The Mountain Cargo Challenge

PARTH PAI (22B3305),SUMIT PRAKASH (22B3306),AYUSH BABDE (22B3307),KASHISH TALWAR (22B3308),SANJEET KUMAR (22B3309),DURGAM SANGEETHA (22B3310)

Compiled June 8, 2023

The following presents our report of the bot made by group B03H under the undergraduate 1st year course MS101, IIT Bombay. We have designed and built a bot that climbs a given track by following the line while carrying the required payload and unloading it in the delivary area.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

INTRODUCTION

The given project was a whole new and challenging task for us. None of us had prior experience in this domain. After reading, researching and getting a brief knowledge of how things work, to organise stuffs, we divided ourselves into 3 subsystems viz. Electronics, Coding and Mechanical.

1. ELECTRONICS SUBSYSTEM

The following are the components used for making the bot:

Arduino UNO

Arduino UNO is used as a microcontroller that can be operated using Arduino IDE.

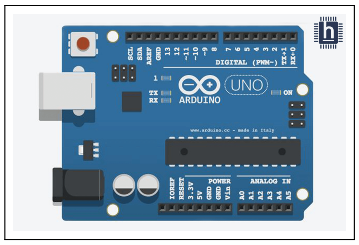

Fig. 1. Arduino UNO

Motors

4 single-shaft 200 RPM battery-operated (BO) DC motors are used which are connected to the respective wheels. $0.1\mu F$ ceramic capacitors are soldered along the terminals of the BO Motors to reduce the noise.

Motor Driver Shield

L293D Motor Driver Shield is used for controlling the motors. It also reduces the hassle of wiring and works with the *Adafruit motor library* which additionally simplifies the code.

Fig. 2. BO DC Motor with wheel.

IR Sensors

3 IR sensors are used for line following by detecting whether the sensor is on white or black surface and accordingly sending infrared rays which reflects and returns back when on white surface but not on black and thus provides information on whether the bot has to turn right or left or move forward or stop depending upon the conditions.
(Explained in more detail in the coding subsystem)

Servo

SG90 servo is used for rotating the payload through the desired angle in order to drop the given payload.
(discussed in detail in the Mechanical subsystem)

Battery

Rechargable Lithium 3-cell battery is used as the power source for the bot.It supplies 10-12V with 1200mAh power.

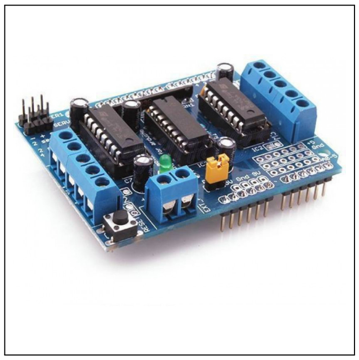

Fig. 3. L293D motor driver shield.

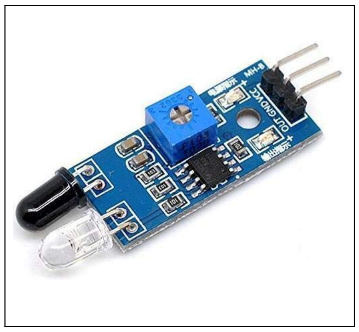

Fig. 4. IR sensor.

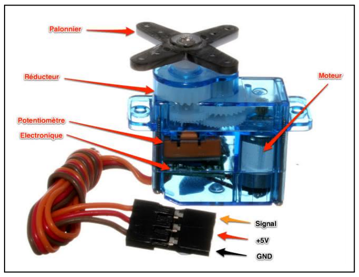

Fig. 5. SG90 servo motor.

2. CODING SUBSYSTEM

```
#include <AFMotor.h>
#include <Servo.h>

unsigned long startMillis=0;//to store the time when all the sensors detect white for the first time

Servo myservo;

AF_DCMotor motor1(1);//Motor 1 connected to M1 pin
AF_DCMotor motor2(2);//Motor 2 connected to M2 pin
AF_DCMotor motor3(3);//Motor 3 connected to M3 pin
AF_DCMotor motor4(4);//Motor 4 connected to M4 pin

int leftSensor=A1;
int rightSensor=A0;
int middleSensor=A3;

bool timer_flag=true; //state flag to trigger the start timer for the first time all sensors detect white and reset when any of them detect black

int threshold= 500;//Setting appropriate threshold

void setup() {
  Serial.begin(9600);

  myservo.attach(10);//Servo attached to pin 10
  myservo.write(175);//Initial angle of servo

  motor1.setSpeed(255);//Initial speed of BO motor
  motor2.setSpeed(255);//Initial speed of BO motor
  motor3.setSpeed(255);//Initial speed of BO motor
  motor4.setSpeed(255);//Initial speed of BO motor
}


void linefollow(int leftValue, int middleValue, int rightValue){
  if(leftValue > threshold && rightValue > threshold && middleValue > threshold){
    //All sensors detects black - rotates to find the line
    motor1.setSpeed(150);
    motor2.setSpeed(150);
    motor3.setSpeed(150);
    motor4.setSpeed(150);
    motor1.run(FORWARD);
    motor2.run(BACKWARD);
    motor3.run(FORWARD);
    motor4.run(BACKWARD);
  }
  else if(leftValue > threshold && rightValue >threshold) {
    //On the line
    motor1.setSpeed(255);
    motor2.setSpeed(255);
    motor3.setSpeed(255);
    motor4.setSpeed(255);
    motor1.run(FORWARD);
    motor2.run(FORWARD);
    motor3.run(FORWARD);
    motor4.run(FORWARD);
  }
  else if(leftValue < threshold && rightValue > threshold) {
    //Moving towards left
    motor1.setSpeed(150);
    motor2.setSpeed(150);
    motor3.setSpeed(150);
    motor4.setSpeed(150);
    motor1.run(BACKWARD);
    motor2.run(FORWARD);
    motor3.run(BACKWARD);
    motor4.run(FORWARD);
  }
  else if(leftValue > threshold && rightValue < threshold) {
    //Moving towards right
    motor1.setSpeed(150);
    motor2.setSpeed(150);
    motor3.setSpeed(150);
    motor4.setSpeed(150);
    motor1.run(FORWARD);
    motor2.run(BACKWARD);
    motor3.run(FORWARD);
    motor4.run(BACKWARD);
  }
}


void payload_drop() {

  myservo.write(115);//Dropping payload
  delay(3000);
  //Coming back for a while
  motor1.setSpeed(255);
  motor2.setSpeed(255);
  motor3.setSpeed(255);
  motor4.setSpeed(255);
  motor1.run(BACKWARD);
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  delay(400);
  myservo.write(175);//Back to initial position of box


  //Rotating by 180 degrees to follow back the line
  motor1.run(FORWARD);
  motor2.run(BACKWARD);
  motor3.run(FORWARD);
  motor4.run(BACKWARD);
  delay(800);
}


void loop() {
  int leftValue=analogRead(leftSensor);
  int rightValue=analogRead(rightSensor);
  int middleValue=analogRead(middleSensor);


  //To check values in Serial monitor and calibrate the
     sensors
  Serial.print("Right: ");
  Serial.print(rightValue);
  Serial.print("  Middle: ");
  Serial.print(middleValue);
  Serial.print("  Left: ");
  Serial.println(leftValue);


  if(leftValue < threshold && middleValue < threshold &&
      rightValue < threshold) {
    //All sensors detect white - STOP command
    motor1.run(RELEASE);
    motor2.run(RELEASE);
    motor3.run(RELEASE);
    motor4.run(RELEASE);

    if(timer_flag==true){
      startMillis = millis();  //to store the time when
     all the sensors detect white for the first time
      timer_flag=false;
    }

    unsigned long currentMillis = millis();  // Get the
     current time

    if(currentMillis-startMillis >= 2000){
      payload_drop(); //drop payload only if all sensors
       continously detect white for more than 2 seconds
    }
  }
  else {
    timer_flag=true;
    linefollow(leftValue, middleValue, rightValue);
  }
  delay(50);//Appropriate belay between loops
}
```

Listing 1. The full code deployed on the bot

Explanation of the software sketch of bot

The bot is equipped with 3 IR sensors to follow a white line marked over dark track. The placement of sensors is done so as one sensor is in the middle so as to stay above the line and other two on either side of the line. The software stack is designed such that the bot moves forward when the middle sensor detects the line while the other two stays on the darker part of the track and follows the track by turning in the direction in which the sensor detects the line.

If all the sensors detect black surface, it means the bot is not over the line so the bot rotates until it finds the line and once the line is detected it starts following it.

The unloading area is marked by a full white zone on the track and thus the bot detects the unloading area if all the sensors detect a white surface and drops the payload by rotating the container through a required angle and giving a jerk ensuring all materials are dropped.

Then the bot rotates and again starts following the line thus reaching the start point of the track.

Turning Mechanism of the bot
The bot uses differential drive to turn i.e if the bot wants to turn in right direction then the left wheels will move forward but the right wheels will move in backward direction, thus making the bot rotate about the centre in the right direction.

Payload dropping mechanism
When bot detects the dropping zone, the servo is commanded to rotate, thus rotating the container attached to the servo, and giving jerk after 3 seconds by moving backward suddenly in high speed for a fraction of second, thus ensuring all the materials are dropped.

Mechanism for finding line
The bot knows it is not on the line if all the three sensors detect a black surface. It then rotates in the right direction until is finds the line and then starts following it.

Mechanism for returning back to the starting point
After dropping the payload, the bot rotates in the **right** direction for a slight duration which makes the bot get out of the line and thus the bot tries to find the line by rotating again in the **right** direction thus getting upon the line in the **opposite** orientation. And thus starts coming back to the initial point following the line.

Mechanism for preventing payload dropping midway on the track
Due to small errors in IR sensor, there may be a chance when the IRs even on being on the darker part of the track, may detect a white surface for a fraction of second. Its possibility is even higher when the bot reaches the starting of a sudden slope of 30 degrees as the IR sensors comes too close to the track. Even tough this incorrect detection lasts for a fraction of seconds, this may cause the payload to drop then and there only, creating a huge loophole in the software sketch of bot.
To tackle this, we added a software timer which counts the time for which all the IR sensors detect a white surface. A timer starts all the IR sensors, for the first time, detects a white surface and is reset the moment any of them detects a black surface. And we modified the code such that if all the 3 IR sensors detect a white surface continously for more than 2 seconds then the payload will drop.
This solved the issue of accidental dropping of payload midway on the track.

3. MECHANICAL SUBSYSTEM

The mechanical designing was a crucial part of the bot. Given it to climb inclines of upto 30 degrees along with a payload of 300 grams, it needs to have its centre of mass well balanced around the centre and lower in height. Also the appropriate placements of wheels, sensors were important for better functionality.

A. Chassis

The chassis is made of acrylic sheet which is a laser cut model. The sketch of it in Autodesk Fusion 360 software was imported to lasercad, modified and uploaded to the laser cut machine. It includes holes at appropriate places to fix the various components on it. It also includes an engraving of the team name.

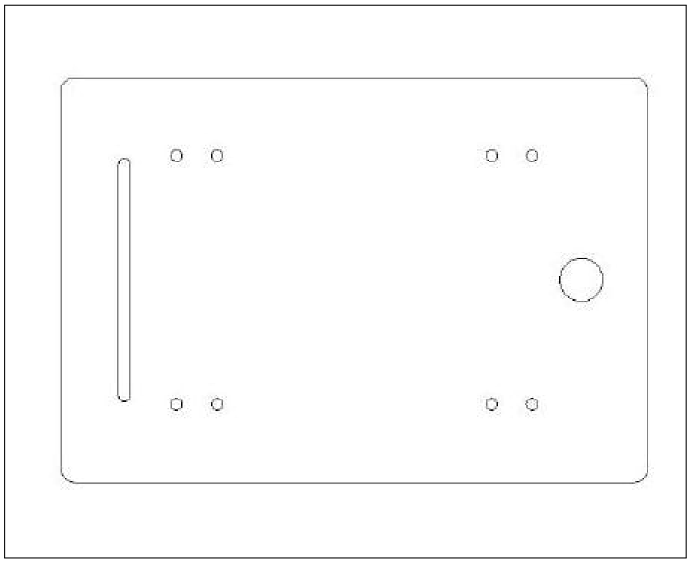

Fig. 6. Chassis design.

B. Payload Container

The payload container is a 3D printed box with a modified inclined facae.The sketch of it in Autodesk Fusion 360 software was imported in Fractory software and arranged in optimum condition in order to minimize weight and time requirements of printing and uploaded to the 3D printing machine.
It is hinged to the chassis of the bot and rotated through a certain angle by a servo to deliver the payload.

The box is designed in such a way that the centre of mass of the payload lies at a distance of around 3 cm from the axis of rotation i.e the hinge, ensuring the sg90 servo (used here) is capable of rotating it along with the payload, as te sg90 servo can produce a max torque of 2.5kg per cm , hence making it even capable of dumping more weight than assigned.

C. Placement of sensors and components

Appropriate placement of sensors is necessary for better functionality of the bot. A long slit was made for IR sensors so as the distance between them can be changed according to the given line thickness.

The battery pack was placed below the chassis so as to optimize space and make a compact bot, and also to move the centre of mass at a lower height

D. Wheel

The bot is a 4 wheel drive one, hence 4 of them are attached to the bot used for its mechanical movement.

SOME STRATEGIES FOLLOWED

We used some strategies to make it successfully do its tasks.

Strategy for climbing inclines without toppling

We designed the chassis appropriately such that the the centre of mass was forward when going up and backward when going down. We achieved this by fixing the Arduino UNO with the motor driver shield at the end (with the connections), battery below the chassis symmetric with respect to the geometric centre and payload mechanism at the front. Since the height of the unloading box was less, the overall centre of mass of the bot was lower further increasing the stability and reducing the chance for toppling.

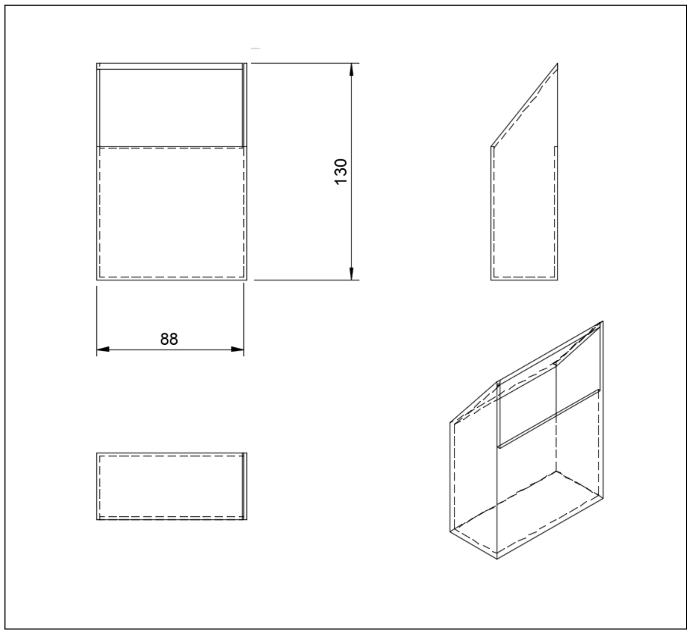

Fig. 7. Schematic view of payload container.

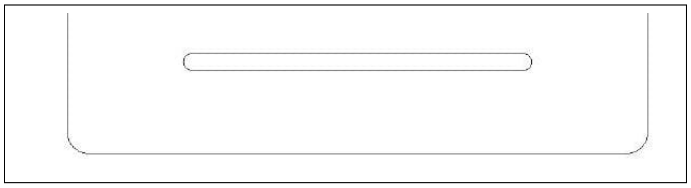

Fig. 8. Slit for IR sensor placement.

Strategy to maximize the power delivered by servo

We alligned the axis of the hinge and the servo motor very carefully to deliver maximum torque and drop the pebbles.

Strategy for precise turns

Smooth turns require an optimum speed. If the speed is more turns may be rough or even the bot may overshoot, but at lower speeds it may be difficult for the bot to climb inclines along with payload. Hence we have given maximum speed to bot when its on straight forward motion and a lesser speed at the turns (as all turns were on plane part of the track).

EXTRA THINGS TRIED BUT UNSUCCESSFUL

We tried many extra things and were even successful in individual testings but weren't able to implement of the final bot

Gyroscope

We used MPU6050 triple axis, 6 *degree of freedom* Gyroscope + Accelerometer module for the vehicle. The main objective was to increase the speed automatically proportional to the slope its climbing. We were able to successfully manipulate speed

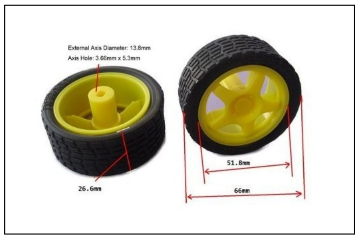

Fig. 9. Wheels used on bot.

with its orientation data. But at the last days it started giving very random and fluctuating data and due to this and lack of time we had to drop this idea. After debugging and testing in isolated environments, we came at the result that the main possible reason might be influence of high vibrations and the Electromagnetic field of the BO motors.

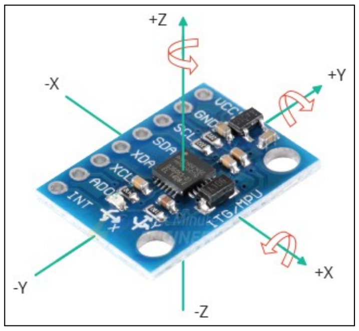

Fig. 10. MPU6050 triple axis - 6 dof gyroscope + accelerometer module.

LED display indicator

We also tried using LED display which indicates the direction of movement and team name. But looking at the connections of the same with the Arduino UNO, we have to drop the idea due to lack of pins available.

Software clock in Arduino code

We decided on using a software timer or s variable which gets incremented again and again. The main objective was to not drop the payload when all three sensors detect white for a slight second. This could happen due white line becoming washed off and some parts of the black sandpaper turning white. Even though we understood the logic behind it, we were again running into some issues with payload drop, etc. So we decided to implement this idea only after the bot works fully and if we have time.

DEBUGGING AND RESOLVING ISSUES

During numerous tests in the track provided, we ran into many small and big issues which we tried debugging for hours.

IR Sensor Calibration

We ran into some issues like the bot successfully climbing the full incline but at the end of the incline it suddenly starts turning around. At first, we thought it is the issue with the grip of the wheels and the bot is slipping, but after numerous trials, we realised that just after when the bot climbed the incline and is on the verge of the plane track, the distance between the track and the sensors becomes large and due to the imperfect calibration of IR sensors, all of them were detecting a black surface and hence the bot thinks it is not on the line and tries to find the line by rotating rightwards. (See the *mechanism for finding line* part on page 3). We fixed this issue by calibrating the sensors and verifying the output using the Serial Monitor in Arduino IDE.

Tyres getting smooth

After many trial runs, the tyres of the bot had worn off and had lost their grip. We fixed this issue by putting rubber bands around the tyres and then heating them appropriately to attach them to the tyre.

Unloading issue

We also ran into the issue of the unloading happening at the beginning irrespective of the sensors not detecting white. We fixed this by doing some changes in the code and freshly writing it down.

EXTRA MENTIONS

Arduino debugging and fixing the motor driver shield

Fixing the Arduino

After multiple testings and continous uploading of codes on Arduino, one of our Arduino stopped responding and was not accepting uploads. We tried to fix it by burning its bootloader by connecting it with another Arduino as instructed in this document Burning the bootloader of Arduino using another Arduino given by Prof. Dinesh K Sharma and were successful in doing so.

Fixing the Motor Driver Shield

During testings our motor shield also stopped working, after debugging we found that the issue is with the motor driver ICs and the 74HC595 shift register IC. We replaced them with new ones and then the motor driver shield started working fine.

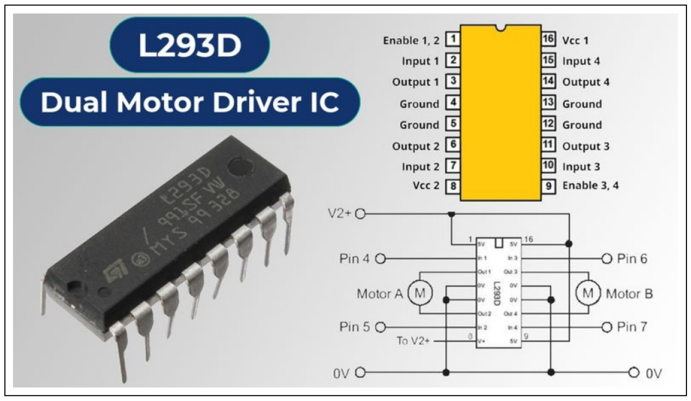

Fig. 11. L293D motor driver IC.

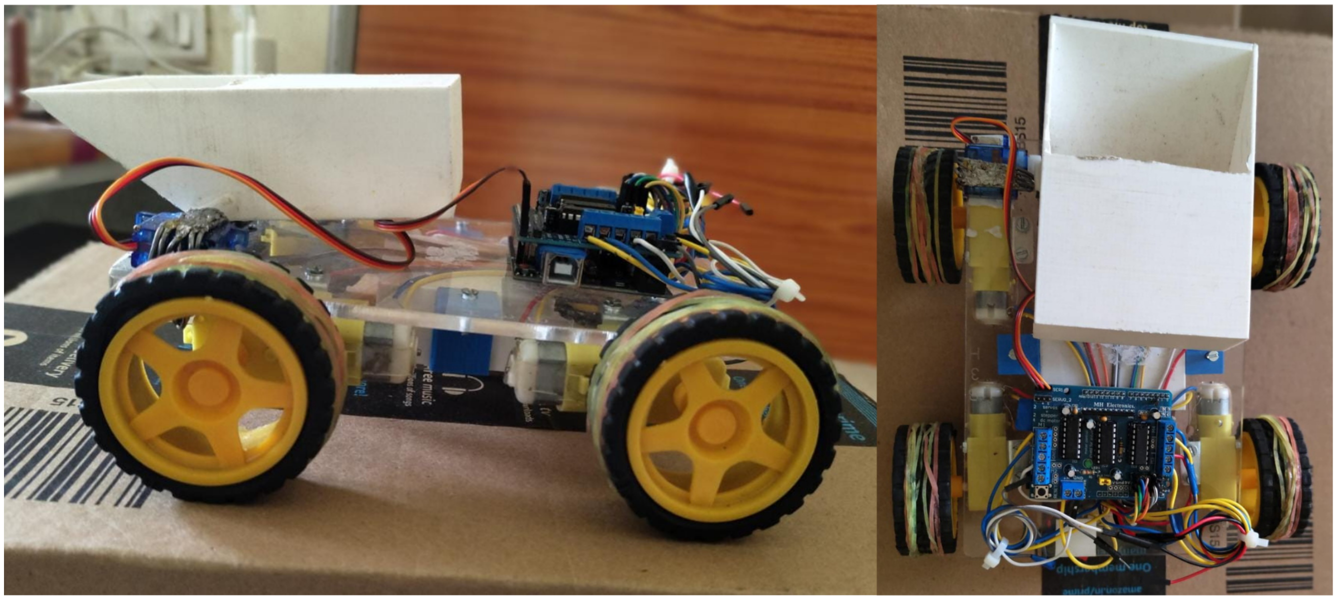

Fig. 12. The final Bot

PROGRESS + DEMONSTRATION VIDEO

- MS Mountain Cargo challenge bot - B03H

CONCEPT DRAWINGS

1. Bot assembly
2. Payload container
3. Battery container clamp
4. Battery container

BILL OF MATERIALS

- Bill of materials

BIBLOGRAPHY

[1] Arduino Guide for MPU-6050 Accelerometer and Gyroscope Sensor

[2] How to Interface Arduino and the MPU 6050 Sensor

[3] How to Make Line Follower Robot

[4] Building an easy Line Follower Robot using Arduino Uno

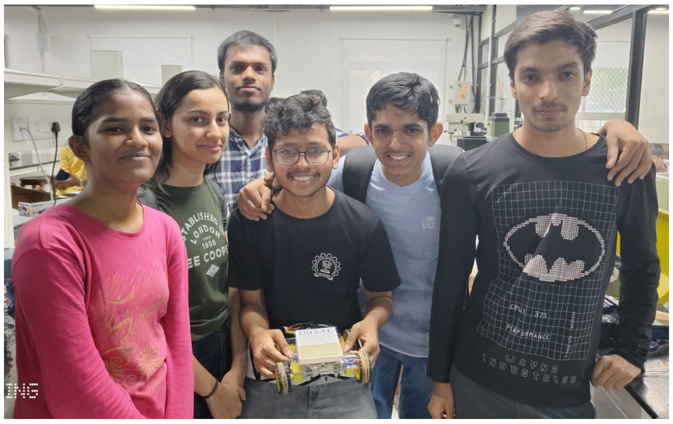

Fig. 13. OUR TEAM
Left to Right: Sangeetha, Kashish, Sanjeet, Sumit, Parth, Ayush