin_3, HIGH);
  Serial.println("wrote MOTOR_3 HIGH line 281");
  }
  else{
    digitalWrite(dirPin_3, LOW);
    Serial.println("wrote MOTOR_3 LOW line 285");
    }
for(int xto=1; xto<=210; xto++){
  if(ystrip == togo_y && half_flag==0){
    xto=(togo_x/20);
    half_flag=1;
    Serial.println("half loop started successfully");
    }
  if(ystrip == y_till && flag_x_till == 0){
    if(dir_count%2==0){
    xto = ((4200-x_till)/20);
    Serial.println("*_*_*_*_*_*_*_*_*_last strip_*_*_*_*_*_*_*_*_*_*");
    Serial.print("xto value increased to ");
    Serial.println(xto);
    flag_x_till = 1;
    }
    else{
    xto = (210-((4200-x_till)/20));
    Serial.println("*_*_*_*_*_*_*_*_*_last strip_*_*_*_*_*_*_*_*_*_*");
```

```
  Serial.print("xto value increased to  ");
  Serial.println(xto);
  flag_x_till = 1;
    }
  }
for(int xtwen=0; xtwen<20; xtwen++){
   digitalWrite(stepPin_3, HIGH);
   delay(2);
   digitalWrite(stepPin_3, LOW);
   delay(2);
  if(dir_count%2==0){
    pos_mxs++;
   }
  else{
    pos_mxs--;
    }
}
depth = hc.dist();
Serial.println(depth);
        if(depth>17.00){
          store_depth[ystrip]= store_depth[ystrip] + depth;
          dcount[ystrip]++;
          if(ystartflag==0){
            ystart=ystrip;
            ystartflag=1;
```

```
            }
            yend = ystrip;
            if(xlow_flag==0){
            fill_xlow[ystrip] = pos_mxs;
            xlow_flag=1;
            }
            fill_xhigh[ystrip] = pos_mxs;
              }
             delay(60);
}
if(store_depth[ystrip]>0){
store_depth[ystrip] = (store_depth[ystrip]/(dcount[ystrip]));
}
Serial.print("X range for ");
Serial.print(ystrip);
Serial.print("th strip is ");
Serial.print(fill_xlow[ystrip]);
Serial.print(" - ");
Serial.println(fill_xhigh[ystrip]);
  for(int yf = 0; yf < 100; yf++)
    {
      digitalWrite(stepPin_1, HIGH);
      digitalWrite(stepPin_2, HIGH);
      delayMicroseconds(2000);
      digitalWrite(stepPin_1, LOW);
```

```
      digitalWrite(stepPin_2, LOW);
      delayMicroseconds(2000);
    }
    pos_yfr++;
    dir_count++;
 }
//_____Bringing Back to Initial Position_____
digitalWrite(dirPin_1, HIGH);
digitalWrite(dirPin_2, LOW);
while(pos_yfr != 0){
  for(int y_back=0; y_back<100; y_back++){
    digitalWrite(stepPin_1, HIGH);
    digitalWrite(stepPin_2, HIGH);
    delayMicroseconds(2000);
    digitalWrite(stepPin_1, LOW);
    digitalWrite(stepPin_2, LOW);
    delayMicroseconds(2000);
  }
  pos_yfr--;
  delay(10);
}
 //----------*_*_*_*_*_*_*_*_*_*_*_*_Calculating Volume_*_*_*_*_*_*_*_*_*_*_*_*----------
```

```
  for(int j=0; j<24; j++){
    Serial.print("Average Depth ");
    Serial.print(j);
    Serial.print(": ");
    Serial.println(store_depth[j]);
    length_strip = dcount[j]*0.32;
    volume = (volume + (store_depth[j]*length_strip*width_strip));
    Serial.print("Volume addition: ");
    Serial.println(volume);
    }
    Serial.println("-----------------------------------------------------------------------------");
    Serial.println("-----------------------------------------------------------------------------");
    Serial.println("-----------------------------------------------------------------------------");
    Serial.println(volume);
Serial.print("Pothole Y range is ");
Serial.print(ystart);
Serial.print(" - ");
Serial.println(yend);
//-------------filling point at center-------------------
ymid = ((ystart + yend)/2);
xmid = (((fill_xlow[ymid])+(fill_xhigh[ymid]))/2);
Serial.print("*_*_*_*_*_*_*_*_mid point is X-");
Serial.print(xmid);
Serial.print(" Y-");
Serial.println(ymid);
```

```
}

void loop(){

  }
```

Chapter 5

Project Working

5.1 Working:

The sensing range of project is first divided into small squares of size 3.2 cm x 3.2 cm. HCSR04 sensor is placed at the initial point (0, 0) by the motor and as per algorithm, the sensor is move 3.2 cm ahead and program first searches for depth change points at the nodes of these 3.2 cm squares. When it find first depth change point, the position of motors of X-axis and Y-axis is recorded. Now, the sensor is moved to extreme last corner point by the motor. The sensor is again moved by motor in steps of 3.2 cm. The program again starts searching for the first depth change point in this direction. The position of motors is recorded when depth change point is found. Now, the motor moves sensor near the very first depth change point. Now onwards the motor will move the HCSR04 sensor in steps of 3.2mm to note the depth. This movement of 3.2 mm will continue until the depth of whole region between the two depth change points (whose position we just recorded) is measured.

Now after the depth is measured over whole region, volume of pothole is calculated as per the algorithm written in the code (the same is explained in Chapter 6 topic 6.1). The mid-point of the pothole is also estimated as per the algorithm in code. The motors are given command to move to this mid-point, so as to deposit the material.

Before depositing the material, “target depth” is defined. Target depth is the depth of the tank corresponding to the volume equal to the volume of pothole. Target depth will be used as a measure unto which the material from tank will be allowed to dispatch.

There is an HCSR04 sensor on the tank. The program first measure initial depth between the sensor and the upper layer of filling material. Program then sends command to motor (attached below the tank), to open the gates so as to dispatch material. This gates are opened until the depth between the HCSR04 sensor and becomes greater than or equal to (initial depth + target depth). Gates are then closed by the same motor, as per command sent by the program.

5.2 Detailed Working Step By Step:

- After constructing the structure, preparing the circuit and uploading code to Arduino Mega, the project operates in following manner.
- Firstly, program sends command to Motor_3 (this motor is responsible for movement along X-axis, this is the motor that moves the HCSR04 sensor) to move in steps of 200 steps or full revolutions that is 3.2 cm linear distance.
- The HCSR04 sensor is also moved along with Motor_3 and the sensor sends data of depth measurement to the program. When the program finds depth change, it records position of all motors (both along X-axis & Y-axis) at that point. There can be total of 21 steps of 3.2 cm size along X-axis. When this 21 steps are completed, the Motor_1 and Motor_2 (both this motors are responsible for movement along Y-axis and it moves the whole bridge like structure along with the HCSR04 sensor and Motor_3) moves 3.2 cm along Y-axis. And Motor_3 is given command to rotate in reverse direction in the same step size of 3.2 cm

- After recording the first depth change point, that motors are given command to move to the extreme end of X-axis and to the extreme end of Y-axis. The motors are then given command to operate in reverse directions in the same step size of 3.2 cm
- When the first depth change (let's call this "Last depth change point of pothole") point is recorded during this movement, the again the positions of all motors are recorded.
- Now we have 2 points, between which the pothole is present. The program sends command to motors to move near the very first depth change point.
- Now onwards the Motor_3 is given command to move in step size of 20 that is 3.2 mm linear distance and the program measures depth at every 3.2mm. Once the Motor_3 is reached at end of X-axis, Motor_1 and Motor_3 are given command to move 1.6 cm along Y-axis, and Motor_3 starts rotating in reverse direction in step size of 3.2 mm
- The motor movements continue till the "Last depth change point of point" that we already recorded earlier.
- All the depths that are greater than the reference depth are taken into consideration for volume estimation.
- After calculating the volume, "Target depth" is defined. Read 5.1 to understand about Target depth.
- After getting the Target depth, initial depth between upper surface of the material and HCSR04 sensor attached above tank is recorded.
- The motor connected below the tank is given command to open the gates of tank.
- The material is allowed to dispatch from the tank till the depth between the upper surface of material and HCSR04 becomes greater than or equal to (initial depth + target depth)
- Program then sends command to motor below the tank to close the gates.
- Whole procedure of estimating the volume and depositing material in the pothole is completed here.

Chapter 6

Calculations, Analysis and Results

6.1 Volume Approximation:

6.1.1 Recording Depths:

The X-axis stepper motor moves 20 steps at every 60ms that is approximately 3mm. The HCSR04 sensor is attached with this acrylic cut out. So, the HCSR04 measure depth at every 3.2 mm and if the depth is greater than the reference depth, the Program/Code will consider this length for the Volume Approximation. Y-axis motors will move 100 steps (1.6 cm) after that X-axis motor completes 4200 steps. The Program records depths strip wise (a strip is 1.6 cm wide). Each strip can have number of depth change points anywhere between 0 and 210.

6.1.2 Average Depths:

Now that the program have all the depths, the program now calculates the average depth for each strip, based on the number of depth change points recorded in that particular strip.

6.1.3 Calculating Length of Strip:

The program records that number of depth change points for every strip. Now as I mentioned earlier, the HCSR04 is recording depth at every 3.2 mm. So the length of each strip is equal to multiplication of number of depth change points with 3.2 mm.

6.1.4 Calculating Volume:

After getting average depths of all the strips, the Program calculated volume by multiplying average depth with the width of the strip (that is 1.6 cm) and also by multiplying it with the length of each strip.

6.2 HCSR04 Sensor Error:

6.2.1 Angular Error:

After running several tests, I found that the ultrasonic sound sensor HCSR04 is not able to measure the depth of the pothole on its immediate boundary. This is because, the HCSR04 sensor have an angular detection range of about 15 degree. Because of this angular detection range drawback, the sensor cannot detect the depth exactly below the sensor. In fact, it returns the depth based on the very first sound wave received by the receiver of the sensor. I found that, because of this limitation, the sensor cannot measure depth accurately for approximately 1.5 cm to 2 cm of region in the direction perpendicular to its length.

6.2.2 Error Due To Sensor Length:

The HCSR04 is about 4 cm long. And this is the reason that it cannot measure the depth in the 4 cm of region in the direction of movement along its length. So, total of 8 cm of region's depth cannot be measured properly, 4 cm when the sensor enters the pothole boundary and another 4 cm when the sensor leaves the pothole boundary.

6.3 Time Taken By HCSR04:

HCSR04 sensor requires minimum of 60ms of time between two consecutive depth measurements. This turns out to be a major drawback of the HCSR04 sensor when we talk about time optimization of the project, because there are thousands of datapoints to be considered.

Chapter 7

Cost Estimation

Sr. No.	Component	Quantity	Price	Cost
1.	Al Section	3	500	1500
2.	GT2 Timing Belt 5m	1	550	550
3.	Gt2 Timing Pulley	3	50	150
4.	M5 Nut & Bolt	8	10	80
5.	M5 Extra Nuts	40	2	80
6.	Long Nut & Bolt	4	5	20
7.	MS Frame	1	700	700
8.	Plastic Rollers	12	120	1440
9.	Arduino Mega	1	700	700
10.	Nema17 Stepper Motor	4	500	2000
11.	A4988 Driver	4	130	520
12.	SMPS	1	390	390
13.	Female Pin	1	10	10
14.	Breadboard	1	60	60
15.	Jumper Cables	1 pack	40	40

Grand Total = 8240

Chapter 8

Conclusion

Main purpose of this project is to solve the issue of large number of unrepaired potholes on road mainly on highways and to overcome the difficulty of filling them under hot weather. To achieve this objective, I researched to decide the structure of project. I searched for the proper motor drive I needed to get 2D movement. I found resources on Internet but the ideas from faculties and seniors were much more helpful.

I learned many new concepts while making this project. I learned how to control stepper motor with Arduino, the importance of proper sequencing. I programmed Arduino mega to collect depth measurement from HCSR04 sensor and to give appropriate command to stepper motor. I learned the importance of an optimized code both in terms of time optimization and space optimization.

After preparing structure, circuit and the code, I started running tests on project. I have explained my observation and analysis of this tests in detail in Chapter 6: CALCULATIONS, ANALYSIS & RESULTS.

Chapter 9

Improvisation and Future Scope

- Further improvements can be made in this project by using an accurate distance measurement sensor, which have very minimal angular detection range, which do not have issue in measuring the depth while moving in the direction along its length and the one whose operating frequency is high as compared to HCSR04. Of course, such accurate sensor will cost far more than the very economical HCSR04 sensor.
- There is a scope of new and more time efficient algorithm to scan the pothole.
- A hydraulic pressing mechanism can be added in this project to press the deposited material properly and to minimal the voids in between the molecules of the material.
- A RC drivable frame can be prepared on which the project can be mounted, so that the project can be easily moved to a pothole.

References

Internet Resources:

Stepper motor driver A4988 Library:

https://github.com/laurb9/StepperDriver

Stepper motor controlling using Arduino and A4988 driver, howtoEelectronics:

https://how2electronics.com/control-stepper-motor-with-a4988-driver-arduino/

Interfacing & connecting A4988 Guide along with setting up current limit:

https://lastminuteengineers.com/a4988-stepper-motor-driver-arduino-tutorial/

Data logging into SD CARD:

https://dronebotworkshop.com/sd-card-arduino/#DataLogger

Other References:

- Laser engraving project made by seniors: Nirav Chhtrola and Aakash Parmar.