SDC – TECH REPORT

The student Design Challenge was organised at Asia-Pacific level by ASME at Delhi Technological University, New Delhi. Our team participated in the competition. Here is the Technical Report for the same.

****



**ACKNOWLEDGEMENTS:**

We, DTU-ALTAIR, express our sincere gratitude to Prof. Rajkumar for supporting us and motivating us to take part in the competition. We thank our sponsors for providing us with all the funds we needed to build the bots.

**CONTENTS:**

* About ASME SDC
* About DTU-ALTAIR
* Our preparation and strategy
* 4 Bots
  + Striker – NikhilOne
  + Midfielder – AdityaAlpha
  + Midfielder – VarunBeta
  + Goalkeeper – AkashSir
* Kicking Mechanism
* List of Electronic Components
* CAD Designs
* Problems Faced and Suggested Solutions



**ABOUT ASME SDC:**

<https://www.asme.org/events/competitions/student-design-competition>

The ASME Student Design Competition provides a platform for ASME Student Members to present their solutions to a range of design problems - from everyday household tasks to groundbreaking space exploration. Each team is required to design, construct and operate a prototype meeting the requirements of an annually determined problem statement.

Problem Statement 2018:

<https://www.asme.org/wwwasmeorg/media/ResourceFiles/Events/Competitions/6087-sdc_2018-competition-rules-final-(4).pdf>

**ABOUT DTU-ALTAIR:**

DTU-ALTAIR comprises of hardworking students from various branches of DTU who put in all the effort to overcome any hurdle and approach any problem with a lot of enthusiasm and determination…

**Our Preparation and Strategy:**

The problem statement did not restrict the number of bots as long as they fit in the dimensions of 50cmx50cmx50cm, so we went with making 4 bots – 1 Striker called NikhilOne, 2 Midfielders called AdityaAlpha and VarunBeta and 1 Goalkeeper called AkashSir. These four bots all had about the same design language with the design base being absolutely same with some superficial changes in the goalkeeper to be able to stop the ball better. The Striker also incorporated a functional change of being able to actually kick the ball using a kicking mechanism which was 3D printed especially for the bot.

The design was essentially a triangle with an omniwheel on each of its vertices. The design was in effect, a slight modification of the Holonomic Drive and provided an extra degree of motion to the bot. The third wheel at the vertex of the triangle gave the sideways motion. The structure of the bot was chosen to be of Galvanised Iron Sheets to provide strength to the bot. The parts like the motor clamps, the ball holding gate, the kicking mechanism, etc were all 3D printed to cut down on the weight and obtain the added advantage of complete customisation of the design according to the size of the bot and our requirements.

We chose the bots to be controlled by controllers normally available in the market used for controlling gaming consoles like the Sony PlayStation and Microsoft Xbox. The microcontroller used in the bots was an Arduino Mega 2560. L298N motor drivers were used to control the motors.

The motors were 500 rpm high torque motors, to be able to drive the heavy Omniwheels without failing. The motors were fitted in the 3D printed clamps designed especially for the motors.

All the bots were powered by 11.6V LiPo batteries.

**4 Bots:**

**NikhilOne:**

The NikhilOne was the striker of the team and incorporated the kicking mechanism.

Arduino code:

/\*

ASME E-Fest 2018, DTU

Student Design Challenge

DTU Altair

Robot: One

Driver: Nikhil

\*/

#include <PS2X\_lib.h>

#include <Servo.h>

#define PS2\_DAT 22

#define PS2\_CMD 12

#define PS2\_SEL 11

#define PS2\_CLK 32

/\* Motor driver 1 has motors {enA,in1,in2} and {enC, in5, in6}

Motor driver 2 has motor {enB, in3, in4}

\*/

#define enB 2 //Motor connected on driver 1

#define in3 3

#define in4 4

#define enA 13 //Motor connected on driver 2

#define in1 48

#define in2 50

#define enC 7 // random pins

#define in5 6 //Motor connected on driver 1

#define in6 58

#define enD 8 //random pins

#define in7 10

#define in8 9

Servo myservo;

int RY, RX, LX, LY, mapped;

//#define pressures true

#define pressures false

//#define rumble true

#define rumble false

PS2X ps2x; // create PS2 Controller Class

int error = 0;

byte type = 0;

byte vibrate = 0;

int flag = 0;

void setup(){

pinMode(A4,OUTPUT);

pinMode(A5,OUTPUT);

Serial.begin(57600);

//myservo.attach(24);

delay(300);

error = ps2x.config\_gamepad(PS2\_CLK, PS2\_CMD, PS2\_SEL, PS2\_DAT, pressures, rumble);

if(error == 0){

Serial.print("Found Controller, configured successful ");

Serial.print("pressures = ");

if (pressures)

Serial.println("true ");

else

Serial.println("false");

Serial.print("rumble = ");

if (rumble)

Serial.println("true)");

else

Serial.println("false");

Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press harder;");

Serial.println("holding L1 or R1 will print out the analog stick values.");

Serial.println("Note: Go to www.billporter.info for updates and to report bugs.");

}

else if(error == 1)

Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshooting tips");

else if(error == 2)

Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for troubleshooting tips");

else if(error == 3)

Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

// Serial.print(ps2x.Analog(1), HEX);

type = ps2x.readType();

switch(type) {

case 0:

Serial.print("Unknown Controller type found ");

break;

case 1:

Serial.print("DualShock Controller found ");

break;

case 2:

Serial.print("GuitarHero Controller found ");

break;

case 3:

Serial.print("Wireless Sony DualShock Controller found ");

break;

}

}

void moveForward(){

mapped = 255 - (2\*LY);

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, mapped);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, mapped);

}

void moveBackward(){

mapped = (2\*LY) - 255;

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, mapped);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, mapped);

}

void turnright()

{

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, 255);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

}

void turnleft()

{

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, 255);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

}

void halt()

{

analogWrite(enA, 0);

analogWrite(enB, 0);

analogWrite(enC, 0);

}

void moveleft()

{

mapped = 255 - (2\*LX);

mapped = mapped\*4 /5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

analogWrite(enC, mapped);

}

void moveright()

{

mapped = (2\*LX) - 255;

//mapped = mapped\*4/5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

analogWrite(enC, mapped);

}

void loop() {

if(error == 1) //skip loop if no controller found

return;

else { //DualShock Controller

ps2x.read\_gamepad(false, vibrate); //read controller and set large motor to spin at 'vibrate' speed

LX = ps2x.Analog(PSS\_LX);

LY = ps2x.Analog(PSS\_LY);

if(LY>127)

{

moveBackward();

delay(100);

}

if(LY<127)

{

moveForward();

delay(100);

}

if(LX<127)

{

moveleft();

delay(30);

}

if(LX>127)

{

moveright();

delay(30);

} ;

if(ps2x.Button(PSB\_R1))

{

halt();

}

if(ps2x.ButtonPressed(PSB\_L2)){ //turning left

turnleft();

delay(50);

if(ps2x.ButtonReleased(PSB\_L2))

halt();

}

if(ps2x.ButtonPressed(PSB\_R2)){ //turning right

turnright();

delay(50);

if(ps2x.ButtonReleased(PSB\_R2))

halt();

}

if(ps2x.ButtonPressed(PSB\_TRIANGLE)){

digitalWrite(enD,255);

digitalWrite(in7,HIGH);

digitalWrite(in8,LOW);

delay(10);

}

if(ps2x.ButtonPressed(PSB\_CROSS)){

digitalWrite(enD,0);

delay(10);

}

}

delay(30);

}

Photos:

**AdityaAlpha:**

The AdityaAlpha was the midfielder of the team and is designed to be fast and lightweight. It incorporated a 3D printed gate to hold the ball. The gate was controlled using a servo motor.

Arduino code:

/\*

ASME E-Fest 2018, DTU

Student Design Challenge

DTU Altair

Robot: Alpha

Driver: Aditya

\*/

#include <PS2X\_lib.h>

#include <Servo.h>

#define PS2\_DAT 12

#define PS2\_CMD 13

#define PS2\_SEL 11

#define PS2\_CLK 10

/\* Motor driver 1 has motors {enA,in1,in2} and {enC, in5, in6}

Motor driver 2 has motor {enB, in3, in4}

\*/

#define enA 2 //Motor connected on driver 1

#define in1 52

#define in2 3

#define enB 6 //Motor connected on driver 2

#define in3 5

#define in4 4

#define in5 8 //Motor connected on driver 1

#define in6 9

#define enC 7

Servo myservo;

int RY, RX, LX, LY, mapped;

//#define pressures true

#define pressures false

//#define rumble true

#define rumble false

PS2X ps2x; // create PS2 Controller Class

int error = 0;

byte type = 0;

byte vibrate = 0;

int flag = 0;

void setup(){

Serial.begin(57600);

myservo.attach(24);

delay(300);

error = ps2x.config\_gamepad(PS2\_CLK, PS2\_CMD, PS2\_SEL, PS2\_DAT, pressures, rumble);

if(error == 0){

Serial.print("Found Controller, configured successful ");

Serial.print("pressures = ");

if (pressures)

Serial.println("true ");

else

Serial.println("false");

Serial.print("rumble = ");

if (rumble)

Serial.println("true)");

else

Serial.println("false");

Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press harder;");

Serial.println("holding L1 or R1 will print out the analog stick values.");

Serial.println("Note: Go to www.billporter.info for updates and to report bugs.");

}

else if(error == 1)

Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshooting tips");

else if(error == 2)

Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for troubleshooting tips");

else if(error == 3)

Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

// Serial.print(ps2x.Analog(1), HEX);

type = ps2x.readType();

switch(type) {

case 0:

Serial.print("Unknown Controller type found ");

break;

case 1:

Serial.print("DualShock Controller found ");

break;

case 2:

Serial.print("GuitarHero Controller found ");

break;

case 3:

Serial.print("Wireless Sony DualShock Controller found ");

break;

}

}

void moveForward(){

mapped = 255 - (2\*LY);

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, mapped);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, mapped);

}

void moveBackward(){

mapped = (2\*LY) - 255;

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, mapped);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, mapped);

}

void turnright()

{

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, 255);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

}

void turnleft()

{

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, 255);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

}

void halt()

{

analogWrite(enA, 0);

analogWrite(enB, 0);

analogWrite(enC, 0);

}

void moveleft()

{

mapped = 255 - (2\*LX);

mapped = mapped\*4 /5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

analogWrite(enC, mapped);

}

void moveright()

{

mapped = (2\*LX) - 255;

mapped = mapped\*4/5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

analogWrite(enC, mapped);

}

void loop() {

if(error == 1) //skip loop if no controller found

return;

else { //DualShock Controller

ps2x.read\_gamepad(false, vibrate); //read controller and set large motor to spin at 'vibrate' speed

LX = ps2x.Analog(PSS\_LX);

LY = ps2x.Analog(PSS\_LY);

if(LY>127)

{

moveBackward();

delay(100);

}

if(LY<127)

{

moveForward();

delay(100);

}

if(LX<127)

{

moveleft();

delay(30);

}

if(LX>127)

{

moveright();

delay(30);

} ;

if(ps2x.Button(PSB\_R1))

{

halt();

}

if(ps2x.ButtonPressed(PSB\_L2)){ //turning left

turnleft();

delay(40);

if(ps2x.ButtonReleased(PSB\_L2))

halt();

}

if(ps2x.ButtonPressed(PSB\_R2)){ //turning right

turnright();

delay(40);

if(ps2x.ButtonReleased(PSB\_R2))

halt();

}

if(ps2x.ButtonPressed(PSB\_CROSS)){

myservo.write(90);

delay(10);

}

if(ps2x.ButtonPressed(PSB\_TRIANGLE)){

myservo.write(135);

delay(10);

}

}

delay(30);

}

Photos:

**VarunBeta:**

The VarunBeta was the midfielder of the team and is designed to be fast and lightweight. It incorporated a 3D printed gate to hold the ball. The gate was controlled using a servo motor.

Arduino Code:

/\*

ASME E-Fest 2018, DTU

Student Design Challenge

DTU Altair

Robot: Beta

Driver: Varun

\*/

#include <PS2X\_lib.h>

#include <Servo.h>

#define PS2\_DAT 12

#define PS2\_CMD 13

#define PS2\_SEL 11

#define PS2\_CLK 10

/\* Motor driver 1 has motors {enA,in1,in2} and {enC, in5, in6}

Motor driver 2 has motor {enB, in3, in4}

\*/

#define enA 6 //Motor connected on driver 1

#define in1 4

#define in2 5

#define enB 7 //Motor connected on driver 2

#define in3 8

#define in4 9

#define in5 52 //Motor connected on driver 1

#define in6 3

#define enC 2

Servo myservo;

int RY, RX, LX, LY, mapped;

//#define pressures true

#define pressures false

//#define rumble true

#define rumble false

PS2X ps2x; // create PS2 Controller Class

int error = 0;

byte type = 0;

byte vibrate = 0;

int flag = 0;

void setup(){

Serial.begin(57600);

myservo.attach(24);

myservo.write(100);

delay(300);

error = ps2x.config\_gamepad(PS2\_CLK, PS2\_CMD, PS2\_SEL, PS2\_DAT, pressures, rumble);

if(error == 0){

Serial.print("Found Controller, configured successful ");

Serial.print("pressures = ");

if (pressures)

Serial.println("true ");

else

Serial.println("false");

Serial.print("rumble = ");

if (rumble)

Serial.println("true)");

else

Serial.println("false");

Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press harder;");

Serial.println("holding L1 or R1 will print out the analog stick values.");

Serial.println("Note: Go to www.billporter.info for updates and to report bugs.");

}

else if(error == 1)

Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshooting tips");

else if(error == 2)

Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for troubleshooting tips");

else if(error == 3)

Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

// Serial.print(ps2x.Analog(1), HEX);

type = ps2x.readType();

switch(type) {

case 0:

Serial.print("Unknown Controller type found ");

break;

case 1:

Serial.print("DualShock Controller found ");

break;

case 2:

Serial.print("GuitarHero Controller found ");

break;

case 3:

Serial.print("Wireless Sony DualShock Controller found ");

break;

}

}

void moveForward(){

mapped = 255 - (2\*LY);

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, mapped);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, mapped);

}

void moveBackward(){

mapped = (2\*LY) - 255;

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, mapped);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, mapped);

}

void turnright()

{

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, 255);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

}

void turnleft()

{

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, 255);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

}

void halt()

{

analogWrite(enA, 0);

analogWrite(enB, 0);

analogWrite(enC, 0);

}

void moveleft()

{

mapped = 255 - (2\*RX);

mapped = mapped\*4 /5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

analogWrite(enC, mapped);

}

void moveright()

{

mapped = (2\*RX) - 255;

mapped = mapped\*4/5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

analogWrite(enC, mapped);

}

void loop() {

if(error == 1) //skip loop if no controller found

return;

else { //DualShock Controller

ps2x.read\_gamepad(false, vibrate); //read controller and set large motor to spin at 'vibrate' speed

LX = ps2x.Analog(PSS\_LX);

LY = ps2x.Analog(PSS\_LY);

if(LY>127)

{

moveBackward();

delay(100);

}

if(LY<127)

{

moveForward();

delay(100);

}

if(LX<127)

{

moveleft();

delay(30);

}

if(LX>127)

{

moveright();

delay(30);

} ;

if(ps2x.Button(PSB\_R1))

{

halt();

}

if(ps2x.ButtonPressed(PSB\_L2)){ //turning left

turnleft();

delay(40);

if(ps2x.ButtonReleased(PSB\_L2))

halt();

}

if(ps2x.ButtonPressed(PSB\_R2)){ //turning right

turnright();

delay(40);

if(ps2x.ButtonReleased(PSB\_R2))

halt();

}

if(ps2x.ButtonPressed(PSB\_TRIANGLE)){

myservo.write(135);

delay(10);

}

if(ps2x.ButtonPressed(PSB\_CROSS)){

myservo.write(90);

delay(10);

}

}

delay(30);

}

Photos:

**AkashSir:**

The AkashSir was the goalkeeper of the team and is designed to be large in order to help stop the ball. It incorporated an additional structure to stop the balls along with the gate to hold the ball if and when required.

Arduino Code:

/\*

ASME E-Fest 2018, DTU

Student Design Challenge

DTU Altair

Bot: Goalkeeper

Driver: Akash

\*/

#include <PS2X\_lib.h>

#include <Servo.h>

#define PS2\_DAT 12

#define PS2\_CMD 13

#define PS2\_SEL 11

#define PS2\_CLK 10

/\* Motor driver 1 has motors {enA,in1,in2} and {enC, in5, in6}

Motor driver 2 has motor {enB, in3, in4}

\*/

#define enA 8 //Motor connected on driver 1

#define in1 48

#define in2 46

#define enB 7 //Motor connected on driver 2

#define in3 5

#define in4 6

#define in5 50 //Motor connected on driver 1

#define in6 52

#define enC 9

Servo myservo;

int RY, RX, LX, LY, mapped;

//#define pressures true

#define pressures false

//#define rumble true

#define rumble false

PS2X ps2x; // create PS2 Controller Class

int error = 0;

byte type = 0;

byte vibrate = 0;

int flag = 0;

void setup(){

Serial.begin(57600);

myservo.attach(22);

delay(300);

error = ps2x.config\_gamepad(PS2\_CLK, PS2\_CMD, PS2\_SEL, PS2\_DAT, pressures, rumble);

if(error == 0){

Serial.print("Found Controller, configured successful ");

Serial.print("pressures = ");

if (pressures)

Serial.println("true ");

else

Serial.println("false");

Serial.print("rumble = ");

if (rumble)

Serial.println("true)");

else

Serial.println("false");

Serial.println("Try out all the buttons, X will vibrate the controller, faster as you press harder;");

Serial.println("holding L1 or R1 will print out the analog stick values.");

Serial.println("Note: Go to www.billporter.info for updates and to report bugs.");

}

else if(error == 1)

Serial.println("No controller found, check wiring, see readme.txt to enable debug. visit www.billporter.info for troubleshooting tips");

else if(error == 2)

Serial.println("Controller found but not accepting commands. see readme.txt to enable debug. Visit www.billporter.info for troubleshooting tips");

else if(error == 3)

Serial.println("Controller refusing to enter Pressures mode, may not support it. ");

// Serial.print(ps2x.Analog(1), HEX);

type = ps2x.readType();

switch(type) {

case 0:

Serial.print("Unknown Controller type found ");

break;

case 1:

Serial.print("DualShock Controller found ");

break;

case 2:

Serial.print("GuitarHero Controller found ");

break;

case 3:

Serial.print("Wireless Sony DualShock Controller found ");

break;

}

}

void moveForward(){

mapped = 255 - (2\*LY);

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, mapped);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, mapped);

}

void moveBackward(){

mapped = (2\*LY) - 255;

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, mapped);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, mapped);

}

void turnright()

{

digitalWrite(in1, HIGH);

digitalWrite(in2, LOW);

analogWrite(enA, 255);

digitalWrite(in3, HIGH);

digitalWrite(in4, LOW);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

}

void turnleft()

{

digitalWrite(in1, LOW);

digitalWrite(in2, HIGH);

analogWrite(enA, 255);

digitalWrite(in3, LOW);

digitalWrite(in4, HIGH);

analogWrite(enB, 255);

analogWrite(enC, 255);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

}

void halt()

{

analogWrite(enA, 0);

analogWrite(enB, 0);

analogWrite(enC, 0);

}

void moveleft()

{

mapped = 255 - (2\*LX);

mapped = mapped\*4 /5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, LOW);

digitalWrite(in6, HIGH);

analogWrite(enC, mapped);

}

void moveright()

{

mapped = (2\*LX) - 255;

mapped = mapped\*4/5;

analogWrite(enA, 0);

analogWrite(enB, 0);

digitalWrite(in5, HIGH);

digitalWrite(in6, LOW);

analogWrite(enC, mapped);

}

void loop() {

if(error == 1) //skip loop if no controller found

return;

else { //DualShock Controller

ps2x.read\_gamepad(false, vibrate); //read controller and set large motor to spin at 'vibrate' speed

LX = ps2x.Analog(PSS\_LX);

LY = ps2x.Analog(PSS\_LY);

if(LY>127)

{

moveBackward();

delay(100);

}

if(LY<127)

{

moveForward();

delay(100);

}

if(LX<127)

{

moveleft();

delay(30);

}

if(LX>127)

{

moveright();

delay(30);

} ;

if(ps2x.Button(PSB\_R1))

{

halt();

}

if(ps2x.ButtonPressed(PSB\_L2)){ //turning left

turnleft();

delay(50);

if(ps2x.ButtonReleased(PSB\_L2))

halt();

}

if(ps2x.ButtonPressed(PSB\_R2)){ //turning right

turnright();

delay(50);

if(ps2x.ButtonReleased(PSB\_R2))

halt();

}

if(ps2x.ButtonPressed(PSB\_TRIANGLE)){

myservo.write(120);

delay(10);

}

if(ps2x.ButtonPressed(PSB\_CROSS)){

myservo.write(45);

delay(10);

}

}

delay(30);

}

Photos:

**Kicking Mechanism:**

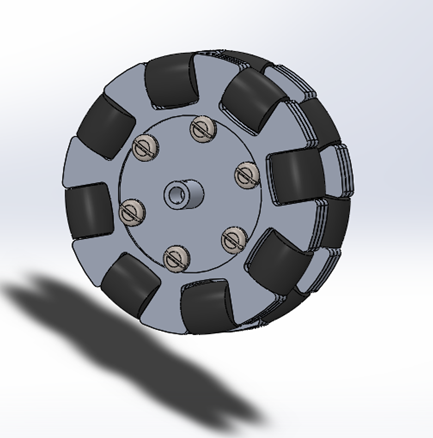
Kicking mechanism was the most important thing for scoring goals. The mechanism was designed using spur gear, rack, spring, high torque gear motor. Spur gear and rack were 3D printed. There was 3D printed rack case which was mounted on the bot. The spring was also placed in it. One end of the spring was attached to the rack and another one was blocked at the end of the rack case. The spur gear was edited with its number of teeth which help the release of rack for kicking the ball. The edited spur gear was mounted on high torque motor and the teeth were incorporated in the rack. While the motor was starting to revolve then the rack was pushed backward by the spur gear, which in turn compressed the spring. After some way of revolve the contact between gear and rack was lost. Then the rack was released forward with a high speed by the potential energy of the spring. This process was repeated for kicking the ball. The kicking mechanism was basically implementation and transformation of rack and pinion mechanism.

**List of Electronic Components:**

|  |  |
| --- | --- |
| Arduino Mega 2650 |  |
| L298N Motor Driver |  |
| SG90 Micro Servo Motor |  |
| PS2 Wireless Receiver |  |

**CAD Designs:**

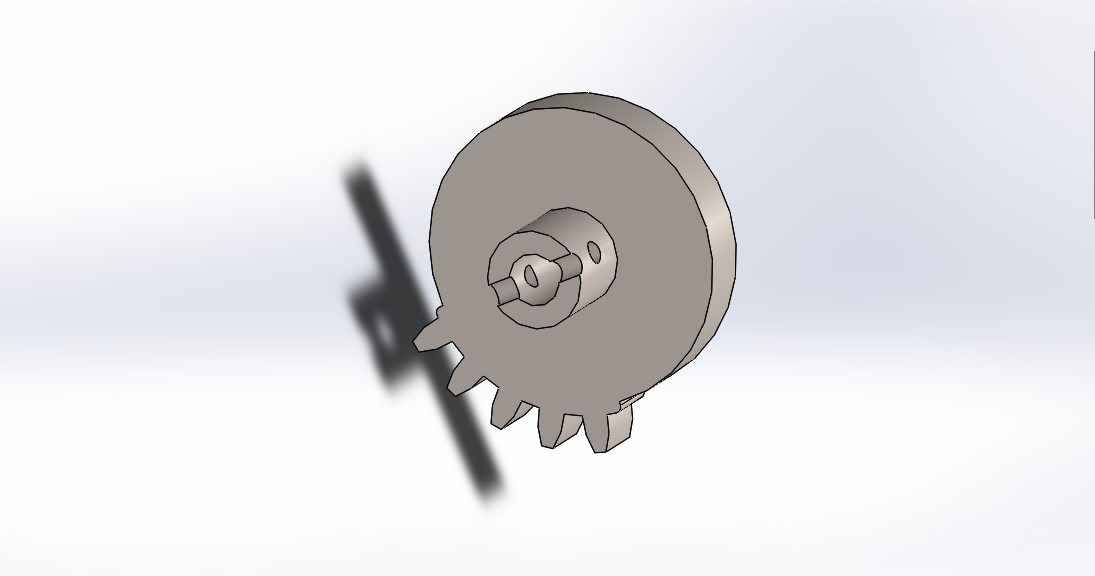
1.Omni Wheel



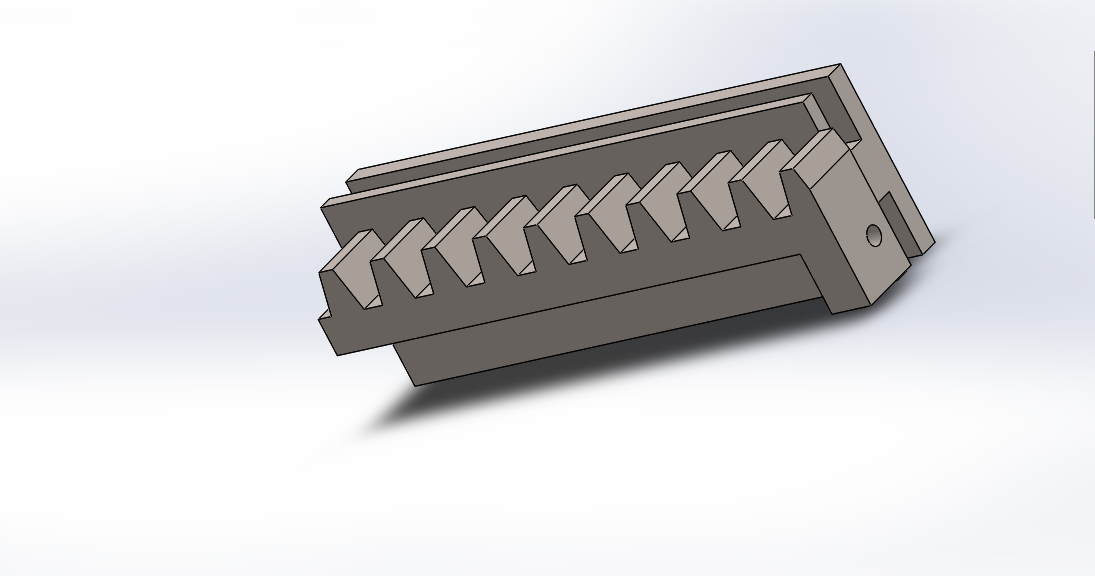
2.Motor Clamp



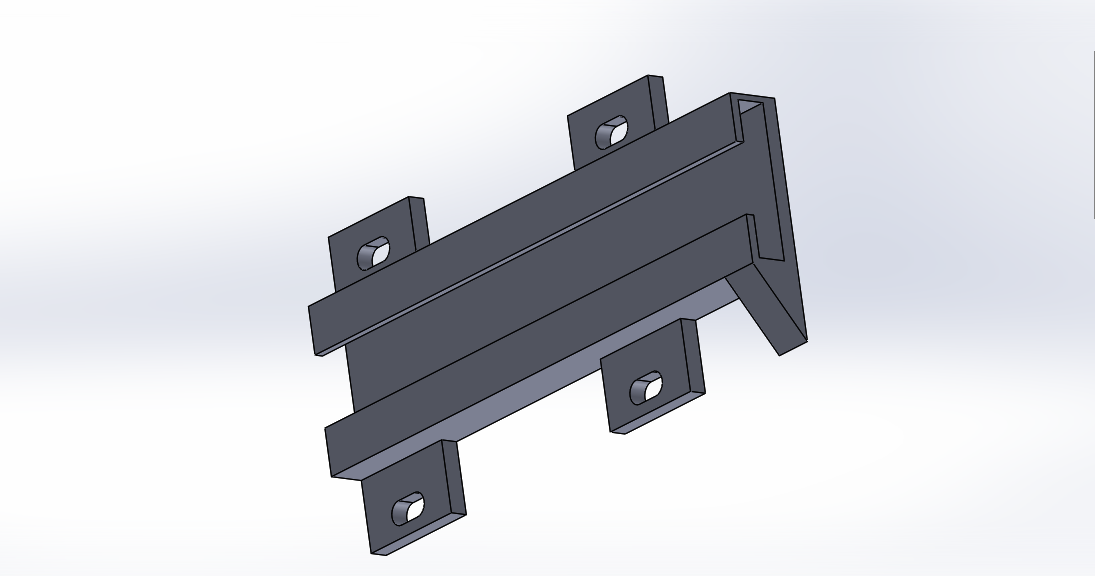
3.Edited spur gear



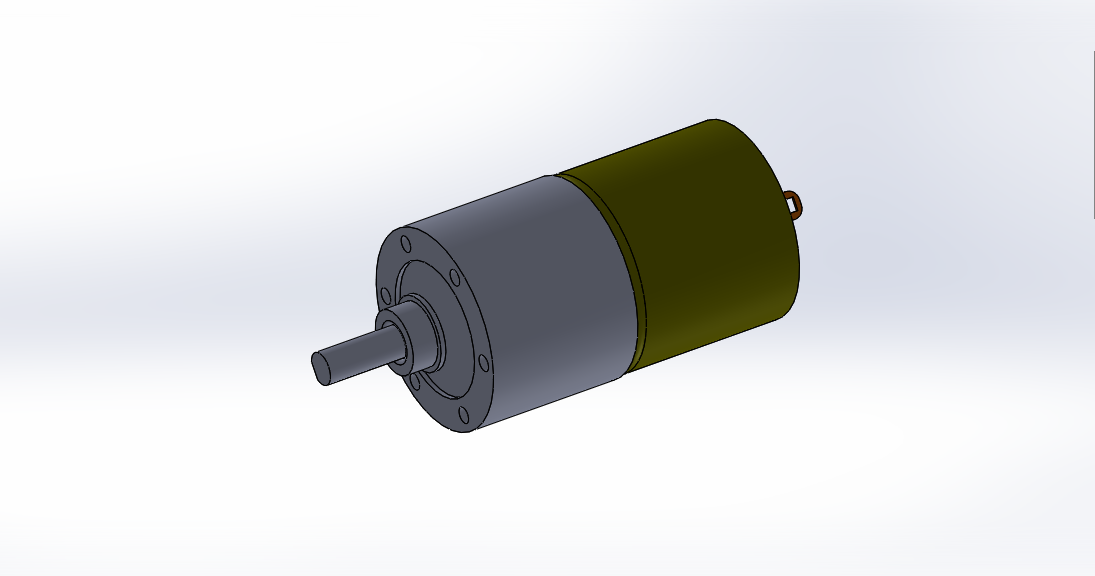
4.Sliding rack



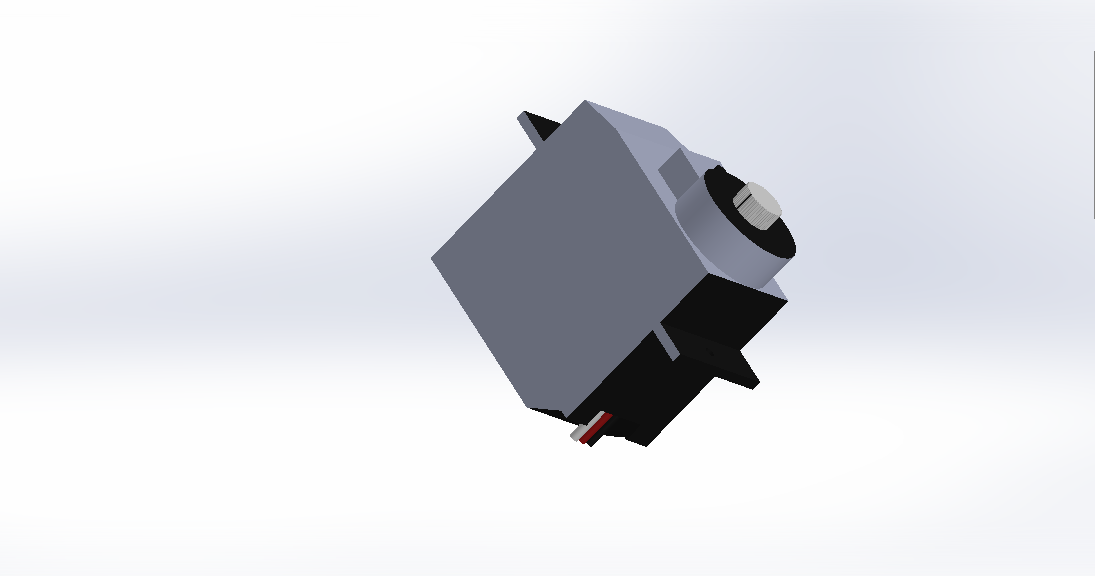
5.Rack slider



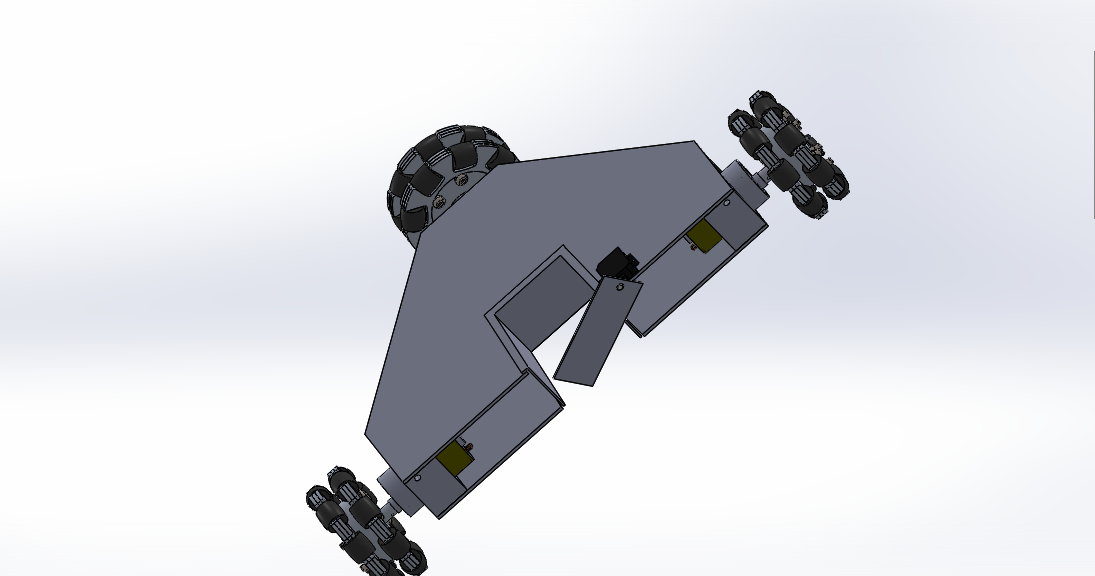
6.DC gear motor



7.Micro servo motor



8.Bot



**Problems Faced and Suggested Solutions:**

1. Extremely long Design Phase:

Explanation: The design phase was extremely long and resulted in lack of time towards the end.

Solution: Proper planning required.

1. Random Electronic Component Failure:

Explanation: We experienced random electronic component failure at the most unexpected times.

Solution: Avoid Chandni Chowk.

1. Damaged bots with very few impacts:

Explanation: During the competition, the bots got damaged with very few impacts from other bots which resulted in the bots being dropped from the competition.

Solution: Design and other mechanical considerations need to be improved to withstand such impacts.

1. Failure of the main striker:

Explanation: The main striker bot with the kicking mechanism failed on the day before the competition and could not be revived.

Probable Reason #1: The motor responsible for executing the kick drew a lot of power which caused the other motors to not function as desired (loading effect).

Proposed Solution: A separate power source for the kicking motor.

Probable Reason #2: Motor driver failure (As explained in Problem #2)

Proposed Solution: Avoid Chandni Chowk.

1. Motor clamp failure:

Explanation: The motor clamps got crushed pretty easily sometimes due to inappropriate printing density of the clamp and the quality of the PLA.

Solution: Proper printing density decided on the basis of stress tests on the printed parts.

1. Loss of contact between the transmitter and receiver:

Explanation: The contact between the transmitter and receiver was sometimes lost during play.

Possible Reason #1: Poor quality controllers and receivers

Proposed Solution: Timely procurement of better controllers.

Possible Reason #2: Faraday Cage effect caused due to iron sheets being used in the design of the body.

Proposed Solution: Better design choices.

**APPENDIX**

**Terminology:**

* **OMNIWHEEL:** Omniwheels are regular wheels with a slight modification that they possess small discs around the circumference which are perpendicular to the turning direction. Consequently, apart from regular rolling motion, the wheel can slide laterally too with great ease.
* **HOLONOMIC DRIVE:** This refers to the ability to move and rotate independently in all directions. It is often used in the realm of robotics.