A PROJECT REPORT ON COMPETITION ROBOCON'19

Faculty Advisors : Mr. RP Gohil, Dr. P.V. Timbadiya

MEMBERS

Abdullah Rangwala
Dhruv Patel
Himanshu Kumar
Manthan Shah
Sandeep Ashiwal
Vatsal Rathod
Devanshu Kanabar
Dhairya Vora
Dharmik Raval
Fenil Desai
Harshwardhan Bhangale
Mihir Chanpura
Mohit Gidwani
Neel Pachchigar
Neel Panji
Rutvik Zanzrukiya
Vivek Adajania

Acknowledgement

We, team DRISHTI would like to thank every individual and whole Drishti members in the successful happening of Robocon 2019 for our college. The team is thankful to the faculty advisors Mr. R.P. Gohil, Dr. P.V. Timbadiya and the Dean Academic of SVNIT for their kind support and guidance at every step. Special thanks to DRISHTI - A REVOLUTIONARY CONCEPT for providing lab workspace to the team.

Abstract

The document mainly focuses on the development, timeline ,methods and the challenges faced to develop the autonomous wheeled (Messenger Robot 1) and quadruped robot (Messenger Robot 2) by Team Drishti for Asia-Pacific Robot Contest 2019.

"GREAT URTUU"

Contents

1 Contest Theme

The concept of the contest:
The mission of the ABU Robocon 2019 Ulaanbaatar is to deliver information fast by using a relay messenger system - the Urtuu, which was first innovated in the world by the nomadic Mongolians. For exchanging information in a long distance, Mongolians had been using the Urtuu system as a messenger for rest (feeding, replacing a horse, etc.,), and in some cases, relay to another messenger. By using the Urtuu system, a messenger was able to travel in distance of 400 kilometres per day. At present days, we are going through massive and abrupt development of exchanging and sharing knowledge and information. This Urtuu system was an important invention that opened a new door for us to exchange and share the knowledge in regardless of space. Based on this concept, ABU Robocon 2019 Ulaanbaatar is designed to promote the idea of "Sharing the knowledge".
A match is between Red and Blue teams. It lasts three minutes at most. Each team has one manual robot known as Messenger-Robot 1, and one automatic robot known as Messenger-Robot 2. The automatic robot has four legs as of horses while wheels are not allowed. The manual robot carries the Gerege as a testimony from the Khangai urtuu, which is the the starting point. It goes along Forest, Bridge, and crosses the Line 1 next to Gobi urtuu, which is the starting point of the automatic robot. After Messenger-Robot 1 reaches Gobi urtuu, Messenger-Robot 1 passes Gerege to Messenger-Robot 2 at Gobi urtuu. Once Messenger-Robot 2 successfully receives Gerege, it can go along the Gobi area. Messenger-Robot 2 must go by four legs, like a horse, and cannot use wheels to move. Messenger-Robot 2 passes through Sand dune and Tussock, and directs to Mountain urtuu. After Messenger-Robot 2 reaches Mountain urtuu, Messenger-Robot 1 can enter Throwing zone to throw Shagai, and must earn 50 or more points. In case that Messenger-Robot 1 earns 50 or more points, Messenger-Robot 2 is allowed to climb the Mountain. Afterwards, if it reaches Uukhai zone and raises the Gerege first, the team is the winner, which is called "UUKHAI".

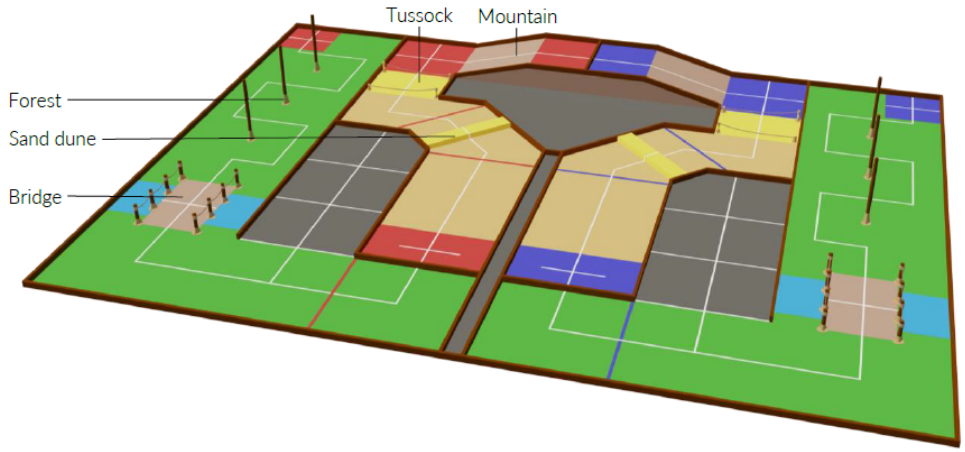


Figure 1: Arena

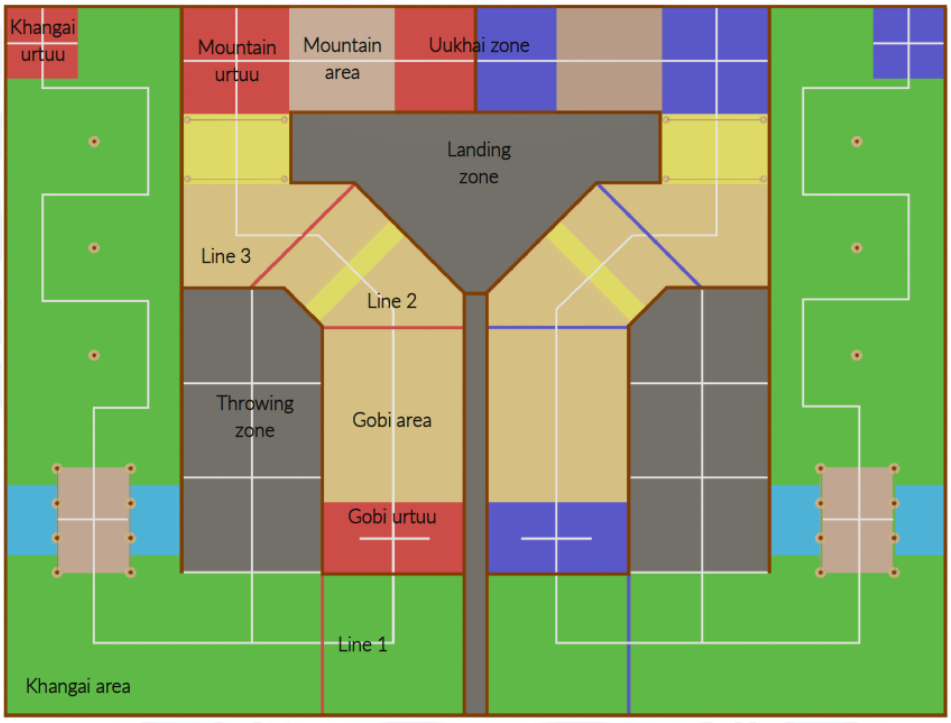


Figure 2: Top View

RULEBOOK

2 Protection Circuits

2.1 Polarity protection circuits

What is a reverse polarity protection circuit?

- Reverse voltage protection circuits prevent damage to power supplies and electronic circuits in the event of a reverse voltage applied at the input or output terminals. Reverse voltage protection is implemented at the input of the power supply or onboard of the custom, multiple output redundant power supplies. This is important in most electronic applications such as laptops, computers, CMOS circuits, etc.

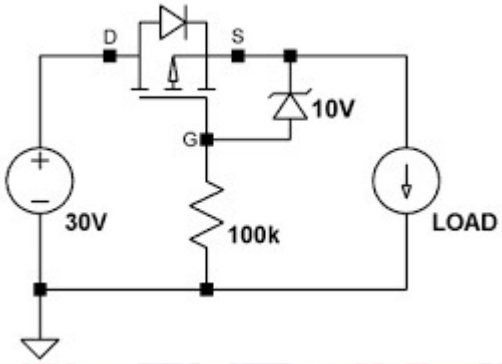
- The protection ensures that the components are not damaged by an accidental swap of the power supply connections. Various methods differ in operation, efficiency, and complexity. While some like a diode or circuit breaker provides only the reversal voltage protection, others such as the protection ICs provide the reverse voltage, over current, and overvoltage protection.

- To block negative voltages, designers usually place a power diode or a P-channel MOSFET in series with the power supply. One drawback of the series diode is that it takes up board space and has high power dissipation at high load currents.

- On the other hand, the MOSFET dissipates less power even though it requires an extra drive circuitry which increases the cost. Both solutions affect low power operations and especial the series diode. Also, the solutions may not be suitable at very high load currents.

Circuit Diagram :

- The circuit consists of a "p-type power MOSFET". The IC used as p-type power MOSFET is "IRF9540N". The ratings of the above mentioned component is as follows:
  $V_{DSS} = -100V$
  $R_{DS}(on) = 0.117\Omega$
  $I_D = -23A$



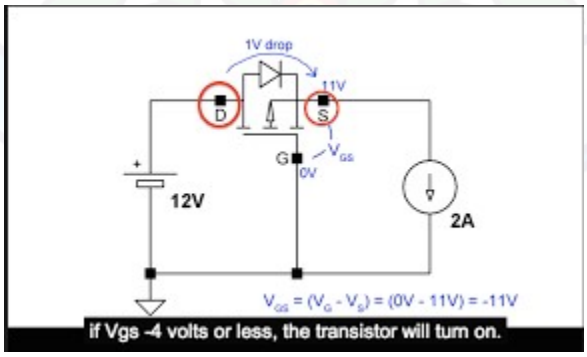
- The other major component used is a 10 v Zener diode . The Zener diode of rating 6.5-10 volts will work perfectly in the above circuit.The Zener diode implemented in the circuit has a rating of 8.2 V.

Working :

- **Correct Polarity Condition**
  If the polarity of the applied voltage is correct then the Source terminal is at $V_{app} - 1$ voltage potential because of 1 V drop in the power MOSFET. Where; V_{app} is the applied voltage.And the Gate terminal is at zero volt potential.Therefore,

$$V_{gs} = (V_g - V_s) = (0 - (V_{app} - 1)) = -(V_{app} - 1)$$

  Hence, when the V_{gs} is less than -4V the power MOSFET starts conducting and thus we get the output the same as that of V_{app}.
  NOTE - If the V_{gs} is more than -4 V the power MOSFET does not conduct and we receive ideally 0(zero) volts at the output.



- **Incorrect Polarity Condition**
  When we apply the V_{app} in reverse polarity to the circuit the Drain terminal is at zero potential and thus the Source terminal is at -1V potential.And the Gate the gate terminal is at V_{app} voltage potential. Therefore ,

$$V_{gs} = (V_g - V_s) = (V_{app} – (-1)) = V_{app} + 1$$

  Hence, as the V_{gs} is greater than -4V the current does not flow through the MOSFET and we get o(zero) volts at the output.

2.2 Over-voltage protection circuits

What is a Over-voltage protection circuit?

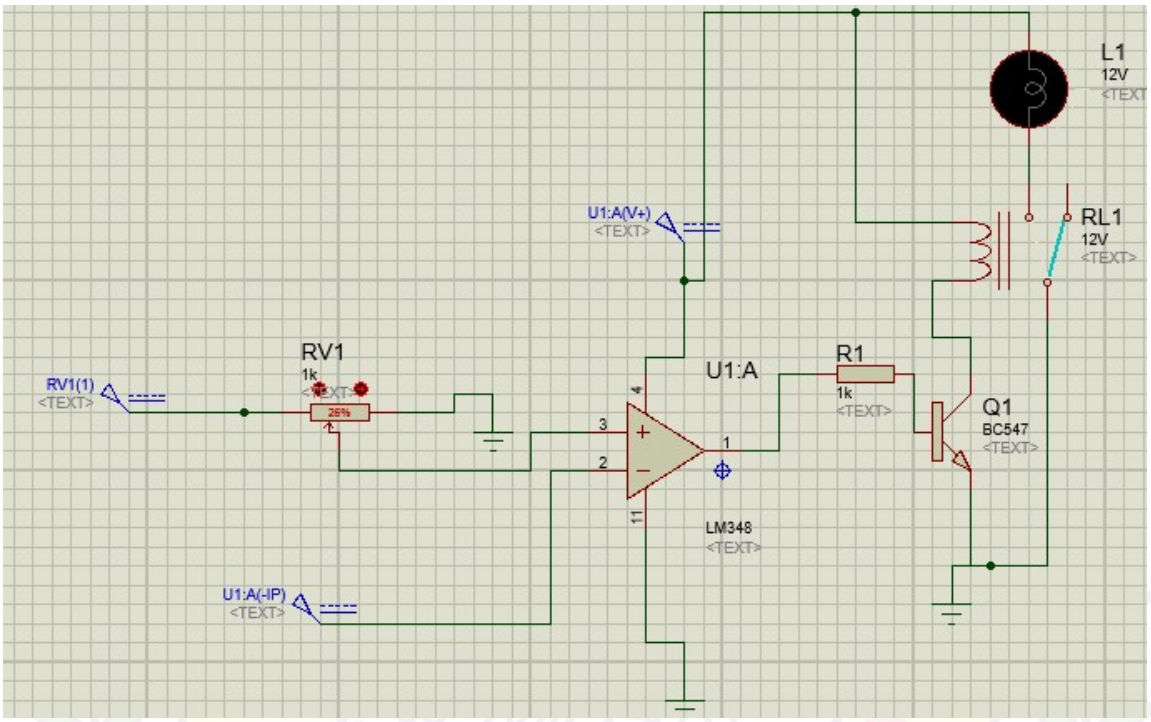
- For the satisfactory working of all electrical and electronic devices, it is recommended to allow voltage at prescribed limits. Voltage fluctuations in electric power supply certainly have adverse effects on connected loads. These fluctuations can be of over voltage and under voltages which are caused by several reasons like voltage surges, lightning, overload, etc. Over-voltages are the voltages that exceed the normal or rated values which cause insulation damage to electrical appliances leading to short circuits.

Use in ROBOCON :

- Though we have dedicated circuits for converting the 12V to 5 Volts and supply to different components of the robot, still there were some glitches due to which at times previous year failures in other circuits and even microcontrollers were observed. One cannot afford such glitches during a competition which may hamper your chance to win and dream of representing India at the international level. So to protect the other circuitries and microcontroller an extra layer of protection in the form of the over-voltage protection circuit is applied which will cut-off the rest of the robot from the main supply once over-voltage scenario is observed. Here we have designed the circuit using LM324N Op-amp. The circuit can be build using timers also and many more ways.

Circuit Diagram :

- **Material Used:**
  LM324N
  Relay Module of DC-12V rating
  7805 voltage regulator
  BC547 transistor
  1KΩ Resistor



Working :

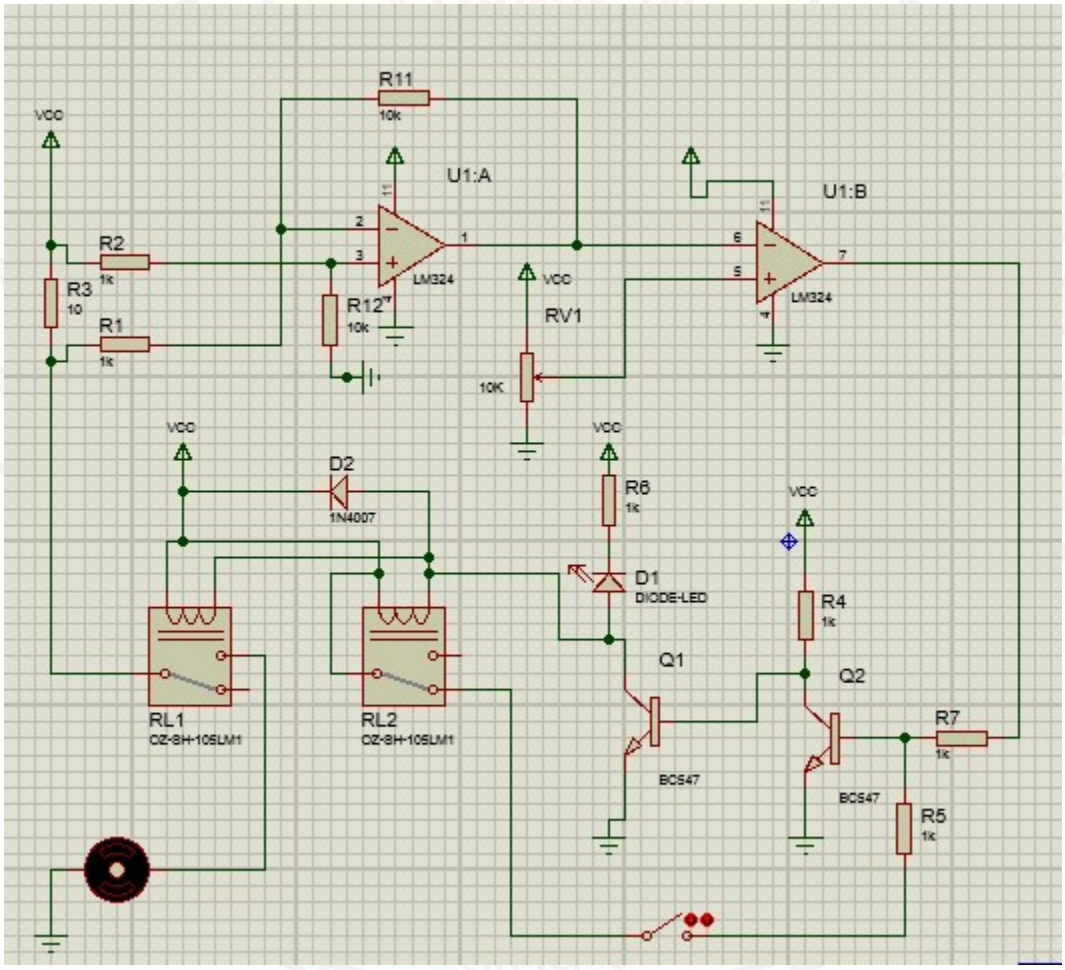
- The 1st step involves setting a reference voltage of 5 Volt to be supplied to the LM324N op-amp which is a non-inverting amplifier, which means the main voltage will be supplied to the +ve terminal and reference to –ve terminal. the output will be a high impedance state when the voltage on the positive terminal or the non-inverting terminal is greater than the reference voltage provided.
- During normal operation when voltage is at a safe level, here we have used a relay in Normal Closed mode, which means that during normal operations the circuit will remain closed.
- When the high impedance state is observed, the coil of the relay module gets energized and shifts to Normal Open terminal and thus the load is separated from the main supply and protected.

2.3 Short-Circuit protection circuit

What is a Short-Circuit protection circuit?

- The circuit is designed in such a way, such that it can protect the main circuit in case of a short circuit (by cutting off the main supply). This can be achieved with the help of relays, transistors, op-amp, etc.

Circuit Diagram :



- **Material Used:** 1x relay(with 2 changeover contacts) or 2 x relay(with single changeover contacts)
  1 x LM358 OpAmp or LM324N OpAmp
  2 x BC547 Transistor

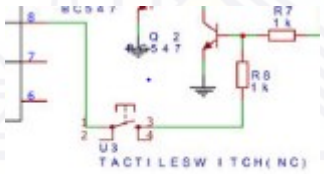
1 x Potentiometer
1 x LED
1 x 1N4007 Diode
Resistors
Power Resistor – (.1 ohm), here we used 10-ohm resistor, but it pretty much solves the purpose for low current circuits.

Working :

- The above circuit is attached in series with a given load. .if any short circuit occurs, we switch off the main supply line, with the help of a relay.
- We are using a power resistor of very small resistance because of this reason only. When a short circuit occurs, the value of current flowing through the mainline increases. The power resistance will let the high current pass through it. This increases the voltage drop on the power resistor. This drop is very low, and hence it is tough to identify the voltage drop directly. Hence we use an OpAmp as an amplifier. This amplifies the voltage drop across the power resistor. This voltage signal is then fed into a comparator(OpAmp being used as a comparator). The reference voltage of the op-amp is to be set in such a way, that it allows the signal to pass, as soon as there is a short circuit. Hence we give the reference voltage on the non-inverting (positive) terminal of OpAmp, and output from the amplifier into the inverting terminal of the comparator. This creates the necessary condition for the functioning of our further circuit.
- So, according to our current progress, we will get a signal from the upper part of the circuit, as soon as we encounter a short circuit. This signal enters the base of the first transistor and results in it going in on state. So, before the short circuit, the transistor was in the off state, resulting in the current not flowing through the emitter. Hence the VCC given on the collector of the first transistor enters into the base of the next transistor. This turns the transistor on and results in the flowing of current through the emitter of the second transistor. Hence the LED is turned on.
- The relay plays the vital role of switching the mainline off, as soon as the short circuit occurs. The series of transistors make such a connection, that as soon as the first transistor receives the signal, the second transistor switches off, making a current pass through the internal coil of a relay, which makes the

relay to switch into the normally open case. Hence we had made the relay connections in such a way, that as soon as the second transistor turns off, the relay switches, breaking the mainline.

- Now we face an issue. As soon as there is a short circuit, it switches off the mainline, and hence the effect of short circuit is over. Hence circuit behaves as if there is no short circuit, and the mainline is again switched on, but cause there is already a short circuit somewhere, it again acts as if there is a short circuit, which results in again switching off of relay. This will create a flickering effect in the circuit,(ie. The relay switches on and off again and again). For this, we use tactile switching. The below figure shows the connections.

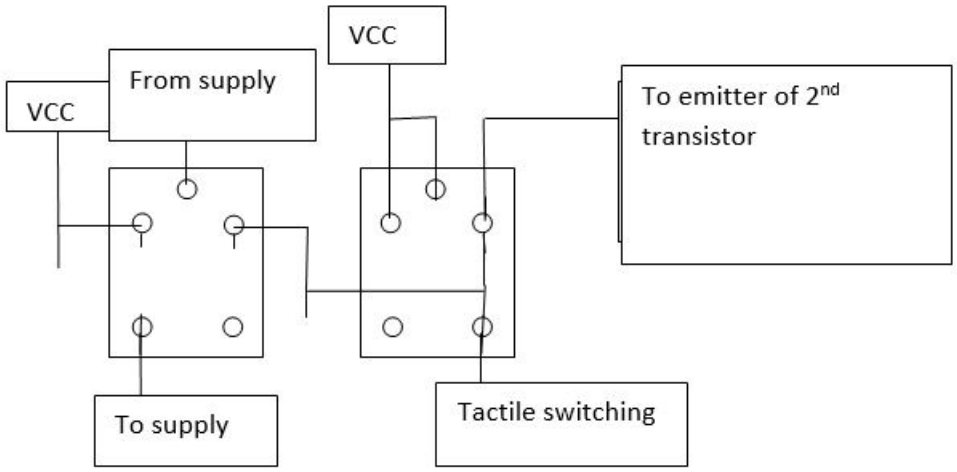


- As soon as there is switching on of relay, it completes the circuit and this creates a constant VCC at base of the first transistor, which is independent of the above amplifier and comparator circuit. Therefore, once there is a short circuit, it results in the occurrence of this phenomenon, which resolves our problem of relay switch flickering.

Problems & Solution :

1. **Problem :**The above circuit has a relay with 2 changeover contacts, while we had a single changeover contact relay switch available with us.

   Solution :we used the relays in a way as shown below.



2. While using the potentiometer, check that you are using the correct ranged potentiometer(ie. 10 K ohm). And check that you are using the correct ranged potentiometer(ie. 10 K ohm). And you need to set the reference value of potentiometer by trial and error.

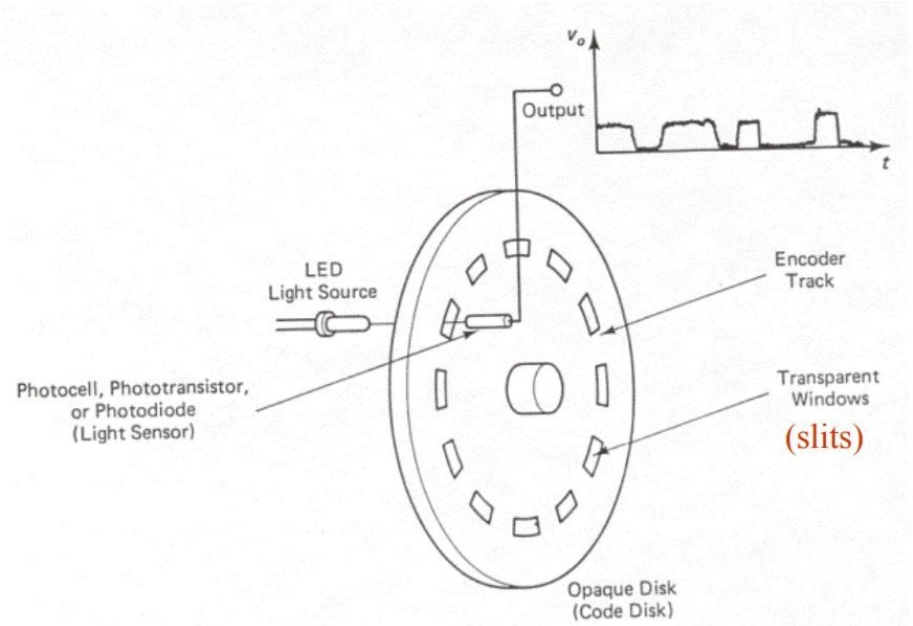
3. As we are using different power resistor, hence the voltage after amplification might have a different value. So check its output value at the time of short circuit, so that we can use that value as our reference voltage value, for OpAmp as a comparator. Hence we used that voltage value as output from out potentiometer.

4. We had a 10-ohm power resistor, and hence we had to change the value of two 20k ohm resistors.

3 Sensors & ICs

3.1 Encoder

- Any transducer that generates a coded reading of a measurement can be termed an encoder.
- Shaft Encoders are digital transducers that are used for measuring angular displacements and velocities.
- Shaft Encoders can be classified into two categories depending on the nature and method of interpretation of the output :
  1. Incremental Encoders
  2. Absolute Encoders
- **Incremental Encoders**
  - Output is a pulse signal that is generated when the transducer disk rotates as a result of the motion that is being measured.
  - By counting pulses or by timing the pulse width using a clock signal, both angular displacement and angular velocity can be determined. – Displacement, however, is obtained with respect to some reference point on the disk, as indicated by a reference pulse (index pulse) generated at that location on the disk. The index pulse count determines the number of full revolutions.
- **Elements of the Optical Encoder**
  - The optical encoder uses an opaque disk (code disk) that has one or more circular tracks, with some arrangement of identical transparent windows (slits) in each track.



Schematic Representation of an Optical Encoder
One Track and One Pick-Off Sensor Shown

- A parallel beam of light (e.g., from a set of light emitting diodes) is projected to all tracks from one side of the disk. The transmitted light is picked off using a bank of photosensors on the other side of the disk that typically has one sensor for each track. The light sensor could be a silicon photodiode, a phototransistor, or a photo-voltaic cell.
- – Since the light from the source is interrupted by the opaque areas of the track, the output signal from the probe is a series of voltage pulses. This signal can be interpreted to obtain the angular position and angular velocity of the disk.
- Note that an incremental encoder disk requires only one primary track that has equally spaced and identical window (pick-off) areas. The window area is equal to the area of the inter-window gap.
- Usually, a reference track that has just one window is also present in order to generate a pulse (known as the index pulse) to initiate pulse counting for angular position measurement and to detect complete revolutions.
- In contrast, absolute encoder disks have several rows of tracks, equal in number to the bit size of the output data word. Furthermore, the track

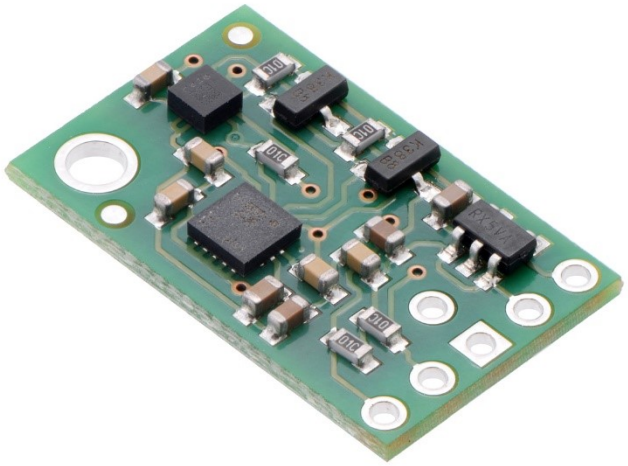
windows are not equally spaced but are arranged in a specific pattern on each track so as to obtain a binary code (or gray code) for the output data from the transducer.

- Some designs of incremental encoders have two identical tracks, one a quarter-pitch offset from the other, and the two pick-off sensors are placed radially without any circumferential offset, i.e., the offset track configuration. A pick-off sensor for a reference pulse is also used.

- Signal interpretation depends on whether the particular optical encoder is an incremental device or an absolute device. We will focus on the incremental optical encoder.
- The output signals from either the offset sensor configuration or the offset track configuration are the same.
- Note that the pulse width and pulse-to-pulse period (encoder cycle) are constant in each sensor output when the disk rotates at constant angular velocity. When the disk accelerates, the pulse width decreases continuously; when the disk decelerates, the pulse width increases continuously.
  – The quarter-pitch offset in sensor location or track position is used to determine the direction of rotation of the disk. It is obtained by determining the phase difference of the two output signals, using phase detection circuitry. One method for determining the phase difference is to time the pulses using a high frequency clock signal.

3.2 Polulu mini IMU 9 v5

- PoluluMinIMU 9v5 was used for obtaining the robot yaw. It is an Inertial Measurement Unit Sensor, which consists of and accelerometer, gyroscope and magnetometer. These sensors give us raw values, which are then passed into AHRS algorithm, which converts it into Roll, Pitch and Yaw. These values are used to represent a robot's orientation in 3-D space.



- We have used an IMU for the feedback of yaw on MR2, because it consists magnetometer,so error in yaw calculated from readings of accelerometer and gyroscope because of jerks and tilts due to motion of MR2 is compensated.

3.3 LSA 08

- LSA 08 is Advanced Auto calibrating Line sensor by cytron. It offers many function such as junction detection, dark line or light line detection, etc. You can also set threshold for line to considered as detected and width of a junction for getting pulse for junction detection. You can get data in 3 different forms :

  1. Analog Mode
  2. Digital Mode
  3. UART Mode

- User Manual
- **Note :**Some user manual on website has given wrong pin diagram of port, be care full.

3.4 Adjustable Infrared Proximity Sensor

- IR sensors work on the principle of emitting IR light and receiving the reflected light from the object it is pointing to.Whenever it detects the object the output of the IR goes LOW else it is HIGH.

- **Connections :**

- Green : Ground
- Red : 5 V DC
- Yellow : Signal Pin

- We used proximity sensor at 3 different places :
  1. To detect receiving of Gerege and start motion of Mr2 from Gobi Urtuu.
  2. To check shagai on throwing platform to avoid actuation of throwing mechanism without shagai which could be fatal for throwing link.
  3. To start motion of MR2 from Mountain Urtuu by detecting Hand of operator after scoring total 50 points by throwing.

3.5 Ultrasonic Sensor (HC-SR04)

- Ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception.

- **Connections :**

- VCC : 5 V Supply
- GND : Ground
- Trigger Pulse : Input
- Echo Pulse : Output

- **Electric Parameter :**
- Working Voltage : 5 V DC
- Working Current : 15mA
- Working Frequency : 40Hz
- Max Range : 4m
- Min Range : 2cm
- MeasuringAngle : 15 degree
- Trigger Input Signal : 10μs TTL pulse
- Echo Output Signal : Input TTL level signal and the range in proportion

- **<u>Attention :</u>**

1. The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected to the module first, otherwise, it will affect the normal work of the module.

2. When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise, it will affect the results of measuring.
3. As speed of sound depends on density of air medium, you should feed correct temperature of atmosphere for getting good result. HC-SR05 comes with internal temperature sensor.

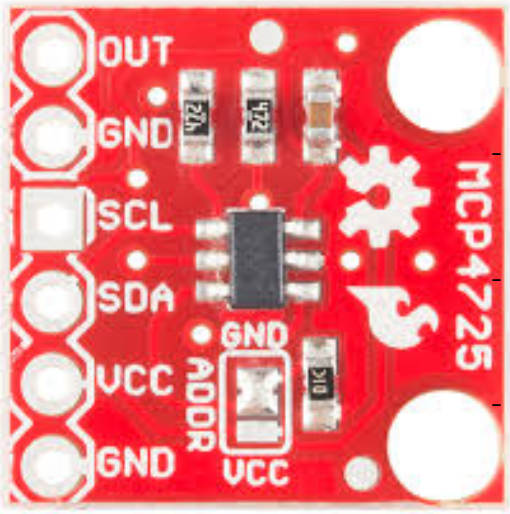
3.6 PPR

- The proportional pressure regulator is a device used for controlled pressure output. The device takes voltage between 0 and 5V and gives pressure between 0 bar and max pressure as output. It linearly maps the voltage input and gives the output accordingly.
- **Connections :**

- Analog : No connection
- Digital : Analog voltage according to required voltage
- GND : Ground
- VCC : 24 V DC

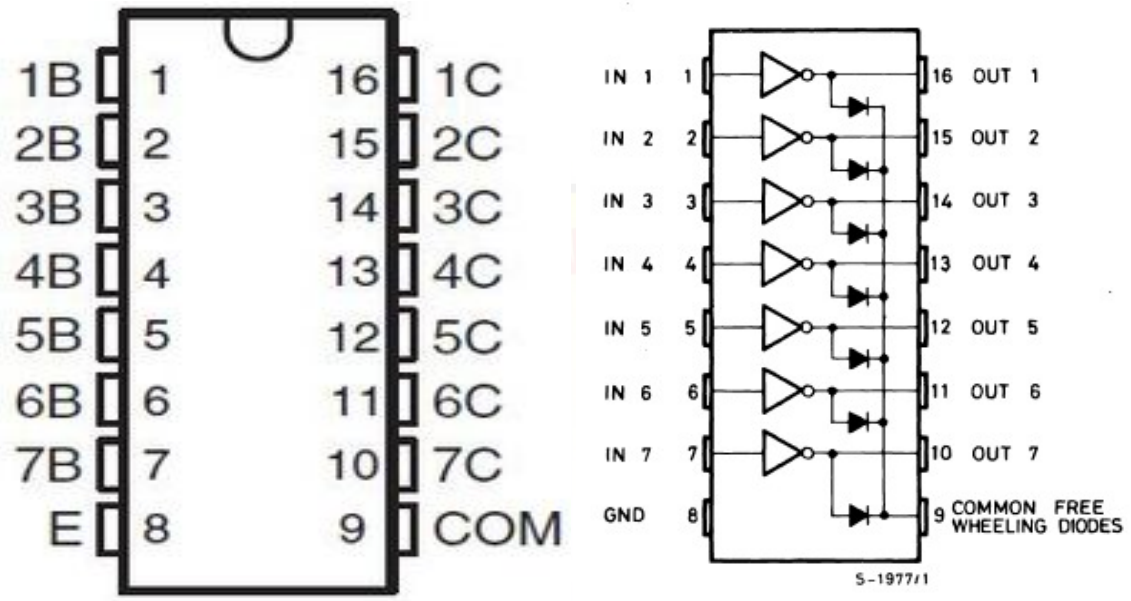
3.7 DAC (MCP4725) IC

- MCP4725 is 12 bit Digital to Analog Converter Integrated Chip which gave constant DC voltage without any fluctuation which is required for PPR. The device's communication protocol is an interintegrated circuit (I2C). In I2C communication DAC IC behaves as slave and MCU behaves as master.
- **Connection :**



- OUT : Output of DAC (In Digital Pin of PPR)
- GND : Ground
- SCL : SCL of MCU
- SDA :SDA of MCU
- VCC : 5 V DC

3.8 ULN2003A IC



- The IC ULN2003A comprises of 7-NPN Darlington pairs and is typically used to switch inductive loads (dissipates voltage spikes if any using suppression diode) and to drive stepper motors.

- We used ULN2003A for pneumatic actuation and LED operation to protect MCU from high current drawn by LEDs.

4 MESSENGER ROBOT 1

4.1 MECHANICAL

4.1.1 Encoder Mouting for odometry

1. **Using journal bearing**

   - We have tried different methods for fitting of X and Y encoders which includes spring setup and slider setup. In the spring setup we have used a compression spring which was bent at both ends in such a way that a bolt can be inserted. Two plates were kept hinged and a spring was attached in between which allowed the encoder to remain in direct contact with the ground. But the problem with this orientation was that, out of the two wheels only one touches the ground due to which sometimes ticks were missed and was not that efficient.

2. **Using spring**

   - .....

3. **Using CNC bearing**

   - In this mechanism we mounted a clamp on the chassis and vertical piece of plywood with it. Then a shaft support was fixed on the other side of plywood and a shaft was mounted vertically downwards. A CNC bearing was used in the shaft and a plate was fixed on it and then encoder were fixed on the lower side of it.

4.1.2 Shagai gripping and lifting mechanism

1. Wedge type gripping

   - In order to save time and to complete tasks faster it was proposed, to develop a mechanism such that it can pick up the shagai in desired orientation while throwing it simultaneously.
     The idea was to load the shagai by sliding it over a wedge and then throwing it once loaded.
     For this there were two approaches:-

(a) 1st approach

Earliest approach was to pick up the shagai using the walls of the arena by a wedge type platform i.e to make the shagai slide on the platform of the mechanism using the arena boundary wall as a restriction for the shagai movement.

For loading by wedge there were two ideas by using two rollers such that shagai gets slid upon the platform.

- This approach failed as it resulted in the requirement of a lot of force to actuate the piston considering the pressure limitation of 6 bar to be used in bottles and also for the piston.

- It was observed that the piston in its orientation was unable to lift the platform due to the heavy shagai. This was mainly due to the partial utilization of piston force in the required direction while other component was acting parallel to the platform resulting in loss of power.

- Even after applying the required force the projectile obtained was not sufficient to make the shagai land in the landing zone from the throwing zone.

(b) 2nd approach By using two pistons, first piston responsible for lifting

the platform to the desired angle with respect to the vertical and the second piston responsible for throwing the shagai after the platform had reached the desired position.

- This approach was better than the first approach in providing

better projectile results than the first approach.

- The force required to lift the platform to desired orientation was

quite high because of the reverse torque generated as a result of the force of gravity of shagai. The torque of component of piston force was lesser than the gravitational torque of the shagai.

- Also apart from the force factor to make the platform touch the

ground, a slope lesser than 15 to 20 degrees was required so that the shagai can slide on it. this resulted in an increase in the length of the link on which the platform was attached (throwing link). The length calculated was more than the robot dimension limitations. Also as the length increased the torque increased as the distance between the piston and the shagai increased.

Hence this idea was also dropped.

2. Double piston gripping

   - In this mechanism, both the flat surface top and bottom of the shagai were gripped using pneumatic pistons. Two pistons of 5mm stroke length were used on one flat moving side while the other side is fixed, so when the piston is actuated the shagai gets gripped. Now by using motor the gripper is rotated about the x-axis to place the shagai on the throwing platform.

     Reasons for the rejection of the mechanism are:-

     – More precision is needed i.e the maximum distance between the two arms of gripping was quite less. Thus, for the process of gripping the gripper needed to be precisely aligned about the shagai, which decreased the speed of the process and thus reduced the over all speed of the robot.

     – The desired face of the shagai on the throwing platform .i.e flat surface at the top was not being achieved as desired. (flat surface on the top increased the chances of scoring 40 or 50 points depending on which side was on the top)

3. Parallel gripping or linear gripping mechanism

   - This mechanism used the flat and curved surface for the purpose of gripping.The flat surface held with the stationary link of the mechanism and the curved surface was held with the moving link of the mechanism. The mechanism was as such a C shaped mechanism with the moving link being actuated using two pistons. The pistons were placed parallel to each other such that the actuation of the pistons resulted in the generation of a normal force on the shagai. This normal force provided the frictional

force for lifting the shagai and keeping it intact in the gripping mechanism when the mechanism moved or rotated to load the shagai on the loading mechanism.

Problems

- When the side with curved surface was gripped with a flat link it resulted in poor gripping and slipping of shagai from the mechanism.
- Optimum pressure control was required in order to not damage the soft material of shagai which required extra circuitry for pneumatic control.
- Also while loading of the shagai , probability of shagai slipping was seen to be quite higher.
- Also by changing the orientation of the gripping of the shagai resulted in the increase in the size of the mechanism which made it difficult to handle and to mount on the robot as there were dimension constraints for the same.
- To improve the friction , use of sandpaper and a special mould for providing a snug fit to the shagai was used , but it requires a high level of precision to place the shagai accurately in the mechanism so that the mould fits on the shagai, this approach worked but it required higher time for implementation considering the game time.

4. Two jaw mechanism (2 jaw encompassing angular gripper)

   The 2 jaws are actuated using a pneumatic linear actuator. The stroke length required to actuate the gripper is calculated on the basis of the angular distance needed to be covered by the jaws to grip the Shagai. Bore diameter is calculated on the basis of gripping force required to hold the Shagai. The angular motion of gripper mechanism for loading Shagai is controlled by high torque motor.

   Also one piston was used to change the height of gripping platform : this was done with the thought of piling two shagai and then gripping them them one

after another from the same position. But the idea was dropped because of complication in the actuation system as it increased the number of actuators.

The two jaw of gripper was used to grip the curved surface of the shagai. Firstly the encoder was Mounted along with the motor but the encoder was replaced with a limit switch. The use of encoder was not very efficient because the inertia was high enough due to which the shagai cannot be dropped at the desired location. So the limit switch was used for gripping and loading shagai on the throwing platform.

For moving the shagai from ground to the throwing platform mechtex motor is used. Rotation was constrained by limit switches on both sides.

- The mechanism was difficult to build compared to the parallel gripping mechanism but it provided various advantages such as better gripping,

smaller in size as well as the freedom to grip the shagai in desired orientation.

- In this case also in order to increase the grip, we used sandpaper and two curved moulds which gripped the two curved surfaces providing a better fit to the shagai.

- The tolerance in case of fitting the shagai was higher compared to the parallel gripping mechanism.

- Changing the angles of the gripper links helped in controlling the force acting, to grip the shagai so that its soft material is not damaged.

- The smaller mechanism also resulted in weight reduction and required lesser torque motor.

Actual gripper CAD model

In manufacturing the curved shape of gripping the shagai we used acrylic. For bending acrylic, we tried using a hot air gun but it took a little more time. So we used the heat of a gas burner. Holding acrylic on the gas burner makes it flexible thus allowing us to bend it to the required shape. On cooling, it hardens and the desired shape was obtained.

4.1.3 Shagai throwing mechanism

1. **With the help of pneumatic piston**

First of all we started making a throwing mechanism in which we mounted piston vertically. Our main aim was to throw the shagai at least 5 to 6 metres away. In this mechanism, we converted the translational motion of the piston into an arc motion of the platform, on which the shagai was placed.

Vertical mounting

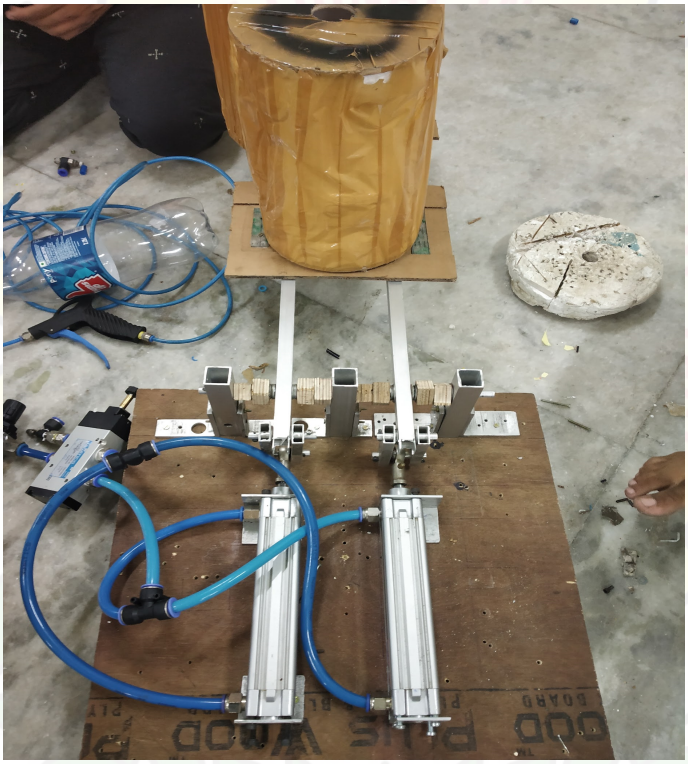
Displacement of the shagai was about 2 to 2.5 m. The major drawback of this mechanism was that when actuated at a pressure greater than 4 bar it resulted in bending of links.

Then we decided to mount the piston horizontally. But faced the same problem that the displacement of the shagai was 2.6 to 3.2m. When this mechanism was mounted at some height, displacement of the shagai was coming out to be higher than that of ground level.

Horizontal mounting

As this mechanism was not giving the required displacement of shagai so we dropped this idea and thought of mounting two pistons horizontally. The displacement of shagai obtained with this mechanism was about 5 to 6.2m.

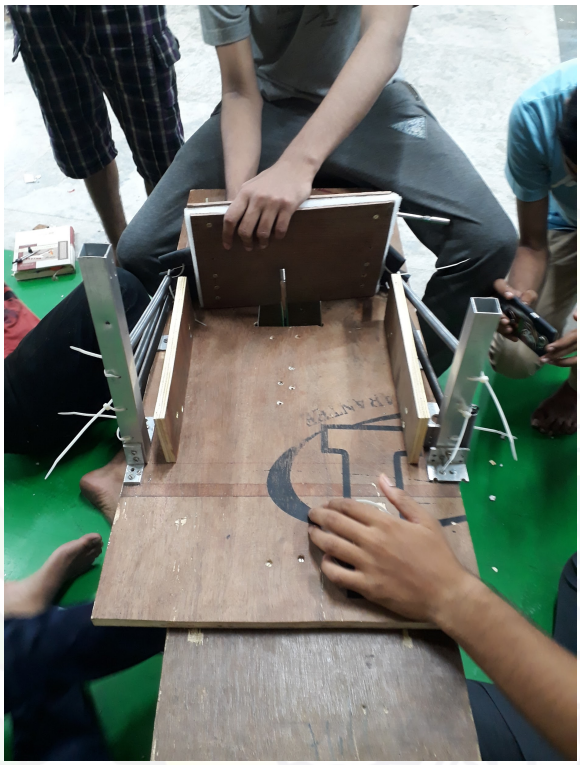
The major drawback of this mechanism was that the reaction force or the impulsive force, provided by the pistons was very large due to which the robot became unstable.



Double piston mounting

2. **Using Spring Force with combination of piston**

As the main problem we faced till now was of range of the shagai, we decided to use a piston and springs together. The assembly is as shown below.




As shown, Two pistons and 3 springs are the driving parts of the mechanism.

It uses two supporting links on side(Made of box section), 3 springs(can be altered according to the requirements), two guides and a hitting board made of foam which is covered by ply from both sides.
The springs and the hitting board is connected by means of key chain pins and tyre tube material.
The entire final assembly is shown in the images above.

The problem with this was the range. Also, due to very high reaction, the actuation of assembly resulted in high jerk. Another problem was of resetting the system by extending the hitting board to its original position. Bringing back the board to its original position was a challenge as the force required for the same was higher than the pneumatic or electric actuation capacity.

3. **STRIKER MECHANISM (Using push from piston)** In this mechanism we focus on range as well as orientation of shagai. the push from piston

is transferred to shagai for throwing it .

First of all we tested it by mounting it on plywood.For changing the angle of piston we changed the angle of plywood on which the piston was mounted.We obtained the range of 3.5 m and orientation of shagai as required.

So we decided to assemble on manual bot for further test.

The plywood shown above is mounted on manual bot and second ply is hinged to the ply from one end so that we could change angle from the other side of ply.

The actuation of piston at a pressure of 5 bar resulted in a high reaction force which resulted in error in the encoder readings of the autonomous robot.

4. **THROWING USING WEDGE AND PISTON:**

Now as we achieved the required range, the next step was to create a loading system of shagai on the mechanism.
By analysis, We found out the optimum angle at which the shagai must be thrown so that the range is maximum.

We decided on a wedge for loading and then the shagai lands in front of piston , so that on opening of piston the required push is obtained.

The arrangement of mechanism is shown below-

The shagai will be picked up by the surface touching the ground and then the wedge will turn up to the angle required. After this ,the piston will now be actuated.

Gate type wedge for shagai picking

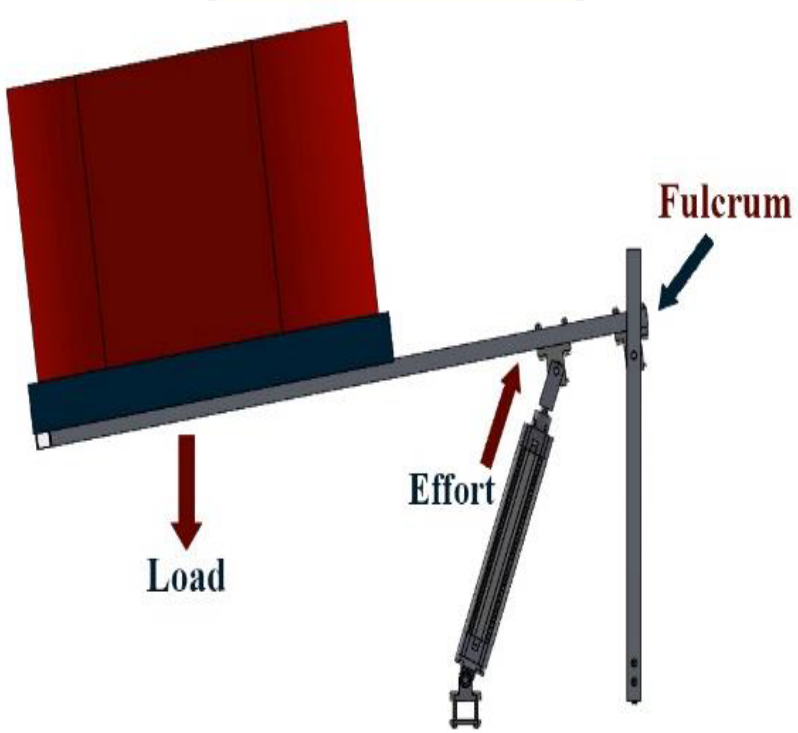
An additional setup was thought for stopping the shagai from sliding back on the ground once loaded on the mechanism platform.

5. **FINAL THROWING MECHANISM (USED)** It is based on class three lever system. Here the load is Shagai, Fulcrum is the pivot, Effort is the force applied by pneumatic linear actuator. A Quick Exhaust Valve is used to increase the velocity in forward stroke of pneumatic linear actuator which causes increment of momentum of Shagai.

   Pneumatic linear actuator for throwing.

Bore Diameter 32mm
Stroke length 150mm

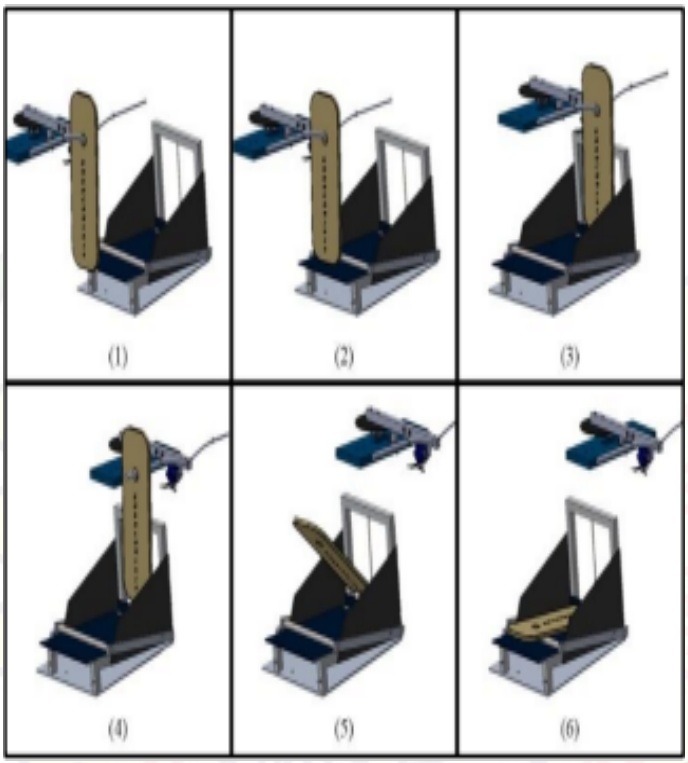
This mechanism gave the advantage of varying projectiles which depended upon the pressure with which the piston was actuated. Full effort by the piston was utilized for throwing the shagai as compared to other mechanisms.



4.1.4 Gerege Gripping Mechanism for MR1

- According to our task, we only have to hold gerege in MR1 and change the direction of gerege according to movement of MR1.
- we mount one servo motor on a plywood platform on which one bent cane was mounted So, we can hold gerege using hole provided in it.
- platform rotate by servo motor according to requirement.

- we use one more servo motor to prevent gerege from dropping.



4.2 ELECTRONICS

4.2.1 Omni drive

What is omnidirectional wheels ?

- Omnidirectional wheel is based on a general principle : when the wheel provide proper traction in the direction normal to the axis of wheel , the wheel can slide in the direction of the axis of wheel without friction.

Inverse Kinematics of Omni Drive

- In order to derive inverse kinematics, we need to have the basic information of the robot's dimensions. The skeleton of the 4-wheeled omni drive robot is shown in the *Figure 8*.

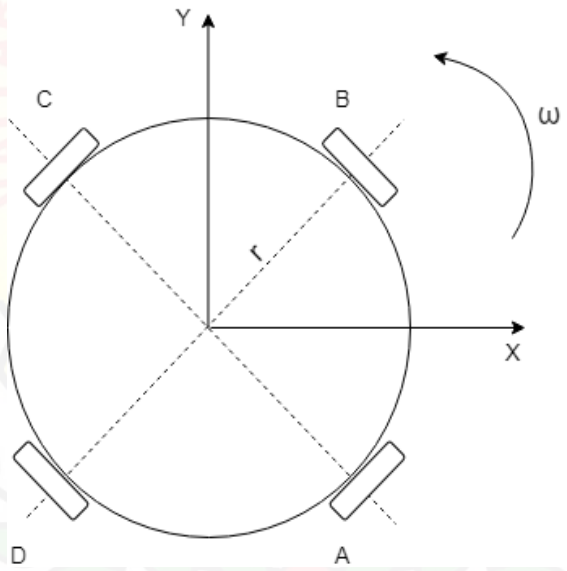


Figure 1: Omni Drive Skeleton

- Consider the $wheel_i$ of the robot, where $i \in \{A, B, C, D\}$. Let $\vec{v_i}$ be the its velocity and $\vec{v_{r_i}}$ be the velocity of its roller, which is perpendicular to the direction of $\vec{v_i}$. We need to obtain wheel's velocity when the robot's velocity is known, i.e. $\vec{v}$ and $\vec{\omega}$ are known.

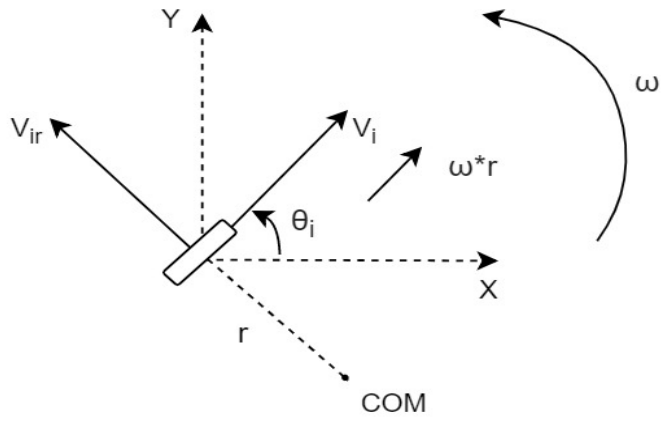


Figure 2: Kinematic diagram of $wheel_i$

From the basic idea of Kinematics, we know

$$\vec{v} = \vec{\omega} \times \vec{r}. \tag{1}$$

So, we can resolve the $\vec{v}$ and $\vec{\omega}$ vectors in X and Y directions as shown in *Figure 9*. Using equation (1) and resolving the velocity vectors into components.
X-component:

$$v_x + \omega r \cos\theta_i = v_i \sin\theta_i - v_{r_i} \sin\theta_i, \tag{2}$$

Y-component:

$$v_y + \omega r \sin\theta_i = v_i \cos\theta_i + v_{r_i} \cos\theta_i. \tag{3}$$

- Using (2) and (3), we have

$$v_{r_i} = \frac{v_y + \omega r \sin\theta_i - v_i \sin\theta_i}{\cos\theta_i}, \tag{4}$$

$$\therefore v_i = v_x \cos\theta_i + v_y \sin\theta_i + \omega r, \tag{5}$$

$$\therefore v_i = v \cos\theta \cos\theta_i + v \sin\theta \sin\theta_i + \omega r, \tag{6}$$

$$\therefore v_i = v \cos(\theta - \theta_i) + \omega r. \tag{7}$$

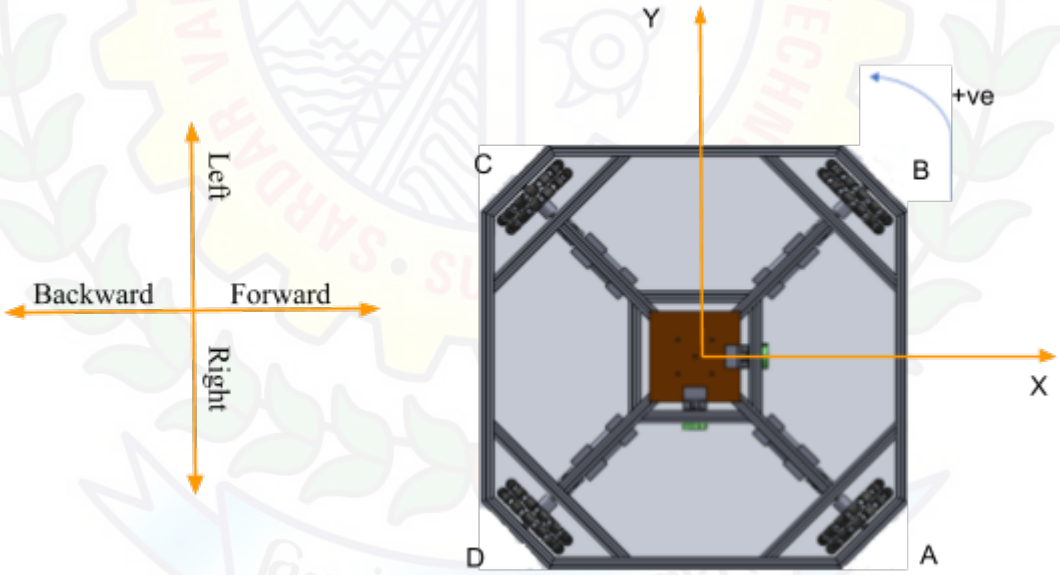
- Using (7), one can write it in matrix form as

$$\begin{bmatrix} v_A \\ v_B \\ v_C \\ v_D \end{bmatrix} = \begin{bmatrix} \cos\theta_A & \sin\theta_A & \mathrm{r} \\ \cos\theta_B & \sin\theta_B & \mathrm{r} \\ \cos\theta_C & \sin\theta_C & \mathrm{r} \\ \cos\theta_D & \sin\theta_D & \mathrm{r} \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \tag{8}$$

- Using the matrix equation(8), we obtain the velocity of each wheel that would move the robot at velocity $\vec{v}$, $\vec{\omega}$.

Convention Used :

- Convention which was used by us in 4 wheel omni drive is shown using following images :

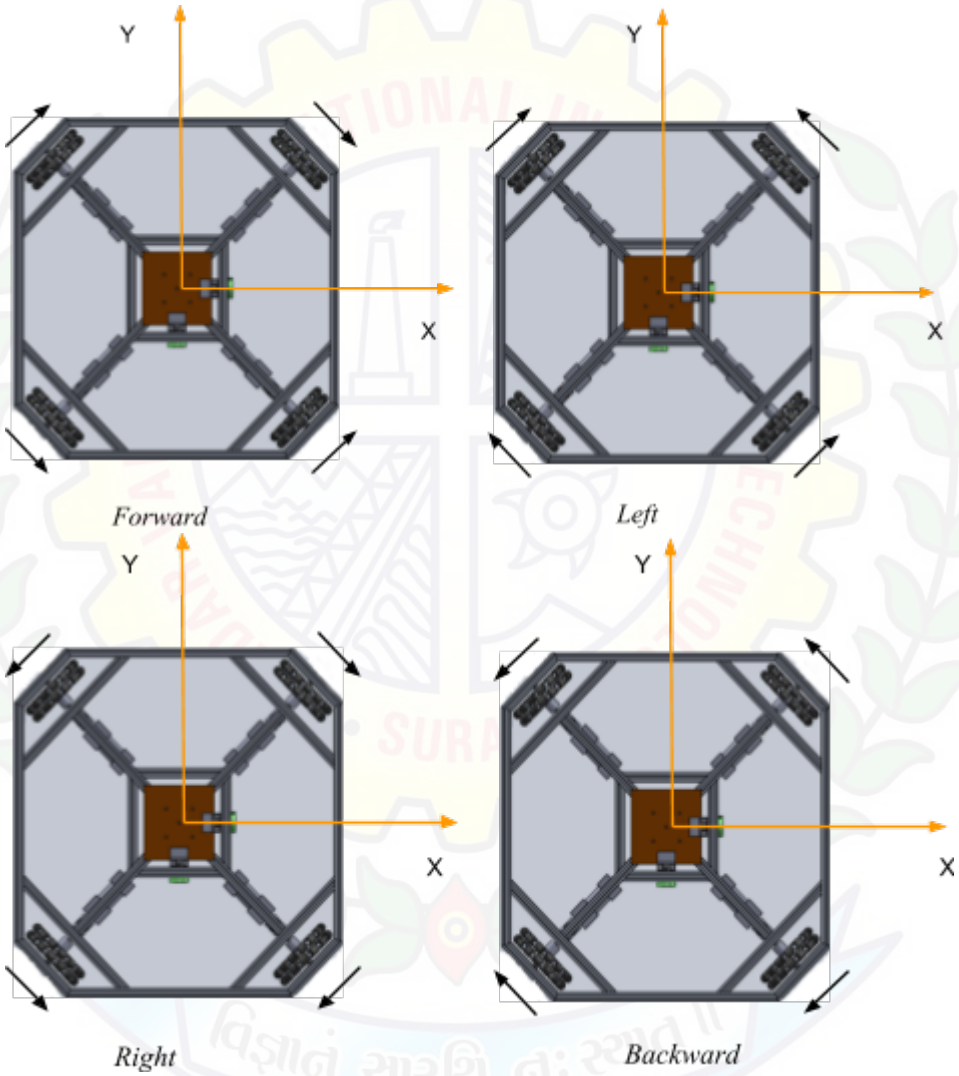


Anticlockwise +ve

RPM of each wheel to rotate robot in Anticlockwise direction was considered as +ve RPM.

Wheel	Angle
A	315°
B	45°
C	135°
D	225°

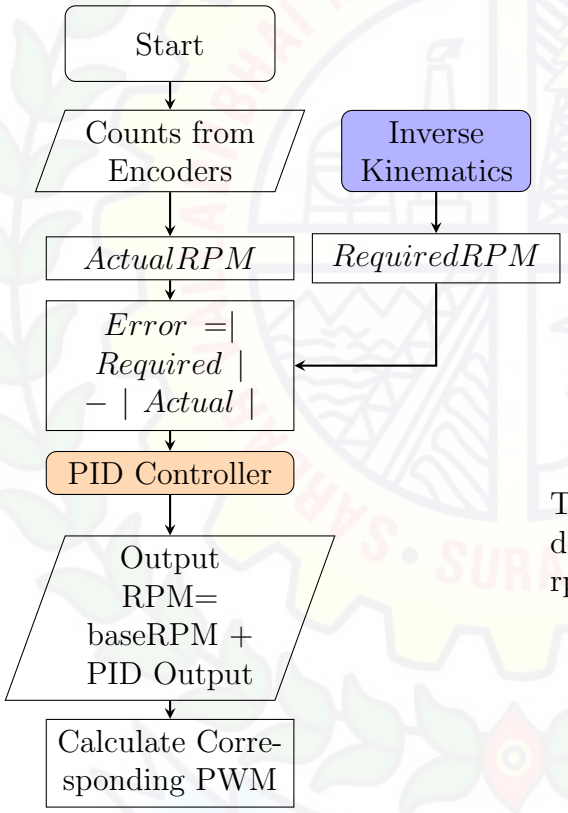
Side	Direction
AB	Forward
BC	Left
CD	Backward
DA	Right



4.2.2 Lower Control

- The desired velocity and direction of each motor is calculated using Inverse Kinematics and data ($\vec{v}$, $\vec{\omega}$) received from Upper Control, After that the desired velocity of each motor is attained and maintained by this subsystem. It consists of a control system PID(Proportional Integral Derivative). The control system is applied to the velocity of each motor.

- **Flow Diagram :**General flow of code is explained below

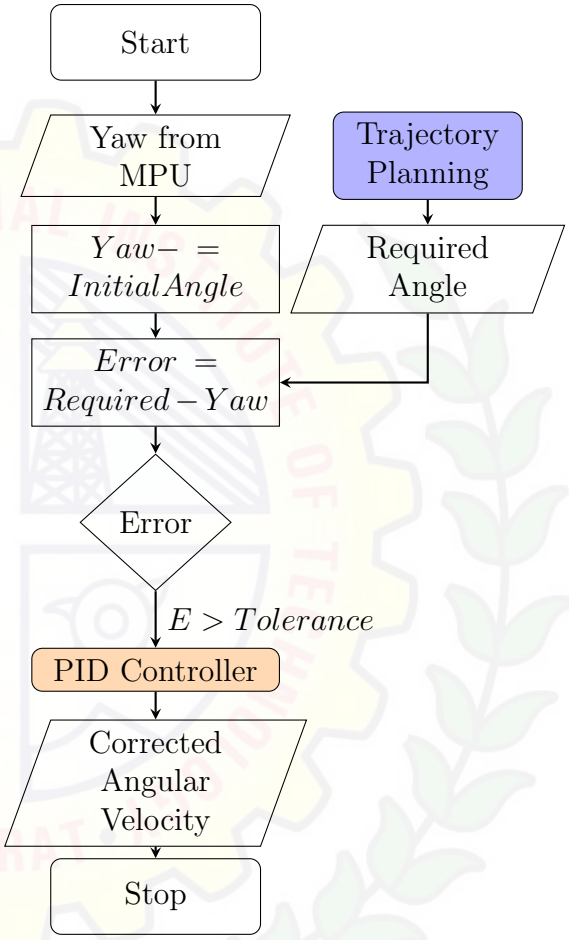


1. Required RPM of each wheel is calculated using Inverse Kinematics.
2. The direction of the motors is decided depending on the sign of the required rpm.
3. The rpm feedback from the encoders is fed into the control system PID and the output of the PID is scaled into PWM and fed into Motor Drivers.

The direction of the motors is decided depending on the sign of the required rpm.

4.2.3 Omega Control

- We have used Digital Motion Processor-(DMP) of MPU6050 sensor for the feedback of orientation of the robot. We read the data (Yaw) from sensor at a fixed frequency. We are using the initial position of the robot as a reference frame. Once we get the orientation of the robot with respect to it's original orientation we compare it with the desired orientation and calculate the error(E).If error is less than the tolerance, we don't need to do any correction in angular velocity. Tolerance is required because of some internal inaccuracy of the sensors which results in oscillation of the angle, which is too small and difficult to correct with filter. If error is larger than the tolerance, we apply PID on that and corrected angular velocity is fed for calculations of RPM of each wheel in Inverse Kinematics.
- This system was a part of Upper Control System.



4.2.4 Localization

- The robot is localized with the help of two rotary encoders and an MPU-6050 sensor. The distance moved by a wheel coupled to rotary encoder can be obtained by

$$dist = 2\pi r \frac{count}{ppr}, \tag{9}$$

where
r = radius of wheel coupled to encoder,
count = counts received from encoder,
ppr = pulse per revolution of encoder.

- The MPU-6050 device combine a 3-axis gyroscope and a 3-axis accelerometer on the same silicon die, together with an onboard Digital Motion Processor™ (DMP™), which processes complex 6-axis MotionFusion algorithms. We obtain roll, pitch and yaw from the sensor.

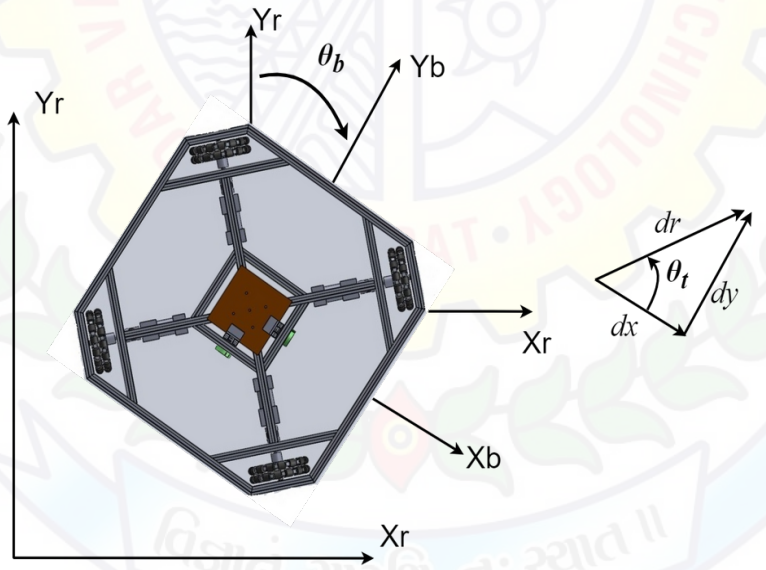


Figure 3: Axis of robot and Axis of reference

The data obtained from the three sensors are fed into the following localization algorithm.

In the *Figure 3*, dx and dy are the small distance travelled by robot in the directions of X_b and Y_b respectively and yaw of the robot (θ_b).

$$dr = \sqrt{(dx)^2 + (dy)^2}. \tag{10}$$

The total distance travelled by robot during this small time will be dr.This distance dr is travelled at an angle θ_t with respect to X_b

$$\theta_t = \tan^{-1}(\frac{dy}{dx}).$$

So, the angle θ (angle of $\vec{dr}$ with respect to X_r) can be given by

$$\theta = \theta_t - \theta_b. \tag{11}$$

Now we can divide $\vec{dr}$ in its components in the directions of X_r and Y_r as

$$BotX = prevBotX + dr * cos\theta, \tag{12}$$

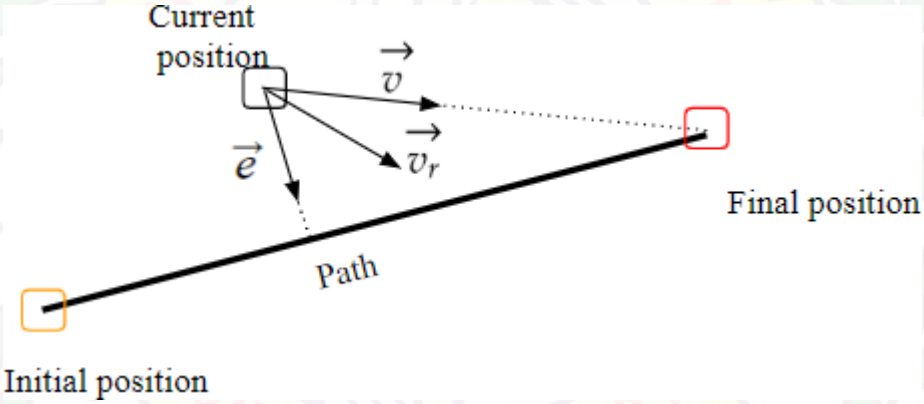
$$BotY = prevBotY + dr * sin\theta. \tag{13}$$

Now, using equations (12) and (13) we can calculate current position of the robot ($BotX$ and $BotY$) with respect to X_r and Y_r with the help of position of the robot in the last time interval($prevBotX$ and $prevBotY$), values of whose are zero initially and later calculated by substituting $prevBotX = BotX$ and $prevBotY = BotY$ after executing equations (12) and (13).

- This system was a part of Upper Control System.

4.2.5 Trajectory Planning

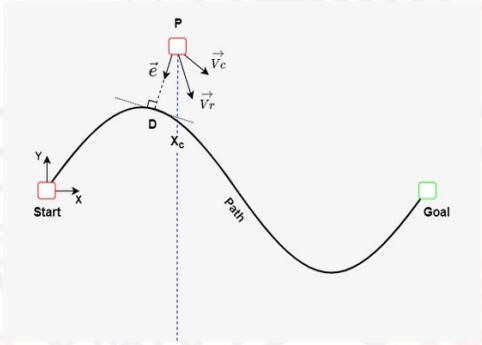
1. **Tracing a line :**
   With the knowledge of pose, we can know the error with respect to our desired pose and accordingly plan the trajectory. An example is shown below. The robot responses to any deviation in the shown manner. $\vec{v}$ represents the velocity vector pointing towards the goal with a constant magnitude, $\vec{e}$ represents the error vector pointing to the closest point on the path. Its magnitude is a function of error in the path. The robot moves with resultant vector $\vec{v_r} = \vec{v} + \vec{e}$.



2. **Tracing Sine :**
In the figure shown below, the robot is required to trace the path shown. The direction of velocity at any point on path can be obtained with the help of slope at that point.

$$\theta = \tan^{-1}(\frac{dy}{dx}), \tag{6}$$

where θ is the angle the robot should make with X-axis. Now, if somehow the robot deviate from its path and reach some other point P. The obtained velocity vector from above equation at that point will lead the robot to a wrong path. So, we have to reach to the nearest point on the curve and then trace the path.



In the above, P is the current position and D is nearest point on the curve.

Now, the equation of the curve is

$$y = A\sin(kx), \tag{7}$$

hence

$$\frac{dy}{dx} = Ak\cos(kx). \tag{8}$$

Let coordinates of point P be (x, y) and that of D be (x_d, y_d). We know the coordinates of P, we need to know the coordinates of D so as to get the $\vec{e}$, which can be used further to obtain the resultant velocity vector $\vec{v_r}$.

Now, the slope of tangent at D is

$$\frac{dy}{dx}_D = Ak\cos(kx_d). \tag{9}$$

For D to be nearest to P, the line PD have to be normal to curve at D. For two perpendicular lines with slopes m_1 and m_2, we have

$$m_1 m_2 = -1. \tag{10}$$

Using the equations (9) and (10), we have slope of line PD,

$$M_{PD} = -\frac{1}{Ak\cos(kx_d)}. \tag{11}$$

Also, the slope of line PD using two-point form of line is

$$M_{PD} = \frac{y - y_d}{x - x_d}. \tag{12}$$

Using (11) and (12), we have

$$(y - y_d)Ak\cos(kx_d) = (x_d - x), \tag{13}$$

or

$$(y - Asin(kx_d))Ak\cos(kx_d) = (x_d - x). \tag{14}$$

Clearly, equation (14) doesn't have any analytical solution. It is a transcendental equation in x_d. The equation can be solved with numerical methods such as Newton Raphson method.

Once x_d is known, we have value of y_d using the equation of curve. So, now robot will move with a vector

$$\vec{v_r} = \vec{e} + \vec{v_c}, \tag{15}$$

where $\vec{e}$ is proportional to the distance d_{PD} and $\vec{v_c}$ is the velocity vector using equation (9) at X_c, the intersection of curve and projection from P to X-Axis. Hence, each point of the curve will be traced with the above algorithm.

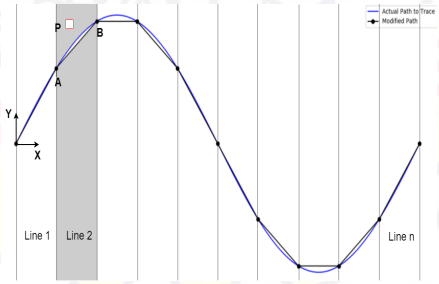
The point-to-line tracing

As shown before, if we try to find nearest point on a path from the current position of the robot, we may get a transcendental equation. To solve the equation, we have to use iterative methods which requires high computation. So, we have developed an efficient approach, which is described below.

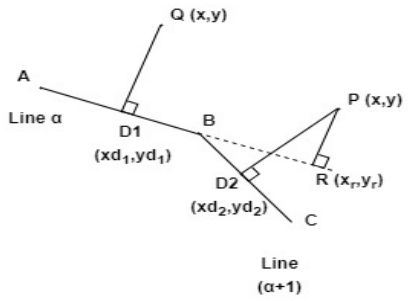
In any case of deviation from the path, the sine wave path will be considered as a sine wave made up of n lines. The robot is then bound to find the nearest line for the correction in path.

To trace the sine wave curve using this approach, we first need to choose the number of points(n) in a wavelength(λ). As n$\rightarrow \infty$, this approach tends to the point-to-point tracing approach, which is described above. Now, we have to trace n lines with horizontal component

$$\Delta x = \lambda/n. \tag{16}$$



Now, we have to trace these lines in order. We can decide which line is currently under progress with the help of current position. The above figure illustrates the situation, the robot's position is P and Q in two different scenarios.



Let α be an integer such that

$$\alpha = \left\lfloor \frac{BotX}{\Delta x} \right\rfloor + 1. \tag{17}$$

Clearly $\alpha \in \{1, 2, 3, ..., n+1\}$. Line α is a line between points from (x_α, y_α) to $(x_{\alpha-1}, y_{\alpha-1})$. We also have,

$$x_\alpha = \alpha * \Delta x, \tag{18}$$

$$y_\alpha = A \sin(k\alpha * \Delta x). \tag{19}$$

The slope of line α is

$$m_\alpha = \frac{y_\alpha - y_{\alpha-1}}{x_\alpha - x_{\alpha-1}}. \tag{20}$$

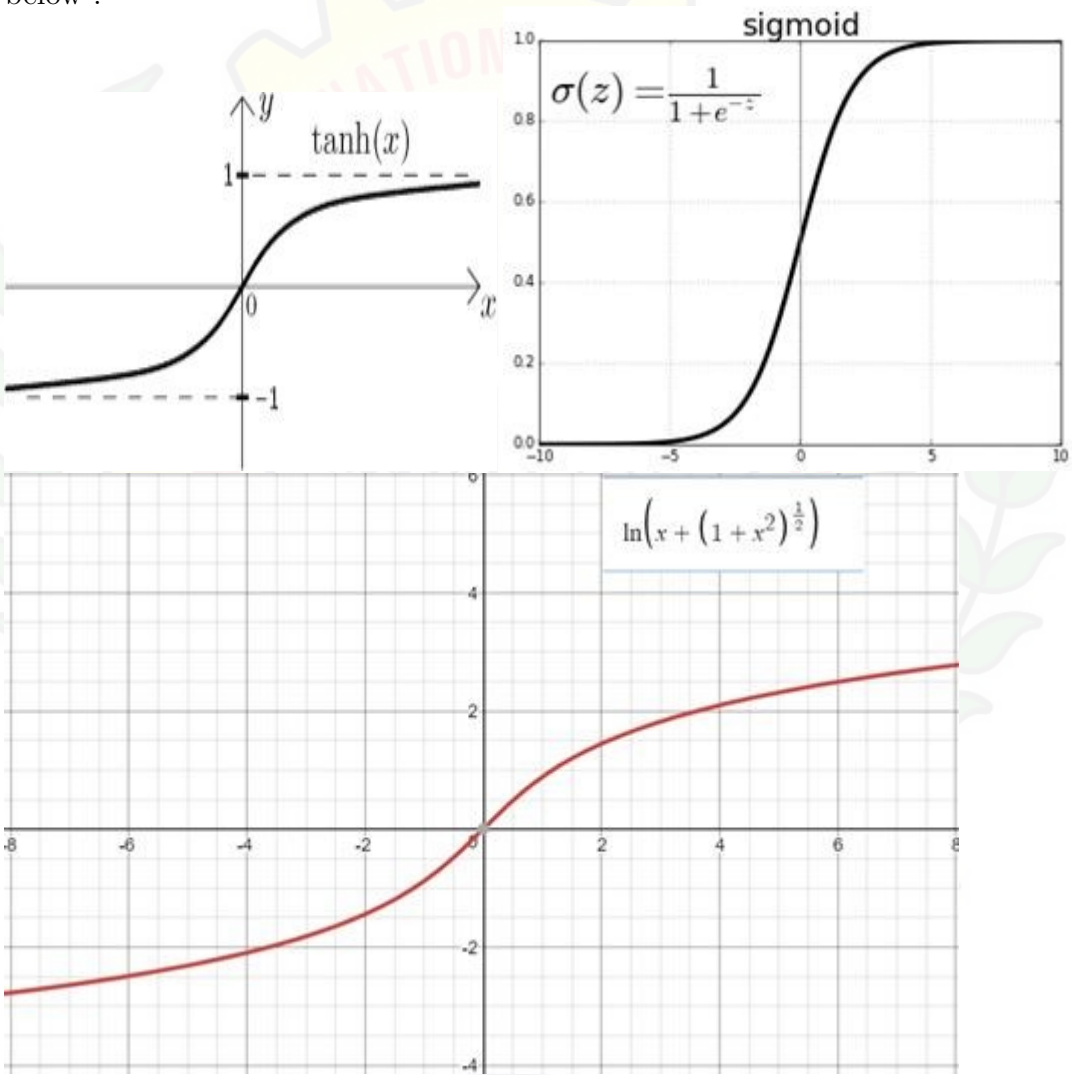
Using (18),(19),(20) and line equations, we can obtain the coordinates of point D.

$$x_d = \frac{m_\alpha(BotY - y_\alpha) + m_\alpha^2 x_\alpha + BotX}{m_\alpha^2 + 1}, \tag{21}$$

$$y_d = m_\alpha(x_d - x_\alpha) + y_\alpha. \tag{22}$$

Now, with the help of coordinates of D (equations (21) and (22)), we can obtain the $\vec{v_c}$ (equation (15)). Since, this method is analytical and independent of any initial guess, the method is efficient and doesn't need high computational device.

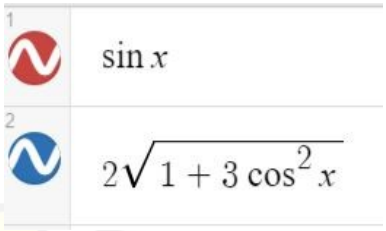
3. **Velocity as a function of distance :**
   To provide sufficient traction and for smooth transaction we used acceleration - deceleration with the help of counter first but it was dependent on code loop frequency.
   To overcome this we varied velocity of the robot depending on the current position of the robot.We tried different function for this purpose which are shown below :

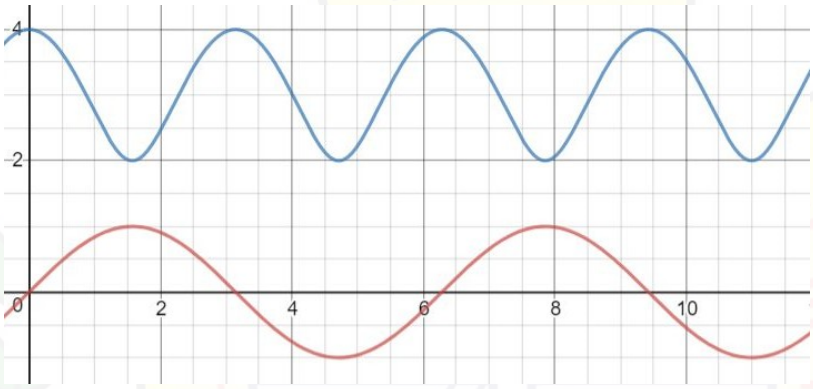


   For traversing sine wave it was difficult to set acceleration - deceleration on it's small parts, so we used following function to feed velocity to robot.

$$y = \sin x$$

$$v(x) = \frac{v_{max}}{2}\sqrt{1 + 3\cos^2 x}$$

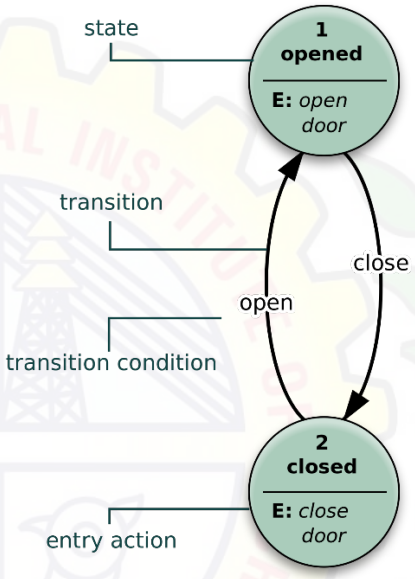




4.2.6 Finite State Machine (FSM)

What is FSM?

- A finite-state machine (FSM) or finite-state automaton, finite automaton, or simply a state machine, is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some external inputs; the change from one state to another is called a transition. An FSM is defined by a list of its states, its initial state, and the conditions for each transition.
- This system was a part of Upper Control System.



Implementation :

- We implemented FSM in our code using switch case. There was a case corresponding to each state. This approach was helpful for developing independent code for different portion and it's debugging.
- **STATES IN ARENA :**

- khangaiUrtuu
- forest
- bridge
- khangaiArea
- towardsGobi
- passGerege
- loadShaghai
- waitForAutonomous
- throwShaghai
- TZ1 (Throwing point 1)
- TZ2 (Throwing point 2)
- TZ3 (Throwing point 3)

4.2.7 Line Following

- We used line following in bridge area, as bridge was narrow so, even a small error in localization of Odometry can result in collision. With addition of this we were resetting our coordinate system by using junction on the bridge.
- Before this we also tried to use line following for Khangai Area.

- As in Khangai area first of all we have to traverse through Forest& River. As arena is not symmetric, we should make the general code for both, blue and red zones. For line following using holonomic drive, our main 2 aims are to maintain robot's orientation and traverse desired path using line.

- **How to traverse line :**To traverse line we keep centre of Active Line Sensor(LSA in the direction of motion) on the centre of the line using PID control. To traverse desired path we change the Active Line Sensor & Active Omega Sensor (LSA opposite to Active Line Sensor) according to algorithm developed based on requirement.

- **How to maintain orientation of drive :**

  1. **Using Omega Senor :** We can use the feedback of Active Omega Sensor to keep bot in desired orientation. If forward and backward both sensors are exactly on the of line then bot is in right orientation and do not need to give omega. If there is difference in readings of Active Line Sensor & Active Omega Sensor then we have to give omega to the bot according to the readings.
  2. **Using IMU :** We can use IMU to maintain orientation of drive. We set our initial IMU angle as our required angle. And use PID control on the error calculated with the help of reading of IMU.

     As described above to maintain orientation using Omega Sensor we need both (Active Line Sensor & Active Omega sensor) on line but in Forest region there are sharp 90° turns in which this was not possible, so we have to use to feedback from IMU in such situations.

- **Algorithm for Forest :**

  1. At starting forward sensor will be active sensor means bot will be aligned such that forward sensor will stay on center of line and feedback of backward sensor will also be considered in omega control to keep bot aligned on line.

2. As soon as forward sensor's reading stops we will start taking readings of both sensors on side i.e. left and right. As soon as line is detected by one of them we will make that sensor active sensor, until that it will continue moving in forward direction.
3. Now bot will be moving according to the reading of left or right sensor. Now when line will not be detected on active sensor, we will start taking readings from forward sensor. Now as soon as line is detected at forward sensor we will make it as active line sensor.
4. Repeating step 2 & 3 continuously we will reach GOBI area and we will not require separate code for red and blue arena.

- This method was taking more time than Odometry because of number of sharp turns in path, so we used Odometry in competition.

4.2.8 PS2

- We used PS2 to give some specific signals to robot like Selecting Arena (Red or Blue), Enter Throwing Zone (Had to wait for MR2 to reach Mountain Urtuu region.), etc..
- We also kept Manual driving as back up if some problem arrives in Odometry. We were able to switch to manual at any time in match and can switch back to Odometry from some fixed reference points.
- As per the rule book we cannot use RF PS2 but we had only RF PS2 available. So we used a microcontroller to receive PS2 data nad send it to robot using Bluetooth.
- For more information about PS2 data refer : A PROJECT REPORT ON COMPETITION ROBOCON'18

5 MESSENGER ROBOT 2

5.1 MECHANICAL

5.1.1 Walking Mechanism

1. THEO JANSEN MECHANISM

   We made CAD model of theo jansen mechanism, from which we came to know how much distance it covers in x direction and the y direction in one revolution.

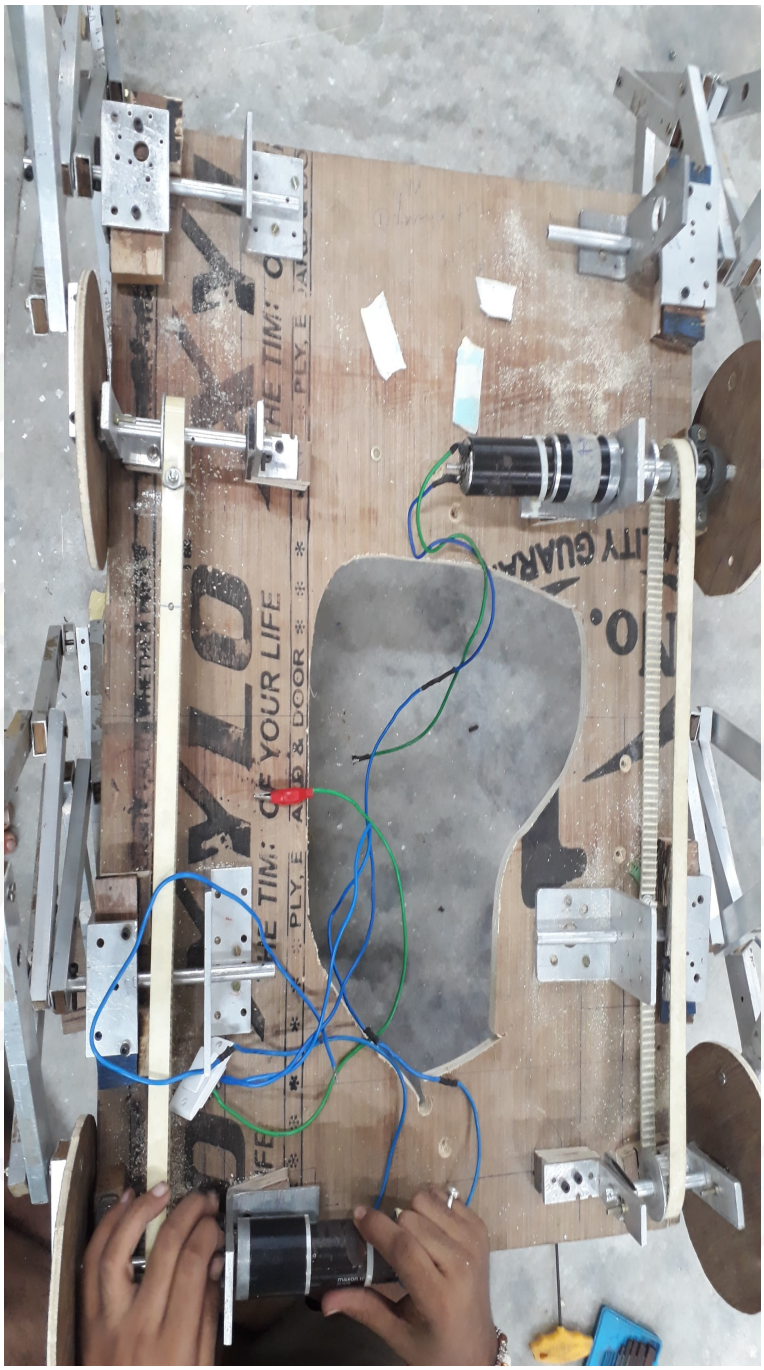
   According to design, first of all we have to manufacture 11 links of different dimensions for only one leg.

   So, we made total 44 links of aluminium box section of 1×0.5 inch . Major problem we faced was due to more no. of links. Another problem was due to high weight of bot.

   Then, we have to make holes in those links in aluminum box section, to avoid bending of links, we use wood padding, which are fixed in hollow section and then we make hole.

   After this, according to design, we assembled all links. As shown in the image below-

Then we assembled all four legs on plywood.At the end of the leg, we have attached one small plywood, which is hinged at the bottom of the leg.Due to this, we get more area of contact of leg with ground.
In this mechanism, we use 2 planetary motors which are connected diagonally in legs.


YOUR LIFE
PLY, E
& DOOR

And then we use time belt and pulley for power transmission.So ,due to this, our right side legs are rotating with one motor in phase difference of 180°. Entire Assembly is as shown-

We have to control planetary motor because it has very high rpm.So, we use a motor driver to control it.

When, we tested the bot, major problems we faced were

(a) All links are not in one plane.

(b) Because of the high weight, bot was not stable.

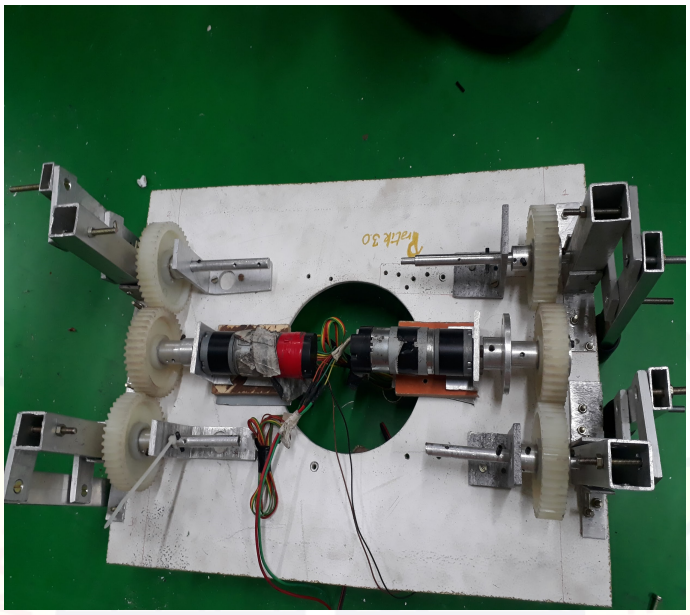
(c) As time-belt is not ready made, we cut it and and then joined it, so it was accurate and we faced slip of belt problem.

2. MODIFIED THEO JANSEN MECHANISM

   On analysing mechanism we came to know that there is no relative motion between few links of the mechanism, so we removed those links and replaced by a single link which was made of composite. And also we merged both cranks and made single crank Theo Jansen.
   Due to modification number of links were reduced and difficulties of motion between the links.

3. USING DOUBLE CRANK

   After trying Theo Jansen, Modified Theo Jansen, and Klann Mechanism we tried Double crank sort of assembly for the movement.
   This mechanism is shown-



We used two planetary motors for this mechanism. As seen, Each leg has two cranks and two links. Every link according to appropriate dimensions were made. Also,We used 6mm Thick ply for the base of the bot. All together was assembled and the final assembly is shown in the figure above.

The problem with this mechanism was the stability. Also, the constrained motion of links were observed. Unlike our expectation, both cranks rotated in different directions. As a result these mechanisms was not giving desirable outputs.

4. Klann mechanism

   In klann mechanism the X and Y displacement are comparatively more and the number of links are less so it is advantageous.
   For the manufacturing we decided to use acrylic to make link as it provides good strength and the chassis was made with composite because of its low weight with enough strength. It was decided to make to mechanism with smaller scale than the actual for testing. Also two planetary motors were used to provide the motion. So in the manufacturing the links, chase, the L clamp of required size were made and mounting was done. Problems faced during assembly were the under and over meshing of gears which we solved after. During testing in air sometimes the plane of the link gets disturbed so motion gets blocked so locknuts were used. 180 degree phase difference between legs was established and was kept on the ground. We observed that the bot was stable and can stand on four legs easily. Mechanism is designed as at the time of walking on the ground, two legs are in the air and two stays on the ground, but it couldn't be stable on two legs. So it was rejected and better design is need to be think.

KLANN MECHANISM USING VARIABLE SPEED GEARS:

As of now, we had tried several mechanisms in which the gait patterns consist of two legs on the floor at a time and two in air. Due to this, The stability desired was not achieved as the bot used to topple on the side where leg was in the air. To avoid this problem, at least three legs should stay on the ground. This was achieved by using a variable speed gear assembly which is shown in image below-

In this assembly, 3 variable gears using Acrylic as material were made. As the driven gear has smaller diameter for first 180° and larger for another 180°. By synchronising this gear with driver gear, we got the design given in the picture. Driven gear rotates fast when it is subjected to smaller diameter, so this is how the leg was supposed to move fast when it is in the air. It moves with regular speed for another 180° when it is on the ground.

Due to variable speed gears, the 3 legs remained on the ground while one was in the air. Stability up to an extent was achieved. The main problem we faced in this mechanism was of meshing of gears. As the shape of the gear is not uniformly changing, the meshing was difficult task to be done.

At the time of testing the part of gears where it changes it's diameter gave a jerk and didn't rotate smoothly. Also the torque required from the motor was different for different instant but motor couldn't compatible with it, so we couldn't get the desired output.

5. WALKING MECHANISM USING SLIDER, RACK AND PINION.

In this mechanism we have used two sliders and 2 pairs of rack and pinion and 4 piston of stroke length 10 cm. Basic chassis is in the shape of a plus

sign. First of all horizontal piston will be actuated due to which the bot will be lifted upon on 2 legs and then motor will be actuated and with the help of rack and pinion, the rotational motion will converted into horizontal linear motion of the upper part of bot. This will result in the forward motion of the upper part of the bot and the motor will be actuated until there is no signal received from limit switch. Signal received from limit switch will indicate that the upper part has reached the maximum position. Now the pistons which are lying on the vertical line will be actuated and after that, two pistons which lying on horizontal line will be closed and motor will be actuated once again in reverse direction, thus the lower part of the bot will come forward. Result of this will be that the bot will move in the x direction by a distance equal to the displacement of the pinion in half cycle.

6. Jumping mechanism

1st test Actuating a single piston

- The birth of this basic jumping bot was from the fact that when a pneumatic cylinder is actuated at a certain pressure it causes the cylinder to jump or causes it to gain height due to reaction force obtained from ground on the action of piston actuation when in ground contact.
- The picture of single piston actuation test is as shown below.

Conclusion

- The piston used was of 25 mm bore diameter and 100 mm opening length
- At a pressure of 2 to 2.5 bar the piston was reaching a satisfying height of 8 to 10 cm above the ground, where the only weight being lifted was the self weight of the piston assembly which was around 700 to 750 grams.
- As weight was increased to 2kg in the piston by tying some weight on it the height of jump above the ground decreased to 6 to 7 cm.
- The same height with weight was obtained by increasing the pressure from previous pressure to 4 bar.

2nd test

Set up of 4 legged robots –legs at a 90 degree to the platform or to vertical.

- After the test of single piston for its jump, we set up a four legged bot i.e 4 identical piston leg assembly attached to a plywood of 400 by 400 mm in dimension and 10 mm thick.
- This set up weighed nearly of 3.5 to 4 kg on weighing.
- The piston assembly was set up on plywood on which the piston was attached using clamps.
- The plywood leg set up was attached to base of bot using aluminium 3mm 'L' clamps.
- The pneumatic controlling system with air supply was set up
. Picture as followed

Conclusion

- On actuating all the 4 pistons together the bot would jump to a height of 8 to 9 cm at a pressure of 2.5 bar
- Max height obtained of about 18-20 cm at a pressure of 5.5 bar.
- Appreciable height of 15 cm was obtained after putting a weight of 3 to 3.5kg on the bot .
- Only displacement in the y direction was obtained.

Problems

- Bending of clamps of legs due to high reaction force.
- Problem of perpendicular clamps and in same plan of each leg, as different planes resulted in force in different direction resulting in change in landing position of bot after the jump.
- Controlling the speed of piston opening and opening all pistons at exact time was a challenge as there was a lag between piston opening due to length of pipe used to make the pneumatic circuit of each leg.
- The air consumption rate was quite high and a large storage for the same was needed for more cycles of the piston actuation.

3rd test

- Legs at an angle of 45 degree with vertical

- Now the legs were set up at an angle of 45 degrees to the vertical.
- Now instead of wooden base for legs aluminium box section was used as a base and the piston were mounted on same using zip tie to speed up the testing process and to cut the time of making clamps again.
- Aim was to obtain a displacement in x and y direction at the same time.

Conclusion

- A reasonable displacement in the y direction 15 cm and in x direction 15 to 18 cm was obtained on a pressure of 4.5 bar on actuation of piston.
- The experimental setup of using zip tie did work without budging even a little bit and thus holding the piston tightly.

Problems

- The air consumption rate was quite high and a large storage for the same was needed for more cycles of the piston actuation.
- Balancing of bot was difficult after it landed due to improper alignment of legs and non planar set up due to dimensional inaccuracies.
- Bending of legs from 45 degrees to a higher angle was also an issue.

4th test

- Legs at an angle of 30 degrees to the vertical
- This setup was done in order to improve balancing of the bot after its landing.
- Again the leg set up was used as it was in the earlier set up of 45 degrees.

Conclusion

- Results were satisfactory enough with y direction displacement greater than x direction displacement.

Problems

- The air consumption rate was quite high and a large storage for the same was needed for more cycles of the piston actuation.
- Balancing of bot was difficult after it landed due to improper alignment of legs and non planar set up due to dimensional inaccuracies.
- Bending of legs from 30 degrees to a higher angle was also an issue.

7. Piston based walking mechanism

   Using of piston for y direction displacement and use of motor for x direction displacement simultaneously. One leg set up tests

   1st test

   One leg set up to see the tip motion graph of the leg along with usage of high torque servo motor.

   - Servo motor used in order to restrict the motion of leg in an angle of 60 degree i.e plus minus 30 degrees.
   - Also servo motor has low weight and appreciable torque.
   - The assembly was that the same aluminium piston leg made earlier was directly attached to the servo using its star accessory.
   - The servo assembly was mounted on plywood of 12 mm thickness such that the total weight was 3kg.

   Conclusion

   - An expected motion of simultaneous X AND Y direction was achieved easily with the actuation of motor and piston simultaneously with approx tuning of both for the desired motion.
   - Weight of 3 kg was being lifted easily by one leg assembly without any kind of shuffling at the ground by the leg.

   Problems

- The shaft of servo often started to rotate without the star accessory mounted on it or the leg, because of the wear and tear between the accessory and the shaft which acted as a pinion and circular rack setup. Thus resulting in failure of the leg. The torque and motion were more than sufficient but due to a very small shaft of the servo mounting of leg on that shaft was very difficult.

2nd test

- Instead of servo a mini stepper was used (name 17)
- Set up same as servo motor.

textbfConclusion

- Mini stepper did not provide enough torque for the expected motion as well it was heavy compared to servo motor thus the set up failed.

3rd test

Use of nema 23 stepper motor

Conclusion

- Again torque was the issue as it wasnt able to lift its own weight of 1,1 kg so lifting the leg was out of bounds.

4th test

Use of chinese motor (smaller mechtex)

In order to restrict the motion of the motor it was used with an encoder Set up was done using gear where in leg and encoder were connected to driven gear using shafts. Set up is as shown below.

Conclusion

- Torque problem was solved.
- Desire motion was obtained through the bot got a bit heavy but motion was good enough as it was able to lift about 4 kg including the set up assembly weight.

Problems

- Tuning i.e simultaneous motion of motor and piston to provide the right path or graph.

Walking mechanism based on the 4th test

After the success of 4th test a whole bot with all 4 legs was made.
After completion of the bot it weighed 8.5 to 9 kg. The idea behind its motion was to actuate two diagonal legs together and other two legs would not be actuated in y direction but will rotate in opposite direction of one's being actuated.
The motion idea was sourced from the walking style of animals which is called trotting in which two diagonal legs move together.

Problems

- All the components available for mounting were not uniform as the encoders used had different diameter shaft , thus spring couplers of same were not available as such.
- Gears of the same gear ratio were not available so two gears with different gear ratio were used.
- Tuning required for the synchronized motion of the 4 legs was not achieved as the motor was able to provide enough torque when piston was not actuated but once it was actuated the amount of torque required resulted in the drawing more current by the motor resulting in excessive heating up of the circuit elements. It also resulted in the malfunctioning of circuit due to overloading.
- Also the timing for two diagonal pair of motors to work together was not achieved due to the difference between the performance of both motors in loaded and unloaded conditions even though the circuit was receiving feedback from encoders.

- Also once the pistons were actuated it was necessary for the time difference to be minimum between the motion of the motor and the release of air from the piston.

  After so many attempts and considering the hardware limitations this idea was scrapped.

- **FOUR BAR MECHANISM FOR WALKING (FOUR LEGGED ROBOT)**

  Four bar mechanism using single motor:-
  The whole idea was by seeing a YouTube video.
  The link is given below:-
  https://www.youtube.com/watch?v=T5c4j9Ou9EE

  For the four bar mechanism different CAD models were made by changing

the gait pattern of the leg.

Firstly the four bar mechanism using one motor was manufactured in such a way that the motor is mounted vertically up and then by using bevel gear setup the power was transmitted in both sides of the legs. The 180 degree phase difference was maintained between the two legs on the same side by the gear setup. By seeing in the video we used long rods at the bottom of the legs.

The long length of the rod was giving good stability to the bot while it was walking.

Here we faced several problems like touching of motor on the sand dune while crossing it, which was against the rules of the competition and the other problem was the backlash in the gears due to which error of upto 10-15 degree used to occur in phase of the legs which cannot be neglected.

So to overcome the problem it was decided to mount the motor upside down by using the same as the rest of the setup. But turning was not possible in this type of setup.

Two motor setup for four bar mechanism
For these two planetary motors were used and along with it the gear setup was used to transmit power and also encoders were used on both sides to get feedback.

We then designed a mechanism taking hint from a video on Youtube
Final entry of robot legs made of four bar mechanism and its assembly

In four bar walking mechanism

1)Driving with 2 motors and gear setup

Driving was done with 2 planetary motors. 1 motor controls 2 legs of one side of MR2 and other planetary motor controls another 2 legs of the other side. Power is transmitted by using gear setup. Our Required torque is high to drive MR2. We used 4 gear setup for one side to drive two legs. Motor is connected with one gear using coupler. Two cranks are connected with oth6er two gears respectively. Motor gear are connected with one crank gear also. One gear used to connect two gears of crank. When motor operates, Gear connected with motor rotates, By this crank gear which is meshed with motor gear also rotates. Gear ratio of crank gear to motor gear is high. So, torque increase. Connecting gear also rotates with crank gear and provide motion to another crank gear due to meshing. Here, both crank gears have the same number of teeth. So, torque generated at that points are same.

Here, two motors are connected diagonally to each other like, on right side, motor connected in front side, then on the left side, motor connected rear side.

For driving four leg mechanism perfectly, First requirement is that both legs of one side has 180 degree phase difference between them. So, when one leg is at upper position, another leg is at bottom position. By this, toppling of bot is ignored.

In this setup, all gears are connected linearly with chassis by use of pedestal bearings.

2) Turning Mechanism in Four Bar mechanism

BASED ON CAR STEERING MECHANISM

Idea conception based on car steering mechanism.
The idea was to somehow make the front legs to turn or steer in left or right direction as per requirement.
We implemented this in the 4 bar walking mechanism.
For this we had to make the fixed links of the legs to be mobile to make the whole system of front legs to take left or right.
For making the fixed links points movable we introduced self made sliders for changing the position of fixed link joint.

In order to move the sliders / rotate them through an angle some sort of actuation was needed, for which we tried using pneumatic actuators. The logic was to set up small pistons attached to the sliders such that when the piston was actuated it resulted in the bot turning left and when the piston was retracted the bot took right turn. When it was semi actuated it resulted in straight line motion. and for transmission of power from motor we used universal joint in between to make the front movable compared to the back.
The whole bot was powered by a single motor with use of 90 degree coupler gear system to transmit power to back and front legs.

Universal coupler helps in transmitting power at various angles.

Problems

Issue with smooth motion for turning of the robot.

Restrictions in power transmission due to the robot dimension constraints.
The force required to actuate the pneumatic was very high which was not achieved using 6 bar pressure limit.
The self made sliders did not work smoothly as it was expected. Custom made sliders from the market were costly and increased weight and complexity.

Bevel gear mounting and its meshing set up was difficult, which was done for 90 degree transmission.
Mounting of the pneumatic actuator on the robot was also a challenge as the actuator itself needed to be movable so that it can rotate the legs as and when actuated.

USING UNIVERSAL COUPLING

In this setup universal coupling is used to try turning in 4 bar mechanism.

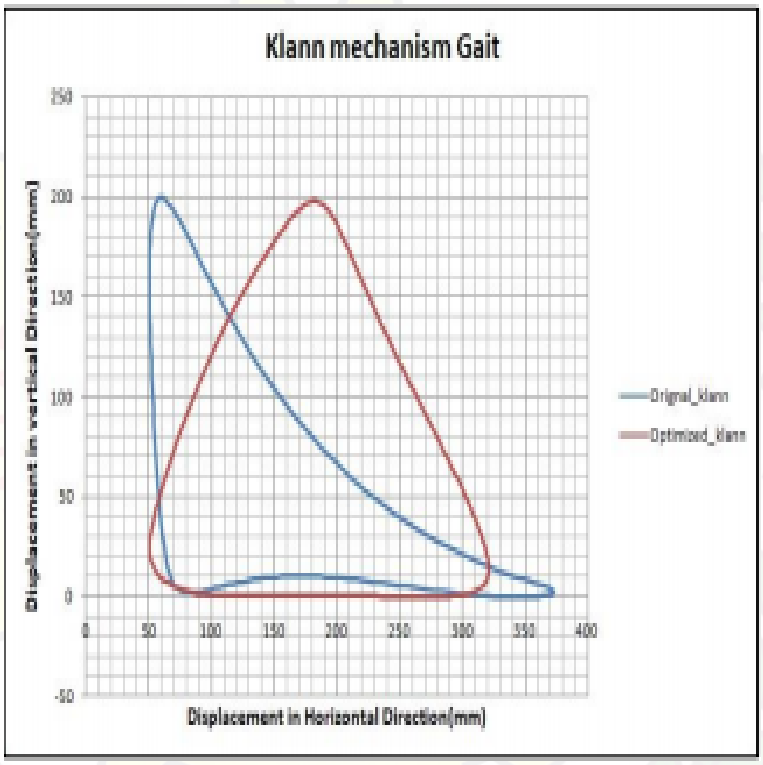
In this mechanism also one motor was used which was mounted vertically upwards and power was transmitted in the front and back side with the help of bevel gear setup. So another two bevel gears were also used one each on the front and back side to transmit the power coming from motor to respective legs.
Now for turning we introduced a universal coupler between the two bevel gear setup on the front side i.e. the shaft coming out from the bevel setup of motor then universal coupler and then the bevel setup on the front end.
Now by using pneumatic piston on the front side to change the angle of shaft as per required rotation of robot.

But due to the use of so many gears there was a large amount of backlash due to which there was considerable problem in the walking which cannot be ignored.
So the idea of bevel gears was rejected.

Modified Klann Mechanism:

Modified klann mechanism works on the same principle used in klann mechanism. Front and rear legs are at 180 phase difference and both sides of legs are at 180 degree phase difference.

In modified klann mechanism , we modified gait pattern of mechanism by changing dimensions of links. Blue color pattern shown in fig is gait pattern of klann mechanism



And Red one is gait pattern of modified klann mechanism

Requirements of Optimizing Gait Pattern:
According to problem statement, we have to cross sand dune and tussock. For this, We want minimum 16 cm lift of one leg to cross sand dune and tussock.
In klann mechanism gait pattern, there are many sharp points in pattern. So , when leg moves in this pattern, sudden jerk generated in leg. Due to this, There is a chance of bending chassis, links and bolts. By optimizing pattern, sharp points are removed at some basis.
Here, we also want symmetric gait pattern to cross tussock. Because, in tussock there are two strings at some distance. So, when front legs crossed

the first string and at starting point of other string, Rear leg must be at upper position for crossing the first string. So, if we don't have a symmetric gait pattern, there is difficult to cross both strings simultaneously.

DIFFERENT DRIVING MECHANISM USED IN MODIFIED KLANN MECHANISM:

Gear setup

We used 2 types of gear setup.
1) Normal Gear Setup
2) Compound Gear Setup

1) Normal Gear Setup:

In normal gear setup, initially we connected all gears linearly which discussed above in four bar walking mechanism.
Then We assembled gear in such a way that connecting gear mounted at upper side of chassis.

PHOTOS OF MODIFIED KLANN CHASSIS IN WHICH GEAR CONNECTED LINEARLY AND AT UPPER PART OF CHASSIS

2) Compound gear setup
We used compound gear setup for more torque.
Compound Gears are used at both gears connected with shaft. 2 gears of different number of teeth are connected with crank shaft.

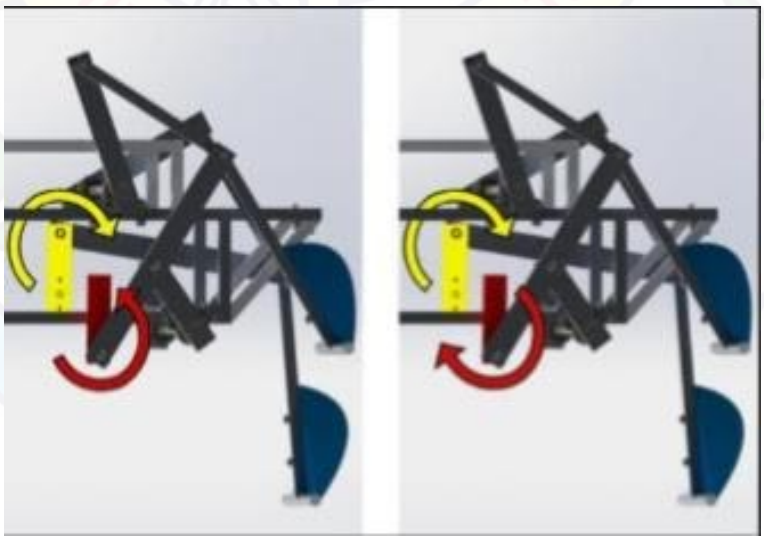
2) Chain Setup In this chain setup, gears and chain are used to drive legs. In this, 2 gears are used . First gear is connected with motor and other gear is connected with crank shaft. Motor is connected with gear and this gear drive other gear connected with crank shaft. 2 sprockets are connected with 2 crank shaft. Chain are connected between it. When motor operates, gear rotates and sprockets also rotates. Due to this, chain moves and drive other crank by this.

Advantage: Less number of gears are used in this We don't have to face meshing problem. Disadvantages: - Lagging in chain occurs - Chain Utri jati hti .

5.2 ELECTRONICS

5.2.1 Walking & Rotation Algorithm

- The basic idea for functioning of this bot is, we rotate both the motors in forward direction, which results in forward motion of robot, and at point of turning, we rotate one motor in clockwise direction and the other in anticlockwise, or vice-verse. During this motors should not loose sync.
- By sync, what we mean is, both the motors are moving exactly like each other, ie. maintain the same instantaneous speed. This is achieved with help of our control system, and phase PID algorithm was used to maintain the phase of each motor.
- There is also a backup, in case excessive torque results in sudden difference in angle. If the motors are not able to maintain the same angle, and one is lagging far behind, phase correct algorithm is used.
- **Phase PID**
  - To make robots forward and backward motion smooth, we have used phase PID algorithm. The robot will perform smooth forward and backward motion, only when we move both the motors in forward and backward direction at same angular speed.



  - In this algorithm, we take feedback from our motors, using Rotary Encoders, and hence obtain current angle of the motors. Now we find the

error by subtracting right motor angle from left motor angle. This angle is passed through a PID controller, which results in an output. Now according to the error, we obtain PWM for right and left motor.

- Therefore, if
  $Error > 0$:
  $RightPWM = BasePWM + Output(fromPID)$
  $LeftPWM = BasePWM$

  $Error < 0$:
  $RightPWM = BasePWM$
  $LeftPWM = BasePWM + Output(fromPID)$

- This ensures synchronized motion of both motors.

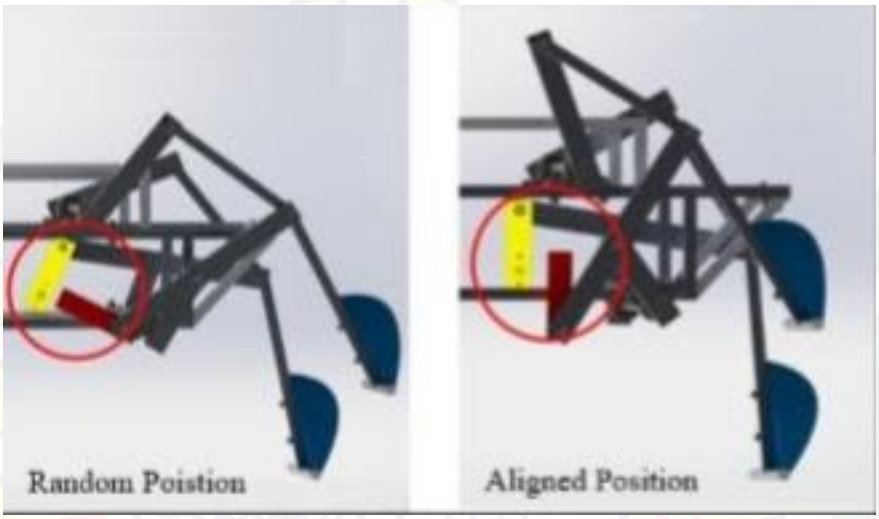
- **Phase correct**
  - Our bot followed a gait pattern, which required variable torque at each point in time. But at some points, these torque requirements are very high, which causes motors to loose synchronization.
  - To prevent this from happening, we have used phase correct algorithm. In this algorithm, as soon as phase difference between the motors exceeds a specific threshold value, we stop the motor which was ahead, and other one is given base PWM.
  - Therefore, if
    $Error > Threshold$:
    $RightPWM = BasePWM$
    $LeftPWM = 0$

    $Error < Threshold$:
    $RightPWM = 0$
    $LeftPWM = BasePWM$

  - This ensures smooth motion of robot, even when the motor requires a high torque.

- **Align**

- The robot had a specific characteristic, that it would support rotation, only when we start from a specific alignment of crank.
- To reset our bot into this alignment from any alignment, we developed an align algorithm. Here for sake of simplicity, we link motor rotation with rotation of cranks.



- The basic concept behind this algorithm is, whenever we need to switch from forward/backward motion to rotation motion or vice - verse, we have to use align algorithm, to help robot achieve its stable alignment.
- Here the aligned state of the bot is stored, in form of initial angle. When robot receives the command to align, the motor, irrespective of its current position has to reach to reach to its initial angle position.
- This was achieved with help of keeping in mind the previous direction of crank rotation and current position of crank with respect to initial position.

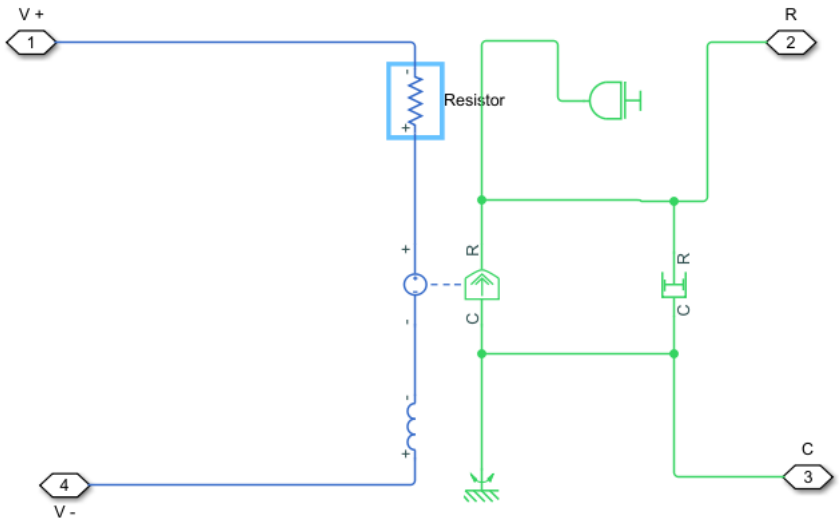
5.2.2 MATLAB Simulation

- For purpose of analysis, we implemented our control system in MATLAB's Simulink. Various tool-sets in the Simulink library such as the Simscape. an external library such as Simscape multi body physics and contact forces along with the foundation library toolboxes were all that fulfilled our requirements regarding the designing and modelling.Testing of models for Phase-Correct and PhasePID block was done using the Simscape library which helped in error

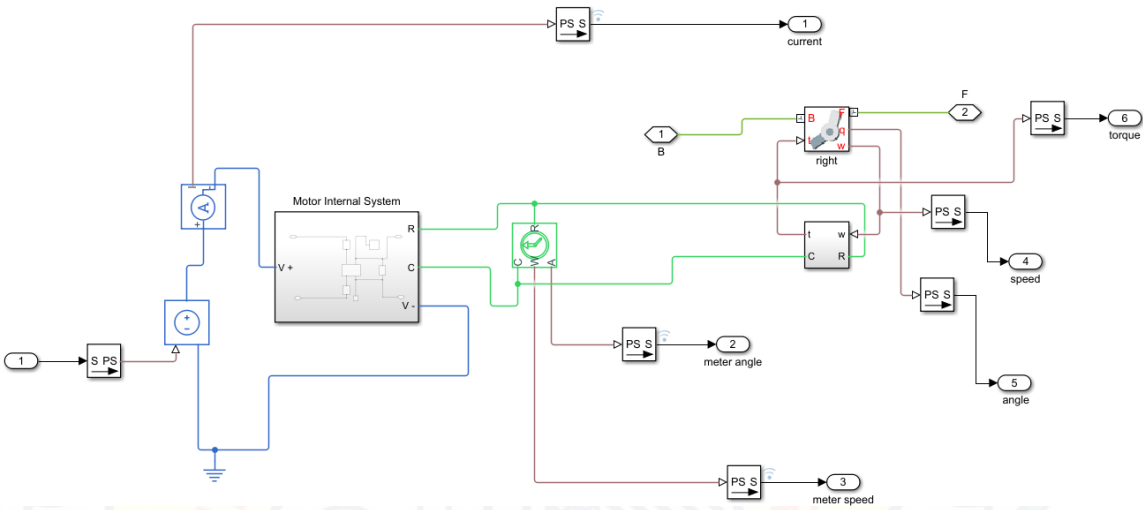
recognition of the control system. To check feasibility of the control system, we implemented it on a four-bar mechanism.

place here pic of main blocks of matlab

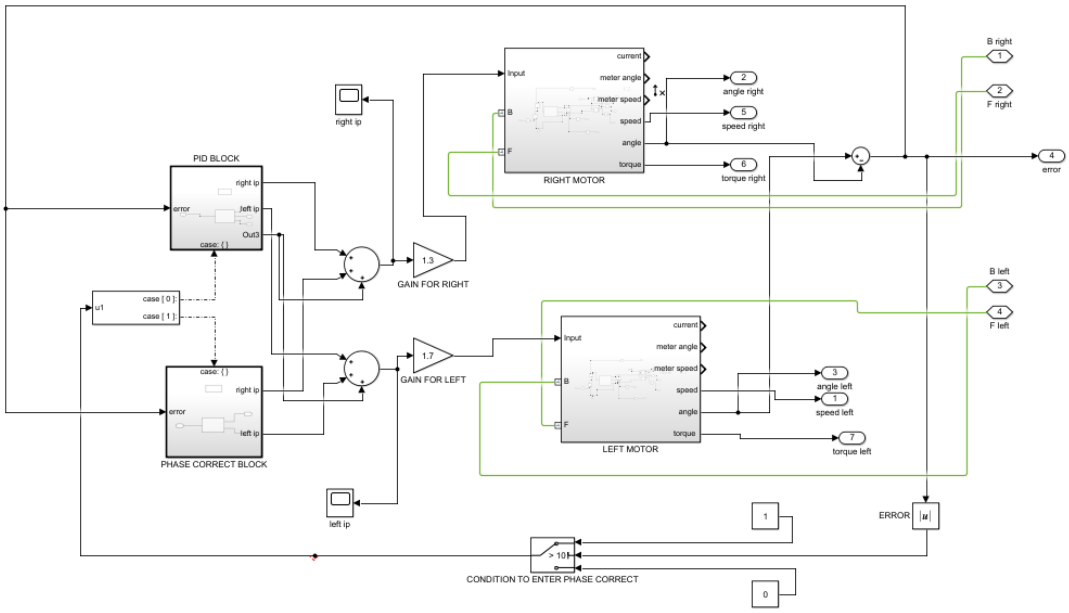
- For the purpose of implementation of this system, we had to construct some basic modules like motors, on which we may apply our control system for analysis.



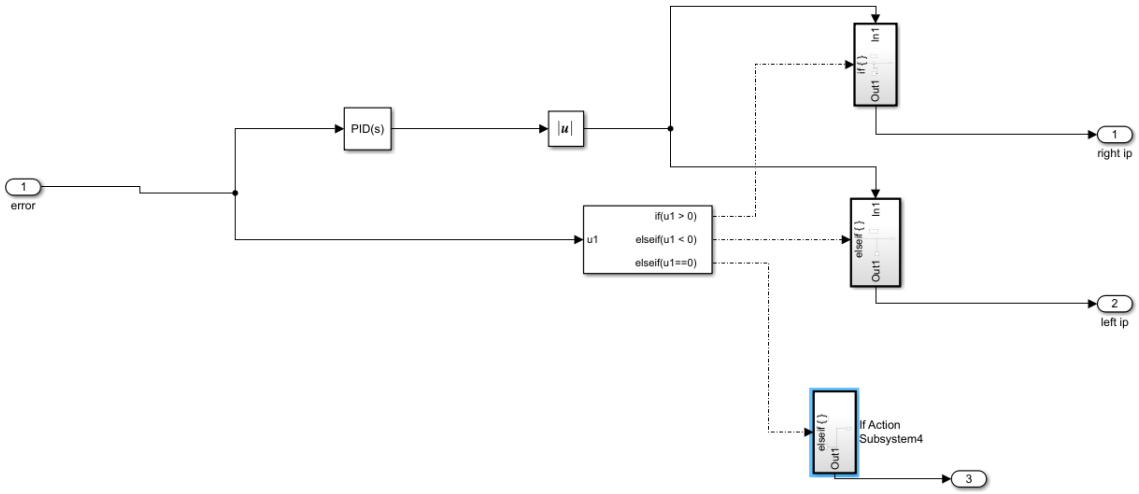
- Fig. is the basic design of motor, which consists of all characteristics of a motor. We have made it using Simscape library in Simulink. This gave us the access to all the physical aspects of a motor.



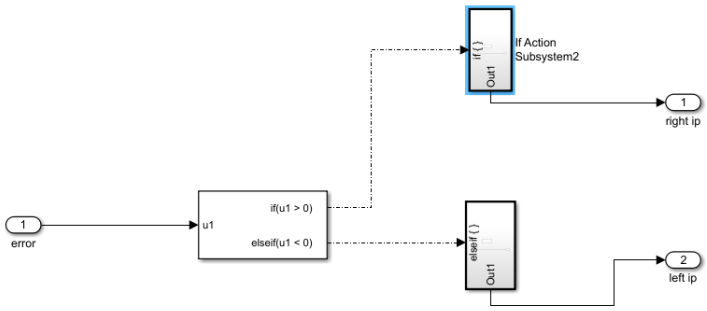
- Now, we used a controlled voltage source, current sensor, etc. to make motor controllable with help of some input voltage value. To convert these physical signals to Simulink units, we use a PS-S converter and a S-PS converter for vice-versa.



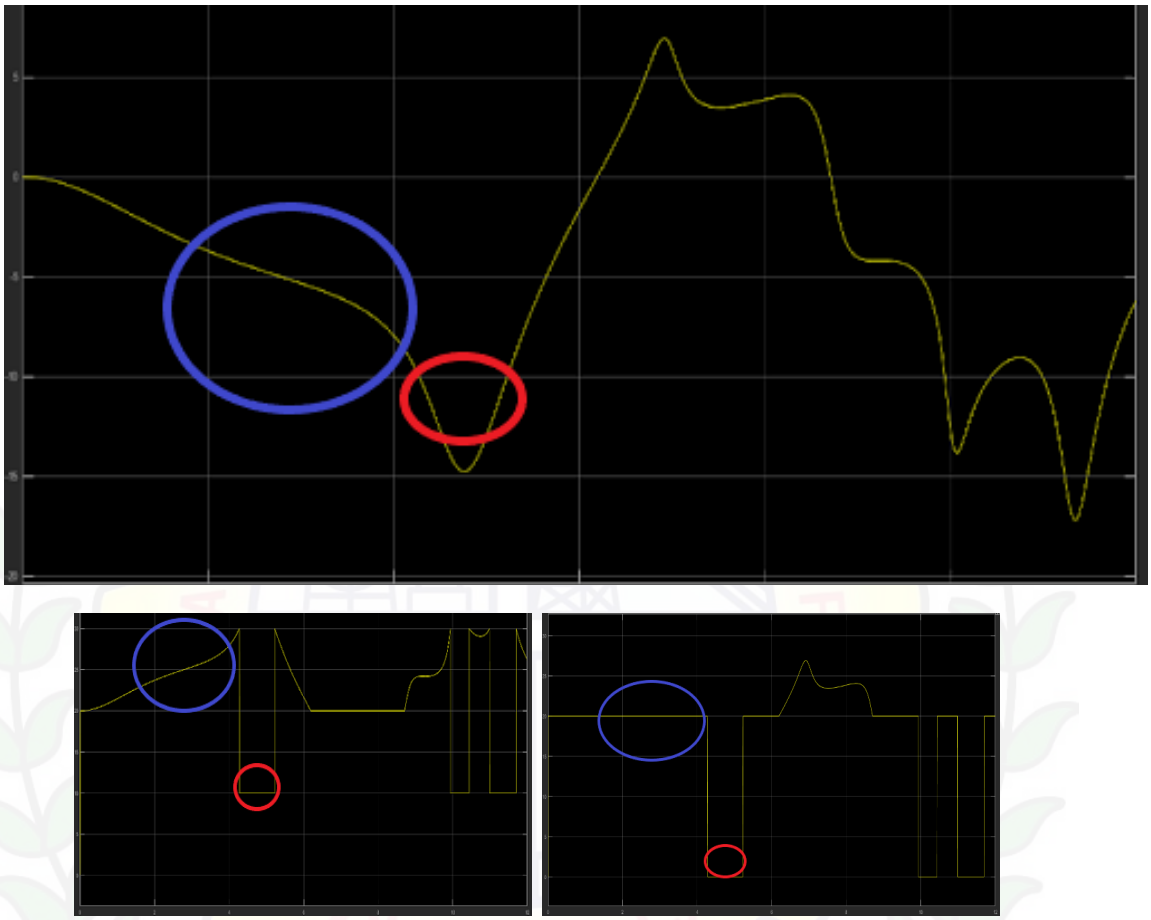
- Fig. shows the control system block. It consists of Phase PID and Phase Correct block which assign appropriate input to both motors.



- Fig. shows the internal arrangement of Phase PID block.



- Fig. shows the internal arrangement of Phase Correct block.
- MATLAB gives us the feature to plot the desired entities, for the purpose of analysis of system. We have plotted here error, right motor input and left motor input.



- In above figures, we see both Phase PID and Phase Correct in action.
- The Blue markings show working of Phase PID. One motor is moving at base PWM, while other is moving at an increased speed, in correspondence to the angle difference (error). But as soon as error exceeds a certain limit, we observe a change in graphs.
- At this point in time, Phase Correct is executed, and we observe one motor PWM is reduced to 0, while the one which was lagging behind is now given Base PWM. This results in reduction in error, and as soon as error is under permissible range, we observe that Phase PID again starts controlling the motors.

5.2.3 Set up for Raspberry Pi-3 :

- OS setup :

  1. Download the latest rasbian Jessie from official website of Raspberry Pi.
  2. Download win32 disk manager.
  3. Format the SD card and then burn the image file of Jessie in card.
  4. After the burning is complete you need to make a ssh file for the new versions of Jessie.
  5. Open the memory card in your PC and make an empty notepad file named as "ssh" and remove the extension ".txt".

- Accessing Pi using Putty :

  1. Download Putty on your PC.
  2. For connecting to Pi for the first time plug in Pi into your PC with the help of LAN cable and Power up Pi.
  3. Now you need to share your virtual WiFi first with your LAN (Ethernet).

  - For sharing

    (a) Open Network and Sharing centre → change adapter settings → Select your WiFi

    (b) Select properties of the WiFi. Select Sharing tab check both sharing check-boxes and Select appropriate network to which it is to be shared (in our case Ethernet) → apply changes.

  4. Run command prompt under admin permissions and give command `arp -a` keep doing this until you get a dynamic address of Raspberry Pi.
  5. Try this dynamic addresses in putty one by one until you get the right one in host name (or IP address) under ssh connection type.
  6. Once you get the right address, you will be asked for login id. Then you will be asked for login password for Pi.
  7. Now you can access pi through its terminal (command line).
  8. VNCserver is by default installed in Pi use

     v n c s e r v e r

     or

```
vncserver -geometry 1024x728 -depth 24
```

command in command line of Putty in Pi.

9. If VNC server is not installed in Pi then use the following command in Putty terminal of Pi to install VNC server

```
sudo apt-get install tightvncserver
```

- Setup VNC Server on pi :
  1. Download VNC server on your PC.
  2. In VNC server enter the IP of Pi.
  3. You can use GUI of VNC server after connecting to Pi by entering Username and Password of Pi.
  4. Open terminal in Pi.
  5. To change password for VNC server type `vncpasswd`then restart VNC server by command `service vncserver restart`.
- VNC server autostart on boot up :
  1. Method 1 :
     (a) Execute `sudo nano rc.local` command in terminal of Pi which will open file rc.local in nano editer.
     (b) Above the line `exit 0` enter the following line

```
vncserver :1 -geometry 1024x728 -depth 16
```

     (c) Save changes done in the file by `CTRL+X` → `Y` → `Enter`
  2. Method 2 :
     (a) Execute following commands in terminal of Pi :
        i. cd /home/pi
        ii. cd .config
        iii. mkdir autostart"
        iv. cd autostart
        v. sudo nano tightvnc.desktop
     (b) The last command (e) will open a file in nano editor. Type the following commands in file as stated below

```
[Desktop Entry]
Type=Application
Name=TightVNC
Exec=vncserver :1
StartupNotify=false
```

*(you may change `exec= vncserver :1 -geometry 1024x728 -depth 24` to set your desired geometry for your VNC autostart)

(c) Save changes done in the file by `CTRL+X` → `Y` → `Enter`

- Enabling I2C on Pi3 :

1. I2C is not turned on by default.
   (a) Use `Sudo raspi-config` command in terminal.
   (b) Select `9 Advanced Option` using down arrow key.
   (c) Select `A7 I2C` using down arrow key.
   (d) Select `yes` to enable I2C.
   (e) Select `yes` when it asks about automatically loading the kernel module.
   (f) Use the right arrow to select the `<Finish>` button.
   (g) Select `yes` when it asks to reboot.
2. Type `sudo nano /boot/config.txt` in terminal of Pi and add following lines to it :

```
adtparam=i2c1=on
dtmparam=i2c_arm=on
```

3. Save changes done in the file by `CTRL+X` → `Y` → `Enter`
4. Type `i2cdetect -y 1`. If you get *I2C detect is not a recognised function* then execute step 5 and 6.
5. Connect internet to your Pi.
6. Install i2c- tools package on Raspbian using `sudo apt-get install i2c-tools`.

- Setting up UART on Pi for serial communication :

1. First u need to disable console accessing over serial .type `sudo raspi-config`.
2. Go to advanced option select serial.
3. Select enable/disable login shell access over serial to no(disable it).

4. Reboot Pi if asked.
5. Now open terminal and type `sudo nano /boot/config.txt`.
6. Add the following lines into file

```
    enable_uart=1     //if it exists already change enable
to enable_uart=1
   core_freq    =250
*(by default the serial port is /dev/ttyS0 and Bluetooth
port is /dev/ttyAMA0 in pi3 if u want to disable
Bluetooth and swap ports add following line below the
above lines
>if you want to change the blutooth to miniuartport(bad)
dtoverlay=pi3-miniuart-bt
>if you want to disable the blutooth(good)
dtoverlay=pi3-disable-bt)
```

7. You can use serial communication on port `/dev/ttyS0` after restarting Pi.Ensure that ports are not switched.

- Setup minimu9 on Pi3 :

1. Enable I2C using above steps.
2. Verify that I2C is enabled, using `i2cdetect {y 1`. It should display `/dev/i2c-1`.
3. Install minimu9-ahrs using the Debian package. You can find the latest Debian package at
   `http://www.davidegrayson.com/minimu9-ahrs/debian/` .
   Copy the URL of the latest `minimu9-ahrs_*.deb` file. Use the `wget` utility to fetch it from the web to your Pi board, and then use `dpkg -i` to install it.
   Example for installing minimu9-ahrs version- 2.0.0-1 :

```
wget http://www.davidegrayson.com/minimu9-ahrs/debian/
minimu9-ahrs_2.0.0-1_armhf.deb

dpkg -i minimu9-ahrs_2.0.0-1_armhf.deb
```

4. Connect IMU to Pi as shown below :

Raspberry Pi pin	MinIMU-9 pin
GND	GND
3V3 Power	VDD
GPIO 0 (SDA)	SDA
GPIO 1 (SCL)	SCL

   Do not provide 3.3 V to IMU from Raspberry Pi as it may draw more current from Raspberry Pi pin than its ratting, which may cause damage to Raspberry Pi.

   Make sure that ground is common.

5. Run `minimu9-ahrs -b /dev/i2c-1 --mode raw` to see raw readings. The output should look something like this:

```
-138  129  -416  112  -8  228  -50  14  9
-138  129  -419  120  -4  232  -49  20  18
-138  129  -419  116  -12  228  -51  15  8
-138  129  -419  116  -12  228  -50  21  17
-137  130  -421  116  -8  232  -51  22  11
-137  130  -421  120  -12  220  -56  20  14
```

   There will be noise in all the readings, even if your IMU is not moving at all. That is normal for any sensor.

6. Now calibrate IMU by running following command in terminal of Pi : `minimu9-ahrs -b /dev/i2c-1 -calibrate`.

   Follow onscreen instructions. You run the above command it will start calibrating.

   It will show some tuple error if you are using the same version mentioned above to solve tuple error follow step 7.

7. To solve the tuple error, run the following command is terminal of Pi `sudo nano /usr/bin/minimu9-ahrs-calibrator`.

   Edit the 100 th line of code by changing

   `args=[raw\_readings],`

   by

   `args=(raw\_readings,),`

   Follow the step 6 again to calibrate the sensor until it's successful.

8. Now you can get Euler angle with the help of following command `minimu9-ahrs --output euler`.

   It will print a stream of nine floating-point numbers per line; which are pitch, yaw and roll respectively in x, y and z axis in degrees.

   All three Euler angles should be close to zero when the board is oriented with the Z axis facing down and the X axis facing towards magnetic north.

   From that starting point:

   - A positive yaw corresponds to a rotation about the Z axis that is clockwise when viewed from above.
   - A positive pitch correspond to a rotation about the Y axis that would cause the X axis to aim higher into the sky.
   - A positive roll would correspond to a counter-clockwise rotation about the X axis.

   If you face any trouble setting up min IMU follow the following page `https://github.com/DavidEGrayson/minimu9-ahrs/wiki`

5.2.4 Image Processing :

- Python and OpenCV on Raspberry PI:
  For installing Python and OpenCV on raspberry Pi follow the link given below :
  `https://www.alatortsev.com/2018/09/05/installing-opencv-3-4-3-on-raspberry-pi-3`

  Problems which may arise:

  - If there are any memory issues when you try to install softwares and packages like python and OpenCV as it may run out of memory, then you can check it by using the following command:
    `df -h`
  - It would display the memory usage and the available memory on the SD card.
  - If the SD Card runs out of memory, you may not be able to run the VNC server i.e. when you run VNC server on putty it may login to server but again asks for login information in the VNC server.
  - This Error can be handled by uninstalling some package through putty itself which will help freeing the memory.
    `sudo apt-get --purge remove <package>`

– Here the package refers to the name of the package you want to uninstall from pi.‘

To get basic idea of opencv functions , you can follow the following website : `https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html`

- Colour Detection Approach:
  1. First assign the frames coming from camera using `cv2.VideoCapture()` to a variable.
     Initally set the dimensions of the frame and the frame rate using `cap.set()`.
  2. Read the frame using `cap.read()` function.
  3. Apply some kind of blurring.
     Eg : Gaussian Blur for reducing the noise in the image.
  4. Convert the RGB image to HSV image using `cv2.cvtcolor` function in order to make it independent from pseudo illumination. **NOTE:** HSV is a better color space than RGB for image processing.
  5. Set the lower and upper threshold and pass it to `cv2.inRange` function for creating a mask corresponding to that color.
  6. Use `cv2.bitwise_and` function to create a binary image from the mask obtained from the above step.
  7. Apply contour to the image and find the area of the corresponding color using `cv2.contourArea` and if its greater than a threshold value than the color is detected.
- Sending Data over Serial through Raspberry PI :
  1. Method 1 :
     Print data in the file, to be sent.
     Eg : `print('A')`
     Now Run the file.
     Eg : `python filename.py /dev/ttyS0`
     This indicates that we are sending data that is printed in the terminal itself through Serial 0.
  2. Method 2 :
     Another way is to directly transmit the data by importing Serial in the file itself.
     Eg :

```
import serial
ser=None
ser = serial.Serial('/dev/ttyS0', 9600)
ser.write('A')
```

This will transmit 'A'.

- Parallel Processing using Threads :

  Threads can be created and can be assigned to tasks that are independent of each other. Threading is a way to run two processes independently and they can share data.

  Eg : For reading frames corresponding to 2 cameras on Raspberry pi serially in loop, instead we can assign 2 different threads to 2 cameras so that they can execute their processes in parallel.

  ```
  import threading
  t1=threading.Thread(target = camera1, name = 'thread1')
  t2=threading.Thread(target = camera2, name = 'thread2')
  t1.start()
  t2.start()
  ```

  This indicates that 2 threads namely t1 and t2 are created and are binded with camera1 and camera2 functions i.e these functions are called when the thread runs. The threads are started using the start() function as shown above.

  The thread needs to be executed and stopped when the program terminates and so the join() function is called at the end of the program. *Eg :* `t1.join()`.

  NOTE : When using Threading, one thing to keep in mind is that as threads are independent of each other but use or share same resources so we need to share the resource. It can be done by using the keyword "global" in functions assigned to threads to the shared variables between threads. Now as the threads share same variables, problem may arise such that when one thread is executing, the other thread updates the value of the variable. So we need to lock the shared variable during it's execution.

  For more information :
  `https://www.geeksforgeeks.org/multithreading-in-python-set-2-synchronization/`