A Project on

Shuttle Cock Throwing Bots

Submitted for fulfillment of award of

Bachelor of Technology

In
Electronics and Communication Engineering
&
Electrical and Electronics Engineering

By

Ujjwal Baranwal (ECE-1402731159),
Sandesh Singh (EN-1402721080) &
Siddhant S Goral (EN-1402721105)

Under the Guidance of

(Mr. Gaurav Srivastava)

AJAY KUMAR GARG ENGINEERING COLLEGE, GHAZIABAD
YEAR: 2017-2018

AJAY KUMAR GARG ENGINEERING COLLEGE, GHAZIABAD

CERTIFICATE

This is to certify that Project Report entitled "SHUTTLE COCK THROWING BOTS" which is submitted by ***Ujjwal Baranwal(ECE), Siddhant S Goral(EN) and Sandesh Singh(EN)*** *in the partial fulfillment of requirement for the award of degree of Bachelor of Technology (Electronics and Communication Engineering & Electrical and Electronic Engineering) submitted to Uttar Pradesh Technical University, Lucknow is a record of students' own work carried out under my supervision. The matter in this report has not been submitted to any University or Institution for award of any degree.*

Supervisor :
Mr. Gaurav Srivastava
Asst. Professor
Electrical and Electronics Engineering Deptt.

HOD :
Prof. Ashiv shah
CORE

Date : 12th April, 2018

ACKNOWLEDGEMENT

We take this opportunity to express our deep sense of gratitude and regard to **Mr. Gaurav Srivastava** (Electrical and Electronics Engineering Deptt.), Ajay Kumar Garg Engineering College, Ghaziabad for his continuous encouragement and able guidance, we needed to complete this project.

We would pay sincere gratitude to the Head of the Deptt.(CORE) **Prof. Ashiv Shah** for his precious and enlightening words of wisdom which motivated us throughout our project work.
We are also grateful to Mr. **Amit Sharma** and Mr. **Puneet Raj** for their continuous support and valuable suggestions.

We are also thankful to **Team Robocon AKGEC** to make this project complete within deadline.

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief , it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma by the university or other institute of higher learning, except where due acknowledgement has been made in the text.

Signature:

1.
(Signature)
Ujjwal Baranwal
1402731159

2.
(Signature)
Sandesh Singh
1402721080

3.
(Signature)
Siddhant S Goral
1402721105

TABLE OF CONTENTS

Chapter I: Introduction

1.1 Preface

Robocon (short for Robotic Contest) is organized by Asia-Pacific Broadcasting Union (ABU), a collection of over 20 countries of Asia Pacific Region. NHK, Japan had already been organizing such contests at national level and also became the host of the first ABU Robocon in 2002. Since then, every year one of the member broadcasters hosts this international event.

The broadcasters of each participant country are responsible for conduct of their national contests to select the team which will represent their country in the International Contest. Teams from Engineering and Technological colleges are eligible for participation. Participating Teams are expected to design and fabricate their own robots and organize their teams including an Instructor, Team Leader, Manual Robot Operator and an Automatic Robot Operator.

Doordarshan, the national public service broadcaster organises National Robocon event every year and the winning team gets an opportunity to represent India at the international competition. First Robocon was held in 2002, in which only 3 teams had participated at IIT Kanpur. Going from strength to strength, this number has reached 107 in Indian National Robocon 2018 held at the Badminton Arena of Shree Shiv Chhatrapati Sports Complex, Balewadi, Pune.

Mumbai Kendra of Doordarshan has been organising National Robocon since 2005 with support from co hosts MIT Group of Educational Institutes. The initiative was taken by Mr. Mukesh Sharma, the Director of Doordarshan Kendra Mumbai and Dr. Sunil Karad, Executive Director of MIT Group of Institutions Pune, with approval of Dr. Vishwanath D Karad, Founding Director General of MIT Group Pune.

Every year finals of Robocon India are held on First week of March. The best engineering institutes across the country compete with each other for the honor of representing Indian National team at the International version of the competition for three grueling days. Elimination rounds are held on each days of the event. This routine schedule been fixed in India through mutual consultations of participating colleges and Doordarshan of India. International Contests are held in August every year when the national contests of all the countries have been completed and their representatives selected. The host country has the privilege of fielding two teams. India hosted its first international contest in August 2008 in the MIT Sports grounds, Pune. In 2014 again, India hosted the international contest on August 24. For last four years the National Robotics Championship has been organized in the famous Shree Shiv Chhatrapati Sports Complex, Mahalunge, Balewadi, Pune by a joint collaboration of Doordarshan and Maharashtra Institute of Technology Academy Of Engineering (MITAOE), Alandi, Pune.

Being the leading broadcaster in India, Doordarshan has been instrumental in providing wide coverage and public reach to Robocon.

2.2 Background

Robocon India 2018

The National ABU Robocon 2018 took place in the Shri Shiv Chhatrapati Krida Sankul, Pune on 1-3 March 2018. The contest theme has been declared as "“NÉM CÒN: The Festival Wishing Happiness and Prosperity”. Institute of Technology, Nirma University, Ahmedabad won the National ABU Robocon 2018 Contest. Maharashtra Institute of Technology, Pune, was the first runner-up and K J Somaiya Institute of Engineering and Information Technology, Mumbai, was the second runner up.

Robocon India 2017

The National ABU Robocon 2017 took place in the Shri Shiv Chhatrapati Krida Sankul, Pune on 2-4 March 2017. The contest theme has been declared as "Asobi : The Landing Disc”. College of Engineering, Pune won the National ABU Robocon 2017 Contest. Maharashtra Institute of Technology, Pune, was the first runner-up.

Robocon India 2016

The National ABU Robocon 2016 took place in the Shri Shiv Chhatrapati Krida Sankul, Pune on 3-5 March 2016. The theme for Robocon 2016 has been declared as 'Chai-Yo : Clean Energy Recharging the World'. Vadodara Institute of Engineering, Vadodara won the National ABU Robocon 2016 Contest.

Robocon India 2015

The National ABU Robocon 2015 took place in the Badminton Hall of the Shree Shiv Chhatrapati Sports Complex, Balewadi, Pune on 7 March 2015. The theme for Robocon 2015 has been declared by Televisi Republik Indonesia (TVRI) as "Robomintion: Badmintion Robo-Game".Institute of Technology, Nirma University, Gujarat won the National ABU Robocon 2015 Contest. Krishna Institute Of Engineering And Technology (KIET), Ghaziabad were the first runner ups.

Robocon India 2014

The National ABU Robocon 2014 was held in the Badminton Hall of the Shree Shiv Chhatrapati Sports Complex, Balewadi, Pune on the 6, 7 & 8 March 2014. The theme for Robocon 2014 declared by MITAOE was "A SALUTE TO PARENTHOOD". Institute of Technology, Nirma University, Gujarat won the National Contest while Veermata Jijabai Technological Institute (VJTI), Mumbai were the First Runner-up. Both these Institutes got the chance to represent India in the International Contest. MIT Robocon Tech Team representing Maharashtra Institute of Technology, Pune were the Second Runner-up.

The International ABU Robocon 2014 was held in India.Lạc Hồng University, Vietnam won the International ABU Robocon 2014 held at Balewadi, Pune.

Robocon India 2013

The National ABU Robocon 2013 was held in the Badminton Hall of the Shree Shiv Chhatrapati Sports Complex, Balewadi, Pune. MIT Tech Team from Maharashtra Institute of Technology won the contest and represented India in the international contest.

The International ABU Robocon 2013 was held in Danang, Vietnam. The Theme for Robocon 2013 declared by VTV was "The Green Planet". The International ABU Robocon 2013 was won by Kanazawa Institute of Technology, Japan.

Robocon India 2012

The National ABU Robocon 2012 was held in the Boxing Arena of The Shree Shiv Chhatrapati Sports Complex, Balewadi, Pune. MIT Tech Team from Maharashtra Institute of Technology won the contest and represented India in the international contest.

The Theme for Robocon 2012 declared by Hong Kong, "Peng On Dai Gat". It was won by University of Electronic Science and Technology of China.

Robocon India 2011

The Theme for Robocon 2011 declared by Thailand was Krathong, Lighting Happiness with Friendship.It was won by Institute of Technology, Nirma University and this institute got a chance to represent India at International Robocon 2011. The winner was Dhurakij Pundit University coming from Thailand.

Robocon India 2010

The Theme for Robocon 2010 declared by Egypt was Robo-Pharaohs Build Pyramids. It was won by MIT Robocon Tech Team from MIT Pune.

Robocon India 2009

Theme of Kago, the traditional Japanese palanquin, carried by human beings replaced by robots. It was won by IIT Madras.

Robocon India 2008

The Theme for Robocon2008 declared by India was Govinda, a traditional Indian Deity who used to play earthly games by capturing Butter/Cheese from heads of Gopis.

It was won by Institute of Technology, Nirma University. Runner-up was MIT Robocon Tech Team from MIT Pune. Both team got to represent India in ABU Robocon as host country gets an opportunity to be represented by two teams.

Robocon India 2007

The domestic contest was won by IIT Delhi.

Robocon India 2006

The domestic contest was won by Institute of Technology, Nirma University.

Robocon India 2003

The domestic contest was won by Institute of Technology, Nirma University.

Robocon India 2002

The domestic contest was won by Institute of Technology, Nirma University.

Table 1: List of Themes of International ABUROBOCON from 2002-2018

Year	Host city	Theme	Grand Prix	National Winner
2002	Tokyo, Japan	Reach for the Top of Mt. Fuji	Ho Chi Minh City University of Technology	Institute of Technology, Nirma University
2003	Bangkok, Thailand	Takraw Space Conqueror	Sawangdandin Industrial and Community Education College	Institute of Technology, Nirma University
2004	Seoul, South Korea	Reunion of Separated Lovers, Gyeonwoo & Jiknyeo	Ho Chi Minh City University of Technology	VESIT, Mumbai
2005	Beijing, China	Climb on the Great Wall Light the Holy Fire	University of Tokyo	IIT Bombay
2006	Kuala Lumpur, Malaysia	Building the World's Tallest Twin Tower	Ho Chi Minh City University of Technology	Institute of Technology, Nirma University
2007	Hanoi, Vietnam	Halong Bay Discovery	Xi'an Jiaotong University	IIT, Delhi
2008	Pune, India	Govinda	Xi'an Jiaotong University	Institute of Technology, Nirma University

2009	Tokyo, Japan	Travel Together for the Victory Drums	Harbin Institute of Technology	IIT Madras
2010	Cairo, Egypt	Robo-Pharaohs built pyramids	University of Electronic Science and Technology of China	MIT Robocon Tech Team
2011	Bangkok, Thailand	Loy Krathong, Lightning Happiness with friendship	Dhurakij Pundit University	Institute of Technology, Nirma University
2012	Hong Kong	Peng On Dai Gat (In pursuit of peace and prosperity)	University of Electronic Science and Technology of China	MIT Robocon Tech Team
2013	Vietnam	The Green Planet	Kanazawa Institute of Technology	MIT Robocon Tech Team
2014	Pune, India	A Salute for Parenthood	Lac Hong University	Institute of Technology, Nirma University
2015	Yogyakarta, Indonesia	Badminton	Hung Yen University of Technology and Education	Institute of Technology, Nirma University
2016	Bangkok, Thailand	Clean Energy Recharging the World	Universiti Teknologi Malaysia	Vadodara Institute of Engineering, Kotambi Vadodara
2017	Tokyo, Japan	The Landing Disc	Lac Hong University	College of Engineering, Pune
2018	Vietnam	NEM CON: The festival of prosperity and Happiness	To be determined in August 2018.	Institute of Technology, Nirma University

Chapter II: Literature Review, Requirement Analysis & Feasibility Study

2.1 Requirements Analysis

2.1.1 Theme and Rulebook

"NÉM CÒM"

The Festival Wishing Happiness and Prosperity

Vietnam is on the fast development track. However, the traditional culture is always treasured.

The concept of the ABU Robocon 2018's Theme & Rules is based on an interesting folk game called " Shuttlecock Throwing".

So what is required for Shuttlecock Throwing?

The game consists of an open field with a 15m bamboo trunk planted in the middle.

A bamboo ring is hung on top of the bamboo stick. The ring is covered in yellow and pink papers. Yellow color represents the moon, while pink represents the sun.The heart and soul of this game is the Shuttlecock.The shuttlecock is an object made of cotton balls or filled with rice husks, symbolizing prosperity and happiness.

FIG.1: NEM COM PLAY

The shuttlecock is hung by a string sewn to the center of the Shuttecock. It is decorated with colorful cotton representing the colors of the rainbow. When playing, the player holds the end of the string to swing it clockwise various times before throwing and aims for the center of the ring.
If the shuttlecock makes it through the ring, the player wins. The flying shuttlecock depicts a flying dragon, iconic of human power and the universe.

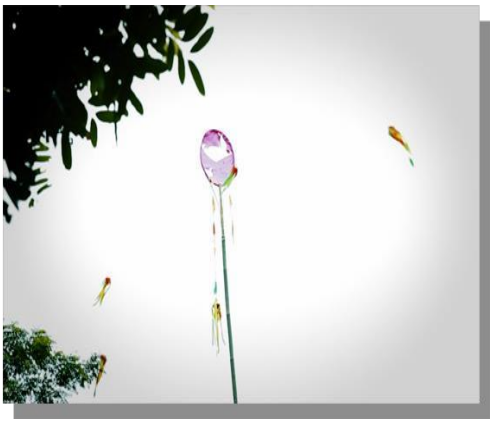


FIG.2: SHUTTLE COCK RING

FIG.3: SHUTTLE COCK USED IN VEITNAAM

FIG.4: SHUTTLE COCK PASSING THROUGH RING

That's why the Shuttlecock Throwing Festival opening is commenced with a ceremony to pray for deitties of the Land and Sky. After the first shuttlecock hits the target, it will be opened. The husks inside it are shared among the people as a wish for a prosperous year.

Nobody knows when the game started. Today, it is not only a folk game for both men and women, where they can meet and find their other halves but also a sport for players to show their skills.
In this spirit, teams of ABU Robocon 2018 are expected to design formidable robots and charismatic shuttlecocks showing their colors.

2.1.2 THE IMPORTANCE OF SAFETY

In ABU Robocon, safety is a top priority. Participants shall give safety precedence over everything at all times, from the robot designing and building stages to taking part in the actual contest. They are also also asked to cooperate fully with the organizer in order to ensure a safe running of the contest for everyone involved, including team members, spectators, officials and staff, as well as for the surrounding enviroment.

Members of all teams are required to wear shoes with rubber soles and helmets when participating the game.

2.1.2 CONTEST RULES

Table 2: Terms and Definitions

Terms	Definitions
Manual Robot	The robot which is operated by operator via wireless or cable connection. Abbreviation: MR.
Automatic Robot	The robot which is able to work independently without any help from an operator. Abbreviation: AR.
Manual Robot Start Zone	An area, from where the manual robot starts the game. Abbreviation: MRSZ.

Manual Robot Area	An area which manual robot and operator are allowed to operate. Abbreviation: MRA.
Automatic Robot Start Zone	An area, from where the Automatic Robot starts the game. Abbreviation: ARSZ.
No Contact Area	An area which robots cannot come in contact with. Robots are able to enter the space above. Abbreviation: NC.
Throwing Area	An area from which Automatic Robot throws Shuttlecock. Throwing Area consists of three zones: The first throwing zone: Abbreviation: TZ1. The second throwing zone: Abbreviation: TZ2. The third throwing zone: Abbreviation: TZ3.
Loading Zone	Areas where teams allocate Shuttlecock or Shuttlecock Rack before the game begins. Abbreviation: LZ.
Ring	The circle attached vertically on top of Ring Tree. The Automatic Robot will throw the Shuttlecock through the Ring. Ring includes 2 types: Normal Ring and Golden Ring.
Ring Tree	The pole attached vertically against the game field, with the Ring on top.
Golden Cup	The Cup that receives the Golden Shuttlecock threw by Automatic Robot. Abbreviation: GC.
Shuttlecock	The object is used during the game, with a sphere shape or other shapes. Shuttlecock is designed and made by teams with a team name or symbol or logo so that which shuttlecock belongs to which team. Shuttlecock is made of soft material (natural fiber or synthetic fiber) and is attached by Tail and Fringe. Shuttlecock has 2 types: Normal Shuttlecock: ten (10) in whatever colors, can be single color (except for gold), or multi colors. Golden Shuttlecock: five (5) in gold color.
Tail	The component attached to the Shuttlecock which is used to keep and throw the Shuttlecock . Tail is made of soft material (natural fiber or synthetic fiber), not elastic and have different colors.
Fringe	The decoration component, freely attached to different po Shuttlecock. Fringe is made of soft material (natural fiber or synthetic fiber) in different colours with minimum 3 colors.
Keeping point	The point on the Tail, created from one or more kinks or tie the tail forming circle (without additional materials).
Rack	Rack is used to place or hang the Shuttlecock, designed and made by teams. Team can make as many racks as they wish and there is no regulation in dimension. All the Shuttlecock Racks have to be fit inside the Loading Zone.

	Rack can be used or not used by teams.

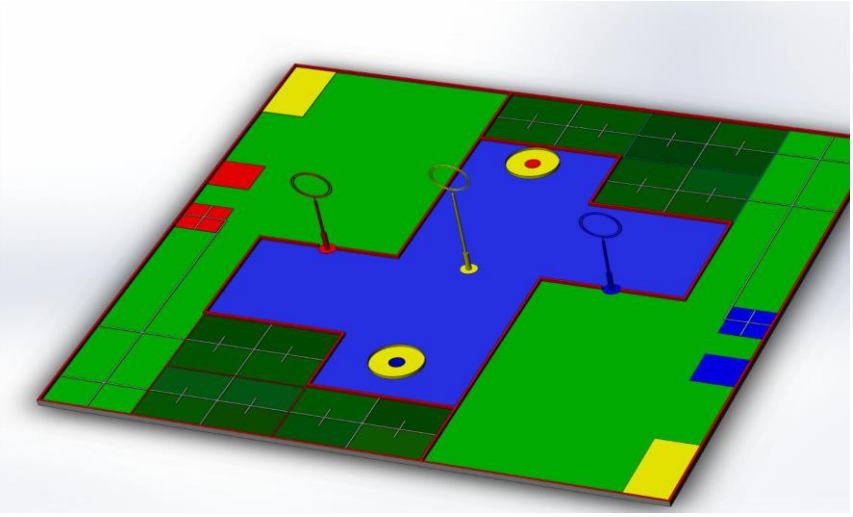


FIG.5: MAIN ARENA

2.1.3 Contest Outlines

- A game between two teams takes place within 3 minutes. Each team has two robot of:
  - One (1) manual robot and one (1) automatic robot or
  - Two (2) automatic robots.

Only one (1) automatic robot is allowed to throw Shuttlecock.

- Game filed is divided into 3 areas, including fields for the 2 teams and NC area (See figure 1.1).
- A team field consists of: Start Zone, Loading Zone, Throwing Zone, Manual Robot Area, and Automatic Robot Area.
- NC Area is placed with Ring Trees, Normal Rings, Golden Ring and Golden Cup.
- Before a game starts, ten (10) Normal Shuttlecocks and five (5) Golden Shuttlecocks are placed inside Loading Zone.
- When a game starts, Manual Robot will pick Normal Shuttlecocks and handle it to Automatic Robot.
- After receiving the Normal Shuttlecock, Automatic Robot will move into TZ1, TZ2 and throw the shuttlecock at the Normal Ring. If the shuttlecock goes through the ring successfully, points will be given.
- Manual Robot can go and pick up the Golden Shuttlecock only after at least one Normal Shuttlecock thrown from each TZ1 and TZ2 went through the Normal Ring successfully.
- After receiving Golden Shuttlecock from Manual Robot, Automatic Robot can move to TZ3 and throw the Golden Shuttlecock at the Golden Ring. If the Golden Shuttlecock goes

through the Golden Ring successfully, points will be given.

- When Golden Shuttlecock is thrown through the Golden Ring and then is landed on the Golden Cup, that team will gain the victory and the match will be finished. This victory is called ***"Rongbay"*** ("Flying Dragon").
- If neither team reaches ***"Rong bay"***, and the game time of three (3) minutes passes, the game shall end. The winner will be decided by who has the higher score at the said end of the game. In case two (2) teams get the same scores, the winner will be determined under Article 3.7.

2.1.4 Game Procedure

- Set up

a. Before a game, there is one (1) minute for each team to set up, moving the robot and shuttlecocks or the rack (if any) to the regulated position.

b. Three (3) team members and up to three (3) pit crew members shall be allowed to participate in the set-up.

c. The set-up time will start right after the signal from referees and will end right after one (1) minute.

d. If a team fails to complete its set-up within the given one (1) minute, it may resume set-up after the start of the game by obtaining permission from the referee.

- Start of the Game

a. When the set-up time is over, referees will signal to start the game.

b. Teams that complete their set-up after the start of the game shall obtain permission from the referee at that moment to commence moving their robots.

c. When the game starts, robots have to be abided by the Article 7.5.

- Team members during the game

a. Only members who controlling Manual Robot are allowed to move inside MRA. If moving to another zones, they must be permitted by referees.

b. Other members have to stand inside the pre-assigned area out of the game filed.

c. Team members are not allowed to touch robots without referees'permission.

- Shuttlecock Loading, Handling/ Receiving and Throwing

a. Loading the Shuttlecock

Manual Robot picks the Shuttlecock in the LZ area.

In each time, Manual Robot is allowed to load one (1) or more shuttlecocks or the racks (if any).

Manual Robot can go and pick up the Golden Shuttlecock only after at least one shuttlecock thrown

from each TZ1 and TZ2 went through the Normal Ring successfully.

b. Handling/Receiving the Shuttlecock

The Manual Robot must pass Automatic Robot one Normal Shuttlecock at a time. Only after the Automatic Robot has finished throwing the Normal Shuttlecock, Manual Robot can pass on the next Normal Shuttlecock.

Robots are allowed to handle and receive one (1) or more Golden Shuttlecock or the rack (if any) each time.

Handling and Receiving the Shuttlecock is considered successful only when having enough 4 factors as follows:

- Automatic Robot successfully keeps or hold Shuttlecock or Tail or Fringe.
- Shuttlecock is not allowed to touch the game field.
- None of the Manual Robot part makes any contact with shuttlecock.
- None of the Manual Robot part makes any contact with Automatic Robot.

During the process of handling and receiving shuttlecock, the Manual Robot is allowed to make contact with Automatic Robot. But, the Manual Robot is not allowed to send any signal to or control the Automatic Robot.

However, the Automatic Robot is allowed to control itself using sensors and so on.

- Throwing the Shuttlecock

Automatic Robot is allowed to choose the order of throwing Normal Shuttlecock in TZ1, TZ2.

Only when throwing successfully at least one (1) Normal Shuttlecock in TZ1 and one (1) Normal Shuttlecock in TZ2, the Automatic Robot is allowed to take the Golden Shuttlecock.

The Automatic Robot is allowed to carry multiple shuttlecocks if they are Golden Shuttlecocks. The Automatic Robot can throw the maximum of five (5) Golden Shuttlecocks in TZ3.

When the Automatic Robot fails to throw the Golden Shuttlecock, it is allowed to come back and throw Normal Shuttlecock in TZ1 and TZ2 for scoring.

For each throwing:

- The Automatic Robot is allowed to throw one (1) Normal Shuttlecock at a time.
- The Automatic Robot is allowed to throw one (1) or more Golden Shuttlecocks at a time.
- The Automatic Robot has to hold the Tail to throw from the Keeping point or a further position from the shuttlecock. The Automatic Robot is not allowed to hold the Shuttlecock. Distance between the Keeping point and Shuttlecock is 250 milimeters, at minimum.

When throwing Normal Shuttlecock, the Automatic Robot has to entirely stay inside TZ1, TZ2. The vertical shape of robot is not allowed to fall on the border line between MRA and TZ1, TZ2.

When throwing Golden Shuttlecock, the Automatic Robot has to entirely stay inside TZ3. The vertical shape of robot is not allowed to fall on the border line between TZ2 and TZ3.

- Picking up the Shuttlecock

The Manual Robot is allowed to pick falling Normal Shuttlecocks in the MRA and handle them to the Automatic Robot.

Falling Normal Shuttlecocks and Golden Shuttlecocks in the NC are not allowed to be picked.

The Normal Shuttlecocks fell outside of the game field can be picked up and placed in the LZ by the team members without asking for a retry. In case of Golden Shuttlecock, team members could pick it up without re-using.

- If the team asked for retry and granted by referee:
  - Team members can pick up Normal Shuttlecocks that fell in TZ1, TZ2 and TZ3 and place them in LZ.
  - Team members can pick up Golden Shuttlecocks that fell in TZ1, TZ2 and TZ3. However, they will not be re-used.
  - The Automatic Robot must restart from ARSZ.
  - Score is calculated as follows:
  - A success of one (1) shuttlecock from the Handle and Receive between Manual Robot and Automatic Robot: **One point**
  - A success of throwing one (1) Normal Shuttlecock by the Automatic Robot through the Normal Ring in TZ1: **10 points**
  - A success of throwing one (1) Normal Shuttlecock by the Automatic Robot through the Normal Ring in TZ2: **15 points**
  - A success of throwing one (1) Golden Shuttlecock by the Automatic Robot through the Golden Ring in TZ3 and not landing on the Golden Cup: **30 points**
  - A success of throwing one (1) Golden Shuttlecock by the Automatic Robot through the Golden Ring and landing on the Golden Cup: Winning the **"Rong bay"**.
- End of the Game

The game ends when:

  - A team wins the **"Rong Bay"**, or
  - The game time of three (3) minutes is over.

- Deciding the Winner

A Winning Team is determined as follows::

  - The **"Rong Bay"** Winner.
  - A team whose best score.
  - In case 2 teams have the same points:

a. Team of higher scores in TZ3.

b. Team of higher scores in TZ2.

c. Team of higher scores in TZ1.

d. Team with less total weight of the robots.

e. Determination by Judge Committee.

- Retries

a. There is no limitation for retry. A retry is considered by the rule with approval from referee.

b. Before a retry takes place, Teams have to bring their robots back to start zone.

c. The Shuttlecocks or Racks on Robot will stay in the same position even during the retry. If the team wishes to rearrange the shuttlecocks or racks (if any), they have to place them in LZ first and then pick them up again.

2.1.5 Violations

The team who commits the following shall be deemed to be in violation of the rules and subject to a mandatory retry:

- Manual Robot enters the opponent's game field (including space area);
- A team member touches the robot without referee permission;
- Manual Robot enters throwing zones or no contact area.
- Any other acts deemed to be an infringement on the rules

2.1.6 Disqualifications

If a team is deemed to have committed the following intentionally, the team shall be disqualified for that game.

- Any acts that pose danger to the game field, its surroundings, the robots, and/or people.
- Any other act that goes against the spirit of fair play.
- Any act of disobedience against a referee's warning.

2.1.7 Teams

- One (1) representing team from each country or region shall participate in ABU Robocon 2018. As the host country, Vietnam shall be represented by two (2) teams.
- A team consists of three (3) team members who are students and one instructor, who all belong to the same university/college/polytechnic.
- Besides three (3) team members, three members are allowed to be registered as the pit crew. The members of the pit crew shall also be students from the same university/college/polytechnic as those in 6.2. The .pit crew may assist in the work in the pit area, in carrying the robot from the pit area to the game field.

- Graduate students cannot participate in ABU Robocon 2018.

2.1.8 Robot

- Each team is allowed to bring one (01) Manual Robot and one (01) Automatic Robot to participate in the contest. In case of using two (02) Automatic Robot, the robot starting from MRSZ shall be considered as Manual Robot.
- The robot must not split into parts during the game.
- The robot must be hand-built by students from the same university/college/polytechnic.
- Wireless/infrared/laser/super sonic communications between Manual Robot and Automatic Robot is not allowed.
- Robot size

The robot (including the controller and cable) must fit into the Start Zone (1000mm x 1000mm x 1000mm). Throughout the game, the robot together with any rack that will be attached to the robot after shuttlecocks loading shall not exceed regulated 1500mm length x 1500mm width x 1800mm height.

- Robot weight
  - The total weight of each robot, including Racks, battery, controller, cables and any other equipment that the team brings for use in the game must not exceed 25kg.
  - Back-up batteries (of the same type as that originally installed in the robot) are exempt.
- Power source of the robot
  - Each team shall prepare its own power source.
  - All batteries used in the robot, controller, and any other device used during the game shall not exceed 24V.
  - The maximum voltage within the circuits shall not exceed 42V.
  - Teams using compressed air must use either a container made for the purpose, or a plastic soda bottle in pristine condition that is prepared appropriately. Air pressure must not exceed 600kPa.
  - Any power source deemed dangerous may be banned from use.

2.1.9 Safety

a. Emergency stop buttons must be built on all robots.

b. Robots must be designed and built so as to pose danger to no one, including the team, the opposing team, the people in the surroundings, and the venue.

c. The use of the following are prohibited:

- Accumulator, lead-acid batteries (including colloidal), power sources that involve flames and/or high temperatures,anything that may contaminate the game field, as well as anything that may cause the robots to break down and/or create a situation that hinders the procession of the contest.
- If laser is used, it shall be class 2 or less. Care must be take not to damage the eyes of anyone in the venue.

2.1.10 Others

a. For anything not mentioned in this Rule Book, the teams are required to obey the decisions of the organisers and referees.

b. Dimensions, weights, etc. of the game field described in this Rule Book have a margin of error of plus or minus 5% unless otherwise stated.

c. Questions must be sent in through the contact page on the ABU Robocon 2018 Ninh Binh – Vietnam http://www.aburobocon2018.vtv.vn.

d. Any additions/changes to the rules will also be posted on the ABU Robocon 2018 official website http://www.aburobocon2018.vtv.vn. Rule changes will be updated on the official website http://www.aburobocon2018.vtv.vn

e. For anything that has to do with the safety of the robots and/or the people in the vicinity, the teams are to obey the directions from the organisers and referees.

f. Transporting of the robots

- The organiser shall arrange for the transport of the robots participating in the ABU Asia-Pacific Robot Contest 2018 Ninh Binh - Vietnam. Details will be given separately to each representing team.

- For ABU Robocon 2018 Ninh Binh - Vietnam, the robot must fit inside a single box of length 1000mm x width 1600mm x height 1400mm.

Note: Participants should take note in designing and building their robots to accommodate for this box size; its dimensions are smaller than that allowed for the robots during the contest.

2.1.11 Shuttlecock Award

ABU Robocon Contest 2018 Ninh Binh Vietnam shall award the prize of the best Shuttlecock to the team designing the most beautiful and impressive shuttlecock.

2.2 Literature Review

A rapid evolving society with swift development in the field of information technology and globalization requires a graduated students to possess tools and skills which enable him or her to function successfully in ever-changing environments, take decisions, handle responsibilities and work in teams. Thus, students need to acquire not only a sound base of knowledge specific to their domain of expertise, but also a number of managerial, administrative and executional skills.

They must be capable of applying the theoretical knowledge gained through classroom studies in practical applications. Alternative learning methods and environments like project-based learning are playing an increasingly important role in shaping the students for their future professional life.

Project-based education is a learning environment congruent with the principles of student- and competence-centered vision. It can be seen as a pedagogical innovation which integrates theory and

practice by means of problem solving of working life issues. A number of experimental and theoretical investigations have been done on the application, development, and benefits of project-based learning in education. Integrated group-work leads to improvement in the attitude towards working with others and academic-performance of students. The role of the student changes from that of a consumer in a traditional classroom based setup to a more involved role of an actor in project-based learning. Excellent results in teaching basic electrical measurement at the fourth year of university studies have been achieved by Eugene due to increased maturity of students at this stage.

According to current literature, traditional assessment methods are considered to be less appropriate to measure the level of understanding and skills students acquire by project-based learning. An assessment format that relies on many different evaluation aspects and fits the characteristics of the particular learning environment is required. Here, a project-based education through open-competition is highlighted. In project-based education through open-competition, say, RoboCon, the goal is defined and final outcome is judged by people from all walks of life instead of a group of examiners. This emulates real life engineering and product development.

Each year, the students at **Ajay Kumar Garg Engineering College, Ghaziabad** develop robots over a period of seven months for the Annual National RoboCon. The robots developed by them are appraised through peer review, expert reviews and through the competition between the participating colleges. After the competition, further development of robots and assessment of the students is done through a Mini-project. In the Mini-project the students are encouraged to theoretically study and optimize the robots. They are motivated to extend the work they have done while preparing for the RoboCon and modify the robots for industrial and other practical applications. The students are then evaluated by a group of examiners (professors) on the basis of how the students could correlate these projects with their lectures; understand the importance of design, production drawings, circuit diagram etc.

2.3 Feasibility Study

The National level. Robocon competition puts every year a new challenge in front of the teams participating in the event across the nation. After the theme is out , the making of the robot goes through various steps involving various proposed techniques, repeated prototypes and different strategies. Every theme has some challenges and the first work to be done is to find out those hidden challenges and adopt a proper methodologies that leads to the making of the perfectly working machine ‘The Robot’.

The theme revolves around the concept of 'play' which is the basic philosophy behind Robocon. The theme for 2018 is "Shuttlecock throwing" where two teams compete with each other to score points by throwing shuttlecocks through rings on poles on various heights and from different distances. The scoring is done on various parameters specified in the rule book. A match will be contested between two teams, Red and Blue, lasting for 3 minutes. The score at the end of 3 minutes will be compared and the winner will be declared.

After going through the rulebook and properly analyzing the theme we started watching different videos and collecting related information that helps a team in beating the challenges. After going

through all this our first focus was to make an effective drive train so that we can cover the distance to loading area in a time as minimum as possible. We make drive train with the help of omni wheels whose special orientations helped us in beating all the challenges of moving drive train and with this we have completed our first task. After this we started searching for various mechanism that can be used for different functions of the robot.

According to this year's theme, 2 teams will compete with each other in 3 minutes. Each team is allowed to have 2 automatic robots or 1 automatic and 1 manual robots (2 robots in total).The aim is to have one automatic robot from one team to throw a shuttlecocks pass through the provided rings.

The Manual Bot is used for loading of shuttlecock while the autonomous bot is used to throw the shuttle from the rings.

The two main mechanism for two bots to be taken into account were loading mechanism and throwing mechanism. We focus on getting proper throwing mechanism, to get a perfectly working throwing mechanism we again started watching different videos, doing brainstorming, adopting different mechanism and testing them by making different prototypes. With each prototype, we came to know about our faults and with each new prototype we solve our fault and beat the challenge. The first prototype we took over was based on pneumatics. Then arise the need for an appropriate Piston for it. We have tried pistons like 7' or 8' pencil piston or piston with large bore diameter. With the help of this we have many prototypes of throwing mechanism using them at different angles and different positions. Now we started experimenting with our prototype and the spectrum of experimentation narrowed down to finding appropriate location and angle and a search for the strategy. We have attain all these facts with the help of practices, observations and calculations. For proper result, we have to minimize the nmber of variables. By try and error, we different position and angles suitable for arrangements.

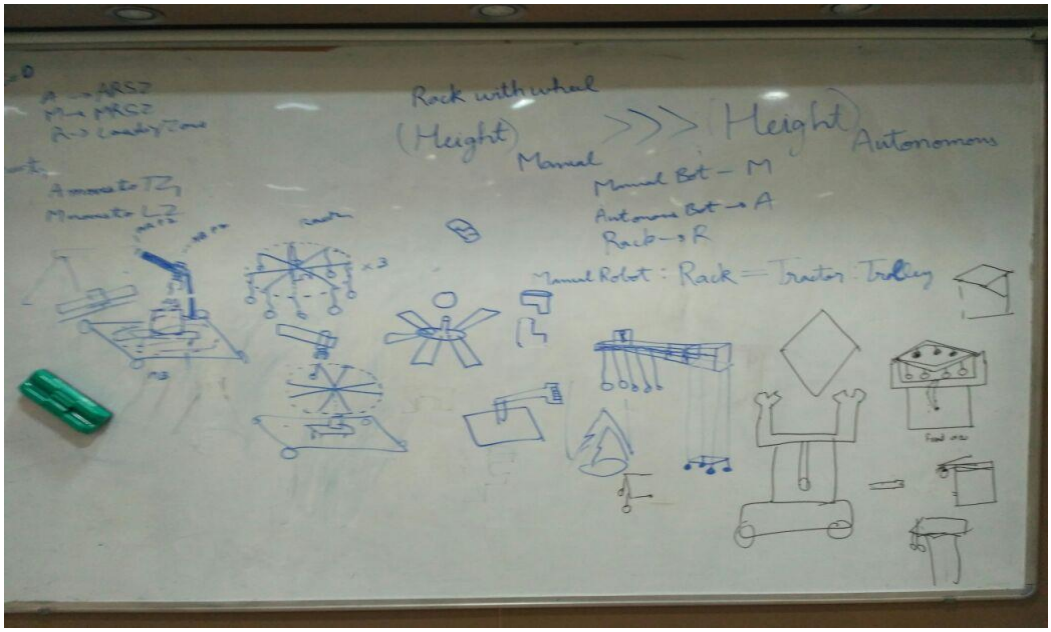


FIG.6: BRAIN STROMING TIME

One major thing that hadn't paid much attention for quite a long time was proper shuttle cock. We should have fixed one proper shuttle according to given rules. This had taken quite a very long time to have a proper shuttle. We have tried different shaped and studied its kinematics.

Chapter III: System Analysis, Design and Testing

3.1 System Analysis

The complete machine operation relied on the activity of the defined "Shuttlecock". Thus it becomes extensively important to understand the basic structure of Shuttlecock to achieve a design of Shuttlecock and a throwing mechanism.

Understanding shuttle cock Structure

1- Desired Mass = 60 – 100 gm
2- Limit over any dimension of Shuttlecock head = (Dia) greater than or equal to 12 cm
3- Minimum Length of Keeping Point = 25 cm
4- Minimum numbers of compulsory fringes = 3
5- Material : Soft and Elastics
6- String cannot be Elastic
7- Atleast two different colors of Shuttlecock compulsory required in the Match

Relation between Machine Design and Shuttlecock

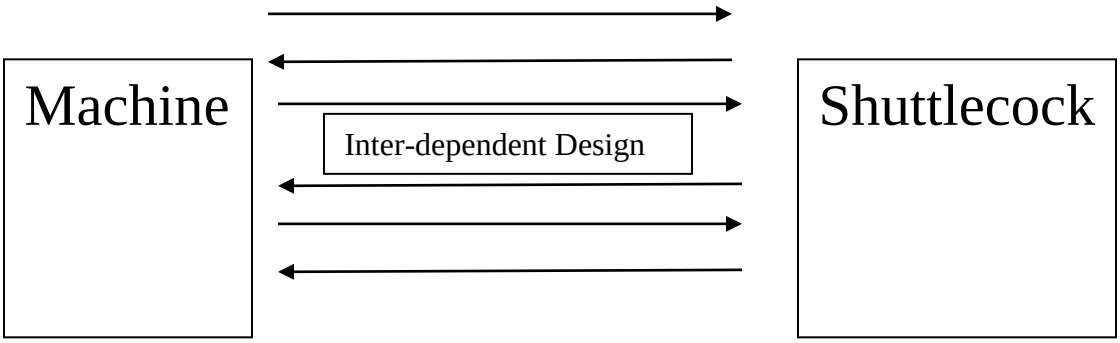


FIG.7: INTERDEPENDENCY OF ROBOT DESIGN WITH SHUTTLE COCK

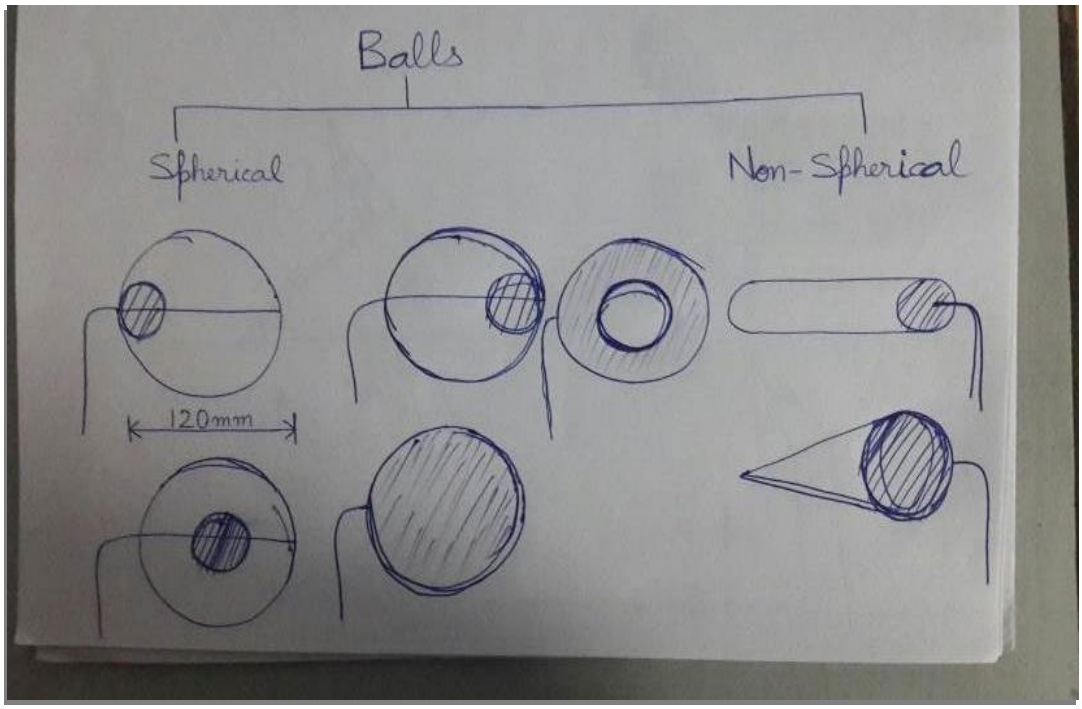


FIG.8: MASS DEPENDENCY OF SHUTTLE COCK

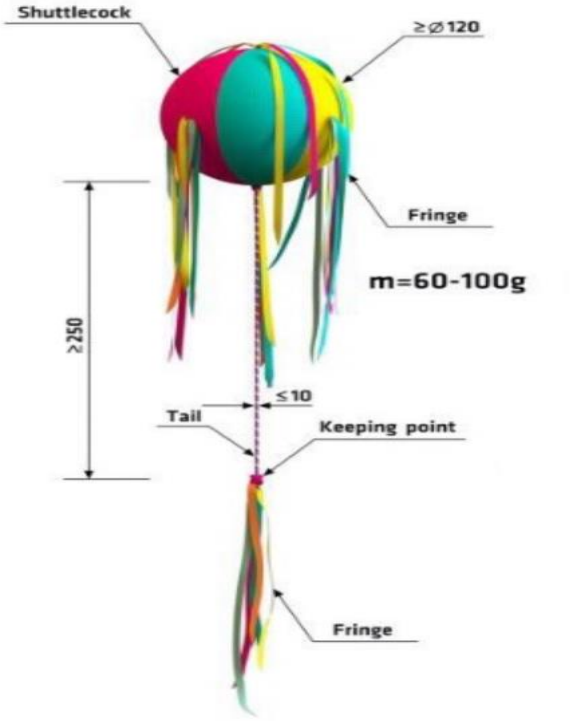


FIG.9: SHUTTLE COCK SPECIFICATION

Phases of Development of Shuttlecock

The Shuttlecock design phases comprises of implementation and testing of following seven types of Shuttlecocks.

Initial Version of Shuttlecock

In this model we wrapped wool over thermacol material to attain the desired mass and volume.

FIG.10: FEW SHUTTLE COCK DESIGN

Final Version of Shuttlecock

Final version of Shuttlecock posses following features :

1- Uniform Mass distribution all over the head
2- Material employed is natural cotton fibre
3- Has a covering of Synthetic clothes
4- Shape *not precisely* Sphere but *tend to be* sphere
5- Tail : Plait winding of Synthetic Fibre
6- Possessed nine Fringes

PROJECT FROM THE PAST:
Dazzling Spheres by Anna Diamond (Threads no. 50, page 52)

Just in time for the holidays, then and now, Anna Diamond showed *Threads* readers how to create beautiful decorations with these Japanese-style embroidered Styrofoam balls using a technique called ito-mari, or temari. Take a look back at the original article for even more detail and information on this intriguing technique.

The guide thread

1. Thread a crewel needle with gold thread long enough to go around ball at least 6 times. Knot end and secure thread at A.
2. Carry thread to C, laying it beside D; bring it back to A, passing beside H. From A, take thread around ball, laying it beside next group of pins. Continue, until ball is divided into 8 segments. Make two small stitches at A to secure thread.
3. Take 2-3 stitches under wrapping to bring needle out at D; be careful that no gold thread shows.
4. Carry thread around ball at equator, and secure with a few small stitches. Fasten off by taking a long stitch through wrapping away from pin. Cut thread close to ball. Do not remove pins; they prevent guide lines from distorting as you embroider.

Wind sewing thread onto the Styrofoam ball as randomly as possible to cover it completely.

Plotting the Pattern

1. Pin a 12 in. long strip of paper to the ball (A). Wrap around widest part of ball, reaching pin (A), fold end back to measure circumference (B).
2. Fold strip (A, B) in half, holding flat against ball. Rotate to find bottom opposite A. Insert pin C.
3. Fold in half again to find "equator" (D). Rotate strip around ball placing 8 pins along equator.
4. Remove strip; replace pin A. Fold into 8 equal divisions. Cut notch at each fold.
5. Lay strip against equator pins. Overlap first and last notches. Place a pin at each notch.

Equator pins

Stitching the square star

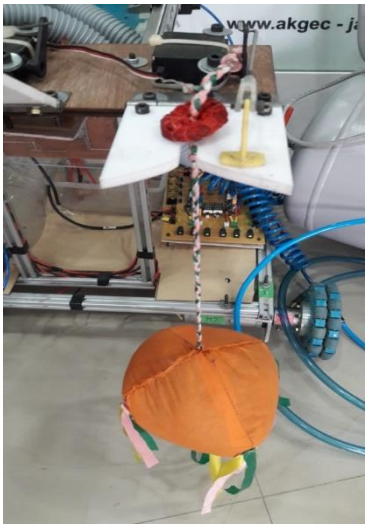
1. Using notched strip, mark points M, N, P, and Q halfway to equator along 4 guidelines.
2. Thread crewel needle with embroidery cotton, knotted at end. Bring it out to left of and just below M. Lay it to the right of guide thread up to P.
3. Turn ball 180°. Stitch ¼-in. from right to left below P, going under guide thread, through wrapping. Lay thread parallel and to right of P, M.
4. Turn ball 180°; take a stitch right to left just below stitch at M. Repeat steps 3 and 4; make each stitch a bit further from pin. Use 2nd needle and thread to repeat on N, Q. Continue with different colors. Repeat steps 1-4 on other side of ball.
5. Using a finer gold thread, such as Fil or DMC à broder, work 4 star motifs centered on pins E, G, J, and L in the unembroidered spaces. Stitch across E working clockwise until a star fills the area. Secure the spokes at the center with several small stitches.

To hang the completed ball, you can attach a cord at one gold star and a tassel on the star opposite it. (Photos by Susan Kahn)

74 THREADS

FIG.11: THE FINAL VERSION IS INSPIRED FROM THE FOLLOWING DOCUMENT AVAILABLE ON INTERNET

The final version of Shuttlecocks has two different colours, Orange and Blue, as follows:



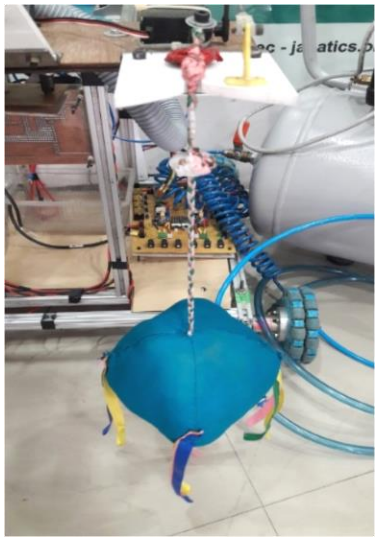


FIG.11: FINAL SHUTTLE DESIGN a) GOLDEN SHUTTLE COCK and b) NORMAL SHUTTLE COCK

Base Cad

The base CAD for the two different bots is different in size, both employing **Modified Holonomic Drive.** Basic Features of Modified Holonomic Drive is as follows

1- Comprises four pairs of Motors and wheels at two perpendicular axes
2- Wheels employed are Omni wheels
3- Two motors are responsible for motion in a given direction
4- The resultant motion is the vector sum of the actuated motion in two orthogonal directions

Chasis employed 20 X 20 Aluminium Profiles.

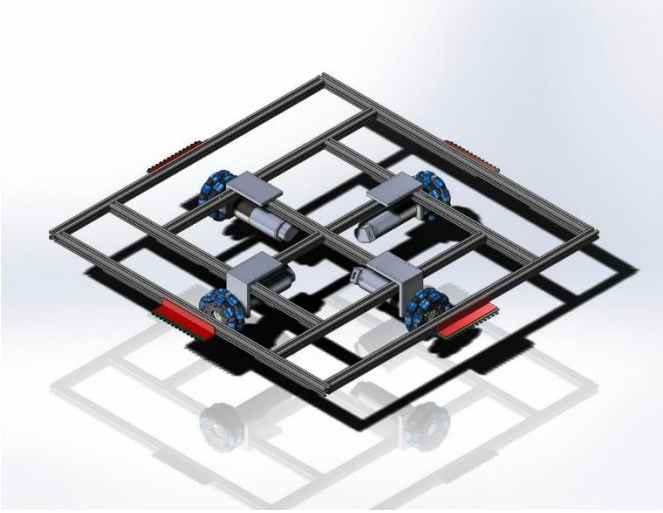


FIG.12: ISOMETRIC VIEW OF BASE CAD OF DRIVE

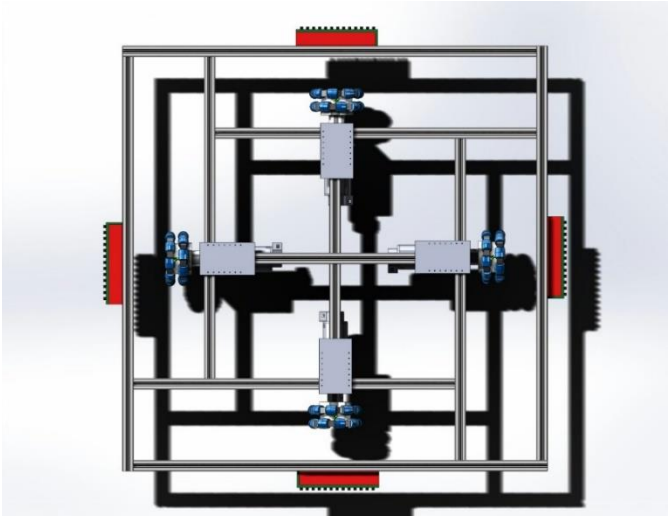


FIG.13: TOP VIEW OF BASE CAD OF DRIVE

Shuttlecock lift and drop Mechanism

The handling of Shuttlecock is done using comb-like structure having definite spacings between it's fingers to accommodate the tail string but object the knot at the keeping point.
The comb-like structure with it's dimensions is as follows :-

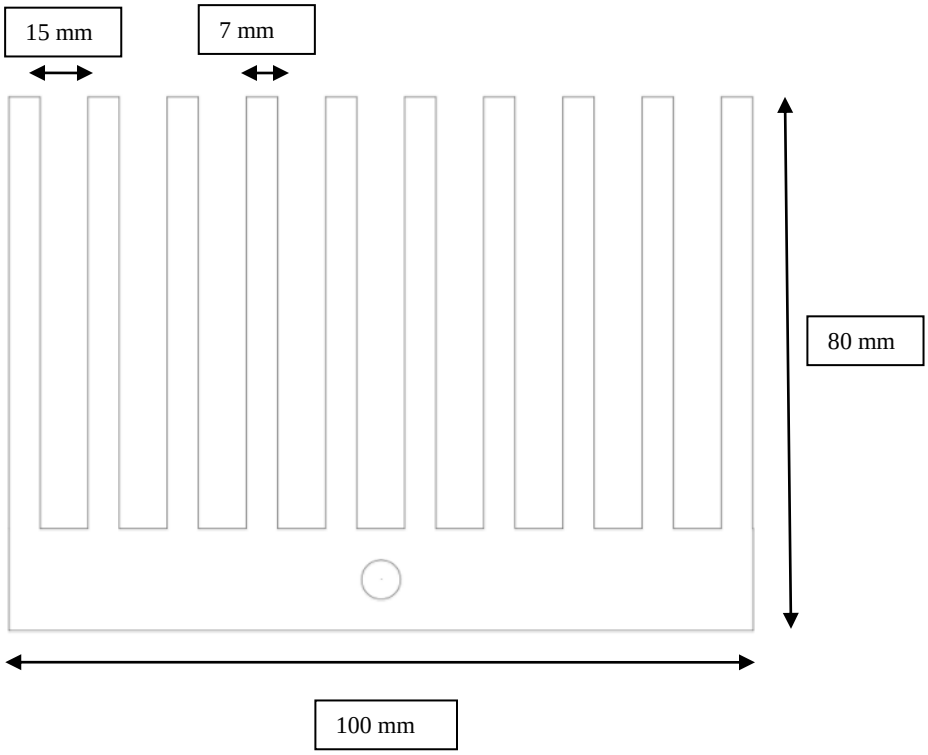


FIG.14: COMB DESIGN

The **lift-hold-drop process** of the arm was controlled using Horns and Clevis.
This mechanism employing Horns and Clevis controlled the angle of the comb with the horizontal. In case of positive angle the Shuttlecock is in Hold State, in case of negative to positive transition the Shuttlecock is picked up, while in case of positive to negative transition the Shuttlecock is droped down.

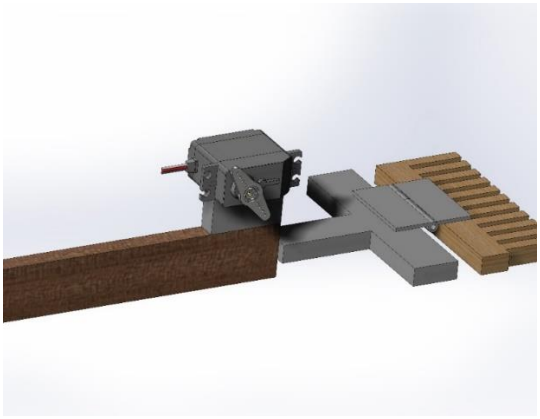


FIG.15: CAD DESIGN OF COMB

Manual Robot CAD

The manual Bot employed an arm with four sets of Comb-Motor assembly mounted on a Modified Holonomic Drivetrain. The basic structure of the Manual Bot is as follows:

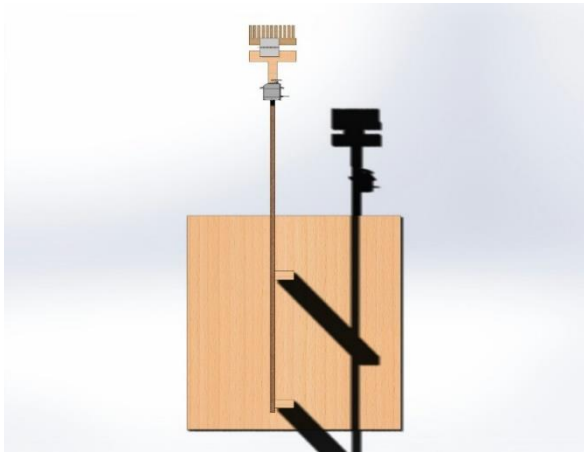


FIG.16: CAD DESIGN: TOP VIEW OF MANUAL BOT

FIG.17: CAD DESIGN: ISOMETRIC VIEW OF MANUAL BOT

Autonomous Robot CAD

Autonomous Robot is designed to drive itself sensing the requirements. It is a Line-follower Robot having Throw-arm mounted over a Modified Holonomic drivetrain. The basic features of Autonomous Robot is as follows:

1- Modified Holonomic Drivetrain
2- Employs Three LSA 08 Sensors to accomplish line following
3- Utilizes the input from the camera to take decisions.
4- Throw-arm employs Encoder-controlled External gear motor to thow the Shuttlecock

The basic CAD of the Autonomous Robot is as follows:

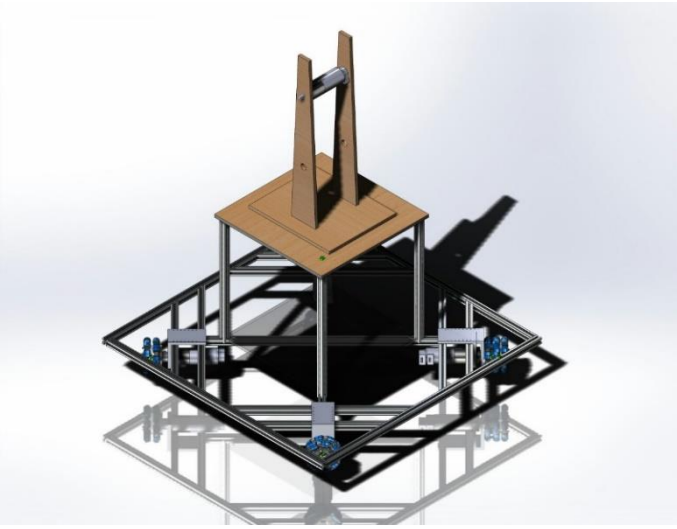


FIG.18: CAD DESIGN: ISOMETRIC VIEW OF MOTOR PROTOTYPE-1 OF AUTO BOT

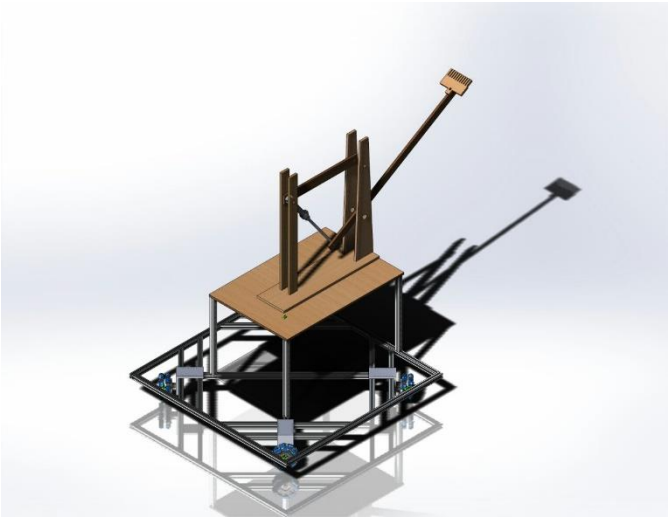


FIG.19: CAD DESIGN:ISOMETRIC VIEW OF PROTOTYPE-2 OF AUTOBOT

The development of the Autonomous Robot involved two major Phases:

1- Pneumatic Cylinder based throw mechanism
2- Motor based throw mechanism

The pneumatic Cylinder based throw mechanism employed a two link mechanism having one link replaced by a pencil-size Pneumatic Cylinder. The Basic Structure of the Mechanism is as shown below :

FIG.20: CAD DESIGN: SIDE VIEW OF PROTOTYPE-2 OF AUTOBOT

Prototype involving Pneumatic Cylinder based throw mechanism :

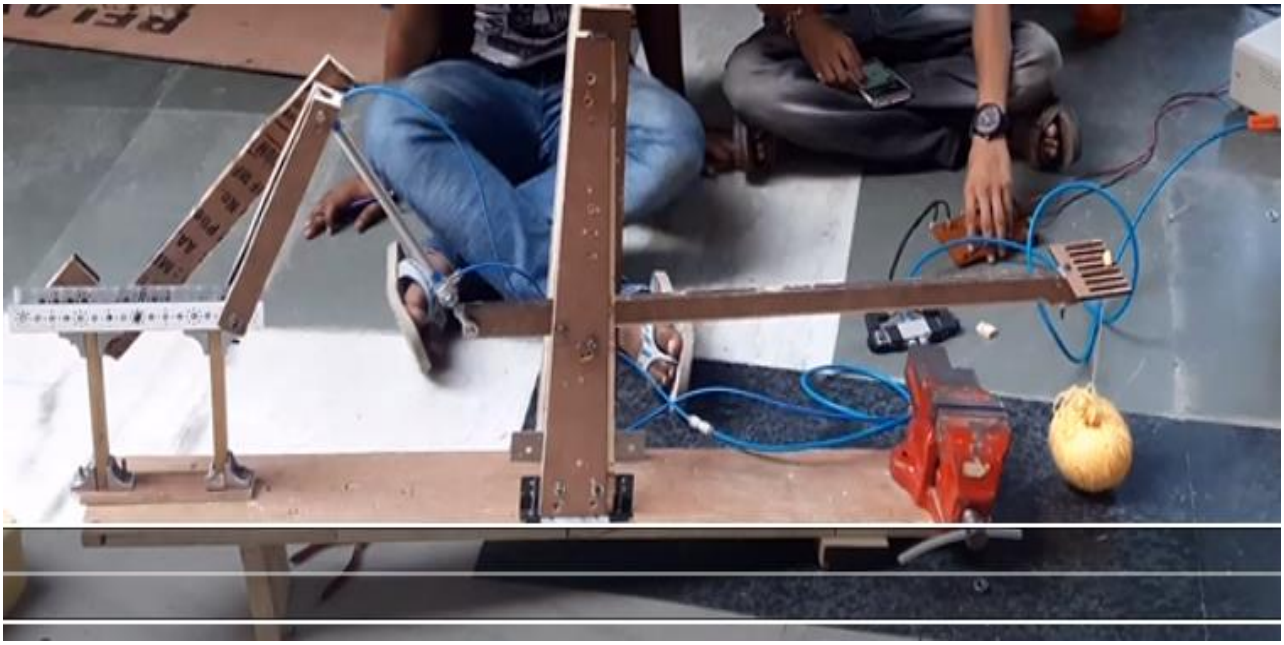


FIG.21: ACTUAL PROTOTYPE-2 (PISTON PROTOTYPE)

Motor based throw mechanism employed an encoder-controlled DC Motor to provide the required trajectory to the Shuttlecock. The implementation of Motor based throw mechanism which resulted out to be the final mechanism is as follows :

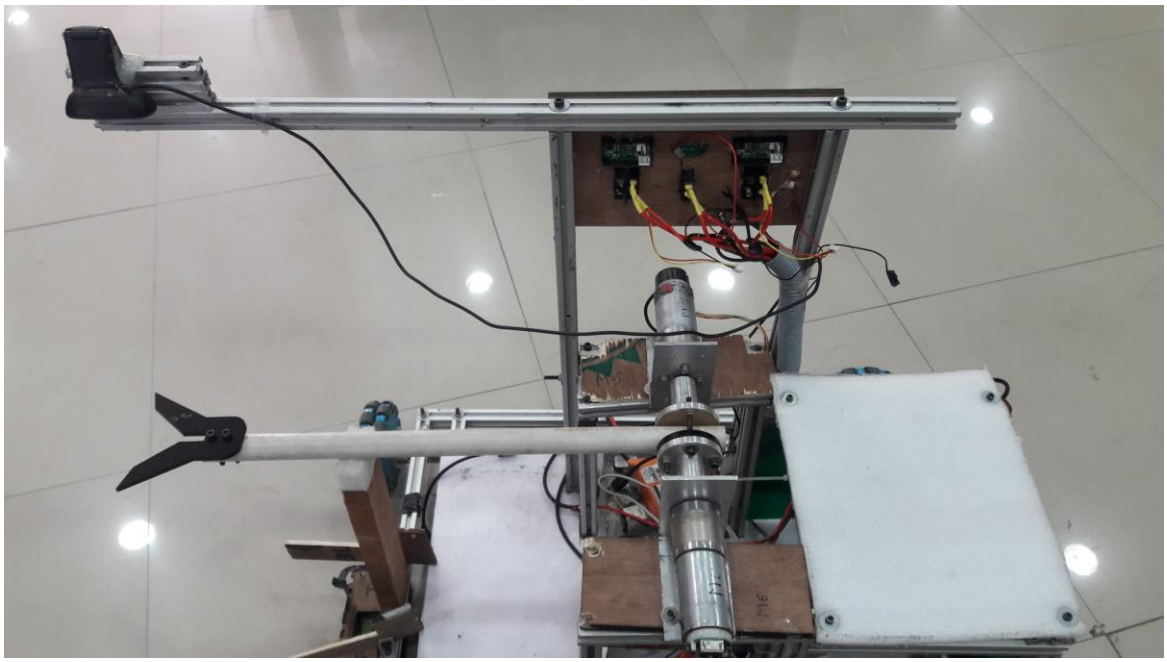


FIG.22: ACTUAL FINAL PROTOTYPE-3 OF AUTOBOT (2-MOTOR PROTOTYPE)

Block Diagram Autonomous Robot

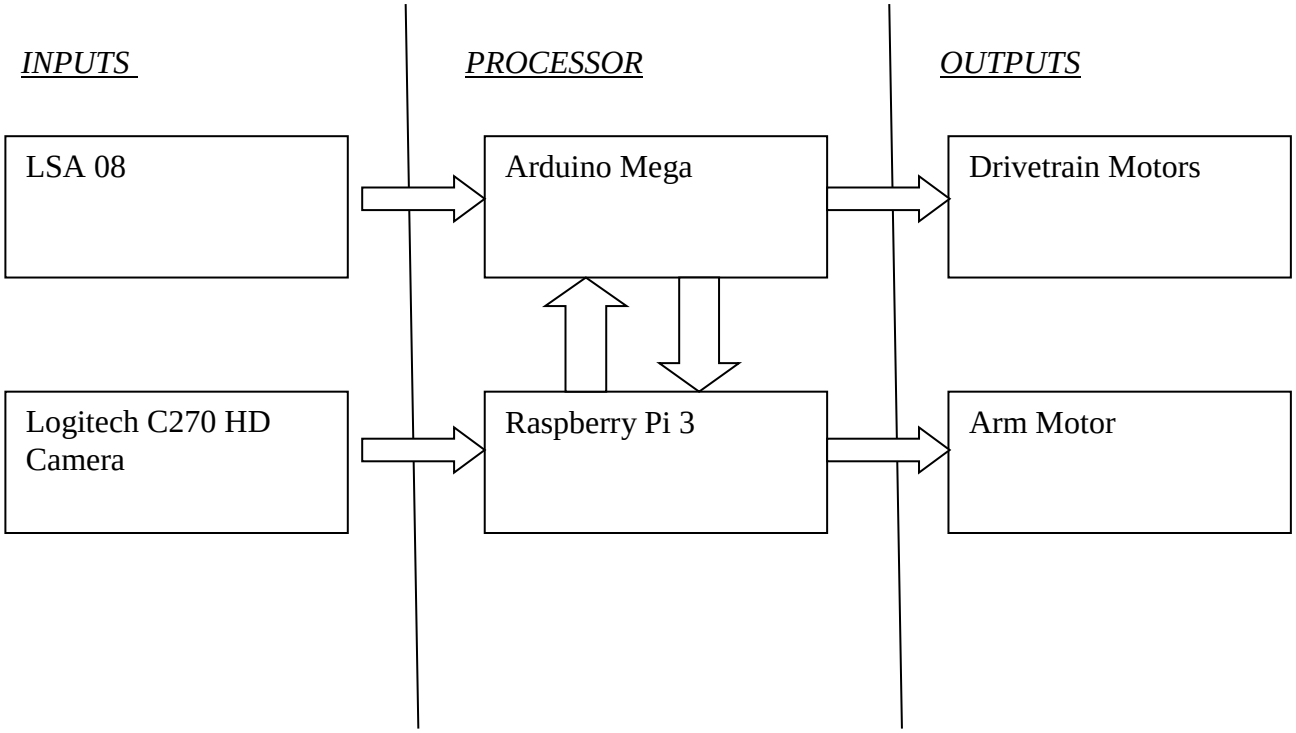


FIG.23: BLOCK DIAGRAM AUTONOMOUS ROBOT

Arena

Arena is the artificial environment defined for the working of a Machine. In our case it is a 14 X 14 m square area provided with materials of prescribed dimensions to test the functioning of the Robots.
For the complete testing of the working of our Robots we require the arena.

Following are the feature of our Arena:

1- Painted floor of Plywood
2- Comprises two pair zones of five different colors
3- Comprises three poles with ring overhead of 2,3 and 2 m in series.
4- Comprises two baskets of dia. 2 m
5- Contains White lines for guided motion of Autonomous Robot

PLAYING ARENA

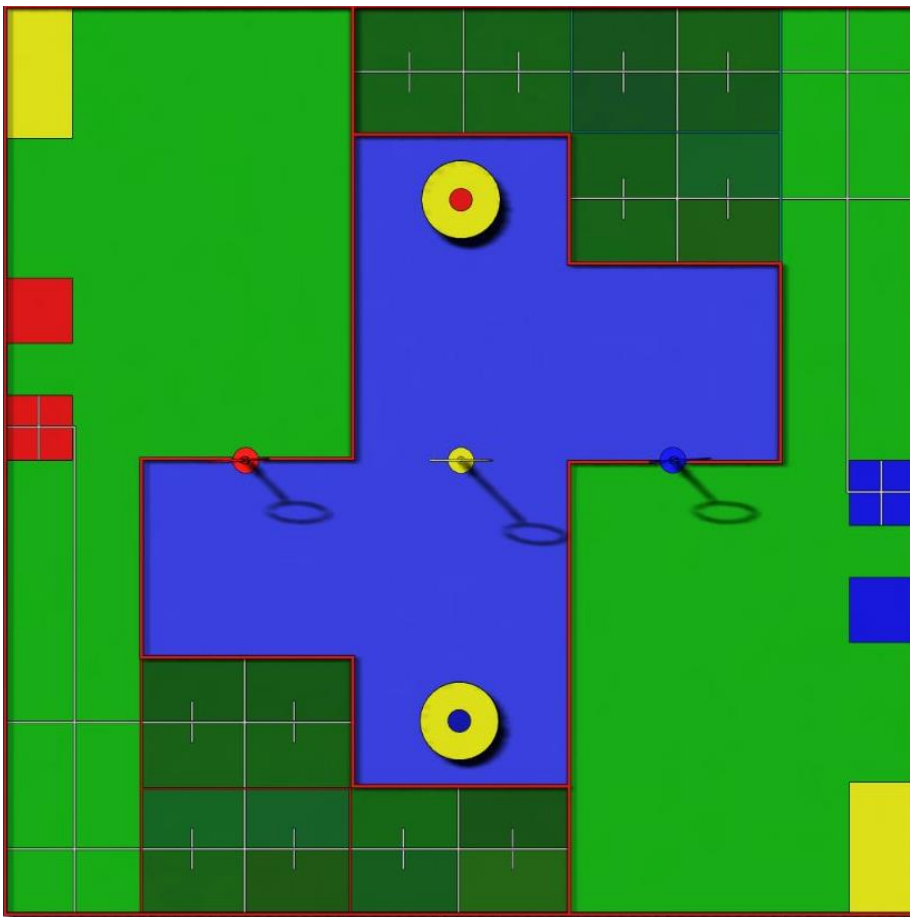


FIG.24: CAD DESIGN OF TOPVIEW OF MAIN ARENA

Arena Replica

We replicated the Arena in a small area within our college campus to test and observe the working of our Machines.

FIG.25: PROPOSED ARENA REPLICA AT COLLEGE

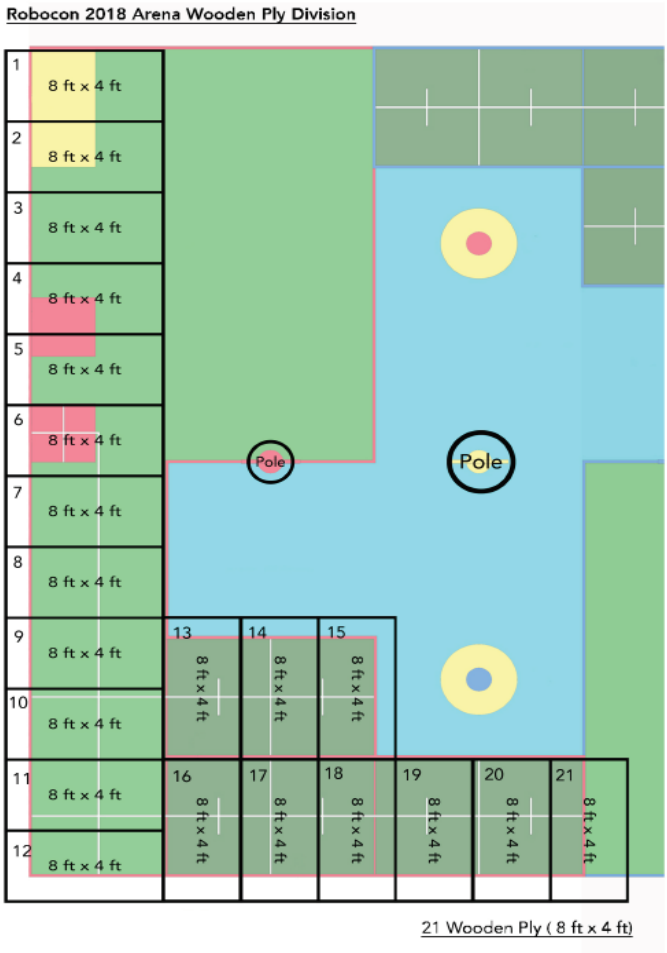


FIG.26: PROPOSED ARENA REPLICA AT COLLEGE

Chapter IV: Hardware and Software Specifications

4.1 Mechanical Components

Aluminum profiles

Extrusion is a process used to create objects of a fixed cross-sectional profile. A material is pushed through a die of the desired cross-section. The two main advantages of this process over other manufacturing processes are its ability to create very complex cross-sections, and to work materials that are brittle, because the material only encounters compressive and shear stresses. It also forms parts with an excellent surface finish.

Drawing is a similar process, which uses the tensile strength of the material to pull it through the die. This limits the amount of change which can be performed in one step, so it is limited to simpler shapes, and multiple stages are usually needed. Drawing is the main way to produce wire. Metal bars and tubes are also often drawn.

Extrusion may be continuous (theoretically producing indefinitely long material) or semi-continuous (producing many pieces). The extrusion process can be done with the material hot or cold. Commonly extruded materials include metals, polymers, ceramics, concrete, play dough, and foodstuffs. The products of extrusion are generally called "extrudates".

Hollow cavities within extruded material cannot be produced using a simple flat extrusion die, because there would be no way to support the centre barrier of the die. Instead, the die assumes the shape of a block with depth, beginning first with a shape profile that supports the center section. The die shape then internally changes along its length into the final shape, with the suspended center pieces supported from the back of the die. The material flows around the supports and fuses together to create the desired closed shape.

FIG.27 Aluminium extrusion profile 20x20

Omni wheels

Omni wheels or **poly wheels**, similar to Mecanum wheels, are wheels with small discs around the circumference which are perpendicular to the turning direction. The effect is that the wheel can be driven with full force, but will also slide laterally with great ease. These wheels are often employed in holonomic drive systems.

A platform employing three omni wheels in a triangular configuration is generally called Kiwi Drive. The Killough platform is similar; so named after Stephen Killough's work with omnidirectional platforms at Oak Ridge National Laboratory. Killough's 1994 design used pairs of wheels mounted in cages at right angles to each other and thereby achieved holonomic movement without using true omni wheels.

They are often used in intelligent robot research for small autonomous robots. In projects such as VEX Robotics, Robocon and FIRST Robotics, many robots use these wheels to have the ability to move in all directions. Omni wheels are also sometimes employed as powered casters for differential drive robots to make turning faster. Omniwheels are often used to allow for movement on the horizontal axis on a drivetrain, as well as forward and backward movement. Usually, this is achieved is by using an H-drive.

FIG.28 Omni wheel of diameter 150mm

Telescopic Channels

When it comes to runners and drawers, product diversity from Hettich is unbeatable. With an option for any loading capacity, tried and proven established ball-bearing slides and the unique Quadro runner provide the best solution for convenient, reliable drawer opening and closing in all classes of furniture.

FIG.29 Telescopic Channels

Acrylic Sheets

Poly(methyl methacrylate) (PMMA), also known as acrylic or acrylic glass as well as by the trade names Crylux, Plexiglas, Acrylite, Lucite, and Perspex among several others (see below), is a transparent thermoplastic often used in sheet form as a lightweight or shatter-resistant alternative to glass. The same material can be used as a casting resin, in inks and coatings, and has many other uses.

Although not a type of familiar silica-based glass, the substance, like many thermoplastics, is often technically classified as a type of glass (in that it is a non-crystalline vitreous substance) hence its occasional historical designation as *acrylic glass*. Chemically, it is the synthetic polymer of methyl methacrylate. The material was developed in 1928 in several different laboratories by many chemists, such as William Chalmers, Otto Röhm, and Walter Bauer, and was first brought to market in 1933 by the Rohm and Haas Company under the trademark Plexiglas.

PMMA is an economical alternative to polycarbonate (PC) when tensile strength, flexural strength, transparency, polishability, and UV tolerance are more important than impact strength, chemical resistance and heat resistance.[5]Additionally, PMMA does not contain the potentially harmful bisphenol-A subunits found in polycarbonate. It is often preferred because of its moderate properties, easy handling and processing, and low cost. Non-modified PMMA behaves in a brittle manner when under load, especially under an impact force, and is more prone to scratching than conventional inorganic glass, but modified PMMA is sometimes able to achieve high scratch and impact resistance

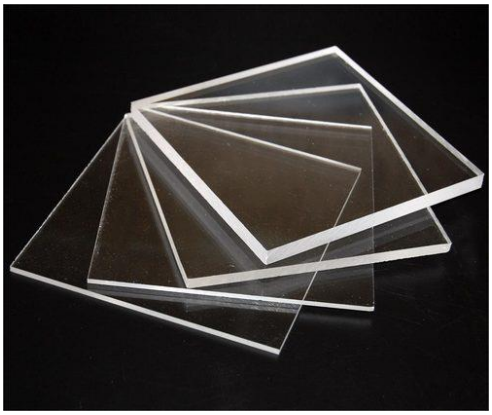


FIG.30 Acrylic Sheets

4.2 Electronics Components

Loading Servo motors

A servomotor is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration.[1] It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors.

Servomotors are not a specific class of motor although the term *servomotor* is often used to refer to a motor suitable for use in a closed-loop control system.

Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

A servomotor is a closed-loop servomechanism that uses position feedback to control its motion and final position. The input to its control is a signal (either analogue or digital) representing the position commanded for the output shaft.

The motor is paired with some type of encoder to provide position and speed feedback. In the simplest case, only the position is measured. The measured position of the output is compared to the command position, the external input to the controller. If the output position differs from that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the positions approach, the error signal reduces to zero and the motor stops.

The very simplest servomotors use position-only sensing via a potentiometer and bang-bang control of their motor; the motor always rotates at full speed (or is stopped). This type of servomotor is not widely used in industrial motion control, but it forms the basis of the simple and cheap servos used for radio-controlled models.

FIG.31 Servo Motor

Drive motors

A **DC motor** is any of a class of rotary electrical machines that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic

fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current flow in part of the motor.

DC motors were the first type widely used, since they could be powered from existing direct-current lighting power distribution systems. A DC motor's speed can be controlled over a wide range, using either a variable supply voltage or by changing the strength of current in its field windings. Small DC motors are used in tools, toys, and appliances. The universa motor can operate on direct current but is a lightweight motor used for portable power tools and appliances. Larger DC motors are used in propulsion of electric vehicles, elevator and hoists, or in drives for steel rolling mills. The advent of power electronics has made replacement of DC motors with AC motors possible in many applications.

FIG.32 12V DC geared motor (1500 rpm)

Microcontroller and Drive circuits

The Arduino Due is a microcontroller board based on the Atmel SAM3X8E ARM Cortex-M3 CPU. It is the first Arduino board based on a 32-bit ARM core microcontroller. It has 54 digital input/output pins (of which 12 can be used as PWM outputs), 12 analog inputs, 4 UARTs (hardware serial ports), a 84 MHz clock, an USB OTG capable connection, 2 DAC (digital to analog), 2 TWI, a power jack, an SPI header, a JTAG header, a reset button and an erase button.
Warning: Unlike most Arduino boards, the Arduino Due board runs at 3.3V. The maximum voltage that the I/O pins can tolerate is 3.3V. Applying voltages higher than 3.3V to any I/O pin could damage the board.
The board contains everything needed to support the microcontroller; simply connect it to a computer with a micro-USB cable or power it with a AC-to-DC adapter or battery to get started. The Due is compatible with all Arduino shields that work at 3.3V and are compliant with the 1.0 Arduino pinout.
The Due follows the 1.0 pinout:

- **TWI**: SDA and SCL pins that are near to the AREF pin.
- **IOREF**: allows an attached shield with the proper configuration to adapt to the voltage provided by the board. This enables shield compatibility with a 3.3V board like the Due and AVR-based boards which operate at 5V.
- An unconnected pin, reserved for future use.

Arduino DUE is used in Autonomous Bot for Fast processing which is integrated with mega, 3x lsa08 and raspberry pi.

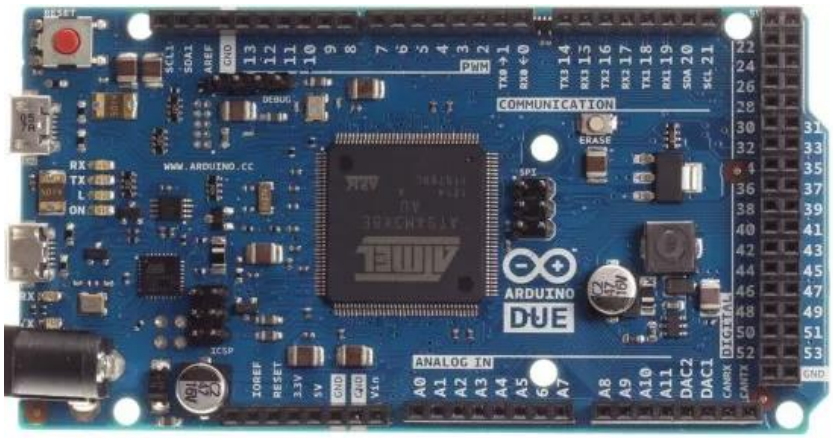


FIG.33 Atmel SAM3X8E ARM Cortex-M3 CPU

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a ACto-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Arduino Mega is used in Manual Robot and in Autonomous robot in throwing mechanism.

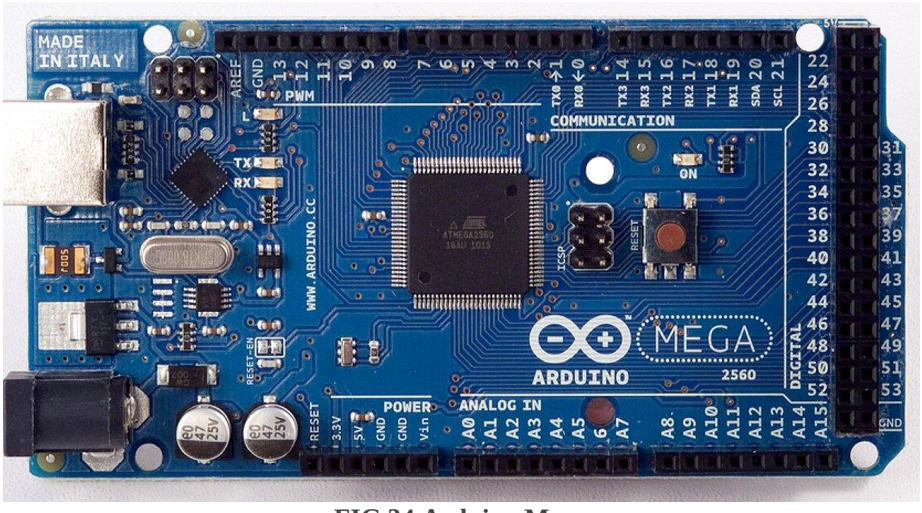


FIG.34 Arduino Mega

The **Raspberry Pi** is a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to promote the teaching of basic computer science in schools and

in developing countries. The original model became far more popular than anticipated,[7] selling outside its target market for uses such as robotics. It does not include peripherals (such as keyboards, mice and cases). However, some accessories have been included in several official and unofficial bundles.

Raspberry Pi 3 Model B was released in February 2016 with a 64 bit quad core processor, and has on-board WiFi, Bluetooth and USB boot capabilities. On Pi Day2018 **model 3B+** appeared with a faster 1.4 GHz processor and a 3 times faster network based on gigabit ethernet (300 Mbit / s) or 2.4 / 5 GHz dual-band Wi-Fi (100 Mbit / s). Other options are: Power over Ethernet (PoE), USB boot and network boot (an SD card is no longer required). This allows the use of the Pi in hard-to-reach places (possibly without electricity).
The Raspberry Pi 3, with a quad-core ARM Cortex-A53 processor, is described as 10 times the performance of a Raspberry Pi 1. This was suggested to be highly dependent upon task threading and instruction set use. Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelised tasks.

FIG.35 Raspberry Pi B+

MD30C is the successor of MD30B which is designed to drive medium to high power brushed DC motor with current capacity up to 80A peak and 30A continuously. Fully NMOS design not only provides faster switching time, it is also more efficient and no heatsink or fan is required.

Besides that, MD30C also incorporates some user friendly features such as reverse polarity protection and onboard PWM generator which allows it to operate without a host controller. The motor can simply be controlled with the onboard switches and speed potentiometer. External switches and potentiometer can also be used.

FIG.36 Drive circuit for PMDC motors 30 amps

MD10C is a newer version of the MD10B which is designed to drive high current brushed DC motor up to 13A continuously. It offers several enhancements over the MD10B such as support for both locked-antiphase and sign-magnitude PWM signal as well as using full solid state components which result in faster response time and eliminate the wear and tear of the mechanical relay.

FIG.37 Drive circuit for DC geared motors 10 amps

A **buck converter** (**step-down converter**) is a DC-to-DC power converter which steps down voltage (while stepping up current) from its input (supply) to its output (load). It is a class of switched-mode power supply (SMPS) typically containing at least two semiconductors (a diode and a transistor, although modern buck converters frequently replace the diode with a second transistor used for synchronous rectification) and at least one energy storage element, a capacitor, inductor, or

the two in combination. To reduce voltage ripple, filters made of capacitors (sometimes in combination with inductors) are normally added to such a converter's output (load-side filter) and input (supply-side filter).

Switching converters (such as buck converters) provide much greater power efficiency as DC-to-DC converters than linear regulators, which are simpler circuits that lower voltages by dissipating power as heat, but do not step up output current.

FIG.38 XL4015 5a DC-DC step down buck convertor

Sensors

LSA08 (Advance Line Following Sensor Bar) consist of 8 sensors pair. LSA08 is typically used for embedded system or robots for line following task. The specially selected wavelength of **super bright green LED** as the sensor's transmitter enables LSA08 to operate on various different colour surfaces. LSA08 is capable to operate on surface with colour of Red, Green, Blue, White, Black, Gray and possibly other colours with distinct brightness different. LSA08 has several different output modes, for the convenience of use for any system. Namely, the digital output port (8 parallel output line), the serial communication port (UART) and the analog output port.

FIG.39 LSA08 line following sensor array (IR sensor)

Camera: LOGITECH C525

FIG.40 Logitech Camera C525

- HD video calling (1280 x 720 pixels) with recommended system
- HD video capture: Up to 1280 x 720 pixels
- Logitech Fluid Crystal™ Technology
- Autofocus
- Photos: Up to 8 megapixels (software enhanced)
- Built-in mic with Logitech RightSound technology
- Hi-Speed USB 2.0 certified (recommended)
- Universal clip fits laptops, LCD or CRT monitors

Logitech webcam software:

- Pan, tilt, and zoom controls
- Video and photo capture
- Face tracking
- Motion detection

Chapter V: Working of project

5.1 How to Start the (Autonomous and manual) Robot

- To start the robot you first have to make sure that all the components are properly connected.
- Connect all the batteries in their respective connectors and make sure that there should be no case of short circuiting.
- Turn on both the emergency switches and if all the voltage display got light up you are good to go.
- Also make sure status led on ps2 remote is turned on.(Only for manual bot)
- Please wear safety helmet and safety shoes as per industry standards.

5.2 Various Circuit Boards of Bots

MANUAL BOT PCBs

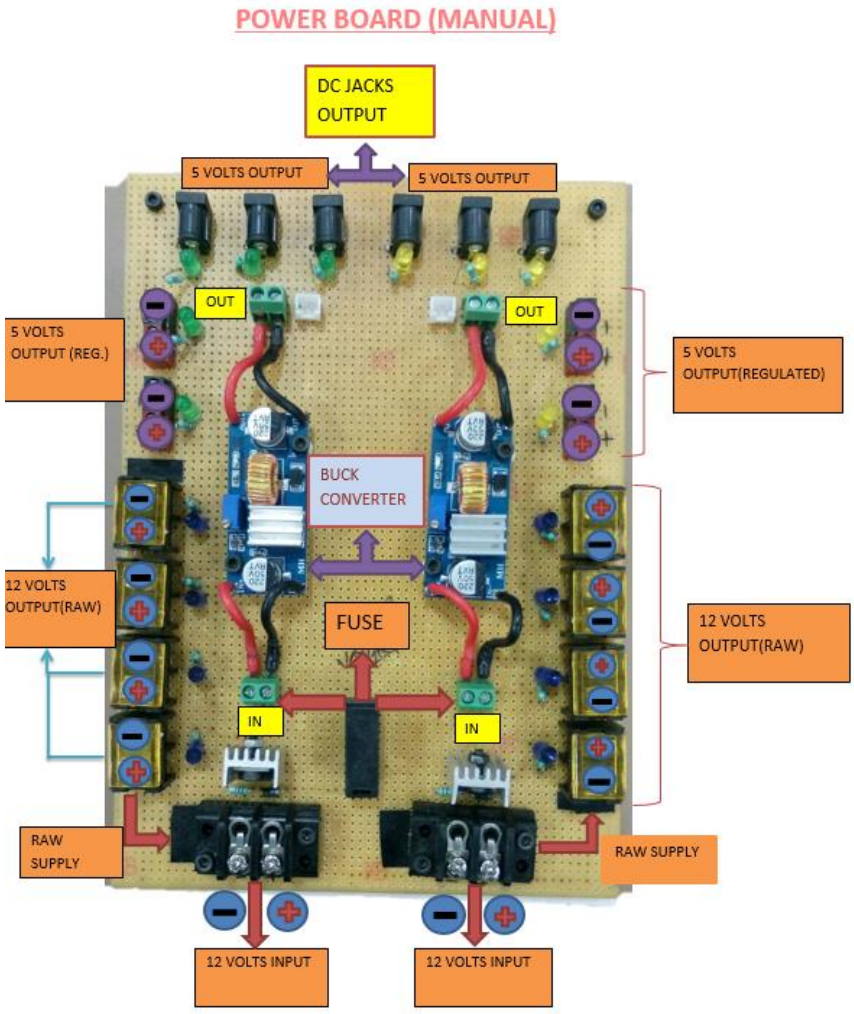


FIG.41: Power Board for Manual Bot

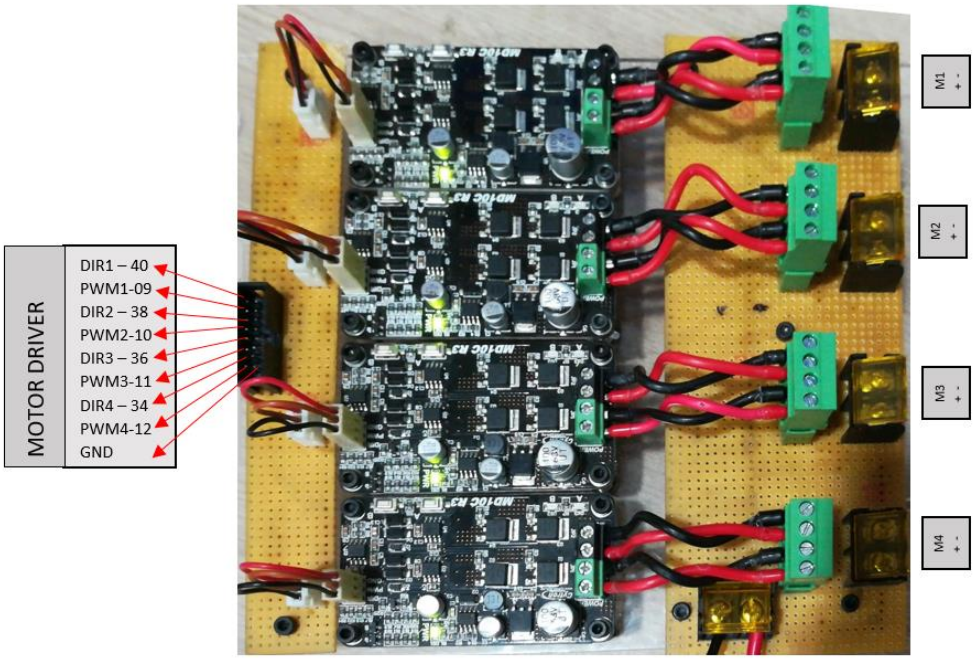


FIG.42: Motor Driver board for Manual Bot

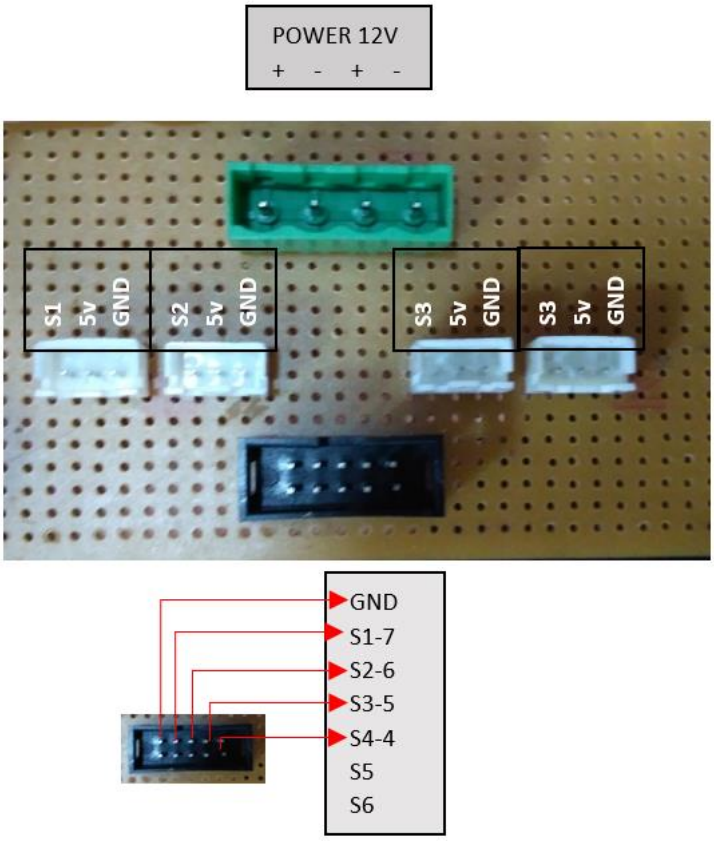


FIG.43: Manual Servo Power and Control Distribution Board

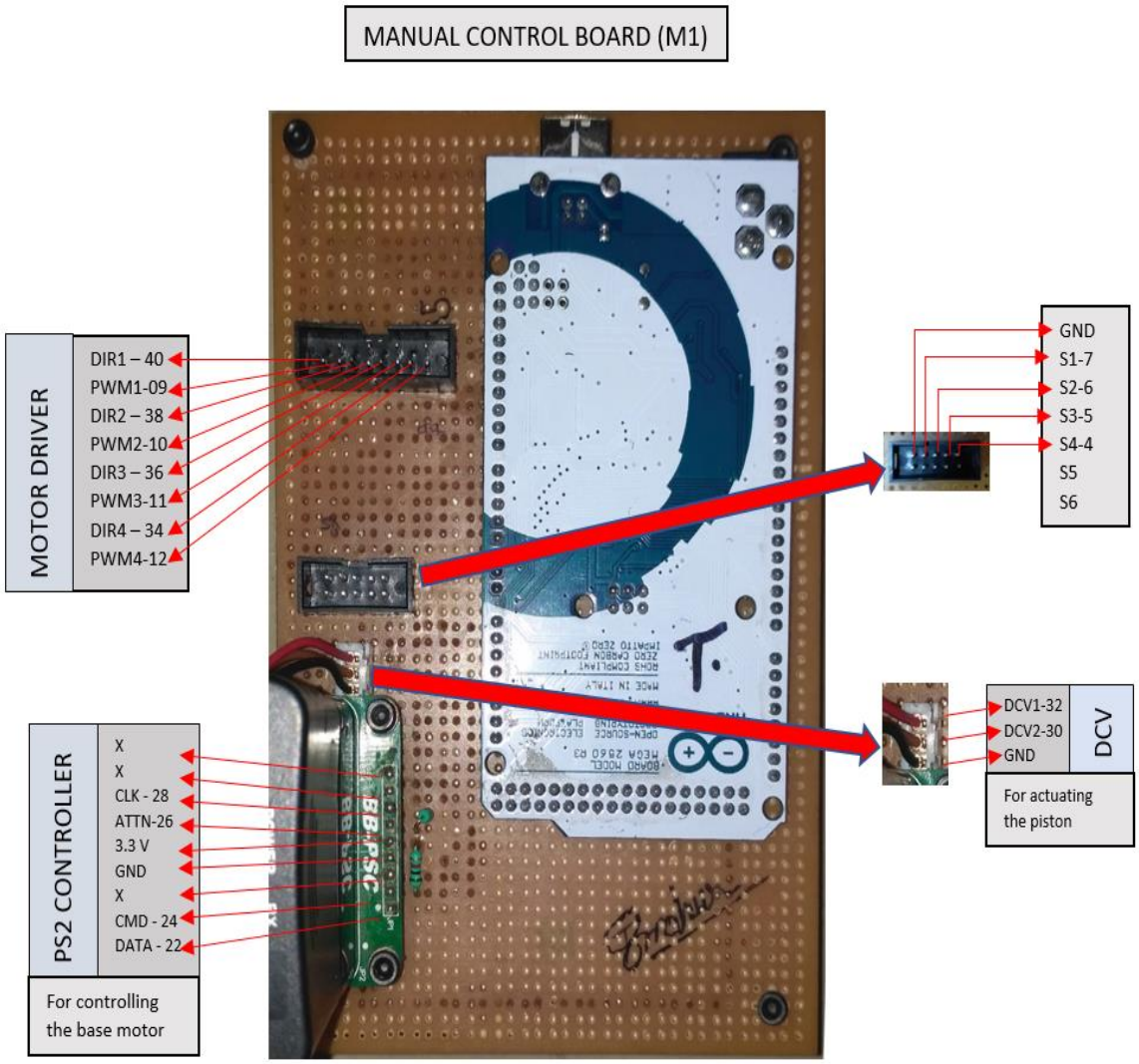


FIG.44: Manual Main Control Board

This PCB is the controlling PCB of Manual Robot. It copromises of Motor Driver, PS2 Controller, DCV actuation. This board is Arduino based board which have Arduino MEGA as its microcontroller.

MANUAL FLOW CHART

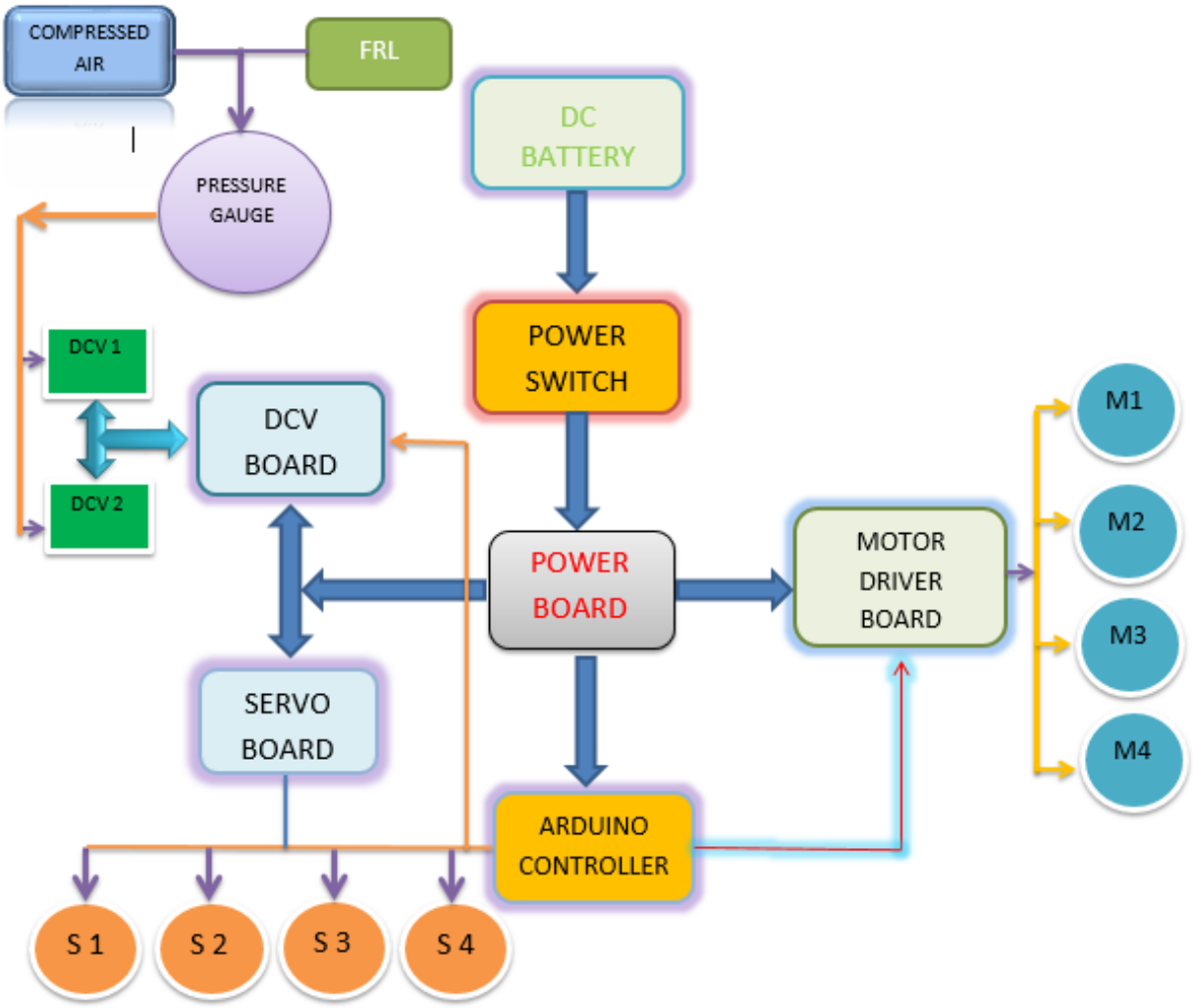


FIG.45: MANUAL BLOCK DIAGRAM

This is structural block diagram of Manual Robot. It shows each connection of power and signal distribution system. Here M denotes Base Motor which are base driving motors and S represent Servo motors for picking up mechanism.

AUTONOMOUS BOT PCBs

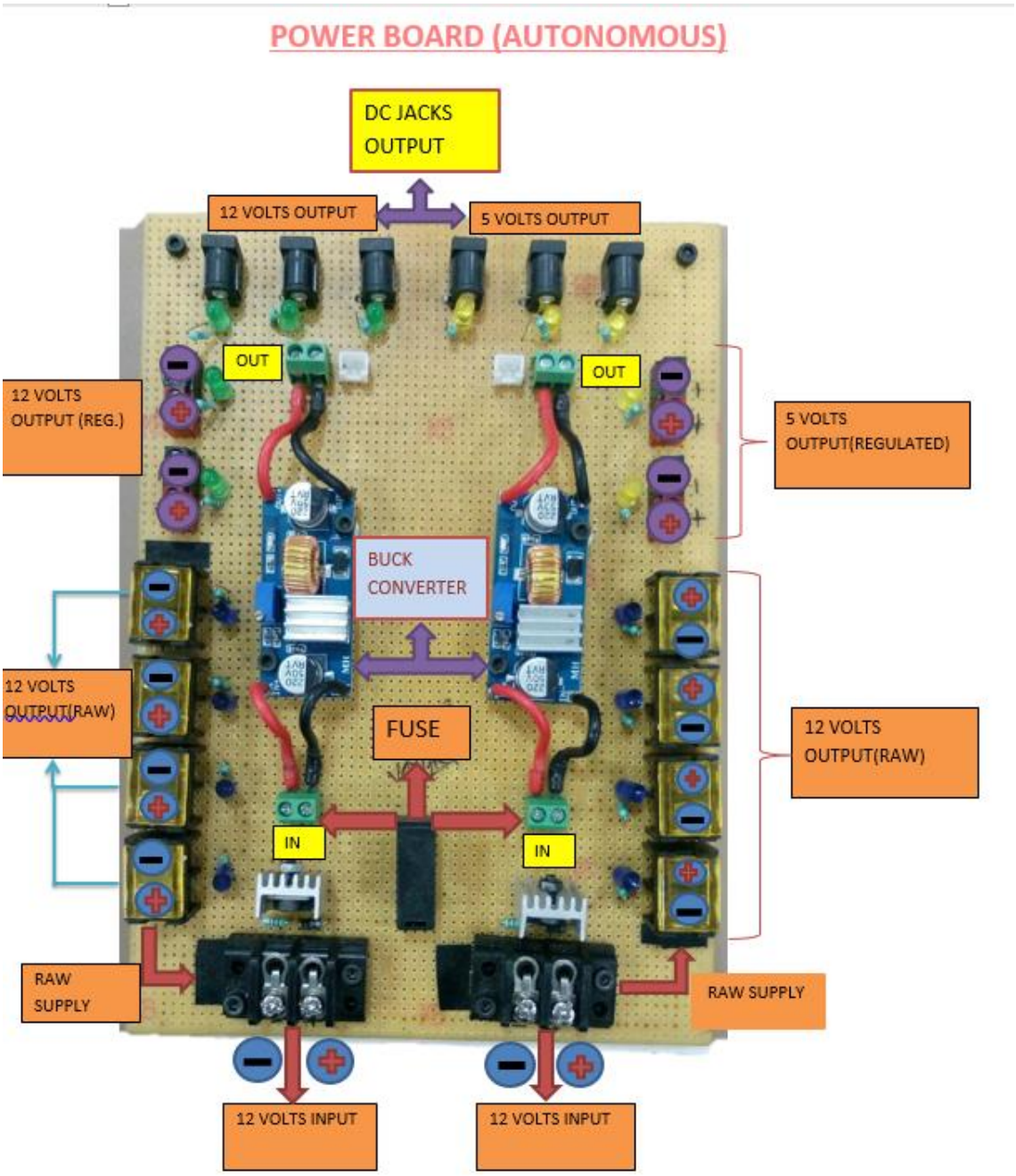


FIG.46: Power Board for Autonomous Bot

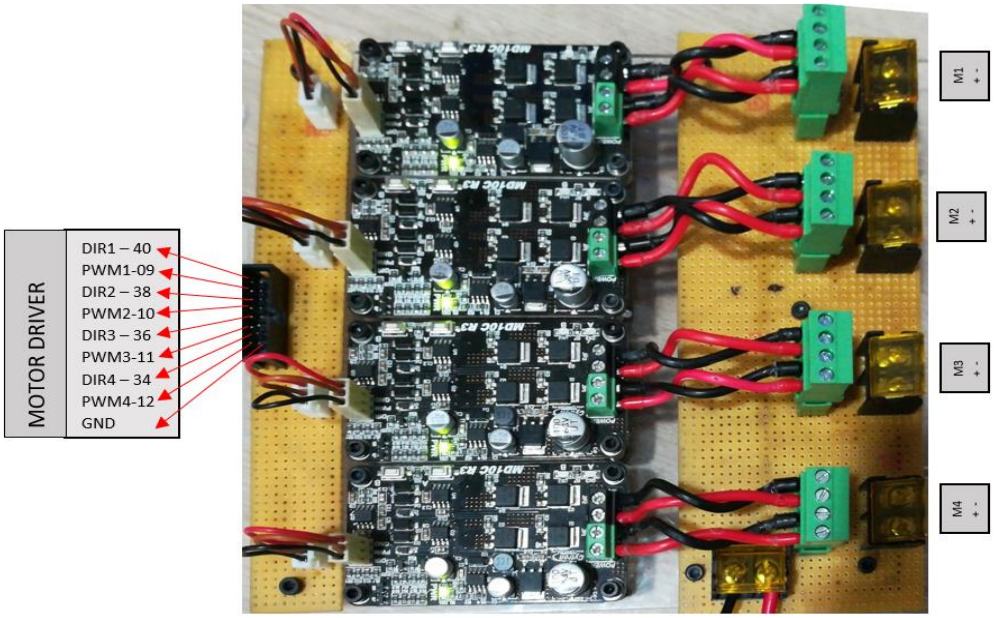


FIG.47: Motor Driver board for Autonomous Bot

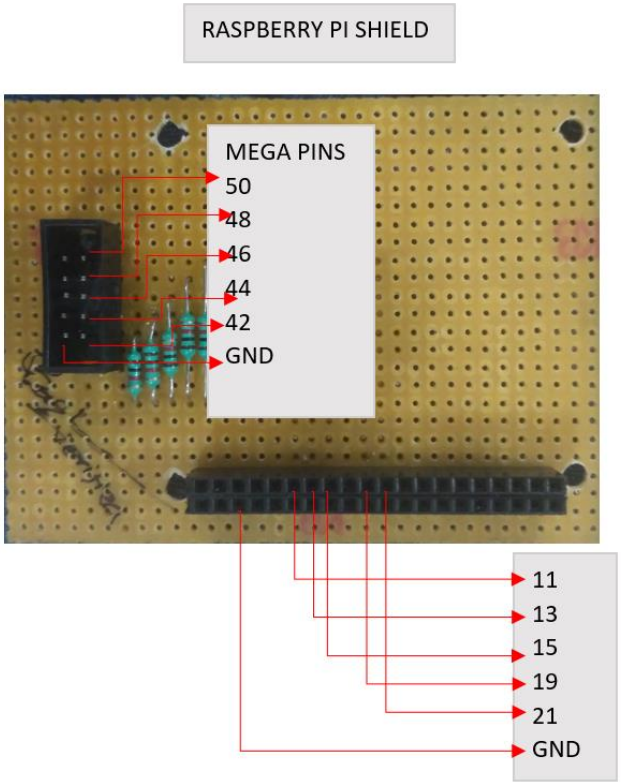


FIG.48: Raspberry Shield for RPI B+

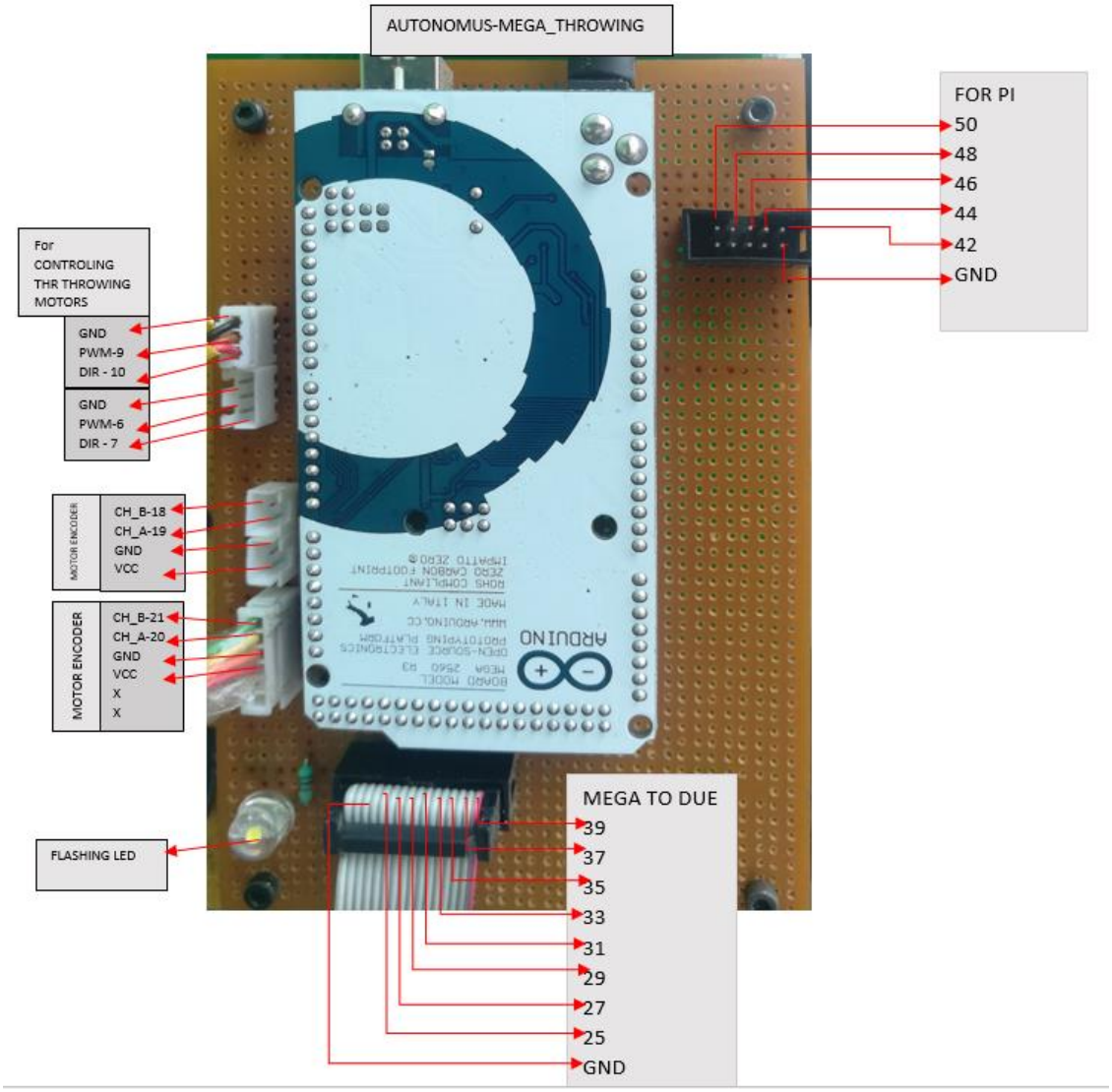


FIG.49: Autonomous Board of Arduino MEGA for Throwing

This PCB is the controlling PCB of Autonomous Robot. It compromises of Motor Driver, microcontroller interconnection, encoder signals. This board is Arduino based board which have Arduino MEGA as its microcontroller.

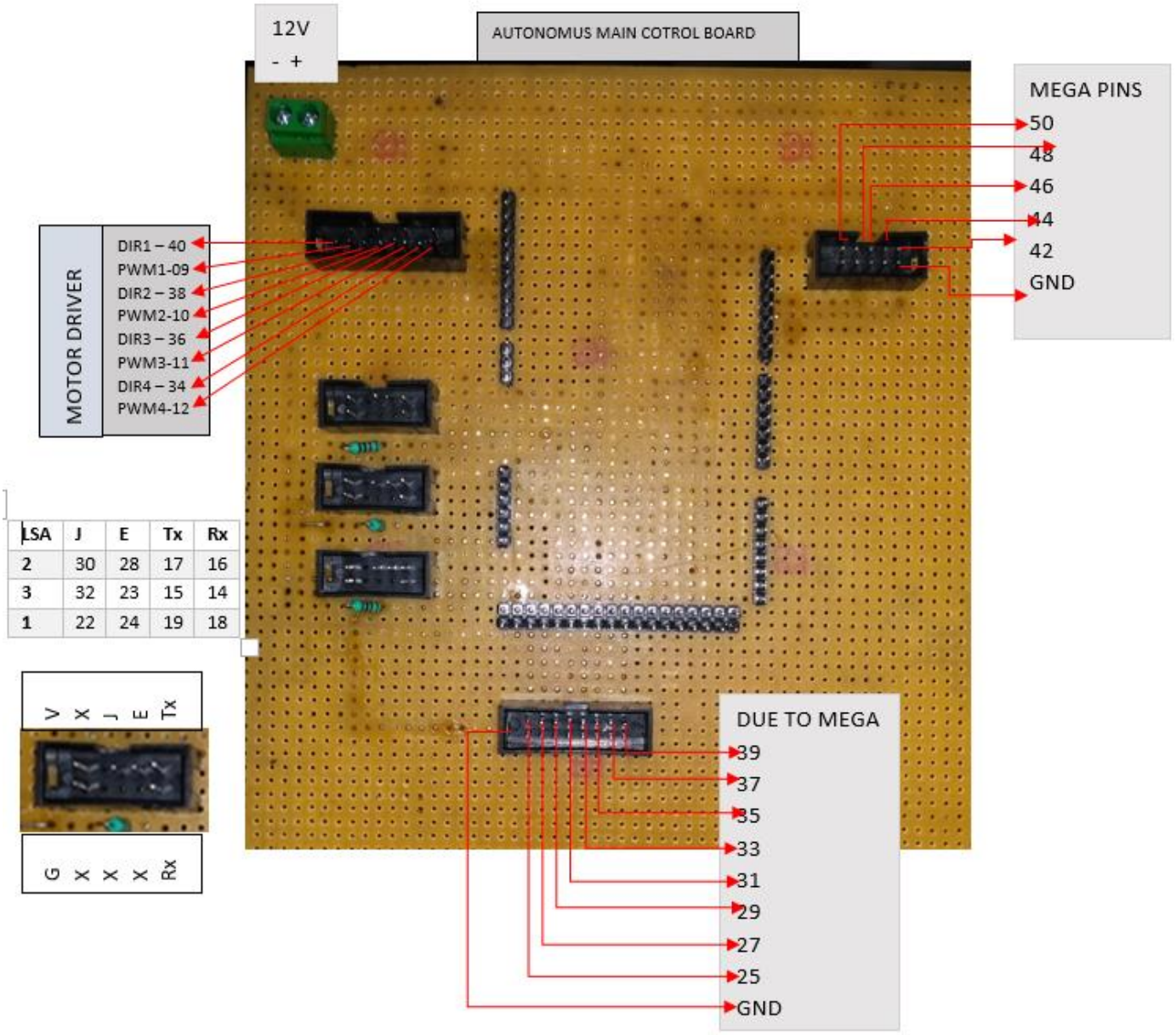


FIG.50: Autonomous Board of Arduino DUE for Main Control

This PCB is the controlling PCB of Autonomous Robot. It compromises of Motor Driver, LSA 08 controlling, encoder signals. This board is Arduino based board which have Arduino DUE as its microcontroller.

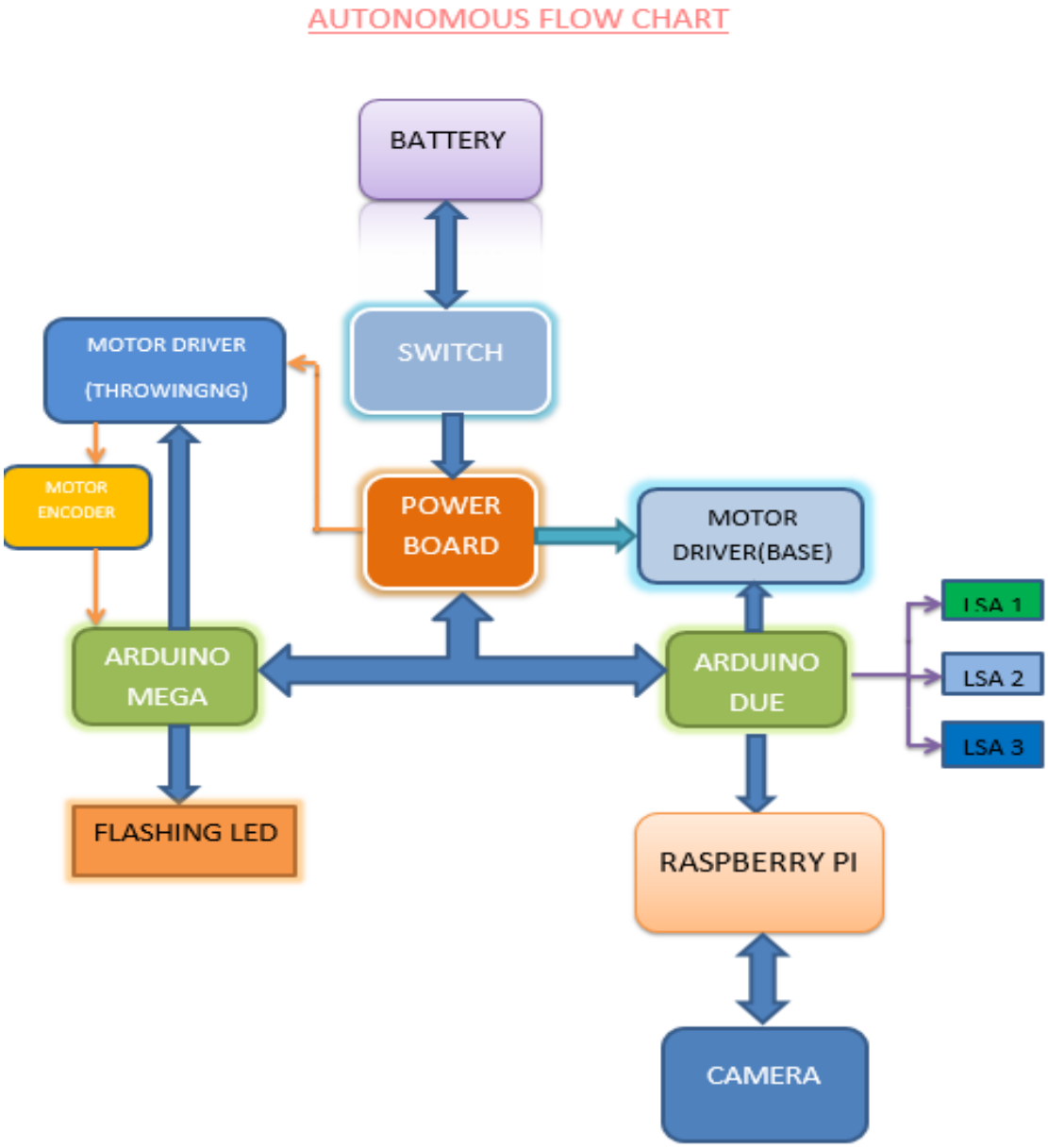


FIG.51: AUTONOMOUS BLOCK DIAGRAM

This is structural block diagram of Autonomous Robot. It shows each connection of power and signal distribution system.

Final Designs

Manual Bot

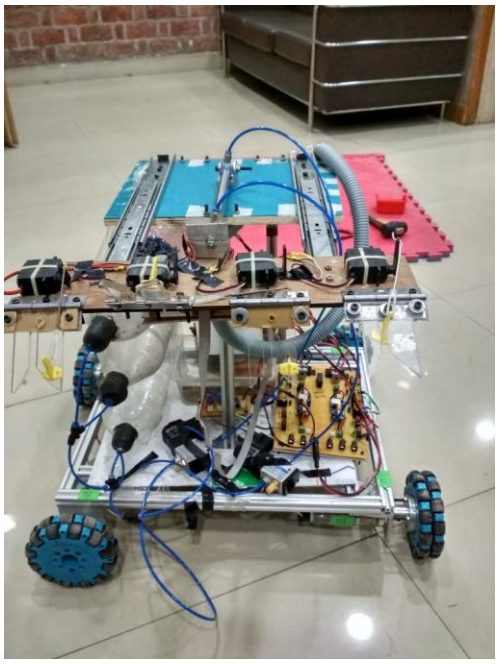


FIG.52 Front View

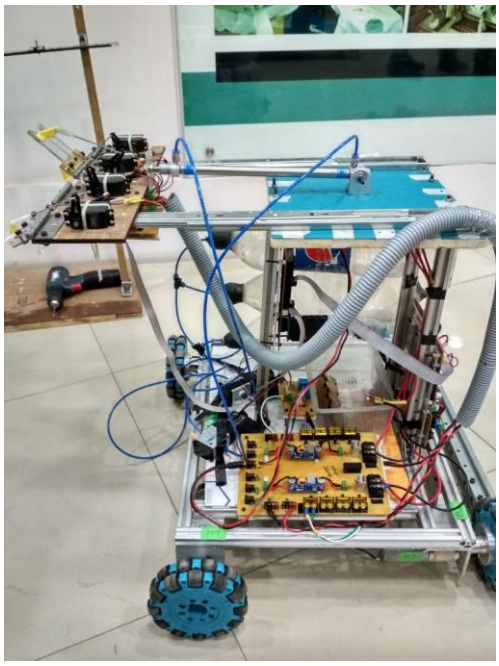


FIG.53 Side View

Autonomous Bot

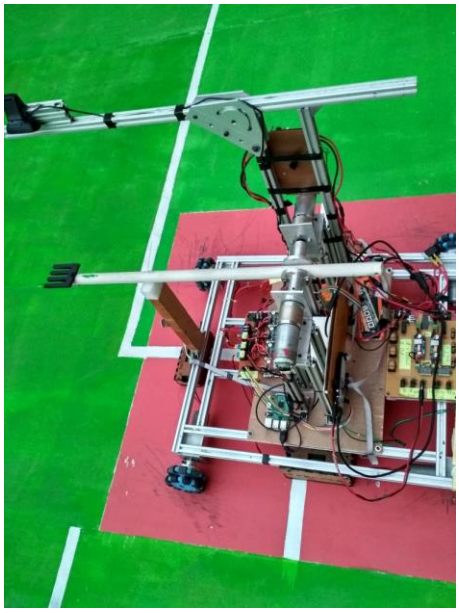


FIG.54 Autobot Left View

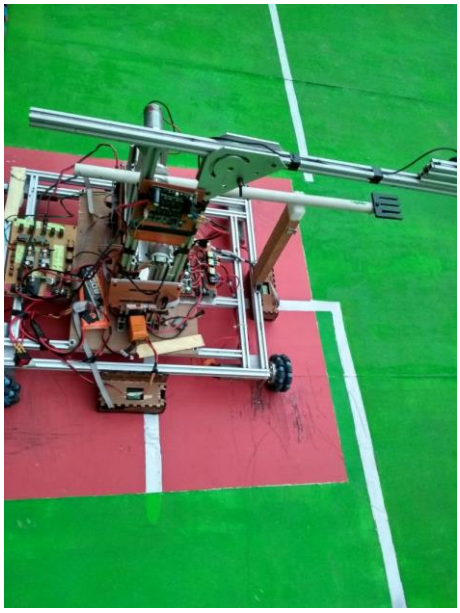


FIG.55 Autobot Right View

Chapter VI: Results

Average Time to complete the Task : **1 Min 30 Sec**

Table.3: Average timing information

TASK	Average Time
First loading	17 sec
First Throw	30 sec
Second Loading	40 sec
Second Throw	45 sec
Third Loading	1 Min 25 sec
Third Throw	1 Min 30 sec

FIG.56 Practice Time

Chapter VII: Conclusion and suggestion for further work.

7.1 Conclusion

A new venture is always the harbinger of opportunity for success as well as a way to learn and mend the ways thus instigating improvisation.

This year robocon was the best journey for AKGEC ROBOCON Team till yet .We managed to occuoy 10^{th} position among 107 teams across the nation and also defeated the known teams of robocon. But this is just the start there is lot more to achieve in this field. This experience boosted our confidence level and also motivated us to go beyond this.

LESSONS LEARNT

Team spirit:
Team spirit is the most important thing and a major factor too for the success of the team. At nights when other team members worked, hardly any one of us went to hostels to sleep. We used to sleep on the arena made in front of the club when it became impossible for us to keep our eyes open. This helped a lot in motivating others to keep working.

Time Management:
Time management plays an important role for the success of any event. If we could have practiced more we must have performed better. So this is one of the areas to be worked on.

EXPERIENCE:
To be a best team we need to have experience and this reflected in this years performance as we are improving year by year. The experience we got from this year will surely take our performance to a new level in the coming years.

Team Robocon AKGEC 2018

7.2 Suggestions for further work

- We can include some more sensors such as laser sensors to precise robot movements.
- We can use some more ergonomic controller in future like ps4 controllers.
- Robustness of the machine can be improved.
- Capacity to load shuttle can be increased.

Appendix (for source code)

Appendix: A SOURCE CODE

A.1 MANUAL BOT

```
#include "PS2X_lib.h"
#include <Servo.h>

#define PS2_DAT     22   //Pins for ps2
#define PS2_CMD     24
#define PS2_SEL     26
#define PS2_CLK     28

int dir1 = 40, dir2 = 38, dir3 = 36, dir4 = 34;
//direction pins of motor driver
int pwm1 = 9, pwm2 = 10, pwm3 = 11, pwm4 = 12;
// pwm pins of motor driver
int mapped;
int pwm=180;                //initial speed

void move_forward();        // drive train controlling
loops
void move_left();
void move_right();
void move_backward();
void move_clockwise();
void move_anticlockwise();

void initial_condition();

void stick_values();

void release_servo1();      //servo controlling
funnctions
void release_servo2();
void release_servo3();
void release_servo4();
void load_shuttle();
void load_initialpostion();

int servoPin1 = 7;          // servo piins
int servoPin2 = 6;
int servoPin3 = 5;
int servoPin4 = 4;

int dcv1=30,dcv2=32;

Servo Servo1;
Servo Servo2;
Servo Servo3;
Servo Servo4;

#define pressures   false
#define rumble      false
int flagCross = 0;

PS2X ps2x;        // create PS2 Controller Class

int error = 0;
byte type = 0;
byte vibrate = 0;

void setup()
{
  pinMode(pwm1, OUTPUT);    //pwm
  pinMode(pwm2, OUTPUT);
  pinMode(pwm3, OUTPUT);
  pinMode(pwm4, OUTPUT);
  pinMode(dir1, OUTPUT);     //dir
  pinMode(dir2, OUTPUT);
  pinMode(dir3, OUTPUT);
  pinMode(dir4, OUTPUT);
  pinMode(dcv1, OUTPUT);     //dcv
  pinMode(dcv2, OUTPUT);


  Servo1.attach(servoPin1);  //servo
  Servo2.attach(servoPin2);
  Servo3.attach(servoPin3);
  Servo4.attach(servoPin4);

  delay(300);  //added delay to give wireless ps2
module some time to startup, before configuring it

   analogWrite(pwm1,0);  // initially pwm is zero
   analogWrite(pwm2,0);
   analogWrite(pwm3,0);
   analogWrite(pwm4,0);


  //setup pins and settings: GamePad(clock, command,
attention, data, Pressures?, Rumble?) check for error
  error = ps2x.config_gamepad(PS2_CLK, PS2_CMD,
PS2_SEL, PS2_DAT, pressures, rumble);
 }
void loop() {
  if (error == 1)        //skip loop if no controller found
    return;
    ps2x.read_gamepad();                  //read controller
    ps2x.reconfig_gamepad();               //configure
again automatically each time

if (ps2x.Button(PSB_L2))
```

```
 {                              // INTIAL POSITION FOR
THE SERVO MOTORS TO BE PARALLEL TO THE
GROUND
  load_initialpostion();
  }

// LOADING ALL THE SHUTTLECOCKS IN ONE
GO

if (ps2x.Button(PSB_R2))
 {
  load_shuttle();
 }

//RELEASING CODE ONE BY ONE

if (ps2x.Button(PSB_TRIANGLE))  //releasing via
servo1
  {
    release_servo1();
  }

if (ps2x.Button(PSB_CIRCLE))     //releasing via
servo2
  {
    release_servo4();
  }

if (ps2x.Button(PSB_CROSS))      //releasing via
servo3
  {
     release_servo3();
  }

if (ps2x.Button(PSB_SQUARE))     //releasing via
servo3
  {
    release_servo2();
  }


//Base motor code
// if ((ps2x.Analog(PSS_LY) > 120 &&
ps2x.Analog(PSS_LY) < 140) &&
(ps2x.Analog(PSS_LX) > 120 &&
ps2x.Analog(PSS_LX) < 140) &&
(ps2x.Analog(PSS_RY) > 120 &&
ps2x.Analog(PSS_RY) < 140) &&
(ps2x.Analog(PSS_RX) > 120 &&
ps2x.Analog(PSS_RX) < 140))
//   initial_condition();

    if((ps2x.Analog(PSS_LY)) > 120  &&
(ps2x.Analog(PSS_LY) < 140))
    {
     analogWrite(pwm2, 0);
     analogWrite(pwm4, 0);
    }
    else if (ps2x.Analog(PSS_LY) < 120)
      move_forward();
    else if (ps2x.Analog(PSS_LY) > 140)
      move_backward();

    if (ps2x.Button(PSB_L1))
     {
       if (ps2x.Analog(PSS_RX) < 120)
        move_anticlockwise();
       else if (ps2x.Analog(PSS_RX) > 140)
        move_clockwise();
       else
        initial_condition();
     }
    else if(ps2x.Analog(PSS_RX) > 120 &&
ps2x.Analog(PSS_RX) < 140)
    {
     analogWrite(pwm1,0);
     analogWrite(pwm3,0);
    }
     else if ((ps2x.Analog(PSS_RX) < 120))
      move_left();
     else if ((ps2x.Analog(PSS_RX) > 140))
      move_right();


if(ps2x.Button(PSB_R1))  // change pwm from ps2
button
  {
   pwm=255;
  }
else
  {
   pwm=180 ;          // initial pwm
  }


if (ps2x.Button(PSB_PAD_UP))   // controls  FOR
DCV (FOR PNEUMATIC CYLINDER)
  {
   digitalWrite(dcv1,HIGH);
   digitalWrite(dcv2,LOW);
  }
else if (ps2x.Button(PSB_PAD_DOWN))
  {
    digitalWrite(dcv2,HIGH);
    digitalWrite(dcv1,LOW);
  }
else
  {
    digitalWrite(dcv1,LOW);
    digitalWrite(dcv2,LOW);
```

```
  }


   delay(50);
  //}
}


void move_forward()
{
  digitalWrite(dir2, LOW);
  digitalWrite(dir4, HIGH);
  mapped = map(ps2x.Analog(PSS_LY), 120, 0,
0,pwm);
  analogWrite(pwm2, mapped);
  analogWrite(pwm4, mapped);
}

void move_backward()
{
  digitalWrite(dir2,HIGH);
  digitalWrite(dir4,LOW);
  mapped = map(ps2x.Analog(PSS_LY), 140, 255,
0,pwm);
  analogWrite(pwm2, mapped);
  analogWrite(pwm4, mapped);
}

void move_right()
{
  digitalWrite(dir1, HIGH);
  digitalWrite(dir3, LOW);
  mapped = map(ps2x.Analog(PSS_RX), 140, 255,
0,pwm);
  analogWrite(pwm1, mapped);
  analogWrite(pwm3, mapped);
}

void move_left()
{
  digitalWrite(dir1, LOW);
  digitalWrite(dir3, HIGH);
  mapped = map(ps2x.Analog(PSS_RX), 120, 0,
0,pwm);
  analogWrite(pwm1, mapped);
  analogWrite(pwm3, mapped);
}

void move_anticlockwise()
{
  digitalWrite(dir1, HIGH);
  digitalWrite(dir2, HIGH);
  digitalWrite(dir3, HIGH);
  digitalWrite(dir4, HIGH);
  mapped = map(ps2x.Analog(PSS_RX), 120, 0, 0,
100);
  analogWrite(pwm1, mapped);
  analogWrite(pwm2, mapped);
  analogWrite(pwm3, mapped);
  analogWrite(pwm4, mapped);
}

void move_clockwise()
{
  digitalWrite(dir1, LOW);
  digitalWrite(dir2, LOW);
  digitalWrite(dir3, LOW);
  digitalWrite(dir4, LOW);
  mapped = map(ps2x.Analog(PSS_RX), 140, 255, 0,
100);
  analogWrite(pwm1, mapped);
  analogWrite(pwm2, mapped);
  analogWrite(pwm3, mapped);
  analogWrite(pwm4, mapped);
}

void initial_condition()
{
  analogWrite(pwm1, 0);
  analogWrite(pwm2, 0);
  analogWrite(pwm3, 0);
  analogWrite(pwm4, 0);
  digitalWrite(dir1, LOW);
  digitalWrite(dir2, LOW);
  digitalWrite(dir3, LOW);
  digitalWrite(dir4, LOW);
}


void release_servo1() {
  Servo1.write(45);                    // release from
servo1
  delay(300);
}

void release_servo2() {
  Servo2.write(135);                     // release from
servo2
  delay(300);
}

void release_servo3() {
  Servo3.write(45);                     // release from
servo3
  delay(300);
}

void release_servo4() {
```

```
  Servo4.write(45);                    // release from
servo4
  delay(300);
}

void load_shuttle() {
  Servo1.write(135);                   // load all the
shuttlecocks
  Servo2.write(45);
  Servo3.write(135);
  Servo4.write(135);
  delay(300);
}

void load_initialpostion() {
  Servo1.write(90);                  // intial position for
the comb
  Servo2.write(90);
  Servo3.write(90);
  Servo4.write(90);
  delay(300);
}
```

A.2 AUTONOMOUS BOT

A.2.1 INTELLIGENT MOVEMENT (Arduino Due)

```
//Initialisation of motor dir and pwm pins
int pwm1 = 12, pwm2 = 11, pwm3 = 10, pwm4 = 9;
int dir1 = 34, dir2 = 36, dir3 = 38, dir4 = 40;

int set = 0;
int h = 0;
int check = 0;
int initial = 0;
int retry = 0;
int stable = 0;

unsigned int flag = 0;
int flagzone = 0;
int Bot_state = 0;
int lz1 = 0;

// LSA1, LSA2, LSA3 junction count variables
unsigned int countJP1 = 0;
unsigned int countJP2 = 0;
unsigned int countJP3 = 0;

int tz1 = 42;
int tz2 = 44;
int tz3 = 46;

int throw1 = 25;
int throw2 = 31;
int throw3 = 29;
int thrown = 37;

//Initialisation of LSA1, LSA2, LSA3 RX, TX, Jpulse,
Serial encoder pins
byte rxLSA1 = 18, txLSA1 = 19, rxLSA2 = 16,
txLSA2 = 17, rxLSA3 = 14, txLSA3 = 15;
byte sEN_LSA1 = 24, sEN_LSA2 = 28,  sEN_LSA3 =
23;
byte jPulse_LSA1 = 22, jPulse_LSA2 = 30,
jPulse_LSA3 = 32;

//Initialisation of SET POINT, LSA Position Values,
error Values, Last Error
int setPoint = 35;
int posValLSA1 = 35, posValLSA2 = 35, posValLSA3
= 35;
int errorLSA1 = 0, errorLSA2 = 0, errorLSA3 = 0;
int lastError_LSA1 = 0, lastError_LSA2 = 0,
lastError_LSA3 = 0;

// Initialisation of PID constants
float kp = 10.5, kd = 7.0;          // PID constants of
LSA2, LSA3
float kp1 = 10.5, kd1 = 7.0;       // PID constants of
LSA1
float kp2 = 0.65, kd2 = 0.15;     // bot state = 1
float kp3 = 15.0, kd3 = 7.0;       //bot state = 4
float p = 0.0, d = 0.0;           // PID constants of LSA2,
LSA3
float p1 = 0.0, d1 = 0.0;         // PID constants of LSA1

// Initialisation of motor speed variables
int motorPWM1 = 0, motorPWM2 = 0, motorPWM3 =
0;

// Initialisation of address, command, data for LSA1,
LSA2, LSA3
char addressLSA1 = 0x01, addressLSA2 = 0x02,
addressLSA3 = 0x03;
char commandLSA1, dataLSA1, commandLSA2,
dataLSA2, commandLSA3, dataLSA3;

// Update commands for settings of LSA through codes
void sendCommand_LSA1(char, char, char);
void sendCommand_LSA2(char, char, char);
void sendCommand_LSA3(char, char, char);

// Declaration of line following functions
void correction_LSA2();
void correction_LSA3();
```

```
// For LSA1
void correction_LSAline();
void correction();

void rest();


void setup() {

  Serial.begin(230400);     //Serial monitor
  Serial1.begin(230400);    //LSA-1
  Serial2.begin(230400);    //LSA-3
  Serial3.begin(230400);    //LSA-2

  pinMode(tz1, INPUT);
  pinMode(tz2, INPUT);
  pinMode(tz3, INPUT);

  pinMode(throw1, OUTPUT);
  pinMode(throw2, OUTPUT);
  pinMode(throw3, OUTPUT);

  digitalWrite(throw1, LOW);
  digitalWrite(throw2, LOW);
  digitalWrite(throw3, LOW);

  pinMode(thrown, INPUT);

  pinMode(dir1, OUTPUT);
  pinMode(dir2, OUTPUT);
  pinMode(dir3, OUTPUT);
  pinMode(dir4, OUTPUT);

  pinMode(pwm1, OUTPUT);
  pinMode(pwm2, OUTPUT);
  pinMode(pwm3, OUTPUT);
  pinMode(pwm4, OUTPUT);

  pinMode(jPulse_LSA1, INPUT_PULLUP);
  pinMode(jPulse_LSA2, INPUT_PULLUP);
  pinMode(jPulse_LSA3, INPUT_PULLUP);

  pinMode(sEN_LSA1, OUTPUT);
  pinMode(sEN_LSA2, OUTPUT);
  pinMode(sEN_LSA3, OUTPUT);

  analogWrite(pwm1, 0);
  analogWrite(pwm2, 0);
  analogWrite(pwm3, 0);
  analogWrite(pwm4, 0);

  digitalWrite(dir1, LOW);
  digitalWrite(dir2, LOW);
  digitalWrite(dir3, LOW);
  digitalWrite(dir4, LOW);
  digitalWrite(jPulse_LSA3, LOW);
  digitalWrite(jPulse_LSA2, LOW);
  digitalWrite(jPulse_LSA1, LOW);

  digitalWrite(sEN_LSA1, HIGH);
  digitalWrite(sEN_LSA2, HIGH);
  digitalWrite(sEN_LSA3, HIGH);

  attachInterrupt(digitalPinToInterrupt(jPulse_LSA1),
count1, RISING);
  attachInterrupt(digitalPinToInterrupt(jPulse_LSA2),
count2, RISING);
  attachInterrupt(digitalPinToInterrupt(jPulse_LSA3),
count3, RISING);
  //  attachInterrupt(digitalPinToInterrupt(interrupt),
rest, RISING);

  commandLSA1 = 'D';
  dataLSA1 = 0x02;
  sendCommand_LSA1(commandLSA1, dataLSA1,
addressLSA1);

  commandLSA2 = 'D';
  dataLSA2 = 0x02;
  sendCommand_LSA2(commandLSA2, dataLSA2,
addressLSA2);

  commandLSA3 = 'D';
  dataLSA3 = 0x02;
  sendCommand_LSA3(commandLSA3, dataLSA3,
addressLSA3);

}

void sendCommand_LSA1(char command, char data,
char address) {
  char checksum = address + command + data;

  Serial1.write(address);
  Serial1.write(command);
  Serial1.write(data);
  Serial1.write(checksum);
}

void sendCommand_LSA3(char command, char data,
char address) {
  char checksum = address + command + data;

  Serial3.write(address);
  Serial3.write(command);
  Serial3.write(data);
  Serial3.write(checksum);
}
```

```
void sendCommand_LSA2(char command, char data,
char address) {
  char checksum = address + command + data;

  Serial2.write(address);
  Serial2.write(command);
  Serial2.write(data);
  Serial2.write(checksum);
}

void loop() {

  while (Serial.available() <= 0)
  {
    digitalWrite(sEN_LSA1, LOW);
    digitalWrite(sEN_LSA2, LOW);
    digitalWrite(sEN_LSA3, LOW);

    // Getting LSA data
////////////////////////////////////////////////////////////////////////////////
///////

    while ((Serial2.available() <= 0) &&
(Serial1.available() <= 0) && (Serial3.available() <=
0));
    {
      posValLSA1 = Serial1.read();
      posValLSA2 = Serial2.read();
      posValLSA3 = Serial3.read();

      /* Serial.print("LSA1->");      //For printing the
LSA values
        Serial.print(posValLSA1);
        Serial.print("LSA2->");
        Serial.println(posValLSA2);
        Serial.print("LSA3->");
        Serial.println(posValLSA3);*/
    }

////////////////////////////////////////////////////////////////////////////////
////////////////////////////
    digitalWrite(sEN_LSA1, HIGH);
    digitalWrite(sEN_LSA2, HIGH);
    digitalWrite(sEN_LSA3, HIGH);

    // If error occured in LSA position values, then
ignoure errors ////////////////////////////////////////////
    if (posValLSA1 != -1)
      errorLSA1 = posValLSA1 - setPoint;

    if (posValLSA2 != -1)
      errorLSA2 = posValLSA2 - setPoint;

    if (posValLSA3 != -1)
      errorLSA3 = posValLSA3 - setPoint;

////////////////////////////////////////////////////////////////////////////////
////////////////////////////

    // Calculate motor speed according to error
calculated  ////////////////////////////////////////////////
    if (errorLSA1 > 0)   {
      if (Bot_state == 0) {                              //
PID FOR LINE
        p1 = (errorLSA1 * kp1) + 70;
        d1 = (errorLSA1 - lastError_LSA1) * kd1;
        motorPWM1 = p1 + d1 ;
      }
      else if (Bot_state == 1) {                         //
PID FOR JUNCTION
        p1 = (errorLSA1 * kp2) + 70;
        d1 = (errorLSA1 - lastError_LSA1) * kd2;
        motorPWM1 = p1 + d1 ;
      }
      else if (Bot_state == 2) {                         //
PID FOR JUNCTION
        p1 = (errorLSA1 * kp2) + 70;
        d1 = (errorLSA1 - lastError_LSA1) * kd2;
        motorPWM1 = p1 + d1 ;
      }
      else if (Bot_state == 3) {                         //
PID FOR JUNCTION
        p1 = (errorLSA1 * kp1) + 50;
        d1 = (errorLSA1 - lastError_LSA1) * kd1;
        motorPWM1 = p1 + d1 ;
      }
      else if (Bot_state == 4) {                         //
PID FOR JUNCTION
        p1 = (errorLSA1 * kp3) + 50;
        d1 = (errorLSA1 - lastError_LSA1) * kd3;
        motorPWM1 = p1 + d1 ;
      }
      if ( motorPWM1 > 255)
        motorPWM1 = 255;
    }
    if (errorLSA1 < 0)   {
      if (Bot_state == 0) {
        p1 = ((-errorLSA1) * kp1) + 70;
        d1 = (errorLSA1 - lastError_LSA1) * kd1 ;
        motorPWM1 = p1 + d1;
      }
      else if (Bot_state == 1) {
        p1 = ((-errorLSA1) * kp2) + 70;
        d1 = (errorLSA1 - lastError_LSA1) * kd2 ;
        motorPWM1 = p1 + d1;
      }
      else if (Bot_state == 2) {                         //
PID FOR JUNCTION
        p1 = ((-errorLSA1) * kp2) + 70;
```

```
      d1 = (errorLSA1 - lastError_LSA1) * kd2;
      motorPWM1 = p1 + d1 ;
    }
    else if (Bot_state == 3) {                    //
PID FOR JUNCTION
      p1 = ((-errorLSA1) * kp1) + 50;
      d1 = (errorLSA1 - lastError_LSA1) * kd1;
      motorPWM1 = p1 + d1 ;
    }
    else if (Bot_state == 4) {                    //
PID FOR JUNCTION
      p1 = ((-errorLSA1) * kp3) + 50;
      d1 = (errorLSA1 - lastError_LSA1) * kd3;
      motorPWM1 = p1 + d1 ;
    }
    if ( motorPWM1 > 255)
      motorPWM1 = 255;
  }

  if (errorLSA2 > 0)   {
    if (Bot_state == 0) {
      p = (errorLSA2 * kp) + 20;
      d = (errorLSA2 - lastError_LSA2) * kd;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 1) {
      p = (errorLSA2 * kp2) + 40;
      d = (errorLSA2 - lastError_LSA2) * kd2;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 2) {
      p = (errorLSA2 * kp2) + 100;
      d = (errorLSA2 - lastError_LSA2) * kd2;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 3) {
      p = (errorLSA2 * kp) + 50;
      d = (errorLSA2 - lastError_LSA2) * kd;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 4) {
      p = (errorLSA2 * kp3) + 50;
      d = (errorLSA2 - lastError_LSA2) * kd3;
      motorPWM2 = p + d;
    }
    if ( motorPWM2 > 255)
      motorPWM2 = 255;
  }
  if (errorLSA2 < 0)    {
    if (Bot_state == 0) {
      p = ((-errorLSA2) * kp) + 120;
      d = (errorLSA2 - lastError_LSA2) * kd;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 1) {
      p = ((-errorLSA2) * kp2) + 40;
      d = (errorLSA2 - lastError_LSA2) * kd2;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 2) {
      p = ((-errorLSA2) * kp2) + 20;
      d = (errorLSA2 - lastError_LSA2) * kd2;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 3) {
      p = ((-errorLSA2) * kp) + 50;
      d = (errorLSA2 - lastError_LSA2) * kd;
      motorPWM2 = p + d;
    }
    else if (Bot_state == 4) {
      p = ((-errorLSA2) * kp3) + 50;
      d = (errorLSA2 - lastError_LSA2) * kd3;
      motorPWM2 = p + d;
    }

    if ( motorPWM2 > 255)
      motorPWM2 = 255;
  }
  if (errorLSA3 > 0)  {
    if (Bot_state == 0) {
      p = (errorLSA3 * kp) + 20;
      d = (errorLSA3 - lastError_LSA3) * kd;
      motorPWM3 = p + d ;
    }
    else if (Bot_state == 1) {
      p = (errorLSA3 * kp2) + 40;
      d = (errorLSA3 - lastError_LSA3) * kd2;
      motorPWM3 = p + d ;
    }
    else if (Bot_state == 2) {
      p = (errorLSA3 * kp2) + 100;
      d = (errorLSA3 - lastError_LSA3) * kd2;
      motorPWM3 = p + d ;
    }
    else if (Bot_state == 3) {
      p = (errorLSA3 * kp) + 50;
      d = (errorLSA3 - lastError_LSA3) * kd;
      motorPWM3 = p + d ;
    }
    else if (Bot_state == 4) {
      p = (errorLSA3 * kp3) + 50;
      d = (errorLSA3 - lastError_LSA3) * kd3;
      motorPWM3 = p + d ;
    }
    if ( motorPWM3 > 255)
      motorPWM3 = 255;
  }
  if (errorLSA3 < 0)   {
    if (Bot_state == 0) {
      p = ((-errorLSA3) * kp) + 120;
```

```
      d = (errorLSA3 - lastError_LSA3) * kd ;
      motorPWM3 = p + d;
    }
    else if (Bot_state == 1) {
      p = ((-errorLSA3) * kp2) + 40;
      d = (errorLSA3 - lastError_LSA3) * kd2 ;
      motorPWM3 = p + d;
    }
    else if (Bot_state == 2) {
      p = ((-errorLSA3) * kp2) + 100;
      d = (errorLSA3 - lastError_LSA3) * kd2 ;
      motorPWM3 = p + d;
    }
    else if (Bot_state == 3) {
      p = ((-errorLSA3) * kp) + 50;
      d = (errorLSA3 - lastError_LSA3) * kd ;
      motorPWM3 = p + d;
    }
    else if (Bot_state == 4) {
      p = ((-errorLSA3) * kp3) + 50;
      d = (errorLSA3 - lastError_LSA3) * kd3 ;
      motorPWM3 = p + d;
    }
    if ( motorPWM3 > 255)
      motorPWM3 = 255;
  }

////////////////////////////////////////////////////////////////////////////////
//////////////////////////

  if (initial == 0) {    //Delay time for Lsa to get started
    delay(1500);
    initial = 1;
  }

  // Start Position to Junction at Start Position
////////////////////////////////////////////////////////
  if (flag == 0)
  {
    analogWrite(pwm1, 120);
    digitalWrite(dir1, HIGH);
    analogWrite(pwm4, 120);
    digitalWrite(dir4, HIGH);
    // correction();
    correction_LSA2();
    correction_LSA3();
    // If junction after start is detected then, it will go
straight until values at LSA1 is not detected in range
    if (countJP2 >= 1)    {
      flag = 4;
    }
  }
  // Moves straight until values at LSA1 is not is range
////////////////////////////////////////////////////
  if (flag == 4)   {
    analogWrite(pwm1, 50);
    digitalWrite(dir1, HIGH);
    analogWrite(pwm4, 50); //70
    digitalWrite(dir4, HIGH);
    analogWrite(pwm2, 0);
    digitalWrite(dir3, LOW);
    analogWrite(pwm3, 0);
    digitalWrite(dir2, LOW);
    // void correction_LSA2temp();
    // Here flagzone is 1 means bot is going in TZ1 area
and bot moves throught single LSA area
    if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{//30
      flagzone = 1;
      flag = 17;
    }
  }
  //added on 23 feb
  if (flag == 17) {
    Bot_state = 1;
    correction();
    correction_LSA2();
    if ((errorLSA1 >= -3 && errorLSA1 <= 3) &&
(errorLSA2 >= -3 && errorLSA2 <= 3) ) {
      flag = 1;
    }
  }

////////////////////////////////////////////////////////////////////////////////
//////////////////////////

  // Bot is going for loading junction
////////////////////////////////////////////////////
  if (flag == 1)   {
    if (retry == 0) {
      Bot_state = 0;
      if (check == 0) {
        if ((errorLSA2 >= -35 && errorLSA2 <= 35)) {
          correction();
          analogWrite(pwm2, 220);  //180
          digitalWrite(dir2, LOW);
          analogWrite(pwm3, 220);
          digitalWrite(dir3, HIGH);
        }
        else {
          check = 1;
        }
      }
      if (check == 1 ) {
        correction_LSAline();
        analogWrite(pwm2, 240);
        digitalWrite(dir2, LOW);
        analogWrite(pwm3, 240);
        digitalWrite(dir3, HIGH);
      }
```

```
      if (check == 2 ) {
       correction();
       analogWrite(pwm2, 240);
       digitalWrite(dir2, LOW);
       analogWrite(pwm3, 240);
       digitalWrite(dir3, HIGH);
      }

      // --->  loading Junction 1   <---
      if (flagzone == 1) {
       if ((countJP1) >= 1)     {
        check = 3;
        correction();
        analogWrite(pwm2, 5);
        digitalWrite(dir2, LOW);
        analogWrite(pwm3, 5);
        digitalWrite(dir3, HIGH);
        if ((errorLSA3 >= -35 && errorLSA3 <= 35)
&& (errorLSA2 >= -35 && errorLSA2 <= 35) &&
(errorLSA1 >= -35 && errorLSA1 <= 35) )
         {
          analogWrite(pwm2, 65);
          digitalWrite(dir2, HIGH);
          analogWrite(pwm3, 65);
          digitalWrite(dir3, LOW);
          flag = 12;
         }
       }

      }

      // --->  loading Junction 2   <---
      if (flagzone == 2) {
       check = 2;
       if ((errorLSA2 >= -35 && errorLSA2 <= 35)
&& (errorLSA3 >= -35 && errorLSA3 <= 35)) {
        correction();
       }
       else {
        correction_LSAline();
       }
       if ((countJP1) >= 2)     {
        check = 3;
        analogWrite(pwm2, 50);
        digitalWrite(dir2, LOW);
        analogWrite(pwm3, 50);
        digitalWrite(dir3, HIGH);
        if ((errorLSA3 >= -35 && errorLSA3 <= 35)
&& (errorLSA2 >= -35 && errorLSA2 <= 35) &&
(errorLSA1 >= -35 && errorLSA1 <= 35) )
         {
          analogWrite(pwm2, 60);
          digitalWrite(dir2, HIGH);
          analogWrite(pwm3, 60);
          digitalWrite(dir3, LOW);
            flag = 12;
           }
          }
         }
        }

       //for retrying condition
       if (retry == 1) {
        check = 1;
        if (check == 1 ) {
         correction();
         Bot_state = 0;
         analogWrite(pwm2, 200); //240
         digitalWrite(dir2, LOW);
         analogWrite(pwm3, 200);
         digitalWrite(dir3, HIGH);
        }
        if ((countJP1) >= 2) {
         check = 2;
         if (check == 2) {
          analogWrite(pwm2, 90);
          digitalWrite(dir2, HIGH);
          analogWrite(pwm3, 90);
          digitalWrite(dir3, LOW);
          check = 3;
         }
         correction();
         analogWrite(pwm2, 50);//20
         digitalWrite(dir2, LOW);
         analogWrite(pwm3, 50);
         digitalWrite(dir3, HIGH);
         if ((errorLSA3 >= -35 && errorLSA3 <= 35)
&& (errorLSA2 >= -35 && errorLSA2 <= 35) )
         {
          flag = 18;
         }
        }
       }
      }

      if ( flag == 18)
      {
       correction();
       correction_LSA2();
       correction_LSA3();
       if ((errorLSA3 >= -10 && errorLSA3 <= 10) &&
(errorLSA2 >= -10 && errorLSA2 <= 10))       {
        Bot_state = 1;
        correction();
        correction_LSA2();
        correction_LSA3();
       }
       if ((errorLSA3 >= -4 && errorLSA3 <= 4) &&
(errorLSA2 >= -4 && errorLSA2 <= 4) &&
(errorLSA1 >= -4 && errorLSA1 <= 4) )    {
```

```
      Bot_state = 1;
      correction();
      correction_LSA2();
      correction_LSA3();
      flag = 5;
     }
    }

   //  #flag 12# -->   LSA correction at loading
Junctions 1 & 2   <--
   if ( flag == 12 )
   {
     correction();
     if ((errorLSA3 >= -35 && errorLSA3 <= 35) &&
(errorLSA2 >= -35 && errorLSA2 <= 35) &&
(errorLSA1 >= -35 && errorLSA1 <= 35) )
     {
      Bot_state = 0;
      correction();
      correction_LSA2();
      correction_LSA3();
      if ((errorLSA3 >= -10 && errorLSA3 <= 10) &&
(errorLSA2 >= -10 && errorLSA2 <= 10) &&
(errorLSA1 >= -10 && errorLSA1 <= 10) )        {
        Bot_state = 1;
        correction();
        correction_LSA2();
        correction_LSA3();
       }
      if ((errorLSA3 >= -5 && errorLSA3 <= 5) &&
(errorLSA2 >= -5 && errorLSA2 <= 5) &&
(errorLSA1 >= -5 && errorLSA1 <= 5) )        {
        Bot_state = 1;
        correction();
        correction_LSA2();
        correction_LSA3();
        check = 1;
        flag = 10;
       }
     }
   }
   ////////////////////////////////////////////////////////////////
```

```
   //  # flag 10 #  -->  Reads signal from pi to decide to
go to respective TZ on the basis of color of shuttlecock
<--

   if (flag == 10) {
     Bot_state = 3;      //fine correction >>>>>>> bot3
     correction();
     correction_LSA2();
     correction_LSA3();
     // loading for tz1 zone with blue shuttlecock
     if (digitalRead(tz1) == HIGH) {
      if (flagzone == 1) {
       flag = 5;
      }
      else if (flagzone == 2 || flagzone == 3) {
       flag = 7;
       retry = 0;
       check = 1;
       flagzone = 1;
       countJP1 = 1;
       countJP2 = 1;
       countJP3 = 0;
      }
     }
     // loading for tz2 zone with violet shuttlecock
     if (digitalRead(tz2) == HIGH) {
      if (flagzone == 2 || flagzone == 3) {
       flag = 5;
       flagzone = 2;
      }
      if (flagzone == 1) {
       flag = 1;
       flagzone = 2;
       retry = 1;
      }
     }
     // loading for t`z3 with golden shuttlecock
     if (digitalRead(tz3) == HIGH) {
      if (flagzone == 2 || flagzone == 3) {
       flag = 5;
       flagzone = 3;
      }
      if (flagzone == 1) {
       flag = 1;
       flagzone = 3;
       retry = 1;
      }
     }
    }

////////////////////////////////////////////////////////////////////////////
////////////////////////////////
```

```
   //    # flag 5 #   -->  bot is moving in the TZ, by line
following using LSA2 & LSA3
   if (flag == 5)   {
     Bot_state = 3;
     correction_LSA2();
     correction_LSA3();
     //  BOT moves towards TZ
     analogWrite(pwm1, 255);
     digitalWrite(dir1, HIGH);
     analogWrite(pwm4, 255);
```

```
    digitalWrite(dir4, HIGH);

    //  # flagzone 1 #  BOT moves towards TZ1
    if (flagzone == 1)     {
     if (countJP3 >= 1) {
       analogWrite(pwm1, 80);
       digitalWrite(dir1, HIGH);
       analogWrite(pwm4, 80);
       digitalWrite(dir4, HIGH);
       correction_LSA2();
       correction_LSA3();
     }
     if (countJP3 >= 2 || countJP2 >= 3)     {
       analogWrite(pwm1, 25);
       digitalWrite(dir1, HIGH);
       analogWrite(pwm4, 25);
       digitalWrite(dir4, HIGH);
       correction_LSA2();
       correction_LSA3();
       if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{
         flag = 3;
       }
     }
    }

    //  # flagzone 2 #  BOT moves towards TZ2
    if (flagzone == 2)     {
     if (countJP3 >= 1) {       // speed of the bot in
flagzone 2 and flagzone 3 is decreased linearly on 5th
feb
       analogWrite(pwm1, 100);
       digitalWrite(dir1, HIGH);
       analogWrite(pwm4, 100);
       digitalWrite(dir4, HIGH);
       correction_LSA2();
       correction_LSA3();
     }
     if (countJP3 >= 2 || countJP2 >= 3)      {
       analogWrite(pwm1, 20);//50
       digitalWrite(dir1, HIGH);
       analogWrite(pwm4, 20);
       digitalWrite(dir4, HIGH);
       if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{
         flag = 3;
       }
     }
    }

    //  # flagzone 3 #  BOT moves towards TZ3
    if (flagzone == 3)    {
     if (countJP3 >= 4) {
       analogWrite(pwm1, 100);
       digitalWrite(dir1, HIGH);
       analogWrite(pwm4, 100);
       digitalWrite(dir4, HIGH);
       correction_LSA2();
       correction_LSA3();
     }
     if (countJP3 >= 5 || countJP2 >= 6)     {
       analogWrite(pwm1, 30);//50
       digitalWrite(dir1, HIGH);
       analogWrite(pwm4, 30);
       digitalWrite(dir4, HIGH);
       if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{
         flag = 3;
       }
     }
    }
   }

////////////////////////////////////////////////////////////////////////////////
//////////////////////////


   //   # flag 3 #   --> LSA correction at TZ1, TZ2, TZ3
stop Junction   <--
   if (flag == 3)  {                //correction code, the
autonomous bot will correct itself before throwing the
shuttlecock
     correction();
     correction_LSA2();
     correction_LSA3();
     if ((errorLSA3 >= -5 && errorLSA3 <= 5) &&
(errorLSA2 >= -5 && errorLSA2 <= 5) &&
(errorLSA1 >= -5 && errorLSA1 <= 5))   {
       Bot_state = 1;
       correction();
       correction_LSA2();
       correction_LSA3();
       flag = 2;
     }
   }

////////////////////////////////////////////////////////////////////////////////
//////////////////////////

   //   # flag 2 #   --> BOT throws Shuttlecock by giving
signal to MEGA  <--
   if (flag == 2)    {
     correction();
     correction_LSA2();
     correction_LSA3();
     if (set == 0)     {
       if ((errorLSA3 >= -3 && errorLSA3 <= 3) &&
(errorLSA2 >= -3 && errorLSA2 <= 3) &&
(errorLSA1 >= -3 && errorLSA1 <= 3))      {
         correction();
```

```
      correction_LSA2();
      correction_LSA3();

      //  BOT throws blue shuttlecock
      if (flagzone == 1) {
        digitalWrite(throw1, HIGH);
        if (digitalRead(thrown) == HIGH)       {
          digitalWrite(throw1, LOW);
          countJP1 = 0;
          countJP2 = 0;
          countJP3 = 0;
          flag = 6;
        }
        set = 1;
      }

      //  BOT throws violet shuttlecock
      if (flagzone == 2) {
        digitalWrite(throw2, HIGH);
        if (digitalRead(thrown) == HIGH)    {
          digitalWrite(throw2, LOW);
          countJP1 = 0;
          countJP2 = 0;
          countJP3 = 0;
          flag = 6;
        }
        set = 1;
      }

      //  BOT throws golden shuttlecock
      if (flagzone == 3) {
        digitalWrite(throw3, HIGH);
        if (digitalRead(thrown) == HIGH)       {
          digitalWrite(throw3, LOW);
          countJP1 = 0;
          countJP2 = 0;
          countJP3 = 0;
          flag = 6;
        }
        set = 1;
      }
     }
    }
    if (set == 1) {
     if ((errorLSA3 >= -10 && errorLSA3 <= 10) &&
(errorLSA2 >= -10 && errorLSA2 <= 10) &&
(errorLSA1 >= -10 && errorLSA1 <= 10))       {
      correction();
      correction_LSA2();
      correction_LSA3();
      if (flagzone == 1) {
        if (digitalRead(thrown) == HIGH)       {
          digitalWrite(throw1, LOW);
          countJP1 = 0;
          countJP2 = 0;
          countJP3 = 0;
          flag = 6;
          set = 0;
        }
      }
      if (flagzone == 2) {
        if (digitalRead(thrown) == HIGH)       {
          digitalWrite(throw2, LOW);
          countJP1 = 0;
          countJP2 = 0;
          countJP3 = 0;
          flag = 6;
          set = 0;
        }
      }
      if (flagzone == 3) {
        if (digitalRead(thrown) == HIGH)       {
          digitalWrite(throw3, LOW);
          countJP1 = 0;
          countJP2 = 0;
          countJP3 = 0;
          flag = 6;
          set = 0;
        }
      }
     }
    }
   }

////////////////////////////////////////////////////////////////////////////////


   //   # flag 6 #   -->  BOT moves from TZ to loading
Junction 2  <--
   if (flag == 6)
   {
    Bot_state = 3;
    analogWrite(pwm1, 255);
    digitalWrite(dir1, LOW);
    analogWrite(pwm4, 255);
    digitalWrite(dir4, LOW);
    correction_LSA2();
    correction_LSA3();

    if (flagzone == 1)
    {
     if (countJP3 == 2)
     {
      correction_LSA2();
      correction_LSA3();
      analogWrite(pwm1, 100);
      digitalWrite(dir1, LOW);
      analogWrite(pwm4, 100);
      digitalWrite(dir4, LOW);
     }
```

```
      if (countJP2 >= 2)    {
       correction_LSA2();
       correction_LSA3();
       analogWrite(pwm1, 10);
       digitalWrite(dir1, LOW);
       analogWrite(pwm4, 10);
       digitalWrite(dir4, LOW);
       if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{
         flag = 16;
       }
      }
     }

     //   BOT moves from TZ2 to loading Junction 2
     if (flagzone == 2)     {
      if ((countJP3 >= 2) && (countJP2 < 2) )      {
       Bot_state = 3;
       correction_LSA2();
       correction_LSA3();
       //150 pwm and 11.8 V autoBot takes 8 sec for
tz2
       analogWrite(pwm1, 120);//200
       digitalWrite(dir1, LOW);
       analogWrite(pwm4, 120);//
       digitalWrite(dir4, LOW);
      }
      if (countJP2 >= 2)      {
       correction_LSA2();
       correction_LSA3();
       analogWrite(pwm1, 10);
       digitalWrite(dir1, LOW);
       analogWrite(pwm4, 10);
       digitalWrite(dir4, LOW);
       if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{
         flag = 14;
       }
      }
     }

     //   BOT moves from TZ3 to loading Junction 2
     if (flagzone == 3)   {

      if (countJP2 >= 4 && countJP3 >= 3)       {
       correction_LSA2();
       correction_LSA3();
       //150 pwm 11.7 V 10.5 sec tz3
       analogWrite(pwm1, 120);//90
       digitalWrite(dir1, LOW);
       analogWrite(pwm4, 120);
       digitalWrite(dir4, LOW);
      }
      //added so that BOT doesn't oscillates at LZ2
      if (countJP2 >= 5 && countJP3 >= 5)       {
       correction_LSA2();
       correction_LSA3();
       analogWrite(pwm1, 10);//0
       digitalWrite(dir1, LOW);
       analogWrite(pwm4, 10);
       digitalWrite(dir4, LOW);
       if ((errorLSA1 >= -35 && errorLSA1 <= 35))
{
         flag = 14;
       }
      }
     }
     ////////////////////////////////////////////////////////////////////
     if (flag == 16) {
      correction();
      correction_LSA2();
      correction_LSA3();
      if ((errorLSA3 >= -10 && errorLSA3 <= 10) &&
(errorLSA2 >= -10 && errorLSA2 <= 10) &&
(errorLSA1 >= -10 && errorLSA1 <= 10))       {
       Bot_state = 1;
       correction();
       correction_LSA2();
       correction_LSA3();
       if ((errorLSA3 >= -3 && errorLSA3 <= 3) &&
(errorLSA2 >= -3 && errorLSA2 <= 3) &&
(errorLSA1 >= -3 && errorLSA1 <= 3))       {
        countJP1 = 1;
        countJP2 = 1;
        countJP3 = 0;
        flag = 1;
        flagzone = 2;
       }
      }
     }
     //  # flag 14 #  -->  LSA correction at loading
Junction 2  <--
     if ( flag == 14 )
     {
      correction();
      correction_LSA2();
      correction_LSA3();
      if ((errorLSA3 >= -10 && errorLSA3 <= 10) &&
(errorLSA2 >= -10 && errorLSA2 <= 10) &&
(errorLSA1 >= -10 && errorLSA1 <= 10))      {
       Bot_state = 3;
       correction();
       Bot_state = 1;
       correction_LSA2();
       correction_LSA3();
       countJP1 = 1;
       countJP2 = 1;
       countJP3 = 0;
       flag = 10;
```

```
    }
   }
   ////////////////////////////////////////////////////////////////////


   //  # flag 7 #   -->    <--

   if (flag == 7)      // this flag is used for returning from
tz2 to tz1
   {
    Bot_state = 0;      //3
    analogWrite(pwm2, 155);
    digitalWrite(dir2, HIGH);
    analogWrite(pwm3, 155);
    digitalWrite(dir3, LOW);
    correction();
    if (countJP1 >= 2)
    {
     //   Bot_state = 3;
     analogWrite(pwm2, 155);
     digitalWrite(dir2, HIGH);
     analogWrite(pwm3, 155);
     digitalWrite(dir3, LOW);

     correction();
     /*     analogWrite(pwm2, 90);//200
         digitalWrite(dir2, HIGH);
         analogWrite(pwm3, 90);
         digitalWrite(dir3, LOW);
     */
     if ((errorLSA3 >= -35 && errorLSA3 <= 35) &&
(errorLSA2 >= -35 && errorLSA2 <= 35))
     {
      /*      analogWrite(pwm2, 50);//100
           digitalWrite(dir2, LOW);
           analogWrite(pwm3, 50);
           digitalWrite(dir3, HIGH);
      */     flag = 8;//15
     }
    }
   }
   ////////////////////////////////////////////////////////////////////


   //  # flag 15 #   -->    <--
   /*if ( flag == 15 )
   {
    if ((errorLSA3 >= -35 && errorLSA3 <= 35) &&
(errorLSA2 >= -35 && errorLSA2 <= 35))
    {
     Bot_state = 0;  //3
     correction();
     correction_LSA2();
     correction_LSA3();
     /*     analogWrite(pwm2, 40);//100
         digitalWrite(dir2, LOW);
         analogWrite(pwm3, 40);
         digitalWrite(dir3, HIGH);
     *///     flag = 8;
  ////
//   */

////////////////////////////////////////////////////////////////////////////


   //   # flag 8 #   -->   <--
   if (flag == 8) {
    //Bot_state = 0;     //0      //added on 20 feb
    correction();
    correction_LSA2();
    correction_LSA3();
    if ((errorLSA3 >= -35 && errorLSA3 <= 35) &&
(errorLSA2 >= -35 && errorLSA2 <= 35) &&
(errorLSA1 >= -35 && errorLSA1 <= 35)) {
     Bot_state = 0;    //2,     //0
     correction();
     correction_LSA2();
     correction_LSA3();

     if ((errorLSA3 >= -10 && errorLSA3 <= 10) &&
(errorLSA2 >= -10 && errorLSA2 <= 10) &&
(errorLSA1 >= -10 && errorLSA1 <= 10)) {
      Bot_state = 0;      //1
      correction();
      correction_LSA2();
      correction_LSA3();
     }
     if ((errorLSA3 >= -3 && errorLSA3 <= 3) &&
(errorLSA2 >= -2 && errorLSA2 <= 2) &&
(errorLSA1 >= -1 && errorLSA1 <= 1)) {  //changed
from 4 to 5
      Bot_state = 1;    //1,//0
      correction();
      correction_LSA2();
      correction_LSA3();
      flag = 100;     //5
     }
    }
   }
   if(flag == 100)
  {
     if ((errorLSA3 >= -1 && errorLSA3 <= 1) &&
(errorLSA2 >= -2 && errorLSA2 <= 2) &&
(errorLSA1 >= -3 && errorLSA1 <= 3)) {  //changed
from 4 to 5
      Bot_state = 1;    //1,//0
      correction();
```

```
        correction_LSA2();
        correction_LSA3();
        flag = 5;
       }
  }

  }
  digitalWrite(throw1, LOW);
}



void count1() {
  countJP1++;
}
void count2() {
  countJP2++;
}
void count3() {
  countJP3++;
}



void correction_LSA1() {

  if ((errorLSA1 > -4) && (errorLSA1 < 4 )) {
    lastError_LSA1 = errorLSA1;
    analogWrite(pwm1, 0);
  }
  else if ((errorLSA1 >= 4) && (errorLSA1 <= 35))  {
    lastError_LSA1 = errorLSA1;
    digitalWrite(dir1, LOW);
    analogWrite(pwm1, motorPWM1);
  }
  else if ((errorLSA1 <= -4) && (errorLSA1 >= -35))  {
    lastError_LSA1 = errorLSA1;
    digitalWrite(dir1, HIGH);
    analogWrite(pwm1, motorPWM1);
  }
  else if (errorLSA1 > 215)  {
    if (lastError_LSA1 < 0)   {
      analogWrite(pwm1, 160);
      digitalWrite(dir1, HIGH);
    }
    else    {
      analogWrite(pwm1, 160);
      digitalWrite(dir1, LOW);
    }
  }
}


void correction_LSA2() {
  // HIGH <-->LOW when the pcb and motors was not
alined as expected
  if ((errorLSA2 > -4) && (errorLSA2 < 4))  {
    lastError_LSA2 = errorLSA2;
    analogWrite(pwm2, 0);
  }
  else if ((errorLSA2 >= 4) && (errorLSA2 <= 35)) {
    lastError_LSA2 = errorLSA2;
    digitalWrite(dir2, LOW);
    analogWrite(pwm2, motorPWM2);
  }
  else if ((errorLSA2 <= -4) && (errorLSA2 >= -35))  {
    lastError_LSA2 = errorLSA2;
    digitalWrite(dir2, HIGH);
    analogWrite(pwm2, motorPWM2);
  }
  else if (errorLSA2 > 215)  {
    if (lastError_LSA2 < 0)   {
      analogWrite(pwm2, 160);
      digitalWrite(dir2, HIGH);
    }
    else   {
      analogWrite(pwm2, 160);  //160
      digitalWrite(dir2, LOW);
    }
  }
}

void correction_LSA3() {
  if ((errorLSA3 > -4) && (errorLSA3 < 4))  {
    lastError_LSA3 = errorLSA3;
    analogWrite(pwm3, 0);
  }
  else if ((errorLSA3 >= 4) && (errorLSA3 <= 35))  {
    lastError_LSA3 = errorLSA3;
    digitalWrite(dir3, HIGH);
    analogWrite(pwm3, motorPWM3);
  }
  else if ((errorLSA3 <= -4) && (errorLSA3 >= -35))  {
    lastError_LSA3 = errorLSA3;
    digitalWrite(dir3, LOW);
    analogWrite(pwm3, motorPWM3);
  }
  else if (errorLSA3 > 215) {
    if (lastError_LSA3 < 0)  {
      analogWrite(pwm3, 160);
      digitalWrite(dir3, LOW);
    }
    else    {
      analogWrite(pwm3, 160);
      digitalWrite(dir3, HIGH);
    }
  }
}
```

```
void correction() {// HIGH TO LOW

 if ((errorLSA1 > -4) && (errorLSA1 < 4))  {
  lastError_LSA1 = errorLSA1;
  analogWrite(pwm1, 0);
  analogWrite(pwm4, 0);
 }
 else if ((errorLSA1 >= 4) && (errorLSA1 <= 35))  {
  lastError_LSA1 = errorLSA1;
  digitalWrite(dir1, HIGH);
  analogWrite(pwm1, motorPWM1);
  digitalWrite(dir4, HIGH);
  analogWrite(pwm4, motorPWM1);
 }
 else if ((errorLSA1 <= -4) && (errorLSA1 >= -35))  {
  lastError_LSA1 = errorLSA1;
  digitalWrite(dir1, LOW);
  analogWrite(pwm1, motorPWM1);
  digitalWrite(dir4, LOW);
  analogWrite(pwm4, motorPWM1);
 }
 else if (errorLSA1 > 210)  {
  if (lastError_LSA1 < 0)    {
   analogWrite(pwm1, 160);
   digitalWrite(dir1, LOW);
   analogWrite(pwm4, 160);
   digitalWrite(dir4, LOW);
  }
  else    {
   analogWrite(pwm1, 160);
   digitalWrite(dir1, HIGH);
   analogWrite(pwm4, 160);
   digitalWrite(dir4, HIGH);
  }
 }
}


void correction_LSAline() {
 // HIGH TO LOW
 if ((errorLSA1 > -4) && (errorLSA1 < 4))  {
  lastError_LSA1 = errorLSA1;
  analogWrite(pwm1, 0);
  analogWrite(pwm4, 0);
 }
 else if ((errorLSA1 >= 4) && (errorLSA1 <= 35))  {
  lastError_LSA1 = errorLSA1;
  digitalWrite(dir1, HIGH);
  analogWrite(pwm1, motorPWM1);
  digitalWrite(dir4, LOW);
  analogWrite(pwm4, motorPWM1);
 }
 else if ((errorLSA1 <= -4) && (errorLSA1 >= -35)) {
  lastError_LSA1 = errorLSA1;
  digitalWrite(dir1, LOW);
  analogWrite(pwm1, motorPWM1);
  digitalWrite(dir4, HIGH);
  analogWrite(pwm4, motorPWM1);
 }
 else if (errorLSA1 > 210)  {
  if (lastError_LSA1 < 0)   {
   analogWrite(pwm1, 160);
   digitalWrite(dir1, LOW);
   analogWrite(pwm4, 160);
   digitalWrite(dir4, HIGH);
  }
  else  {
   analogWrite(pwm1, 160);
   digitalWrite(dir1, HIGH);
   analogWrite(pwm4, 160);
   digitalWrite(dir4, LOW);
  }
 }
}



int getJunction() {
 char address = 0x01;
 char command = 'X';
 char data = 0x01;
 char checksum = address + command + data;

 Serial1.write(address);
 Serial1.write(command);
 Serial1.write(data);
 Serial1.write(checksum);

 while (Serial1.available() <= 0);
 return (int(Serial1.read()));
}


void clearJunction() {
 char address = 0x01;
 char command = 'X';
 char data = 0x00;
 char checksum = address + command + data;

 Serial1.write(address);
 Serial1.write(command);
 Serial1.write(data);
 Serial1.write(checksum);
}
```

A.2.2 AUTONOMOUS BOT INTELLIGENT THROWING (Arduino Mega)

```
//TZ1 --> 140, 127  #23.1V
//TZ2 --> 130, 140  #23.2V
//TZ3 --> 140, 250

int throw1 = 25;
int throw2 = 31;
int throw3 = 29;
int led = 6; //23
int setp = 0;

int dir = 10;
int pwm = 9;
int pulses = 0;
int k = 0;
int flag = 0;
int encoderA = 21, encoderB = 20 ;

int thrown = 37;

void setup() {

  Serial.begin(9600);

  pinMode(dir, OUTPUT);
  pinMode(pwm, OUTPUT);

  pinMode(encoderA, INPUT);

  attachInterrupt(digitalPinToInterrupt(encoderA),
A_CHANGE, CHANGE);
  //attachInterrupt(digitalPinToInterrupt(encoderB),
B_CHANGE, CHANGE);

  pinMode(throw1, INPUT);
  pinMode(throw2, INPUT);
  pinMode(throw3, INPUT);
  pinMode(led, OUTPUT);
  digitalWrite(led, HIGH);
  pinMode(thrown, OUTPUT);
  digitalWrite(thrown, LOW);


}
void loop() {

  if ( digitalRead(throw1) == HIGH) {
    if (setp == 0)
    {
      digitalWrite(led, LOW);
      delay(1400); //1650
      digitalWrite(led, HIGH);
      setp = 1;
    }
    if (flag == 0)  {
      if (pulses < 137) {
        digitalWrite(dir, HIGH);
        analogWrite(pwm, 145); //145
      }
      else     {
        digitalWrite(dir, LOW);
        analogWrite(pwm, 0);
        flag = 1;
      }
    }
    if ( flag == 1 )  {
      if ( pulses > 30)    {
        digitalWrite(dir, LOW);
        analogWrite(pwm, 30);
      }
      else flag = 3;
    }
    if ( flag == 3)  {
      digitalWrite(dir, LOW);
      analogWrite(pwm, 0);
      digitalWrite(thrown, HIGH);
      flag = 0;
      delay(50);
      setp = 0;
      digitalWrite(thrown, LOW);                    //
SEND COMMAND TO DUE THAT
SHUTTLECOCK HAS BEEN THROWN

    }
  }

  else if ( digitalRead(throw2) == HIGH) {
    if (setp == 0)
    {
      digitalWrite(led, LOW);
      delay(1400); //1650
      digitalWrite(led, HIGH);
      setp = 1;
    }
    if (flag == 0)  {
      if (pulses < 135) { //133
        digitalWrite(dir, HIGH);
        analogWrite(pwm, 210); //210
      }
      else     {
        digitalWrite(dir, LOW);
        analogWrite(pwm, 0);
        flag = 1;
      }
    }
    if ( flag == 1 )  {
      if ( pulses > 30)    {
        digitalWrite(dir, LOW);
        analogWrite(pwm, 30);
```

```
    }
    else flag = 3;
   }
   if ( flag == 3)  {
    digitalWrite(dir, LOW);
    analogWrite(pwm, 0);
    digitalWrite(thrown, HIGH);
    flag = 0;
    setp = 0;
    delay(50);
    digitalWrite(thrown, LOW);                    //
SEND COMMAND TO DUE THAT
SHUTTLECOCK HAS BEEN THROWN
   }


 }
 else if ( digitalRead(throw3) == HIGH) {
  if (setp == 0)
  {
   digitalWrite(led, LOW);
   delay(1900); //1650
   digitalWrite(led, HIGH);
   setp = 1;
  }
  if (flag == 0)  {
   if (pulses < 123) {//125, 135,
    digitalWrite(dir, HIGH);
    analogWrite(pwm, 235);//240,230 //235
   }
   else    {
    digitalWrite(dir, LOW);
    analogWrite(pwm, 0);
    flag = 1;
   }
  }
  if ( flag == 1 )  {
   if ( pulses > 30)    {
    digitalWrite(dir, LOW);
    analogWrite(pwm, 30);
   }
   else flag = 3;
  }
  if ( flag == 3)  {
   digitalWrite(dir, LOW);
   analogWrite(pwm, 0);
   digitalWrite(thrown, HIGH);
   flag = 0;
   setp = 0;
   delay(50);
   digitalWrite(thrown, LOW);                    //
SEND COMMAND TO DUE THAT
SHUTTLECOCK HAS BEEN THROWN
  }
 digitalWrite(thrown, LOW);
 Serial.println(pulses);
}



void A_CHANGE() {
 //Interrupt function to read the x2 pulses of the
encoder.

 if ( digitalRead(encoderB) == 0 ) {
  if ( digitalRead(encoderA) == 0 ) {
   // A fell, B is low
   pulses++; // Moving forward
  } else {
   // A rose, B is high
   pulses--; // Moving reverse
  }
 }
 else {
  if ( digitalRead(encoderA) == 0 ) {
   pulses--; // Moving reverse
  } else {
   // A rose, B is low
   pulses++; // Moving forward
  }
 }
}
```

A.2.3 AUTOMOUS BOT IMAGE PROCESSING (Raspberry pi)

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

# USAGE
# python match.py --template cod_logo.png --images
images

# import the necessary packages

import sys
import numpy as np
import argparse
import imutils
import glob
import cv2
import time
import RPi.GPIO as GPIO

intTz1 = 11
```

```
intTz2 = 13
intTz3 = 15

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)
GPIO.setup(intTz1, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(intTz2, GPIO.OUT, initial=GPIO.LOW)
GPIO.setup(intTz3, GPIO.OUT, initial=GPIO.LOW)

j = 0
startX = 0
startY = 0
endX = 0
endY = 0
c = 100
c1 = 400
i = 0
cap = cv2.VideoCapture(0)
while(cap.isOpened() == False):
   cap.release()
   cap = cv2.VideoCapture(0)
ret, imageCam = cap.read()


while(True):



   if i == 0 :

      ret, imageCam = cap.read()

      if j == 0 :
         template = cv2.imread('image.png')
         template = cv2.cvtColor(template,
cv2.COLOR_BGR2GRAY)
         template = cv2.Canny(template, 50, 200)
         (tH, tW) = template.shape[:2]
         #cv2.imshow('Template', template)
         image = imageCam
         gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
         found = None

         for scale in np.linspace(0.2, 1.0, 20)[::-1]:

         # resize the image according to the scale, and
keep track
         # of the ratio of the resizing

            resized = imutils.resize(gray,
width=int(gray.shape[1] * scale))
            r = gray.shape[1] / float(resized.shape[1])

            # if the resized image is smaller than the
template, then break
            # from the loop

            if resized.shape[0] < tH or resized.shape[1] <
tW:
               break

            # detect edges in the resized, grayscale image
and apply template
            # matching to find the template in the image

            edged = cv2.Canny(resized, 50, 200)
            result = cv2.matchTemplate(edged, template,
cv2.TM_CCOEFF)
            (_, maxVal, _, maxLoc) =
cv2.minMaxLoc(result)

            # if we have found a new maximum
correlation value, then ipdate
            # the bookkeeping variable
            if found is None or maxVal > found[0]:
               found = (maxVal, maxLoc, r)
         # unpack the bookkeeping varaible and
compute the (x, y) coordinates
         # of the bounding box based on the resized ratio

         (_, maxLoc, r) = found
         (startX, startY) = (int(maxLoc[0] * r),
int(maxLoc[1] * r))
         (endX, endY) = (int((maxLoc[0] + tW) * r),
int((maxLoc[1] + tH)* r))

         # draw a bounding box around the detected
result and display the image
         j = 1

      #cv2.rectangle(image, (startX, startY), (endX,
endY), (0, 0, 255), 2)
      #cv2.imwrite('Image.jpg', image)
      #

      imagebl = imageCam
      imagebl = imagebl[startY:endY+20, startX-
10:endX+10]
      blur=cv2.blur(imagebl,(5,5))
      blur=cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)
      lower = np.array([0,196,98], dtype="uint8")
      upper = np.array([12,228,255], dtype="uint8")
      thresh = cv2.inRange(blur, lower, upper)
      thresh2 = thresh.copy()
      imagebl, contours, hierarchy =
cv2.findContours(thresh, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
```

```
        max_area = 0
        best_cnt = 1

        for cnt in contours:
            area = cv2.contourArea(cnt)
            if area > max_area:
                max_area = area
                best_cnt = cnt


        M = cv2.moments(best_cnt)
        cx, cy = int(M['m10']/M['m00']),
int(M['m01']/M['m00'])
        #cv2.circle(blur,(cx,cy),10,(0,0,255),-1)
        #cv2.imwrite("Frameblur.jpg", blur)
        if (cx != 0 and cy != 0) :

            if (cx+startX >= startX and cx+startX <= endX)
and (cy+startY >= startY and cy+startY <= endY) :
                print("Loaded")

                imageuse =
imageCam[cy+startY:cy+c1+startY, cx-
c+30+startX:cx+c+startX]
                #cv2.imwrite("roi.jpg", imageuse)

                blurS=cv2.blur(imageuse,(5,5))
                blurS=cv2.cvtColor(blurS,
cv2.COLOR_BGR2HSV)
                lowerB = np.array([105,179,110],
dtype="uint8")#Blue Shuttlecock
                upperB = np.array([140,255,255],
dtype="uint8")
                lowerBr = np.array([10,100,20],
dtype="uint8")#brown Shuttlecock
                upperBr = np.array([20,255,200],
dtype="uint8")
                lowerG = np.array([20,183,158],
dtype="uint8")#Golden Shuttlecock
                upperG = np.array([35,255,255],
dtype="uint8")

                threshB = cv2.inRange(blurS, lowerB,
upperB)
                threshB2 = threshB.copy()
                #cv2.imshow("fram", threshB2)
                threshBr = cv2.inRange(blurS, lowerBr,
upperBr)
                threshBr2 = threshBr.copy()
                threshG = cv2.inRange(blurS, lowerG,
upperG)
                threshG2 = threshG.copy()

                image_B, contours_B, hierarchy_B =
cv2.findContours(threshB, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
                image_Br, contours_Br, hierarchy_Br =
cv2.findContours(threshBr, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)
                image_G, contours_G, hierarchy_G =
cv2.findContours(threshG, cv2.RETR_LIST,
cv2.CHAIN_APPROX_SIMPLE)

                max_area_B = 0
                best_cnt_B = 1

                max_area_Br = 0
                best_cnt_Br = 1

                max_area_G = 0
                best_cnt_G = 1

                for cnt_B in contours_B:
                    area_B = cv2.contourArea(cnt_B)
                    if area_B > max_area_B:
                        max_area_B = area_B
                        best_cnt_B = cnt_B
                for cnt_Br in contours_Br:
                    area_Br = cv2.contourArea(cnt_Br)
                    if area_Br > max_area_Br:
                        max_area_Br = area_Br
                        best_cnt_Br = cnt_Br
                for cnt_G in contours_G:
                    area_G = cv2.contourArea(cnt_G)
                    if area_G > max_area_G:
                        max_area_G = area_G
                        best_cnt_G = cnt_G
                M_B = cv2.moments(best_cnt_B)
                cx_B, cy_B = int(M_B['m10']/M_B['m00']),
int(M_B['m01']/M_B['m00'])

                M_Br = cv2.moments(best_cnt_Br)
                cx_Br, cy_Br =
int(M_Br['m10']/M_Br['m00']),
int(M_Br['m01']/M_Br['m00'])

                M_G = cv2.moments(best_cnt_G)
                cx_G, cy_G = int(M_G['m10']/M_G['m00']),
int(M_G['m01']/M_G['m00'])

                if(cx_B != 0 and cy_B !=0 ):
                    GPIO.output(intTz1, GPIO.HIGH)
                    print("TZ1")

                    sys.exit()
                '''if(cx_Br != 0 and cy_Br !=0 ):
                    GPIO.output(intTz2, GPIO.HIGH)
                    sys.exit()'''
```

```
            if(cx_G != 0 and cy_G !=0 ):
                GPIO.output(intTz3, GPIO.HIGH)
                print("TZ3")
                sys.exit()
            #cv2.waitKey(0) & 0xFF

        else :
            print("Unloaded")
        GPIO.output(intTz1, GPIO.LOW)
        GPIO.output(intTz2, GPIO.LOW)
        GPIO.output(intTz3, GPIO.LOW)

        #cv2.imwrite("final.jpg", imageCam)

        #if cv2.waitKey(1) & 0xFF == ord('q'):
            #break

cap.release()
cv2.destroyAllWindows()
```