AiStarter User Guide

Issue：V1

Date：2018-12-26

ShenZhen Yuejiang Technology Co., Ltd.

Disclaimer

To the maximum extent permitted by applicable law, the products described (including its hardware, software and firmware, etc.) in this document are provided **AS IS**, which may have flaws, errors or faults. Yuejiang makes no warranties of any kind, express or implied, including but not limited to, merchantability, satisfaction of quality, fitness for a particular purpose and non-infringement of third party rights. In no event will Yuejiang be liable for any special, incidental, consequential or indirect damages resulting from the use of our products and documents.

Before using our product, please thoroughly read and understand the contents of this document and related technical documents that are published online, to ensure that the robotic arm is used on the premise of fully understanding the robotic arm and related knowledge. Please use this document with technical guidance from professionals. Even if follow this document or any other related instructions, Damages or losses will be happen in the using process, Dobot shall not be considered as a guarantee regarding to all security information contained in this document.

The user has the responsibility to make sure following the relevant practical laws and regulations of the country, in order that there is no significant danger in the use of the robotic arm.

ShenZhen Yuejiang Technology Co., Ltd.

Address: 3F, Building NO.3, Tongfuyu Industrial Town, Nanshan District, Shenzhen, China

Website: https://www.dobot.cc/

Preface

Purpose

This manual describes the functions of the AiStart smart car, how to use the car and its API functions to help users quickly understand and use the car.

Intended Audience

This document is intended for:

- Customer Engineer
- Sales Engineer
- Installation and Commissioning Engineer
- Technical Support Engineer

Change History

Date	Change Description
2018/12/26	Add demo and update API
2018/04/27	The first release

Symbol Conventions

The symbols that may be founded in this document are defined as follows.

Symbol	Description
⚠DANGER	Indicates a hazard with a high level of risk which, if not avoided, could result in death or serious injury
⚠WARNING	Indicates a hazard with a medium level or low level of risk which, if not avoided, could result in minor or moderate injury, robotic arm damage
⚠NOTICE	Indicates a potentially hazardous situation which, if not avoided, can result in robotic arm damage, data loss, or unanticipated result
🕮NOTE	Provides additional information to emphasize or supplement important points in the main text

Contents

1. Safety Precautions

1.1 Service Security

- Users need to assemble the AiStarter by themselves. Please do not tighten screws hard when install to avoid the treads stripping.
- When put the batteries inside, please make sure that the polarity is not reversed. Only the Li-ion rechargeable battery of 18650 type is supported.
- Please keep an eye on the AiStarter when AiStarter works, to avoid motor stalling for a long time, which could damage the control board and motor of the AiStarter.

1.2 After-sales Service Terms

1.2.1 Warranty Regulation

- The warranty period of machine body shall be 6 months.
- The battery is not included in warranty service. Only replacement of damage by default is provided.
- The battery does not enjoy the warranty service, only the factory damage replacement
- The vulnerable parts such as package, giveaway, USB, screws, wrench, structure parts, etc. shall be not covered by the warranty service. However, if there is any non-artificial performance failure at the first time using after purchase
- Please confirm the integrity of the package at the time of signing for acceptance of the goods. Within 7 days upon purchase of the goods (inclusive, calculated from the date of receipt of the goods), if the goods are found to be missing or damaged due to transportation, or a non-artificial performance failure occurs to the goods or accessories, please immediately contact the local after-sales service department for supplement or replacement; any application for supplement or replacement beyond this time limit shall be considered invalid.
- During the validity period of three guarantees (for repair, replacement or compensation of faulty products), where a product is in compliance with the replacement conditions but the seller has no product of the same time or specification, and the customer requires the return of goods as he or she is not willing to replace it with the product of a different type or specification, the customer may return the goods.
- The maintenance period of replacement parts within the warranty period is 2 months. The detective parts replaced shall be owned by DOBOT.

⚠NOTICE

The freight for any replacement or return that meets the above conditions shall be borne by DOBOT officially, while the tariff at the destination which is required to be paid due to its policy shall be borne by the customer.

To provide the free maintenance service, the following conditions shall be met:

- The product is normal used within the specified warranty period and has any non-artificial performance failure.
- The valid purchase certificate, receipt and the number thereof are provided.
- If additional package, accessories or translation are needed, our company will charge a certain cost.

The free product maintenance service shall not be provided under any of the following circumstances:

- Damage caused by unauthorized maintenance, alteration and other acts.
- Damage caused by non-standard operating environment, such as overload, high voltage, high current, high temperature, etc
- Appearance and function abnormality caused by man-made collision and dropping.
- The abnormal function caused by wet environment, soak or burn.
- Damage caused by improper use, installation and operation that are not in accordance with the official instructions.
- Damage caused by circuit redesign, improper installation of the battery, overcharge, improper installation and connection, and improper use of the charger that are not in accordance with the official instructions.
- Damage caused by reliability and compatibility problems arising from simultaneous use of third-party parts that are not certified by our company.

Spacial instructions

- **The related after-sales service is proved by the local agent**
- **If you have questions about the related after-sale service, please contact the official after-sales service department. We will help you to solve your problem without delay**

2. Introduction

2.1 Features

AiStarter is a smart car aimed to education and competition. It is designed with no-soldering assembly technique, using brass stubs to fix control board, and ribbon cables to connect each board, of which the layer is clear. The control board is designed based on Arduino Mega2560, compatible with Arduino, which is very easy to get started

Function features:

- Intelligent obstacle avoidance
- Automatic tracking
- Identify scenes based on color and perform different tasks
- Graphical programming, user can program by building blocks to control AiStarter

2.2 Parts List

Table 2.1 Parts List

Part	Number
Chassis	1
Shell	1
Control Board	1
Ultrasonic Sensor	3
Infrared Sensor	1
Color Sensor	2
DC Gear Motor (with Encoder)	2
Universal Wheel	1
Coupling	2
Tire	2
18650 Li-ion Battery	2
18650 Battery Holder	1
Brass Stud M3*32+4	4
R2048 Nylon Rivet	10
M3*5 Cross Round Head Screw	30
M3*5 Cross Countersunk Screw	4
M3*6 Cross Round Head Screw	6

Part	Number
M4*6 Cross Round Head Screw	3
4PIN Ultrasonic Sensor Cable	3
6PIN Color Sensor Cable	2
8PIN Tracking Module Cable	1
6PIN Motor Cable	2
USB Cable	1
Acrylic Plate A	1
Acrylic Plate B	1
Cross Screwdriver	1
User Guide	1

2.3 Technical Parameters

Table 2.2 Technical Parameters

Parameters	Description
Operating Voltage	7.4V
Control Board	DuDuino Mega (compatible with Arduino Mega 2560)
MPU	ATmega2560
Battery	18650 Li-ion rechargeable battery
Ultrasonic Measurement Range	3mm~500mm
VTBOT Size	195mm*172mm*79mm
VRBOT Weight	810g
Maximum Load	500g
Tire Diameter	67mm
Operating Environment	0°C~40°C
Control Software	Arduino IDE or Mixly
Communication Interface	USB communication, Serial communication
Expansion Interface	4PIN general I/O interface *2
Sensor	• Ultrasonic sensor *3 • Color sensor*2 • Infrared tracking model*1

Parameters	Description
	• Geomagnetic sensor*1 • Light sensor *1
Motor Parameters	• Reduction ratio: 48 : 1 • Voltage: 7V • No-load current: 150mA • Stall current: 700mA • Maximum rotate speed: 100r/m • Encoder resolution ratio: 585pulse/r

3. Feature Description

3.1 Aistarter Controller

3.1.1 Overview

AiStarter control board is designed based on Arduino Mega2560, compatible with Arduino, not only integrate motor drive, geomagnetic sensor, light sensor, button module, LED module and so on, but also integrate infrared tracking interface, ultrasonic sensor interface, USB, Xbee, bluetooth, serial port and so on. The main control board of Aistarter, which is shown in Figure 3.1 presets two servo signal interfaces for user expansion.

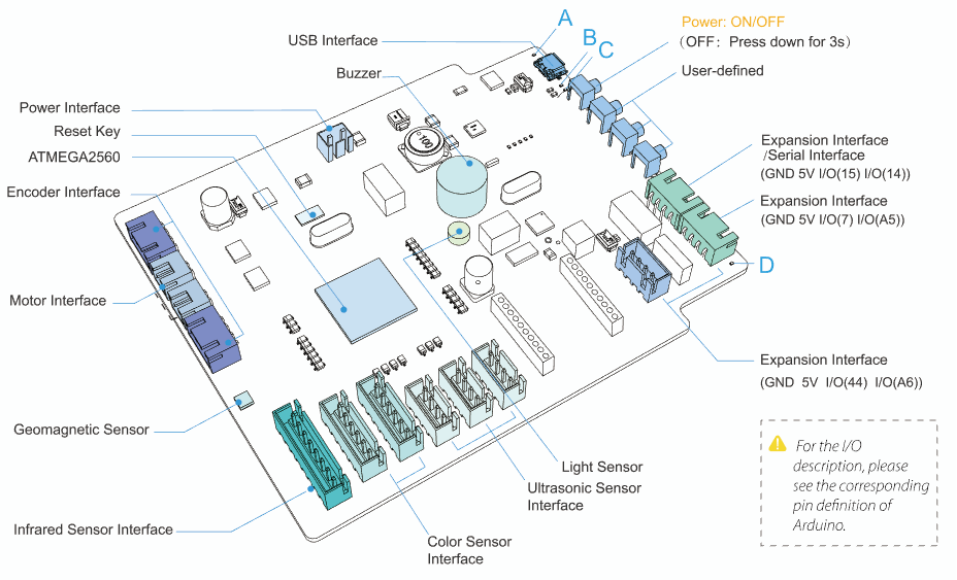


Figure 3.1 AiStarter control board description

3.1.2 AT Mega2560 Processor

AiStarter processor is atmega2560, which compatible with arduino2560, you can develop by arduinoIDE. Meanwhile we provide Mixly graphical programming environment.

3.1.3 Button

The end of AiStarter controller integrates four independence buttons, as shown in Figure 3.1, the detail descriptions are shown in Table 3.1.

Table 3.1 Button description

No.	Description
1	AiStarter Switch button, start or stop AiStarter When you stop AiStarter, you need to press and hold this button for about 3 seconds.
2	User-defined User can set function in Mixly or Arduino IDE environment

No.	Description
3	User-defined User can set function in Mixly or Arduino IDE environment
4	User-defined User can set function in Mixly or Arduino IDE environment

3.1.4 LED

The end of AiStarter controller integrates four LED indicators, as shown in Figure 3.1, the detail descriptions are shown in Table 3.2.

Table 3.2 LED indicators description

No.	Color	description
A	Blue	User-defined User can set function in Mixly or Arduino IDE environment
B	Red	• Off: Indicate that the AiStarter is uncharged. • Steady Red is always on: Indicates that the AiStarter is charging.
C	Red	• Off: Indicates that the AiStarter battery voltage is normal. • Steady Red: Indicates that the AiStarter battery has a low voltage
D	Blue	User-defined User can set function in Mixly or Arduino IDE environment

3.1.5 USB

AiStarter intergrates USB download function, you can download program to AiStarter by USB, and make AiStarter run by program. Meanwhile, USB charging function are supported by Aistarter, you can charge AiStarter by connecting AiStarter to computer with USB cable when AiStarter battery has a low voltage.

Operating systems are supported by USB driver:

- Win7
- Win8
- Win10

⚠NOTICE

When you install Mixly or Arduino IDE, the Arduino USB drive will be installed automatically. After connecting AiStarter to computer by USB cable and starting computer, you can find the corresponding COM port on device manager, as shown in Figure 3.2. If the corresponding COM port is not found, you need to install Arduino

USB drive again in ***Arduino-X*/drivers** path, X is the Arduino version, please replace it according to the actual situation.

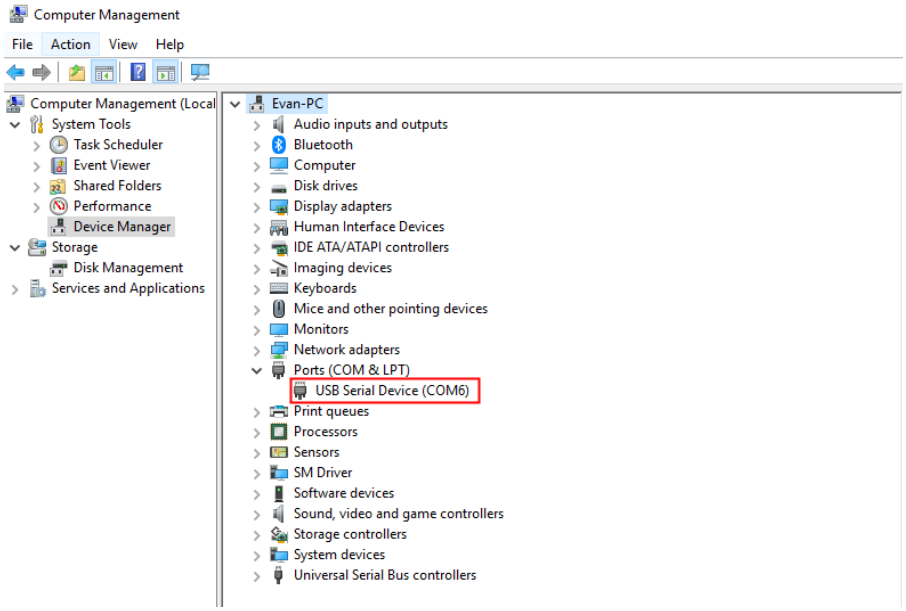


Figure 3.2 USB drive

3.1.6 Interface Description

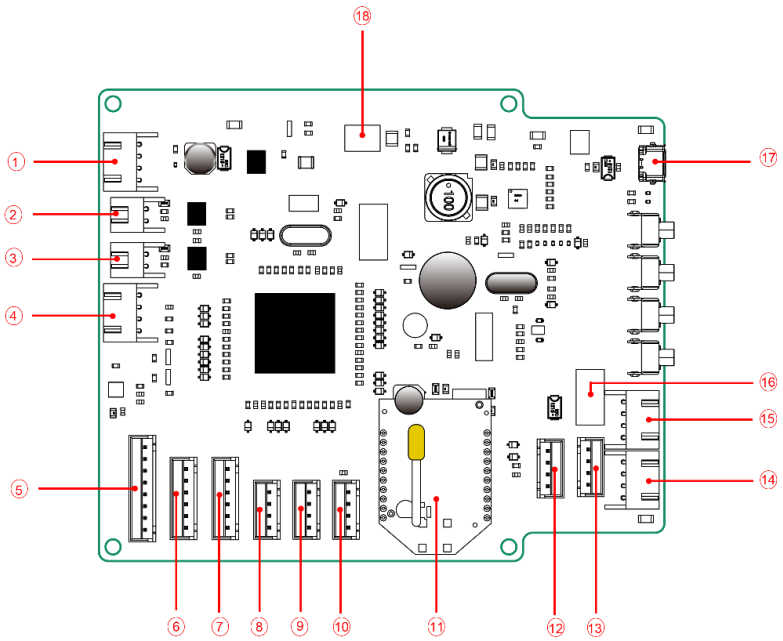


Figure 3.3 AiStarter main control board interface description

Table 3.3 Interface description

No.	Description
1	Encoder interface, connect to the right motor on the AiStarter chassis
2	Motor interface, connect to the right motor on the AiStarter chassis
3	Motor interface, connect to the left motor on the AiStarter chassis
4	Encoder interface, connect to the left motor on the AiStarter chassis
5	infrared sensor interface, connect to AiStarter infrared tracking sensor
6	Color sensor interface, connect to the right color sensor on the AiStarter chassis
7	The left color sensor interface, connect to the left color sensor on the AiStarter chassis
8	Ultrasonic sensor interface, connect to the ultrasonic sensor on the right side of AiStarter
9	Ultrasonic sensor interface, connect to the ultrasonic sensor on the front of AiStarter
10	Ultrasonic sensor interface, connect to the ultrasonic sensor on the left side of AiStarter
11	Xbee interface,
12	Reserved servo interface
13	Bluetooth interface, which is UART interface.
14	Reserved servo interface
15	Reserved serial interface, which is UART interface
16	Wifi interface, which is UART interface
17	USB interface, which is standard Micro-USB interface
18	Power interface, connected to battery on the AiStarter chassis

3.2 Infrared Sensor

An infrared sensor is built into the bottom of AiStarter, which can recognize black line by judging ground color for automatic patrol. 6-channel high-precision infrared pair tubes and 6 adjustable potentiometers are built into infrared sensor, which are used to adjust the distance detected by infrared pair tube. Infrared sensor can detect track black line accurately, the detection range is 3cm, and the accuracy is 0.1cm.

3.3 Ultrasonic Sensor

Three ultrasonic sensors are built into the head of AiStarter, which can detect obstacle distance in the range of 3mm to 500mm in front of AiStarter.

3.4 Color Sensor

Two color sensors are built into the bottom of AiStarter, which can recognize the color on the ground to execute different task according to different color.

4. Installation

4.1 Mixly Installation

Mixly is an open Arduino graphical programming software based on Google Blockly graphical programming frame. AiStarter adds wrapper function on Mixly, user can call wrapper function by building blocks and upload it to AiStarter to control AiStarter.

You can use Mixly directly after download without complicated installation. The download path is http://mixly.org/explore/software/mixly-arduino

📖NOTE

At present, AiStarter only support Mixly 0.995, the other versions are not supported.

4.2 Arduino IDE

AiStarter supports Arduino C programming language. Arduino is a convenient and flexible open source electronic platform, which includes Arduino develop tools Arduino IDE and core library.

The Arduino IDE package is nested in the Mixly package and can be used directly after extracting the Mixly installation package. The Arduino IDE software path is ***Installation path*/Dobot_Mixly/arduino-*XXX***. *XXX* is Arduino version, please replace it according to the actual situation.

5. User Operation

5.1 Mixly Introduction

Mixly page is shown in Figure 5.1. When you use Mixly, you need to select **Arduino\Genuino Mega or Mega 2560[atmega2560]** Development board and choose the corresponding serial port, as shown in Figure 5.1.

User can drag the programming blocks on the left of Mixly page, and upload program to AiStarter after compiling, AiStarter will work according to the program. For the Blockly details, please see *Blockly Description*. For the API function details, please seeAPI Function.

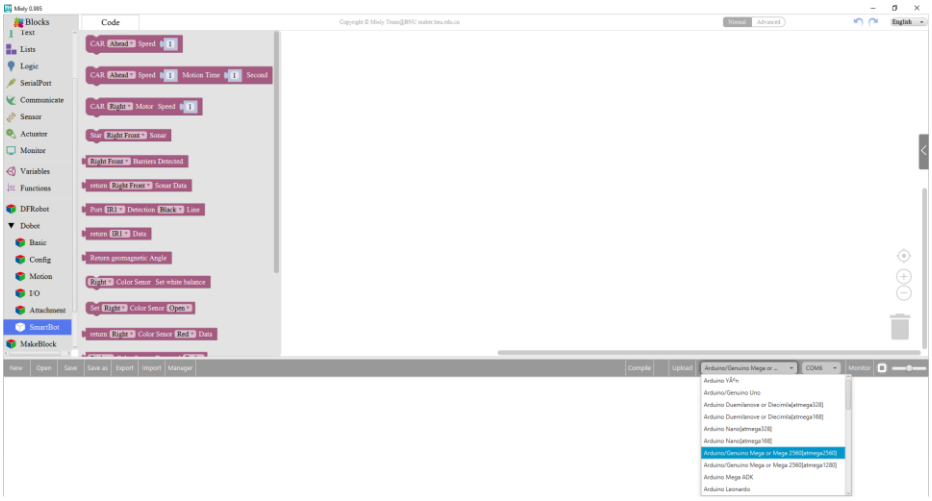


Figure 5.1 The page of Mixly

NOTICE

If you can not get the corresponding serial port of AiStarter when you open Mixly, please ensure the Arduino USB drive has been installed. If you still can not get serial port information after installing, please open Mixly as an administrator.

Mixly instructions are not described in detail in this manual. For the detail about Mixly, please see the related manual on Mixly official website.

5.2 Arduino IDE Introduction

The page of Arduino IDE is shown in Figure 5.2.

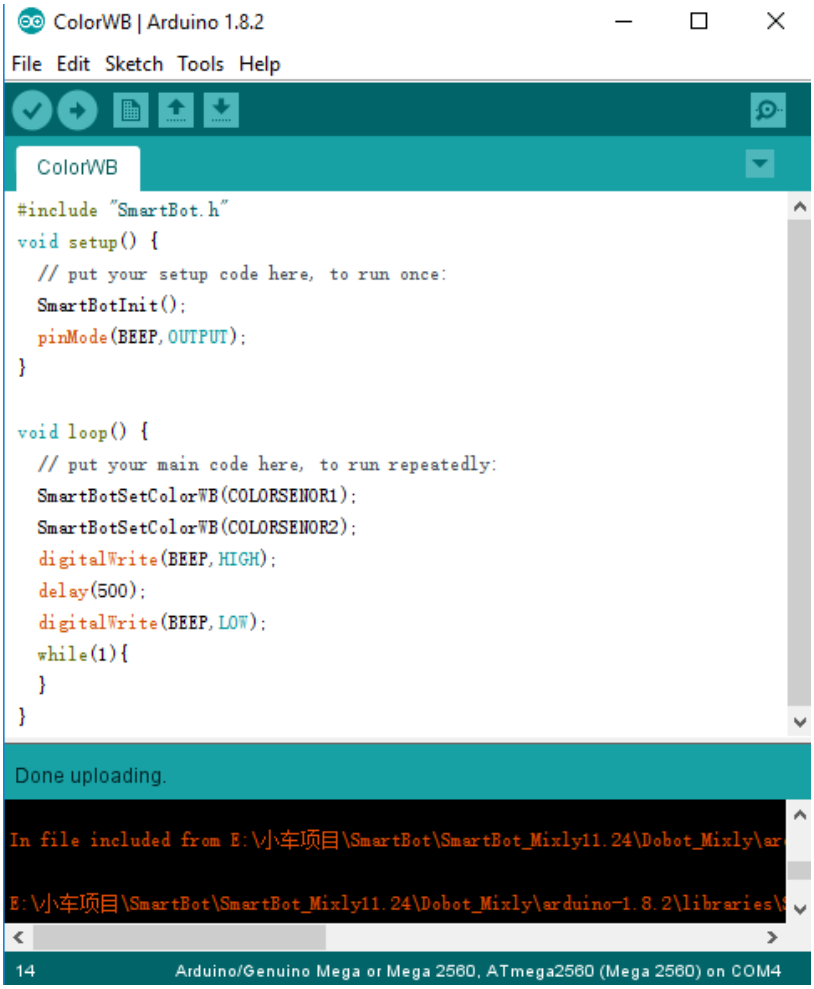


Figure 5.2 The page of Arduino IDE

After opening Arduino IDE, select **Arduino/Genuino Mega or Mega 2560** on the **Tools** > **Borad** page, select **ATmega2560 (Mega 2560)** in **Tools** > **Processor** path, select the corresponding serial port in **Tools** > **Port** path.

User can select example on the **File** > **Examples** > **AiStarter** page, click  to upload example to AiStarter, which will make AiStarter work according to the example. You can also refer to API Function to upload program to AiStarter by calling API function, which will make AiStarter work according to the program.

5.3 Blockly Description

5.3.1 Setting Direction and Speed

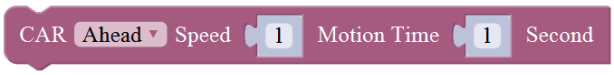
Table 5.1 Set direction and speed

Blockly	

Description	This module is used to set direction and speed
Parameter	Direction: • Ahead • Back • Turn left • Turn right Speed: Set running speed. Value range: 0r/m – 100r/m
Return	None

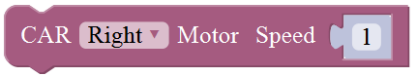
5.3.2 Setting Direction\Speed\Time

Table 5.2 Set direction\speed\time

Blockly	
Description	This module is used to set direction\speed\time
Parameter	Direction: • Ahead • Back • Turn left • Turn right Speed: Set running speed ,value range: 0r/m – 100r/m Motion Time: Set running time
Return	None

5.3.3 Setting Motor Speed

Table 5.3 Set motor speed

Blockly	
Description	This module is used to set the motor speed
Paramater	Motor: Select the motor (the left motor or the right motor) Speed: Set the motor speed. Value range: 0r/m – 100r/m
Return	None

5.3.4 Starting AiStarter Sonar

Table 5.4 Start AiStarter sonar

Blockly	
Description	This module is used to start AiStarter sonar
Paramater	Sonar position: • Right Front • Front • Left Front
Return	None

5.3.5 Detecting Barrier

Table 5.5 Detect barrier

Blockly	
Description	This module is used to detect whether a barrier is exist in front of AiStarter, before calling this module, please start the corresponding sonar
Paramater	Obstacle position: • Right front • Front • Left front
Return	Return 1: There is a barrier Return 0: There is no barrier

5.3.6 Getting The Detection Distance of Sonar

Table 5.6 Get the detection distance of sonar

Blockly	
Description	This module is used to get the detection distance of sonar, which is the distance between AiStarter and barrier.
Paramater	Sonar position: • Right front • Front • Left front

Return	Return detection distance

5.3.7 Detecting AiStarter Track

Table 5.7 Detect AiStarter track

Blockly	
Description	This module is used to detect whether tracking line is black or white
Paramater	Select infrared tracking sensor ports: • IR1 • IR2 • IR3 • IR4 • IR5 • IR6 Tracking line color: select color, black or white
Return	Return 1: There is a line Return 0: There is no line

5.3.8 Getting Infrared Sensor Data

Table 5.8 Get infrared tracking sensor data

Blockly	
Description	This module is used to get infrared sensor data
Paramater	Infrared sensor ports: • IR1 • IR2 • IR3 • IR4 • IR5 • IR6
Return	Return 0: It is black line Return 1: It is white line

5.3.9 Getting Geomagnetic Angle

Table 5.9 Get geomagnetic angle

Blockly	
Description	This module is used to get geomagnetic angle
Paramater	None
Return	Geomagnetic angle

5.3.10 Setting White Balance

Table 5.10 Set white balance

Blockly	
Description	This module is used to set white balance
Paramater	Select the left color sensor or the right color sensor
Return	None

5.3.11 Setting Color Sensor Status

Table 5.11 Set color sensor status

Blockly	
Description	This module is used to set color sensor enable state
Paramater	Color sensor: Selet the left color sensor or the right color sensor Status: Select open or close
Return	None

5.3.12 Getting RGB Value

Table 5.12 Get RGB value

Blockly	
Description	This module is used to get RGB color
Paramater	Color sensor: Select the left color sensor or the right color sensor

	Color: • Red • Green • Blue
Return	Return RGB value. Value range: 0 - 255

5.3.13 Detecting Color

Table 5.13 Detect color

Blockly	
Description	This module is used to detect color
Paramater	Color sensor: Select the left color sensor or the right color sensor Color: • Red • Green • Blue
Return	Return 1: There is a color Return 0: There is no color

5.3.14 Getting Switch Status

Table 5.14 Get switch status

Blockly	
Description	This module is used to get switch status
Paramater	Switch: • Switch1 • Switch2 • Switch3
Deturn	Retuen 1: Press Return 0: Release

5.3.15 Detecting Switch Status

Table 5.15 Detect switch state

Blockly	
Description	This module is used to detect switch state
Paramater	Switch: • Switch1 • Switch2 • Switch3 Switch state: Select switch state press or release
Deturn	Return 1: Status detected Retuen 0: No status detected

5.3.16 Getting Photosensitive Value

Table 5.16 Get photosensitive value

Blockly	
Description	This module is used to get photosensitive value
Paramater	None
Return	Return photosensitive value. Value range: 0 – 4096

6. AI-Starter Demo

6.1 Line Tracking Demo

6.1.1 Description

This demo realizes that AI-Starter tracks line automatically on testing map.

6.1.2 Procedure

Step 1 Turn on AI-Starter.

Step 2 Press down the key **start|stop**, AI-Starter starts tracking line after buzzer beeps one time.

Step 3 Press down the key **start|stop** once agian, AI-Starter stops tracking line after the buzzer beeps one time.

6.1.3 Code Description

1） Get the current infrared sensor value.

Program 6.1 Get infrared sensor value

```
void getCurrentIRState(int *irstate)
{
    *irstate = 0;
    for (int i = 0; i < IR_NUM; i++) {
        *irstate |= AIStarter_SmartBotGetIRModuleValue(i) << i;
    }
}
```

2） Get AI-Starter pose.

Program 6.2 Get AI-Starter pose

```
float getCurrentPos(const int irstate)
{
    const float coeff = 0.7;
    const int irPos[] = {-30, -18, -6, 6, 18, 30};
    static float lastPos;
    float curPos;
    float readPos;
    int total = 0;
    int irOffCnt = 0;
```



```
    //calculate the car position offset
    for (int i = 0; i < IR_NUM; i++) {
        if (irstate & (1 << i)) {
            total += irPos[i];
            irOffCnt++;
        }
    }
    if (irOffCnt) {
        readPos = total / irOffCnt;
    }
    else {
        readPos = lastPos;
    }
    //calculate the current position
    curPos = (1 - coeff) * lastPos + coeff * readPos;
    lastPos = curPos;
    return curPos;
}
```

3) Set AI-Starter speed.

Program 6.3 Set AI-Starter speed

```
void setCarSpeed(const float curPos)
{
    const int baseSpeed = 50; //rpm
    const float kp = 1;
    const float ki = 0.06;
    const float kd = 0.0;
    const float errorsumLimit = 50;

    float error = curPos;
    static float lastError;
    static float errorsum;
    float errorChange;
    int speedLeftWheel;
    int speedRightWheel;
```



```
    int speedOffset;


    //pid
    errorsum += error;
    if (errorsum > errorsumLimit) {
        errorsum = errorsumLimit;
    }
    else if (errorsum < -errorsumLimit){
        errorsum = -errorsumLimit;
    }
    errorChange = error - lastError;
    speedOffset = kp * error + ki * errorsum + kd * errorChange;
    lastError = error;


    //calculate the wheel speed
    speedLeftWheel    = baseSpeed + speedOffset;
    speedRightWheel = baseSpeed - speedOffset;


    AIStarter_SmartBotSetMotor(MOTORL, speedLeftWheel);
    AIStarter_SmartBotSetMotor(MOTORR, speedRightWheel);
    Serial.println("go ahead");
}
```

6.2 Obstacle Avoiding Demo

6.2.1 Description

This demo realizes that AI_Starter avoids obstacle automatically.

6.2.2 Procedure

Step 1 The motors stop running after turning on AI-Starter.

Step 2 Press down the key **start|stop**, AI-Starter starts avoiding obstacle.

Step 3 AI-Starter moves back for a certain distance after detecting obstacle, and turn left or right according to the ultrasonic sensors.

Step 4 Press down the key **start|stop** once agian, AI-Starter stops tracking line after the buzzer beeps one time.

6.2.3 Code Description

1） Initialize AI-Starter.

Program 6.4 Initialize AI-Starter

```
AIStarter_SmartBotInit();
```

2) Put the ultrasonic sensor values into the array **dis**.

Program 6.5 Save ultrasonic sensor values

```
dis[0] = AIStarter_SmartBotGetSonar(SONAR1);
dis[1] = AIStarter_SmartBotGetSonar(SONAR2);
dis[2] = AIStarter_SmartBotGetSonar(SONAR3);
```

3) Set turning mode according to the distance between obstacle and AI-Starter.

Program 6.6 Set turning mode

```
    if(dis[0] > 2*DIS ){
        motorStatus = AHEAD;
    }else if(dis[0] > DIS    && dis[0] < 2*DIS ){
        motorStatus = motorStatus;
    }else if(dis[0] > 0    && dis[0] < DIS ){
        motorStatus = BACKRIGHT;
        break;
    }
    if(dis[1] > 2*DIS ){
        motorStatus = AHEAD;
    }else if(dis[1] > DIS    && dis[1] < 2*DIS ){
        motorStatus = motorStatus;
    }else if(dis[1] > 0    && dis[1] < DIS ){
        motorStatus = BACKRIGHT;
        break;
}
if(dis[2] > 2*DIS ){
        motorStatus = AHEAD;
    }else if(dis[2] > DIS    && dis[2] < 2*DIS ){
        motorStatus = motorStatus;
    }else if(dis[2] > 0    && dis[2] < DIS ){
        motorStatus = BACKLEFT;
        break;
    }
```


4) AI-Starter turns around according to obstacle position.

- BACKRIGHT: turn right
- BACKLEFT: turn left
- AHEAD: go straight.

Program 6.7 Turns around according to obstacle position

```
    switch(motorStatus){
        case BACKRIGHT:
            AIStarter_SmartBotSetMotor(MOTORR,BACKSPEED);
            AIStarter_SmartBotSetMotor(MOTORL,BACKSPEED);
            delay(BACKTIME);
            AIStarter_SmartBotSetMotor(MOTORR,DIFSPEED);
            AIStarter_SmartBotSetMotor(MOTORL,FRONTSPEED);
            delay(SWERVETIME);
        break;
        case BACKLEFT:
            AIStarter_SmartBotSetMotor(MOTORR,BACKSPEED);
            AIStarter_SmartBotSetMotor(MOTORL,BACKSPEED);
            delay(BACKTIME);
            AIStarter_SmartBotSetMotor(MOTORR,FRONTSPEED);
            AIStarter_SmartBotSetMotor(MOTORL,DIFSPEED);
            delay(SWERVETIME);
        break;
        case AHEAD:
            AIStarter_SmartBotSetMotor(MOTORR,FRONTSPEED);
            AIStarter_SmartBotSetMotor(MOTORL,FRONTSPEED);

        break;
        default:
        break;
}
```

6.3 White Balance Calibration Demo

6.3.1 Description

This demo realizes white balance calibration.

6.3.2 Procedure

Step 1 Put AI-Starter on a piece of A4 paper.

Step 2 Put down the key **start|stop** to calibrate white balance.

6.3.3 Code Description

1) Initialize AI-Starter.

Program 6.8 Initialize AI-Starter

```
AIStarter_SmartBotInit();
```

2) Calibrate white balance.

Program 6.9 Calibrate white balance

```
AIStarter_SmartBotSetColorWB(COLORSENOR1);
AIStarter_SmartBotSetColorWB(COLORSENOR2);
```

6.4 Color Recogition and Line Tracking Demo

6.4.1 Description

This demo realizes the function that combines color recognition with line tracking.

6.4.2 Procedure

Step 1 Press down the key **start|stop**, AI-Starter starts tracking line after buzzer beeps one time.

Step 2 In tracking line process, when detecting black line, AI-Starter stops moving, and begins to detect color.

Step 3 When the recognized color is red, AI-Starter stops moving for 3s, and the buzzer beeps three times.

Step 4 When the recognized color is red, AI-Starter stops moving for 3s, and the buzzer makes a long call for 3s.

Step 5 AI-Starter goes ahead after finishing the above steps

Step 6 Press down the key **start|stop** once agian, AI-Starter stops tracking line after the buzzer beeps one time.

6.4.3 Code Description

1) Initialize AI-Starter.

Program 6.10 Initialize AI-Starter


```
AIStarter_SmartBotInit();
```

2) Detect color.

Program 6.11 Detect color

```
if(AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR)
AIStarter_SmartBotGetColorSenor(COLORSENOR1,GCOLOR) > 30 &&
    AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR)
AIStarter_SmartBotGetColorSenor(COLORSENOR1,BCOLOR) > 30) {
            colorState = RLINE;
}
```

3) AI-Starter executes different action according to different color status.

Program 6.12 Execute action

```
    switch(colorState) {
        case OTHERLINE:
            //colorRec = false;
            lineState = LINEPATROL;
        break;
        case RLINE:
            delay(3000);
            //colorRec = false;
            lineState = LINEPATROL;
        break;
        case GLINE:
            delay(3000);
            //colorRec = false;
            lineState = LINEPATROL;
        break;
        default:
        break;
  }
```

6.5 Cooperation Demo

6.5.1 Description

This demo realizes the coorperation of AI-Starter and Magician.

1) AI-Starter moves to the green line and then stops moving for one minute.

2) Magician Detects AI-Starter moves to the carring position by Pixy.

3) Magician starts to grab cubes to AI-Starter.

4) After one minute, AI-Starter finishes carrying, and then starts line tracking function.

6.5.2 Procedure

AI-Starter

Step 1 Press down the key **start|stop**, AI-Starter starts tracking line after the buzzer beeps one time.

Step 2 In tracking line process, when detects black stoping line, AI-Starter stops moving, and begins to detect color.

Step 3 When the recognized color is red, AI-Starter stops moving for 3s, and the buzzer beeps three times.

Step 4 When the recognized color is green, AI-Starter stops moving for 1 minute, and the buzzer beeps one time.

Step 5 AI-Starter keeps tracking line.

Step 6 Press down the key **start|stop** once agian, AI-Starter stops tracking line after the buzzer beeps one time.

Magician

Step 1 Open Magician file in the Magician Coorperation file folder to upload Magician firmware to Arduino expansion board.

Step 2 When detecting that AI-Starter moves to the carrying position, Magician begins to grab cubes to AI-Starter.

Step 3 Ai-Starter keeps tracking line.

6.5.3 Code Description

AI-Starter

1) Initialize AI-Starter.

Program 6.13 Initialize AI-Starter

```
AIStarter_SmartBotInit();
```

2) Detect color.

Program 6.14 Detect color

```
if(AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR)
AIStarter_SmartBotGetColorSenor(COLORSENOR1,GCOLOR) > 30 &&
```

```
    AIStarter_SmartBotGetColorSenor(COLORSENOR1,RCOLOR)
AIStarter_SmartBotGetColorSenor(COLORSENOR1,BCOLOR) > 30) {
                colorState = RLINE;
}
```

3） AI-Starter executes different action according to different color status.

Program 6.15 Execute action

```
    switch(colorState) {
            case OTHERLINE:
                //colorRec = false;
                lineState = LINEPATROL;
            break;
            case RLINE:
                delay(3000);
                //colorRec = false;
                lineState = LINEPATROL;
            break;
            case GLINE:
                delay(3000);
                //colorRec = false;
                lineState = LINEPATROL;
            break;
            default:
            break;
   }
```

Magician

1） Set cube position.

Program 6.16 Set cube position

```
float AreaPoint[4][3] = {
   {137.05, -206.94, -39},
   {137.05, -244.31, -39},
   {100.50, -206.94, -39},
   {100.50, -244.31, -39}
};
```


2） Set AI-Starter carrying position.

Program 6.17 Set carring position

```
float trayPoint[4][3] = {
   {308.12,    25.92,    28},
   {308.12, -15.92,    28},
   {258.12,    25.92,    28},
   {258.12, -15.92,    28}
};
```

3） Magician grabs cubes to AI-Starter.

Program 6.18 Grab cude

```
void AreaToAIStarter()
{
      for(uint8_t i=0; i<4; i++){
            Dobot_SetPTPCmdEx(JUMP_XYZ, AreaPoint[i][0], AreaPoint[i][1], AreaPoint[i][2], 0);
            Dobot_SetEndEffectorSuctionCupEx(true);
            Dobot_SetPTPCmdEx(MOVL_XYZ, AreaPoint[i][0], AreaPoint[i][1], AreaPoint[i][2]+70, 0);
            Dobot_SetPTPCmdEx(JUMP_XYZ, trayPoint[i][0], trayPoint[i][1], trayPoint[i][2], 0);
            Dobot_SetEndEffectorSuctionCupEx(false);
            Dobot_SetPTPCmdEx(MOVL_XYZ, trayPoint[i][0], trayPoint[i][1], trayPoint[i][2]+30, 0);
      }
      Dobot_SetPTPCmdEx(MOVJ_XYZ, InitPositionX, InitPositionY, InitPositionZ, InitPositionR);
}
```

4） Initialize Pixy and Magician.

Program 6.19 Initialize Pixy and Magician

```
pixy.init();
Dobot_Init();
```

5） Detect cube number.

Program 6.20 Detect cube number

```
pixy.ccc.getBlocks();
```


7. API Function

7.1 Initialization

Table 7.1 Initialization

Function	int AIStarter_SmartBotInit ();
Description	Initialization
Paramater	None
Return	None

7.2 Setting Direction and Speed

Table 7.2 Set direction and speed

Function	int AIStarter_SmartBotSetMovment (int dir, int speed)
Description	Set direction and speed

Paramater	dir: Set direction
	enum{
	FRONT,
	BACK,
	RIGHT,
	LEFT
	};
	speed: Set Duty Ratio, Value range: 0 – 255

Return	None

7.3 Setting Direction\Speed\Time

Table 7.3 Set direction\speed\time

Function	int AIStarter_SmartBotSetMovmentTime (int dir, int speed, float time)
Description	Set direction, speed and time
Paramater	dir: Set direction enum{ FRONT, BACK, RIGHT,

	LEFT }; speed: Set Duty Ratio. Value range: 0 – 255 time: Set running time (unit: s)
Return	None

7.4 Setting Motor Speed

Table 7.4 Set motor speed

Function	int AIStarter_SmartBotSetMotor (int port,int speed)
Description	Set motor speed

Paramater	port: Select motor
	enum{
	MOTORR,
	MOTORL
	};
	speed: Set speed. Value range: 0 – 200rpm

Return	None

7.5 Setting Motor Parameter

Function	int AIStarter_SmartBotSetMotorPI (float KP, float KI)
Description	Set motor parameter
Paramater	KP: Proportion factor. Value range: 0.5~2.5 KI: Integral factor. Value range: 0.05~0.5
Return	None

7.6 Getting Motor Pose

Function	float AIStarter_SmartBotGetMotorPose (int port)
Description	Get motor pose
Paramater	port: Select motor enum{ MOTORR, MOTORL

	};
Return	Return motor value

7.7 Initializing Ultrasonic Sensor

Table 7.5 Initialize ultrasonic sensor

Function	int AIStarter_SmartBotSetSonar (int port)
Description	Initialize ultrasonic sensor

Paramater	port: Select ultrasonic sensor
	enum {
	SONAR1,
	SONAR2,
	SONAR3
	};

Return	None

7.8 Getting the Detection Distance of Ultrasonic Sensor

Table 7.6 Get the detection distance of ultrasonic sensor

Function	float AIStarter_SmartBotGetSonar (int port)
Description	Get the detection distance of ultrasonic sensor

Paramater	port: Select ultrasonic sensor
	enum {
	SONAR1,
	SONAR2,
	SONAR3
	};

Return	Return detection distance (Unit: cm)

7.9 Detecting obstacle

Table 7.7 Detect obstacle

Function	bool AIStarter_SmartBotGetBarrier (int port)
Description	Detect obstacle

Paramater	port: Select ultrasonic sensor
	enum {
	SONAR1,
	SONAR2,
	SONAR3
	};

Return	1: Obstacle detected 0: No obstacles detected

7.10 Getting Infrared Sensor Data

Table 7.8 Get infrared sensor data

Function	int AIStarter_SmartBotGetIRModuleValue (int port)
Description	Get infrared sensor data

Paramater	port: Select infrared sensor ports
	enum {
	IR1,
	IR2,
	IR3,
	IR4,
	IR5,
	IR6
	};

Return	Return 1: Black line detected Return 0: No black line detected

7.11 Getting Geomagnetic Angle

Table 7.9 Get geomagnetic angle

Function	float AIStarter_SmartBotGetCompass ()
Description	Get geomagnetic angle
Paramater	None
Return	Return geomagnetic angle

7.12 Geomagnetic Calibration

Table 7.10 Geomagnetic calibration

Function	void AIStarter_SmartBotSetCompassCalibration()
Description	Calibration method: Press down the left-most key after starting up, make AI-Starter rotate 360° around space axes X, Y, Z respectively, press down the left-most key once again to finish calibration
Paramater	None
Return	None

7.13 Setting White Balance

Table 7.11 Set white balance

Function	int AIStarter_SmartBotSetColorWB(int port)
Description	Set white balance

Paramater	port: Color sensor
	enum{
	COLORSENOR1,
	COLORSENOR2
	};

Return	None

7.14 Setting Color Sensor Status

Table 7.12 Set color sensor status

Function	int AIStarter_SmartBotSetColorSenor (int port,bool ison)
Description	Set color sensor status

Paramater	port: Color sensor
	enum{
	COLORSENOR1,
	COLORSENOR2
	};
	Ison: True: open, False: close

Return	None

7.15 Detecting Color

Table 7.13 Detect color

Function	bool AIStarter_SmartBotDetColorSenor (int port, int color)
Description	Detect color

Paramater	port:Color sensor
	enum{
	COLORSENOR1,
	COLORSENOR2
	};
	color: Detect color
	enum{
	RCOLOR,
	GCOLOR,
	BCOLOR,
	};

Return	Return 1: Color detected Return 0: No color detected

7.16 Getting RGB Value

Table 7.14 Get RGB value

Function	int AIStarter_SmartBotGetColorSenor (int port,int color)
Description	Get RGB value

Paramater	port: Color sensor
	enum{
	COLORSENOR1,
	COLORSENOR2
	};
	color: Get color
	enum{
	RCOLOR,
	GCOLOR,
	BCOLOR,
	};

Return	Return color value

7.17 Initializing Switch

Function	int MobilePlatform_SmartBotSetKeyInit ()
Description	Initiallize switch
Paramater	None
Return	None

7.18 Getting Switch Status

Table 7.15 Get switch status

Function	int AIStarter_SmartBotGetKeyValue (int key)
Description	Get switch status

Paramater	key: Select switch
	enum{
	SW1,
	SW2,
	SW3
	};

Return	Return 1: Press down Return 0: Release

7.19 Setting LED Status

Function	int AIStarter _SmartBotSetLED(int port,int state)
Description	Set LED

Paramater	port: Select LED
	enum{
	LED1
	LED2
	}
	state: Set status
	enum{
	ON,
	OFF,
	BLINK
	};

Return	None

7.20 Getting Photosensitive Value

Table 7.16 Get photosensitive value

Function	int AIStarter_SmartBotGetLightAnalog ()
Description	Get photosensitive value
Paramater	None
Return	Return photosensitive value

7.21 Setting Ultrasonic Threshold

Function	int AIStarter_SmartBotSetSonarThreshold (int dis)
Description	Set ultrasonic threshold
Paramater	dis: Set threshold。(Unit: cm)
Return	None