Car tracking

The K1 button is used to control the on and off of the car's tracking (line patrol) function. The environment for debugging the car code is to follow the black line under a white background.

```
Since the previous tutorial has explained the basic knowledge of all car modules,
the car control tutorial will not be repeated, and the main focus is on the car
function implementation ideas!
```

Device connection

Hardware connection

Use Type-B data cable to connect Arduino Uno and computer.

Software connection

Open the "Arduino IDE" software and select the model and serial port number corresponding to the development board.

Implementation ideas

Use the three-way line patrol module to obtain the position of the car relative to the black route, and control the movement of the car according to this position.

Button K1: turn on or off the car tracking function;

Three-way patrol module: get the position of the car head relative to the black line;

Car control:

The three-way patrol module is located on the left side of the black line, and the car turns left;

The three-way patrol module is located in the middle of the black line, and the car goes straight;

The three-way patrol module is located on the right side of the black line, and the car turns right.

```
The actual code also analyzes the situation where the three-way patrol module
detects all black lines, and the code controls the forward movement in this
situation.
```

Code analysis

Here is only a brief introduction to the code content. For detailed code, please refer to the corresponding code file, which is provided in the download area!

- Include `Wire`、`Adafruit_PWMServoDriver` library

```
#include <Wire.h>                     // 包含Wire(I2C)通讯库 Include Wire library
#include <Adafruit_PWMServoDriver.h>  // 包含Adafruit PWMServoDriver库 Include
Adafruit PWMServoDriver library
```

- Define the motor, button, tracking control pin, I2C address, motor frequency and initial speed

```
// 定义电机控制引脚 Define motor control pins
#define Motor_L1_F_PIN 11  // 控制小车左前方电机前进 Control the motor on the left
front of the car
#define Motor_L1_B_PIN 10  // 控制小车左前方电机后退 Control the motor back on the
left front of the car
#define Motor_L2_F_PIN 8   // 控制小车左后方电机前进 Control car left rear motor
forward
#define Motor_L2_B_PIN 9   // 控制小车左后方电机后退 Control the car left rear motor
back
#define Motor_R1_F_PIN 13  // 控制小车右前方电机前进 Control the right front motor of
the car to move forward
#define Motor_R1_B_PIN 12  // 控制小车右前方电机后退 Control the motor back on the
right front of the car
#define Motor_R2_F_PIN 14  // 控制小车右后方电机前进 Control car right rear motor
forward
#define Motor_R2_B_PIN 15  // 控制小车右后方电机后退 Control car right rear motor
back

// 定义底层驱动芯片参数 Bottom-layer driver chip related parameters
#define Bottom_Layer_Driver_ADDR 0x40

// 定义PWM频率 Define PWM frequency
#define PWM_FREQUENCY 50

// 定义按键引脚和控制状态 Define pin and key(button) states
#define KEY_PIN 7
#define Press_KEY 0
#define Release_KEY 1

// 定义三路循迹模块引脚 Define the three-way line patrol module pins
#define L_TRACK_PIN A2
#define M_TRACK_PIN A1
#define R_TRACK_PIN A0

// 定义巡线阈值 Define the patrol threshold
const int Threshold = 500;

bool bCar_Switch = false;

int iCarSpeed = 50;
unsigned int uTimeOut = 0;
```

- Enumerate the common movement modes of omnidirectional cars

```
// 枚举全向小车的常见运动方式 Enumerate the common movement modes of omnidirectional cars
enum OmniDirectionalCar {
  STOP,
  FORWARD,
  BACKWARD,
  LEFT,
  RIGHT,
  LEFT_ROTATE,
  RIGHT_ROTATE,
  LEFT_FORWARD,
  RIGHT_BACKWARD,
  RIGHT_FORWARD,
  LEFT_BACKWARD,
};
```

- Create an instance of the Adafruit_PWMServoDriver class

```
// 创建Adafruit_PWMServoDriver类的实例 Create an instance of the Adafruit_PWMServoDriver class
Adafruit_PWMServoDriver pwm = Adafruit_PWMServoDriver(Bottom_Layer_Driver_ADDR);
```

- Setting the Motor Speed

```
/**
 * @brief 设置单个电机速度 Setting the Motor Speed
 * @param motor_forward_pin: 控制电机前进引脚 Control the motor forward pin
 * @param motor_backward_pin: 控制电机后退引脚 Control the motor backward pin
 * @param motor_speed: 设置电机速度 Setting the Motor Speed
 * @retval 无 None
 */
void setMotorSpeed(uint16_t motor_forward_pin, uint16_t motor_backward_pin, int motor_speed) {
  motor_speed = map(motor_speed, -255, 255, -4095, 4095);
  if (motor_speed >= 0) {
    pwm.setPWM(motor_forward_pin, 0, motor_speed);
    pwm.setPWM(motor_backward_pin, 0, 0);
  } else if (motor_speed < 0) {
    pwm.setPWM(motor_forward_pin, 0, 0);
    pwm.setPWM(motor_backward_pin, 0, -(motor_speed));
  }
}
```

- Set the car movement mode and speed

```
/**
 * @brief 设置小车运动方式和速度 Set the car movement mode and speed
 * @param Movement: 小车运动方式 Car movement
 * @param Speed: 小车运动速度 Car speed
 * @retval 无 None
 */
```

```
void setCarMove(uint8_t Movement, int Speed) {
  switch (Movement) {
    case STOP:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case FORWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case BACKWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case LEFT:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case RIGHT:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case LEFT_ROTATE:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case RIGHT_ROTATE:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    case LEFT_FORWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case RIGHT_BACKWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, -Speed);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, -Speed);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
    case RIGHT_FORWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, Speed);
      break;
    case LEFT_BACKWARD:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, -Speed);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, -Speed);
      break;
    default:
      setMotorSpeed(Motor_L1_F_PIN, Motor_L1_B_PIN, 0);
      setMotorSpeed(Motor_L2_F_PIN, Motor_L2_B_PIN, 0);
      setMotorSpeed(Motor_R1_F_PIN, Motor_R1_B_PIN, 0);
      setMotorSpeed(Motor_R2_F_PIN, Motor_R2_B_PIN, 0);
      break;
  }
}
```

- Get key(button) status

```
/**
 * @brief 获取按键状态 Get key(button) status
 * @param pin: 按键控制引脚 Control key(button) pins
 * @retval 按键状态 Key(button) Status
 */
int getKeyState(uint8_t pin) {
  if (digitalRead(pin) == LOW) {
    delay(20);
    if (digitalRead(pin) == LOW) {
      while (digitalRead(pin) == LOW)
        ;
      return Press_KEY;
    }
    return Release_KEY;
  } else {
    return Release_KEY;
  }
}
```

- Car line patrol function

```
/**
 * @brief 小车巡线功能 Car line patrol function
 * @param iLeftPin: 左边循迹传感器 Left tracking sensor
 * @param iMidPin: 中间循迹传感器 Middle tracking sensor
 * @param iRightPin: 右边循迹传感器 Right tracking sensor
 * @retval 无 None
 */
void PatrolCar(int iLeftPin, int iMidPin, int iRightPin) {
  int leftValue = analogRead(iLeftPin);
  int middleValue = analogRead(iMidPin);
  int rightValue = analogRead(iRightPin);

  // 进行简单的方向判断打印 The following is a simple judgment
  if ((leftValue < Threshold && middleValue > Threshold && rightValue <
Threshold) | (leftValue > Threshold && middleValue > Threshold && rightValue >
Threshold)) {
    setCarMove(FORWARD, iCarSpeed);
  } else if (leftValue > Threshold && middleValue < Threshold && rightValue <
Threshold) {
    setCarMove(LEFT_ROTATE, iCarSpeed);
  } else if (leftValue < Threshold && middleValue < Threshold && rightValue >
Threshold) {
    setCarMove(RIGHT_ROTATE, iCarSpeed);
  } else if (leftValue < Threshold && middleValue < Threshold && rightValue <
Threshold) {
    uTimeOut++;
    delay(20);
    if (uTimeOut > 100) {
      uTimeOut = 0;
      setCarMove(STOP, 0);
    }
  }
}
```

- Set the car function switch

```
/**
 * @brief 设置小车功能开关 Set the car function switch
 * @param 无 None
 * @retval 开启/关闭 true/false
 */
bool setCarSwitch() {
  if (getKeyState(KEY_PIN) == Press_KEY) {
    bCar_Switch = !bCar_Switch;
  }
  return bCar_Switch;
}
```

- Initialization Code

```
void setup() {
  Wire.begin();                   // 初始化I2C通讯 Initialize I2C communication
  delay(1000);                    // 如果小车功能异常，可以增加这个延时 If the function
is abnormal, you can increase the delay
  pwm.begin();                    // PWM初始化 Initialize the Pulse Width
Modulation (PWM) library
  pwm.setPWMFreq(PWM_FREQUENCY);  // 设置PWM频率 Set the PWM frequency
  setCarMove(STOP, 0);            // 设置小车停止状态 Set the car to stop state
}
```

- Looping code

```
void loop() {
  // 按键控制小车循迹功能启停 The key control car tracking function start and stop
  if (setCarSwitch()) {
    PatrolCar(L_TRACK_PIN, M_TRACK_PIN, R_TRACK_PIN);
  } else {
    setCarMove(STOP, 0);
  }
}
```

Experimental results

After compiling the program successfully, upload the code to the Arduino Uno development board.

After the program starts, press the K1 button to start and stop the car tracking (line patrol) function.

Actual use: We can put the car on the specified track route for testing (the track selects the track with a white background and a black route), press the K1 button to start the car function, and then you can observe the entire line patrol effect. If you need to stop the line patrol function, you can press the K1 button to turn off the car line patrol function.

```
The drive motor needs an external battery pack and turn on the expansion board
switch to drive normally.

The burning program cannot use other programs to occupy the serial port or the
external serial port communication module (for example: WiFi camera module),
otherwise the program cannot be burned or an error message will be prompted!
```

Notes

If the car jitters severely during the line patrol process, you can try to modify the car speed and the judgment threshold of the three-way line patrol module.

Modify speed: modify the motor speed directly in the car code;

Modify threshold: place the three-way patrol module under a black background and a white background, and record the thresholds under the two environments respectively. The threshold in the general program takes the median of the two, and users can modify and debug it by themselves.

```
We can burn the code of [1.Car peripherals foundation: 09.Track_Print.ino] to the
car, and the serial port will print the interface data of the three-way patrol
module and the direction that the car needs to change. We only need to pay
attention to the interface data of the three-way patrol module.
```

Car environment: The width of the black track is about 1.5cm, which is more suitable. In order to avoid infrared light interfering with the sensor, we need to use this module indoors.