X.1 Integer drive layout scheme

When design drive layout scheme, following problems should be considered:

1. The vehicle needs two drive motors. One is used to move, the other is used to drive the four bar linkage.
2. To equip the vehicle with turn function, a servo motor is needed.

Scheme: as shown in figure 1, one stepper motor is connected to the linkage axis, which will drive the four bar linkage; the other stepper motor is connected to the wheel axis, which will drive the wheel to move. The servo motor will make the two parts angled, which will make the vehicle turn.

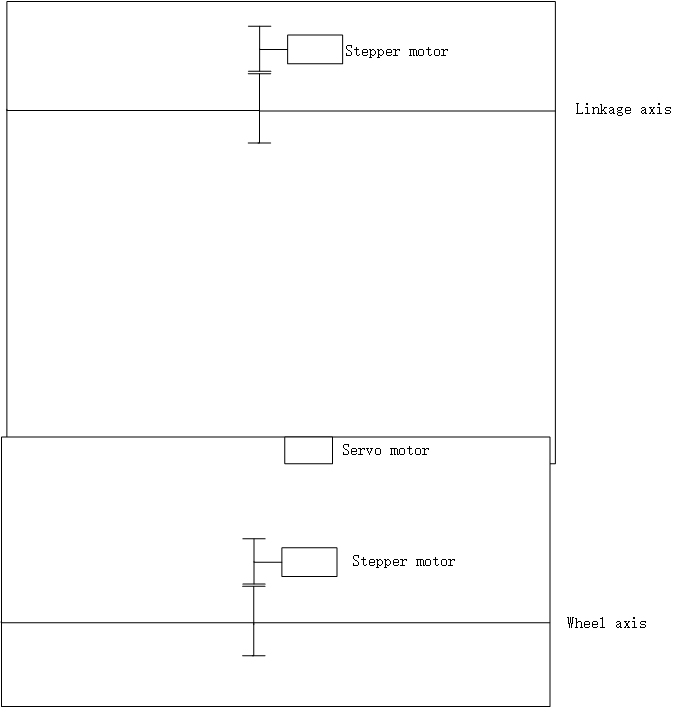


Figure 1 Integer driver layout scheme

X.2 choose of motors

Motor must have following features:

1. Controllability. Drive motor is used to transform control signal to mechanical motion, so controllability is very important.
2. High accuracy. To meet requirements of motion, the motor must have high accuracy.
3. Reliability. The reliability of motor is important.

Stepper motor is a motor controlled by a series of electromagnetic coils. The center shaft has a series of magnets mounted on it, and the coils surrounding the shaft are alternately given current or not, creating magnetic fields which repulse or attract the magnets on the shaft, causing the motor to rotate.

The unipolar stepper motor 28-BYJ48 meets requirements. 28-BYJ48 has five or six wires and four coils (actually two coils divided by center connections on each coil). The center connections of the coils are tied together and used as the power connection. They are called unipolar steppers because power always comes in on this one pole.

Specification for this Motor  
  
 Rated voltage ： 5VDC  
 Number of Phase 4  
 Speed Variation Ratio 1/64  
 Stride Angle 5.625° /64  
 Frequency 100Hz  
 DC resistance 50Ω±7%(25℃)  
 Idle In-traction Frequency > 600Hz  
 Idle Out-traction Frequency > 1000Hz  
 In-traction Torque >34.3mN.m(120Hz)  
 Self-positioning Torque >34.3mN.m  
 Friction torque 600-1200 gf.cm  
 Pull in torque 300 gf.cm

As for servo motor, sg90 is a good choice. Sg90 can rotate 180 degrees, and can provide stall torque up to 1.8 kgf·cm, which is enough for turning.

Appendixes

1. control system

as shown in figure X, the control system consists of drivers, microcontroller, Bluetooth module and battery.

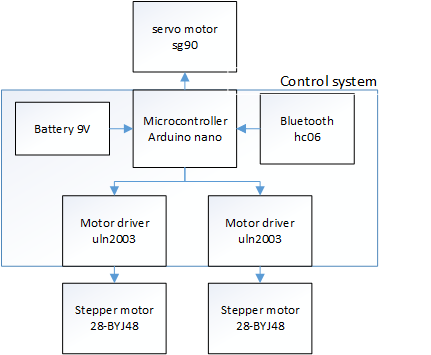


Figure x: control system

Detail connection is shown in figure X.x.

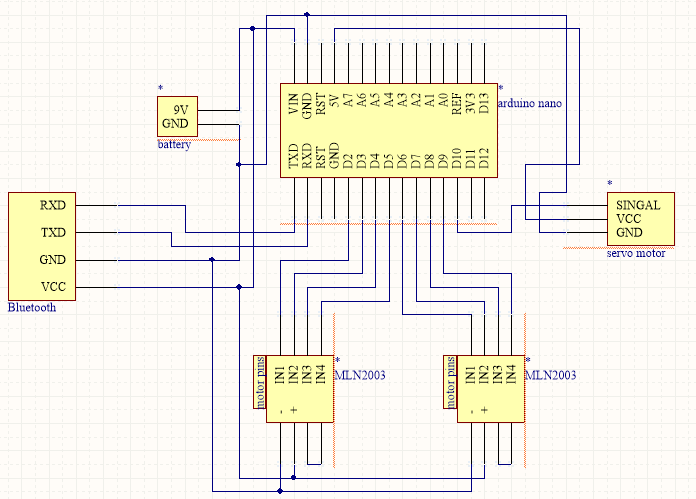


Figure X.x detail connection

1. codes for microcontroller

|  |
| --- |
| #include <AccelStepper.h>  #include <Servo.h>  #define HALFSTEP 8  #define LED\_PIN 13 // (Arduino is 13, Teensy is 11, Teensy++ is 6)  // Motor pin definitions  #define motorPin1 5 // IN1 on the ULN2003 driver 1  #define motorPin2 6 // IN2 on the ULN2003 driver 1  #define motorPin3 7 // IN3 on the ULN2003 driver 1  #define motorPin4 8 // IN4 on the ULN2003 driver 1  #define motorPin5 9 // IN1 on the ULN2003 driver 2  #define motorPin6 10 // IN2 on the ULN2003 driver 2  #define motorPin7 11 // IN3 on the ULN2003 driver 2  #define motorPin8 12 // IN4 on the ULN2003 driver 2  // Initialize with pin sequence IN1-IN3-IN2-IN4 for using the AccelStepper with 28BYJ-48  AccelStepper stepper1(HALFSTEP, motorPin1, motorPin3, motorPin2, motorPin4);  AccelStepper stepper2(HALFSTEP, motorPin5, motorPin7, motorPin6, motorPin8);  Servo myservo;  // variables  int stepperSpeed1 = 1000; //speed of move (steps per second)  int stepperSpeed2 = 1000; //speed of climb (steps per second)  //4076 steps/per full revolution  String str="";  String upBack="";  String leftRight="";  String climb="";  int angle=15;  void setup() {  delay(3000); //sime time to put the robot down after swithing it on    stepper1.setMaxSpeed(2000.0);  stepper2.setMaxSpeed(2000.0);    myservo.attach(13);  myservo.write(90);      Serial.begin(115200);  while (!Serial); // wait for Leonardo enumeration, others continue immediately  pinMode(LED\_PIN, OUTPUT);  }//--(end setup )---  void loop() {  if(Serial.available()>0)  {  str=Serial.readStringUntil('\n');  if(str=="up"||str=="back"||str=="stopMove"){  upBack=str;  }else if(str=="left"||str=="right"||str=="stopTurn"){  leftRight=str;  }else if(str=="climb"||str=="stopClimb"){  climb=str;  }  }    if(upBack=="up")  {  stepper1.setSpeed(stepperSpeed1);  stepper1.runSpeed();  }  else if(upBack=="back")  {  stepper1.setSpeed(-stepperSpeed1);  stepper1.runSpeed();  }  else if(upBack=="stopMove")  {  stepper1.stop();  }    if(leftRight=="left")  {  if(myservo.read()!=(90-angle))  {  myservo.write(90-angle);  }  }  else if(leftRight=="right")  {  if(myservo.read()!=(90+angle))  {  myservo.write(90+angle);  }  }  else if(leftRight=="stopTurn")  {  if(myservo.read()!= 90)  {  myservo.write(90);  }  }      if(climb=="climb")  {  stepper2.setSpeed(stepperSpeed2);  stepper2.runSpeed();  }  else if(climb="stopClimb")  {  stepper2.stop();  }  } |

1. android software

Android software is designed to control the vehicle remotely through Bluetooth. Its interface is shown in figure xxx.

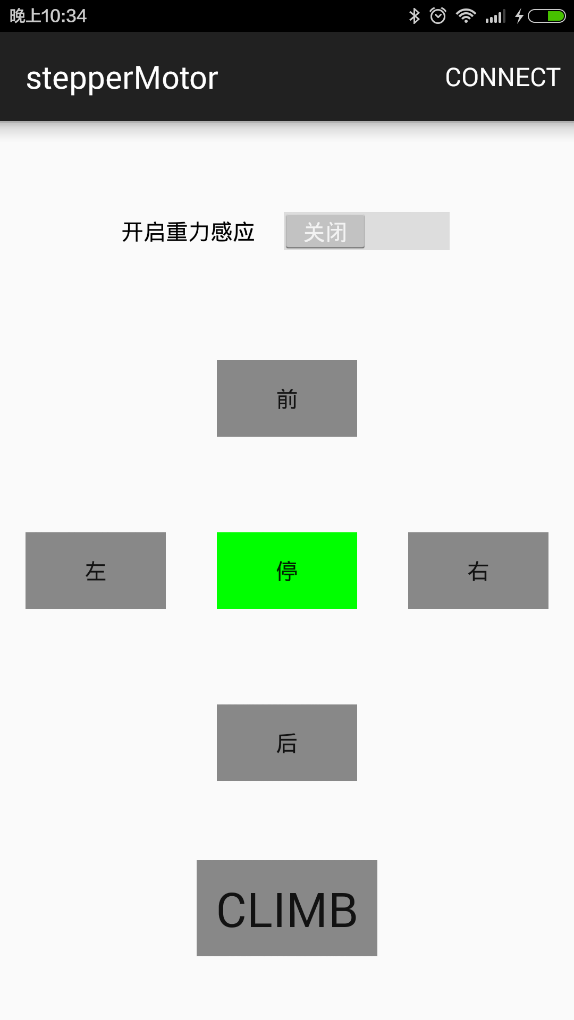


Figure xxx interface of android software

The software is built in android studio, and the main codes are as follows:

|  |
| --- |
| package com.meiqua.steppermotor;  import android.app.Activity;  import android.bluetooth.BluetoothAdapter;  import android.content.Context;  import android.content.DialogInterface;  import android.content.Intent;  import android.graphics.Color;  import android.hardware.Sensor;  import android.hardware.SensorEvent;  import android.hardware.SensorEventListener;  import android.hardware.SensorManager;  import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.util.Log;  import android.view.Menu;  import android.view.MenuItem;  import android.view.View;  import android.widget.Button;  import android.widget.Switch;  import android.widget.Toast;  import app.akexorcist.bluetotohspp.library.\*;  public class MainActivity extends AppCompatActivity implements SensorEventListener{  BluetoothSPP bt;  private SensorManager mSensorManager;  private Sensor mSensor;  private Button midButton;  private Button upButton;  private Button backButton;  private Button leftButton;  private Button rightButton;  private Button climbButton;  private Switch aSwitch;  boolean midFlag=true;  boolean upFlag=false;  boolean backFlag=false;  boolean rightFlag=false;  boolean leftFlag=false;  boolean climbFlag=false;  final float pi=3.1415926f;  final float angleFlag=0.25f;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_main);  mSensorManager=(SensorManager)getSystemService(Context.SENSOR\_SERVICE);  mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE\_GRAVITY);  bt = new BluetoothSPP(this);  //bt.setupService();  if(!bt.isBluetoothAvailable()) {  Toast.makeText(getApplicationContext()  , "Bluetooth is not available"  , Toast.LENGTH\_SHORT).show();  // finish();  }  midButton=(Button)findViewById(R.id.mid);  leftButton=(Button)findViewById(R.id.left);  rightButton=(Button)findViewById(R.id.right);  upButton=(Button)findViewById(R.id.up);  backButton=(Button)findViewById(R.id.back);  aSwitch=(Switch)findViewById(R.id.switch1);  climbButton=(Button)findViewById(R.id.climb);  updateButtonState();  climbButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if (climbFlag){  climbFlag=false;  }else {  climbFlag=true;  }  updateButtonState();  }  });  midButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  midFlag=true;  leftFlag=false;  rightFlag=false;  upFlag=false;  backFlag=false;  updateButtonState();  }  });  leftButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  leftFlag=true;  rightFlag=false;  midFlag=false;  updateButtonState();  }  });  rightButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  rightFlag=true;  leftFlag=false;  midFlag=false;  updateButtonState();  }  });  upButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  upFlag=true;  backFlag=false;  midFlag=false;  updateButtonState();  }  });  backButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  backFlag=true;  upFlag=false;  midFlag=false;  updateButtonState();  }  });  }  public void onStart() {  super.onStart();  if (!bt.isBluetoothEnabled()) {  Intent intent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE);  startActivityForResult(intent, BluetoothState.REQUEST\_ENABLE\_BT);  } else {  if(!bt.isServiceAvailable()) {  bt.setupService();  bt.startService(BluetoothState.DEVICE\_OTHER);  // setup();  }  }  }  @Override  public boolean onCreateOptionsMenu(Menu menu) {  // Inflate the menu; this adds items to the action bar if it is present.  getMenuInflater().inflate(R.menu.menu\_main, menu);  return true;  }  @Override  public boolean onOptionsItemSelected(MenuItem item) {  // Handle action bar item clicks here. The action bar will  // automatically handle clicks on the Home/Up button, so long  // as you specify a parent activity in AndroidManifest.xml.  int id = item.getItemId();  //noinspection SimplifiableIfStatement  if (id == R.id.connect) {  if (item.getTitle().equals("disconnect")) {  bt.disconnect();  bt.send("stop", false);  item.setTitle("connect");  } else {  bt.setDeviceTarget(BluetoothState.DEVICE\_OTHER);  Intent intent = new Intent(getApplicationContext(), DeviceList.class);  startActivityForResult(intent, BluetoothState.REQUEST\_CONNECT\_DEVICE);  item.setTitle("disconnect");  }  }  return super.onOptionsItemSelected(item);  }  public void onActivityResult(int requestCode, int resultCode, Intent data) {  if (requestCode == BluetoothState.REQUEST\_CONNECT\_DEVICE) {  if (resultCode == Activity.RESULT\_OK)  bt.connect(data);  } else if (requestCode == BluetoothState.REQUEST\_ENABLE\_BT) {  if (resultCode == Activity.RESULT\_OK) {  bt.setupService();  bt.startService(BluetoothState.DEVICE\_ANDROID);  // setup();  } else {  // Do something if user doesn't choose any device (Pressed back)  Toast.makeText(getApplicationContext()  , "Bluetooth was not enabled."  , Toast.LENGTH\_SHORT).show();  // finish();  }  }  }  @Override  protected void onStop() {  mSensorManager.unregisterListener(this);  super.onStop();  }  @Override  protected void onPause() {  mSensorManager.unregisterListener(this);  super.onPause();  }  @Override  protected void onResume() {  mSensor=mSensorManager.getDefaultSensor(Sensor.TYPE\_GRAVITY);  mSensorManager.registerListener(this,mSensorManager.getDefaultSensor(Sensor.TYPE\_GRAVITY)  ,SensorManager.SENSOR\_DELAY\_NORMAL);  super.onResume();  }  @Override  public void onSensorChanged(SensorEvent event) {  if (aSwitch.isChecked()){  if (event.sensor.getType() == Sensor.TYPE\_GRAVITY) {  updateButton(event.values[0],event.values[1],event.values[2]);  }  }  }  @Override  public void onAccuracyChanged(Sensor sensor, int accuracy) {  }  private void updateButton(float x,float y,float z) {  float Gyz =(float)Math.sqrt( y \* y + z \* z);  float Gzx = (float)Math.sqrt(x \* x + z \* z);  float tanX = 0;  float tanY = 0;  if (Gzx == 0) {  if (y > 0)  tanY = pi / 2;  else  tanY = -pi / 2;  } else  tanX = x / Gzx;  if (Gyz == 0) {  if (x > 0)  tanX = pi / 2;  else  tanX = -pi / 2;  } else  tanY = y / Gyz;  Log.i("-->", "updateButton tanx: "+tanX);  Log.i("-->", "updateButton tany: "+tanY);  Log.i("-Gyz->", "updateButton Gyz: "+Gyz);  if (tanY >= angleFlag) {  upFlag = false;  backFlag = true;  } else if (tanY < -angleFlag) {  backFlag = false;  upFlag = true;  } else {  upFlag = false;  backFlag = false;  }  if (tanX >= angleFlag) {  leftFlag = true;  rightFlag = false;  } else if (tanX < -angleFlag) {  rightFlag = true;  leftFlag = false;  } else {  leftFlag = false;  rightFlag = false;  }  if (leftFlag == false && rightFlag == false &&  upFlag == false && backFlag == false) {  midFlag = true;  } else{  midFlag = false;  }  updateButtonState();  }  private void updateButtonState(){  if (climbFlag){  midFlag=true;  upFlag=backFlag=leftFlag=rightFlag=false;  }  if (upFlag){  upButton.setBackgroundColor(Color.GREEN);  if (bt.isServiceAvailable())  bt.send("up\n",false);  }else {  upButton.setBackgroundColor(Color.GRAY);  }  if (backFlag){  backButton.setBackgroundColor(Color.GREEN);  if (bt.isServiceAvailable())  bt.send("back\n", false);  }else {  backButton.setBackgroundColor(Color.GRAY);  }  if(backFlag==false&&upFlag==false){  if (bt.isServiceAvailable())  bt.send("stopMove\n",false);  }  if (leftFlag){  leftButton.setBackgroundColor(Color.GREEN);  if (bt.isServiceAvailable())  bt.send("left\n", false);  }else {  leftButton.setBackgroundColor(Color.GRAY);  }  if (rightFlag){  rightButton.setBackgroundColor(Color.GREEN);  if (bt.isServiceAvailable())  bt.send("right\n", false);  }else {  rightButton.setBackgroundColor(Color.GRAY);  }  if (leftFlag==false&&rightFlag==false){  if (bt.isServiceAvailable())  bt.send("stopTurn\n", false);  }  if (midFlag){  midButton.setBackgroundColor(Color.GREEN);  // if (bt.isServiceAvailable())  // bt.send("mid\n", false);  }else {  midButton.setBackgroundColor(Color.GRAY);  }  if (climbFlag){  climbButton.setBackgroundColor(Color.GREEN);  if (bt.isServiceAvailable())  bt.send("climb\n",false);  }else {  climbButton.setBackgroundColor(Color.GRAY);  if (bt.isServiceAvailable())  bt.send("stopClimb\n",false);  }  }  } |

All the codes has been open-source, which can be seen on website:

https://github.com/meiqua/stepperMotor