JOINT INSTITUTE
交大密西根学院

UM–SJTU Joint Institute

Design and Manufacturing II
VM350

All-terrain Vehicle Based on Caterpillar Bands and Transformable Wheels

Yixuan Wang, Student ID: 517021910182
Zhenting Wang, Student ID: 517370910118
Yunzheng Yang, Student ID: 516370910179
Yihao Chen, Student ID: 516370910244

08/02/2019

Contents

1 Abstract

Robots have a broad application all over the world. To extend their usage, the transformable wheels are introduced to increase its flexibility and to adapt to different kinds of terrain. However, for some terrain such as sand, traditional transformable wheels are easy to be stuck in. In this study, we manage to build a device that is both flexible and stable, which can complete specific tasks such as climbing and fit the height limitation. We combine the caterpillar and transformable wheels in this study to achieve this objective. We use sensors to control the device autonomously, four motors to drive the device and four servos to control the expansion. We find our device is practical and can be applied under many kinds of complex terrain. However, improvement is also needed, not only in terms of detailed material choice, but also in terms of overall design to improve its ability.

2 Introduction

To make the application of robots more and more broader, many researchers commit into the development of robots with transformable wheels. Compared with the traditional robots, they are more flexible and can fit into different situations. However, how to improve the performance of the new kind of robots is also a new problem.

There are many kinds of transformable robots. For example, the leg-like robot has been developed by Kenjiro TADAKUMA, etc [1]. Another wheel-like robot was developed by Yu She, etc [2]. There are many similar robots with similar structures and shapes. But they all have the problem that they may be stuck in some terrain like sand. Therefore, a caterpillar is needed to adapt to more kinds of terrain. Zirong Luo, etc. have developed a transformable robots similar to be combined with caterpillar [3]. However, it is complicated and deviates a lot from traditional caterpillar. The literature review is listed in Table 1.

No.	Method	Major findings	Unresolved Issues	Refs.
L1	Armadillo-inspired wheel-leg retractable module		Maybe stuck in the sand	[1]
L2	A transformable wheel robot with a passive leg		Maybe stuck in the sand	[2]
L3	Hybrid wheel-track mobile robot		To complicated	[3]

Table 1: Literature Review

In our study, we develop a device combining the caterpillar and active-expanding transformable wheels using double-layer structure. Four DC motors and four servos are used to drive and control the expansion of the device. Sensors are used to detect the distance and the gesture of the device to control the behavior of it.

3 Design

3.1 Creativity

In this device, we combined caterpillar bands and deformable wheels to realized the functions of climbing steps and running on sand. We designed a double-layer structure for the wheels to combined chain wheel and the deformable linkage together, which is shown in figure 1. The driving plate of the deformable linkage is connected to a servo through a gear train with a transmission ratio of -1. The servo is used to control the open and close of the deformable wheel actively at any condition, which makes our device more adaptable to various terrains.

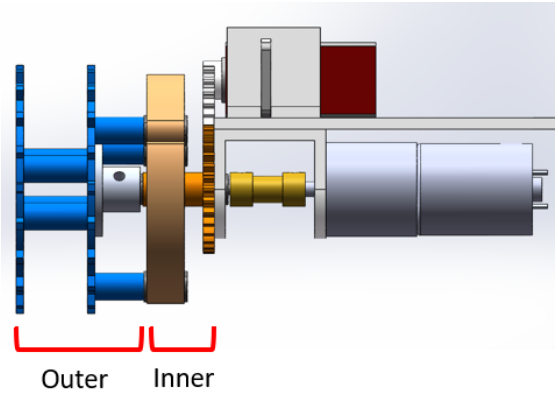


Figure 1: Schematic of double-layer structure

For the linkage structure, we modified the conventional four-bar linkage by replacing one rotational full-joint and a binary link with a sliding half-joint, and the dimensions of the linkage is specially designed to achieve a self-locking function. The method is rotate the driving plate slightly over the toggle position in term of the rocker, i.e., make the extreme angle between the output rocker and the driving rocker slightly larger than 90 degrees. As a result, the load applying of the tip of the output rocker will have a tendency of opening, which will be restrained by the geometrical design and hold a stable posture (figure 2).

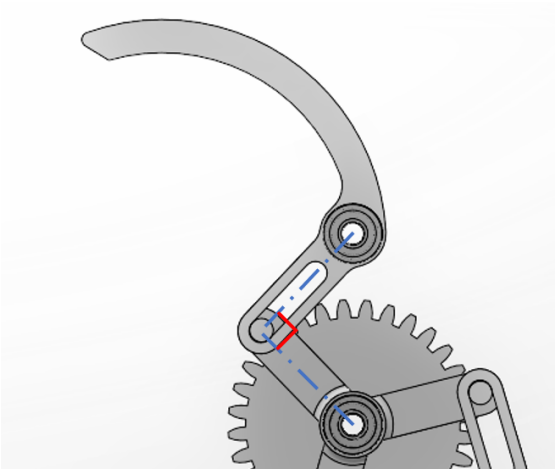


Figure 2: Schematic of self-locking

Our device also applies the design concept of modularization. The assembly of circuit and the assembly of mechanical structure are conducted individually before the final assembly. As a result, the whole circuit components are placed between two acrylic board, which greatly avoid the exposure to the flying sand. All the four wheel module can be uninstalled from the device easily by just unscrewing two screws, which greatly reduces the difficulty of maintenance.

3.2 Graphical Linkage Synthesis

Since we modifies the four-bar linkage to fit our condition, the movement of the linkage is restrain by the geometric characteristic. The two extreme position are shown in figure 3. The designed output range is shown in figure 4

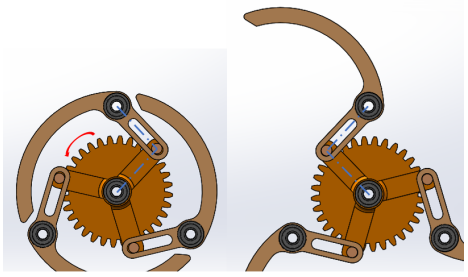


Figure 3: Extreme positions

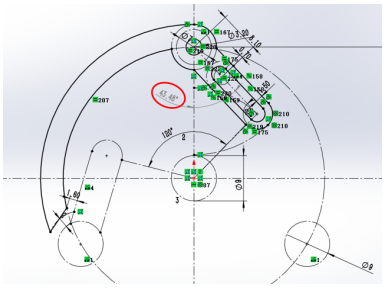


Figure 4: Range of output rocker

The designed dimension listed below in Table 2. The dimensions of the linkage yield a moving range of 87 degree. The two position are symmetric about the connection line of the two fixed points. The radius of the deformable wheel under closed condition is 30.5 mm, when the output rocker is completely open, the distance between the axle and the farthest edge is 64.5 mm.

Measurand	length [mm]
The distance between spindles	26
Driving Rod	17.9
Slider	7.4 - 18.3

Table 2: Dimensions of the linkage

The whole CAD schematic of the device is shown below, the caterpillar bands and the circuit plate and other circuit components are not displayed. The real picture of the device is shown in figure 6.

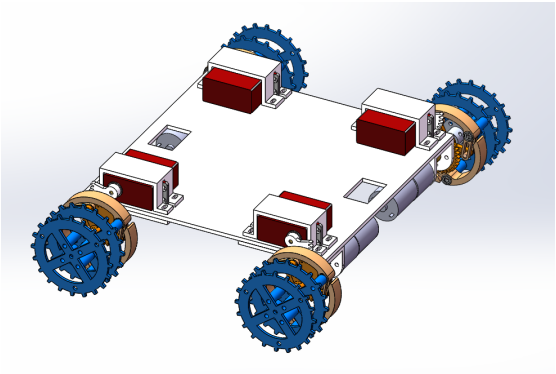


Figure 5: Schematic of the whole vehicle

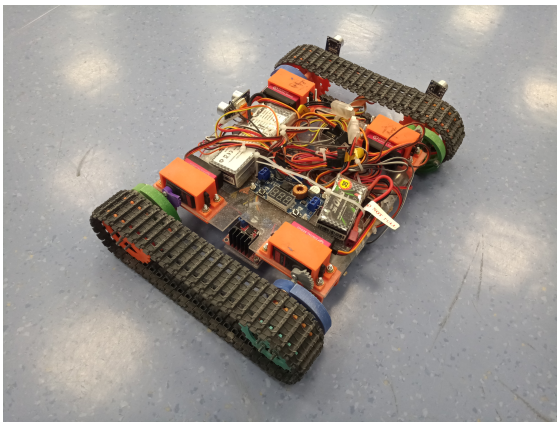


Figure 6: Picture of the real vehicle

4 Manufacturing

4.1 Selection of Materials

The device uses components made of various materials. The double chassis layers are made of acrylic board with a thickness of 3 mm. We use totally four motors and four servos on the device. The motor type is 25GA370 with 60 rpm, which can generated a locked-rotor torque of 8 kg·cm. For the servo, we use 180 degree servos that can generate a torque of 20 kg·cm. We use 3D-printing technology and PLA as material to produce small components that have complicate outlines, such as the servo and bracket holders, the servo gears, the driving plate and the output rockers. For the wheel axle, we choose four M3*40 mm carbon steel axles, the deformation of carbon steel rod is very small under loading. The axle connectors are brought, the material of which is brass. The flange that used to connect the wheel and the axle are also brought, and the material is aluminum alloy. In order to make the rotation of the wheel smoothly, we use three ball bearings on the axle for each wheel assembly, and another six bearing on the linkages for each wheel. The type of the bearings is NSK F683ZZ with flange.

4.2 Procedure of Manufacturing

1. Cut the acrylic board using laser cutting.

2. Build the motor brackets, servo brackets, driving plates, servo gears, output rockers and the wheels by 3D printers

3. Purchase the Arduino board(Mega), voltage-reduce module, motor control module(L298N), ultrasonic sensors(HC-SR04), the caterpillar bands, , the batteries(2200mAh), the motors(25GA370) and the servos(DS3218).

4.3 Procedure of Assembly

The assembly of the device can be separated into the mechanical part and the electrical part.

4.3.1 Assembly of the mechanical part

1. Install the servo and the motor on the chassis. Also, install the wheel axle and the motor with a connector. Fixed an bearing through the axle on the motor bracket, and keep the flange side at the inner side of the bracket.

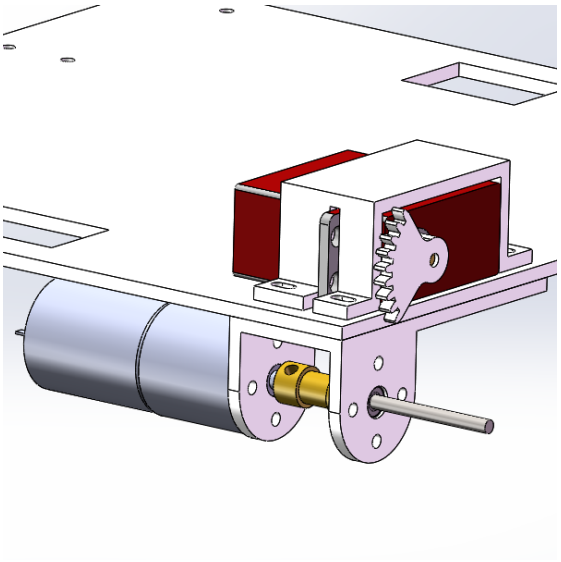
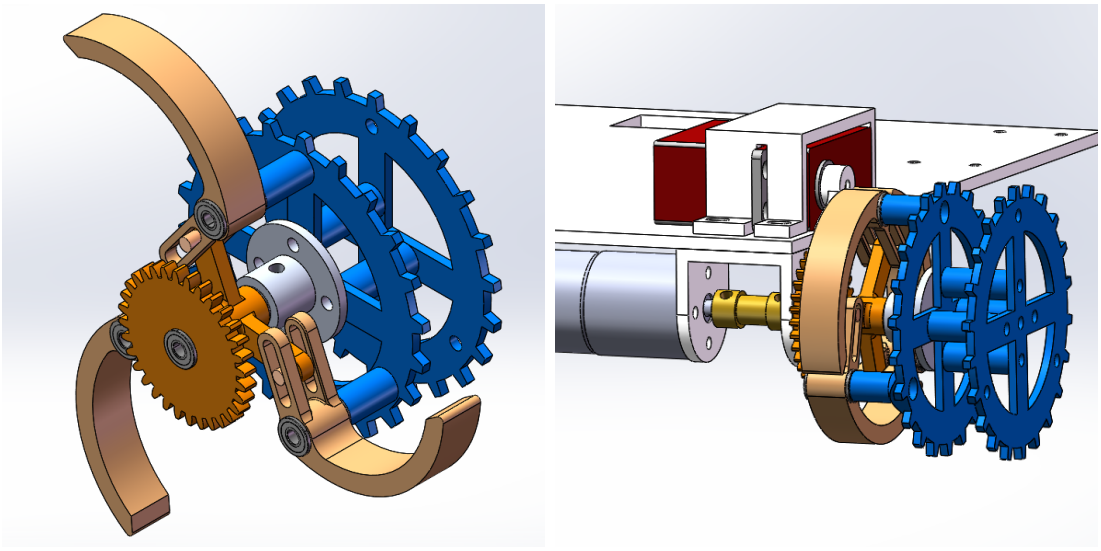


Figure 7: Install the servo and the motor.

2. Install the wheel assembly individually first, and then install it on the axle. For the wheel assembly, install the outer layer and the inner layer separately first and then put the together. Note that there is a bearing between the driver plate and the flange.



(a) (b)

Figure 8: Install the wheel assembly.

3. Repeat the previous for the other three wheels. The caterpillar band should be installed together with the two wheels on the same side. Moderate the caterpillar bands to an appropriate length before assembly.

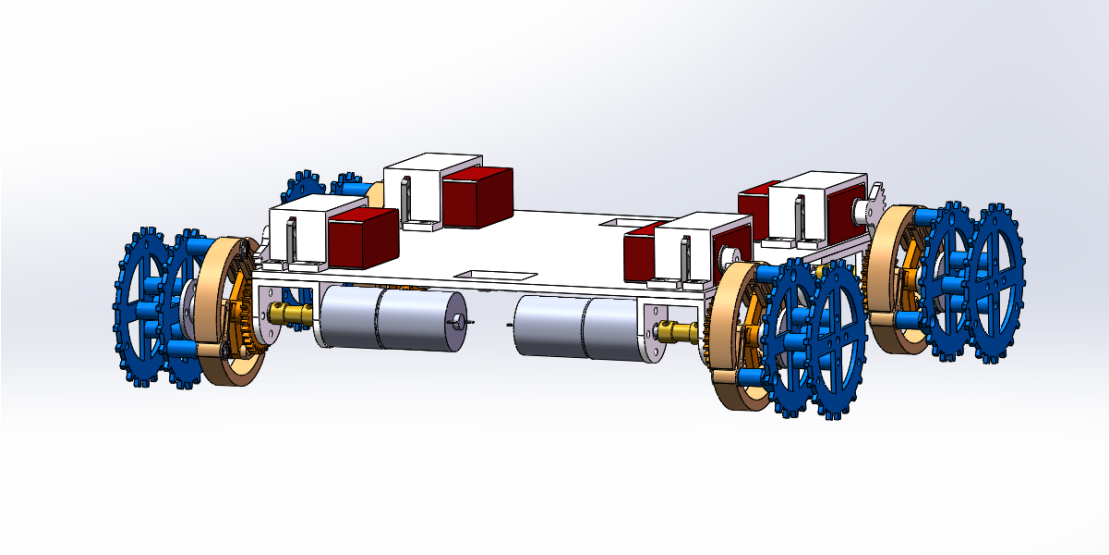


Figure 9: Install the other wheels.

4. Install the bracket for the ultrasonic sensors on the right side of the device.

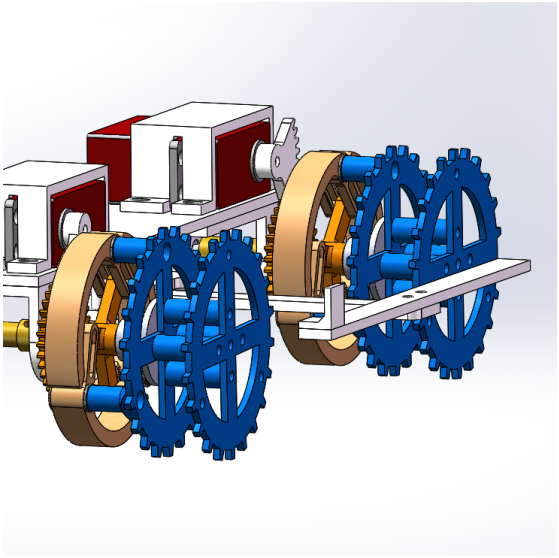


Figure 10: Install the bracket for ultrasonic sensors.

4.3.2 Assembly of the electronic part

1. Install the circuit components on the acrylic board of the lower chassis layer.
2. Connected the wires.
3. Fix the circuit part and the mechanical part together.

5 Algorithm

The flow chart of the algorithm is shown in Figure 11. There are a few noticeable points. First, we built a library using class in C++, which avoid copying and pasting code for many times and flexible to develop and modify. In addition, we divided the process of the movement into five stages and label it in the program to identify specific stages.

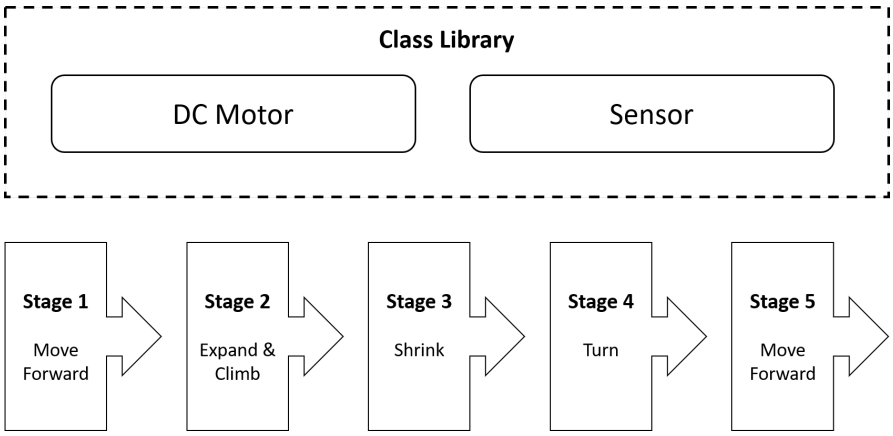


Figure 11: Flowchart of the Algorithm

Another point to note is that we created a negative feedback in the program requiring the device to move in a straight line. The circuit is shown in Figure 12, which can be used as a reference to rebuild the device.

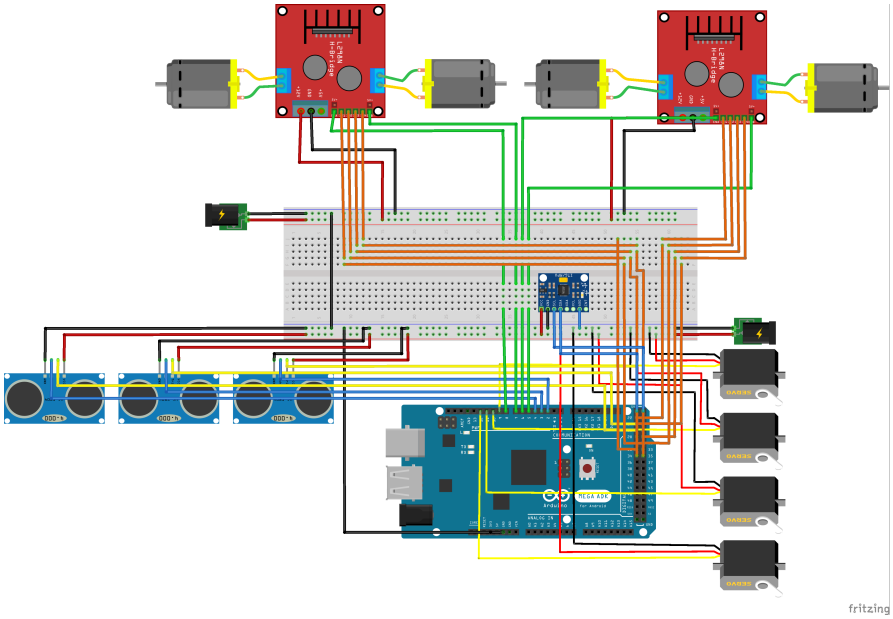


Figure 12: Circuit

6 Analysis

6.1 Classification of Linkage

The linkage of the transformable is shown in Figure 13. This linkage has one degree of freedom. But this linkage is not a four bar linkage, which is designed by transforming the traditional four bar linkage. Therefore, we cannot analyze it using Grashof condition. However, we can come into the conclusion that both of them cannot fully rotate. Because $l_1 = 17.9mm$, $7.4 \leq l_2 \leq 18.3mm$ and $p = 26mm$. These three bars can form a triangle at the extreme position, which means that they cannot rotate fully.

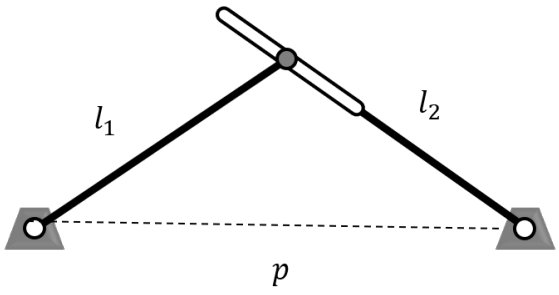


Figure 13: Linkage

6.2 Position Analysis

The position we analyze is the farthest point of the linkage after the expansion. We choose this point because it is the point that will contact the ground when expanding. The position analysis video is attached in the folder, with file name "*simulation*". The analysis result is shown in Figure 14. As the figure indicates, the acceleration is the largest around the middle point of the expansion. Therefore, the failure of expansion may appear around the middle point.

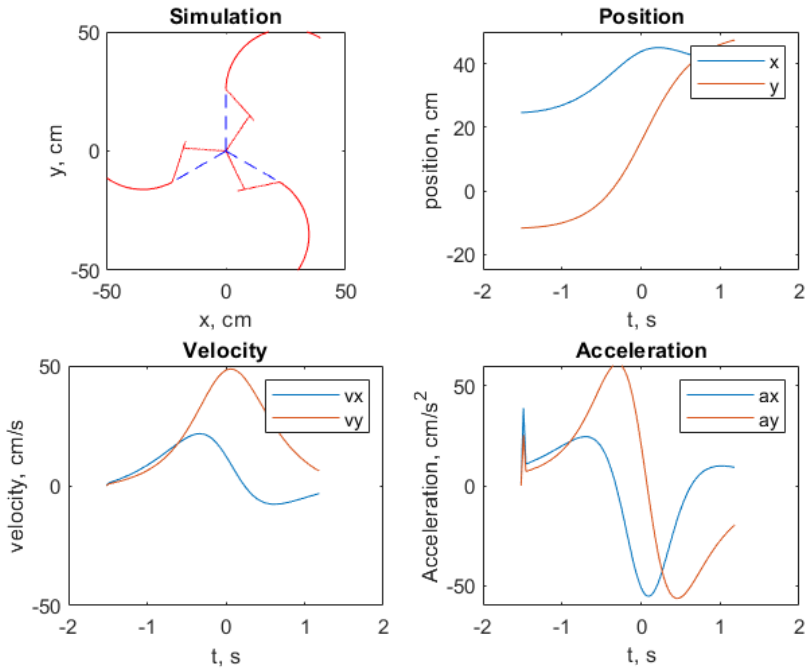


Figure 14: Position Analysis

6.3 Force Analysis

The process we care is the climbing process, which is very fragile and easy to fail. Therefore, we only analyze the process of climbing here. Figure 15 shows a general scheme while climbing. It is noticeable that the actual gesture of legs can be complex. For example, there may be a phase difference between four legs, which results in an uneven gesture of the device. Therefore, it is very hard to conduct an absolutely accurate analysis.

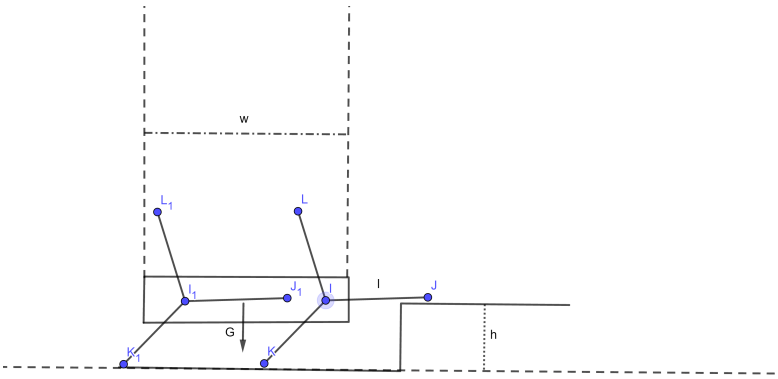


Figure 15: Overall Scheme while Climbing

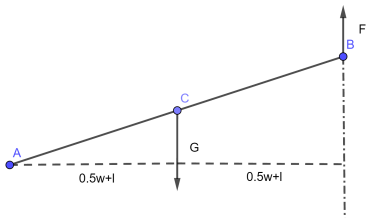


Figure 16: Force Analysis while Climbing

Instead, we simplify the model as one straight rod, as shown in Figure 16, with only the front legs responsible of climbing. Therefore the force needed is half of the weight. With two fore legs, one leg should provide force about a quarter of weight. According to our measurement, the weight is about 1.84kg. The position force exerting on is about 6cm away from the center. Therefore, the torque needed is about $3\text{kg}\cdot\text{cm}$. The DC motor's stalling torque we use is about $8\text{kg}\cdot\text{cm}$, which is much larger than this very conservative estimation. Therefore, our design is very safe. Therefore, the maximal expected loading capacity is about

$$m_{loading} = \frac{8}{3} \times 1.84[\text{kg}] = 4.88[\text{kg}].$$

If we take the variament of the direction of the force, we may get a conservative estimation of the expected loading capacity based on the following equation and lever principle.

$$\frac{T_{motor}}{r_{rocker}} \cdot L\cos(\alpha) = mg\frac{L + r_{rocker}}{2}\cos(\theta);$$

Since the front motors are responsible for climbing, and the angle α and θ vary all the time. Roughly speaking, $\theta \approx \arccos(\frac{h}{L})$, where h is the height of the step, and L is the distance between the front axle and the rear axle. And since we use three centrosymmetric output rockers, so $|\alpha|$ will never be larger than 60 degree. Hence the rough estimation gives total mass $= \frac{1.48 \cdot 16[\text{kg} \cdot \text{cm}] \cdot \cos(17^\circ)}{4[\text{cm}] \cdot \cos(60^\circ)} \approx 3.11[\text{kg}]$.

6.4 Rolling Speed Analysis

6.4.1 Expansion vs. Shrink

We assume the rotation speed under two modes are the same. Therefore, the rolling speed difference only comes from the difference of the radius. The rotation speed of the motor is $60rpm$. While shrinking, the device's wheel has the radius of $26mm$. Therefore, the moving speed is

$$v_{shrink} = \omega \times r_{shrink} = \frac{2 \times \pi \times 0.026 \times 60}{60} m/s = 0.16 m/s$$

Similarly, we can get the speed while expanding, when the radius is about $64.5mm$.

$$v_{expand} = \omega \times r_{expand} = \frac{2 \times \pi \times 0.0645 \times 60}{60} m/s = 0.41 m/s$$

6.5 Smooth Ground vs. Sand

Because we use the caterpillar to drive the device, the smoothness of the contact surface has small influence on the moving speed while shrinking. Therefore, we expect a similar result between them. However, while expanding, the device on the sand will have similar speed as the device on the sand while shrinking, because the expanded legs are stuck in the sand and the caterpillar rolls on the sand. The theoretical speed is summarized in Table 3.

v, m/s	smooth ground	sand
expand	0.41	0.16
shrink	0.16	0.16

Table 3: Summarized Theoretical Speed

7 Experiment

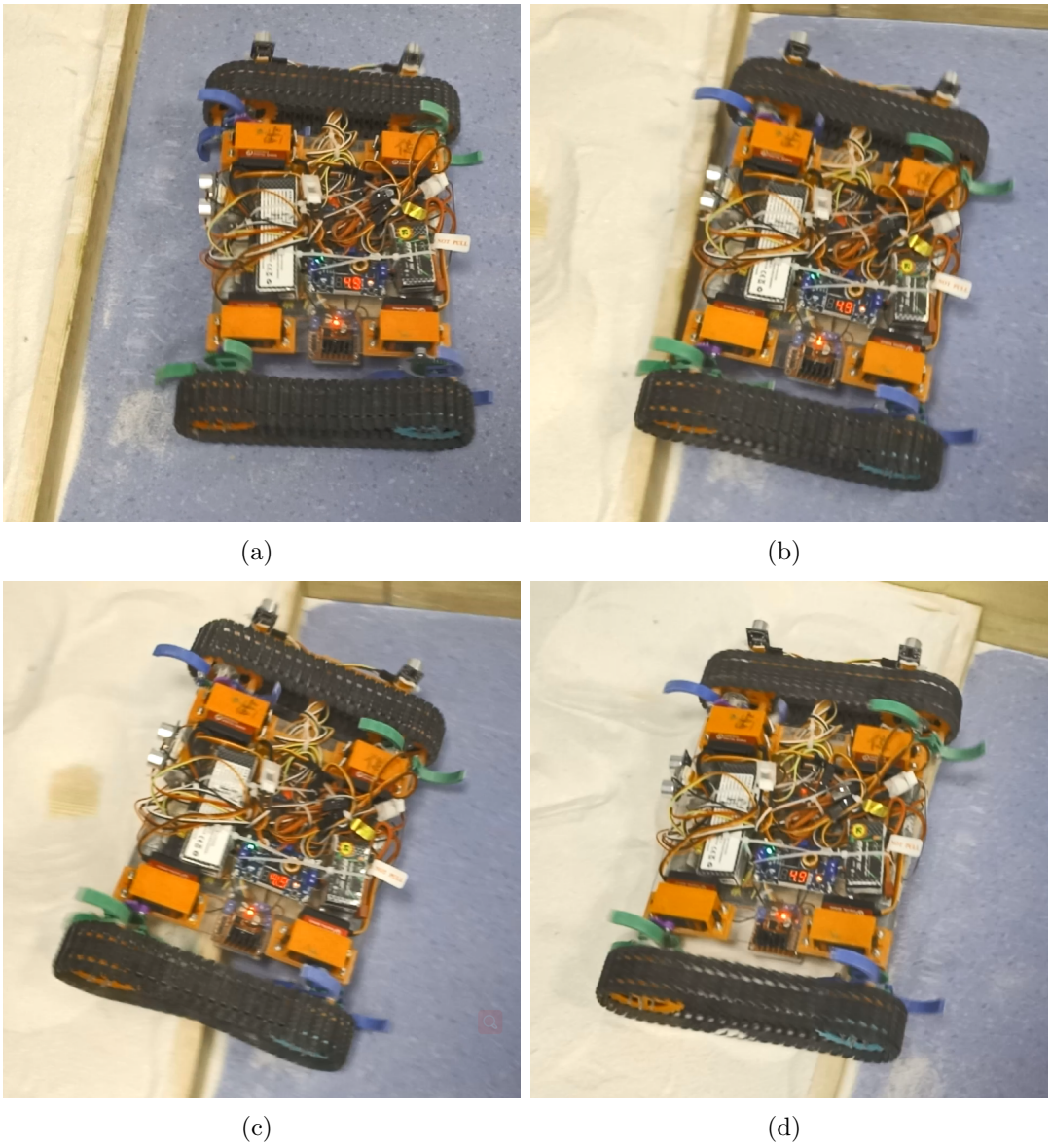
7.1 Load Carrying Capacity

The total mass of our prototype is 1840 grams. We tested its load carrying ability by adding a 1-kg weight on it. We observed that the body of the device suffering a great amount of deformation, and it succeeded in carrying the 1-kg weight onto the sand box. However, although the first attempt was successful, it cause some cracks on the weak points on PLA structures that stop our further attempts. The load carrying ability of the prototype is restrained by the materials or the structure we designed, not the torque of the motor. We estimate that the theoretical load carrying capacity may reach 1500 grams after we fix the design flaw of the wheel by redesign the structure or use resin as the 3D-printing materials. This result is within the range of our estimation.

7.2 Demonstration of Climbing

Figure 17 shows how our device climbs up the sand box. The output rockers are responsible for grasping the edge of the sand box. The strong grip of the caterpillar bands is able to push the device onto the step as long as one of the rocker grapples the edge. Since we have a deformable structure on each front wheel, the adaptation of our device to the unpredictable situation in the reality is greatly improve.

The process of climbing is very quick. The device takes less than 3 seconds from opening the deformable wheels to being completely on the sand box.



(a) (b)

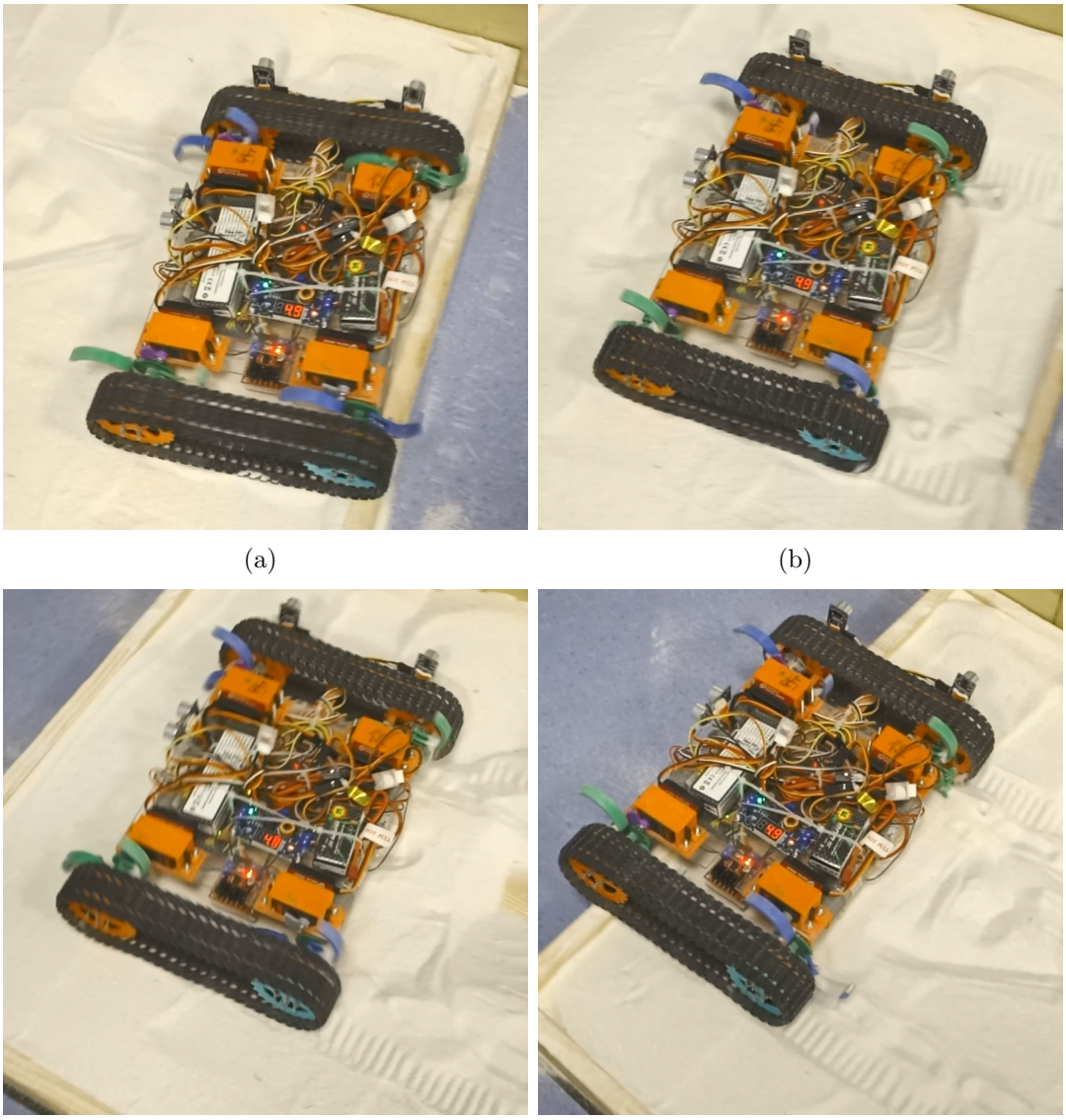
(c) (d)

Figure 17: Climbing sand box.

7.3 Demonstration of Rolling On Sand

Figure 18 shows how the device travels on the sand. Since we apply the caterpillar bands, the device covers a distance over 40 cm in 3 seconds easily

without sinking into the sand. The average speed of the device on the sand is over 13 cm/s. The theoretical speed of the device on the sand is 16 cm/s. The actual speed is lower than expected, this is because the output speed of the motor is lower than its fixed value, 60 rpm, when it is loaded.



(a) (b)

(c) (d)

Figure 18: Rolling on sand.

7.4 Demonstration of Rolling On Smooth Surface

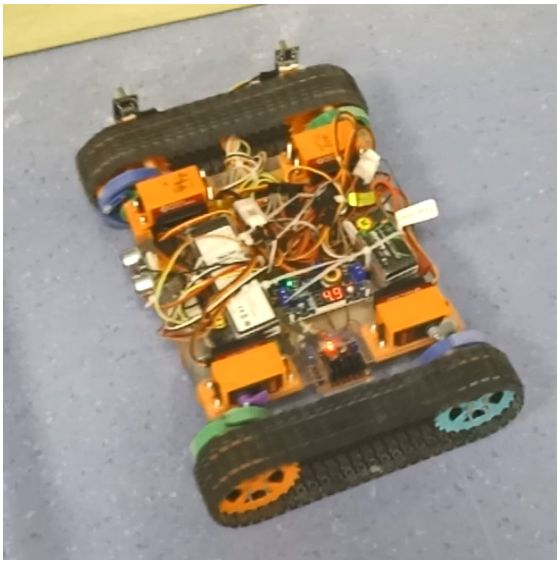


Figure 19: Rolling on smooth surface.

Figure 19 displays the situation of moving on smooth surface. As we do not make the motor produce its maximum power, the speed of the device remains nearly the same as that on sand, which is about 15 cm/s. The speed on smooth surface is slightly higher, which is probably because the resistance force on the smooth surface is smaller than that on sand. And the result is more closer to the theoretical value calculated above, which means the motor is rotates at an angular speed close to 60 rpm.

7.5 Transformability

The extreme angles of expansion and shrinkage in reality are 43 degrees, which are exactly the same as designed. However, the sliding joint was not so smooth as we expect because of the quality of the 3D-printer. Before assembly, we need to polish the surface of the sliding joint to make the linkage usable.

In our design, the moving range of the driving plate needed to make the full expansion of the output is 89 degrees. However, our experiment shows that the servo gear with this dimension cannot 100 percent ensure a successful transmission. The real dimension we need depend on the module of the gear. In our design, both gears have a pitch circle with a radius of 16 mm, and the module is 1 mm. The result of our experiment shows that a fan-shaped gear with 9 teeth fit our requirement best. and the central angle of the gear is 97 degree. The CAD of the servo gear is shown below in figure 20

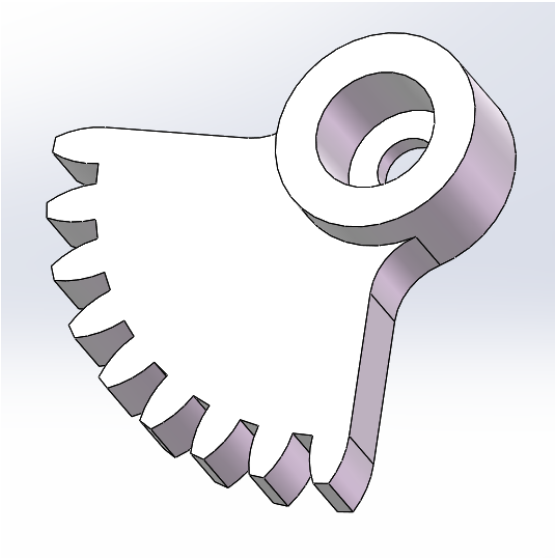


Figure 20: CAD of the servo gear

8 Discussion

Because there is no previous published work focusing on theoretical research of this kind of device which combines transformable wheels and caterpillar, we cannot make compare and contrast between our work and previous work. Therefore, agreement and disagreement with previously published work is ignored.

8.1 Creativity

There are many creative points in our design.

- Combination of caterpillar and transformable wheels

  We observed that the sand is very smooth and easy to slip without proper design. To avoid this kind of problem, we combine the caterpillar with the transformable wheels, which combine the stability and the flexibility. Moreover, the meaning of this kind of design is that it can extended to real life and adapt to a broader kind of terrain, rather than competing in a course project.

- Two layers of wheels enabling the movement and the expansion

  It is challenging to have a structure that can control movement and expansion independently. To achieve this goal, we design two layers of wheels and control the movement and expansion by motor and servo independently.

- Flexible program development method

  Instead of traditional Arduino programming method by repeating the command, we create classes using interdisciplinary knowledge from the computer science, which enable us to modify the program easily.

- Function adjusting device's trajectory along the straight line

Because there are many noises in the physical world, we cannot expect the device to move along a straight line strictly. Therefore, the function that can respond to noises is necessary. We achieve this function by reading the output from the side sensors to know the gesture and adjusting it accordingly.

- Modular electric components

  We integrate all of electric components in a small board compactly to save the space. In addition, it is flexible to develop the program and modify it during the process of the development.

8.2 Impact

Our device have some broader impact. First, we propose a new kind of design combining stability and flexibility, which can be applied under a broader situations.

In addition, we find that this kind of design may encounter some limitation on the expansion, which means that the design may be better improved. The limitation is caused by the requirement that the diameter of transformable wheels is required to be smaller that caterpillar while shrinking, which limits the largest radius it can expand. To solve this problem, we should come up with a new design.

This kind of device can be used under many complex situations. For example, when the device tries to reach specific position to save people after the disaster such as earthquake, it is very possible that the device will encounter high obstacles on the way. Our design can solve such problem. Above is just one example of complex situations a device may encounter. There are also many other fields this kind of design can be applied such as military.

8.3 Phenomenon Out of Expectations

Previously we make many analysis, but we find many phenomenon out of our expectations.

- The assembly between keyway on the servo and 3D printing part is unreliable and easy to break or slip.
- The teeth number of the gear connected to the servo to control the expansion is much larger than theoretical calculation result due to pressure on the legs.
- The strength of acrylic board and legs are expected to be larger to avoid large deflection.

9 Conclusion

In this project, we are expected to develop a device that can climb, walk on the sand, turn autonomously and pass a region with height limitation. We achieve these objectives by building a device with caterpillar and transformable wheels. Four DC motors are used to drive the whole device, and four servos are used to control transformable wheels. Three supersonic sensors are used to detect the distance and the gesture of the device to control the behavior of the device.

For this specific project and device, we find:

- Our motor is strong enough to expand and climb.
- The limitation on the loading carrying is the strength of the material, but not the torque provided by the motor.
- More gear tooth of servo gear is needed than the theoretical value.
- Acrylic is more elastic than our expectations, and large deformations of it should be taken into account in the design.

For a broader context, we find that this device is very practical and useful. However, more improvement is needed to increase the expansion radius, which can enable a broader use.

Our device has a broader application, especially in fields dealing with complex and complicated terrain, because it combines stability of the caterpillar and flexibility of the transformable wheels.

10 References

[1] K. Tadakuma et al., "Armadillo-inspired wheel-leg retractable module," *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Guilin, 2009, pp. 610-615.

doi: 10.1109/ROBIO.2009.5420604

[2] Yu She, C. J. Hurd and H. Su, "A transformable wheel robot with a passive leg," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, 2015, pp. 4165-4170.

doi: 10.1109/IROS.2015.7353966

[3] Z. Luo et al., "A reconfigurable hybrid wheel-track mobile robot based on Watt II six-bar linkage," *Mechanism and Machine Theory*, 2018, pp. 16-32.

11 Appendix

11.1 Contribution

Yixuan Wang is an undergraduate student of University of Michigan - Shanghai Jiao Tong University Joint Institute.

He is responsible for the electrical components, connecting the circuit and programming.

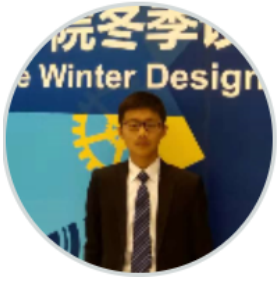


Figure 21: Portrait of Yixuan Wang

Zhenting Wang is an undergraduate student of University of Michigan - Shanghai Jiao Tong University Joint Institute.

He is responsible for the mechanical parts and assembly, designing the deformable structures, building CAD.

Figure 22: Portrait of Zhenting Wang

Yunzheng Yang is an undergraduate student of University of Michigan - Shanghai Jiao Tong University Joint Institute.

He is responsible for purchasing and transportation.

Figure 23: Portrait of Yunzheng Yang

Yihao Chen is an undergraduate student of University of Michigan - Shanghai Jiao Tong University Joint Institute

He is responsible for safety and management.

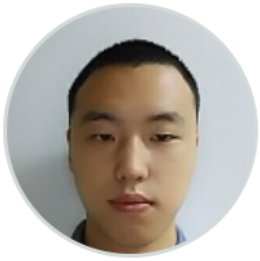


Figure 24: Portrait of Yihao Chen

11.2 Gantt Chart

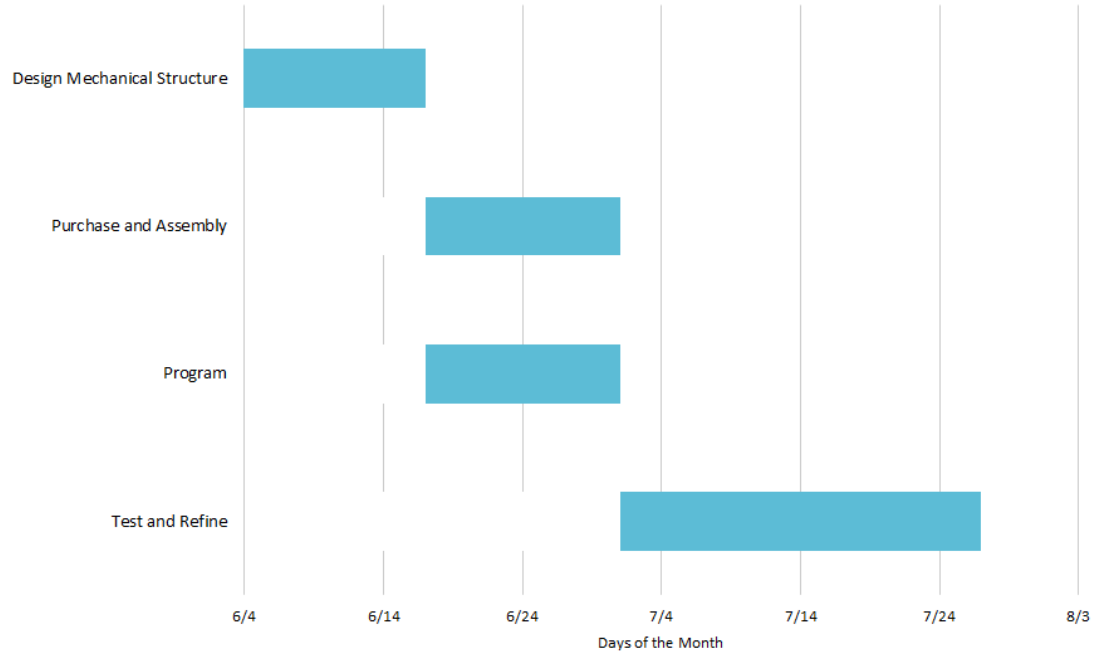


Figure 25: Gannt Chart for the Project

11.3 Budget and Real Cost

Item	Budget [RMB]	Cost [RMB]
Acylic chassis board	40.0	38.6
DS3218 Servos	200.0	245.0
25GA370 Motors	150.0	70.8
Axle connectors	20.0	14.4
NSK F683ZZ bearings	100.0	88.2
Aluminum flanges	30.0	38.2
HC-SR04 Ultrasonic Sensors	20.0	27.6
Screws and fasteners	30.0	33.6
Battery	100.0	75.0
Wires	10.0	8.6
Nylon buckle belt	10.0	7.7
Total	710.0	647.7

11.4 Programming Codes

11.4.1 Main program

```
/*
 * 07/06/2019
 * Prepare for test:
 * - test all functions
 * - test all conditions
```

```
 */

/*
 * Following is to define necessary lib
 */

#include "Sensor.h"
#include "DCMotor.h"
#include "Accelerometer.h"
#include <Servo.h>
#include <Wire.h>

/*
 * Following is for the communication
 */

const int MPU_ADDR = 0x68;
const byte PWR_MGMT_1 = 0x6B;
const byte ACCEL_CONFIG = 0x1C;
const byte ACCEL_XOUT_H = 0x3B;
const byte GYRO_CONFIG = 0x1B;

/*
 * Following is to define necessary function
 */

void goStraight();
void goStraight_full();
void wait();
void turn();
void reverse();
void foreExpand();
void foreShrink();
void backExpand();
void backShrink();
bool ifClimbing(float ax, float boundVal);
bool ifFalling(float ax, float boundVal);

/*
 * Following is to define necessary parameter of the program
 */

int generalSpeed = 125; // cannot be too large or too small: large:
    cannot analogWrite; small: not powerful
int targetDis = 15; // This is the target distance of sensor to the
    right wall
int expandDis = 10; // This is the position to expand the device
int turnDis = 20; // This is the position to turn.
int reverseDis = 50; // This is the position to reverse.
int stage = 1;
float boundVal1 = 0.3; // The ax of accelerometer to show it is climbing
```

```
float boundVal2 = -0.3; // The ax of accelerometer to show is it falling

/*
 * Following is to define all components we have
 */

Sensor sideSensor1(22,2);
Sensor sideSensor2(24,3);
Sensor frontSensor(27,4);

DCMotor foreLeft(5,28,29);
DCMotor foreRight(6,30,31);
DCMotor backLeft(7,32,33);
DCMotor backRight(8,34,35);

Servo foreLeftServo;
Servo foreRightServo;
Servo backLeftServo;
Servo backRightServo;

void setup() {

  Serial.begin(115200);
  // put your setup code here, to run once:
  foreLeftServo.attach(9);
  foreRightServo.attach(10);
  backLeftServo.attach(11);
  backRightServo.attach(12);

  /*
   * Following is to set up the accelerometer
   * SCL-21
   * SDA-20
   * AD0-GND
   */
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(PWR_MGMT_1);
  Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(MPU_ADDR);
  Wire.write(ACCEL_CONFIG);
  Wire.write(0x10);
  Wire.endTransmission();

  Wire.beginTransmission(MPU_ADDR);
  Wire.write(GYRO_CONFIG);
  Wire.write(0x08);
  Wire.endTransmission();
```

```
  int d1 = sideSensor1.distance();
  int d2 = sideSensor2.distance();
  delay(1000);
  foreShrink();
  delay(1000);
  backShrink();
  delay(3000);
}

void loop() {

  /*
   * The following is to get the data from the accelerometer
   */

  /*Wire.beginTransmission(MPU_ADDR);
  Wire.write(ACCEL_XOUT_H); // starting with register 0x3B
      (ACCEL_XOUT_H) [MPU-6000 and MPU-6050 Register Map and
      Descriptions Revision 4.2, p.40]
  Wire.endTransmission(); // the parameter indicates that the Arduino
      will send a restart. As a result, the connection is kept active.
  Wire.requestFrom(MPU_ADDR, 7*2); // request a total of 7*2=14
      registers

  // "Wire.read()<<8 | Wire.read();" means two registers are read and
      stored in the same variable
  float accelerometer_x = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x3B (ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
  float accelerometer_y = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x3D (ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  float accelerometer_z = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x3F (ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  float temperature = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x41 (TEMP_OUT_H) and 0x42 (TEMP_OUT_L)
  float gyro_x = Wire.read()<<8 | Wire.read(); // reading registers:
      0x43 (GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  float gyro_y = Wire.read()<<8 | Wire.read(); // reading registers:
      0x45 (GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
  float gyro_z = Wire.read()<<8 | Wire.read(); // reading registers:
      0x47 (GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)
  float gyroData[7];
  gyroData[0] = accelerometer_x/4096.2; gyroData[1] =
      accelerometer_y/4096.2; gyroData[2] = accelerometer_z/4096.2;
      gyroData[3] = temperature/340+36.53;
  gyroData[4] = gyro_x/65.5; gyroData[5] = gyro_y/65.5; gyroData[6] =
      gyro_z/65.5;
  */
  // put your main code here, to run repeatedly:
  int d = frontSensor.distance();
  Serial.println();
  Serial.print("Stage: ");
```

```
Serial.println(stage);
Serial.print("Front distance: ");
Serial.println(d);
Serial.print("Side distance 1: ");
Serial.println(sideSensor1.distance());
Serial.print("Side distance 2: ");
Serial.println(sideSensor1.distance());

/*
 * Following controls the device from the beginning to the position
    to expand
 */

if (stage == 1 && d > expandDis){
  goStraight();
}

/*
 * Following controls the device from the position to expand to the
    position preparing to climb
 */

else if (stage == 1 && d <= expandDis) {
  stage = 2;
  wait();
  delay(1000);
  foreExpand();
  delay(1000);
  backExpand();
  delay(1000);
}

else if (stage == 2){
  goStraight_full();
  delay(5000);
  stage = 3;
}

else if (stage == 3 && d > reverseDis){
  goStraight();
}
/*
 * Following controls the device from the position preparing to climb
    to the position finishing climbing
 */

else if (stage == 3 && d <= reverseDis){
  stage = 4;
  wait();
  delay(1000);
  backShrink();
```

```
    delay(1000);
    foreShrink();
    delay(1000);
    reverse();
  }

  else if (stage == 4 && d > turnDis){
    goStraight();
  }

  else if (stage == 4 && d <= turnDis){
    turn();
    stage = 5;
  }

  else if (stage == 5){
    goStraight();
  }
}

void goStraight(){
  int d1 = sideSensor1.distance();
  int d2 = sideSensor2.distance();
  int leftSpeed = generalSpeed *(1+10*(d1-d2)/(d1+d2));
  int rightSpeed = generalSpeed * (1-10*(d1-d2)/(d1+d2));
  Serial.print("Leftspeed: ");
  Serial.println(leftSpeed);
  Serial.print("Rightspeed: ");
  Serial.println(rightSpeed);
  if (leftSpeed < 0) {leftSpeed = 0;}
  else if (leftSpeed > 255) {leftSpeed = 255;}
  if (rightSpeed < 0) {rightSpeed = 0;}
  else if (rightSpeed > 255) {rightSpeed = 255;}
  foreLeft.forward(leftSpeed);
  backLeft.forward(leftSpeed);
  foreRight.forward(rightSpeed);
  backRight.forward(rightSpeed);
}

void goStraight_full(){
  foreLeft.forward(255);
  backLeft.forward(255);
  foreRight.forward(255);
  backRight.forward(255);
}

void wait(){
  foreLeft.forward(0);
  backLeft.forward(0);
  foreRight.forward(0);
  backRight.forward(0);
```

```
}

void turn(){
  foreLeft.backward(generalSpeed);
  backLeft.backward(generalSpeed);
  foreRight.forward(generalSpeed);
  backRight.forward(generalSpeed);
  delay(1900);
  /*
  while (abs(d1-d2)>1){
    foreLeft.backward(generalSpeed);
    backLeft.backward(generalSpeed);
    foreRight.forward(generalSpeed);
    backRight.forward(generalSpeed);
  }
   */
}

void reverse(){
  foreLeft.backward(255);
  backLeft.backward(255);
  foreRight.backward(255);
  backRight.backward(255);
  delay(2000);
}

void foreExpand(){
  int foreLeftExpandAngle = 0;
  int foreRightExpandAngle = 180;
  foreLeftServo.write(foreLeftExpandAngle);
  foreRightServo.write(foreRightExpandAngle);
}

void backExpand(){
  int backLeftExpandAngle = 0;
  int backRightExpandAngle = 180;
  backLeftServo.write(backLeftExpandAngle);
  backRightServo.write(backRightExpandAngle);
}

void foreShrink(){
  int foreLeftShrinkAngle = 180;
  int foreRightShrinkAngle = 0;
  foreLeftServo.write(foreLeftShrinkAngle);
  foreRightServo.write(foreRightShrinkAngle);
}

void backShrink(){
  int backLeftShrinkAngle = 180;
  int backRightShrinkAngle = 0;
  backLeftServo.write(backLeftShrinkAngle);
```

```
  backRightServo.write(backRightShrinkAngle);
}

bool ifClimbing(float ax, float boundVal){
  if (ax > boundVal) { return true;}
  else { return false;}
}

bool ifFalling(float ax, float boundVal){
  if (ax > boundVal) { return false;}
  else { return true;}
}
```

11.4.2 Accelerator library

```
#include "Accelerometer.h"
#include <Wire.h>
#include "Arduino.h"

const int MPU_ADDR = 0x68;
const byte PWR_MGMT_1 = 0x6B;
const byte ACCEL_CONFIG = 0x1C;
const byte ACCEL_XOUT_H = 0x3B;
const byte GYRO_CONFIG = 0x1B;

Accelerometer::Accelerometer(){
  Wire.begin();
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(PWR_MGMT_1);
  Wire.write(0);
  Wire.endTransmission();

  Wire.beginTransmission(MPU_ADDR);
  Wire.write(ACCEL_CONFIG);
  Wire.write(0x10);
  Wire.endTransmission();

  Wire.beginTransmission(MPU_ADDR);
  Wire.write(GYRO_CONFIG);
  Wire.write(0x08);
  Wire.endTransmission();
}

float* Accelerometer::data(){
  Wire.beginTransmission(MPU_ADDR);
  Wire.write(ACCEL_XOUT_H); // starting with register 0x3B
      (ACCEL_XOUT_H) [MPU-6000 and MPU-6050 Register Map and
      Descriptions Revision 4.2, p.40]
  Wire.endTransmission(); // the parameter indicates that the Arduino
      will send a restart. As a result, the connection is kept active.
```

```
  Wire.requestFrom(MPU_ADDR, 7*2); // request a total of 7*2=14
      registers

  // "Wire.read()<<8 | Wire.read();" means two registers are read and
      stored in the same variable
  float accelerometer_x = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x3B (ACCEL_XOUT_H) and 0x3C (ACCEL_XOUT_L)
  float accelerometer_y = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x3D (ACCEL_YOUT_H) and 0x3E (ACCEL_YOUT_L)
  float accelerometer_z = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x3F (ACCEL_ZOUT_H) and 0x40 (ACCEL_ZOUT_L)
  float temperature = Wire.read()<<8 | Wire.read(); // reading
      registers: 0x41 (TEMP_OUT_H) and 0x42 (TEMP_OUT_L)
  float gyro_x = Wire.read()<<8 | Wire.read(); // reading registers:
      0x43 (GYRO_XOUT_H) and 0x44 (GYRO_XOUT_L)
  float gyro_y = Wire.read()<<8 | Wire.read(); // reading registers:
      0x45 (GYRO_YOUT_H) and 0x46 (GYRO_YOUT_L)
  float gyro_z = Wire.read()<<8 | Wire.read(); // reading registers:
      0x47 (GYRO_ZOUT_H) and 0x48 (GYRO_ZOUT_L)
  float result[7];
  result[0] = accelerometer_x/4096.2; result[1] =
      accelerometer_y/4096.2; result[2] = accelerometer_z/4096.2;
      result[3] = temperature/340+36.53;
  result[4] = gyro_x/65.5; result[5] = gyro_y/65.5; result[6] =
      gyro_z/65.5;

  return result;
}
```

```
#ifndef ACCELEROMETER_H
#define ACCELEROMETER_H

#include <Wire.h>
#include "Arduino.h"

class Accelerometer {
  public:
    Accelerometer();
    float* data();
  private:
    int MPU_ADDR;
    byte PWR_MGMT_1;
    byte ACCEL_CONFIG;
    byte ACCEL_XOUT_H;
    byte GYRO_CONFIG;
};

#endif
```

11.4.3 DC motor library

```
#include "DCMotor.h"
#include "Arduino.h"

DCMotor::DCMotor(int ENA, int IN1, int IN2){
  _ENA = ENA;
  _IN1 = IN1;
  _IN2 = IN2;
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
}

void DCMotor::forward(int motorSpeed){
  digitalWrite(_IN1,HIGH);
  digitalWrite(_IN2,LOW);
  analogWrite(_ENA,motorSpeed);
}

void DCMotor::backward(int motorSpeed){
  digitalWrite(_IN1,LOW);
  digitalWrite(_IN2,HIGH);
  analogWrite(_ENA,motorSpeed);
}
```

```
#ifndef DCMOTOR_H
#define DCMOTOR_H

class DCMotor{
  public:
    DCMotor(int ENA, int IN1, int IN2);
    void forward(int motorSpeed);
    void backward(int motorSpeed);
  private:
    int _ENA;
    int _IN1;
    int _IN2;
};
#endif
```

11.4.4 Supersonic sensor library

```
#include "Sensor.h"
#include "Arduino.h"

Sensor::Sensor(int trigPin, int echoPin){
  _trigPin = trigPin;
  _echoPin = echoPin;
```

```
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

double Sensor::distance(){
  double d;
  digitalWrite(_trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(_trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(_trigPin, LOW);
  d = pulseIn(_echoPin, HIGH) / 58.00;
  return d;
}
```

```
#ifndef SENSOR_H
#define SENSOR_H

/*
 * Following is to define the class "Sensor"
 */
class Sensor{
  public:
    Sensor(int trigPin, int echoPin);
    double distance();
  private:
    int _trigPin;
    int _echoPin;
};
#endif
```
