

“RUBIK’S CUBE SOLVING MACHINE”

A Project submitted to
University of Mumbai for partial completion of the degree of Bachelor
of Science in Information Technology

Under the faculty of Science

By

CHINMAY PRABHU

Seat No. 3038874

AND

PRINCE TRIVEDI

Seat No. 3039093

UNDER THE GUIDANCE OF

PROF. PRANAV SHASTRI



Bhulabhai Desai Road, Kandivali (West), Mumbai-400067

NAAC Reaccredited ‘A’ Grade (CGPA 3.27) and ISO 9001:2015 Certified



KANDIVALI EDUCATION SOCIETY'S

B.K. Shroff College of Arts & M.H. Shroff College of Commerce

Bhulabhai Desai Road, Kandivali (West), Mumbai-400067

NAAC Reaccredited 'A' Grade (CGPA 3.27) and ISO 9001:2015 Certified

CERTIFICATE

This is to certify that Mr. Chinmay Prabhu and Mr. Prince Trivedi have worked and duly completed their project work for the Degree of Bachelor of Science (Information Technology) under the Faculty of Science and their project is entitled, "Rubik's cube solving machine" under my guidance and that no part of it has been submitted previously for any Degree or Diploma of any University.

It is their own work and facts are reported by her personal findings and investigations.

Place:

Date:

External Examiner

Internal Examiner/ Guide

Coordinator

Principal

DECLARATION BY LEARNER

We, the undersigned, Mr. Chinmay Prabhu and Mr. Prince Trivedi hereby declare that the work embodied in this project work titled “Rubik’s cube solving machine” forms our own contribution to the research work carried out under the guidance of Prof. Pranav Shastri and is a result of our own research work. It has not been previously submitted to this or any other University for any other Degree/Diploma.

Whenever reference has been made to previous works of others, it has been clearly indicated as such and included in the bibliography.

We, hereby further declare that all information of this document has been obtained and presented in accordance with academic rules and ethical conduct.

Chinmay Prabhu

Prince Trivedi

Certified by

Prof. Pranav Shastri

ACKNOWLEDGEMENT

We take this opportunity to thank the **University of Mumbai** for giving us chance to do this project.

We would like to thank my **Principal, Dr. Lily Bhushan** for providing the necessary facilities required for completion of this project.

We take this opportunity to thank our **Coordinator, Dr. Vishesh Shrivastava**, for his moral support and guidance.

We would also like to express my sincere gratitude towards our project guide **Prof. Pranav Shastri** whose guidance and care made the project successful.

We would also like to express our gratitude to our **College Library**, for having provided various reference books and magazines related to our project.

Lastly, We would like thank each and every person who directly or indirectly helped us in the completion of the project especially our **Parents and Peers** who supported us throughout our project.

ABSTRACT

The Rubik's cube is a challenging puzzle which has confused a lot of logicians and mathematicians for centuries. Rubik's cube was invented back 1974 by professor Erno Rubik. Though there are solutions available for solving the Rubik's cube, the steps are a little tough to come by. The process of solving the Rubik's cube needs efforts and skills. Erno Rubik himself could not solve the Rubik's cube till a month after he invented. It soon became the world's most selling toy. The Rubik's cube solving machine is a machine which will generate algorithm and accordingly solve the Rubik's cube which is scrambled by the people. The solving will be done gradually. The first step would be getting the input, then comes the processing and lastly the carrying out of the algorithm. The input for the data will be done with the help of webcam. The colours captured by the webcam are stored as data into the database. After getting the input, processing is done where the algorithmic solution for the Rubik's Cube is generated. This project works on the principle of Artificial Intelligence. This project features the design and functioning of a mechanism capable of unconventionally solving a Rubik's Cube Puzzle. The process of generating of algorithm as per the cases will be done totally by the machine. The machine will take the cube from the scrambled state and end up with the cube in the solved state.

Table of Contents

Sr.no	Content	Page no.
1	Introduction	1
1.1	Background	1
1.2	Objectives	3
1.3	About the cube	4
1.4	Colours on cube	5
1.5	Scope	6
1.6	Risk	7
1.7	Purpose	7
1.8	Critical Area	8
1.9	Organisation of Report	8
2	Survey of Technologies	10
2.1	Existing System	11
2.2	Proposed System	11
2.3	Requirement Analysis	12
2.4	Hardware Requirement	12
2.5	Software Requirement	18
2.6	Justification of Selection of Technology	21
3	System Design	23
3.1	Module Division	23
3.2	Data Dictionary	26
3.3	Circuit Diagram	27
3.4	Block Diagram	28
4	Implementation and Testing	29
4.1	Code	29
4.2	Testing Approach	34
4.2.1	Unit Testing	34
4.2.2	Integration Testing	38
5	Results and Discussion	39
6	Conclusion and Future Work	43
7	References	45

CHAPTER 1: INTRODUCTION

1.1 Background

Rubik's cube was invented in 1974. Rubik's cube has started to reappear in the marketplace. It is one of the best selling toys worldwide. Understanding the complication of the puzzle is very essential. In total there are 43,252,003,274,489,856,000 i.e. 43 quintillion possible cases. The Rubik's cube is a challenging puzzle which has confused a lot of logicians and mathematicians for centuries. Erno Rubik himself could not solve the Rubik's cube till a month after he invented. There are 9 stickers on each of the six sides of the cube. So, in total there are fifty four stickers. The invention of Rubik's cube was done by Erno Rubik, for teaching three dimensional rotations to his students for a better understanding of geometry. There exist multiple ways and techniques for solving the Rubik's cube. There is also a strong online cubing community who have devoted their time to finding better solutions or time saving solutions to solve the cube. World Cube Association organizes speed cubing competitions globally encouraging more and more competitors to start speed solving. In the year 1980 & 1981 Rubik's cube was awarded as the toy of the year. The puzzle is made up of twenty six miniature cubes known as cubies. The current world record for solving a Rubik's cube was achieved by Yusheng Du where he solved in 3.47 seconds. Selective puzzles are selected as official events for the competitions. The process of solving the Rubik's cube needs efforts and skills. In this project, we have used Arduino to build a Rubik's cube solving machine. The input for the data will be done with the help of webcam. The colours captured by the webcam are stored as data into the database. Scanning of sides through the camera may sometimes result in incorrect input of colours which may lead to a standstill; this can be corrected by scanning the entire cube over again. The generation of right algorithm is done completely by the machine without any human involvement. After getting the input, processing is done where the algorithmic solution for the Rubik's Cube is generated. This project features the design and functioning of a mechanism capable of unconventionally solving a Rubik's Cube Puzzle. The rotation of sides and turning of the

cube will be done with help of servo motors. Servo motors will be turning the hands of the mechanism in clogs as per the requirement of generated algorithm. The mechanical structure consists of an embedded computer, a mechanical apparatus for actually manipulating the cube, and peripheral devices such as cameras. The machine is intended to accept a jumbled Rubik's Cube, visually estimate it, and decide how to solve the cube through direction. We planned to show that problems can be solved much efficiently using microelectronics. The mechanism is equipped with the required hardware to swiftly twist and turn the cube until the puzzle is completely solved. We have used Python, CPP and Arduino. Excellent readability is offered by Python. Also Python is simple to learn. The Rubik's cube solving machine will be displayed in various speed cubing competitions. As per a survey conducted, only 6% of total world population can solve a Rubik's cube. Planning, learning, reasoning and ability to move and manipulate objects are some of the traditional goals of Artificial Intelligence. Artificial Intelligence has taken a huge growth in the 21st Century. Vast amounts of data and theoretical understanding. The Rubik's cube solving machine, solves the cube with this concept of Artificial Intelligence. Rubik's cube has a very definite colour scheme where White is always opposite to Yellow, Green is always opposite to Blue and Red is always opposite to Orange. Every Rubik's cube can be solved in 20 moves or less. First ever world championship took place in 1982 in Budapest where the winning solve was 22.95 seconds. There exist multiple methods for solving a Rubik's cube. This mechanism will encourage more people to get into solving the Rubik's cube. One of the major problems with the mechanism is that if the size of the cube is more or less than 56mm, then the cube may not fit properly into the cube holder of the machine. There exist lots of benefits to the person solving the cube as it improves the eye-hand coordination. It improves logical reasoning. Keep the scrambled cube in the machine, and then the machine will take the input via camera and scan each face of the cube. The colours will then be saved into the record. The machine will then process the data and produce an algorithm therefore, by following this machine generated algorithm, the mechanic robot will do turns on all sides of the cube (clockwise / anticlockwise) until solved state of the cube is reached and the cube will be solved. German engineer Albert Beer achieved a Guinness World Record for creating the fastest robot to solve a Rubik's Cube. The Record was for solving in 0.637 seconds in January 2016. The Robot had six mechanical arms and it solved the cube in perfectly 21 moves. The world record attempt was in Munich, Germany at the end of 2016.

1.2 Objectives:

We planned to show that problems can be solved much efficiently using microelectronics. The main objective of our project was to plan and build a mechanism that can solve a Rubik's cube. This goal can be satisfied by breaking down the project into smaller objectives. Solving a Rubik's cube is a step by step process. Usually 1 colour is solved at the beginning and rest of the colours all together. But when generating algorithm step by step algorithm need not be followed as cube can be solved in 20 moves or less with help of computer generated algorithm. Rubik's cube solving machines have been built earlier and are available but the goal was to significantly reduce the cost of the build. First the robot should be able to imagine all the sides and colours on the cube. Then with the data found run an algorithm in its record to solve the cube for all the same colour on each side. After the algorithm is ran the mechanism shall then take the cube and turn the sides in the equivalent pattern to solve the cube. Image processing is done by scanning the scrambled state of cube and storing the data into database. The objective of the machine is to match the colours on all sides of the cube. Scanning of sides through the camera may sometimes result in incorrect input of colours which may lead to a standstill; this can be corrected by scanning the entire cube over again. The time taken by the machine to solve the cube must not be more than 30 minutes and the final cube must be solved correctly. Rubik's cube is a great way to sharpen the mind, improve eye to hand coordination and increase concentration power. Solving a Rubik's cube will not only give quickness to the fingers but also improve your reflexes. It has been found out that engaging in solving cubes gives you a sharper intellect. Because the algorithms have to be memorized, the memorizing ability becomes strong. The Cube solving machine will be showcased in speed cubing competitions across various levels to inspire people to start learning to solve a Rubik's cube. The model we will be using is the iterative model. We will be building the system incrementally, starting from basic features and gradually adding more features until the entire system is completed. The cube going to be used is QiYi Thunderclap whose dimensions are 56mm x 56mm x 56mm. The puzzle or cube with different dimensions may not fit properly into the machine as the holder on the bottom might not be able to hold the cube firmly for rotations. The rotation of sides and turning of

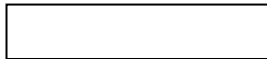
the cube will be done with help of servo motors. Servo motors will be turning the hands of the mechanism in clogs as per the requirement of generated algorithm.

1.3 About the cube:

Rubik's cube was invented in the year 1974 by professor of architecture Erno Rubik at the academy of applied arts and crafts in Budapest, Hungary. Rubik's Cube was invented to as a working model to explain three dimensional geometry. After designing the cube, he himself could not solve it for over a month. Cube was basically invented for the students to have a better understanding about the axes. There are over 43 quintillion possible cases in a Rubik's cube. It was like a secret code he made for himself but was not capable to interpret it. In the year 1980 & 1981 Rubik's cube was awarded as the toy of the year. The puzzle is made up of twenty six miniature cubes known as cubies. Since a couple of years, speed cubing competitions have begun to take place globally. There exist lots of benefits to the person solving the cube as it improves the eye-hand coordination. It improves logical reasoning. It is a puzzle with no secret information. It has been found out that engaging in solving cubes gives you a sharper intellect. A standard Rubik's cube measures 5.7cm on each of its six sides. The cube has eight corners and twelve edges. Centre piece of each side represents the colour of its respective side. German engineer Albert Beer achieved a Guinness World Record for creating the fastest robot to solve a Rubik's Cube. The Record was for solving in 0.637 seconds. The colour scheme on the Rubik's cube is fixed. There are six colours on the cube for each of its six sides. By building the machine for solving the cube we planned to show that problems can be solved much efficiently using microelectronics. The standard size for a regular Rubik's cube is 56mm. Puzzles available in the market have huge variety in sizes and shapes. Rubik's cube has a set of notations which will help the solver to turn the layers in the side of the cube. There are speed cubing competitions arranged globally. Every individual who competes in these speed cubing competitions are known as speedcubers. The cube we will be using is the QiYi Thunderclap.

1.4 Colours on a regular cube:

- White



- Red



- Green



- Yellow



- Orange



- Blue



1.5 Scope

For Rubik's cube solvers who wish to see their cubes getting solved by machine without any human support and understanding about artificial intelligence. The Rubik's Cube Solving Machine will firstly be scanning the cube from all the six sides and then generate an algorithm accordingly within a minute, and then get ready to ready to start execution. The Cube solving machine will be showcased in speed cubing competitions across various levels to inspire people to start learning to solve a Rubik's cube. The process of solving the Rubik's cube needs efforts and skills. In this project, we have used Arduino to build a Rubik's cube solving machine. The input for the data will be done with the help of webcam Rubik's cube is a great way to sharpen the mind, improve eye to hand coordination and increase concentration power. The mechanism is equipped with the required hardware to swiftly twist and turn the cube until the puzzle is completely solved. Because the algorithms have to be memorized, the memorizing ability becomes strong. The machine shows how a machine can do the thinking on its own, generate an algorithm and execute it all by itself. Planning, learning, reasoning and ability to move and manipulate objects are some of the traditional goals of Artificial Intelligence. Artificial Intelligence has taken a huge growth in the 21st Century. Vast amounts of data and theoretical understanding. The Rubik's cube solving machine, solves the cube with this concept of Artificial Intelligence. The servo motors must always function well so that the cube is perfectly flipped, and the slices are turned in right clogs. We will be using Cube Explorer software. Cube Explorer is a great program for finding solution to your own Rubik's cube. Cube Explorer is cube solving software which can be used on desktop computers and runs on all operating systems. This machine will attract more people to learn solving the cube. Getting data from the cube is going to be done with the help of webcam. Getting data from camera as input is also known as Image Processing. It also teaches you to break multifaceted tasks into lesser ones which will improve your skills for problem solving. Also the enthusiasts can race along with the machine to see who can solve it quicker. Image processing will be a critical task. Also generating the algorithm within the stipulated time frame is one of the important tasks. The

cube solver will surely motivate more participants to compete in competitions. Cube solving machine shows how smart the machine can get with the concept of Artificial Intelligence.

1.6 Risks

- Colour detection problems like the scanning might not be proper due to issues in the lighting.
- The algorithm might not be reliable in every case. The algorithm must always have solution with 20 moves or less.
- Machine could consume lot of power. If the power supply is not proper then the machine may not work as intended.
- Risk of too much memory. The data might keep storing up because of which the space might get full. We require more free space to run the solve smoothly.
- If dimensions of the cube are different than regular i.e. 56mm x 56mm x 56mm then the cube may not fit properly into the holder. And if this is the case, the flipping and turning will be a problem.
- Complex machines like this Rubik's cube solvers may be very expensive.
- If the power supply is low, the machine may not work.
- If the puzzle is not kept in properly, it may damage the servo motors.

1.7 Purpose

We planned to show that problems can be solved much efficiently using microelectronics. The aim of our project was to plan and build a mechanism or a machine that can solve a Rubik's cube. Iteration can solve many more problems than just solving the cube. Learning algorithms used in solving of the Rubik's cube can be applied in other fields that require intricate computational investigation such as cryptography. The Rubik's cube solving machine will be displayed in various speed cubing competitions. This mechanism will encourage more people to get into solving the Rubik's cube. The machine shows how a

machine can do the thinking on its own, generate an algorithm and execute it all by itself. Planning, learning, reasoning and ability to move and manipulate objects are some of the traditional goals of Artificial Intelligence. The cube is not just any random puzzle. It improves your thinking capacity. It helps you in growing your physical as well as mental skills. It improves reflexes as you solve it. Gives quickness to fingers. It also teaches you to break multifaceted tasks into lesser ones which will improve your skills for problem solving.

1.8 Critical areas:

- Image processing done by scanning the scrambled state of cube, and storing the data into database.
- Turning of the sides and flipping over the cube with servo motors.
- Generation of the right algorithm for the cube to move from scrambled state to the solved state.

1.9 Organisation of report:

Chapter 1 - Introduction

In this chapter, the basic introduction to our topic “Rubik’s cube solving machine” is given. The first chapter comprises of information such as background, objectives, scope, risks, purpose, critical areas.

Chapter 2 – System Analysis

In this chapter, information about the Existing system and proposed system is given. Then detailed Requirement Analysis is done. All types of Hardware requirements and

Software Requirements for the project are mentioned in this chapter. This chapter also covers the Justification of Selection of Technology.

Chapter 3 – System Design

In this chapter, information regarding the system design is mentioned. This chapter also covers Module Division, Data Dictionary, ER Diagrams and UML Diagrams.

Chapter 4 – Implementation and Testing

In this chapter, the code and testing approach is given. Testing approach involves Unit Testing cases and results. This chapter also involves the cases and results of Integration System.

Chapter 5 – Results and Discussions

In this chapter, we have given scrambles and the time taken by the machine to solve the entire cube. The time may differ for different cases.

Chapter 6 – Conclusion and Future Work

In this chapter, the future plans are mentioned. This is the chapter, where the project is successfully completed and the report is concluded.

Chapter 7 – References

In this chapter, URLs of the various websites referred to for fetching the required data are mentioned. The URLs are for information regarding cubes, existing systems and data.

CHAPTER 2: System Analysis

Intelligence shown by the machines is termed as Artificial Intelligence (AI). As academic regulation, in the year 1956 the term Artificial Intelligence was founded. Planning, learning, reasoning and ability to move and manipulate objects are some of the traditional goals of Artificial Intelligence. Artificial Intelligence has taken a huge growth in the 21st Century. Vast amounts of data and theoretical understanding. One of the main goals for building the Rubik's cube solving machine was to reduce the cost. Instead of using motors for all of its sides we have used only 2 servo motors. One for flipping over the cube and other for turning the sides. The machines available in market are expensive and not portable. The project we built is light weight and cost effective. The Rubik's cube solving machine, solves the cube with this concept of Artificial Intelligence. The objective is building a machine which can process the input and accordingly solve the Rubik's cube. For the input, the machine will scan the Rubik's Cube from all of its six sides with the help of a webcam. Accordingly, data will be processed resulting into solution of the Rubik's Cube. The turning of sides of the cube will be done with the help of servo motors. Generation of right algorithm to solve the cube is entirely done by the machine. The solving machines available in market are made with all different kinds of materials ranging from Lego Blocks to 3D Printed Grippers. We will be using wooden base and arms made with popsicle sticks. We will be using incremental model. The objective of iterative development is to build the system incrementally, starting from basic and gradually adding more features until the entire system is completed. The cube to be used in this Rubik's cube solving machine is the Thunderclap 56mm cube which is also the size of standard Rubik's Cube. Core part of Artificial Intelligence is the knowledge learning. Machines can act and react like humans only and only if they are provided with complete information or data relating to the world. Robotics is one of the major fields related to Artificial Intelligence. We will be using Arduino as it is very convenient to manage power inside it and it had a features or built-in voltage regulation. Machines need intelligence to carry out tasks such as object manipulation along with sub-problems of motion planning.

2.1 Existing System:

There exist Rubik's Cube solving machines which are much higher in cost compared to our costing. The solving machines available in market are made with all different kinds of materials ranging from Lego Blocks to 3D Printed Grippers. German engineer Albert Beer achieved a Guinness World Record for creating the fastest robot to solve a Rubik's Cube. The Record was for solving in 0.637 seconds in January 2016. The Robot had six mechanical arms and it solved the cube in perfectly 21 moves. Rubik's cube solvers which are available are either taking the input data from the camera or the user can input colour block by block on Python GUI. The robot had six mechanical arms and used stepper motors for all the arms. The design of the finished product is too complex and requires high supply of power at all times. Since components used in this mechanism were in abundance, the weight of the whole finished product was high; also the system built was not portable.

2.2 Proposed System:

We have created the Rubik's cube solving machine by using wood as a base to cut the cost of the finished product. For the arms we have used wooden popsicle sticks. The cube will be placed on a cube holding stand which is made with popsicle sticks too. The holder is then connected to a servo motors this will result in turning of the sides of Rubik's cube. For flipping over the cube so that rotations are possible from all the six sides of the cube, a servo motor is connected to the arm of the solving machine. The machines which generally require 6 mechanical arms for turning of its sides, this solver will complete the solve with just one mechanical arm. Also the arm is not 3D printed which is why the whole machine in itself is cost effective. Since the components used are light weight, the system is portable and the maintenance cost is low. The model that we have created requires less power compared to the power supply requirements of available machines. The data will be the colours of the scrambled cube and we will input the data with the help of webcam. Once the data is stored in the database, the data will then be processed into the Cube Explorer software and a valid

solution algorithm will be ready. This algorithm will be executed in machine language and run on servo motors. Servo motors will then rotate the cube, move layers, flip it over as required to finish the cube.

2.3 Requirement Analysis:

This Project shows and describes the design and implementation of a machine capable of solving a Rubik's cube puzzle. The robotic system consists of an embedded computer, a mechanical apparatus for physically manipulating the cube, peripheral devices such as camera. The machine is designed to receive a scrambled Rubik's Cube and determine how to solve the cube through manipulation. The machine is equipped with necessary hardware parts to quickly manipulate the cube until it is solved.

2.4 Hardware Requirements:

- **Pentium III Processor**

The Pentium III is a microprocessor which has been designed by Intel. The Pentium III is comparatively faster, specially for applications written to take advantage of its "Katmai New Instructions" (its the code name for Pentium III). It is possible to run 3-D, imaging, speech recognition and audio applications more swiftly and rapidly. Clock speeds up to 800 MHz is offered by Pentium III Processor. Pentium III was first released for sale on October 25, 1999. It was the first x86 CPU to include a only one of its kind retrievable, identification number, called PSN (Processor Serial Number).



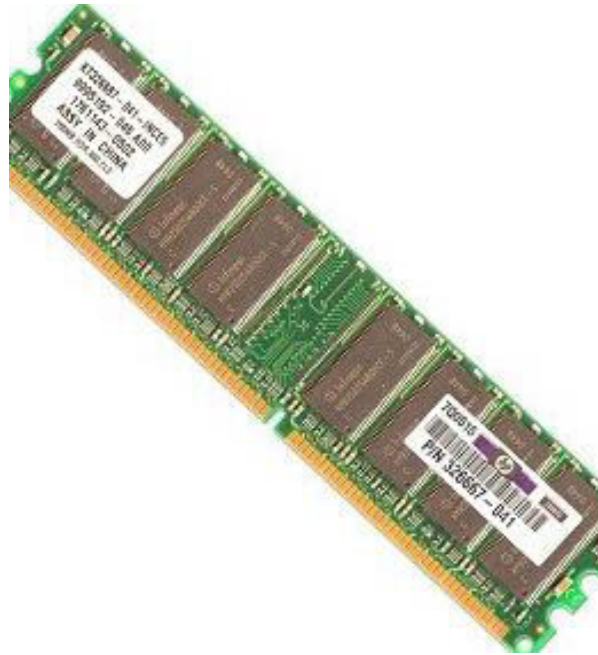
- **Webcam**

A webcam is a video camera. A webcam will feed its image in a computer. The image is stored in the computer database as input after it has been captured by the camera. Camera will act as an input device and data here is the colours of the cube.



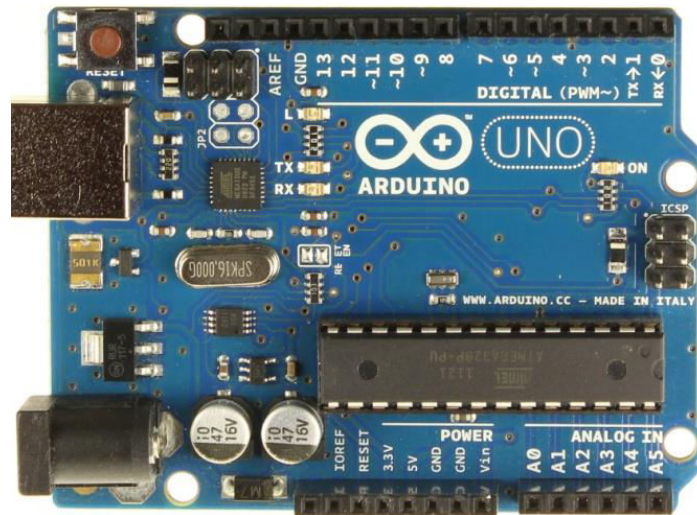
- **256 Megabytes (MB) of RAM**

Random Access Memory, or RAM, is how your computer opens, reads and also runs the software for you to use, from your OS (Operating System) to your PC games. RAM modules are often measured in megabytes and gigabytes. RAM or Random Access Memory is a form of computer data storage which stores the data and the machine code being used. It allows reading or writing data.



- **Arduino UNO**

Arduino is one of the main components of this mechanism of building a Rubik's cube solving machine. Arduino is an open-source electronics platform based on easy-to-use software and hardware. Arduino is a programmable open source micro controller which is available in the market. Computer is where we process the data, and the physical components are the two servo motors. A huge variety of ports and libraries are offered by arduino, enabling endless possibility for usage. Power is supplied to the Arduino by the USB cable which is connected to the computer (PC).



- **Servo Motors**

We will be using two servo motors in this project. A servo motor is a rotary actuator that allows for accurate control of angular position, acceleration and velocity. We use two servo motors of which one will be used to spin the layers of the cube and the other to flip the cube onto other sides. Servo motors are widely used in the fields of robotics and mechanisms. Integrated servomotors are designed so as to include the motor, encoder, and driver.



- **Display with 256 colours**

8-bit CG (colour graphics) is a way of storing image information in a computer's memory or in an image file, such that each pixel is represented by one 8-bit byte. The highest number of colour that can be displayed at a time is 256.

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59
60	61	62	63	64	65	66	67	68	69
70	71	72	73	74	75	76	77	78	79
80	81	82	83	84	85	86	87	88	89
90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109
110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129
130	131	132	133	134	135	136	137	138	139
140	141	142	143	144	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199
200	201	202	203	204	205	206	207	208	209
210	211	212	213	214	215	216	217	218	219
220	221	222	223	224	225	226	227	228	229
230	231	232	233	234	235	236	237	238	239
240	241	242	243	244	245	246	247	248	249
250	251	252	253	254	255	256	257	258	259

- **80GB Hard disk**

A hard disk is an electromechanical data storage tool that uses magnetic storage to store and regain digital information using one or more firm rapidly turning disks (platters) covered with magnetic substance or material.



- 32-bit Windows operating system


The terms 32-bit and 64-bit refer to the way a computer's processor handles information.

[View basic information about your computer](#)


Windows edition

Windows 7 Ultimate

Copyright © 2009 Microsoft Corporation. All rights reserved.



System

Rating:	 Windows Experience Index
Processor:	Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz 2.10 GHz
Installed memory (RAM):	4.00 GB (2.70 GB usable)
System type:	32-bit Operating System
Pen and Touch:	No Pen or Touch Input is available for this Display

2.5 Software Requirements:

- Microsoft Windows 10

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT. It is the successor to Windows 8.1 and on July 15, 2015 was released to manufacturing.



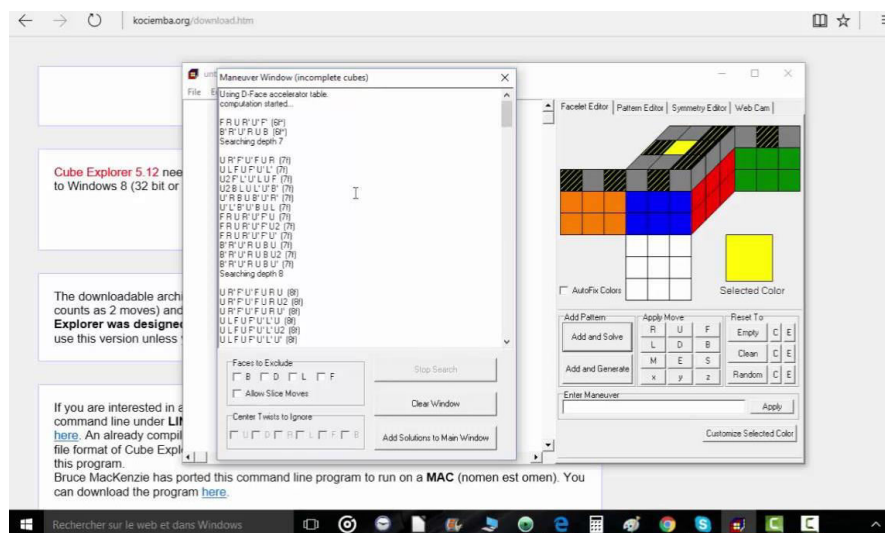
- Arduino IDE

Arduino Integrated Development Environment (IDE) is cross platform application which is written in Java. It is used to write and upload programs on the Arduino board. Arduino also supports special rules of code for C and C++. Many common input and output procedures are supplied by a software library from the writing project in Arduino IDE. User-written code usually requires only the two basic functions, 1.For starting the sketch and 2.main program loop. Arduino IDE employs the program to convert the run able or executable code file into a simple text file that is then loaded into the arduino board by a loader program in the board's firmware.



- Cube Explorer Software

Cube Explorer is software which will give you the best possible algorithm to whatever state the cube currently is in. Cube Explorer is cube solving software which can be used on desktop computers. Cube Explorer manages the algorithm generation process in the Rubik's cube solving machine.



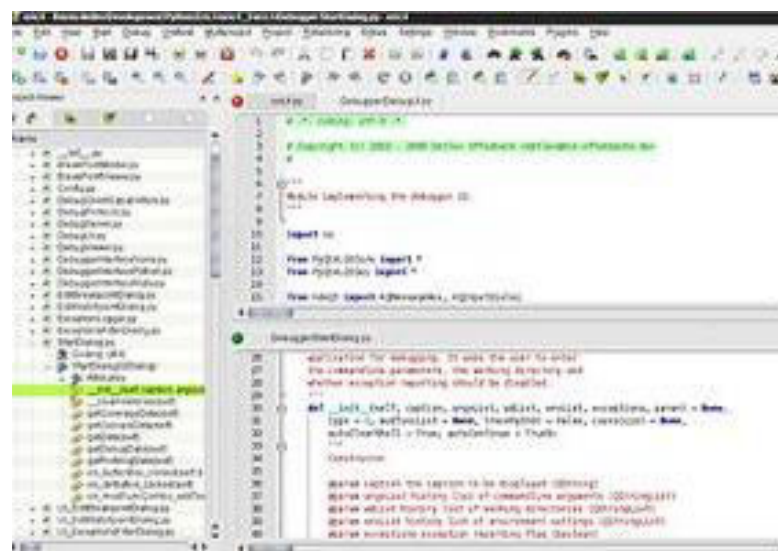
- C Language for Arduino

C is a general purpose, vital computer programming language, supporting structured programming, lexical variable scope and recursion, while a static type system prevents many accidental operations.



- Python Compiler

The Python Compiler package is a device for analyzing Python source code and generating byte code. The compiler contains libraries to generate an abstract syntax tree from Python source code and to generate Python byte code. It uses the built-in parser and standard parser module to generate an abstract syntax tree (AST) and then python byte code



2.6 Justification of Selection of Technology:

Python is third-party module that makes it capable of interacting with most of the other languages and platforms; hence we have decided to use Python Technology. Excellent readability is offered by Python and is simple-to-learn language.

Features of Python

- Easy to code and read.
- Python is portable.
- Python is interpreted.
- Python is freely available - Open-Source.
- Python is Object-Oriented.
- Python is a high level programming language.

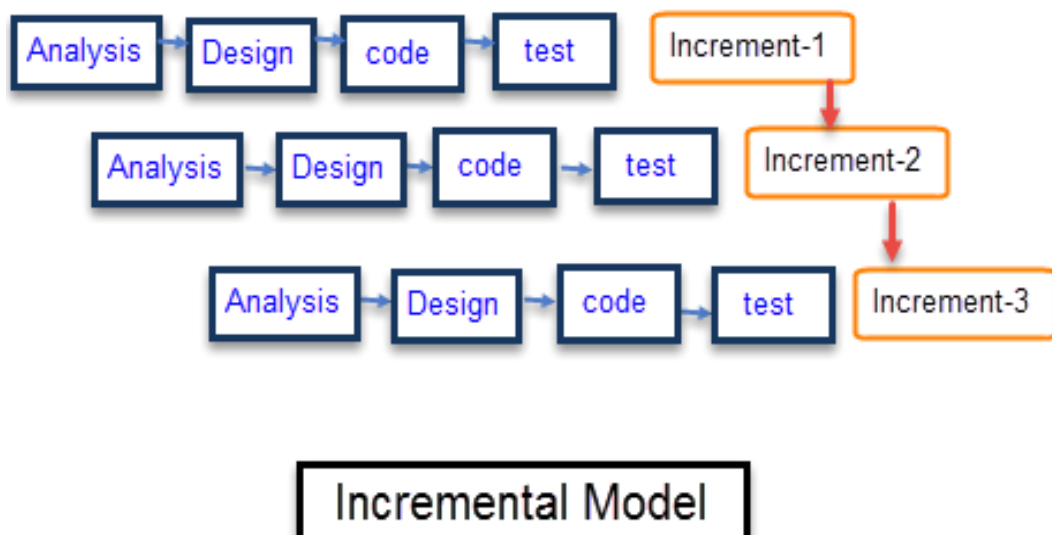
Features of Arduino

- It offers easy USB interface. This allows the interface with USB as this is like a serial device.
- It is very convenient to manage the power inside it as it has features for built-in voltage regulation.
- It also offers 32 kb of flash memory for storing the code.
- It has a button to reset the program on the integrated chip.
- It is an open source design and the advantage of being open source is that it has a large community of people for troubleshooting. Being open sourced makes it easy to debug the programs.

Model Used

We have decided to use Incremental / Iterative model for building the Rubik's cube solving machine. The main reason for using iterative model is to build the system incrementally, starting from basic features and gradually adding more features until the entire system is completed. When it satisfies all of its requirements the product can then be declared as complete. Until the product is finished, the product is tested incrementally.

Incremental model chart



Chapter 3 – System design

3.1 Module Division:

- Main

The Main module is responsible for writing together all the different secondary modules involved into this project. Also the finite state machine responsible for controlling the behaviour of the process of determining initial state of the cube and solving it.

The finite state machine in Main began by keeping the machine in the determine state mode. Here, the machine began rotating the faces of the Rubik's cube to determine the colour of each individual block on each side of the cube. After determining the state of the Rubik's cube using the modules like determine state, the state machine transitioned into the user debug mode. Here, it sent the state representation of the cube over serial, so that a connected computer running a parser python program could output the expected state of the cube. After outputting the cubestate, the machine transitioned into the solution mode where the machine would be using the solving algorithm module and update state module to determine the solution. Then the machine would transition into physical solve mode where the sequencer module would be feeding the servo motor module one by one all of its commands. Once the solution is found and implemented, the machine would go into an idle state and wait for the user to put in a new scrambled cube.

- Determine State

The design of Determine state consisted of finite state machine consisting of setup, prep, idle, observe, and done. In setup all variables are reset. The state then moves to prep.

In prep state, send_setup_moves, which is responsible for signalling spin_all goes high. The cubestate register is bit shifted left by 3 and then the state is set to idle.

In the Idle state, Determine_state module waits for colour sensors to produce a stable and reliable value. After this, the finite state machine transitions to observe state.

Into the observe state, the value given by colour sensor is recorded lowest order bits of cubestate. The counter is then incremented, and the state returns to prep.

Once the counter reaches 48, signalling that all the pieces of every side of the cube have been observed, the state is then transitioned to done. The module then signals main that observation has been completed and process of generation of algorithm can now begin.

- **Spin All**

This module consists of finite state machines with 2 states. idle and send moves are the two states. In the idle state, the module waits for single case statement from determine_state. Each case will return moves that, when executed on the puzzle, places the sticker corresponding to the value given in front of the colour sensor.

- **Solving Algorithm**

This module is responsible for producing a sequence of moves that, when executed to the scrambled cube/puzzle, would solve it. The method implemented in this module is the brute force algorithm. This is a method which will result in the solution of cube in less than 20 moves. It is also termed as god's number.

This module consisted of four states- move, update_state, wait, and setup. The finite machine begins in setup. In this state, all variables associated with solving the puzzle are always set to their initial values.

Once, the start signal is received the finite state machine transitions to move state. Here, the next sequence of moves to be applied to the puzzle is determined. After the sequence is produced, the finite state machine transitions into the update_state.

In the `update_state`, the finite state machine waits for `solving_algorithm` to receive signal from `update_state` showing that it has updated the `cubestate` according to `solving_algorithm`. Once this occurs, finite state transitions into the wait state.

Wait state ensures that `solving_algorithm` does not keep on producing new moves by itself. This state has a counter that counts up to 60 clock cycles, at which point the finite state machine goes back to the move state, and the cycle continues until the cube is solved.

- **Sequencer**

This is the module which is responsible for creating a queue of moves to be executed in the solution phase and for returning one move at a time for the servo motors in physical solve phase.

While the machine is determining solution to scrambled puzzle, sequencer module will queue up the next moves to the solution. This module contains 200 4-bit wide registers capable of holding 200 moves. The module sequentially stores the next moves to reach the solution state.

While the machine is solving the cube, the sequencer module produces the next move to execute as output to the servo motors. Each time the module receives signal of completion of the move, next move is loaded. It increments the counter showing number of completed moves. When number of moves completed equals to the move counter it sets a signal showing that the sequence has been fully executed. And then enters into the idle state.

- **Colour Sensor**

This module contains the necessary setup values for the RGB colour sensor and acts as a finite state machine for each sensor. Setup module is responsible for writing the necessary configuration registers. The poll module is responsible for polling the values of red, green, and blue light sensors at a frequency.

3.2 Data Dictionary:

- Actions

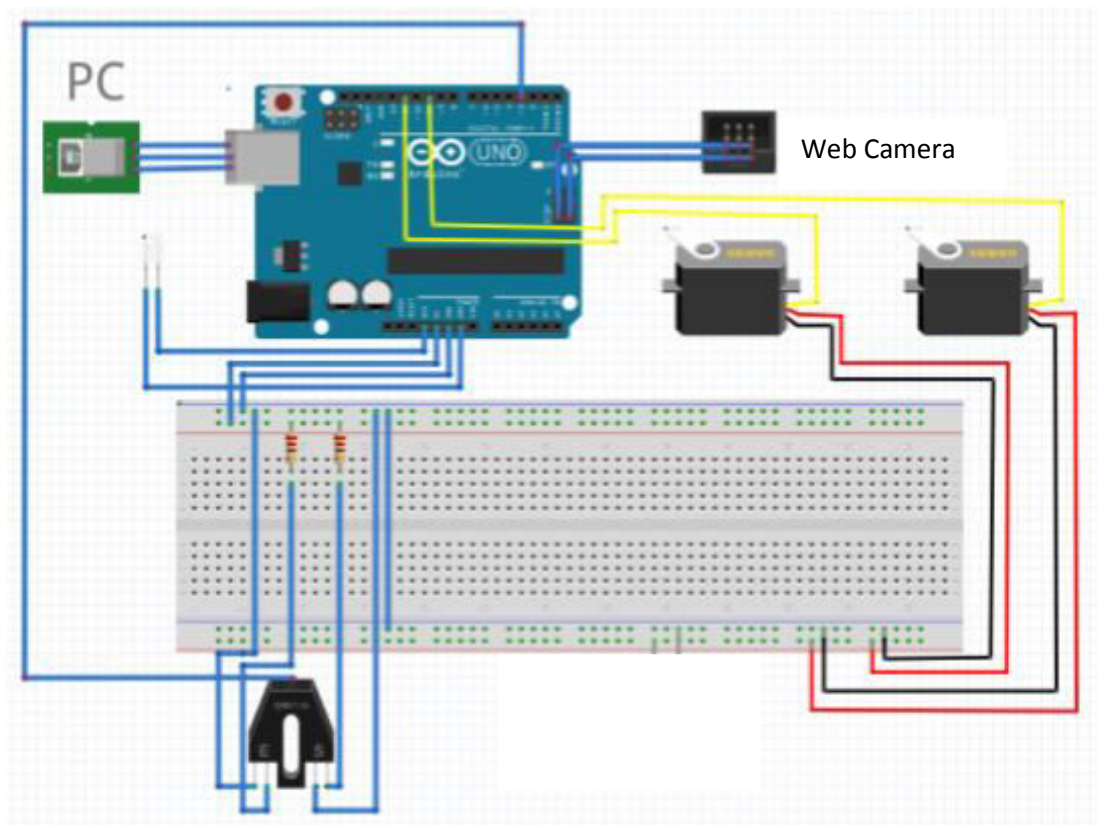
Actions are possible rotations to the cube in any state. There is a way of naming the actions followed by cube sides we are rotating. Names of the sides are : Left, Right, Top, Bottom, Front and back. For every side we have 2 actions, corresponding to clockwise rotation and counter-clockwise rotation. The side to be rotated must always be facing the bottom as the Rubik's cube solving machine that we have built will turn layers of the puzzle only by the cube holder on the base. The arm will help in turning the cube over from side to side until the desired side to be rotated is on the bottom.

- States

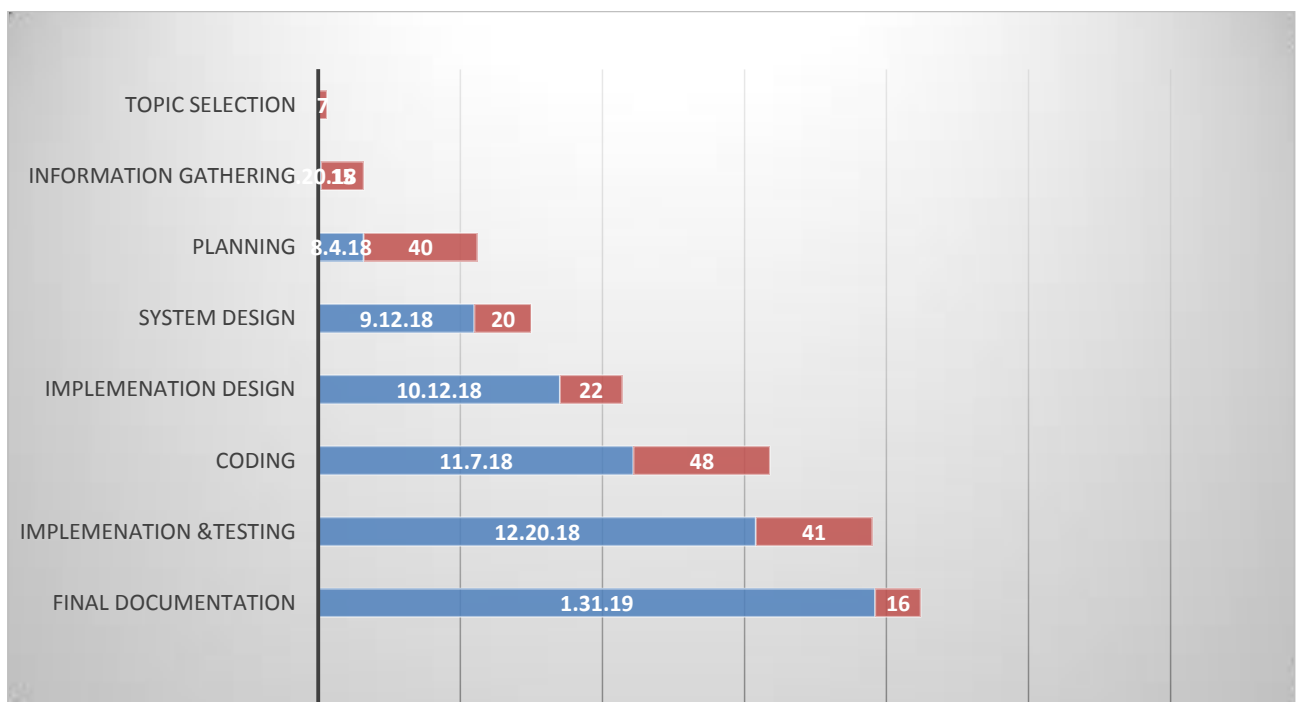
A state is configuration of the stickers on the puzzle. The size of state space is a lot. But amount of state or the size is not the only complication here. We also have different objectives such as:

- Avoiding Redundancy: We can represent state of the puzzle by just saving the colours of every sticker on each side of the cube. But the combinations are higher than the state space size, which is why this is highly redundant.
- Neural-Network Friendliness: Not every data representation is equally good as an input for the neural network. This is applicable for all Machine Learning.
- Performance: We need to implement all the actions and these actions require to be executed as quickly as possible. If the representation is compact but requires lengthy process for every rotation, it will become too slow.

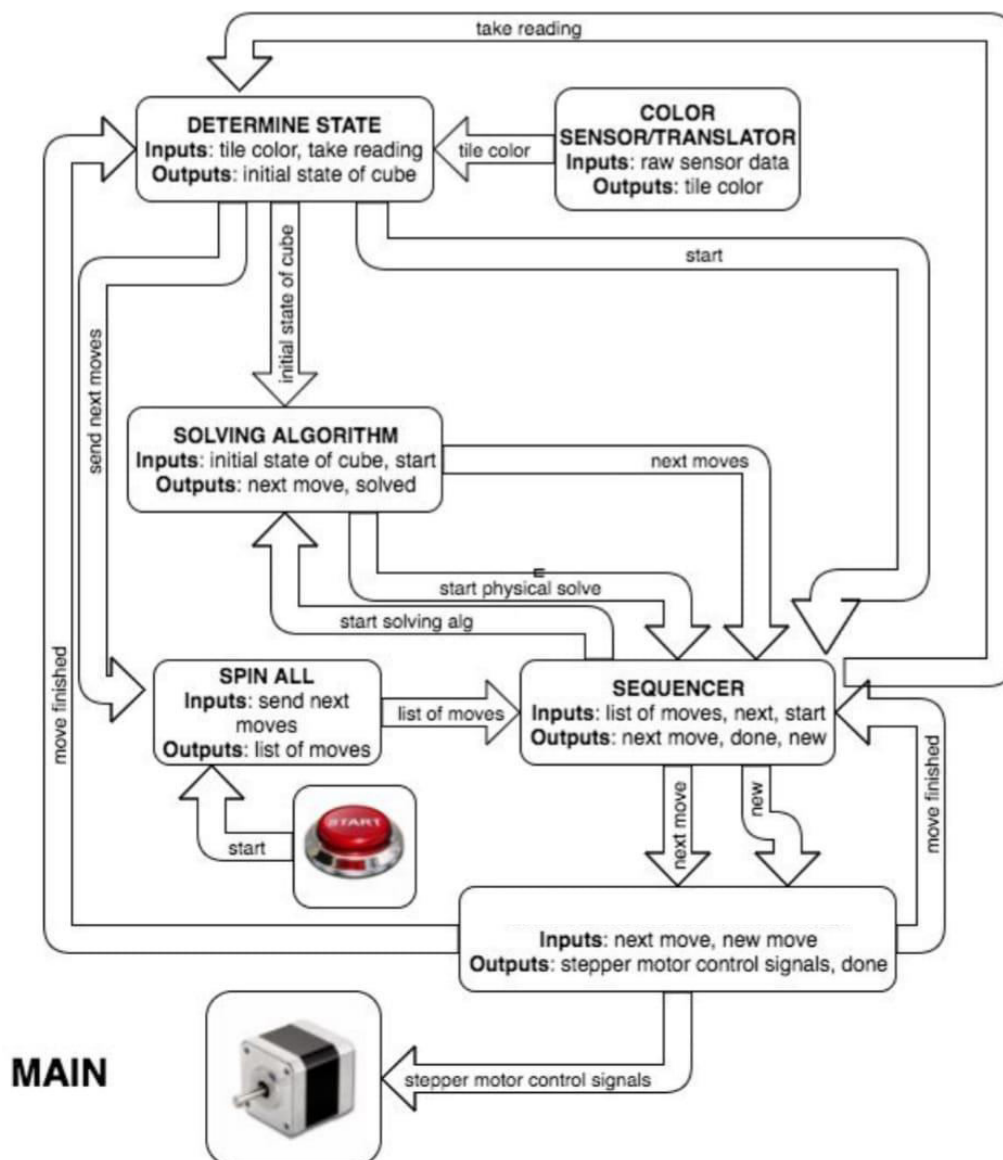
3.3 Circuit Diagram:



GANTT CHART:



3.4 Block Diagram:



GANTT CHART

START DATE	END DATE	DESCRIPTION
7/15/18	7/22/18	TOPIC SELECTION
7/20/18	8/5/18	INFORMATION GATHERING
8/4/18	9/14/18	PLANNING
9/12/18	10/2/18	SYSTEM DESIGN
10/12/18	11/4/18	IMPLEMENTATION DESIGN
11/7/18	12/25/18	CODING
12/20/18	1/31/19	IMPLEMENTATION & TESTING
1/31/19	2/16/19	FINAL DOCUMENTATION

Chapter 4: Implementation and Testing

4.1 Code

```
#include <VarSpeedServo.h>

VarSpeedServo arm;
VarSpeedServo cube;

int sp = 70;
int sphold = 60;
int back = 80;
int front = 33;
int hold = 61;

int center = 80;
int Mvcenteranticlock = 100;
int Mvcenterclock = 70;
int anticlockpos = 155;
int clockwisepos = 10;
int Mvanticlock = 175;
int Mvclockwise = 0;

String rx_byte;

void setup() {
    // put your setup code here, to run once:
    arm.attach(5);
    cube.attach(6);
    arm.write(back);
    delay(500);
    cube.write(center);
    Serial.begin(9600);
```

```
delay(2000);
```

```
}
```

```
void loop() {
```

```
  // put your main code here, to run repeatedly:
```

```
  U();
```

```
  R();
```

```
  R();
```

```
  Fr();
```

```
  L();
```

```
  delay(5000);
```

```
}
```

```
void U()
```

```
{
```

```
  arm.write(front, sp, true);
```

```
  delay(1000);
```

```
  arm.write(back, sp, true);
```

```
  delay(1000);
```

```
  arm.write(front, sp, true);
```

```
  delay(1000);
```

```
  arm.write(hold, sphold, true);
```

```
  delay(1000);
```

```
  cube.write(Mvanticlock, sp, true);
```

```
  delay(1000);
```

```
  arm.write(back, sp, true);
```

```
  cube.write(anticlockpos, sp, true);
```

```
  delay(1000);
```

```
  arm.write(front, sp, true);
```

```
  delay(1000);
```

```
  arm.write(back, sp, true);
```

```
  delay(1000);
```

```
  arm.write(front, sp, true);
```

```

    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
}
void R()
{
    cube.write(clockwisepos, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(hold, sphold, true);
    delay(1000);
    cube.write(Mvcenteranticlock, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    cube.write(center, sp, true);
    delay(1000);

    cube.write(clockwisepos, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);

```

```
delay(1000);
cube.write(center, sp, true);
delay(1000);
arm.write(front, sp, true);
delay(1000);
arm.write(back, sp, true);
delay(1000);
}
```

```
void Fr()
```

```
{
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(hold, sphold, true);
    delay(1000);
    cube.write(Mvanticlock, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    cube.write(anticlockpos, sp, true);
```

```
cube.write(center, sp, true);
delay(1000);
arm.write(front, sp, true);
delay(1000);
```

```

arm.write(back, sp, true);
delay(1000);
cube.write(anticlockpos, sp, true);
delay(1000);
arm.write(front, sp, true);
delay(1000);
arm.write(back, sp, true);
delay(1000);
cube.write(center, sp, true);
delay(1000);
}

```

```

void L()

```

```

{
    cube.write(anticlockpos, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(front, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    delay(1000);
    arm.write(hold, sphold, true);
    delay(1000);
    cube.write(Mvcenterclock, sp, true);
    delay(1000);
    arm.write(back, sp, true);
    cube.write(center, sp, true);
}

```

```
delay(1000);

cube.write(anticlockpos, sp, true);
delay(1000);
arm.write(front, sp, true);
delay(1000);
arm.write(back, sp, true);
delay(1000);
cube.write(center, sp, true);
delay(1000);
arm.write(front, sp, true);
delay(1000);
arm.write(back, sp, true);
delay(1000);
}
```

4.2 Testing Approach

A test approach is the strategy implementation of the project. We have used Proactive test approach technique. In this technique, the testing is started as soon as possible to find and fix defects before the model is completely built. We have used this technique as it helps prevent problems and make development faster, easier, and less aggravating.

4.2.1 Unit Testing

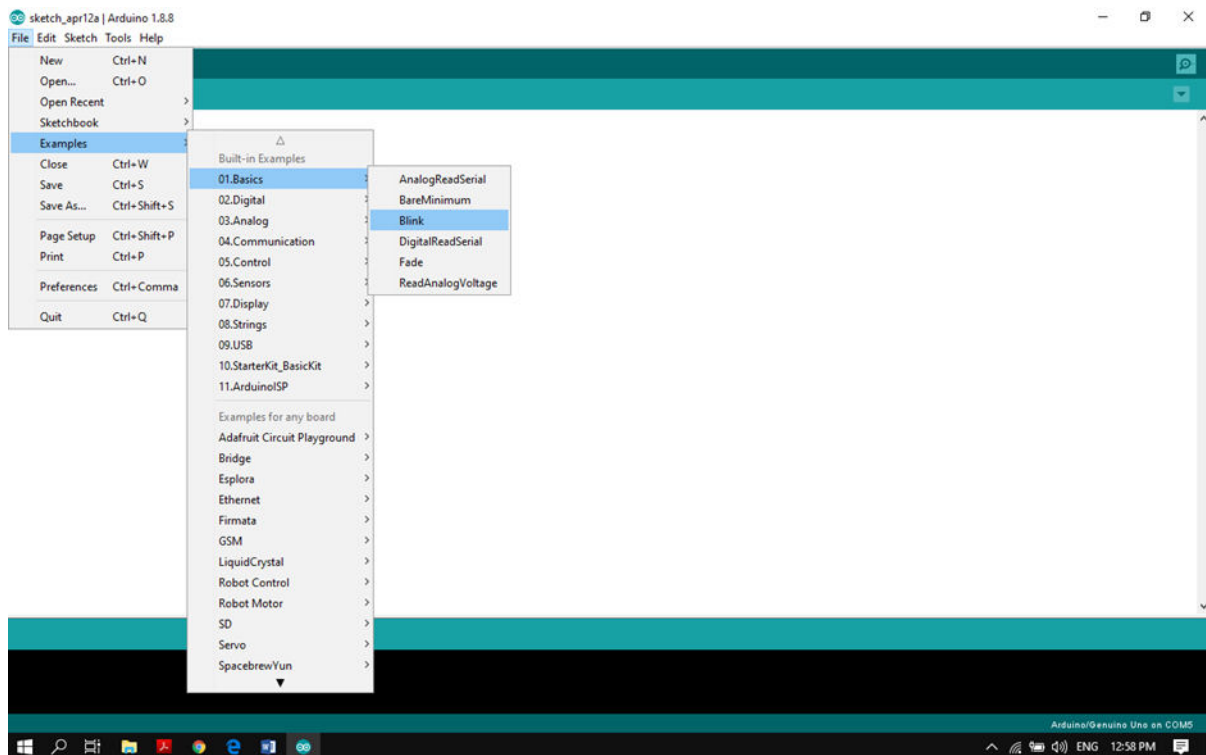
Unit Testing is a technique using which individual modules are tested to determine if there is any issue by the developer himself.

Below is the unit testing of Arduino Uno:

1. Open Arduino IDE Software

2.A blank sketch will open

3.Now go to files >Examples >Basics >Blink



4.This code will open up

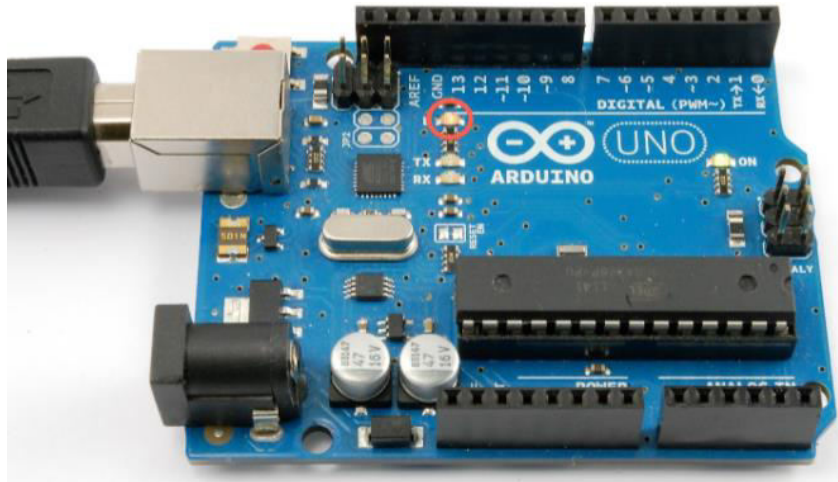
```
void setup() {  
  // initialize digital pin LED_BUILTIN as an output.  
  pinMode(LED_BUILTIN, OUTPUT);}  
  
  // the loop function runs over and over again forever  
  void loop() {  
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)  
    delay(1000);                    // wait for a second  
    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW  
    delay(1000);                    // wait for a second  
  }  
}
```

5.Now verify and upload this sketch to Arduino Uno

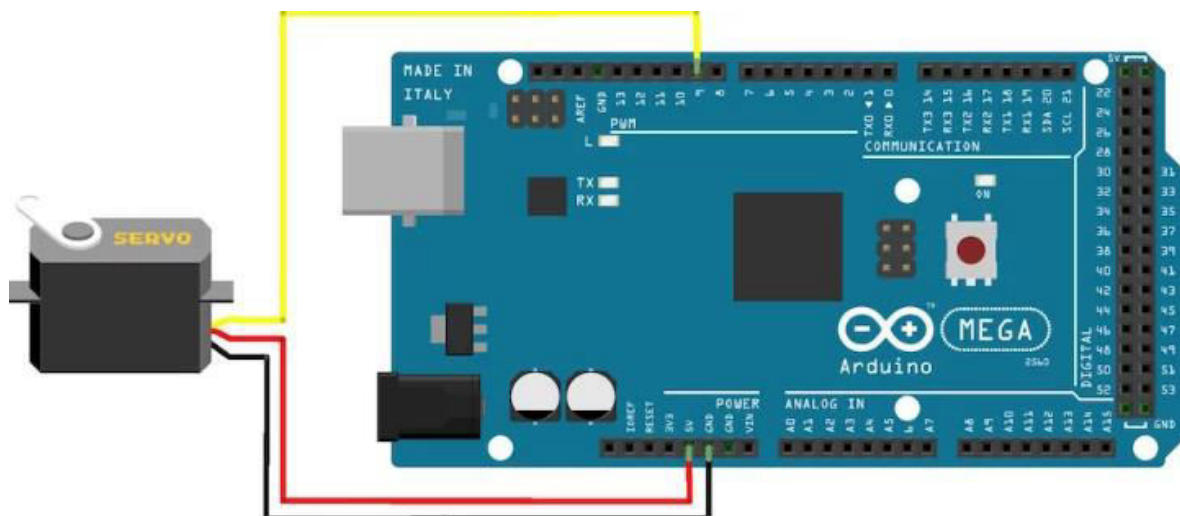
6. Make sure proper port is selected so that it will get uploaded on Arduino or else it won't be uploaded.

7. After Uploading the inbuilt LED of Arduino UNO will blink after 1 sec.

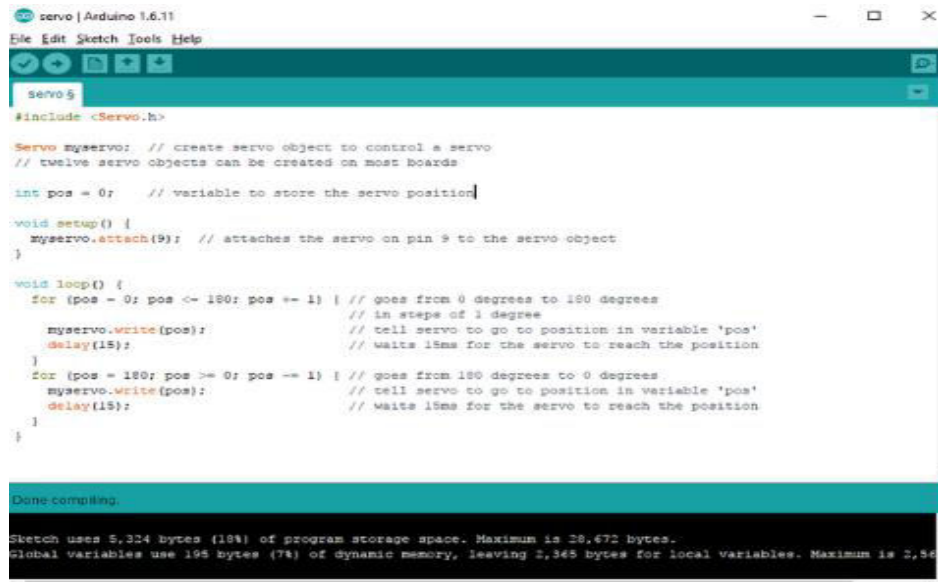
8. the light that is shown in figure below will blink



Servo motor connection



Code for testing servo motors



```
servo | Arduino 1.6.11
File Edit Sketch Tools Help

servo5
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {
  myservo.attach(9); // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos); // tell servo to go to position in variable 'pos'
    delay(15);           // waits 15ms for the servo to reach the position
  }
  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
    // tell servo to go to position in variable 'pos'
    myservo.write(pos); // tell servo to go to position in variable 'pos'
    delay(15);           // waits 15ms for the servo to reach the position
  }
}
```

Done compiling.

Sketch uses 5,324 bytes (18%) of program storage space. Maximum is 28,672 bytes.
Global variables use 192 bytes (7%) of dynamic memory, leaving 2,368 bytes for local variables. Maximum is 2,560 bytes.

- Testing if motors can turn the face side of cube to right
- Testing if motors can turn the face side of cube to left
- Testing if motors can help over flipping the cube to the back
- Testing if motors can listen to the instruction set.
- Testing if camera can recognize colour on the cube
- Testing if camera can recognize 6 different significant colours on the cube.
- Testing if camera can recognize difference in rows
- Test if camera sends data
- Test if camera can process the image after rotation of the cube.
- Test if all circuits are correct
- Test the processor for power drain
- Testing the voltage output at the connection

4.2.2 Integration Testing

In This Testing All the Component Are Combined Together to Make A Final System. After the Integration Testing Has Been Performed on The Component, They Are Readily Available for System Testing.

- Test if instruction set can be worked into servos
- Test if the ports push and pull data
- Test if algorithms can be executed on the servo motors
- Test if algorithms are easily debugged
- Testing if machine can save algorithm list
- Testing the connection with servo motors and arduino uno

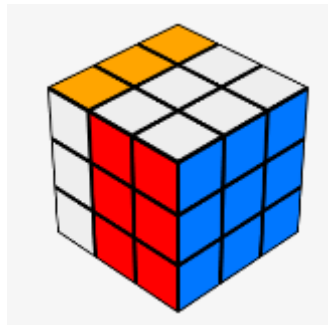
Chapter 5: Results and Discussions

The Rubik's cube solving machine is tested and the built up machine can solve the Rubik's cubes scrambled by the user. Currently, the solving machine runs on limited sets of algorithms. The machine takes more time compared to many speed cubers out there in competitions because the machine is using only two motors, one for flipping and the other for turning. In the near future we plan to use motors on every side to improve the speed by up to 6 times. The algorithm for scrambling and the solution is given below:

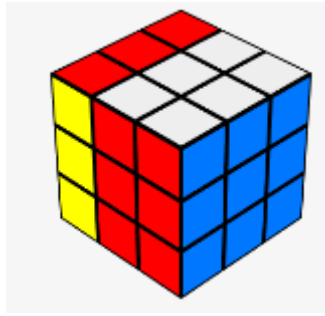
- The algorithm for scrambling the cube is: **L' F' R U'**
- The algorithm for solving the cube is: **U R R F L**

Notations:

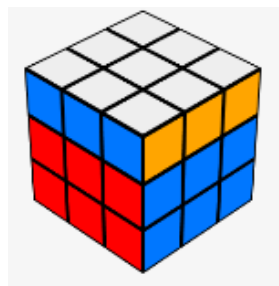
- **L** = Left side clockwise



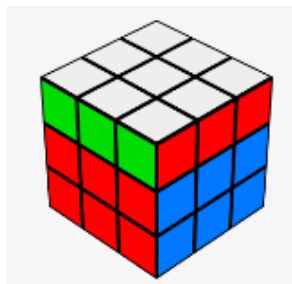
- **L'** = Left side anti-clockwise



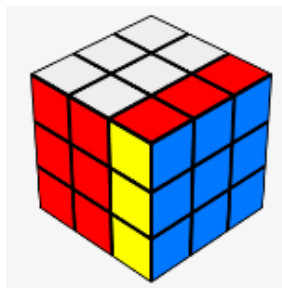
- **U**= Upper side clockwise



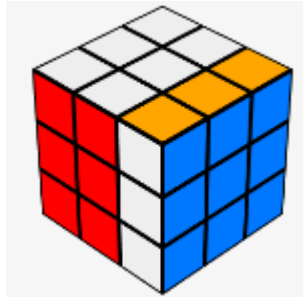
- **U'**=Upper side anti-clockwise



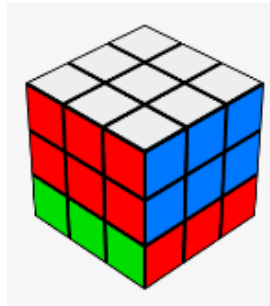
- **R**= Right side clockwise



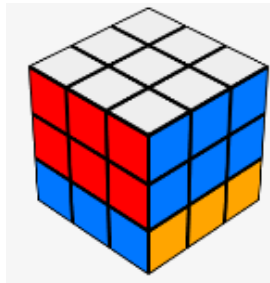
- **R'**= Right side anti- clockwise



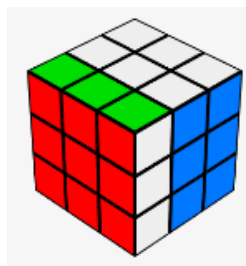
- **D**= Down side clockwise



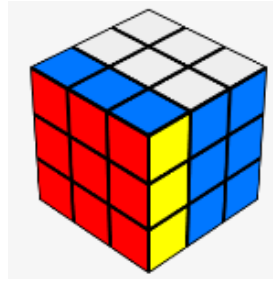
- **D'**= Down side anti- clockwise



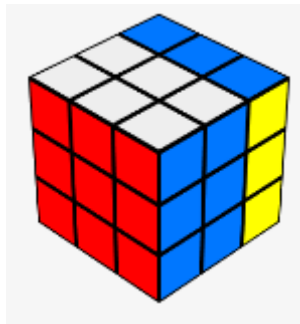
- **F**= Front side clockwise



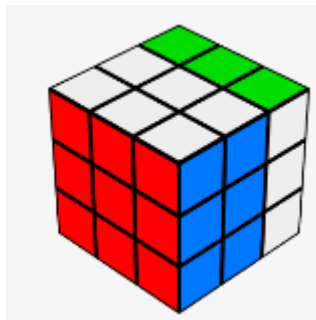
- **F'** = Front side anti- clockwise



- **B** = Back side clockwise



- **B'** = Back side anti- clockwise



Time taken by the machine to solve the cube:

Sr.No.	Scramble	Time Taken
1	U' R' F'	

Chapter 6: Conclusion and Future Work

- **Conclusion**

We have built a machine capable of solving Rubik's cube puzzle. This machine uses a reliable and robust vision, a quick and optimal solution finding algorithm with single armed robot. Our vision is to cope well in various types of environments and still perform with accuracy. This Rubik's cube solving machine is reliable but may not be fully equipped to rival intermediate level speed cubers.

The machine may not be able to solve all the cases given by user as the development of the machine is not fully complete. Currently, only a few cases can be executed on the machine.

We are not hoping to break world records with this machine that we have built, we have always seen this as a chance to explore and compare many different aspects of each component. There have been Rubik's cube solving machines in the past but none of them is as cost effective with such good results.

- **Future Work**

In this section, we discuss the areas that we feel can be improved. We also offer insight into what might help the Rubik's cube solving machine better. Here are a set of points we would like to work upon:

- **Improving the Rotation Speed:** As of now, the speed of rotation is quite low. We sacrificed the rotation speed in favour of stability and accuracy of turning. Problems faced because of high speed turning were vibration and inertia. Vibration can lead to damage to the claws as well as to the arms. Inertia can also throw the cube out of the cube holding stand. To increase the speed of rotations along with no vibrations and inertia we need to build a bigger and

more stable structure. To solve the problem we are facing due to inertia, we can use PID controller adjustments.

- **Generating algorithms for all cases:** The machine must have proper set of logic code which will help solving the cube after scanning. The cube solving machine can currently solve only selective algorithms but in the future we would like to make a development by which cube will be scanned, accordingly algorithm will be generated and then lastly be executed by the machine.
- **Autonomous Scanning:** The start of the solving comes by scanning the cube from all the sides. This scanning is done by user. This should be made completely autonomous. The cube should rotate and flip over the cube and scan it from all the sides. This would also remove the chance of human error.
- **Robust Colour Recognition:** Our current implementation is quite reliable but it becomes less reliable in dim lights. We would like to implement auto brightness feature to this Rubik's cube solving machine.

Chapter 7: References

- 1) <http://ruwix.com/the-rubiks-cube/how-to-solve-the-rubiks-cube-beginners-method/>.
- 2) Tomas Rokicki. God's number is 26 in the quarter-turn metric.
<http://www.cube20.org/qtm/>
- 3) <http://www.jaapsch.net/puzzles/thristle.htm>
- 4) Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search, 2017.
- 5) <http://kociemba.org/math/imptwophase.htm>
- 6) <https://www.instructables.com/id/Rubiks-Cube-Solver/>
- 7) Daniel Kunkle and Gene Cooperman. Twenty-six moves suffice for rubik's cube. In Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation, ISSAC '07, pages 235–242, New York, NY, USA, 2007. ACM.
- 8) Peter Lichodziejewski and Malcolm Heywood. The rubik cube and gp temporal sequence learning: an initial study. In Genetic Programming Theory and Practice VIII, pages 35–54.
- 9) Silviu Radu. Rubik's cube can be solved in 34 quarter turns.
<http://cubezzz.dyndns.org/drupal/?q=node/view/92>, Jul 2007.
- 10) Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In Reinforcement Learning, pages 5–32
- 11) Leon Bottou. From machine learning to machine reasoning, 2011.
- 12) Robert Brunetto and Otakar Trunda. Deep heuristic-learning in the rubik's cube domain: an experimental evaluation. 2017.
- 13) Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In International conference on computers and games, pages 72–83
- 14) <https://octavosystems.com/2017/12/19/osd335x-controls-rubiks-cube-solver/>
- 15) Geoffrey Hinton, Nitsh Srivastava, and Kevin Swersky. Overview of mini-batch gradient descent.
- 16) https://www.speedsolving.com/wiki/index.php/List_of_cube_solving_robots
- 17) Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. The diameter of the rubik's cube group is twenty
- 18) <http://mindcuber.com/mindcub3r/mindcub3r.html>

- 19) Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In European conference on machine learning, pages 282–293.
- 20) <http://www.guinnessworldrecords.com/news/2016/2/new-footage-proves-fastest-robot-to-solve-a-rubiks-cube-record-has-been-broken-y-418225>
- 21) <https://rubiks-cu.be/>
- 22) <https://boingboing.net/2018/03/08/machine-solves-rubiks-cube.html>