Material

1) arduino R3:development board(one block)
2) WisCam :video development board(one block)
3) L298N :motor drive plate(two blocks)
4) Four wheel drive car(one)
5) 12V battery(one block)

Basic principle

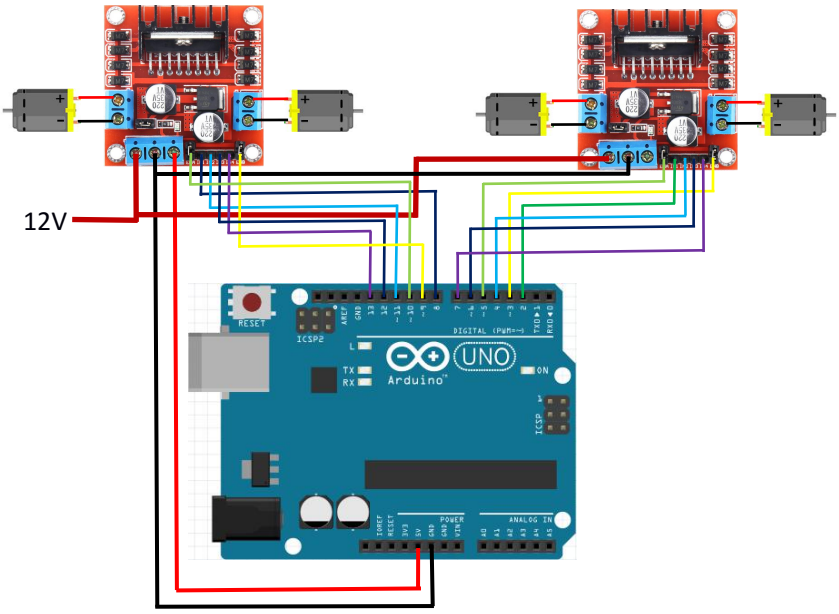
RAK WisCam is ultra-low-cost Modular Based Evaluation Kit to help the developer to design Wi-Fi video product with Linux OS and work with Arduino NUO board. WisCam can transmit video through Wi-Fi to APPs and all the source codes are available on the Github. WisCam also integrate Nabto P2P cloud to create the video connection anywhere with Internet .

WisCam supports YUV RAW data so developers can make video scaling (up to x1~x8 scaling), video cropping , video overlapping etc. or change CMOS image sensor.

WisCam work in AP mode,Upper computer connect WisCam and send control commands to the WisCam,Transmit data to serial port and connect with Arduino serial port,Finally, L298N the motor speed and positive reverse are controlled by GPIO and PWM.

Hardware connection

Each L298N contains two motor control interface, each control interface contains EN, IN1, IN2, EN for enable output, IN1-IN2 for direction control, Arduino can access EN through PWM, realize speed control.



Test code

Upper computer Software

https://github.com/galaxyofo/wiscam_car_controller.git

Download and compile then input
./controller [wisCam ip] 502
Through the keyboard controls the forward turning of the car

Arduino code

```
int forward_r1 = 2;
```

```
int forward_r2 = 4;

int forward_l1 = 6;
int forward_l2 = 7;

int forward_pwm1 = 3;
int forward_pwm2 = 5;

int back_r1 = 8;
int back_r2 = 9;
int back_l1 = 12;
int back_l2 = 13;
int back_pwm1 = 10;
int back_pwm2 = 11;

int driver_status = -1;

int stop_delay = 0;
int duration = 0;

#define MAX_SPEED 0x60
#define ACC_SPEED 2

void setup()
{
   pinMode(forward_r1, OUTPUT);
   pinMode(forward_r2, OUTPUT);
   pinMode(forward_l1, OUTPUT);
   pinMode(forward_l2, OUTPUT);

   pinMode(back_r1, OUTPUT);
   pinMode(back_r2, OUTPUT);
   pinMode(back_l1, OUTPUT);
   pinMode(back_l2, OUTPUT);
   analogWrite(forward_pwm1,0x00);
   analogWrite(forward_pwm2,0x00);
   analogWrite(back_pwm1,0x00);
   analogWrite(back_pwm2,0x00);
   Serial.begin(115200);
}

void loop() {

   if (Serial.available() > 0) {
      char cmd = Serial.read();
      if (driver_status != cmd) {
         switch (cmd) {
         case '0':
            driver_stop();
            break;
         case '1':
            driver_forward();
            duration = 600;
            break;
         case '2':
            driver_backward();
            duration = 600;
            break;
         case '3':
            driver_left();
            duration = 300;
            break;
         case '4':
            driver_right();
            duration = 300;
            break;
         default:
            break;
         }
         driver_status = cmd;
      }
      stop_delay = 0;
   } else {
   if (stop_delay++ > duration) {
      stop_delay = 0;
      driver_status = 0;
```

```
      driver_stop();
   }
   delay(1);
   }
}

void driver_backward()
{
   digitalWrite(forward_r1, HIGH);
   digitalWrite(forward_r2, LOW);
   digitalWrite(forward_l1, HIGH);
   digitalWrite(forward_l2, LOW);
   digitalWrite(back_r1, HIGH);
   digitalWrite(back_r2, LOW);
   digitalWrite(back_l1, HIGH);
   digitalWrite(back_l2, LOW);

   for (int i = 0x10; i < MAX_SPEED; i += ACC_SPEED) {
      analogWrite(forward_pwm1,i);
      analogWrite(forward_pwm2,i);
      analogWrite(back_pwm1,i);
      analogWrite(back_pwm2,i);
   }
}

void driver_forward()
{
   digitalWrite(forward_r1, LOW);
   digitalWrite(forward_r2, HIGH);
   digitalWrite(forward_l1, LOW);
   digitalWrite(forward_l2, HIGH);
   digitalWrite(back_r1, LOW);
   digitalWrite(back_r2, HIGH);
   digitalWrite(back_l1, LOW);
   digitalWrite(back_l2, HIGH);

   for (int i = 0x10; i < MAX_SPEED; i += ACC_SPEED) {
      analogWrite(forward_pwm1,i);
      analogWrite(forward_pwm2,i);
      analogWrite(back_pwm1,i);
      analogWrite(back_pwm2,i);
   }

}

void driver_stop()
{
   analogWrite(forward_pwm1,0x00);
   analogWrite(forward_pwm2,0x00);
   analogWrite(back_pwm1,0x00);
   analogWrite(back_pwm2,0x00);
}

void driver_right()
{
   analogWrite(back_pwm2, 0);
   analogWrite(forward_pwm1, 0);
   for (int i = 0X20; i < MAX_SPEED; i += ACC_SPEED) {
      analogWrite(back_pwm1, i);
      analogWrite(forward_pwm2, i);
   }
}

void driver_left()
{
   analogWrite(back_pwm1, 0);
   analogWrite(forward_pwm2, 0);
   for (int i = 0X20; i < MAX_SPEED; i += ACC_SPEED) {
      analogWrite(back_pwm2, i);
      analogWrite(forward_pwm1, i);
   }
}
```