Course Number: ME_GY 9966

MS Project Report

Autonomous Robotic Cart for Food Delivery on Airplane

Submitted in partial fulfilment for the degree of
Master of Science (MS) in Mechatronics and Robotics
by
Ziyu Wang

To the department of
Mechanical and Aerospace Engineering
(Fall 2017)

NYU | TANDON SCHOOL OF ENGINEERING

Abstract
A robot cart to deliver food on airplane autonomously is developed. In this report, the features of the robot cart are briefly introduced first. Then, the design of whole system is explained including circuit connection and programming algorithms. After that, the mechanical components are listed which are cart frame, wheel track, scissor screw and food box lifting structure. Following this, the electrical components are introduced including microcontroller board, manipulator, stepper motors. After these, there will be experiments to test the whole system and results will be recorded and discussed.

1. Instruction

The aim of this project is to develop a robot cart to deliver food on airplane. Size of the robot cart is designed to be 35in× 34in× 15in. The weight is designed to be 15Kg in total. Two kinds of food will be stored in two sides of the cart and each contains two columns. The process can be roughly divided into four sections: Move, Stop, Choose and deliver. In the first section, if the cart is powered by battery, it can move forward or backward sprightly. It can also turn right or left according to the design of differential wheels. When the camera on the side of that robot cart detects required QR code, it will stop. In this section, image processing is applied. After this, passengers on the airplane can choose what kind of food, for example chicken or vegetable, they like through an interface. Then the robot can deliver the food to that passenger. In section four, a scissor screw system will push out ordered food box into the middle of the cart cabin. Then a manipulator will grip the food box and send it to customers in the same line one by one. Once finishes this entire loop, the robot cart will move into the next loop and send food to next customer. After servicing the last line on the airplane, the robot cart will go back to preparing room automatically.

2. Mechanical section

2.1 Frame design

The structure of this robot is quite similar to that of the food cart served on airplane nowadays. The length of the cart is critical because the aisle is always very long. The key point is to decide the width and the height of the robot cart. According to resource, the width of most of airplanes is 17 inches **[1]**. Therefore, the width is designed to be 15 inches. The height of airplane seats is different in different type of airplane or even same type of airplane but different airplane companies. The height should be good to send out food for a person sitting. In other word, it should be at the level near the neck of a sitting customer. Normally, this height is around 35 inches to 45 inches. The weight of the cart is divided into three parts: food, mechanical parts, electrical parts. Each food box is designed to be 200g, and it is designed to contain around 50 food boxes. Mechanical components are designed to be 2 kilograms whereas the electrical components are 3 Kg. To summary, the length of the cart it designed to be 35 inches, the width is 15 inches, the height is 34 inches (wheels are not included) and the weight is 15 Kg in total. To minimize the manufacture process, the frame is make by wood.

Figure 2.1: 3D vision of the frame using SolidWorks

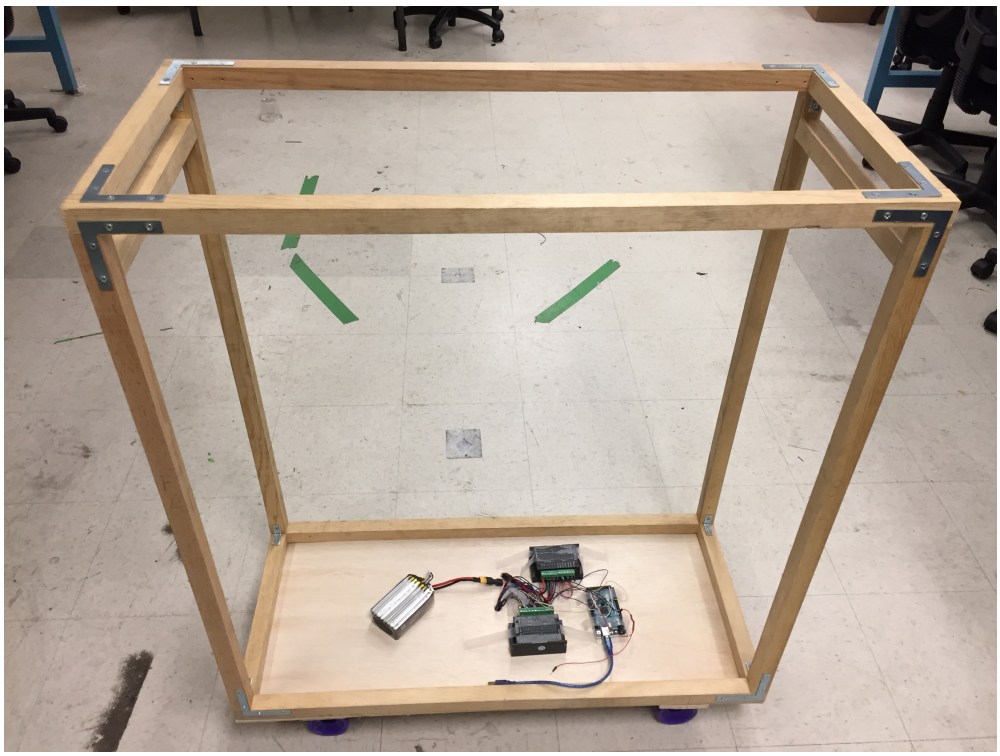

Figure 2.2: Wooden frame of the robot cart

2.2 Wheel track design

Because as the wheel base increasing, stability of the cart is guaranteed whereas the maneuverability of the cart is decreased. In this project, the stability of the car is more important than the maneuverability. Also, the weight center of manipulator should also be considered when mounting wheels. Therefore, the four wheels are mounted near edges. The center of each wheel is 4 inches to the shorter edge and 2 inches to the longer edge. To make the cart move smoothly and reliable, the motor should be mounted stable and the connection between motor and wheel should be designed. A wooden slot is designed to stable lock the motor. Four long screws pass through the holes on motor and holes on wooden slot.

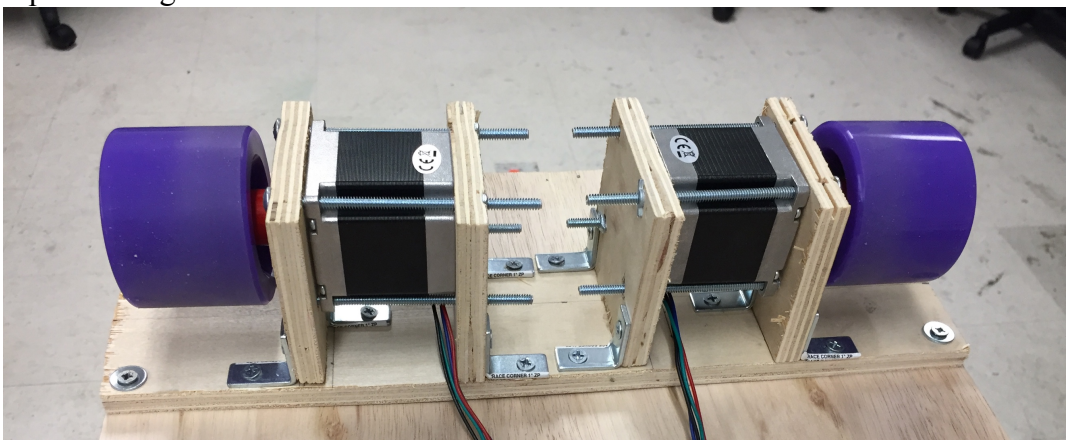

Figure 2.3: Wooden slot to lock motors

2.3 Scissor screw design

Once the robot gets the order from customer, a scissor screw will push a food box out to the middle platform. Figure 2.4 shows the structure of the scissor screw. From bottom to the top, the device is combined by a main frame, a screw bar, two sliders, two arm fixings and eight arm bars. The screw bar is driven by the stepper motor. When the screw bar is spinning, the slider will be driven like a screw hat. Notice that, the left slider is fixed on the main frame, therefore, the right one will slide left or right. In this way, the arms are driven up and down. The stretching speed can be controlled by the speed of stepper motor. The initial stretch length is 2 inches whereas the maximum stretch length is 11 inches. Figure 2.5 shows the scissor screw in fully stretched situation. All of the components are printed out using 3D printer.

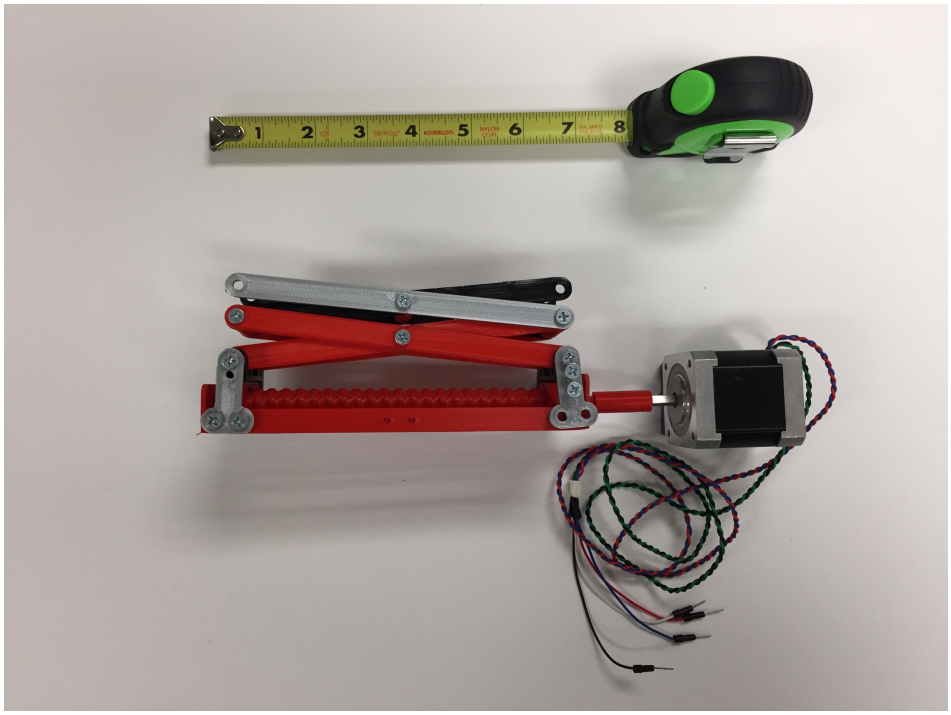

Figure 2.4: Structure of scissor screw device

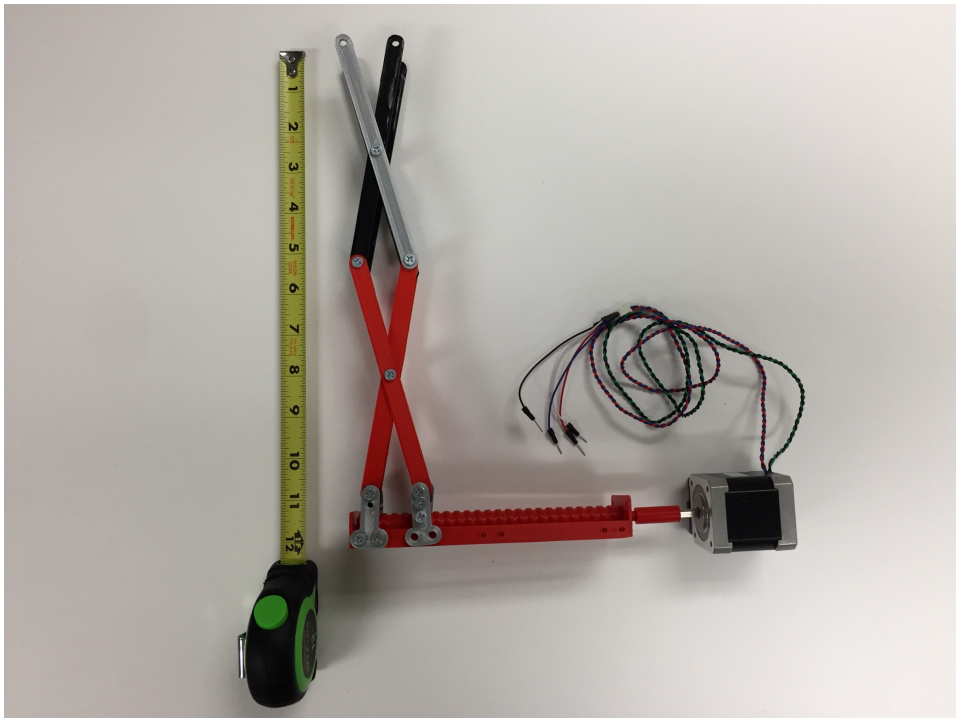

Figure 2.5: Scissor screw in fully stretch state

2.4 Automatically lift system design

After one food box is pushed into the middle platform, the next food box should be compensated to the initial position and wait to be pushed into the middle platform. A lift system is designed to accomplish this process. As shown in Figure 2.6, the platform to hold boxes is connected to top shell through four springs. Once the top food box is pushed out and the scissor screw is stretched back to initial state, a new food box would be driven up by the lift system. The disassembly view of the spring and platform is shown in Figure 2.7

Figure 2.6: Automatic lifting system

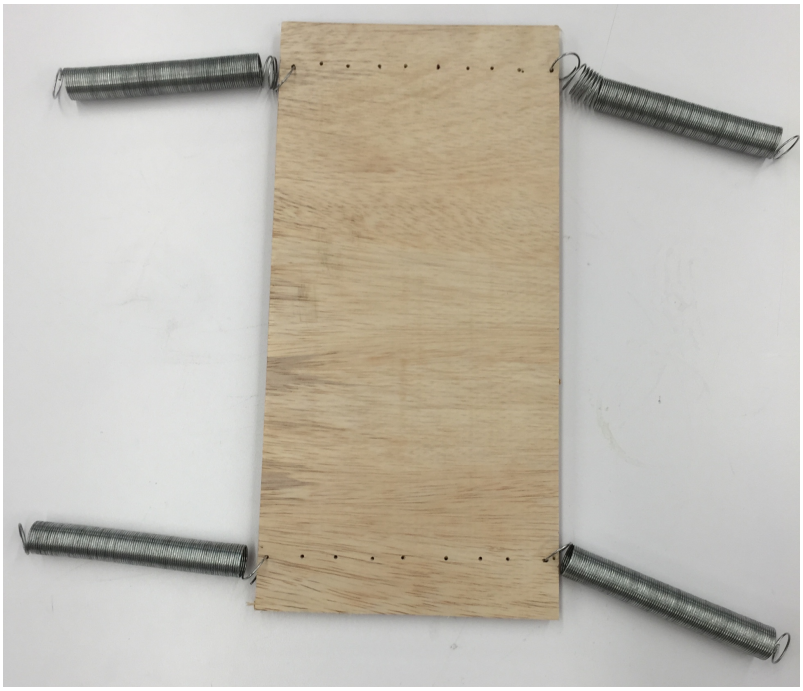

Figure 2.7: Disassembly view of the springs and platform

2.5 Entire robot cart overview

The entire overview of the cart is shown in Figure 2.8.

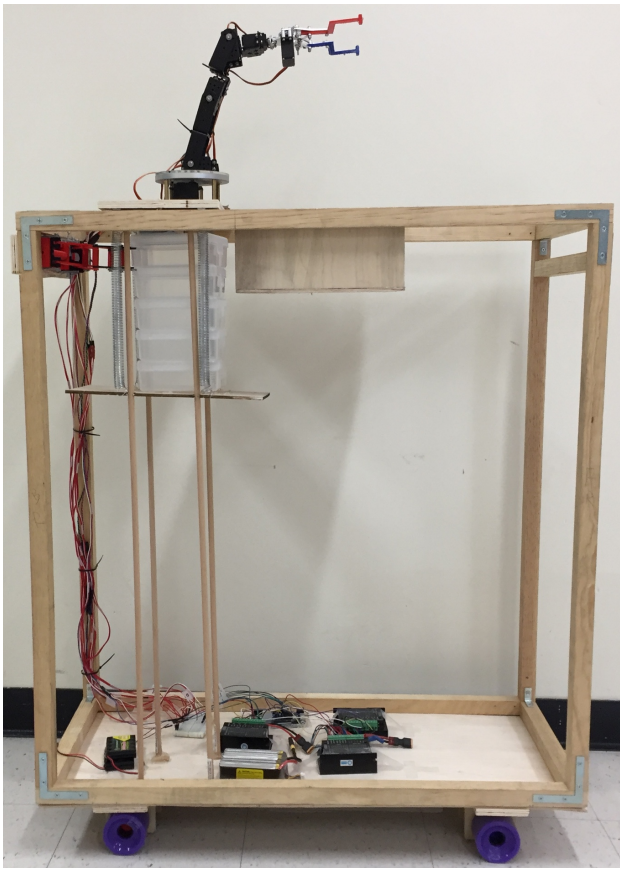

Figure 2.8: Entire robot cart overview

3. Electronics section

3.1 Arduino Mega

Arduino MEGA is a microcontroller of Arduino series boards. It has 15 PWM pins and this is most important feature for this project. As shown in Figure 3.1, 12 PWM pins are used in this project. Among these 12 PWM pins, 6 pins (from pin8 to pin13) are used to control manipulator joint motors and others are used to control stepper motor drivers.

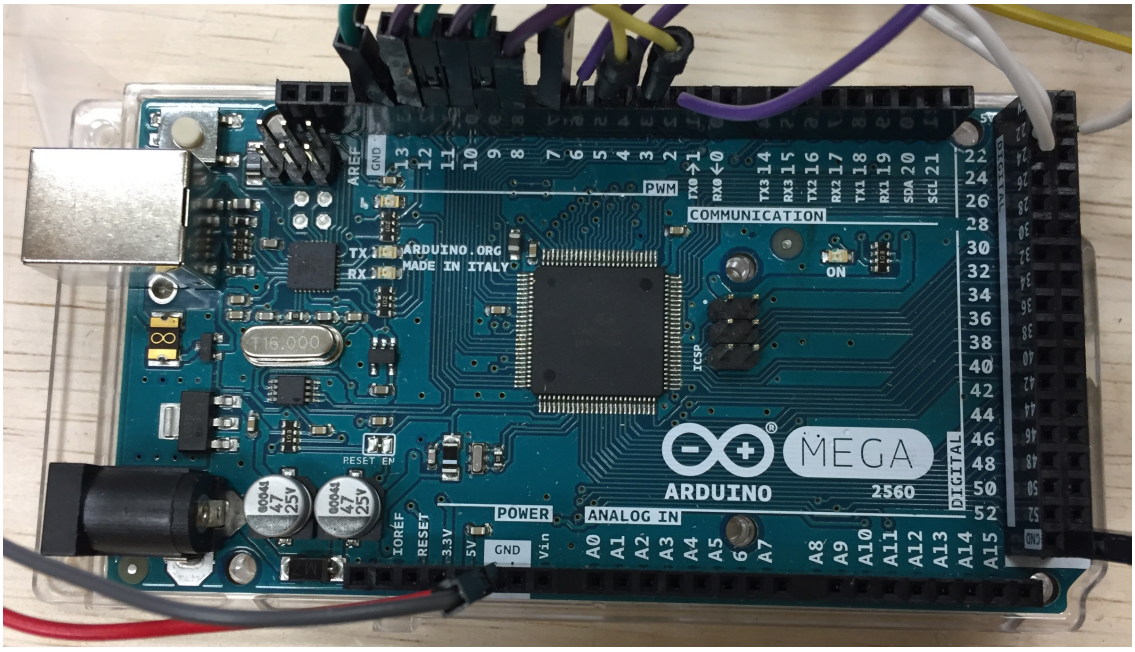

Figure 3.1: Arduino MEGA microcontroller

3.2 Raspberry Pi

Raspberry Pi is normally suitable for image processing. In this project, QR code detection applies image processing. Therefore, a Raspberry Pi microcontroller is needed.

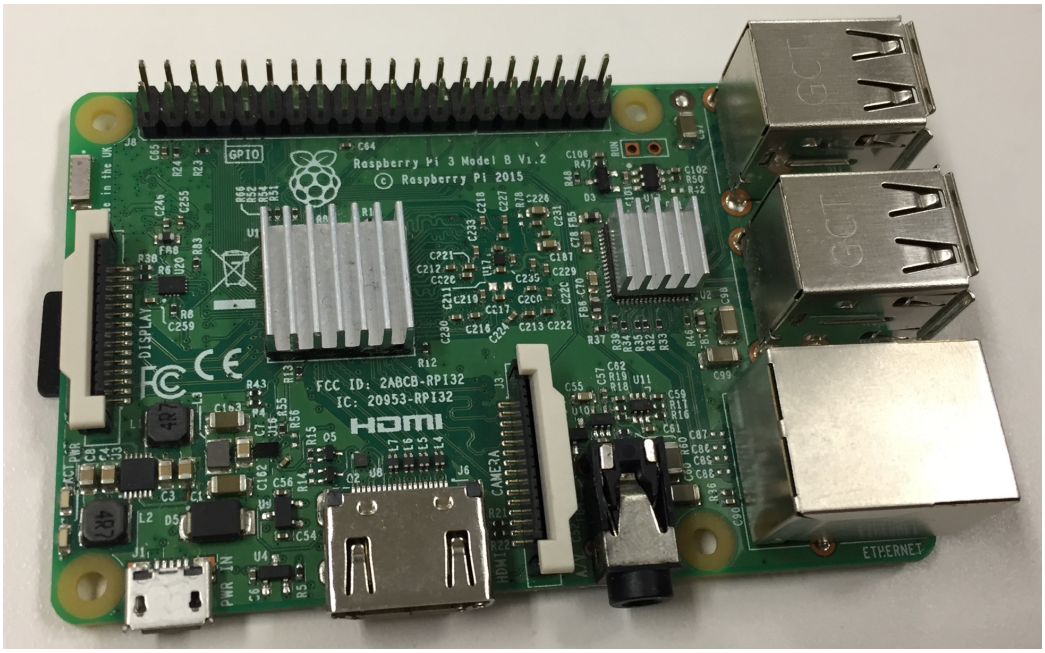

Figure 3.2: Raspberry Pi3 model B

3.3 Manipulator

In this project, a robot arm is used to grip box and send it to customer. This is a 6 degree-of-freedom robot arm. Rotation angle of five joint motors is 180 degrees. The griper is driven by a motor and the rotation angle of this motor is from 120 degree to 180 degree. The motors are with same model, MG996R. Rated torque for this motor is 15Kg-cm (with 6V power), operating voltage is from 4.8V to 7.2V. When the motor is in proper working state, the current for each motor is around 0.4A. However, if the joint motor is blocked or stuck, the current can be around 1 A which is dangerous for that motor, it could be burned because of that high current. Apart from the statistics of this robot arm, the movement of this robot arm is achieved by setting joint angle of each joint motor. The initial position and end effect point are all fixed, angles of motor can be calculated by inverse kinematics. Theoretically, different moving order of these six motors can achieve same position without any barratry. However, the edges of robot cart frame can block the movement of robot arm. Therefore, the real trajectory of the robot arm should be test.

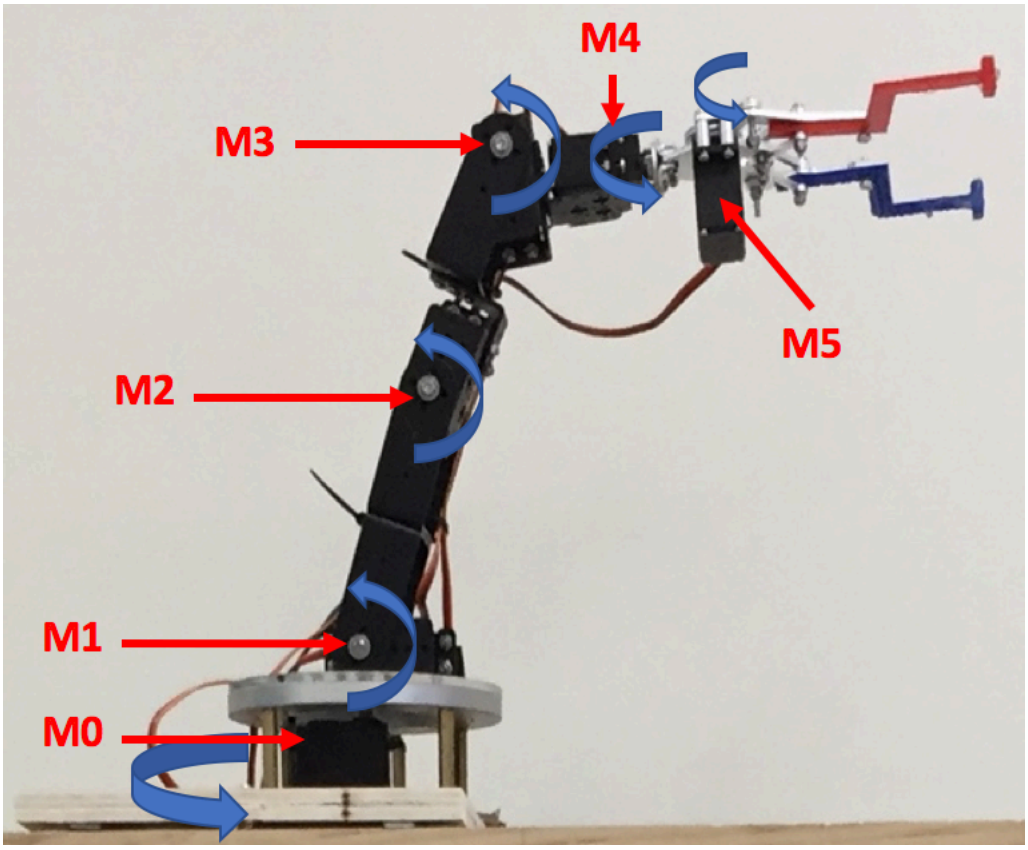

Figure 3.3: Manipulator used in this project

3.4 Stepper motor

There are two types of stepper motors used in this project. Four NEMA23 motors are used to drive wheels and one NEMA17 is used to drive scissor screw set. As for NEMA 23, according to the research, the torque provided by each motor can be calculated as follow:

$$\tau_{total} = \mu \times m \times r \quad (1)$$

where τ_{total} is the total torque provided by the four motors. μ is is the coefficient of friction, which is normally assumed to be 0.6, m is the mass of the whole robot cart and r is the radius of each wheel. Then, the torque for each motor can be calculated as equation (2):

$$\tau_{each} = \frac{1}{4}\,\tau_{total} \quad (2)$$

in this project, the torque of each wheel is calculated in equation (3):

$$\tau_{each} = \frac{1}{4}\,\tau_{total} = \frac{1}{4}\mu \times m \times r = \frac{1}{4} \times 0.6 \times 15 \times 3.5 = 7.875\ \text{Kg} - \text{cm} = 78.75\ \text{N} - \text{cm} \quad (3)$$

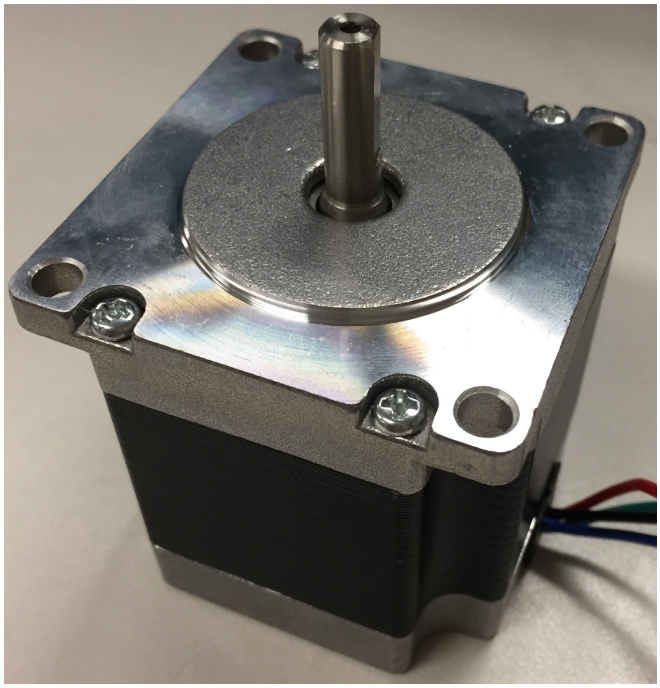

Figure3.4: NEMA23 stepper motor

3.5 Stepper motor driver

High current is required to drive stepper motor, approximately 1A per motor. This amount of current can't be supplied by Arduino. At the same time, 4 wires should be connected to battery for one stepper motor. To meet the former two requirements, a stepper motor driver is need. Stepper motor TB6600 can work under a voltage range of 9V to 42V. Maximum current can be provided is 4A, which cover the requirement of this project.

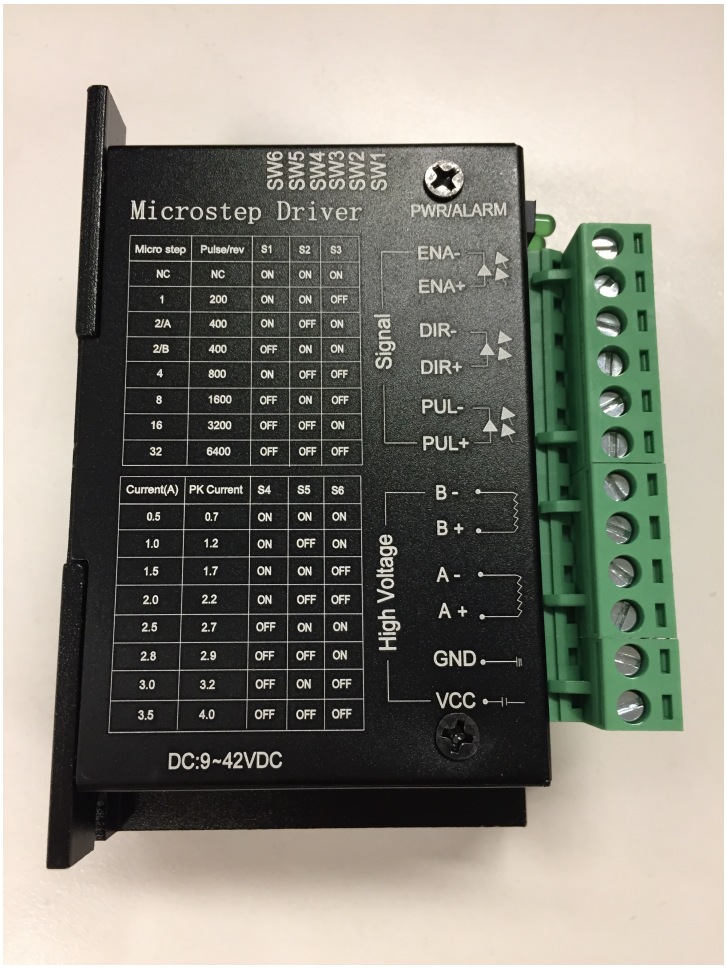

Figure 3.5: Stepper motor driver TB6600

A table is printed on the shell of motor driver shown in Figure 3.5. It is quite necessary to understand this table. This table shows how to set this driver. There are 6 switches to set model of stepper driver. The first 3 switches set the pulse per revenue. This is how a revenue is divided. If it is set to be 200, it will request 200 steps to run a cycle. Three switches with on and off states can generate 8 states in total. Therefore, there are 8 different settings. Other three switches are related with the rated current of the motor which will be driven by this driver. If the rated current of the motor is 2.5A, switch 4 should be off with switch 5 and 6 are on. Similar to the first three switches, these three switches determines 8 current level.

3.6 Camera

Camera used in this project is Pi camera which is shown in Figure 3.6. It is used to detect QR code on chairs. It can support 1080p30 and 720p60 video modes.

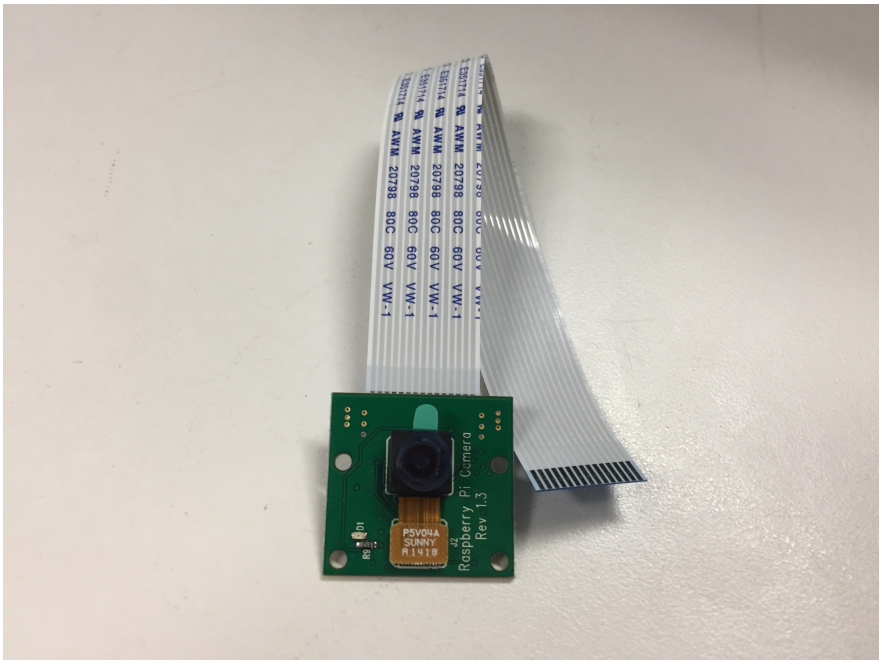

Figure 3.6: Pi camera used in this project

3.7 Interface

An interface for customer to choose food kind should be designed. It is an iPad application in this project. However, this function can be added into customer touch screen in the future.

4. System design and development

4.1 Component connections

Figure 4.1 shows connections between each component of in this project. In general, this robot cart is mainly controlled by Arduino MEGA, which is assisted by a Raspberry Pi. In detail, the connections can be divided into four sections: control of wheels, control of scissor screw, control of manipulator and information transformation. In the wheel section, stepper motor drivers are controlled by Arduino MEGA directly. A stepper motor driver is used to provide external power supply to stepper motor. This is because high current is required to drive a stepper motor whereas the it is much higher than the ability of Arduino MEGA. Then, one driver is connected with two left wheels and the other one is connected with two wheels on the right. The reason for this is, when robot cart moving forward, wheels on different sides spinning in against directions. Both of the stepper motors are powered by a 12 volts battery. Raspberry Pi is also related with this process. A camera is used to detect QR code labelled on chairs. Once a QR code is detected, Raspberry Pi will send a message to Arduino and Arduino will stop stepper motors to make the robot stop. This is everything related with connections in wheel control.

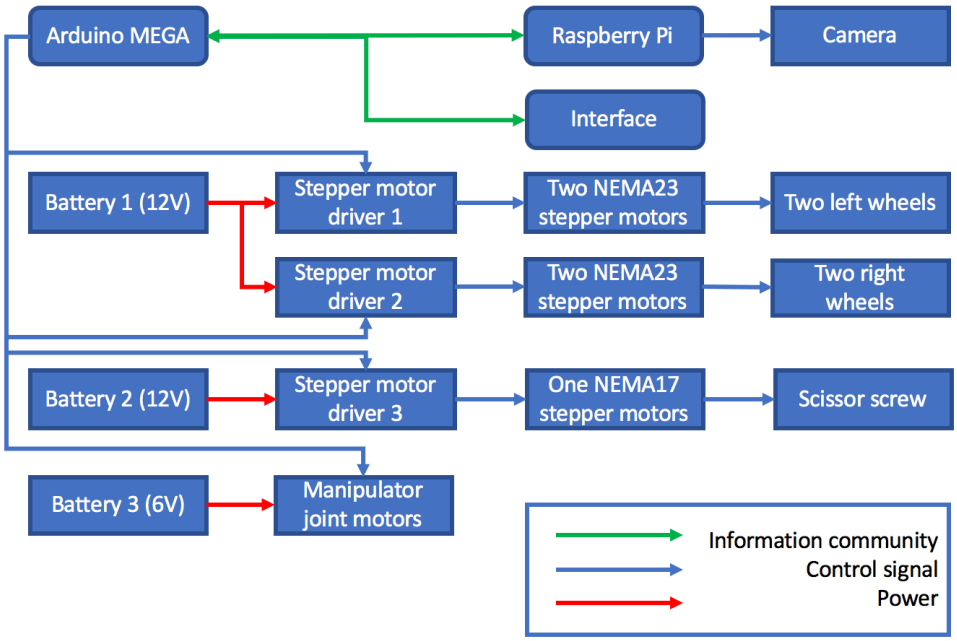

Figure 4.1: Components connection diagram

In scissor screw control part, it is quite similar with that in wheel control part. The different is that a stepper motor with smaller torque is mounted in this part. In detail, a stepper motor driver is connected to Arduino MEGA directly and a stepper motor NEMA17 is controlled to drive the scissor screw. The interface is also related with this process. Customer should be able to choose what kind of food they want and the instruction would be transferred to Arduino then the scissor screw can send out right orders. The last part is manipulator. The working principle of the robot arm in this project is to control the joint angles. Joint motor are the components to control to accomplish a movement. Therefore, six joint motors are directly connected by Arduino MEGA and powered by a 6 volts battery without any motor driver. This is how the components connected in this project.

4.2 Programming algorithms

For Arduino MEGA, the it is a C programming environment. For Raspberry Pi, python is used to program code.

4.3.1 Entire system

Once the system is activated, the robot cart begins to move forward. At the same time, camera is detecting the QR code. If a line number shows up, Raspberry Pi will send a message to Arduino MEGA to stop the movement of robot cart. Then, customer can choose one kind of food through the interface on robot cart. The message will be sent to Arduino, and Arduino will drive scissor screw set to push a food box into the middle platform. After this, the food box will be gripped up by robot arm and deliver to customer. In the end, the manipulator will move back to initial position and a signal will be send to Arduino MEGA to move robot cart forward again. Figure 4.2 shows the process of the entire process.

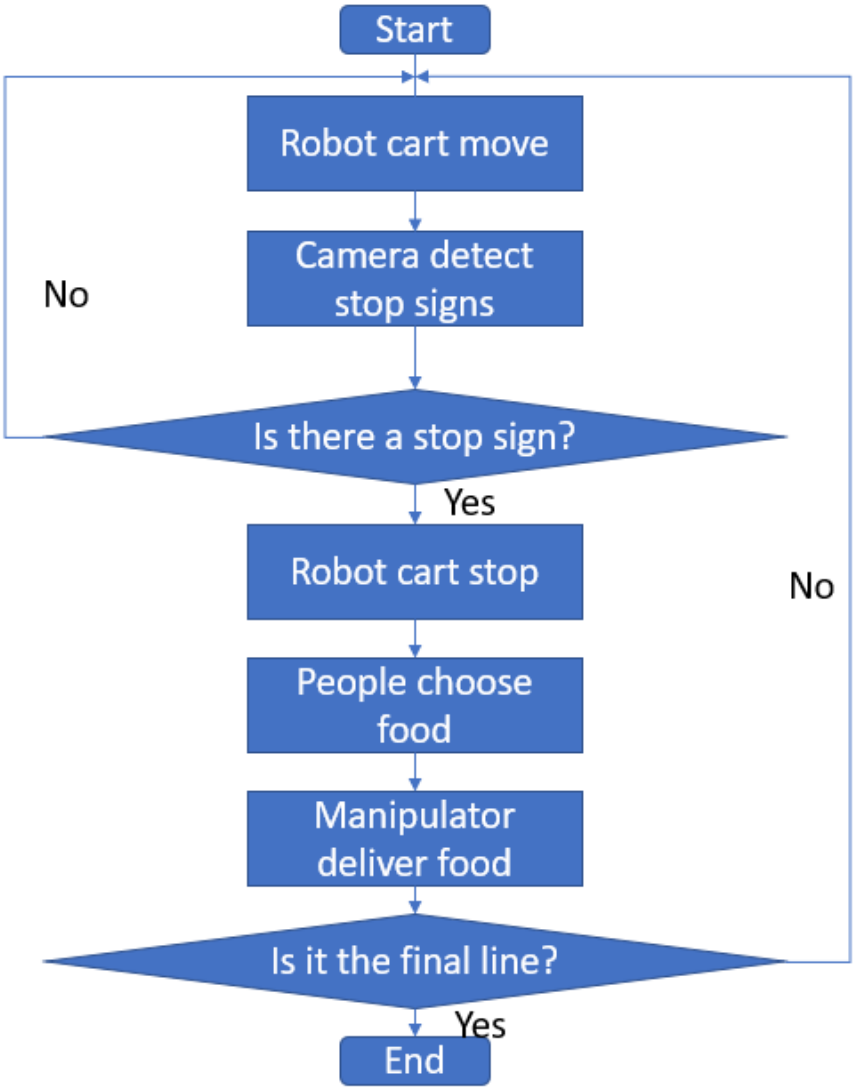

Figure 4.2: Flow chart of the entire system

4.3.2 Manipulator deliver food

After receiving message from interface, scissor screw set will push a food box into middle platform. Then, manipulator will grip it up and send it to customer. If nobody is picking that food box up, the manipulator will grip back that food box and collect it into the middle of robot cart.

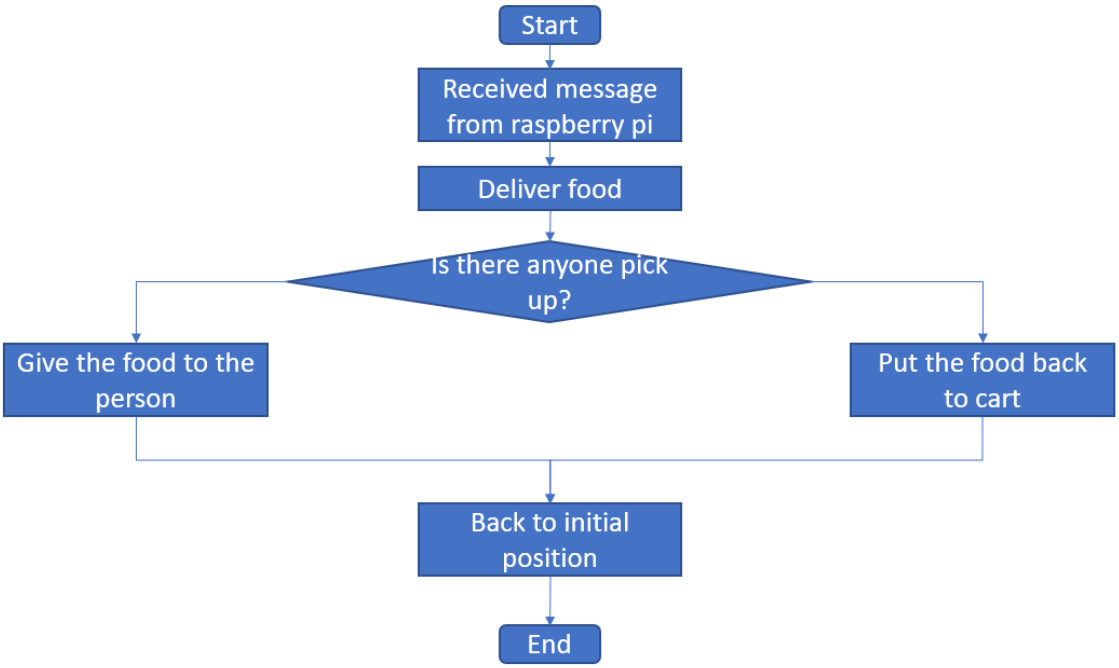

Figure 4.3: Flow chart for delivering food box

5. Experiment

5.1 Experiment plan

Firstly, QR code detection should be tested. Then, the entire system should be tested. After that, gesture recognition should be tested. All experiment results should be recorded for later analysis.

6. Experiment results

7. Discussion

8. Conclusion

Up till now, the robot cart hardware is developed. The robot cart can move, which means the wheel system works. Also, the manipulator can be controlled to pick up a food box and send it to customer. In the near future, the whole system will be tested and some experiments will be conducted to test the function of that cart. Moreover, gesture recognition will be researched and the robot will be able to recognize the existence of hand in the future.

Reference:

[1]. https://ultimaker.com/en/products/ultimaker-2-plus#Ultimaker-2+

Appendix:

Appendix A: Arduino code to drive cart wheels

```
int PULL=22; //define Pulse pin
int DIRL=2; //define Direction pin
int ENAL=3; //define Enable Pin
int PULR=24;
int DIRR=4;
int ENAR=5;

void setup() {
  pinMode (PULL, OUTPUT);
  pinMode (DIRL, OUTPUT);
  pinMode (ENAL, OUTPUT);
  pinMode (PULR, OUTPUT);
  pinMode (DIRR, OUTPUT);
  pinMode (ENAR, OUTPUT);
}
void loop() {
    for (int i=0; i<1000; i++)    //Forward 1000 steps
  {
    digitalWrite(DIRL,HIGH);
    digitalWrite(ENAL,LOW);
    digitalWrite(PULL,LOW);

    digitalWrite(DIRR,HIGH);
    digitalWrite(ENAR,HIGH);
    digitalWrite(PULR,HIGH);
    delayMicroseconds(2000);

    digitalWrite(PULL,LOW);
    digitalWrite(PULR,HIGH);
    delayMicroseconds(2000);
  }

  for (int i=0; i<500; i++)    //stop 500 steps
  {
    digitalWrite(DIRL,LOW);
    digitalWrite(ENAL,LOW);
    digitalWrite(PULL,LOW);

    digitalWrite(DIRR,LOW);
    digitalWrite(ENAR,HIGH);
    digitalWrite(PULR,HIGH);
    delayMicroseconds(2000);

    digitalWrite(PULL,HIGH);
    digitalWrite(PULR,LOW);
    delayMicroseconds(2000);
  }
}
```

Appendix B: Arduino code to control scissor screw set

```
int PUL=23; //define Pulse pin
int DIR=6; //define Direction pin
int ENA=7; //define Enable Pin
void setup() {
  pinMode (PUL, OUTPUT);
  pinMode (DIR, OUTPUT);
  pinMode (ENA, OUTPUT);

}

void loop() {
  for (int i=0; i<5000; i++)    //STOP
  {
    digitalWrite(DIR,HIGH);
    digitalWrite(ENA,LOW);
    digitalWrite(PUL,LOW);
    delayMicroseconds(100);
    digitalWrite(PUL,LOW);
    delayMicroseconds(100);
  }
  for (int i=0; i<4000; i++)   //GO UP
  {
    digitalWrite(DIR,HIGH);
    digitalWrite(ENA,LOW);
    digitalWrite(PUL,LOW);
    delayMicroseconds(300);
    digitalWrite(PUL,HIGH);
    delayMicroseconds(300);
  }
  for (int i=0; i<5000; i++)    //STOP
  {
    digitalWrite(DIR,HIGH);
    digitalWrite(ENA,LOW);
    digitalWrite(PUL,LOW);
    delayMicroseconds(500);
    digitalWrite(PUL,LOW);
    delayMicroseconds(100);
  }
  for (int i=0; i<4000; i++)   //GO DOWN
  {
    digitalWrite(DIR,LOW);
    digitalWrite(ENA,LOW);
    digitalWrite(PUL,LOW);
    delayMicroseconds(300);
    digitalWrite(PUL,HIGH);
    delayMicroseconds(300);
  }
}
```

Appendix C: Arduino code to control robot arm pick up and send food box

```
#include <Servo.h>


Servo servo0;          // create servo object to control servo0
Servo servo1;          // create servo object to control servo1
Servo servo2;          // create servo object to control servo2
Servo servo3;          // create servo object to control servo3
Servo servo4;          // create servo object to control servo4
Servo servo5;          // create servo object to control servo5
int x = 0;

void setup() {
  servo0.attach(13);   // attaches the servo on pin 13
  servo1.attach(12);   // attaches the servo on pin 12 to the servo object
  servo2.attach(11);   // attaches the servo on pin 11 to the servo object
  servo3.attach(10);   // attaches the servo on pin 10 to the servo object
  servo4.attach(9);    // attaches the servo on pin 9 to the servo object
  servo5.attach(8);    // attaches the servo on pin 8 to the servo object
}

void loop () {
//initial position
  servo1.write(10);
  delay(100);
  servo2.write(90);
  delay(100);
  servo3.write(10);
  delay(100);
  servo4.write(30);
  delay(100);
  servo5.write(170);   //girpper
  delay(100);
  servo0.write(90);    //base 90 for forward
  delay(300);


//initial position transfer to the first grip
  servo0.write(65);    //turn base
  delay(300);
  servo4.write(70);     //turn wrist a little bit
  delay(300);
  servo1.write(45);    /backarm drap a little bit
  delay(300);
  servo2.write(70);    // 90 for virtical
  delay(300);
  servo4.write(10);    //change position ready to grip
  delay(300);
  servo5.write(120);   //girpper release
  delay(300);
```

```
  servo3.write(20);
  delay(300);
  servo1.write(60);      //back arm drap a little bit more
  delay(300);

  servo5.write(180);    //girp box!!!!!!!!!!
  delay(300);
  servo1.write(40);      //gripper lift up a little bit with box
  delay(300);
  servo0.write(90);      //base 90 for forward
  delay(300);


//back to initial position
  servo1.write(10);      //back to initial position
  delay(300);
  servo2.write(90);      //back to initial position
  delay(300);
  servo3.write(10);      //back to initial position
  delay(300);
  servo4.write(30);      //back to initial position
  delay(300);
  servo5.write(170);    //girpper change position
  delay(300);


//send box to customer
  servo0.write(170);    //base turn left 90 degree
  delay(300);
  servo1.write(45);      //send out backarm
  delay(300);
  servo3.write(45);      //send out frotarm
  delay(3000);            //wait for 3s
  servo5.write(120);    //girpper release
  delay(300);
  servo1.write(10);      //back to initial position
  delay(300);
  servo3.write(60);      //back to initial position
  delay(300);
//finally back to initial position
  servo0.write(90);      //base turn left 90 degree
  delay(5000);    //wait for 3s

}
```