SMART WIRELESS GANTRY CRANE SYSTEM

A THESIS AND PROJECT
BY

MD. SABBIR HOSSAIN DURLAB
MD. TALHA BAHADUR
MD. SALIM REZA
MD TARIK AZIZ

DEPARTMENT OF MECHANICAL ENGINEERING
SONARGAON UNIVERSITY

FEBRUARY 2020

SMART WIRELESS GANTRY CRANE SYSTEM

A THESIS AND PROJECT

BY

MD. SABBIR HOSSAIN DURLAB
Student ID: BME 1602009194

MD TALHA BAHADUR
Student ID: BME 1602009090

MD SALIM REZA
Student ID: BME 1602009196

MD TARIK AZIZ
Student ID: BME 1503007250

Supervisor: Arup Kumar Haldar
Lecturer

Submitted to the
DEPARTMENT OF MECHANICAL ENGINEERING
SONARGAON UNIVERSITY
In partial fulfillment of the requirements for the award of the degree
of
BACHELOR OF SCIENCE IN MECHANICAL ENGINEERING

FEBRUARY 2020

SMART WIRELESS GANTRY CRANE SYSTEM

A Thesis and Project

By

MD. SABBIR HOSSAIN DURLAB
MD. TALHA BAHADUR
MD. SALIM REZA
MD TARIK AZIZ

....
Supervisor
Arup Kumar Haldar

DEPARTMENT OF MECHANICAL ENGINEERING
SONARGAON UNIVERSITY

FEBRUARY 2020

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

At first, we want to express gratefulness and humbleness to Almighty Allah for His immense blessing upon us for the successfulness completion of this thesis work.

We would like to express our sincere gratitude to our supervisor, Arup Kumar Haldar, Lecturer, Department of Mechanical Engineering, Sonargaon University for his valuable suggestions, guidance and constant encouragement during pursuit of this work.

We express our deep sense of gratitude and thanks to Professor Md. Mostofa Hossain, Head, Department of Mechanical Engineering, Sonargaon University for his motivation and kind collaboration.

We also would like to express gratitude to Prof. Dr. Md. Abul Bashar , Vice-Chancellor of Sonargaon University for his support in every aspect of our academic career especially for our Mechanical Workshop.

Finally, we would like to express our sincere gratitude to our family who have given support to our study and prayed for our life.

ABSTRACT

The main objective of this work is to design robust, fast, and practical controllers for gantry crane. The controllers are designed to transfer the load from point to point as fast as possible and, at the same time, the load swing is kept small during the transfer by automatic smart control system and completely vanishes at the load destination. Moreover, variations of the system parameters, such as the cable length and the load weight, are also included. Practical considerations, such as the control action power, and the maximum acceleration and velocity, are taken into account. In addition, friction effects are included in the design using a friction-compensation technique.

CHAPTER 1
INTRODUCTION

1.1 Introduction

Cranes are widely used to transport heavy loads and hazardous materials in shipyards, factories, nuclear installations, and high-building construction. They can be classified into two categories based on their configurations: gantry cranes and rotary cranes.

Gantry cranes are commonly used in factories, Figure 1.1. This type of cranes in- corporates a trolley, which translates in a horizontal plane. The payload is attached to the trolley by a cable, whose length can be varied by a hoisting mechanism. The load with the cable is treated as a one-dimensional pendulum with one-degree-of-freedom sway. There is another version of these cranes, which can move also horizontally but in two perpendicular directions. The analysis is almost the same for all of them because the two-direction motions could be divided into two uncoupled one-direction motions.

Rotary cranes can be divided into two types: boom cranes which are commonly used in shipyards, and tower cranes which are used in construction, Figure 1.2. In these cranes, the load-line attachment point undergoes rotation. Another degree of freedom may exist for this point. For boom cranes, this point moves vertically, whereas it moves horizontally in tower cranes. Beside these motions, the cable can be lowered or raised. The cable and the load are treated as a spherical pendulum with two-degree-of-freedom sway.

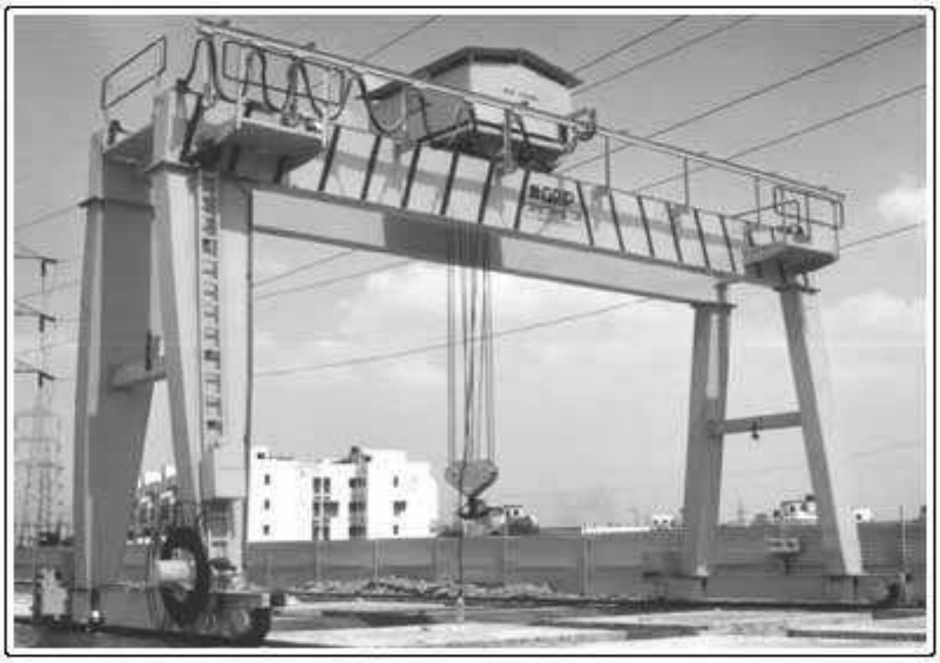

Figure 1.1: Gantry Crane

In this work, we design our controllers based on a linearized model of tower cranes. Hence, the nonlinearities, such as Coulomb friction, are not included. Unfortunately, when the designed controllers were validated on a tower-crane model, we found that the friction is very high. This friction results in high steady-state error for position control even without swing control. If the swing control is included, the response is completely unacceptable. Therefore, controllers designed based on linear models are not applicable to real systems unless the friction is compensated for. This can be done by estimating the friction, and then applying an opposite control action to cancel it, which is known as friction compensation, Figure 1.3. To estimate the friction force, we assume a mathematical model, and then we estimate the model coefficients using an off-line identification technique, such as the method of least squares (*LS*). First, the process of identification is applied to a theoretical model of a DC motor with known friction coefficients. From this example, some guidelines and rules are deduced for the choice of the *LS* parameters. Then, the friction coefficients of the tower-crane model are estimated and validated.

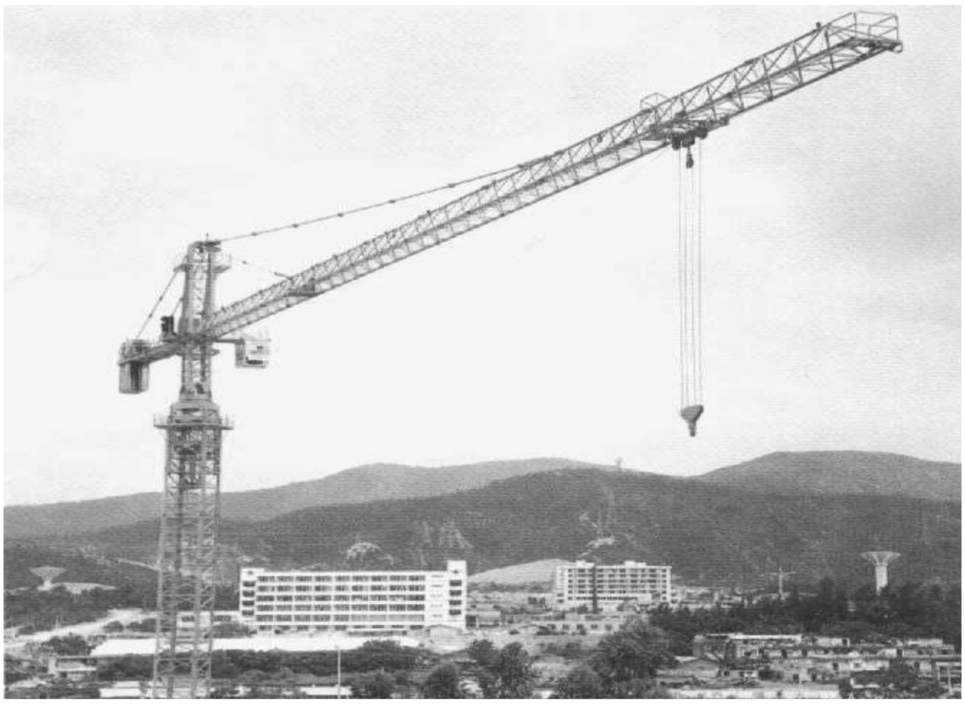
Figure 1.2: Boom crane

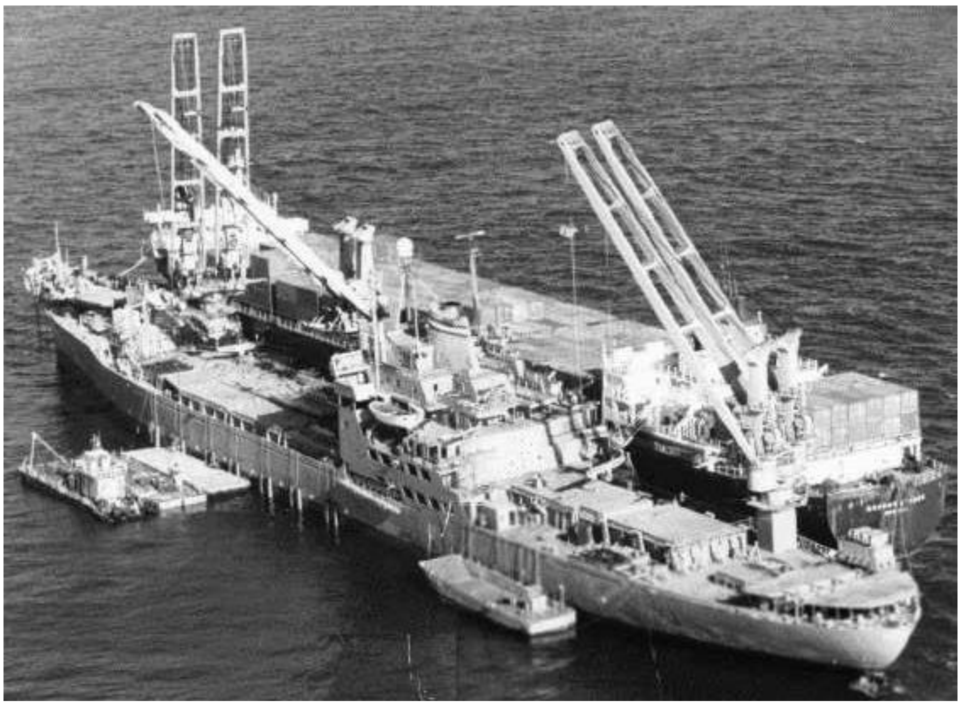

Figure 1.3: Tower crane

.

1.2 Crane Control Approaches

Cranes are used to move a load from point to point in the minimum time such that the load reaches its destination without swinging. Usually a skillful operator is responsible for this task. During the operation, the load is free to swing in a pendulum-like motion. If the swing exceeds a proper limit, it must be damped or the operation must be stopped until the swing dies out. Either option consumes time, which reduces the facility availability. These problems have motivated many researchers to develop control algorithms to automate crane operations. However, most of the existing schemes are not suitable for practical implementation. Therefore, most industrial cranes are not automated and still depend on operators, who sometimes fail to compensate for the swing. This failure may subject the load and the environment to danger. Another difficulty of crane automation is the nature of the crane environment, which is often unstructured in shipyards and factory floors. The control algorithm should be able to cope with these conditions. Abdel-Rahman etal. (2002) presented a detailed survey of crane control. In the following, we concentrate on reviewing the general approaches used in this field.

The operation of cranes can be divided into five steps: gripping, lifting, moving the load from point to point, lowering, and ungripping. A full automation of these processes is possible, and some research has been directed towards that task (Vaha et al., 1988). Moving the load from point to point is the most time consuming task in the process and requires a skillful operator to accomplish it. Suitable methods to facilitate moving loads without inducing large swings are the focus of much current research. We can divide crane automation into two approaches. In the first approach, the operator is kept in the loop and the dynamics of the load are modified to make his job easier. One way is to add damping by feeding back the load swing angle and its rate or by feeding back a delayed version of the swing angle (Henry et al., 2001; Masoud et al., 2002). This feedback adds an extra trajectory to that generated by the operator. A second way is to avoid exciting the load near its natural frequency by adding a filter to remove this frequency from the input (Robinett et al., 1999). This introduces time delay between the operator action and the input to the crane. This delay may confuse the operator. A third way is to add a mechanical absorber to the structure of the crane (Balachandran et al., 1999). Implementing this method requires a considerable amount of power, which makes it impractical.

In the second approach, the operator is removed from the loop and the operation is completely automated. This can be done using various techniques. The first technique is based on generating trajectories to transfer the load to its destination with minimum swing. These trajectories are obtained by either input shaping or optimal control techniques. The second technique is based on the feedback of the position and the swing angle. The third technique is based on dividing the controller design problem into two parts: an anti-swing controller and a tracking controller. Each one is designed separately and then combined to ensure the performance and stability of the overall system.

Since the load swing is affected by the acceleration of the motion, many researchers have concentrated on generating trajectories, which deliver the load in the shortest possible time and at the same time minimize the swing. These trajectories are obtained generally by using optimization techniques. The objective function can be either the transfer time (Manson, 1982), or the control action (Karihaloo and Parbery, 1982), or the swing angle (Sakaw and Shindo, 1981). Another important method of generating trajectories is input shaping, which consists of a sequence of acceleration and deceleration pulses. These sequences are generated such that there is no residual swing at the end of the transfer operation (Karnopp et al., 1992; Teo et al., 1998; Singhose et al., 1997). The resulting controller is open-loop, which makes it sensitive to external disturbances and to parameter variations. In addition, the required control action is bang-bang, which is discontinuous. Moreover, it usually requires a zero-swing angle at the beginning of the process, which cannot be realized practically. To avoid the open-loop disadvantages, many researches (Beeston, 1983; Ohnishi et al., 1981) have investigated optimal control through feedback. They found out that the optimal control performs poorly when implemented in a closed-loop form. The poor performance is attributed to limit cycles resulting from the oscillation of the control action around the switching surfaces. Zinober (1979) avoided the limit cycles by rotating the switching surfaces. This approach can be considered as sub-optimal time control. However, the stability of the system has not been proven. Moreover, the control algorithm is too complex to be implemented practically.

Feedback control is well-known to be less sensitive to disturbances and parameter variations. Hence, it is an attractive method for crane control design. Ridout (1989a) developed a controller, which feeds back the trolley position and speed and the load swing angle. The feedback gains are calculated by trial and error based on the root-locus technique. Later, he improved his controller by changing the trolley velocity gain according to the error signal (Ridout, 1989b). Through this approach, the system damping can be changed during transfer of the load. Initially, damping is reduced to increase the velocity, and then it is increased gradually. Consequently, a faster transfer time is achieved. However, the nominal feedback gains are obtained by trial and error. This makes the process cumbersome for a wide range of operating conditions. Salminen et al. (1990) employed feedback control with adaptive gains, which are calculated based on the pole-placement technique. Since the gains are fixed during the transfer operation, his control algorithm can be best described as gain scheduling rather than adaptation. Hazlerigg (1972) developed a compensator with its zeros designed to cancel the dynamics of the pendulum. This controller was tested on a physical crane model. It produced good results except that the system was under damped. Therefore, the system response was oscillatory, which implies a longer transfer time. Hurteau and Desantis (1983) developed a linear feedback controller using full-state feedback. The controller gains are tuned according to the cable length. However, if the cable length changes in an unqualified way, degradation of the system performance occurs. In addition, the tuning algorithm was not tested experimentally.

As mentioned before, the objective of the crane control is to move the load from point to point and at the same time minimize the load swing. Usually, the controller is designed to achieve these two tasks simultaneously, as in the aforementioned controllers. However, in another approach used extensively, the two tasks are treated separately by designing two feedback controllers. The first task is an anti-swing controller. It controls the swing damping by a proper feedback of the swing angle and its rate. The second task is a tracking controller designed to make the trolley follow a reference trajectory. The trolley position and velocity are used for tracking feedback. The position trajectory is generally based on the classical velocity pattern, which is obtained from open-loop optimal control or input shaping techniques. The tracking controller can be either a classical Proportional-Derivative (PD) controller (Henry, 1999; Masoud 2000) or a Fuzzy Logic Controller (FLC) (Yang et al., 1996; Nalley and Trabia, 1994; Lee et al., 1997; Itho et al., 1994; Al-Moussa, 2000). Similarly, the anti-swing controller is designed by different methods. Henry (1999) and Masoud (2002) used delayed-position feedback, whereas Nalley and Trabia (1994), Yang et al. (1996), and Al-Moussa (2000) used FLC. Separation of the control tasks, anti-swing and tracking, enables the designer to handle different trajectories according to the work environment. Generally, the cable length is considered in the design of the anti-swing controller. However, the effect of the load mass is neglected in the design of the tracking controller. The system response is slow compared with that of optimal control or feedback control.

Raising the load (hoisting) during the transfer is needed only to avoid obstacles. This motion is slow, and hence variations in the cable length can be considered as a disturbance to the system. Then, the effect of variations in the cable length is investigated through simulation to make sure that the performance does not deteriorate. However, there are few studies that include hoisting in the design of controllers (e.g., Auernig and Troger, 1987).

The effect of the load weight on the dynamics is usually ignored. However, Lee (1998) and Omar and Nayfeh (2001) consider it in the design of controllers for gantry and tower cranes. From these studies, we find that, for very heavy loads compared to the trolley weight, the system performance deteriorates if the load weight is not included in the controller design.

1.3 Friction Compensation

Friction in mechanical systems has nonsymmetric characteristics. It depends on the direction of the motion as well as the position (Canudas, 1988). There are several methods to overcome friction effects. The first uses high-feedback-gain controllers, which may reduce the effect of the friction nonlinearities. However, this approach has severe limitations because the nonlinearities dominate any compensation for small errors. Limit cycles may appear as a consequence of the dynamic interaction between the friction forces and the controller, especially when the controller contains integral terms. The second uses high-frequency bias signal injection. Although it may alleviate friction effects, it may also excite high-frequency harmonics in the system. The third uses friction compensation, which aims to remove the effect of friction completely.

The third method has an advantage over the other methods because the system becomes linear after compensation. So, control algorithms based on the linear model can be applied directly. The compensation is done by estimating the friction of the system, and then applying an opposite control action to cancel it. The compensation can be done on-line to track the friction variations, which may occur due to changes in the environment and mechanical wear. Many researchers developed adaptive friction compensation for various applications using different adaptation techniques and models (Canudas et al., 1986; Li and Cheng, 1994). However, to obtain a good estimate of friction using the adaptive approach, one needs to persistently excite the system (Astrom and Wittenmark, 1994). In our system, the input signals do not have this characteristic. Moreover, friction can be assumed to be constant during the operation without affecting the system performance. This enables us to estimate the friction off-line using an appropriate persistent excitation.

The estimation process requires a model of friction. Friction models have been extensively discussed in the literature (Armstrong et al., 1994; Canudas, 1995). It is well established that friction is a function of the velocity; however, there is disagreement about the relationship between them. Among these models, we choose the one proposed by Canudas et al. (1986) because of its simplicity and because it represents most of the friction phenom- ena observed in our experiment, Figure 3.1. This model consists of constant viscous and Coulomb terms. These constants change with the motion direction.

1.4 Objectives

The main objectives of this thesis are-

- To design a wireless gantry crane which is safe and reliable to raise & lower and move the load horizontally easily.
- To construct a wireless gantry crane.
- To test the performance of a wireless gantry crane.

CHAPTER 2
METHODOLOGY

2.1 Introduction

In this chapter we'll discuss about the methods we'll use for this project. We'll discuss about the process of work and design concept of modern gantry crane. We'll also discuss how wireless communication is used to control the crane from remote area.

2.2 Block Diagram

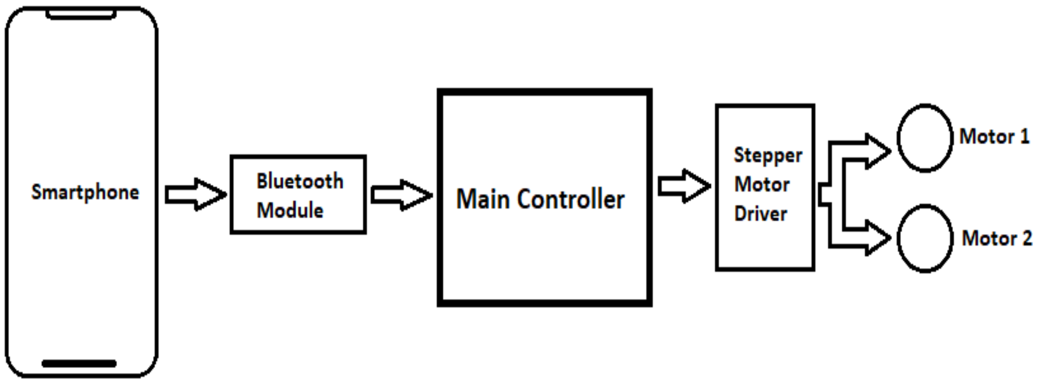

Figure 2.1: Block Diagram

This is our simple block diagram for designing a wireless gantry crane. We'll use Bluetooth communication protocol for wireless communication. With the simple design, we can control two stepper motors for moving and lifting any load. The motor 1 will be used for moving the load within the X axis and the motor 2 will be used for lifting the load within Y axis.

We'll be using an app to control the crane through the Bluetooth module. We'll send control command from the smartphone via Bluetooth, and the Bluetooth module on the system will receive the command and execute the specific program as per the command it received. For example, if we send a command character *"L"* as represented as left, the controller will move the crane to the left by controlling the stepper motor 1 as its use for moving left and right. Similarly for other commands, the system will execute the function for each individual command.

2.3 Circuit Diagram

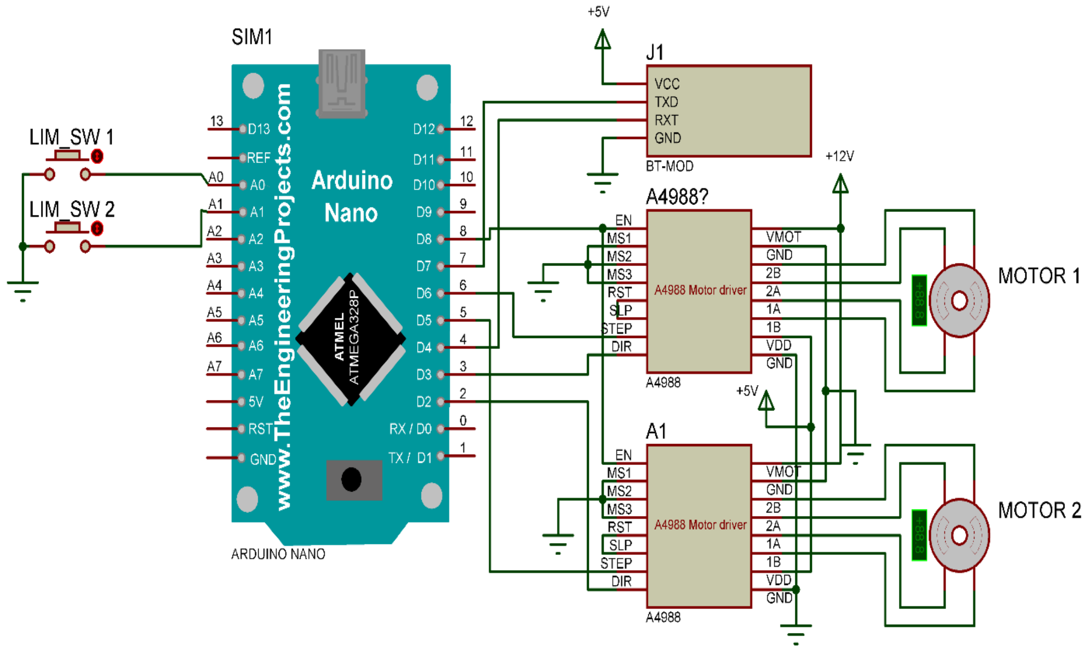

Figure 2.2: Circuit diagram of Wireless Control Crane system

In this circuit, we've used Arduio NANO as the main controller of the system. A Bluetooth module to receive command from smartphone. Two stepper motor driver to drive stepper motor. And two limit switch to stop motors at the end points.

The Bluetooth module is connected to the pin 7 and 4 of the controller. The transmitter pin of the Bluetooth module is connected to the pin 7 of the controller and the receiver pin of the Bluetooth module is connected to the pin 4 of the controller. In this way of connection, the module communicates with the controller.

The limit switch 1 is connected to the pin A0 of the controller and the limit switch 2 is connected to the pin A1 of the controller. Although A0 and A1 are the analog pin, but it can also act as digital IO pin. In this system we've used these pin as digital input.

The motor drivers are connected to the controller as simple as it shows. We've used only two pins to control each motor. For the motor 1, we've connected the DIR (Direction) pin to the pin 2 of the controller. And STEP (Step pin) pin is connected to the pin 5 of the controller. For the motor 2, the DIR pin is connected to the pin 3 of the controller and STEP pin is connected to the pin 6 of the controller. And the motors are connected to each motor driver.

2.4 Working procedure

In this system, when the power is connected, the controller initializes all set up. With the android app we can send command to the controller via Bluetooth module. We’ve set some command characters for controlling the motors. The characters are, “l” for moving the slider motor left, “r” for moving the slider motor right, “u” for lifting the loads up and “d” for lifting the loads down. The limit switches are used to prevent collision at each end point.

After starting the app, we need to connect to the Bluetooth module of the system. To connect with the Bluetooth module, we first need to open the app called “Bluetooth SPP Pro”. In this app, on startup we’ll see the Bluetooth module names to connect. In our case, we’ve used HC-06 model. So we’ll select HC-06 from the list and connect to the module. For the first time connection, we may need to enter a password. The default password is “1234” or “0000”. After connecting with the module we need to select the keyboard mode to enter to the control panel. Here we’ll need to set the buttons to send the commands. We’ll set the buttons in this manner; we’ll put the character “l” to the left button, “r” to the right button, “u” to the up button and lastly “d” to the down button. After setting up the buttons, we’re now set to send commands to the controller.

To move the slider motor right, we’ll press the “Right” button, thus it will send the character “r” to the controller and the controller will move the slider motor to the right. To move the slider motor left, we’ll press the “Left” button, thus it will send the character “l” to the controller. In the similar way the up down commands will work.

CHAPTER 3
HARDWARE AND COMPONENT DESCRIPTION

3.1 Introduction

In this chapter we'll discuss about the elements we've used for our project. We'll briefly discuss the working principle of each component. The pin configuration and connection method of each component. We'll also discuss about the characteristics of each component. The electrical and mechanical parameters of all components.

3.2 Arduino NANO

Arduino is an open source computer hardware and software company, project, and user community that designs and manufactures single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. The project's products are distributed as open-source hardware and software, which are licensed under the GNU Lesser General Public License (LGPL) or the GNU General Public License (GPL), permitting the manufacture of Arduino boards and software distribution by anyone. Arduino boards are available commercially in preassembled form, or as do-it-yourself (DIY) kits.

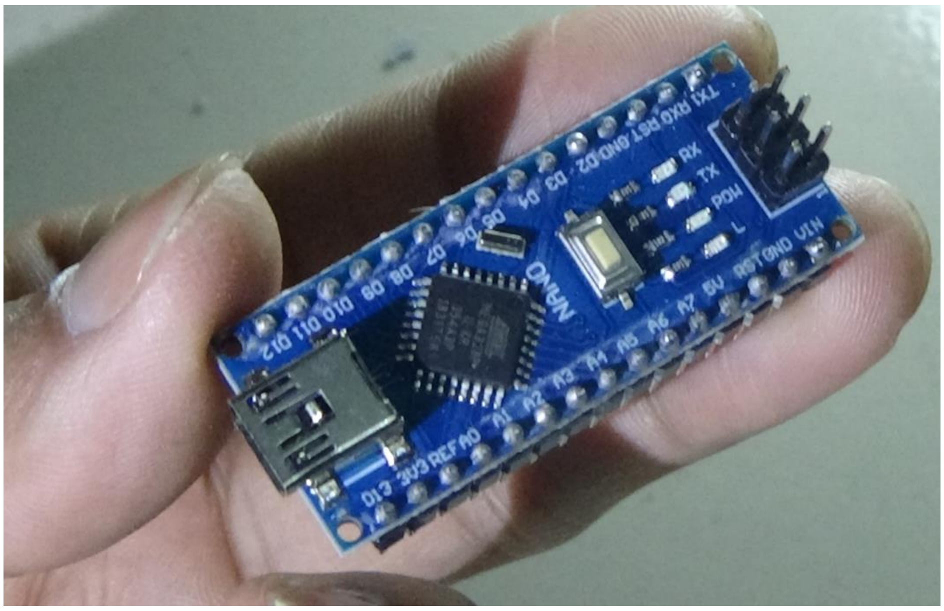

Figure 3.1: Arduino NANO

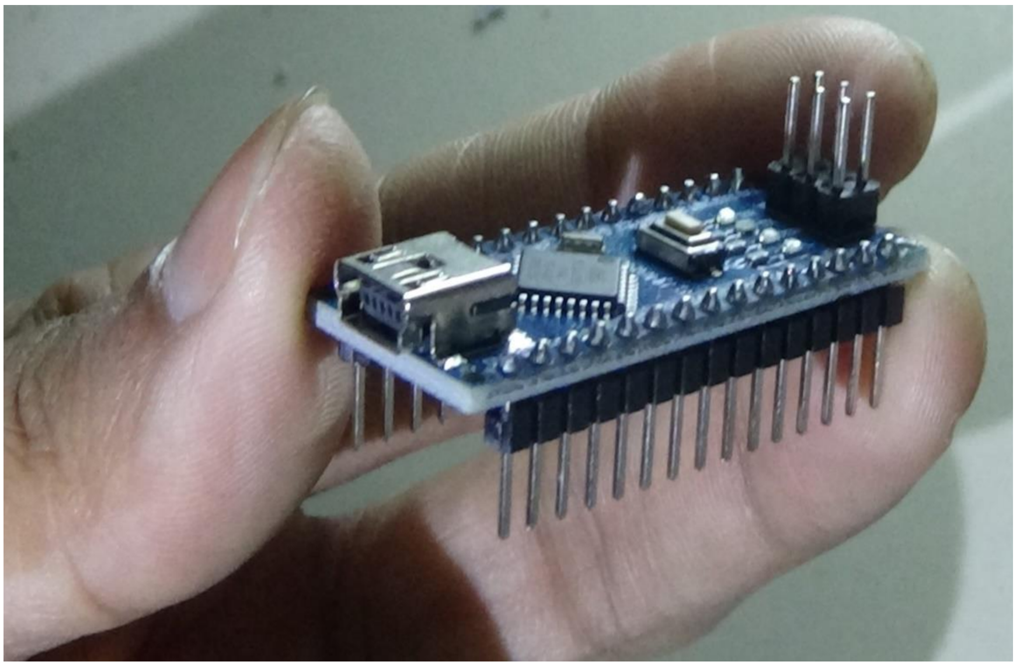

Figure 3.2: Arduino NANO pins

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

The Arduino project started in 2003 as a program for students at the Interaction Design Institute Ivrea in Ivrea, Italy, aiming to provide a low-cost and easy way for novices and professionals to create devices that interact with their environment using sensors and actuators. Common examples of such devices intended for beginner hobbyists include simple robots, thermostats, and motion detectors.

The name Arduino comes from a bar in Ivrea, Italy, where some of the founders of the project used to meet. The bar was named after Arduin of Ivrea, who was the margrave of the March of Ivrea and King of Italy from 1002 to 1014

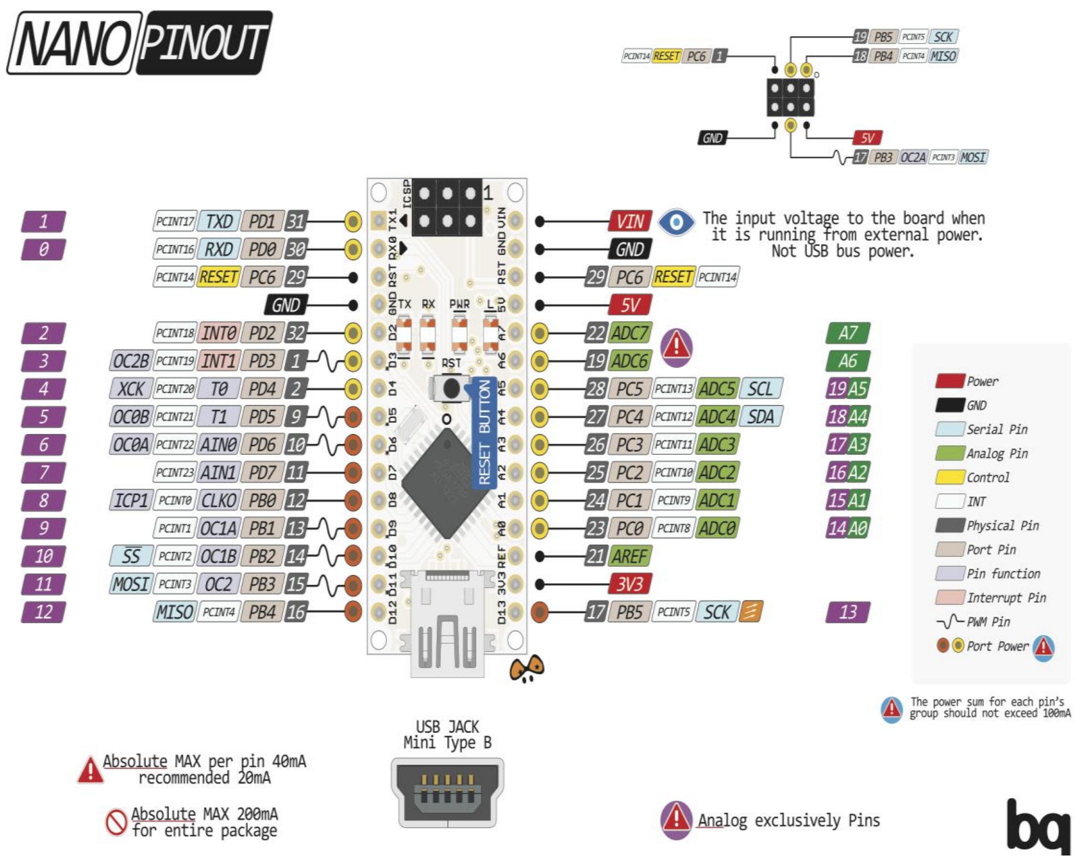
NANO PINOUT
The input voltage to the board when it is running from external power. Not USB bus power.
Absolute MAX per pin 40mA recommended 20mA
Absolute MAX 200mA for entire package
USB JACK Mini Type B
Analog exclusively Pins
The power sum for each pin's group should not exceed 100mA
Power
GND
Serial Pin
Analog Pin
Control
INT
Physical Pin
Port Pin
Pin function
Interrupt Pin
PWM Pin
Port Power
RESET BUTTON
ICSP
bq

Figure 3.3: Arduino NANO pin out

3.2.1 Microcontroller: ATMEGA328P

The Atmel® picoPower® ATmega328/P is a low-power CMOS 8-bit microcontroller based on the AVR® enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimize the device for power consumption versus processing speed.

The Atmel AVR® core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in a single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers. The ATmega328/P provides the following features: 32Kbytes of In-System Programmable Flash with Read-While-Write capabilities, 1Kbytes EEPROM, 2Kbytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, 1 serial programmable USARTs , 1 byte-oriented 2-wire Serial Interface (I2C), a 6- channel 10-bit ADC (8 channels in TQFP and QFN/MLF packages) , a programmable

Watchdog Timer with internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main oscillator and the asynchronous timer continue to run. Atmel offers the QTouch® library for embedding capacitive touch buttons, sliders and wheels functionality into AVR microcontrollers. The patented charge-transfer signal acquisition offers robust sensing and includes fully debounced reporting of touch keys and includes Adjacent Key Suppression® (AKS™) technology for unambiguous detection of key events. The easy-to-use QTouch Suite toolchain allows you to explore, develop and debug your own touch applications. The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega328/P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications. The ATmega328/P is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

3.2.2 Features

High Performance, Low Power Atmel®AVR® 8-Bit Microcontroller Family

• Advanced RISC Architecture

– 131 Powerful Instructions

– Most Single Clock Cycle Execution

– 32 x 8 General Purpose Working Registers

– Fully Static Operation

– Up to 20 MIPS Throughput at 20MHz

– On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory Segments

– 32KBytes of In-System Self-Programmable Flash program Memory

– 1KBytes EEPROM

– 2KBytes Internal SRAM

– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM

– Data Retention: 20 years at 85°C/100 years at 25°C(1)

– Optional Boot Code Section with Independent Lock Bits

• In-System Programming by On-chip Boot Program

• True Read-While-Write Operation

– Programming Lock for Software Security

• Peripheral Features

– Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode

– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode

– Real Time Counter with Separate Oscillator

– Six PWM Channels

– 8-channel 10-bit ADC in TQFP and QFN/MLF package

• Temperature Measurement

– 6-channel 10-bit ADC in PDIP Package

• Temperature Measurement

– Two Master/Slave SPI Serial Interface

– One Programmable Serial USART

– One Byte-oriented 2-wire Serial Interface (Philips I2C compatible)

– Programmable Watchdog Timer with Separate On-chip Oscillator

– One On-chip Analog Comparator

– Interrupt and Wake-up on Pin Change

• Special Microcontroller Features

– Power-on Reset and Programmable Brown-out Detection

– Internal Calibrated Oscillator

– External and Internal Interrupt Sources

– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby, and Extended Standby • I/O and Packages

– 23 Programmable I/O Lines – 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

• Operating Voltage:

– 1.8 - 5.5V

• Temperature Range:

– -40°C to 105°C

• Speed Grade:

– 0 - 4MHz @ 1.8 - 5.5V

– 0 - 10MHz @ 2.7 - 5.5V

– 0 - 20MHz @ 4.5 - 5.5V

• Power Consumption at 1MHz, 1.8V, 25°C

– Active Mode: 0.2mA

– Power-down Mode: 0.1µA

– Power-save Mode: 0.75µA (Including 32kHz RTC)

3.2.3 Architecture & Pin out

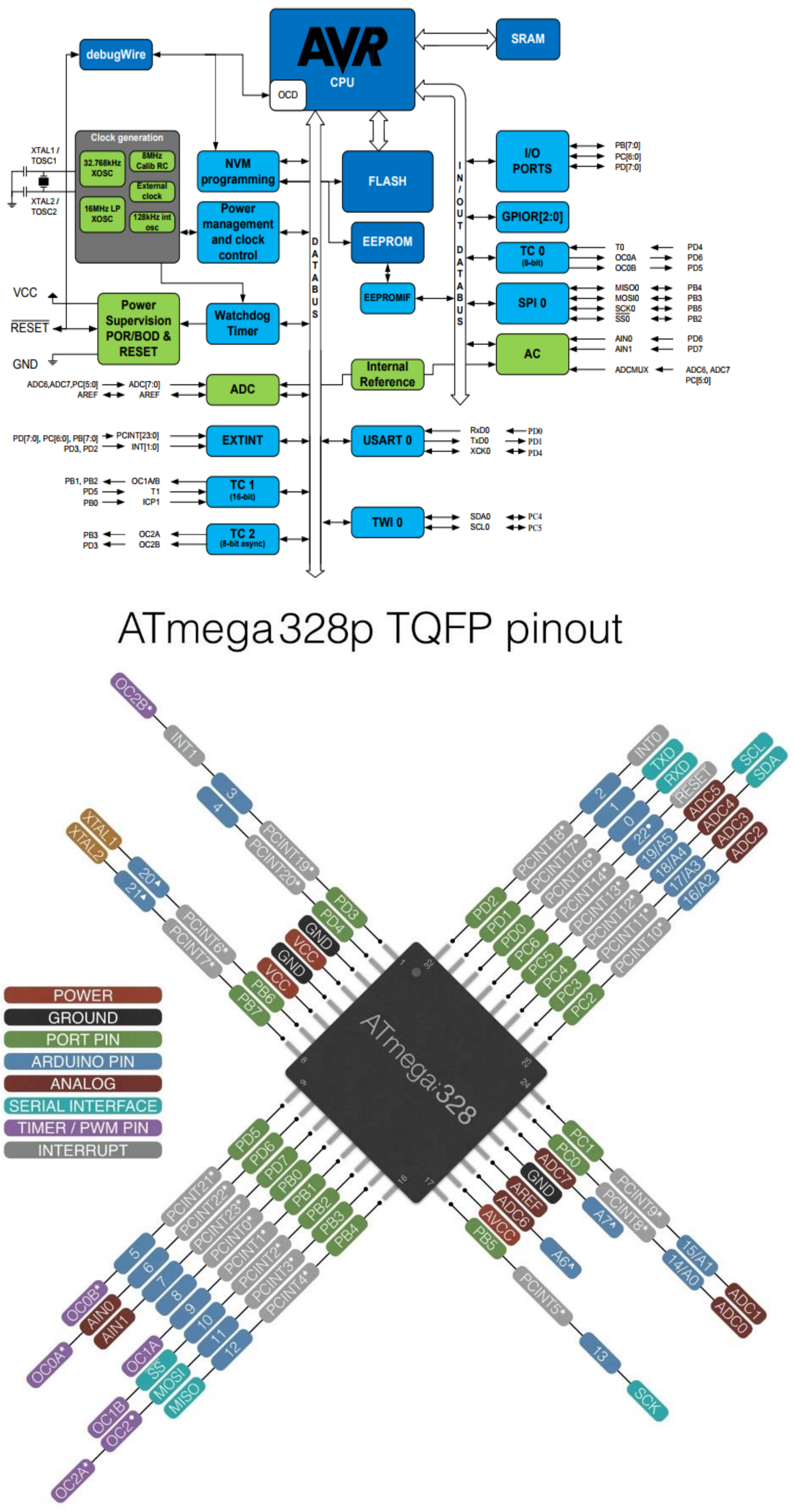

Figure 3.4: ATMEGA328P pin out

3.3 Stepper Motor

A stepper motor, also known as step motor or stepping motor, is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps without any position sensor for feedback (an open-loop controller), as long as the motor is carefully sized to the application in respect to torque and speed.

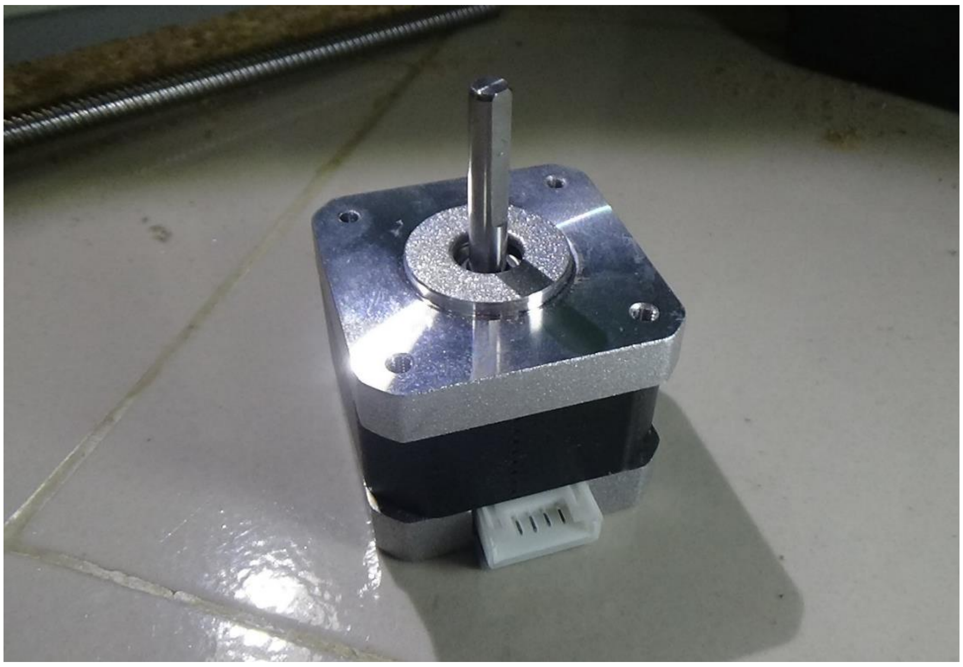

Figure 3.5: Stepper motor

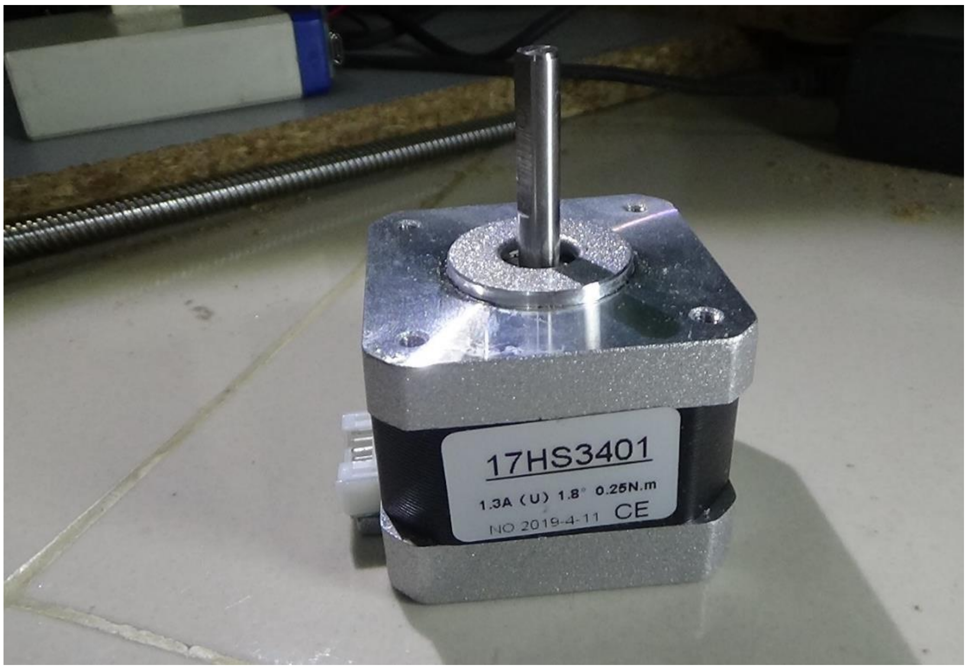

Figure 3.6: Stepper motor model

Switched reluctance motors are very large stepping motors with a reduced pole count, and generally are closed-loop commutated.

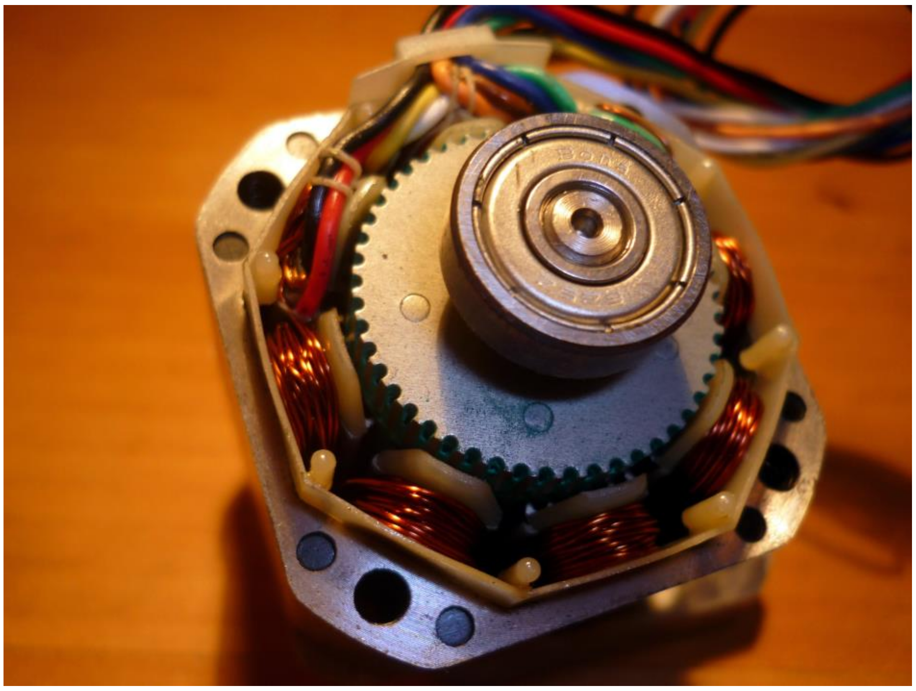

Figure 3.7: Stepper Motor inside

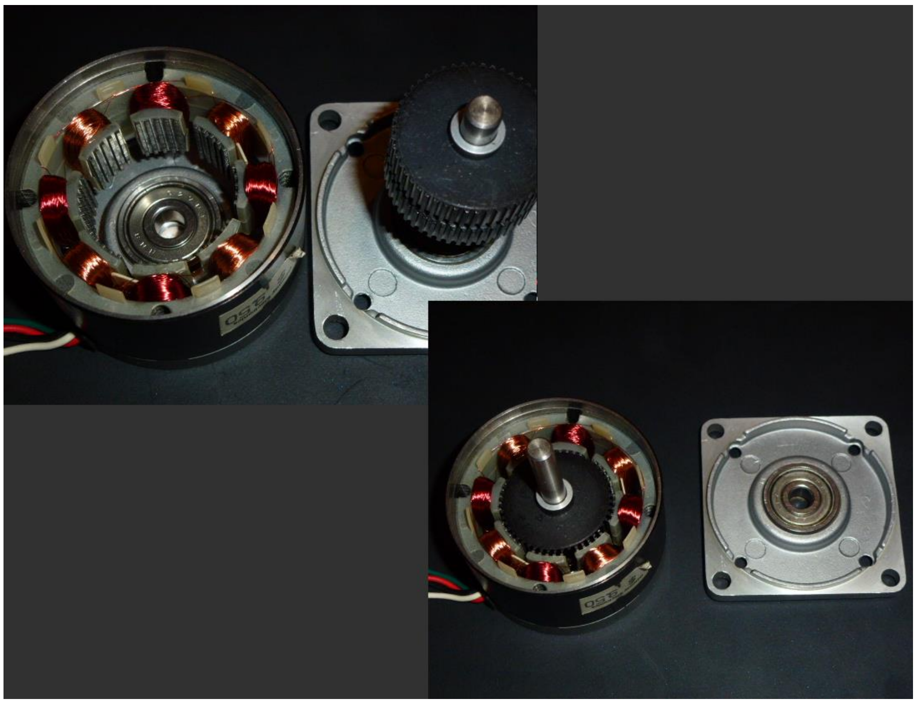

Figure 3.8: Bipolar Stepper Motor

3.3.1 Fundamentals of operation

Brushed DC motors rotate continuously when DC voltage is applied to their terminals. The stepper motor is known by its property of converting a train of input pulses (typically square wave pulses) into a precisely defined increment in the shaft position. Each pulse moves the shaft through a fixed angle.

Stepper motors effectively have multiple "toothed" electromagnets arranged around a central gear-shaped piece of iron. The electromagnets are energized by an external driver circuit or a micro controller. To make the motor shaft turn, first, one electromagnet is given power, which magnetically attracts the gear's teeth. When the gear's teeth are aligned to the first electromagnet, they are slightly offset from the next electromagnet. This means that when the next electromagnet is turned on and the first is turned off, the gear rotates slightly to align with the next one. From there the process is repeated. Each of those rotations is called a "step", with an integer number of steps making a full rotation. In that way, the motor can be turned by a precise angle.

The circular arrangement of electromagnets is divided into groups, each group called a phase, and there is an equal number of electromagnets per group. The number of groups is chosen by the designer of the stepper motor. The electromagnets of each group are interleaved with the electromagnets of other groups to form a uniform pattern of arrangement. For example, if the stepper motor has two groups identified as A or B, and ten electromagnets in total, then the grouping pattern would be ABABABABAB.

Electromagnets within the same group are all energized together. Because of this, stepper motors with more phases typically have more wires (or leads) to control the motor.

3.3.2 Types

There are three main types of stepper motors: [8]

1. Permanent magnet stepper
2. Variable reluctance stepper
3. Hybrid synchronous stepper

Permanent magnet motors use a permanent magnet (PM) in the rotor and operate on the attraction or repulsion between the rotor PM and the stator electromagnets.

Pulses move the rotor in discrete steps, CW or CCW. If left powered at a final step a strong detent remains at that shaft location. This detent has a predictable spring rate and specified torque limit; slippage occurs if the limit is exceeded. If current is removed a lesser detent still remains, therefore holding shaft position against spring or other torque influences. Stepping can then be resumed while reliably being synchronized with control electronics.

Clarification- Stepper motor detent is analogous to brushless dc motor cogging. In the case of constant speed brushless dc motor, magnetic cogging is of concern for torque ripple effects

Variable reluctance (VR) motors have a plain iron rotor and operate based on the principle that minimum reluctance occurs with minimum gap, hence the rotor points are attracted toward the stator magnet poles. Whereas hybrid synchronous are a combination of the permanent magnet and variable reluctance types, to maximize power in a small size[9].

VR motors do not have power off detents.

3.3.3 Two-phase stepper motors

There are two basic winding arrangements for the electromagnetic coils in a two phase stepper motor: bipolar and unipolar.

3.3.3.1 Unipolar motors

A unipolar stepper motor has one winding with center tap per phase. Each section of windings is switched on for each direction of magnetic field. Since in this arrangement a magnetic pole can be reversed without switching the direction of current, the commutation circuit can be made very simple (e.g., a single transistor) for each winding. Typically, given a phase, the center tap of each winding is made common: giving three leads per phase and six leads for a typical two phase motor. Often, these two phase commons are internally joined, so the motor has only five leads.

A microcontroller or stepper motor controller can be used to activate the drive transistors in the right order, and this ease of operation makes unipolar motors popular with hobbyists; they are probably the cheapest way to get precise angular movements. For the experimenter, the windings can be identified by touching the terminal wires together in PM motors. If the terminals of a coil are connected, the shaft becomes harder to turn. One way to distinguish the center tap (common wire) from a coil-end wire is by measuring the resistance. Resistance between common wire and coil-end wire is always half of the resistance between coil-end wires. This is because there is twice the length of coil between the ends and only half from center (common wire) to the end. A quick way to determine if the stepper motor is working is to short circuit every two pairs and try turning the shaft. Whenever a higher than normal resistance is felt, it indicates that the circuit to the particular winding is closed and that the phase is working.

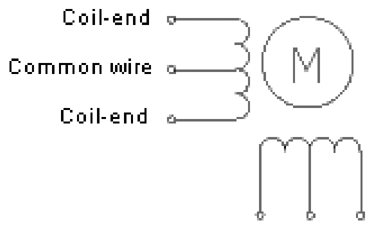

Figure 3.9: Unipolar stepper motor coils

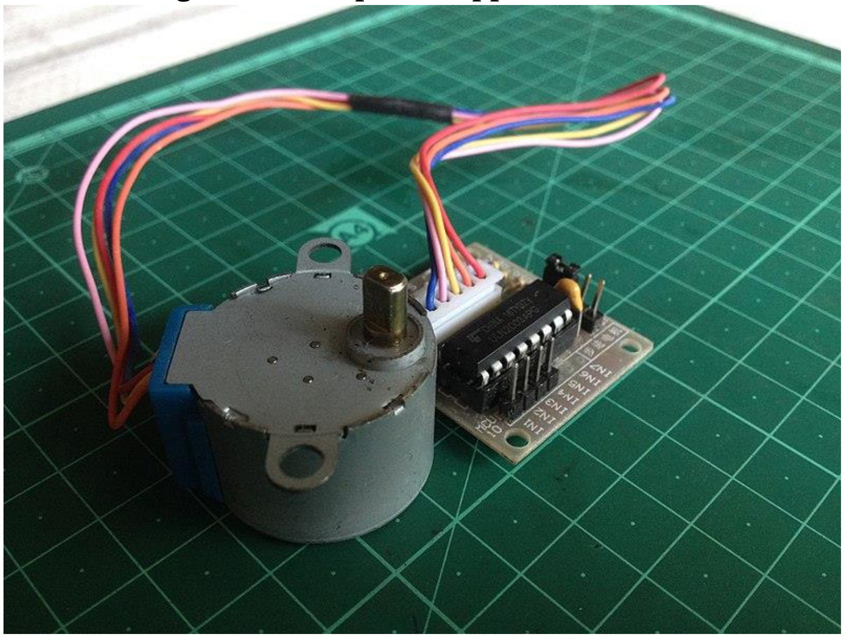

Figure 3.10: Unipolar Motor

3.3.3.2 Bipolar motor

Bipolar motors have a single winding per phase. The current in a winding needs to be reversed in order to reverse a magnetic pole, so the driving circuit must be more complicated, typically with an H-bridge arrangement (however there are several off-the-shelf driver chips available to make this a simple affair). There are two leads per phase, none are common.

A typical driving pattern for a two coil bipolar stepper motor would be: A+ B+ A− B−. I.e. drive coil A with positive current, then remove current from coil A; then drive coil B with positive current, then remove current from coil B; then drive coil A with negative current (flipping polarity by switching the wires e.g. with an H bridge), then remove current from coil A; then drive coil B with negative current (again flipping polarity same as coil A); the cycle is complete and begins anew.

Static friction effects using an H-bridge have been observed with certain drive topologies.[10]

Dithering the stepper signal at a higher frequency than the motor can respond to will reduce this "static friction" effect.

Because windings are better utilized, they are more powerful than a unipolar motor of the same weight. This is due to the physical space occupied by the windings. A unipolar motor has twice the amount of wire in the same space, but only half used at any point in time, hence is 50% efficient (or approximately 70% of the torque output available). Though a bipolar stepper motor is more complicated to drive, the abundance of driver chips means this is much less difficult to achieve.

An 8-lead stepper is like a unipolar stepper, but the leads are not joined to common internally to the motor. This kind of motor can be wired in several configurations:

- Unipolar.
- Bipolar with series windings. This gives higher inductance but lower current per winding.
- Bipolar with parallel windings. This requires higher current but can perform better as the winding inductance is reduced.
- Bipolar with a single winding per phase. This method will run the motor on only half the available windings, which will reduce the available low speed torque but require less current

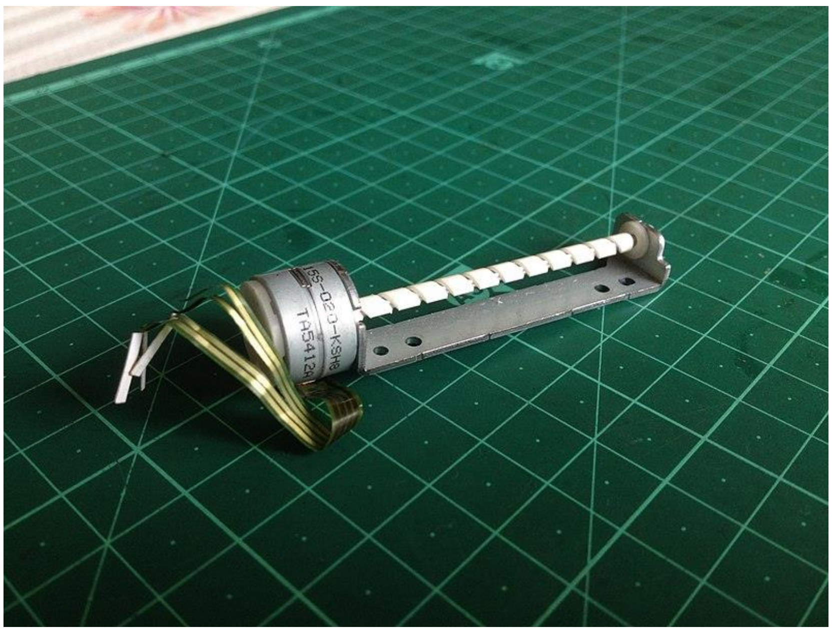

Figure 3.11: Bipolar Stepper Motor

3.4 Stepper Motor Controller

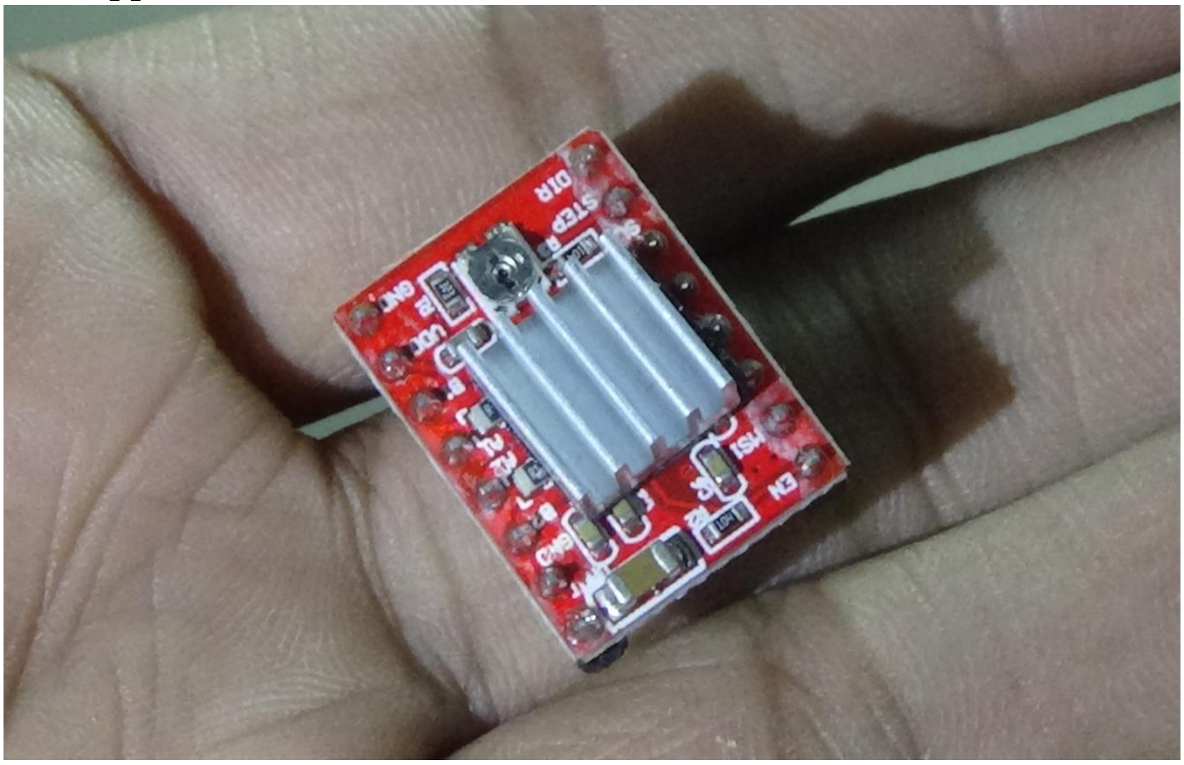

Figure 3.12: Stepper motor driver top view

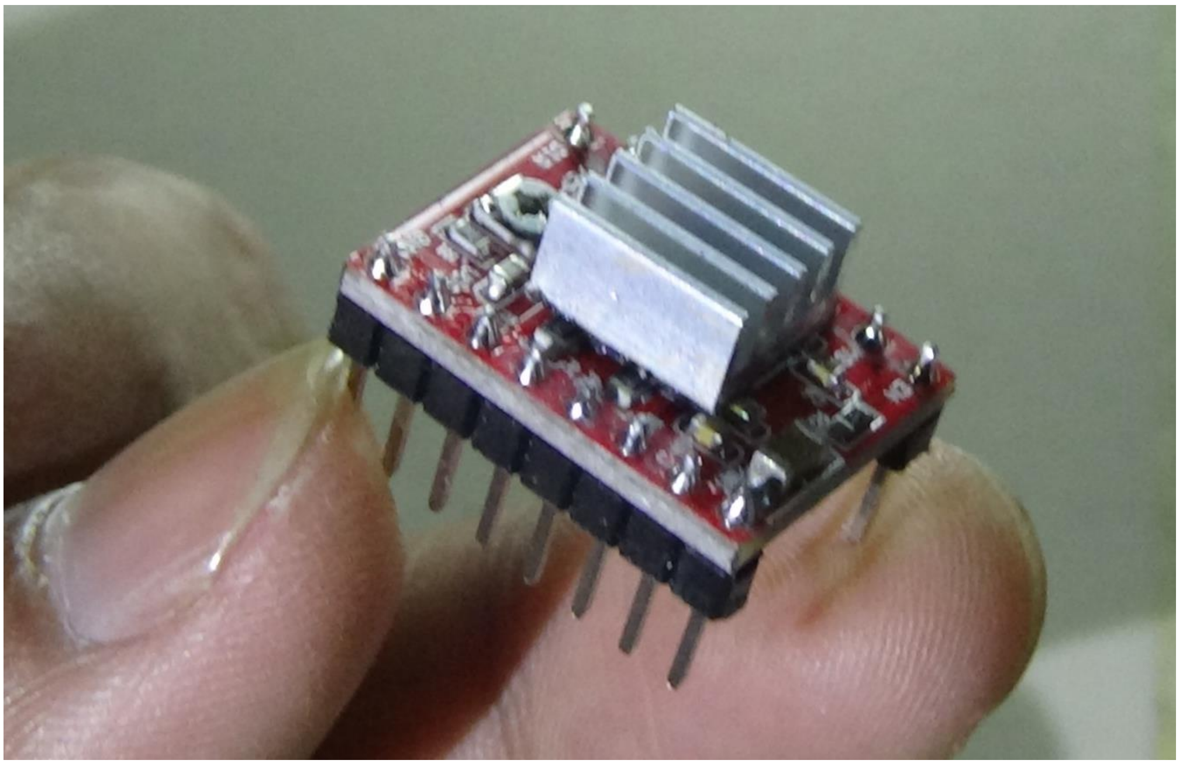

Figure 3.13: Stepper motor driver pin side

At the heart of the module is a Microstepping Driver from Allegro – A4988. It’s small in stature (only 0.8″ × 0.6″) but still packs a punch.

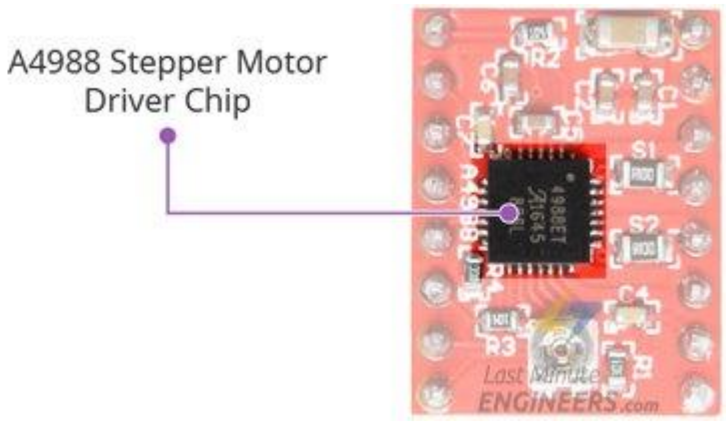

Figure 3.14: A4988 Stepper Motor Driver Chip

The A4988 stepper motor driver has output drive capacity of up to 35 V and ±2A and lets you control one bipolar stepper motor at up to 2A output current per coil like NEMA 17.

The driver has built-in translator for easy operation. This reduces the number of control pins to just 2, one for controlling the steps and other for controlling spinning direction.

3.4.1 A4988 Motor Driver Pin out

The A4988 driver has total 16 pins that interface it to the outside world. The connections are as follows:

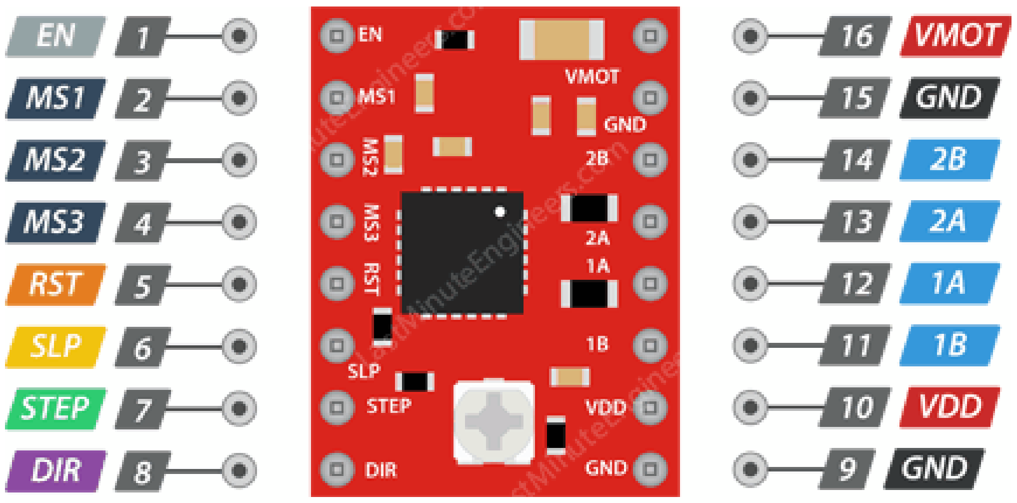

Figure 3.15: A4988 driver module pin out

Let’s familiarize ourselves with all the pins one by one.

3.4.2 Power Connection Pins

The A4988 actually requires two power supply connections.

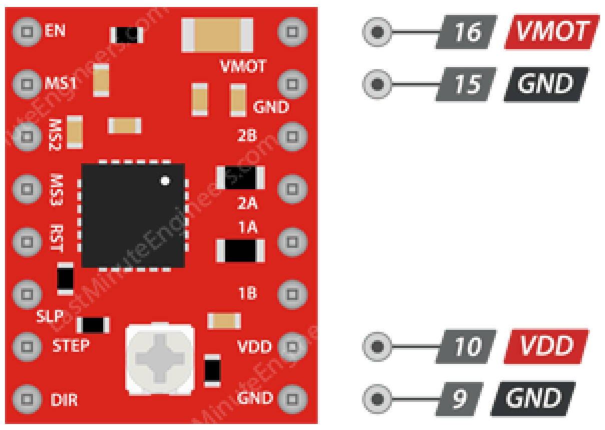

Figure 3.16: A4988 driver power pins

VDD & GND is used for driving the internal logic circuitry which can be 3V to 5.5 V.

Whereas,

VMOT & GND supplies power for the motor which can be 8V to 35 V.

According to datasheet, the motor supply requires appropriate decoupling capacitor close to the board, capable of sustaining 4A.

3.4.3 Micro step Selection Pins

The A4988 driver allows microstepping by allowing intermediate step locations. This is achieved by energizing the coils with intermediate current levels.

For example, if you choose to drive NEMA 17 having 1.8° or 200 steps per revolution in quarter-step mode, the motor will give 800 microsteps per revolution.

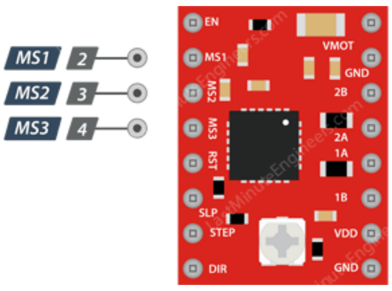

Figure 3.17: Micro step selection pins

The A4988 driver has three step size (resolution) selector inputs viz. MS1, MS2 & MS3. By setting appropriate logic levels to these pins we can set the motors to one of the five step resolutions.

MS1	MS2	MS3	Microstep Resolution
Low	Low	Low	Full step

High	Low	Low	Half step
Low	High	Low	Quarter step
High	High	Low	Eighth step
High	High	High	Sixteenth step

These three microstep selection pins are pulled LOW by internal pull-down resistors, so if we leave them disconnected, the motor will operate in full step mode.

3.4.4 Control Input Pins

The A4988 has two control inputs viz. STEP and DIR.

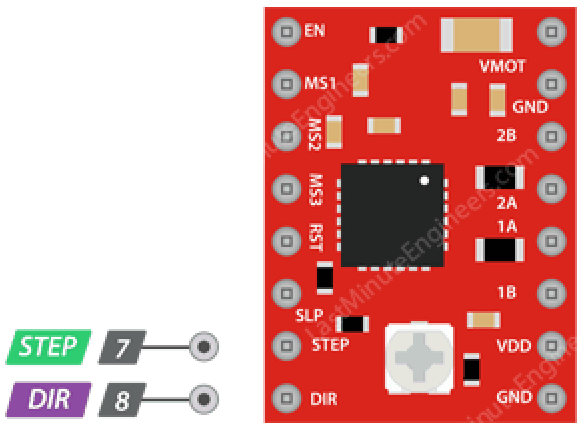

Figure 3.18: A4988 driver Control input pin

STEP input controls the mirosteps of the motor. Each HIGH pulse sent to this pin steps the motor by number of microsteps set by Microstep Selection Pins. The faster the pulses, the faster the motor will rotate.

DIR input controls the spinning direction of the motor. Pulling it HIGH drives the motor clockwise and pulling it LOW drives the motor counterclockwise.

If you just want the motor to rotate in a single direction, you can tie DIR directly to VCC or GND accordingly.

3.4.5 Pins For Controlling Power States

The A4988 has three different inputs for controlling its power states viz. EN, RST, and SLP.

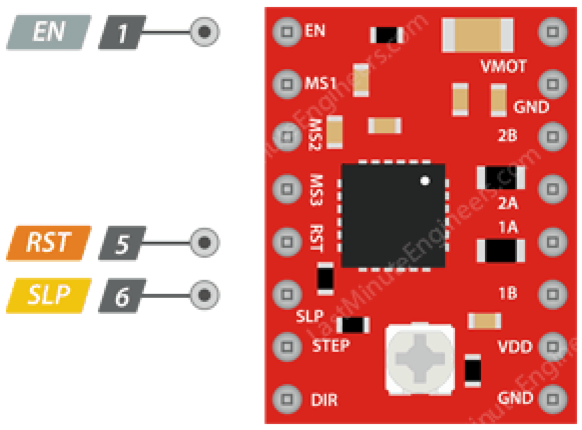

Figure 3.19: A4988 Power States Control pins

EN Pin is active low input, when pulled LOW(logic 0) the A4988 driver is enabled. By default this pin is pulled low so the driver is always enabled, unless you pull it HIGH.

SLP Pin is active low input. Meaning, pulling this pin LOW puts the driver in sleep mode, minimizing the power consumption. You can invoke this especially when the motor is not in use to conserve power.

RST is also an active low input. When pulled LOW, all STEP inputs are ignored, until you pull it HIGH. It also resets the driver by setting the internal translator to a predefined Home state. Home state is basically the initial position from where the motor starts and it's different depending upon the microstep resolution.

3.4.6 Output Pins

The A4988 motor driver's output channels are broken out to the edge of the module with 1B, 1A, 2A & 2B pins.

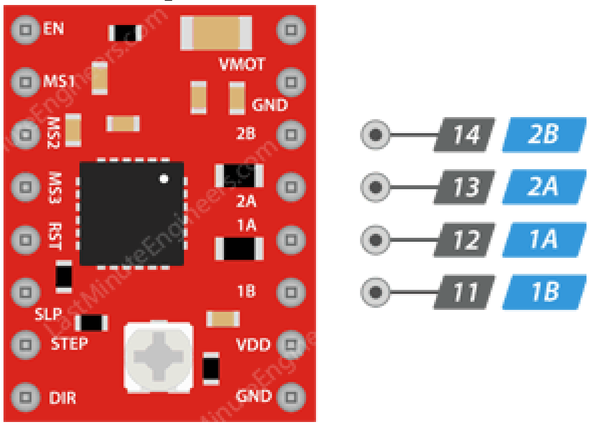

Figure 3.20: A4988 driver Output pins

You can connect any bipolar stepper motor having voltages between 8V to 35 V to these pins.

Each output pin on the module can deliver up to 2A to the motor. However, the amount of current supplied to the motor depends on system's power supply, cooling system & current limiting setting.

3.4.7 Cooling System – Heatsink

Excessive power dissipation of the A4988 driver IC results in the rise of temperature that can go beyond the capacity of IC, probably damaging itself. Even if the A4988 driver IC has a maximum current rating of 2A per coil, the chip can only supply approximately 1A per coil without getting overheated. For achieving more than 1A per coil, a heat sink or other cooling method is required.

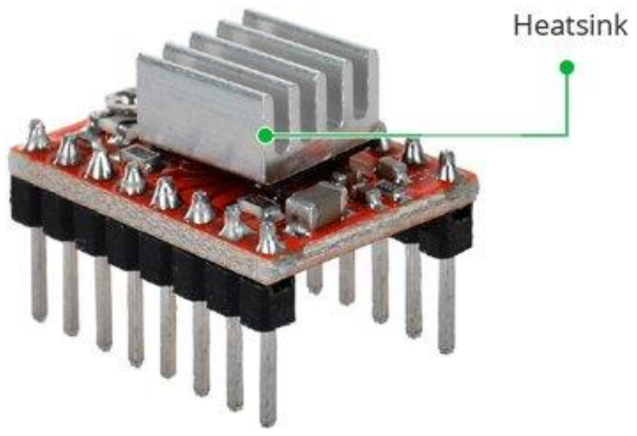

Figure 3.21: Heat sink for A4988 driver

The A4988 driver usually comes with a heatsink. It is advisable to install it before you use the driver.

3.4.8 Current limiting

Before using the motor, there's a small adjustment that we need to make. We need to limit the maximum amount of current flowing through the stepper coils and prevent it from exceeding the motor's rated current.

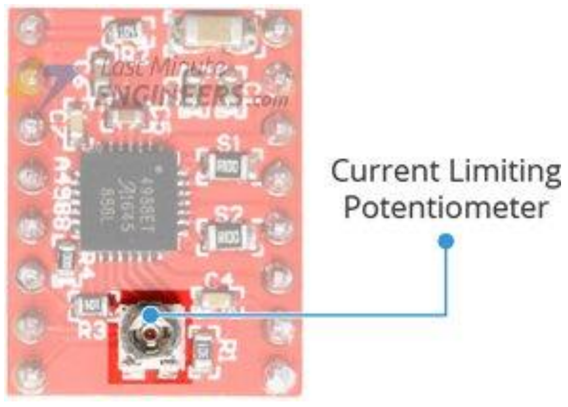

Figure 3.22: Current limiting potentiometer

There’s a small trimmer potentiometer on the A4988 driver that can be used to set the current limit. You should set the current limit to be at or lower than the current rating of the motor.

To make this adjustment there are two methods:

Method 1:

In this method we are going to set the current limit by measuring the voltage (Vref) on the “ref” pin.

1. Take a look at the datasheet for your stepper motor. Note down it’s rated current. In our case we are using NEMA 17 200steps/rev, 12V 350mA.
2. Put the driver into full-step mode by leaving the three microstep selection pins disconnected.
3. Hold the motor at a fixed position by not clocking the STEP input.
4. Measure the voltage (Vref) on the metal trimmer pot itself while you adjust it.
5. Adjust the Vref voltage using the formula

Current Limit = Vref x 2.5

For example, if your motor is rated for 350mA, you would adjust the reference voltage to 0.14V.

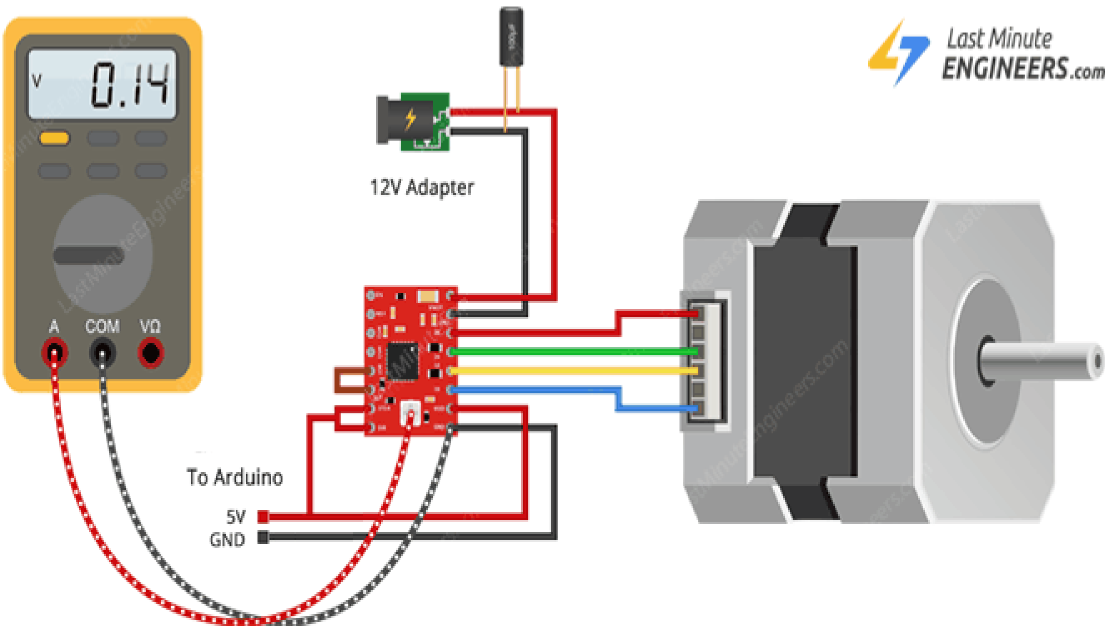

Figure 3.23: Current Adjustment method 1

Method 2:

In this method we are going to set the current limit by measuring the current running through the coil.

1. Take a look at the datasheet for your stepper motor. Note down it's rated current. In our case we are using NEMA 17 200steps/rev, 12V 350mA.

2. Put the driver into full-step mode by leaving the three microstep selection pins disconnected.

3. Hold the motor at a fixed position by not clocking the STEP input. Do not leave the STEP input floating, connect it to logic power supply(5V)

4. Place the ammeter in series with one of the coils on your stepper motor and measure the actual current flowing.

5. Take a small screwdriver and adjust the current limit potentiometer until you reach rated current.

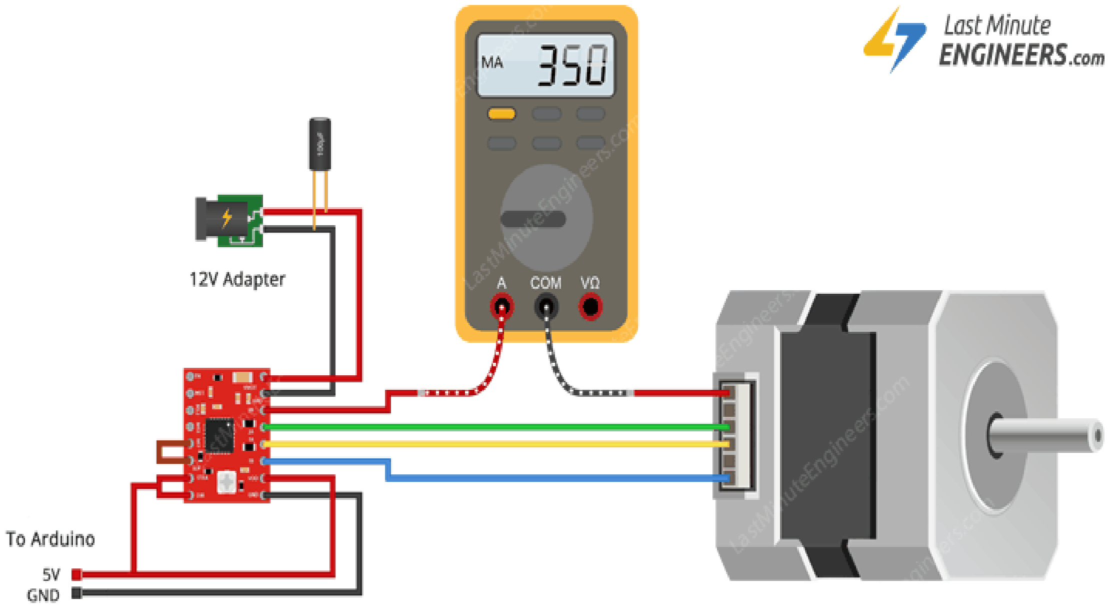

Figure 3.24: Current Adjustment method 2

3.4.9 Wiring A4988 stepper motor driver with Arduino UNO

Now that we know everything about the driver, we will connect it to our Arduino.

Connections are fairly simple. Start by connecting VDD and GND (next to VDD) to the 5V and ground pins on the Arduino. DIR and STEP input pins are connected to #2 & #3 digital output pins on Arduino respectively. Connect the stepper motor to the 2B, 2A, 1A & 1B pins. Actually A4988 is conveniently laid out to match the 4-pin connector on several bipolar motors so, that shouldn't be a problem.

Next, Connect RST pin to the adjacent SLP/SLEEP pin to keep the driver enabled. Also keep the microstep selection pins disconnected to operate the motor in full step mode.

Finally, connect the motor power supply to the VMOT and GND pins. Remember to put a large 100µF decoupling electrolytic capacitor across motor power supply pins, close to the board.

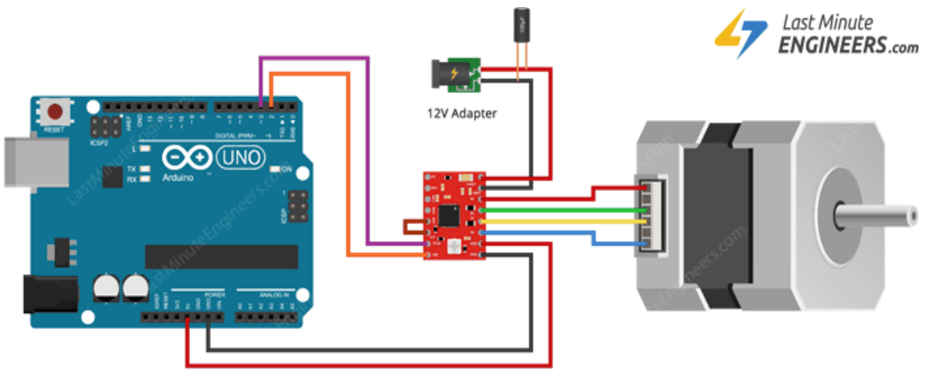

Figure 3.25: Wiring method of the driver

3.5 Bluetooth Module

HC-06 is a **Bluetooth module** designed for establishing short range wireless data communication between two microcontrollers or systems. The module works on **Bluetooth 2.0 communication protocol** and it can only act as a slave device. This is cheapest method for wireless data transmission and more flexible compared to other methods and it even can transmit files at speed up to 2.1Mb/s.

HC-06 uses frequency hopping spread spectrum technique (**FHSS**) to avoid interference with other devices and to have full duplex transmission. The device works on the frequency range from 2.402 GHz to 2.480GHz.

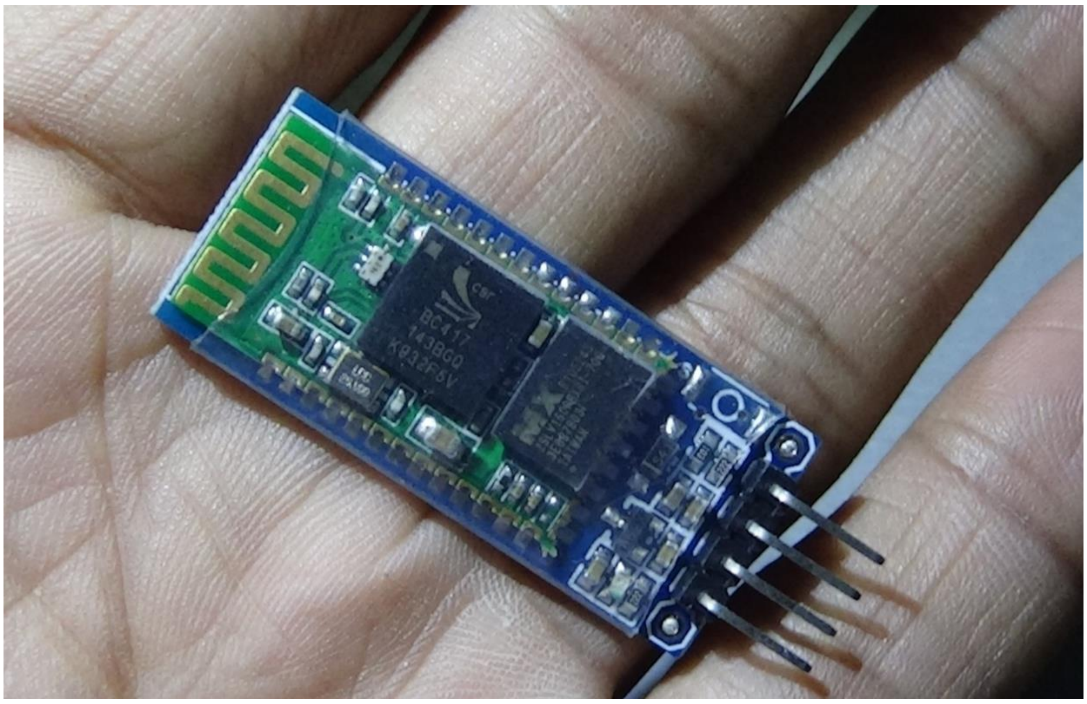

Figure 3.26: Bluetooth module top

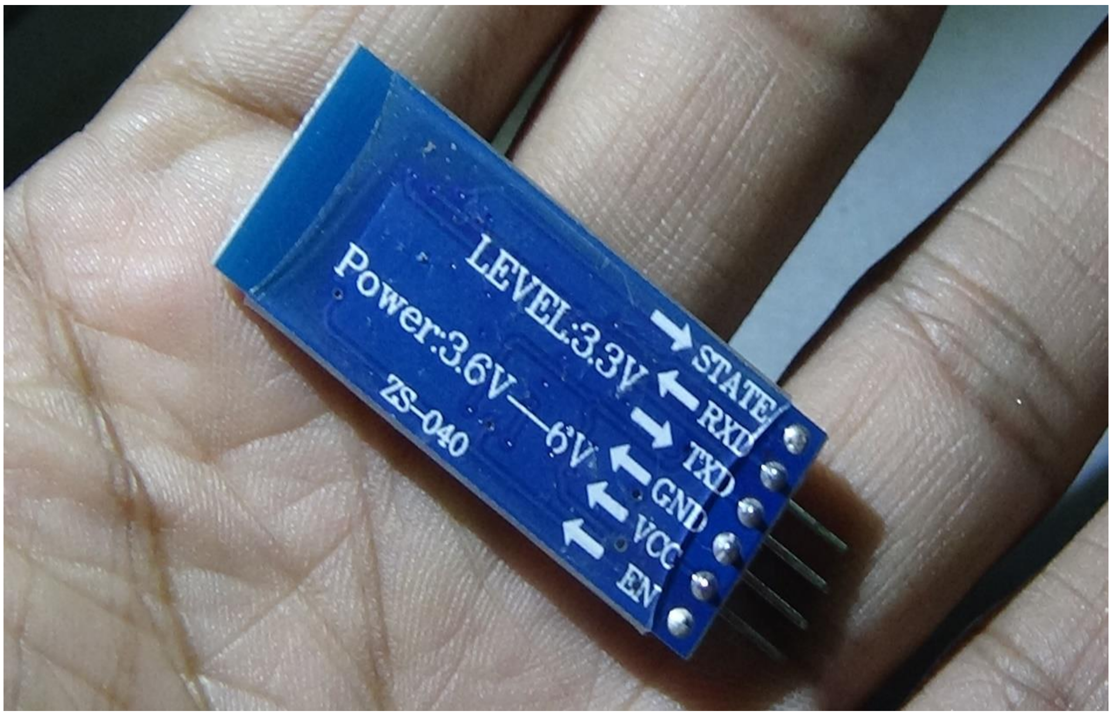

Figure 3.27: Bluetooth module back

3.5.1 Pin configuration

Pin	**Name**	**Function**
1	EN	The pin state determines whether the module works in AT command mode or normal mode [High=AT commands receiving mode(Commands response mode), Low or NC = Bluetooth module normally working]
2	Vcc	+5V Positive supply needs to be given to this pin for powering the module
3	Gnd	Connect to ground

4	TXD	Serial data is transmitted by module through this pin (at 9600bps by default), 3.3V logic
5	RXD	Serial data is received by module through this pin (at 9600bps by default),3.3V logic
6	State	The pin is connected to the LED on the board to represent the state of the module

HC-06 module has six pins as shown in the pinout. In them we only need to use four for successfully interfacing the module. Some breakout boards will only leave four output pins only because of this reason.

3.5.2 HC-06 Features and Electrical characteristics

- Bluetooth protocol: Bluetooth V2.0 protocol standard
- Power Level: Class2(+6dBm)
- Band: 2.40GHz—2.48GHz, ISM Band
- Receiver sensitivity: -85dBm
- USB protocol: USB v1.1/2.0
- Modulation mode: Gauss frequency Shift Keying
- Safety feature: Authentication and encryption
- Operating voltage range:+3.3V to +6V
- Operating temperature range: -20ºC to +55ºC
- Operating Current: 40mA

3.5.3 HC-06 Bluetooth Module Advantages

- HC-06 is best option when short distance wireless communication is needed. The module is used for wireless communications of less than 100 meters.
- The module is very easy to interface and to communicate.
- The module is one of the cheapest solutions for wireless communication of all types present in the market.
- The module consumes very less power to function and can be used on battery operated mobile systems.

- The module can be interfaced with almost all controllers or processors as it uses UART interface.

3.5.4 How to use HC-06 Bluetooth Module

The communication with this HC-06 module is done through **UART interface**. The data is sent to the module or received from the module though this interface. So we can connect the module to any microcontroller or directly to PC which has RS232 port (UART interface). A typical interface circuit of the module to an arduino is shown below.

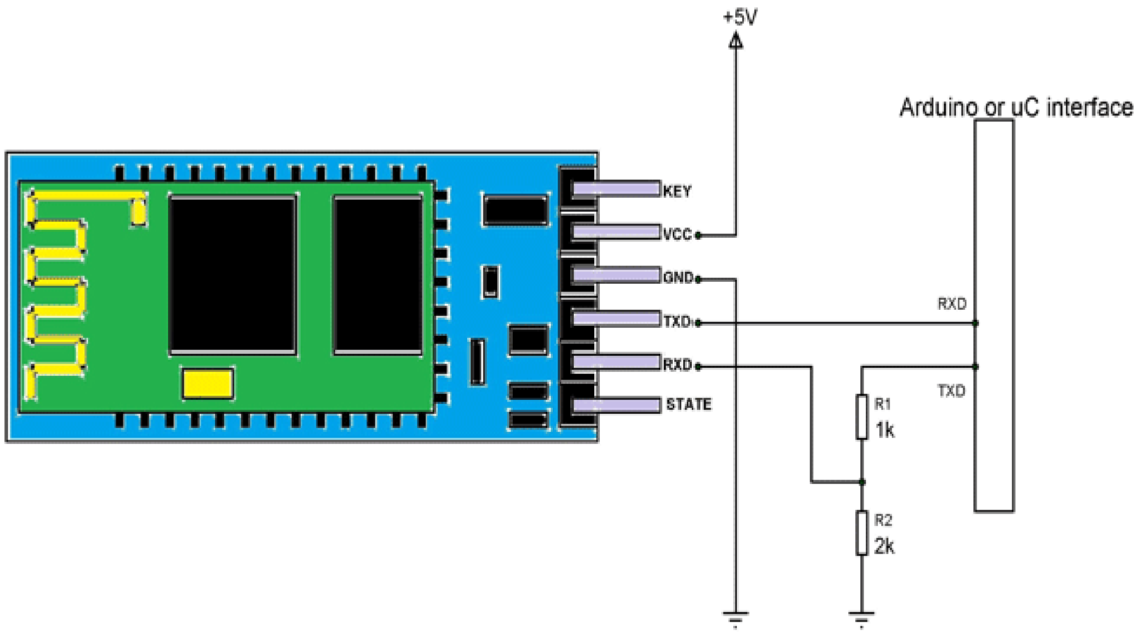

Figure 3.28: Connection diagram of Bluetooth module

Here the module is connected to +5V standard regulated power supply and UART interface is established as shown in figure. All you need to do is connect RXD of arduino to TXD of module and TXD of arduino is connected to RXD of module through a resistor voltage divider. This voltage divider is provided for converting 5V logic signal sent by arduino to +3.3V logic signals which are suitable for the module. The ground of arduino and module must be connected for voltage reference in case separate power sources are used.

After connecting the module you have to write the program in arduino IDE to receive and send data to the module. For successful wireless communication you need to remember a few things:

- In programming you need to set default baud rate of UART serial communication to 9600. The value is default setting of module and can be change in program.
- The module is a slave and so you need a master to establish a successful wireless interface. For that you need another [arduino + module (with master feature)] setup or you can use a smart phone as a master and search for HC-06 slave.
- The master searches for slave and connects to it after authenticated with password. The HC-06 module has default password '1234' which can be changed.
- In program you can receive data master sends (After authentication) and perform tasks based on it.
- Also you can download libraries for module through the websites and use them to make communication easy. All you need to do is download these libraries and call them in programs. Once the header file is included, you can use simple commands in the program to tell the arduino to send or receive data. The module sends this data to master through wireless Bluetooth. If the module receives any data from master, it will transmit it to arduino through UART serial communication.
- You can also interface HC-06 to PC using RS232 cable. Once you interface is done you can use serial terminal on PC or any similar software to send or receive data to module. You need to type in AT command in serial terminal to communicate with the module and these commands can be downloaded here.

3.5.5 Applications

- Hobby projects
- Engineering applications
- Robotics
- Mobile Phone Accessories
- Servers
- Computer Peripherals
- Sports and Leisure Equipment
- USB Dongles

3.6 Leads Screw

A leadscrew (or lead screw), also known as a power screw or translation screw, is a screw used as a linkage in a machine, to translate turning motion into linear motion. Because of the large area of sliding contact between their male and female members, screw threads have larger frictional energy losses compared to other linkages. They are not typically used to carry high power, but more for intermittent use in low power actuator and positioner mechanisms. Leadscrews are commonly used in linear actuators, machine slides (such as in machine tools), vises, presses, and jacks. Leadscrews are a key component in electric linear actuators.

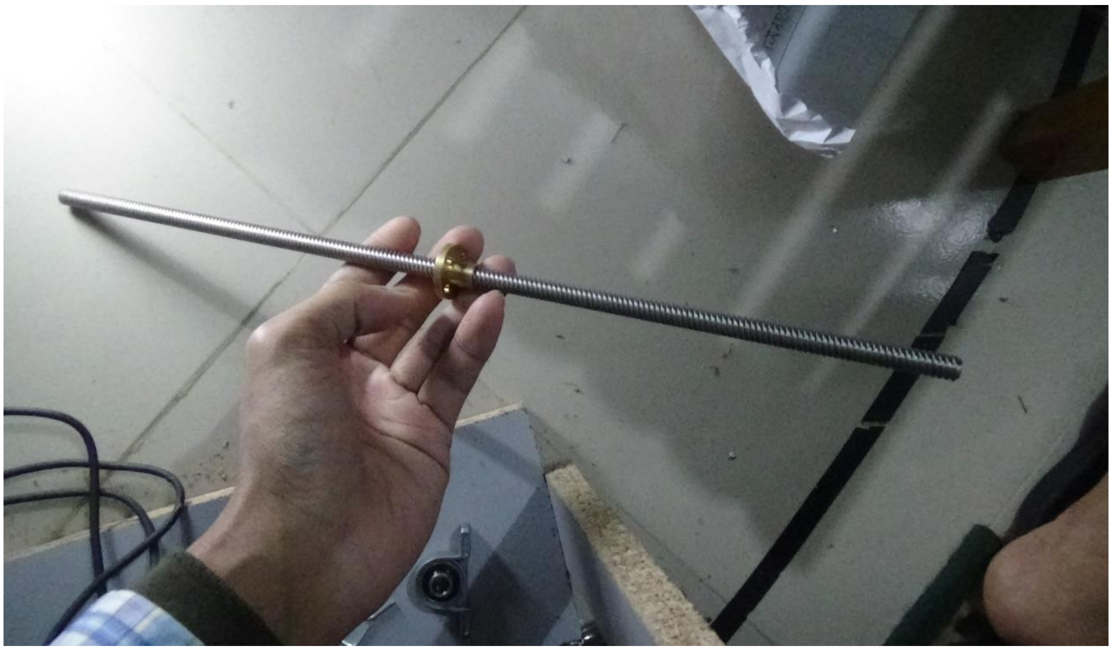

Figure 3.29: Leads Screw

Leadscrews are manufactured in the same way as other thread forms (they may be rolled, cut, or ground).

A lead screw is sometimes used with a split nut also called half nut which allows the nut to be disengaged from the threads and moved axially, independently of the screw's rotation, when needed (such as in single-point threading on a manual lathe).

3.7 Coupling

A coupling is a device used to connect two shafts together at their ends for the purpose of transmitting power. The primary purpose of couplings is to join two pieces of rotating equipment while permitting some degree of misalignment or end movement or both. In a more general context, a coupling can also be a mechanical device that serves to connect the ends of adjacent parts or objects. Couplings do not normally allow disconnection of shafts during operation, however there are torque limiting couplings which can slip or disconnect when some torque limit is exceeded. Selection, installation and maintenance of couplings can lead to reduced maintenance time and maintenance cost.

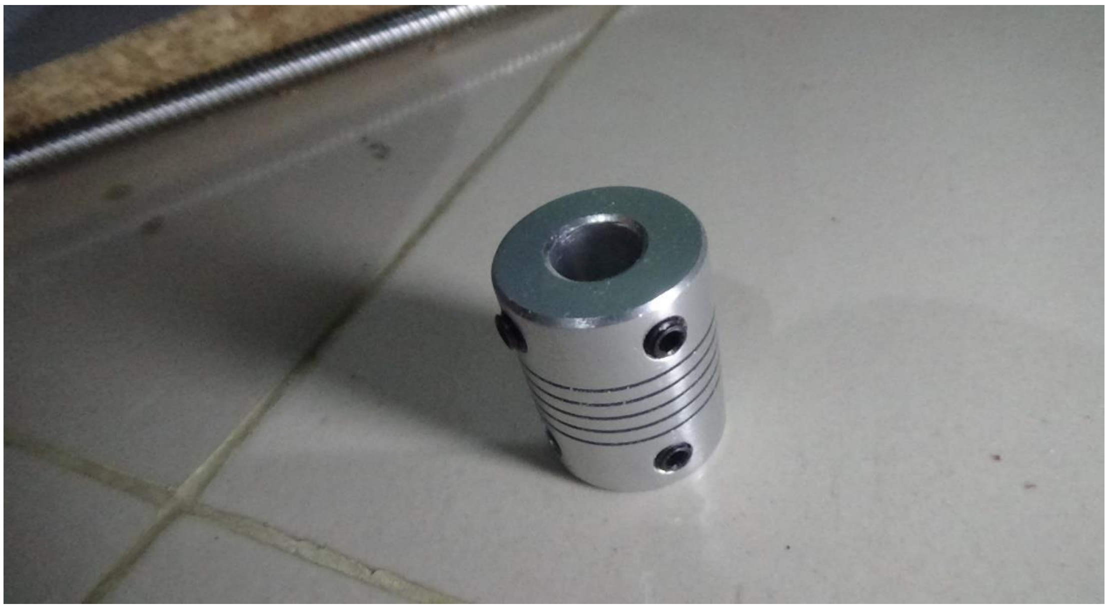

Figure 3.30: Coupler

3.7.1 Usage

Shaft couplings are used in machinery for several purposes. A primary function is to transfer power from one end to another end (ex: motor transfer power to pump through coupling).

Other common uses:

- To alter the vibration characteristics of rotating units
- To connect driving and the driven part
- To introduce protection against overloads
- To provide for the connection of shafts of units that are manufactured separately (such as a motor and generator) and to provide for disconnection for repairs or alterations
- To provide for misalignment of the shafts
- To introduce mechanical flexibility
- To reduce the transmission of shock loads from one shaft to another
- To slip when overload occurs

3.8 Pillow Block Mounted Bearing

A pillow block usually refers to housing with an included anti-friction bearing. A pillow block refers to any mounted bearing wherein the mounted shaft is in a parallel plane to the mounting surface, and perpendicular to the center line of the mounting holes, as contrasted with various types of flange blocks or flange units. A pillow block may contain a bearing with one of several types of rolling elements, including ball, cylindrical roller, spherical roller, tapered roller, or metallic or synthetic bushing. The type of rolling element defines the type of pillow block. These differ from "plummer blocks" which are bearing housings supplied without any bearings and are usually meant for higher load ratings and a separately installed bearing.

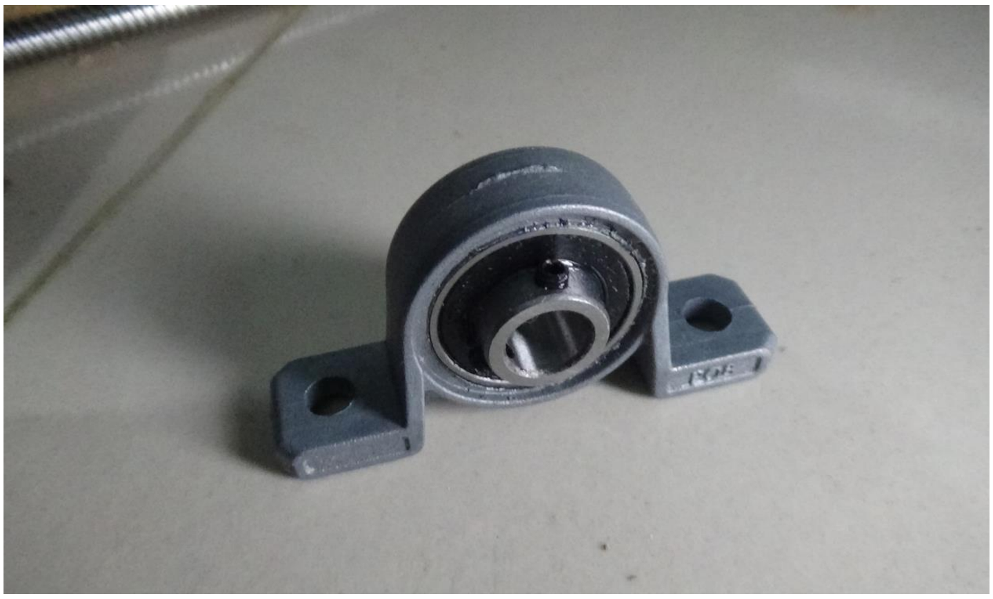

Figure 3.31: Pillow block mounted bearing

The fundamental application of both types is the same, which is to mount a bearing safely enabling its outer ring to be stationary while allowing rotation of the inner ring. The housing is bolted to a foundation through the holes in the base. Bearing housings may be either split type or solid type. Split type housings are usually two-piece housings where the cap and base may be detached, while others may be single-piece housings. Various sealing arrangements may be provided to prevent dust and other contaminants from entering the housing. Thus the housing provides a clean environment for the environmentally sensitive bearing to rotate free from contaminants while also retaining lubrication, either oil or grease, hence increasing its performance and duty cycle.

Bearing housings are usually made of grey cast iron. However, various grades of metals can be used to manufacture the same, including ductile iron, steel, stainless steel, and various types of thermoplastics and polyethylene-based plastics. The bearing element may be manufactured from 52100 chromium steel alloy (the most common), stainless steel, plastic, or bushing materials such as SAE660 cast bronze, or SAE841 oil impregnated sintered bronze, or synthetic materials.

ISO 113 specifies internationally accepted dimensions for plummer blocks.

3.9 Angle bracket

An angle bracket or angle brace or Angle Cleat is an L-shaped fastener used to join two parts generally at a 90 degree angle. It is typically made of metal but it can also be made of wood or plastic. The metallic angle brackets feature holes in them for screws. Its typical use is to join a wooden shelf to a wall or to join two furniture parts together.

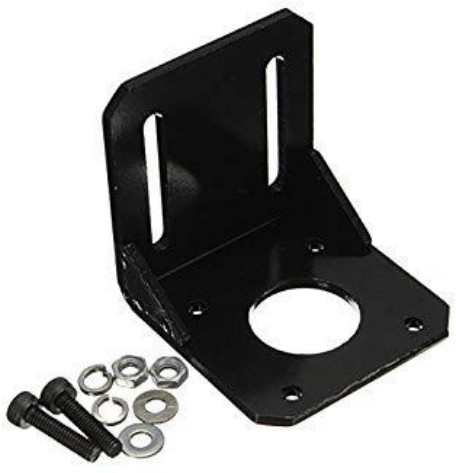

Figure 3.32: L shape angle bracket

Retailers also use names like corner brace, corner bracket brace, shelf bracket, or L bracket.

When the holes are enlarged for allowing adjustments, the name is angle stretcher plates or angle shrinkage.

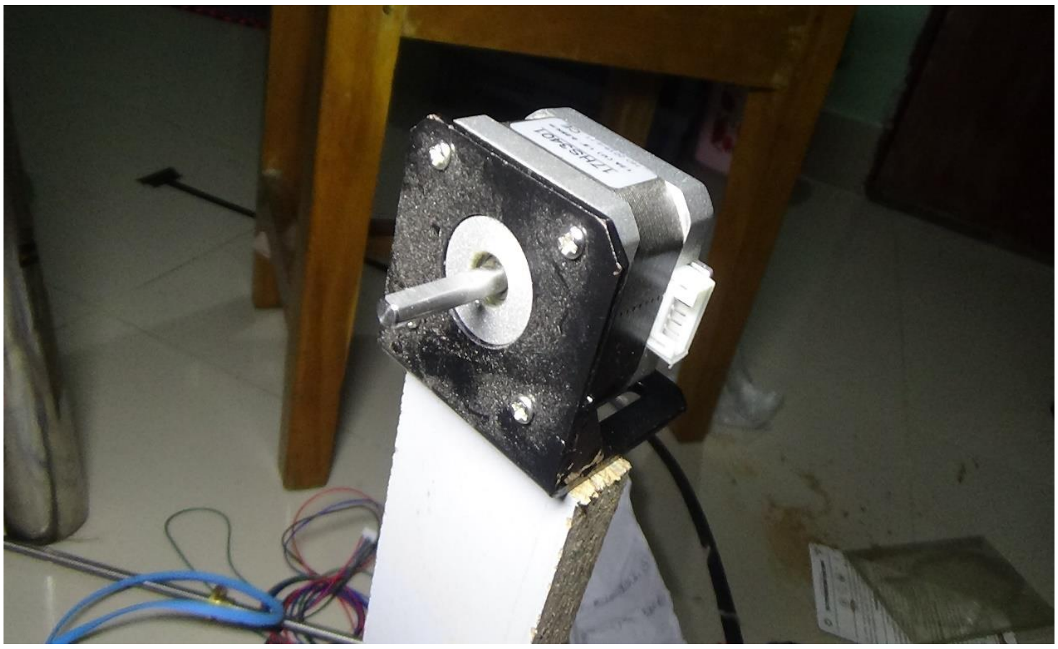

Figure 3.33: L shape angle installed with motor

CHAPTER 4
DISCUSSION AND RESULT

4.1 Introduction

In this chapter we'll discuss about our final project and its result. We'll discuss how it behaves after completing all works. We'll see some working demo of our project. Also we'll check and test all of the procedure to run our project.

4.2 Discussion

In this project we've learned so many things about mechanical and electronic method to control a crane. We've learned how the combination of mechanical and electronics can be very effective in load bearing systems. We can implement these kinds of combination system in many mechanical systems that will help to increase the efficiency of the system and also reduce the man power of total system. We should implement electronic combination system in every possible mechanical system thus it will provide a good service and efficiency of the system. It will be more easy and user friendly to control such mechanical systems.

4.3 Result

After connecting all electronic parts and combining the mechanical parts, we've established a rather basic structure of a gantry crane. The real crane will may not look exactly the same, but will work as the same principle as we've made. Here are some images of our work:

Chapter 5
Conclusion

5.1 Conclusion

In this project a prototype of wireless gantry crane which can be operated by a smartphone has been designed and fabricated. The crane has been designed to a pay load of 0.5-1 kg. The crane is controlled by a smartphone that has Bluetooth connectivity. The crane can be controlled from a distance of 10 to 15 meters.

The system is very easy to use and there is no problem of lots of wires like the traditional gantry cranes.

.

5.2 Future Scope

- We are currently using Bluetooth communication protocol to control the crane wirelessly. But in future we can implement WiFi network system along with internet, that will allow us to not only control the crane from a wider distance than Bluetooth, but also we can control the crane from anywhere in the planet over the internet.
- For now our crane can move only two direction. We can add a third direction to allow the crane to move loads to anywhere on space.
- We can also add voice control system to the crane. So that will allow the physically handicap but educated person to control the crane without any problem.

.

REFERENCES

[1] Abdel-Rahman, E. M., Nayfeh, A. H., and Masoud, Z. N., 2002, "Dynamics and control of cranes: A Review," to appear in *Journal of Vibration and Control* 9.

[2] Al-Alaoui, M. A., 1993, "Novel digital integrator and differentiator," *Electronic Letters* 29(4), 376-378.

[3] Al-Alaoui, M. A., 1994, "Novel IIR differentiator from the Simpson integration rule," *IEEE Transactions on Circuits and Systems-1: Fundamental Theory and Applications* 41(2), 186–187.

[4] Al-Moussa, A., Nayfeh, A., and Kachroo, P., 2001, "Control of rotary cranes us- ing fuzzy logic," in *ASME 2001 Design Engineering Technical Conference and Com- puters and Information in Engineering Conference*, Pittsburgh, PA, September 9-12, DETC2001/VIB-21598.

[5] Armstrong B., Dupont, P., and Canudas, C., 1994, "A survey of models, analysis tools, and compensation methods for the control of the machines with friction," *Automatica* 30(70), 1083–1138.

[6] Astrom, K. J. and Wittenmark, B., 1994, *Adaptive Control*, Addison-Wesley, CA.

[7] Auernig, J. W. and Troger, H., 1987, "Time optimal control of overhead cranes with hoisting of the load," *Automatica* 23(4), 437–447.

[8] Liptak, Bela G. (2005). Instrument Engineers' Handbook: Process Control and Optimization. CRC Press. p. 2464. ISBN 978-0-8493-1081-2.

[9] Tarun, Agarwal. "Stepper Motor – Types, Advantages & Applications"

[10] "Friction and the Dead Zone" by Douglas W Jones https://homepage.divms.uiowa.edu/~jones/step/physics.html#friction

APPENDIX

Code:

```
#include <SoftwareSerial.h>

SoftwareSerial bt(7, 4);

// defines pins numbers
const int stepPin1 = 5;
const int dirPin1 = 2;
const int stepPin2 = 6;
const int dirPin2 = 3;
const int leftLimit = A0;
const int rightLimit = A1;

bool limitFlagLeft = true;
bool limitFlagRight = true;

char A;
int i = 0;
void setup()
{
  Serial.begin(9600);
  bt.begin(9600);
  // Sets the two pins as Outputs
  pinMode(stepPin1, OUTPUT);
  pinMode(dirPin1, OUTPUT);
  pinMode(stepPin2, OUTPUT);
  pinMode(dirPin2, OUTPUT);
  pinMode(leftLimit, INPUT_PULLUP);
```

```
 pinMode(rightLimit, INPUT_PULLUP);

 Serial.println ("Device is ready to go");
}

void loop()
{
 int limL = digitalRead (leftLimit);
 int limR = digitalRead (rightLimit);
 if (bt.available())
 {
  A = bt.read();
  Serial.write (A);
  switch (A)
  {
   case'l': left();
    break;
   case'r': right();
    break;
   case'u': up();
    break;
   case'd': down();
    break;
  }
 }

 if (limL == LOW)
 {
  limitFlagLeft = false;
 }
 else
 {
   limitFlagLeft = true;
 }

 if (limR == LOW)
 {
   limitFlagRight = false;
 }
 else
 {
   limitFlagRight = true;
 }

}
void left()
{
 if (limitFlagLeft)
 {
   digitalWrite(dirPin1, HIGH); // Enables the motor to move in a particular direction
   // Makes 200 pulses for making one full cycle rotation
   for (int x = 0; x < 30; x++) {
     digitalWrite(stepPin1, HIGH);
     delayMicroseconds(800);
     digitalWrite(stepPin1, LOW);
     delayMicroseconds(800);
   }
 }
}

void right()
{
 if (limitFlagRight)
 {
   digitalWrite(dirPin1, LOW); // Enables the motor to move in a particular direction
   // Makes 200 pulses for making one full cycle rotation
   for (int x = 0; x < 30; x++) {
     digitalWrite(stepPin1, HIGH);
     delayMicroseconds(800);
     digitalWrite(stepPin1, LOW);
     delayMicroseconds(800);
   }
 }
}

void up()
{
 digitalWrite(dirPin2, HIGH); // Enables the motor to move in a particular direction
 // Makes 200 pulses for making one full cycle rotation
 for (int x = 0; x < 30; x++) {
   digitalWrite(stepPin2, HIGH);
   delayMicroseconds(800);
   digitalWrite(stepPin2, LOW);
   delayMicroseconds(800);
 }
}

void down()
{
 digitalWrite(dirPin2, HIGH); // Enables the motor to move in a particular direction
 // Makes 200 pulses for making one full cycle rotation
 for (int x = 0; x < 30; x++) {
   digitalWrite(stepPin2, HIGH);
   delayMicroseconds(800);
   digitalWrite(stepPin2, LOW);
   delayMicroseconds(800);
 }
}
```