ACS6502 GroupProject (GP) Final Report

"Grippy-bot"

Saba Firdaus Ansaria (210110201), Eduvie Emagha (210197567), Weijie Zhao (210110186)

1. DEVELOPED SYSTEM OVERVIEW

The robot is a mobile autonomous vehicle with a motor controlled arm and a gripper to pick up and drop small objects at designated locations.

The arm and the gripper have one servo motor, moving with predefined angular rotations. The robot mainly performs two tasks. In the first task, it picks up an object from the starting point and drops the object at a two-metre distance. The second task is more complicated where it picks up the object from the starting line and moves forward. When it locates an obstacle in front, it performs an obstacle avoidance maneuver to avoid clashing with the barrier and finally correct its trajectory to return to the original path and drop the object at a two-metre distance from the starting point. The main components of the robot include a motorised hand, a gripper and a distance sensor. The wheels are controlled by an additional motor driver and a set of DC motors.

The mechanical system has five major components: the wooden arm with servo motor, wooden gripper with mechanical tension spring and servo motor, wooden body chassis, plastic wheels, and round ball-shaped frontal support for the robot vehicle. The electrical circuit system is compiled using a modular approach by dividing each sub-functionalities into separate modules at separate breadboards. The intention is to make the robot's implementation system interpretable and easy to debug. The descriptions of the mechanical and electrical modules have been given below.

1. The components have been assembled while considering the weight distribution of the robot because an unbalanced weight distribution could destabilise the robot and put unnecessary extra loads on the motors. It is overall making the design minimalistic and efficient.
2. The motor control hands are made with seven lightweight wooden robot arm linkage, which is mounted on the chassis board with a plastic printer holder for MG996. The overall structure has minimal vibration and does not hinder the object gripping. The hand is connected to the gripper with a small wooden block at a tilted angle.
3. The plywood made gripper has two semi-circular wooden arms, one mechanical spring and a servo motor and one of the MG996 RC servo motors. One of the armrests is fixed with the mechanical spring, and the other end of the mechanical spring is mounted on the other hand. The servo motor moves the hand from 0-130 degrees to open and close the hand. The mechanical spring puts pressure on the object in the grip to keep it from dropping. Laser cutting is used with the CAD hybrid models to create these components. We have used plywood to reduce the overall weight, as discussed earlier.
4. The chassis is made from plywood using laser cutting. We also created 3D components with acrylic PLA, but finally, we chose plywood components because making impromptu design changes with the modules' positioning was easier in plywood.
5. The plastic wheels given with the Arduino kit were used. However, these wheels are inefficient and do not work well on smooth surfaces (floor). Therefore, rubber bands are semi-melted and mounted on the wheels.
6. A TB6612FNG Motor Driver Board controls the wheels, and the white-blue-green wire colour code has been used for each of the wheel's control connections with the TB6612FNG and the Arduino UNO (digital pins 3-8). The red-black colour-coded wires have been connected from the DFRobot Micro DC Motors to the motor driver.
7. The hand motor is an MG996 RC servo motor where the control wire (yellow) is connected to Arduino UNO(digital pin 2). The gripper motor is also an MG996 RC servo motor where the control wire (yellow) is connected to Arduino UNO(digital pin 12).
8. The ultrasonic HC-SR04 distance sensor's echo and trig pins are connected to Arduino

UNO(digital pin 9,10) and colour-coded as Green-Yellow.

9. The DC motor's, servo motors, ultrasonic sensor, Arduino board's ground and Vcc are connected to an external 9V battery pack.
10. The battery pack is a collection of six 1.5V batteries (total 353 grams approximately) connected in series. The battery material is nickel metal hydride.

Some design considerations have been made in the robot such as power input/output, values of torques, speed and robot-decision making. We have found out that Arduino starts malfunctioning when it powers multiple components, such as two servo motors and a distance sensor in our case. Therefore only the control input/outputs have been taken from the Arduino board, and the Vcc and grounds are connected with the abovementioned external source. Arduino UNO is the perfect choice for the task due to the optimal number of pins and the efficient power consumption for the tasks.

The speed of the wheels is set externally and adaptively with a mapping function. If it is too fast, then the response delay with the obstacle does not work correctly. At task 1 this issue makes the robots stop at different distances from the finish line. At task 2, the obstacle avoidance mechanism does not accurately correct the robot's trajectory and the previous stopping inaccuracy. Also, another factor is the wheels' surface friction, which minimally changes the robot's trajectory but affects the overall accuracy of the object dropping position. We keep the robot's moving and turning speed between 30-400 which maps between 0-1023. Also, the delay between the movement changes is useful for trajectory. The goal was to keep the robot moving in a straight path while correcting its trajectory due to friction and wheel problems.The ultrasonic HC-SR04 is crucial for the two tasks for object detection by threshold distance.

2 COMPONENT LIST & PURCHASED ITEMS

The list of the components used for the mechatronics system are listed in Table 1 (See Appendix).

3 OUTSOURCED DESIGN/LIBRARY/SOFTWARE MATERIALS

Outsourced soft content used in mechanical system design are listed in table 2. (See Appendix)

4 MECHANICAL DESIGN, MATERIALS, AND FABRICATION METHOD

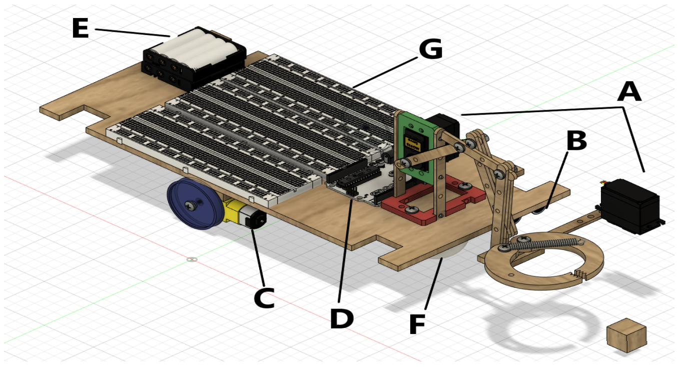

Fig. 1. CAD image of Grippy-bot

The focus of this robotic performance capabilities is detailed in the functional specifications below.

1. -Move from one point to another point on the ground at a two-meter distance.
2. -Avoid obstacles in front of it and perform maneuvering protocols.
3. -Pick up and place back down a 2x2x2 cm box.

To achieve these tasks, varying physical and mechanical components are implemented in the fabrication.

There are numerous kinematic configurations for mobile robots. A popular design is the differential drive system. The grippy-bot implements two-wheel differential drive with passive support wheels, actuators and sensors. Mini direct current (DC) motors with encoders at the left and right of the system, along the center. As stated earlier a For passive support, a ping pong ball at the front centre stabilises the system. Encoder counts revolutions made by the motor, converting the mechanical motion into electric signal in closed loop feedback. This is monitored and allows alterations to optimise speed and torque control (Maung, Latt and Nwe, 2018).The wheels permit motion along two degrees of freedom on y-axis.

Obstacle avoidance achieved with the chosen steering mechanism. The robotic configuration for this robot used two wheels with use of differential drive since the robot is small means each wheel has its own actuator allowing them to move forward and backward independently and turn the entire system on the spot. Ultrasonic sensor helps achieve these functions. It is placed in the front of the system and can detect moving and stationary obstacles. It works using ultrasonic sound waves to measure proximity to an obstacle. A transducer sends ultrasonic pulses that reflect off obstacles, it collects this echo and relays the distance by measuring time between sending and receiving pulse. (Burnett, 2020)). As the system must move in a rectangular workspace and avoid one object these avoidance measures are implemented.

To achieve the last specification, a gripper mechanism was installed using two continuous rotation RC) servo motors as seen in figure 1. The gripper arm to lift and drop the box and gripper above ground level along the z-axis with the first motor. The gripper mechanism relies on the principle of Hooke's law elasticity with a spring to grasp the box. The servo motor applies load to the spring to stretch and open the gripper and release the load returning to its original state, hence closed gripper. MG996R servo motor has a small torque of approximately 11kglcm and 180-degree rotation making it suitable for gripper as the box is light and small (Components101, 2021).

5 ELECTRONICS DESIGN | Exercise 9

In Figure 2 circuit diagram, the Arduino Uno is connected to two MG996R servo motors, two DFRobot micro dc motor, one ultrasonic distance sensor HC-SR04, TB6612FNG motor driver and 6 (1.5V) NiMH batteries. The MG996R servo motor has three wires; the black wires are connected to the ground, the red wires to the 5V pin of the Arduino and the orange wires are connected to pin 2 and 12, respectively. The DFRobot DC motors are connected to the Arduino through the motor driver where the black wires are connected to A02 and B02, respectively, and the red wires are connected to A01 and B01, respectively. The Ultrasonic sensor is connected; the blue and orange are connected to the ground and 5V of the Arduino, and the echo(green) and trig(yellow) are connected to pin 9 and 10 of the Arduino, respectively. The motor driver vm is connected to the battery's positive terminal, and the ground is connected to the battery's negative terminal. The PWMA, AI2,AI1,BI1,BI2,PWMB is connected to pin 3,pin 5,pin 4,pin 8,pin 7,pin 6 of the Arduino respectively and STBY is connected to the positive terminal.

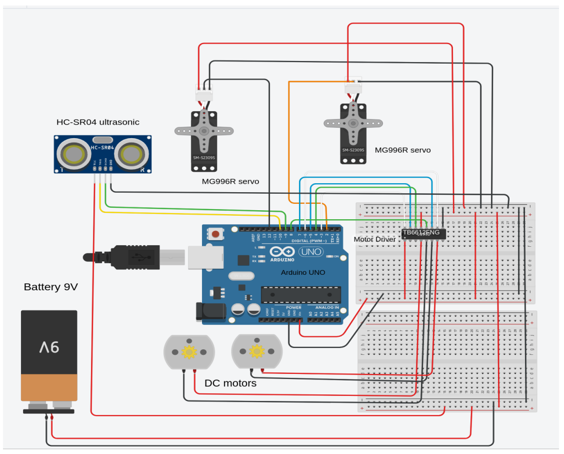

Fig. 2 Schematic representation of the electronic components in the system.

6 BEHAVIOUR (MOTION) DESIGN

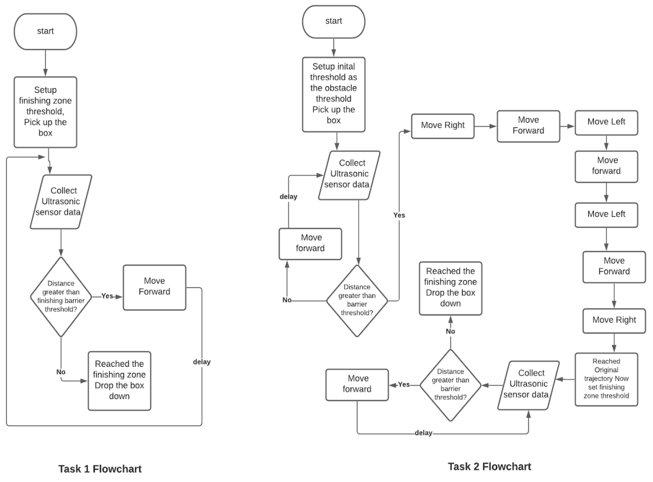
start
Setup finishing zone threshold, Pick up the box
Collect Ultrasonic sensor data
Distance greater than finishing barrier threshold?
Yes
Move Forward
delay
No
Reached the finishing zone Drop the box down
Task 1 Flowchart
start
Setup inital threshold as the obstacle threshold Pick up the box
Collect Ultrasonic sensor data
delay
Move forward
No
Distance greater than barrier threshold?
Yes
Move Right
Move Forward
Move Left
Move forward
Move Left
Move Forward
Move Right
Reached Original trajectory Now set finishing zone threshold
Collect Ultrasonic sensor data
Distance greater than barrier threshold?
No
Reached the finishing zone Drop the box down
Yes
Move forward
delay
Task 2 Flowchart

Fig. 3. Flowchart of the operation logic

To carry out the planned tasks 1 and 2, the logical decisions the robot would make are considered. For each task, the system is required to make a 2cm box from point A to B down. As well, the system must move within a rectangular workspace without hitting obstacles. The navigation method relies on sensors by forward kinematics. Figure 3 shows the behavior algorithm the system follows to complete each task.

The system has object perception, decision and action abilities from ultrasonic sensors. For motor control, there exists high-level/speed control which enables basic system movements. Low-level/speed control which connects obstacle perception and action. A distinction is made in the system program (See Appendix B) with two set proximity reading thresholds resulting in different adaptive speeds.

For trajectory generation the ultrasonic sensor proximity data thresholds determine the acceleration of each wheel and gripper behavior.

At the start of tasks, the gripper picks up the 2cm box by moving the arm along the y-axis through z rotation towards the ground and the gripper opens. At the distance threshold which perceives the obstacle placed at distance before the 2-meter mark. Along with the motor control action, the gripper and arm also move along their degrees of freedom to place the box on the ground.

Due to direct differential drive, the wheels are actuated at the same angular velocity (v) when moving forward.

$$VLeft = Vright \tag{1}$$

For motion causing left and right turns of the system, velocities (v) can be varied. See equations 2, 3 and 4. (Hellström, 2011)

$$w(R + lenght/2) = Vright \tag{2}$$

$$w(R + lenght/2) = Vleft \tag{3}$$

$$R = \frac{length}{2}\frac{Vleft+Vright}{Vright-Vleft};\frac{Vright-Vleft}{length} \tag{4}$$

Both wheels' rate of rotation (w) about the system's instantaneous center of curvature must be identical. Line 37 and 130 show the execution of these motions and thresholds in robot programs.

The links and joints of the system are focused on the gripper mechanism. As shown in figure 1, other components are rigid joints with the exception at the servo motor joints and gripper. These joints are set at 40, 40 and -20 degrees from the arm to gripper to keep the gripper above the entire system.

7. TASK 1 Performance Result

For task 1, the finish line has an obstacle which marks the finishing position given by a threshold distance. When the distance in the ultrasonic HC-SR04 sensor is below the threshold distance, it marks the arrival of the finish zone and the robot stops and performs an object dropping mechanism on the target.

8. TASK 2 Performance Result

Same logic is applied in task 2 while avoiding the obstacle when a secondary obstacle threshold has been set to avoid any first occurring frontal barrier. The robot starts moving right-left-front-left-right to prevent collision with the barrier and fix trajectory when the distance in the sensor is given less than the secondary obstacle threshold.

9. DISCUSSIONS AND PROJECT REFLECTIONS

From evaluation of mechatronics setup of the system, the trajectory controllers for motion, avoidance, task completion and implemented programming,tested to validate the algorithm and accomplish avoidance maneuvers.

This shows the complete design and construction of robots' controllability analysis, kinematic model and steering mechanism integration. The system programming achieved the general pre-determined tasks. The design can also be adapted for larger mobile robot applications. This can be adapted to recovery robots. The Advantages of the system relate to chosen material choices; plywood is light, strong. Mini DC motors have high efficiency, low price, speed and variably control. Servo motors have low torque run on low voltage, light and good rotational freedom.

Currently limitations of the system are motion. When moving with load the system would deviate from the path from tyre slip or mechanical inconsistency. Second passive support placed at back and switch to mecanum wheels. This should balance out the center of gravity moving around the system with load due to the chosen steering mechanism. Also, speed control as the system experienced jerking during motion hence, the rate of acceleration would be reduced to allow smooth motion along the path.

During the design stage we experienced contrasts of theoretical ideas against practical application. When making gripper angles and shapes played a vital role as we ended up with four varied built designs before final application as we found using the different materials and production processes affected our task execution.

Appendix A

Table 1: Mechanical and electronic components and material list

component name/model	count	weight	current/power consumption	total price excl. VAT	link	Labels in Fig.1
MG996R servo motor	2	110g	5A (2.5A /motor)	(included in the kit)	-	A
Ultrasonic distance sensor (HC-SR04 5V Version)	2	8.5g	15mA	£7.76	46130 | Ultrasonic Distance Sensor HC-SR04 5V Version | RS Components (rs-online.com)	B
9mm thick(300X 300) Plywood	1	ignorable	N/A	£1.50	Materials' Prices Troubleshooting – iForge (iforgesheffield.org)	not labelled
Solid core connecting wires		ignorable	N/A	(included in the kit)	-	not labelled
wheels	2	ignorable	N/A	(included in the kit)	-	not labelled
DFRobot Micro DC Motor with Encoder	2	100g	5.6A	(included in the kit)	-	C
Bolts/nuts/screws/washers/adhesives/tension spring)		ignorable	N/A	(included in the kit)	-	not labelled
Arduino UNO	1	25g	2A	(included in the kit)	-	D
Standard AA size NiMH battery (max discharge current XXA) (LiPo&Li-ion prohibited)	10	353.82g	N/A	£10		E
Battery holder	2	ignorable	N/A	(included in the kit)	-	not labelled
Ping pong ball	1	ignorable	N/A	lab stock		F
TB6612FNG Motor Driver Board	1	ignorable	1.2A	(included in the kit)	-	not shown
Plastic Printed holder for MG996R servo motor	2	ignorable	N/A	(included in the kit)	-	not labelled
Plastic snap rivets 4 X 4mm	4	ignorable	N/A	(included in the kit)	-	not labelled
Breadboard / Protoboard	3	141g	N/A	(included in the kit)	-	G
Additional Cost						
Steel Dowel 300 X 300 (1.0mm)	1	ignorable	N/A	£1.75	Materials' Prices Troubleshooting – iForge (iforgesheffield.org)	not labelled
	Total	738.32g	13.815A (excl. batteries)	£21.01		

Table 2: Outsourced design/library/software materials

Material name	Description	Link
Fusion 360 McMaster-Carr software library	Software library which has mechanical parts such as screws, bolts, springs, etc.	https://www.autodesk.com/products/fusion-360/free-trial https://www.nyccnc.com/import-mcmaster-carr-cad-file-fusion-360/

CAD design of 4X AA Battery Holder	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/4x-aaa-alkaline-battery-holder-
CAD design of AA NiMH batteries	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/samsung-inr21700-48g-4-800-mah-li-ion-rechargeable-battery-1
CAD design of Arduino_Uno	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/arduino-uno--2#
CAD design of Breadboard	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/breadboard-17
CAD design of Ultrasonic sensor	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/ultrasonic-sensor-hc-sr04-3
CAD design of MG995 Servo Motor	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/mg995-servo-motor-2
CAD design of DFRobot Micro DC Motor with Encode	Open-source CAD design from GrabCAD Community	https://grabcad.com/library/tt-motor-1

Appendix B

Gripper-bots executed program

```
#include <Servo.h>

// wheels
#define PWM1 3
#define AIN1 4
#define AIN2 5
#define PWM2 6
#define BIN1 7
#define BIN2 8

  // distance sensor

  #define echoPin 9 // attach pin D2 Arduino to pin Echo of HC-SR04
  #define trigPin 10 //attach pin D3 Arduino to pin Trig of HC-SR04

  // defines variables
  long duration; // variable for the duration of sound wave travel
  int distance; // variable for the distance measurement



  // define servo objects

  Servo handMotor;
  Servo gripMotor;

  int handpos,grippos;

  int pot;
  int out;

  // time variables
  int start, current;

  // stop threshold

  float threshold = 30;
```

```

//task id

int task = 2;

void setup() {
  Serial.begin(9600);
  pinMode(PWM1,OUTPUT);
  pinMode(AIN1,OUTPUT);
  pinMode(AIN2,OUTPUT);
  pinMode(PWM2,OUTPUT);
  pinMode(BIN1,OUTPUT);
  pinMode(BIN2,OUTPUT);

  handMotor.attach(2);
  gripMotor.attach(12);

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an OUTPUT
  pinMode(echoPin, INPUT); // Sets the echoPin as an INPUT

}

void loop() {
  initialize();
  if(task==1){
    grip_open();
    delay(100);
    hand_push();
    delay(200);
    grip_close();
    delay(100);
    hand_pull();
    delay(500);
//    while(1){}
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculating the distance
    distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
    // Displays the distance on the Serial Monitor
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    while(distance>threshold){
      drive(300);
      delay(750);
      stop_drive();
      turn_right(98);
      delay(60);
      // Clears the trigPin condition
//      digitalWrite(trigPin, LOW);
//      delayMicroseconds(2);
//      // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
//      digitalWrite(trigPin, HIGH);
//      delayMicroseconds(10);
//      digitalWrite(trigPin, LOW);
//      // Reads the echoPin, returns the sound wave travel time in microseconds
//      duration = pulseIn(echoPin, HIGH);
//      // Calculating the distance
//      distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
      // Displays the distance on the Serial Monitor
      Serial.print("Distance: ");
      Serial.print(distance);
      Serial.println(" cm");
      current = millis();
    }
    stop_drive();
    delay(100);
    hand_push();
    delay(100);
    grip_open();
    delay(100);
    hand_pull();
    task =0;
  }

  else if(task==2){
    int counter = 0;
    grip_open();
    delay(100);
    hand_push();
    delay(200);
    grip_close();
    delay(100);
    hand_pull();
    delay(100);
//    while(1){}
    threshold = 40;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculating the distance
    distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
    // Displays the distance on the Serial Monitor
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    while(distance>threshold){
      drive(400);
      delay(500);
      if (counter<2){
        counter +=1;

      }
      else{
        turn_right(30);
        delay(30);
        counter = 0;
      }

      Serial.print("Distance: ");
      Serial.print(distance);
      Serial.println(" cm");
      current = millis();
    }
    stop_drive();
//    while(1){}
    threshold = 20;
    turn_right(400);
    delay(2000);
    stop_drive();
    drive(400);
    delay(2000);
    stop_drive();
    turn_left(400);
    delay(2000);
    stop_drive();
    drive(500);
    delay(3700);
    stop_drive();
    turn_left(400);
    delay(2100);
    stop_drive();
    drive(400);
    delay(1950);
    turn_right(400);
    delay(2150);
    stop_drive();
//    while(1){}
    threshold = 30;
    stop_drive();
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration = pulseIn(echoPin, HIGH);
    // Calculating the distance
    distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
    // Displays the distance on the Serial Monitor
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    while(distance>threshold){
      drive(300);
      delay(800);
//      turn_left(50);
//      delay(40);
      turn_right(30);
      delay(20);
      Serial.print("Distance: ");
      Serial.print(distance);
      Serial.println(" cm");
      current = millis();
    }
    stop_drive();
    delay(100);
    hand_push();
    delay(100);
    grip_open();
    delay(100);
    hand_pull();
  }



  while(1){}
  exit(1);

}

void initialize(){
  digitalWrite(AIN1,LOW); //Motor A Rotate Clockwise
  digitalWrite(AIN2,HIGH);
  digitalWrite(BIN1,LOW); //Motor B Rotate Clockwise
  digitalWrite(BIN2,HIGH);
  handMotor.write(10);
  gripMotor.write(0);

}

void drive(int speed_motor){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  if(distance>20){
    out = map(speed_motor,0,1023,0,255);
    analogWrite(PWM1,out); //Speed control of Motor A
    analogWrite(PWM2,out); //Speed control of Motor B
  }
  else{
    stop_drive();
  }
}

void turn_right(int speed_motor){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  if(distance>20){
    out = map(speed_motor,0,1023,0,255);
    analogWrite(PWM1,out); //Speed control of Motor A
    analogWrite(PWM2,0); //Speed control of Motor B
  }
  else{
    stop_drive();
  }
}

void turn_left(int speed_motor){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(echoPin, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  if(distance>20){
    out = map(speed_motor,0,1023,0,255);
    analogWrite(PWM1,0); //Speed control of Motor A
    analogWrite(PWM2,out); //Speed control of Motor B
  }
  else{
    stop_drive();
  }
}
```

```

void stop_drive(){
  analogWrite(PWM1,0); //Speed control of Motor A
  analogWrite(PWM2,0); //Speed control of Motor B

}

void hand_push(){
  for (handpos = 10; handpos <= 55; handpos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    handMotor.write(handpos);              // tell servo to go to position in variable 'pos'
    delay(50);                       // waits 15ms for the servo to reach the position
  }
}

void hand_pull(){
  for (handpos = 55; handpos >= 10; handpos -= 1) { // goes from 180 degrees to 0 degrees
    handMotor.write(handpos);              // tell servo to go to position in variable 'pos'
    delay(50);                       // waits 15ms for the servo to reach the position
  }
}

void grip_open(){
  for (grippos = 0; grippos <= 130; grippos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    gripMotor.write(grippos);              // tell servo to go to position in variable 'pos'
    delay(50);                       // waits 15ms for the servo to reach the position
  }
}

void grip_close(){
  for (grippos = 130; grippos >= 0; grippos -= 1) { // goes from 180 degrees to 0 degrees
    gripMotor.write(grippos);              // tell servo to go to position in variable 'pos'
    delay(50);                       // waits 15ms for the servo to reach the position
  }
}
```

REFERENCES

[1] Maung M., Latt M. M. amd, Nwe C. M., (2018) DC Motor Angular Position Control using PID Controller with Friction Compensation, International Journal of Scientific and Research Publications. Available at:

https://www.researchgate.net/publication/329479537_DC_Motor_Angular_Position_Control_using_PID_Controller_with_Friction_Compensation (Accessed 11 December 2021).

[2] Components101 (2019) MG996R Servo Motor. Available at: MG996R Servo Motor Datasheet, Wiring Diagram & Features (components101.com) (Accessed 13 December 2021).

[3] Burnett R. (2020) Understanding how Ultrasonic Sensors Work. MaxBotix. Available at:Understanding How Ultrasonic Sensors Work | (maxbotix.com) (Accessed 14 December 2021).

[4] Hellström, T. (2011) Kinematics Equations for Differential Drive and Articulated Steering. Department of Computing Science, Umeå University. Available at: Microsoft Word - Kinematics Equations for Differential Drive and Articulated Steering UMINF-11.21.doc (umu.se) (Accessed 19 December 2021).