K210-Number recognition

The tutorial mainly demonstrates the recognition of handwritten digits by the balance car combined with the K210 visual module and controls the balance car to perform specified actions according to the recognized digits.

```
The tutorial only introduces the standard library project code
```

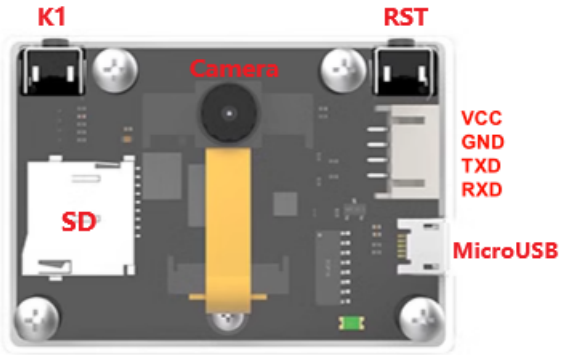
Hardware connection

Peripherals	Development board
K210 visual module: VCC	5V
K210 visual module: TXD	PA2
K210 visual module: RXD	PA3
K210 visual module: GND	GND

Control principle

Handwritten digits are recognized through the K210 visual recognition module, and the recognized digits are sent to the development board, which controls the balance car to perform specified actions according to the digits.

- K210 vision module




```
The K210 vision module itself is a development board. For detailed usage, please
refer to the module supporting tutorial
```

> **Download program and model**

- Download program

Connect the SD card of the K210 vision module to the computer through a card reader, rename the program file to main.py and copy it to the SD card, and then reinstall the SD card into the SD card slot of the K210 vision module.

```
The program filters and analyzes the numbers 1 and 5 separately. For specific
reasons, please refer to the K210 visual module supporting tutorial
The K210 visual module and development board case code are in the same folder:
the folder name will distinguish the development board to which the code belongs
```

- Download model

Put the model file used by the program in the specified location of the SD card (the sd folder refers to the root directory): /sd/KPU/mnist/uint8_mnist_cnn_model.kmodel

```
Product supporting data source code path: Attachment → Source code summary → K210
related model files
Directly unzip the K210 related model files and copy the MPU folder inside to the
SD card
```

> **Communication protocol**

The K210 visual module program will automatically recognize handwritten numbers and send the numbers to the development board for processing.

```
2, 3, and 6 are numbers with relatively high model recognition accuracy
```

Data header	Data (X represents the object category)	Data tail	Example
$	X	#	$2#: represents the recognized number is 2
$	X	#	$3#: represents the recognized number is 3
$	X	#	$6#: represents the recognized number is 6

Main code

The tutorial mainly explains the code for K210 to identify the digital control balance car function. For detailed code, please refer to the corresponding project file.

> **Deal_K210**

Receive valid data sent by K210.

```
void Deal_K210(uint8_t recv_msg)
{
    if (recv_msg == '$' && g_new_flag == 0)
    {
        g_new_flag = 1;
```

```
        memset(buf_msg, 0, sizeof(buf_msg)); // Clear old data
        return;
    }
    if(g_new_flag == 1)
    {
        if (recv_msg == '#')
        {
            g_new_flag = 0;
            g_index = 0;
            g_new_data = 1;
        }

        if (g_new_flag == 1 && recv_msg != '$')
        {
            buf_msg[g_index++] = recv_msg;

            if(g_index > 20) // Array overflow
            {
                g_index = 0;
                g_new_flag = 0;
                g_new_data = 0;
                memset(buf_msg, 0, sizeof(buf_msg)); // Clear old data
            }

        }
    }
}
```

Change_state

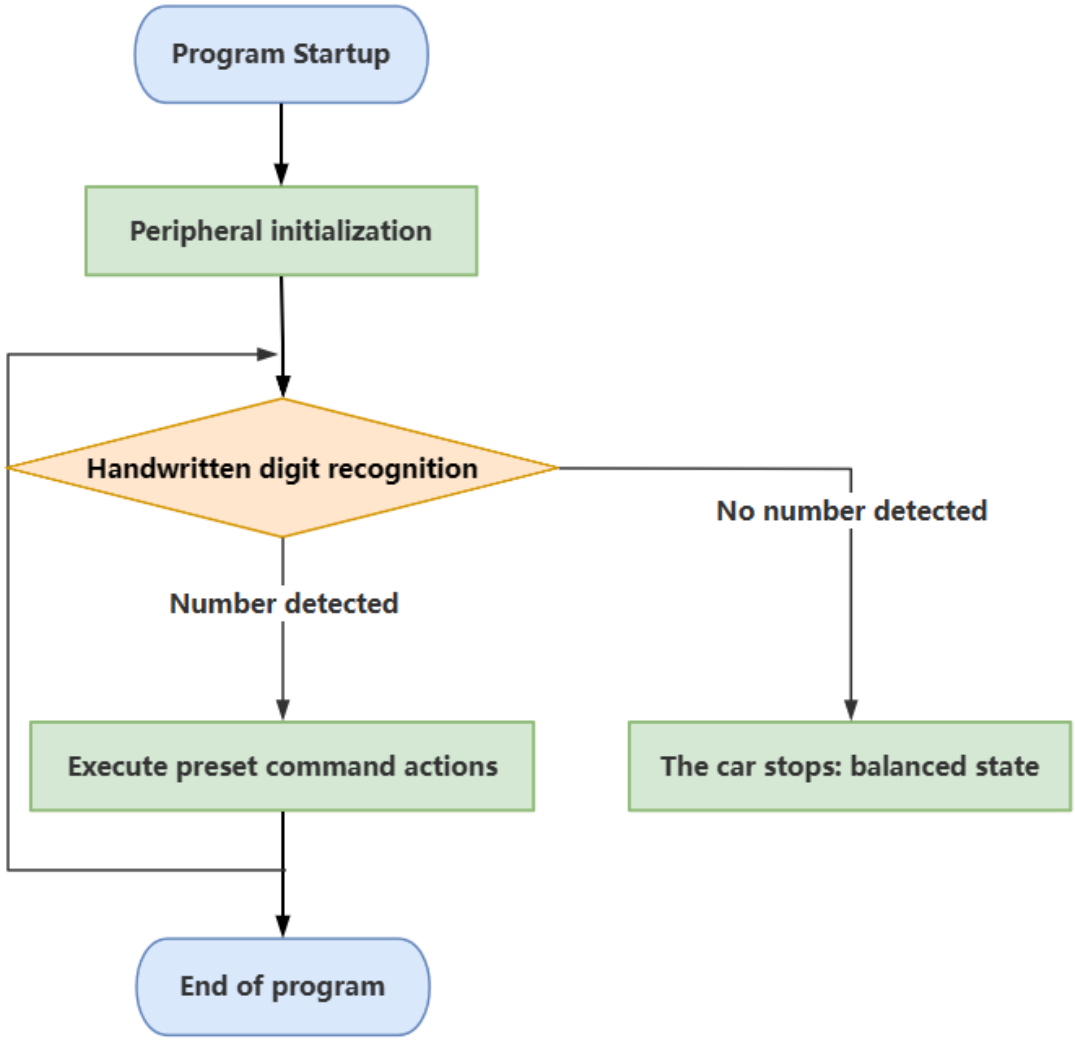
Execute different actions according to different instructions sent by K210.

```
void Change_state(void)
{
    if(g_new_data == 1)
    {
        g_new_data = 0;
        if (strcmp("6", buf_msg) == 0 )
        {
            OLED_Draw_Line("num:6!  ", 3, false, true);
            // Buzzer sounds for 1s
            BEEP_BEEP = 1;
            my_delay(1);
            BEEP_BEEP = 0;
        }
        else if (strcmp("2", buf_msg) == 0)
        {
            OLED_Draw_Line("num:2!  ", 3, false, true);
            // The car turns left for 2 seconds and then stops
            Move_Z = -Trun_speed;
            my_delay(1);
            my_delay(1);
            Move_Z = 0;
        }
        else if (strcmp("3", buf_msg) == 0 )
```

```
        {
            OLED_Draw_Line("num:3!  ", 3, false, true);
            // The car turns right for 2 seconds and then stops
            Move_Z = Trun_speed;
            my_delay(1);
            my_delay(1);
            Move_Z = 0;
        }
    }
}
```

Program flow chart

Briefly introduce the process of function implementation:



Experimental phenomenon

> **Software code**

The K210_BalancedCar_mnist.hex file generated by the project compilation is located in the OBJ folder of the K210_BalancedCar_mnist project. Find the K210_BalancedCar_mnist.hex file corresponding to the project and use the FlyMcu software to download the program to the development board.

```
The corresponding functions can only be realized after both the K210 visual
module and the development board download the program
Product supporting materials source code path: Attachment → Source code summary →
5.Balanced_Car_Extended → 11.K210_BalancedCar_mnist
```

Experimental phenomenon

After the program is started, press the KEY1 button according to the OLED prompt to start the self-learning control function of the balance car: OLED displays the program function name (Start learn num!); K210 visual recognition module recognizes the number (2, 3, 6) and performs the corresponding action; if no number is recognized, the balance car will remain balanced.

```
The program has voltage detection. If the voltage is less than 9.6V, the low
voltage alarm is triggered and the buzzer will sound.
Common situations in which the voltage alarm is triggered:
1. The power switch of the development board is not turned on, and only the Type-
C data cable is connected for power supply
2. The battery pack voltage is lower than 9.6V and needs to be charged in time
```