KitiBot-MG User Manual

PREFACE

KitiBot-MG is intended for DIY Makers education and design at introductory level. Since robots indeed seem to be very attractive for children, most of them wish to have a self-created robot. However, to undergo such endeavor it mostly implies studying dull content to learn essential skills that might be daunting to children. KitiBot-MG offers an interesting approach into the exciting world of mobile robotics, by making it simple for children to learn. KitiBot-MG robot comes with a building blocks style editor, which is a simple diagram programming mode that can assist children to get started quickly, nurture their logic designing tactics, to encourage invention along with exploration.

KitBot-MG comes with a tutorial that is oriented to middle school learners. Using the drag & drop step-by-step method, that provides an intuitive mean to experiment a very easy learning way, with no need of any sort of skills about electronics and programming. It is based on the Arduino Mega2560 and it supports Arduino IDE. Its board is equipped with dual-mode Bluetooth, light sensor, RGB LED, buzzer on-board plus more than 10 connectors for programmable module devices. This robot can be implemented for both Bluetooth and IR remote control and obstacle avoiding using ultrasounds and IR detection functions, etc. KitBot-MG robot have two version. One is the car with 2 wheels and another is a caterpillar-like shaped. Both are compatible for the same tutorial and its logics design use.

Due to the product limited characteristics, as seen on this type of development boards, the user has the responsibility to employ the product only for the intended purposes of this manual, as to avoid any non-reparable damage of restore its full functionality. Please, as much as possible; follow the instructions of its manual/tutorial.

The author would like to advice that due to time constraints and limited extensive knowledge of this document contents and its tutorial, is inevitable to prevent any sort of mistake. Hence, the author would appreciate to the readers for any error content report.

For obtaining extra material of this product, for browsing more specific product information, which include the user manual, schematic, sample code and datasheets, please refer to http://www.waveshare.net/wiki/KitiBot-MG

CONTENT

GETTING READY

Welcome to the world of KitiBot robot! KitiBot is a smart car kit designed for primary and secondary school students. It can be programmed graphically, and can be used with no previous programming experience.

Through studying this tutorial, you will experience the fun of programming, having a direct and deeper understanding of electronic devices and software. This tutorial consists of three core components: a robot, a master control board, and AlphaBlock software. The control board is the brain of the robot, which controls the robot to perform various actions, and the robot is made up of the main control board, various sensors and electric motor batteries.

AlphaBlock is based on Scratch 2.0, which can be added to the robot by graphical programming, enabling him to achieve various functions according to our ideas. The AlphaBlock interface is shown below.

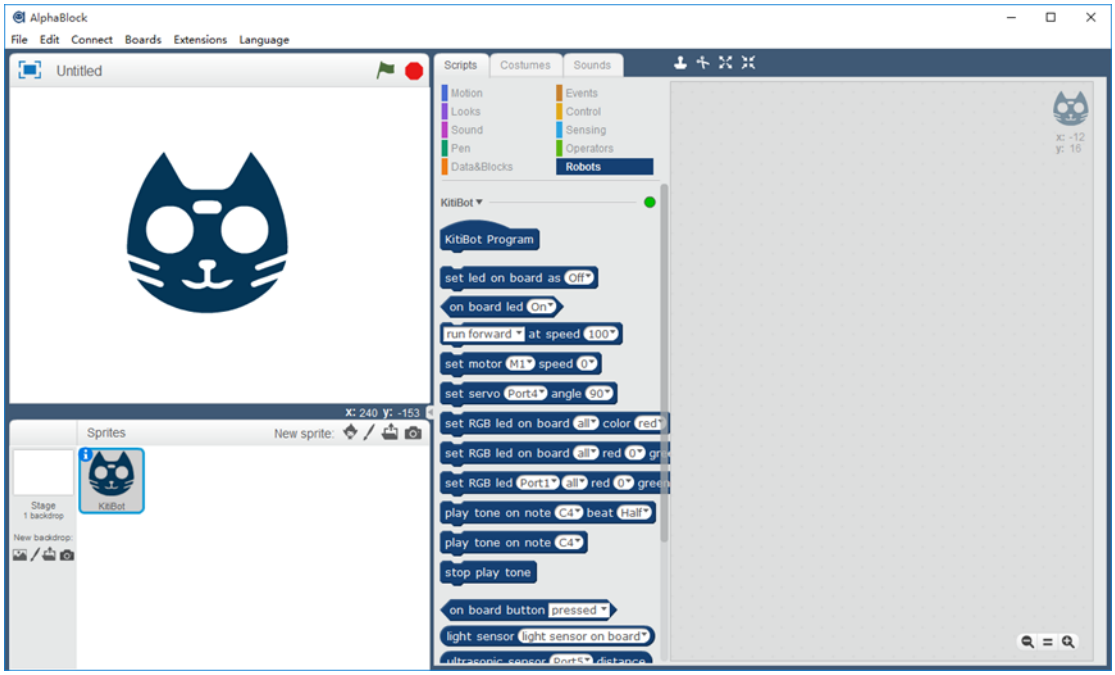


Figure 1 AlphaBok Main Page

The main control board contains various sensors, as shown in the following figure:

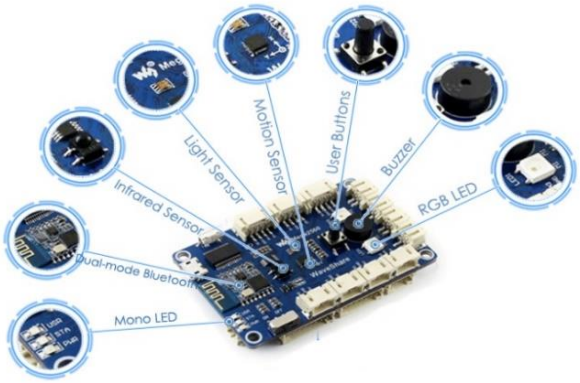


Figure 2 Control Board

SECTION 1 ALPHABLOCK

FILE

Below is an introduction to how to use AlphaBlock software. Beforehand, let's get started knowing the File menu.

"New": Creates a new project

"Load project": Open an existing project

“Save project”: Saves the current project to a file locally

Note: The project file extension is ". sb2". All the working files of this tutorial are already provided. But we recommend you to follow the tutorial steps for learning project creation and programming edition proficiency.

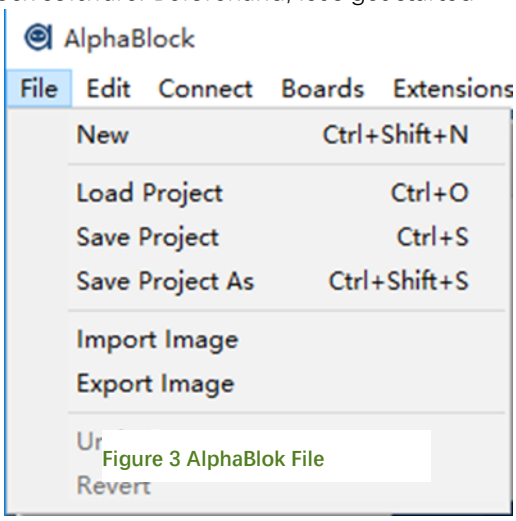


Figure 3 AlphaBlok File

EDIT

"Undelete": Restores back any wrong or accidental edition

"Hide stage layout": hide the stage and role on the left of the program

“Small stage layout”: small the stage and leave more area for program edition

"Turbo mode" : speed up stage redraw of AlphaBlock.

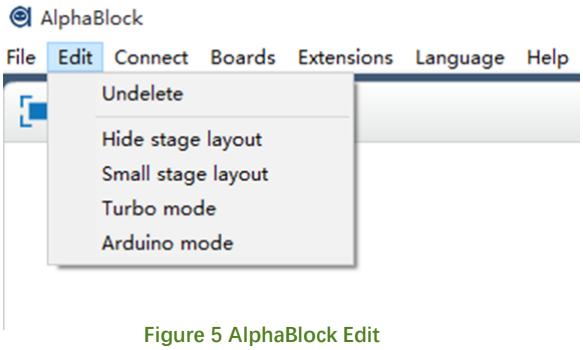


Figure 5 AlphaBlock Edit

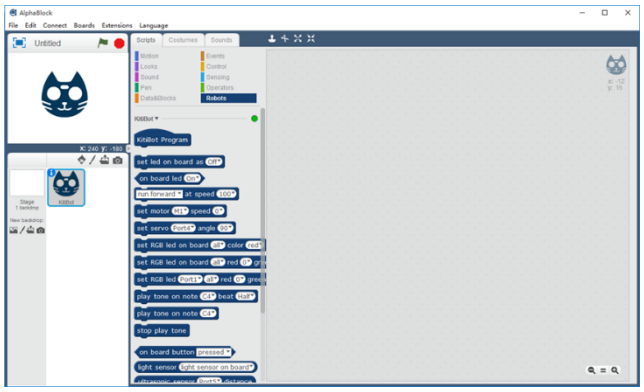


Figure 4 AlphaBlock Edit Turbo Mode

"Arduino mode" : Hides the drawing stage, and the Arduino code editing area will appear on the right. You can convert AlphaBlock's block-based program to Arduino program and upload it to the Arduino main board for offline operation. In the lower right corner you can see that AlphaBlock communicates with the robot directly

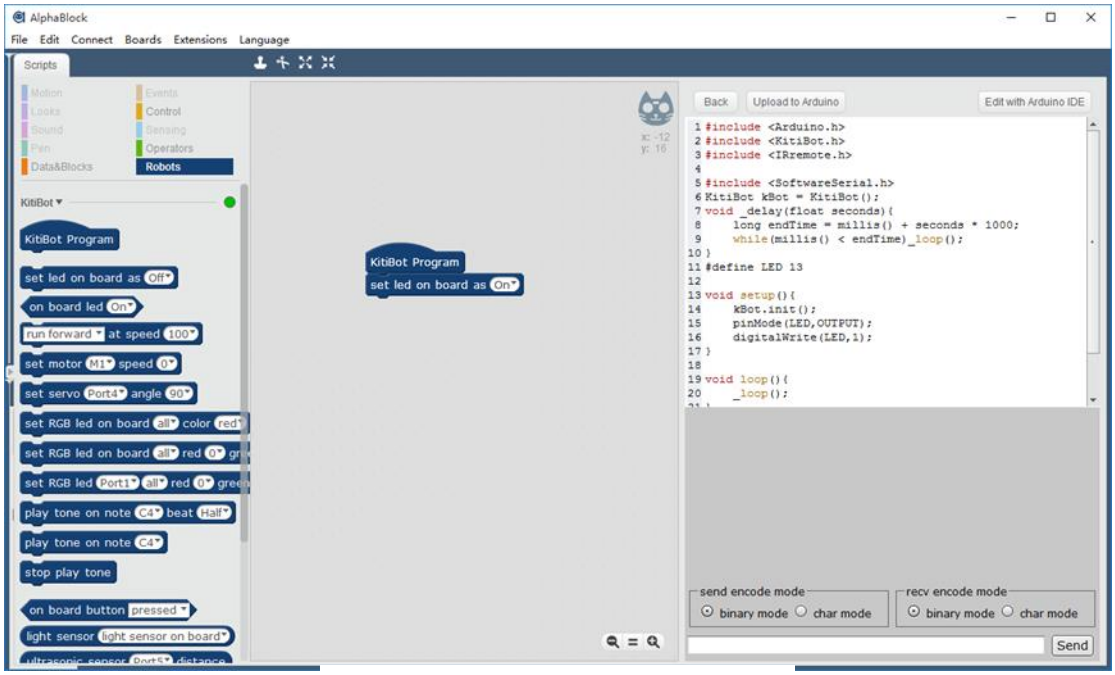


Figure 6 AlphaBlock Edit Arduino Mode

CONNECT

With "Connect" menu, you can select the corresponding connection mode. The robot supports both serial port connection and Bluetooth connection. USB serial connection robot can be used to "install firmware" and "restore factory procedures".

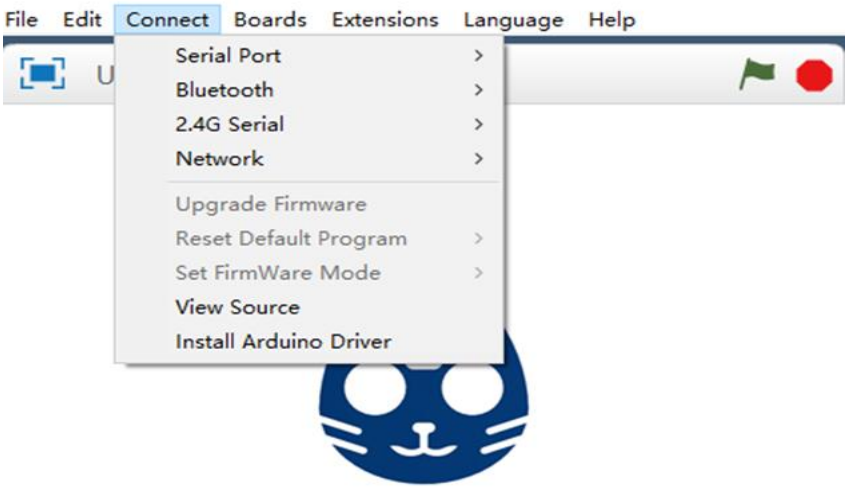


Figure 7 AlphaBlock Connect

BOARDS

"Boards" On Boards Option you can choose to connect to corresponding development board

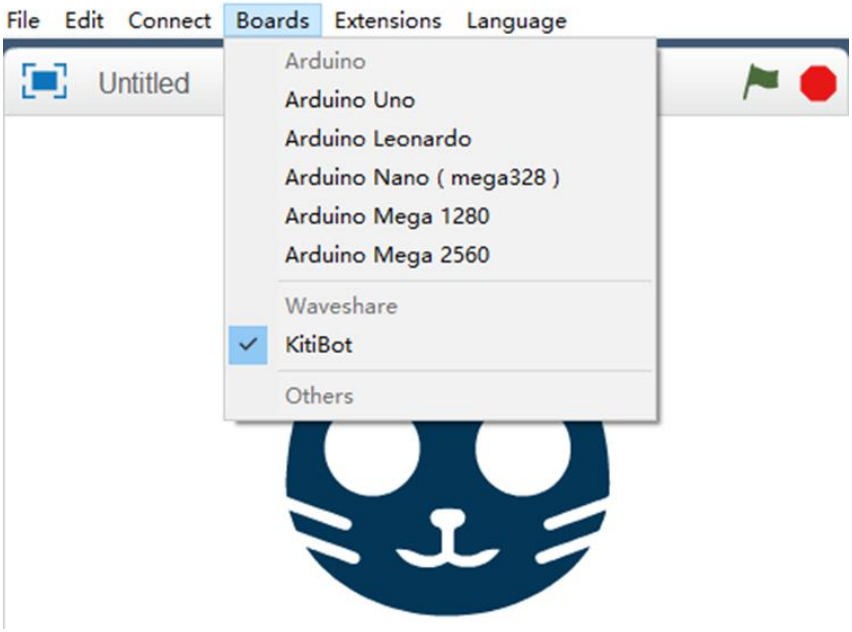


Figure 8 AlphaBlock Boards

EXTENSIONS

The "Extensions" menu allows you to select a different expansion module. After selecting KitiBot, many of the building blocks of KitiBot will appear in the robot module. You can use them to control the robot.

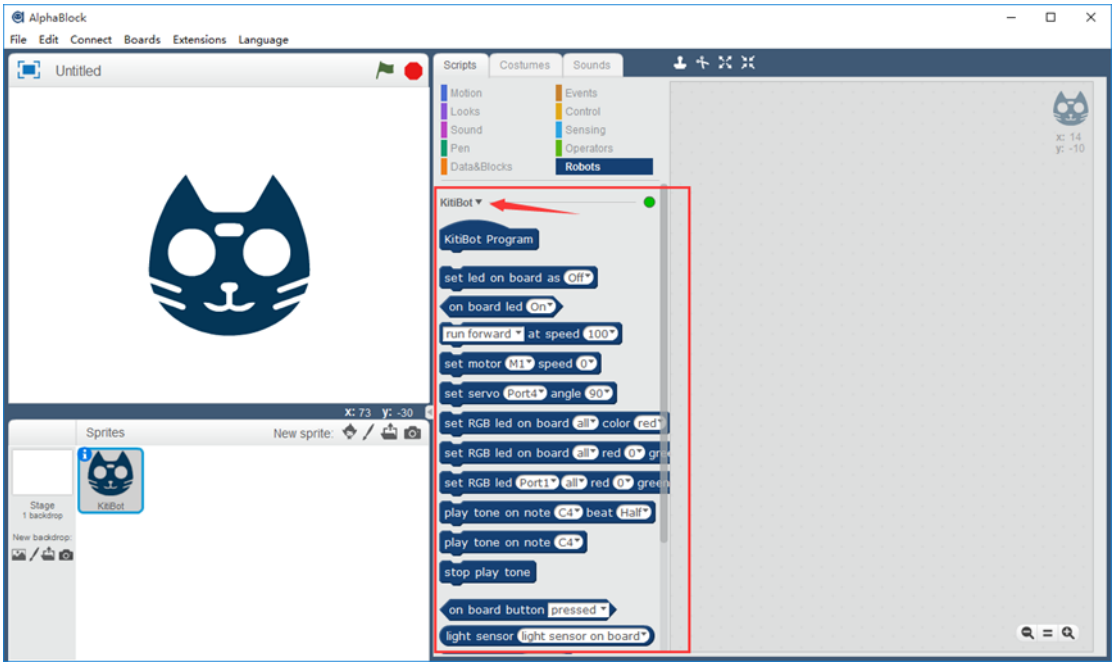


Figure 9 AlphaBlock Extensions

LANGUAGE

The Language menu allows you to choose a different language.

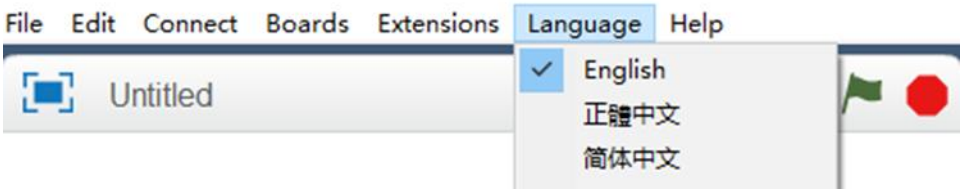


Figure 10 AlphaBlock Language

SECTION 2 DRIVE INSTLLATION AND SERIAL PORT CONNECTION

Only after the robot and AlphaBlock are connected can the robot be controlled through the AlphaBlock software. There are two ways to connect, the first one is through the USB cable connection. We will connect by USB cable, plugging one end of the USB cable to the computer, and the other end into the main control board. Open the Alphablock software and select the corresponding COM port:

Figure 11 AlphaBlock COM

After clicking the corresponding COM, a " √ " symbol will be displayed, indicating it is connected successfully.

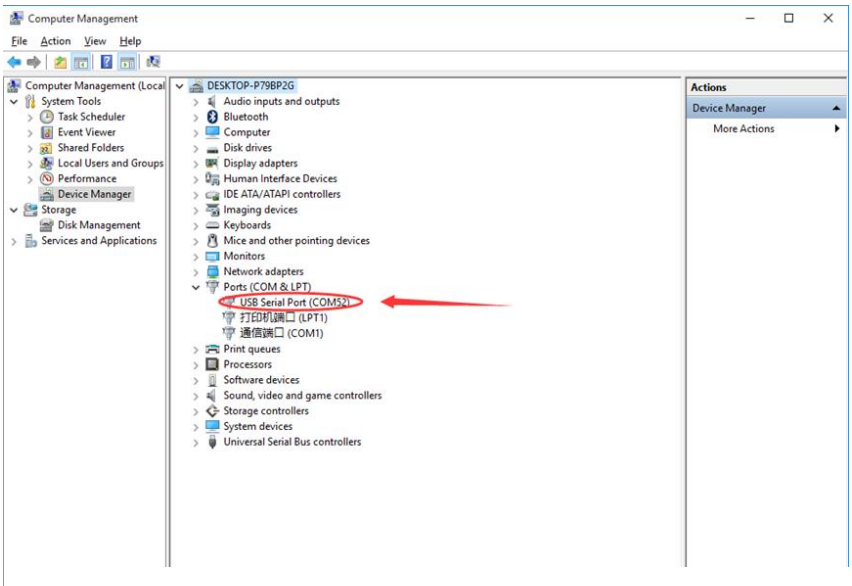


Figure 12 Computer USB Serial Port (Connected)

Connecting to different computers might have different COM port numbers so, how do you find the COM port corresponding to the robot? Open the device manager of your computer and find the "Port (COM and LPT)", where USB Serial Port is the Port of the corresponding robot.

If the port number is not found in the device manager, or if the serial port is not identified, it is possible that the driver is not installed.

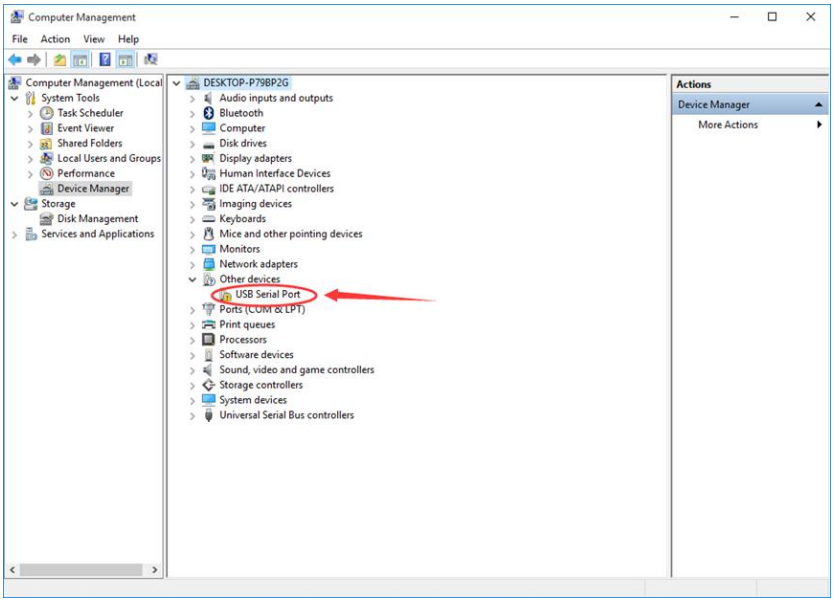


Figure 13 Computer USB Serial Port (Error)

In this case, Right-click and select "Update Driver Software", then select the second option "Browse my computer for driver software". Click Browse to select the driver's subdirectory in the AlphaBlock directory. Click “Next” to install the driver.

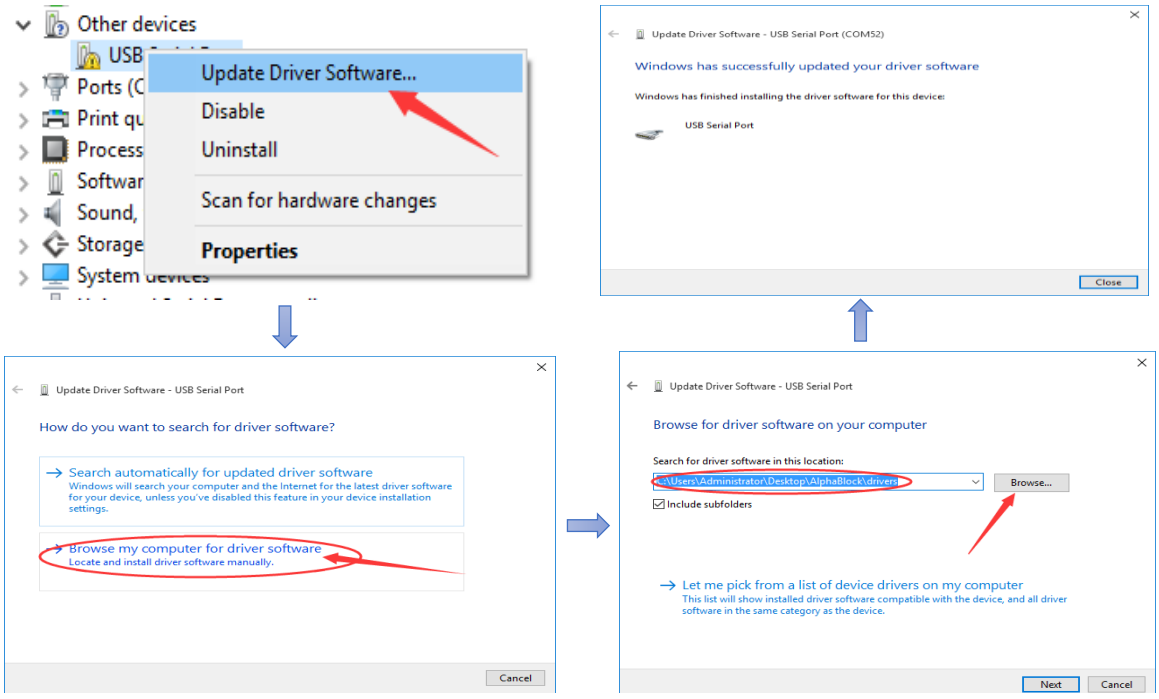


Figure 14 USB Driver Installtion

SECTION 3 FACTORY PROGRAM DEMO

After connecting to the COM port, click "Reset Default Program" to begin to upload the factory program demo, meanwhile, the main control board green LED will start flashing until the program is uploaded.

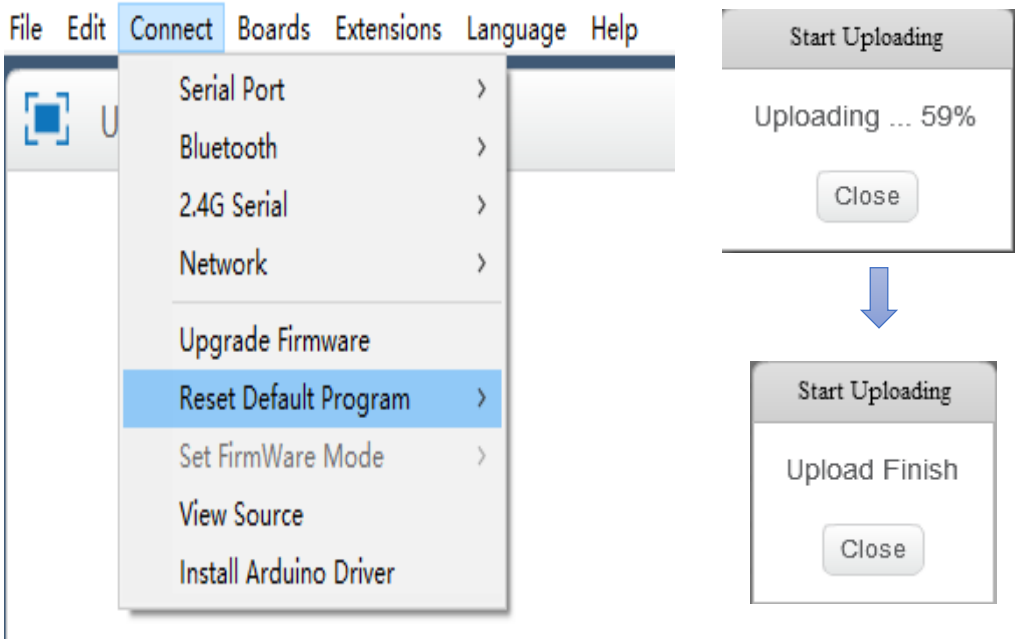


Figure 15 Default Program Uploading

At this point, the robot's main control board already has the factory program uploaded, the green LED on the main control board starts flashing, RGB LED will display different colors. You can now control the robot with the remote controller.

Figure 16 IR Remote Controller

Pressing the IR remote controller's number keys, 2, 8, 4, 6, 5, respectively, it controls the robot moving forward, backwards, turn left, turn right and stop.

At the same time is pressed any movement keys, the RGB LED will show different colors, and the buzzer will make a different sound.

Press "1" : the robot's head will look to the left.
Press "2" : the robot's head will look to the right
Press "-", "+" : to adjust the speed of the robot moving forward and backwards.

There are three kinds of working modes: remote control mode, obstacle avoidance mode and tracking mode. The factory program starts working in remote control mode by default. You can press the following button to switch to other working modes.

"CH-" : remote control mode.
"CH" : obstacle avoidance mode
"CH+" : tracking mode.

"Remote mode": You can remote control the robot to move around, shaking his head.
"Obstacle avoidance Mode": The robot will move forward until any obstacle is encountered. The robot will turn his head left and right, detecting the distance in between two sides of the obstacles, then will choose the farthest distance to turn. You can place your hand in front of the robot to make him turn automatically.
“Tracking mode”: This mode requires the robot working on specific environment with a white background tracking a black line. In such scene, the robot will move along the black line.

The smart cat robot can be remote-controlled by remote control, as well as it can be remote-controlled by Bluetooth. The main control board has integrated a dual mode Bluetooth module, which can be controlled by mobile APP.

Use you Android cell phone to scan the following QR code to download its corresponding APP.

Figure 17 QR Code of Bluetooth APP

Start the APP, click “Bluetooth Control”, and click "Search" (Note: your cell phone needs to turn on the bluetooth function). After about a few seconds, the corresponding Bluetooth device will be displayed in the list. Select the "KitiBot" device. Go to the next page and select remote control mode. At this point, the control page will appear and the "connection success" will be prompted. The blue LED on the main control board will turn from blinking to bright, indicating that the phone has been successfully connected. Press the control buttons to control the robot. And you can choose different modes of work.

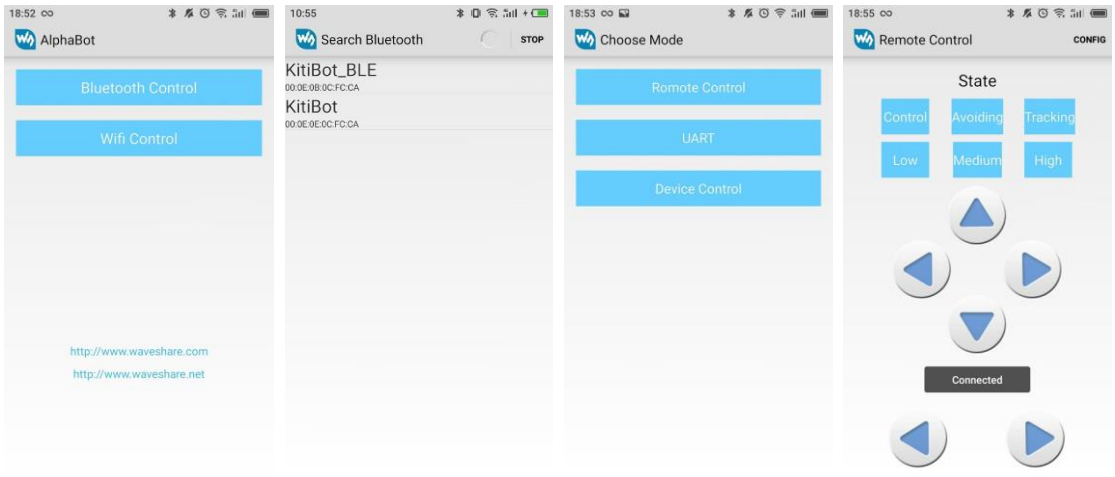


Figure 18 Bluetooth Control

SECTION 4 PROGRAMING MODE

In the last section we introduced how does the robot work using the factory program. You can use the factory program to try further to control your robot. The factory program mainly

controls the robot by pressing buttons, with no need of any programming. We will show you how to programmatically control you robot.

If you want to control the robot through graphical programming, you must download the firmware to the main control board beforehand. Connect the robot to the computer with the USB cable, select the serial number and connect the serial port.

Click "Upgrade Firmware" and upload the program until the upload is complete.

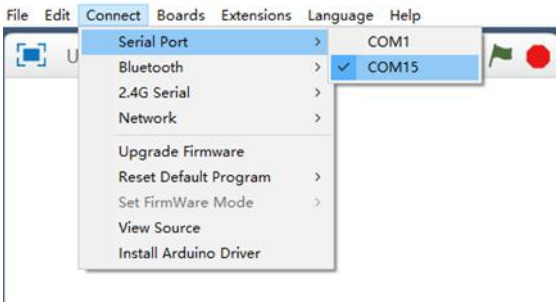


Figure 19 AlphaBlock Serial Port Connect

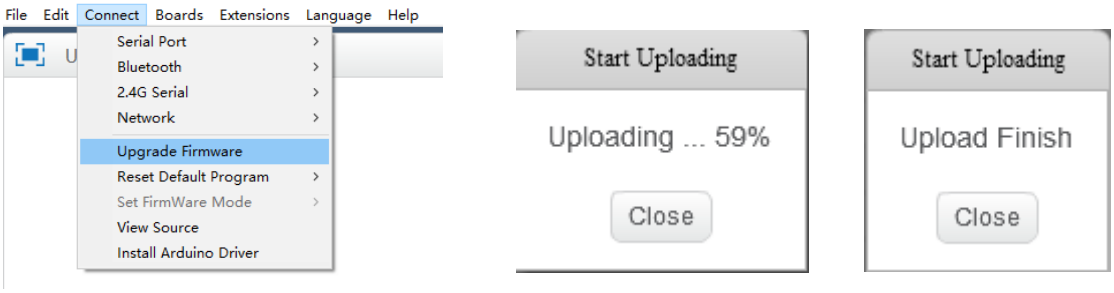


Figure 20 AlphaBlock Upgrade Firmware

Select "Robots" module in the "Scripts" column. It will appear a green dot in the robot module, indicating that the robot has been connected. If this dot appears in red color, then means the connection is not successful.

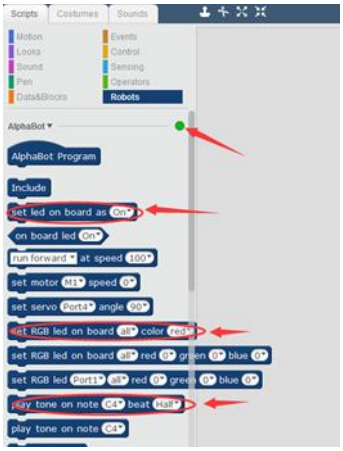
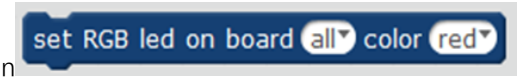


Figure 21 AlphaBlock Script Robots

Look for the 

building block, change "off" to "on", and click on the block. The main control board's green LED lamp will be bright.

By clicking on 

block, the RGB LED on the main control board will show red.

Clicking on 

block, the buzzer on the main control board will sound.

Connecting the robot with a USB cable will limit the movement of the car on the ground, which is very inconvenient. This connection is only used when the firmware must be upgraded or when the factory settings needs to be restored. It can be connected by Bluetooth connection alternatively.

It has been introduced that the mobile phone APP can control the car via bluetooth. Similarly, AlphaBlock can connect to a robot via bluetooth. (For this, the computer must have Bluetooth function, if there's no such function, it can be added a USB Bluetooth module adapter.)

Click on the menu "Connect" - > "Bluetooth" - > "Discover" to search for Bluetooth module

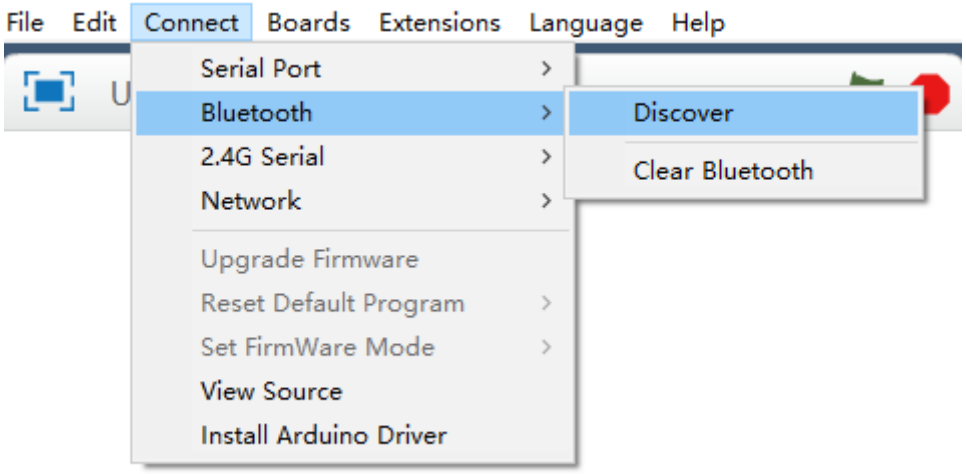


Figure 22 AlphaBlock Connect Bluetooth Discover

When you find a KitiBot (the device address begins with 00:0E:0E) click "connecting". The software will show the progress of the connection status.

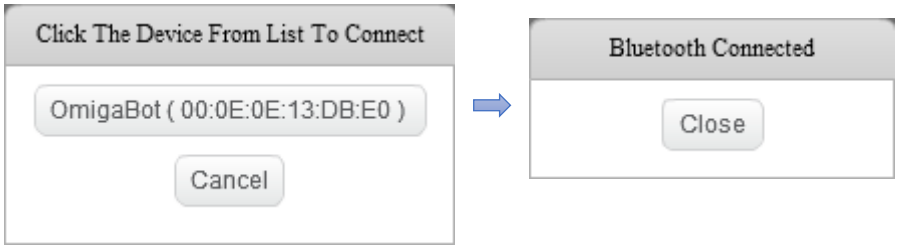


Figure 23 AlphaBlock Bluetooth Connecting

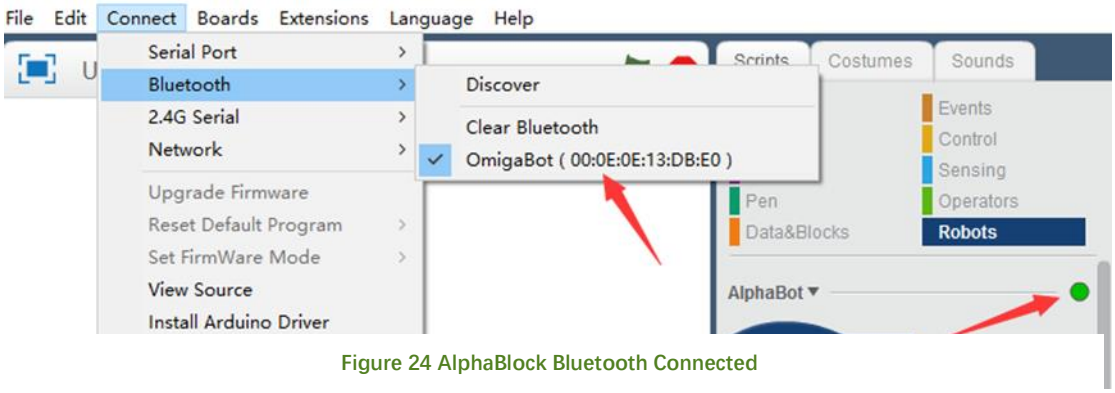


Figure 24 AlphaBlock Bluetooth Connected

At this point, a Bluetooth module will appear in the list, which shows the green dots in the robot module, indicating that the Bluetooth has been successfully connected. You can click on the block module, as shown above, to see if the robot has a response.

By this step, everything should be ready. Let's start off our robot programming journey!

CHAPTER 1 YOUR FIRST ALPHABLOCK

AlphaBlock is a software developed based on Scratch 2.0, and Scratch is a programming language that can be used to design animations, games, music and art works. Scratch is very easy to use and can be easily used by dragging and dropping blocks, without the need for programming experience.

But Scratch software can't be connected to the physical world, and AlphaBlock is adding a robot module to it, increasing the robot's operating blocks, and controlling the robot through building blocks.

Let's talk about how to write a program with AlphaBlock.

SECTION 1 – SCRIPTING WITH BLOCKS

There are many program modules in the middle section of the interface, which are divided into ten categories. Each category contains a lot of building blocks, and different classes correspond to different colors.

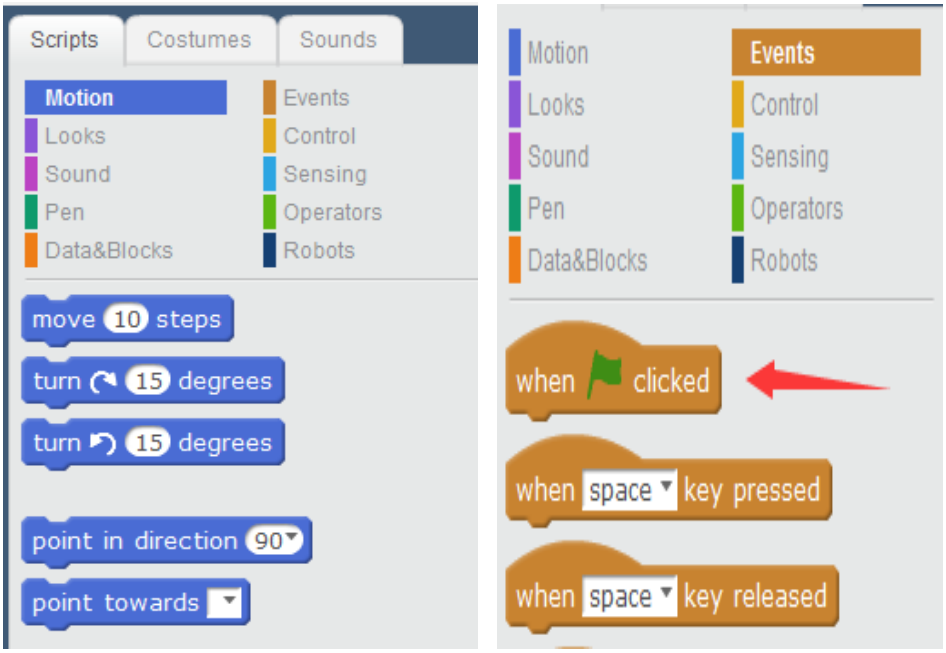


Figure 25 AlphaBlock Script Blocks

Click "Events" to find 

block, click on this block and drag it to the program editing area

Click "action" to find 

block and drag it to the program editing area under the previous program. So, the first program is written

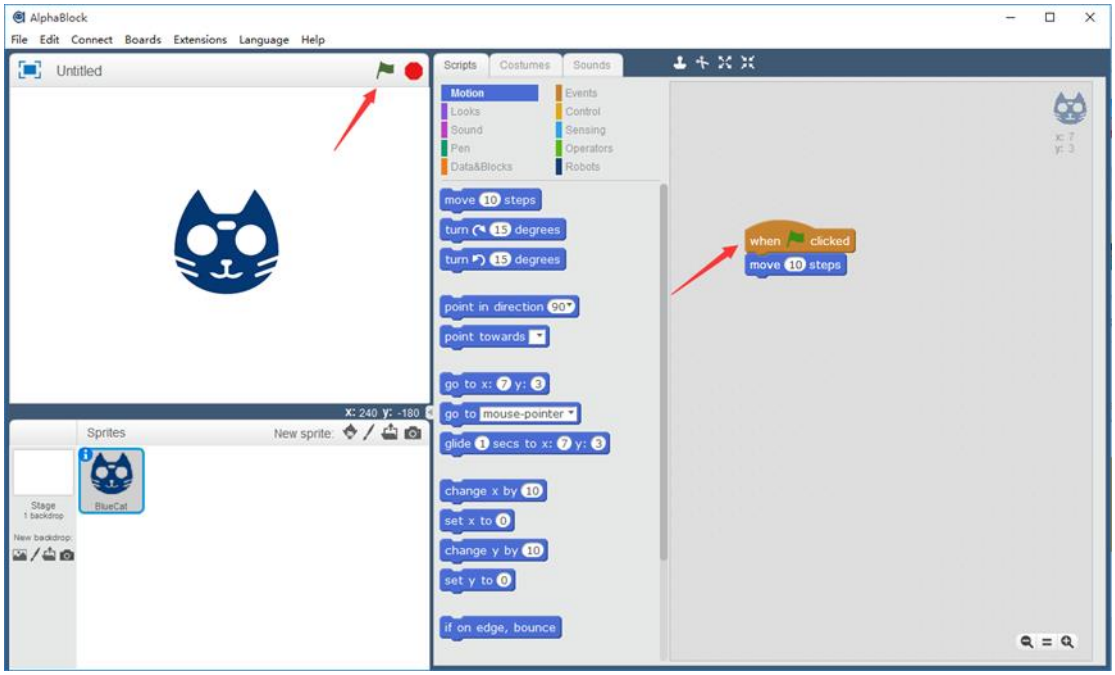


Figure 26 AlphaBlock First Program

Next, let's run this program to see if there is any effect, click the green flag on the stage or click on the 

block program. Every time you run the program, the blue cat image on the stage moves right 10 steps.

Change the value on the 

block will change the action. If change it to "-10", the cat will move left 10 steps when program is run.

SECTION 2 - USING BLOCKS TO BUILD A SIMPLE PROGRAM

Now, let's get familiar with the other blocks.

Click on "Look", find 

block, click on this blocks and drag it to the program you just built.

Click "Sound" to find 

block, click down and hold the block to drag it to the program.

Edit the program through the above steps as shown below. After clicking on

to run the program, the stage character on the left will move right 10 steps and say "hello" while the computer will make a sound.

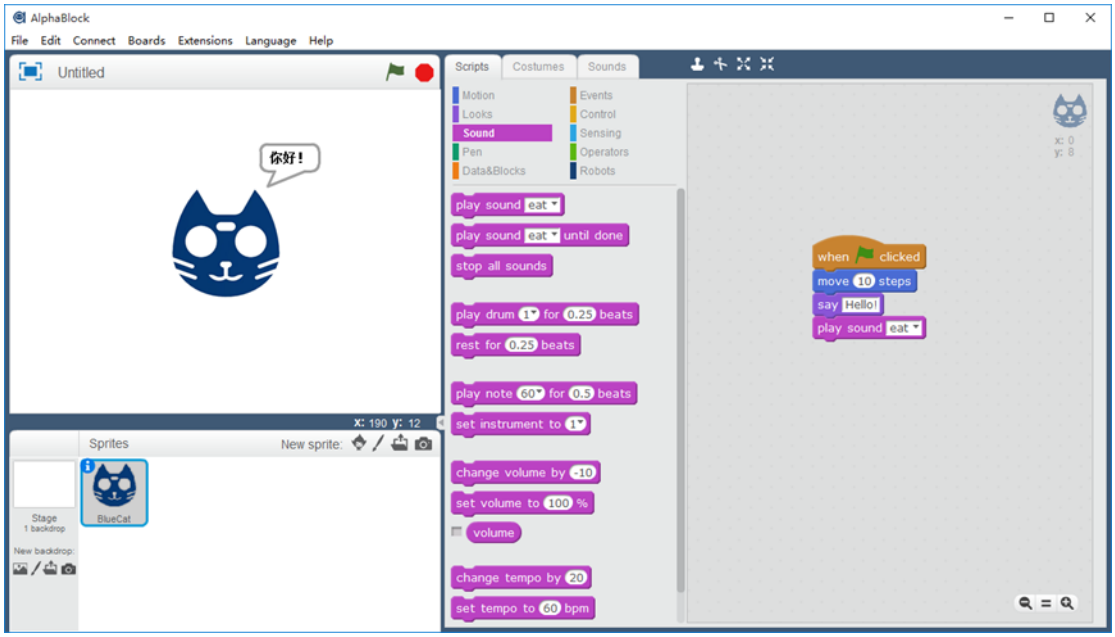


Figure 27 AlphaBlock Hell Cat

SECTION 3 - BUILDING BLOCKS TO CONTROL THE ROBOT

In the previous two sections, the animation was achieved through the program. Many students would ask, how is the robot controlled?

First of all, we click the "File->Save Project" to select an appropriate path to save the project. After saving the project, it can be re-opened and used again. Then click "File-New " to create a new project. There is a "Robots" in the program bar, which is used to control the robot. The next tutorial will use the building blocks in this class, which will control the robot by writing different programs.

According to the method we introduced in the second section, find the following blocks to write the program

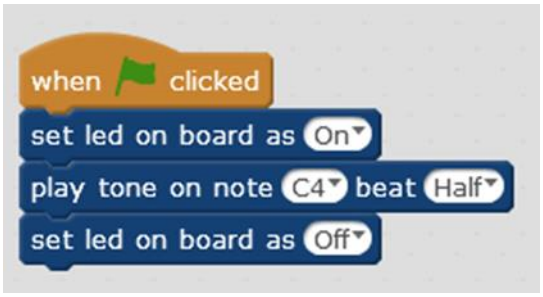


Figure 28 AlphaBlock Control Robot

Click on the green flag to run the program. You should see the main control board of the green LED light flashing, and the buzzer make a tone sound.

If after happens you click to run the program, but it does not run, and the main board does not look to have any response to it, check if AlphaBlock is properly connected to the robot.

CHAPTER 2 LIGHTING UP A LED

The second chapter aims to learn how to control LED lighting and how to write its program.

SECTION 1 – LIGHTING UP

Script:

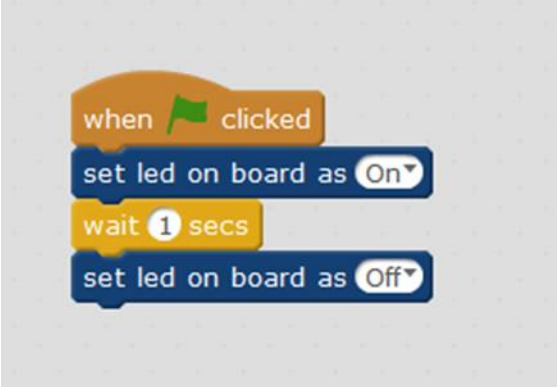
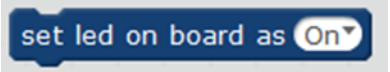


Figure 29 Lighting up script

Description:

: To start the event, click on the green flag to run program.

: Here, it waits for 1 second. The number can be an integer or a decimal



:Is a switch block that turns the green LED light on the main control panel on or off.

Expected Result:

When you click on the green flag on the first block, the program starts to run, and the green LED light on the main control panel will be lit. After waiting for 1 second, the LED lights will turn off. The whole process is that the LED light on the main control panel lights up for a second and turns off finally.

SECTION 2 – FLASHING THE LED

Script:

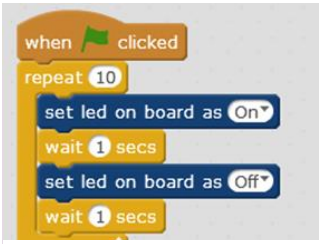
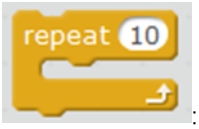


Figure 30 Flashing the LED script

Description:



: Repeated execution of a block set means that the program it contains is repeated. Through the use of this block, with our first section of the program, you can control the number of LED lights flashing.

Expected Result:

You can see the LED on the main control board flashes 10 times and then goes out completely

SECTION 3 – BUTTON CONTROL LED

Script:

Figure 31 Button Control LED Script

Description:

: Determine whether the state of the button on the main control panel is pressed or released.

: Using repeated execution, this block will always detect the button status. Making sure that every keystroke is tested.

: When the program is running, using this block, it will judge whether to light the LED according to the state of the button. This block indicates that if a condition is met, the "if" part of the block is executed. Else, if does not met the condition, the "else" part of the block will be executed instead.

Expected Result:

After running the program, when the button on the main control board is pressed, the LED will be lit. At releasing the button, the LED will turn off.

Thinking:

Question: What happens if you do not use the “repeat execution” block to run this program?

Answer: The repeat module is required because the program need to detect the status of button all the time. Without the repeat block, the program will only detect the button’s status one time and stop, after that even you press the button could not control the LED anymore.

CHAPTER 3 RGB LED

In this chapter, we mainly learn how to control RGB LED lighting and how to display different colors with RGB LED lights.

In the previous chapter we learned how to control LED lights. But you may be wondering, are there only 3 fixed colors on the main board? Can we display other colors?

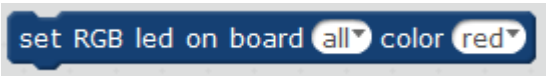
Of course, this is not possible. However, the main control board has two RGB LED lights, with them we can display any color. Let us have a try!

SECTION 1 DISPLAY THREE CLOROS

Script:

Figure 32 Display Three Colors Script

Description:



:Sets the color of the left and right RGB LED lights on the main control board, which can display red, orange, yellow, green, blue, violet, white and black colors.

Expected Result:

When the green flag is clicked, the two RGB LED lights on the main control board will display red, green, blue and white colors for 0.5 seconds.

Tips:

0.5 seconds is equivalent to half a second. 1 second equals 1000 milliseconds, and 0.5 seconds is 500 milliseconds. This number and unit are all together a very short time span, so actually this time is very short.

SECTION 2 RANDOM COLORS

In the beginning of the third chapter we said that, RGB LED can display any color, but in the previous section we say that you can display the red, yellow, green, blue, purple, black and white nine colors. What does this mean? And what exactly is RGB LED light?

In fact, the RGB LED can mix three different colors, which are: Red, Green and Blue. We all know that different two colors can also blend into different color compositions. This is why it can display any color.

Script:

Figure 33 Random Colors Script

Description:

: Constantly repeats the program is wrapping around

: Wait until it meets the conditions. The condition is set in the space.

: Determines if the button on the main board is "pressed" or "released"

: Sets the color of RGB LED lights, each color range is 0 ~ 255. To turn it off, all three colors must be set to 0

: This block allows you to generate random integers. It can be set a range for the random integer result.

Expected Result:

After running the program, every time you press the button on the main control panel, RGB LED will randomly display a color.

Thinking:

Question: How can the RGB LED automatically display a color every 0.5 seconds?

Answer:

Figure 34 Automatically Display RGB LED every 0.5s

Tips:

The LED on the main control board is a three-color light mode (RGB color model, also known as red and green blue color model), which is an additive color model. The colors of Red, Green and Blue are added in different proportions to produce a variety of colors.

From the two right pictures about we can see:

Red + green = yellow

Green + blue = blue light

Blu-ray + red = magenta

red + green + blue = white

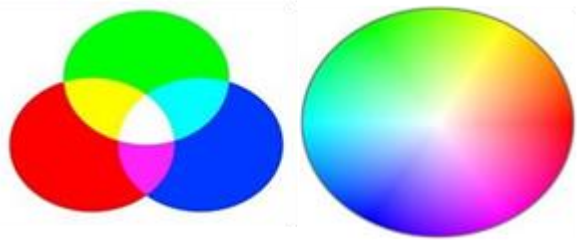


Figure 35 RGB

Try:

Fiddle with setting the RGB values to let the LED display different colors. Now, how to display yellow, purple, and white lights? Give it a try!

SECTION 3 BLINKING ALTERNATELY

Script:

Figure 36 Blinking Alternately Script

Description:

: Select the LED that you want to control with "all/left/right". The number represents the brightness of the corresponding LED (value range 0~255). Notice that, the larger is the number, the brighter is the LED light.

Expected Result:

First, the left LED light shows blue color, and then shuts down after 0.1 second. At the same time, the right LED light displays blue color, and right after 0.1 second, the light is switches off. You should see this process repeating itself.

Thinking

Question: Why we use 

to turn off the LED lights? What happens if you change and run the program to the following?

Answer:

On the board, these two LED lights are individually controlled, keeping constantly their current state (for instance, if you only turn the left LED light on, then its state will remain lighting up). At this very time, if turn on the right LED light, the program won't change the state of the left LED, then you will see both LED lighting up, without flickering effect.

Therefore, in order to achieve the effect of alternately flaring the LED lights on the left and right sides, it is necessary to add a program to turn off the left LED before the right LED lights up. Similarly, the LED on the right should be off before turning on the left LED.

Use this block in the sample program to turn off all on-board LEDs. Of course, we can also change the program to this, with the same flashing effect.

CHAPTER 4 MUSIC

In this chapter we learn how to control the buzzer's pitch.

SECTION 1 PLAYING TONES

Script:

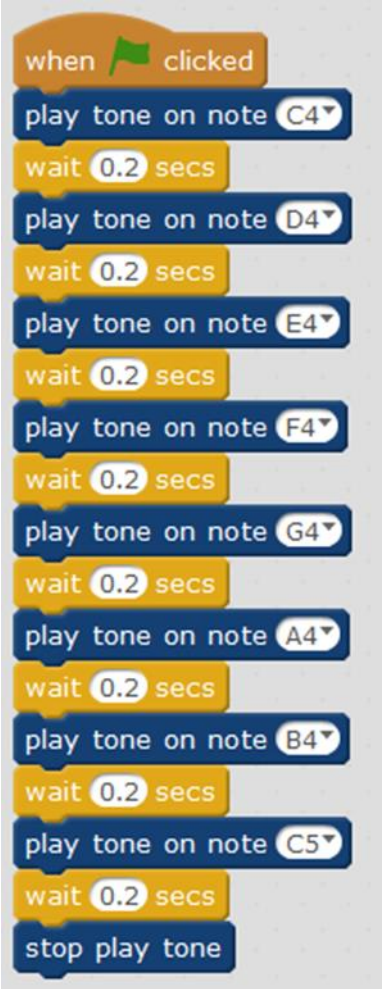


Figure 37 Playing Tones Script

Description:

: The buzzer can output tones between C2 ~ D8. You can set the beat to one-eighth ~ double-beat duration. The buzzer will stop when the time set for the beat is over.

: You can play any melody. But then, If no stop control is added, the buzzer will keep its sound even after the program has stopped. Therefore, you need to add a stop block to the playback as well as add playback time.

: With this blog, it will stop the buzzer playing.

Expected Result:

The buzzer will emit seven tones at a time (do, re, mi, fa, sol, la and si).

Note: If the program does not end with the "stop playing block", the buzzer will sound all the time.

Tips:

Sound is produced by the vibration of an object. We produce sound speaking with different frequencies using our vocal cords vibration. In this same way, the buzzer also works not differently. The main control board sends different levels of high and low frequencies to the buzzer, so it emits a different tone sound. By changing the tone in the block you can change the sound.

This block has two pull-down menus that define the pitch and beat of the note being played.

1) In the tone menu, pitch classes are represented by seven letters C/D/E/F/G/A/B. C corresponds to the major octave of the set Do/Re/Mi/Fa/So/La/Si. The number after the nominal letter represents a different octave level. For example, C4 is the standard octave (or middle C), C5 is one octave higher, and C3 is lower one octave.

2) Within the “tempo menu” we can play a beat sound length as follows: 1/2, 4/1, 8/1, 1 whole of a beat and double beat, being the whole beat 1 second, its half is 0.5 seconds, a double beat 2 seconds, and so on.

音名	唱名	简谱
C	DO	1
D	RE	2
E	MI	3
F	FA	4
G	SO	5
A	LA	6
B	TI	7

Figure 38 Tones

SECTION 2 PLAY MUSIC

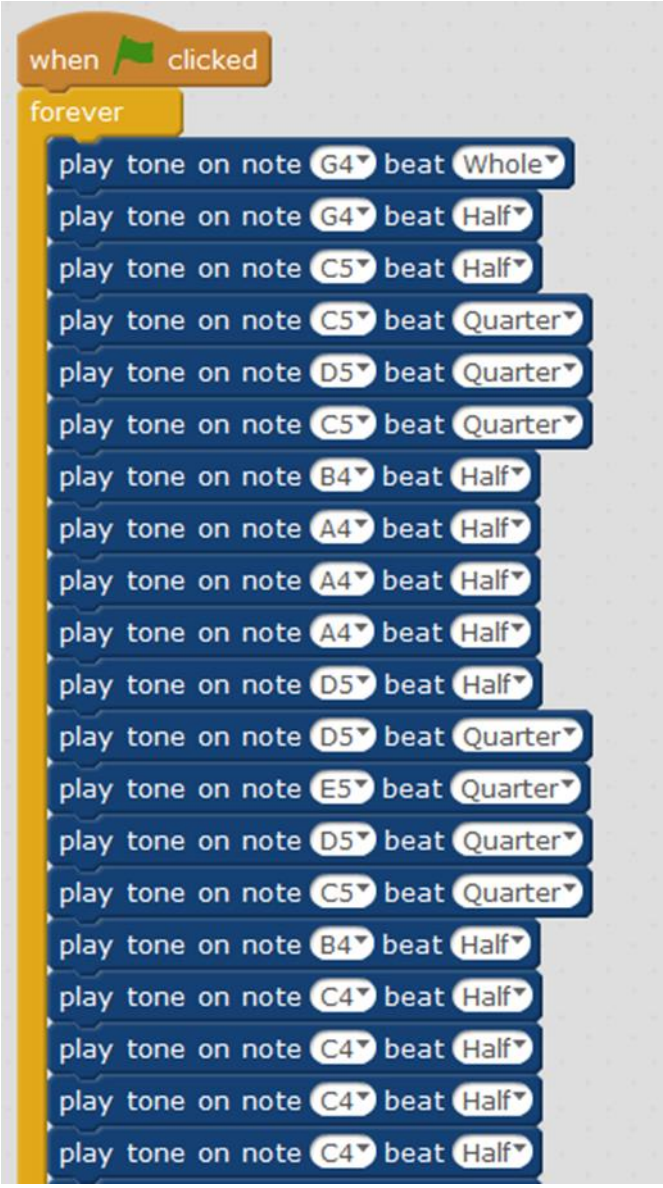


Figure 39 Play Music Script

Script:

Expected Result:

After clicking the green flag to run the program, the buzzer of the control board will play music according to the tone the program mades.

Try:

Try to make small car RGB LED can displays different colors when playing different music, making the effect more cool and gorgeous. Let´s make some fun!

SECTION 3 ELECTRONIC KEYBOARD

Script:

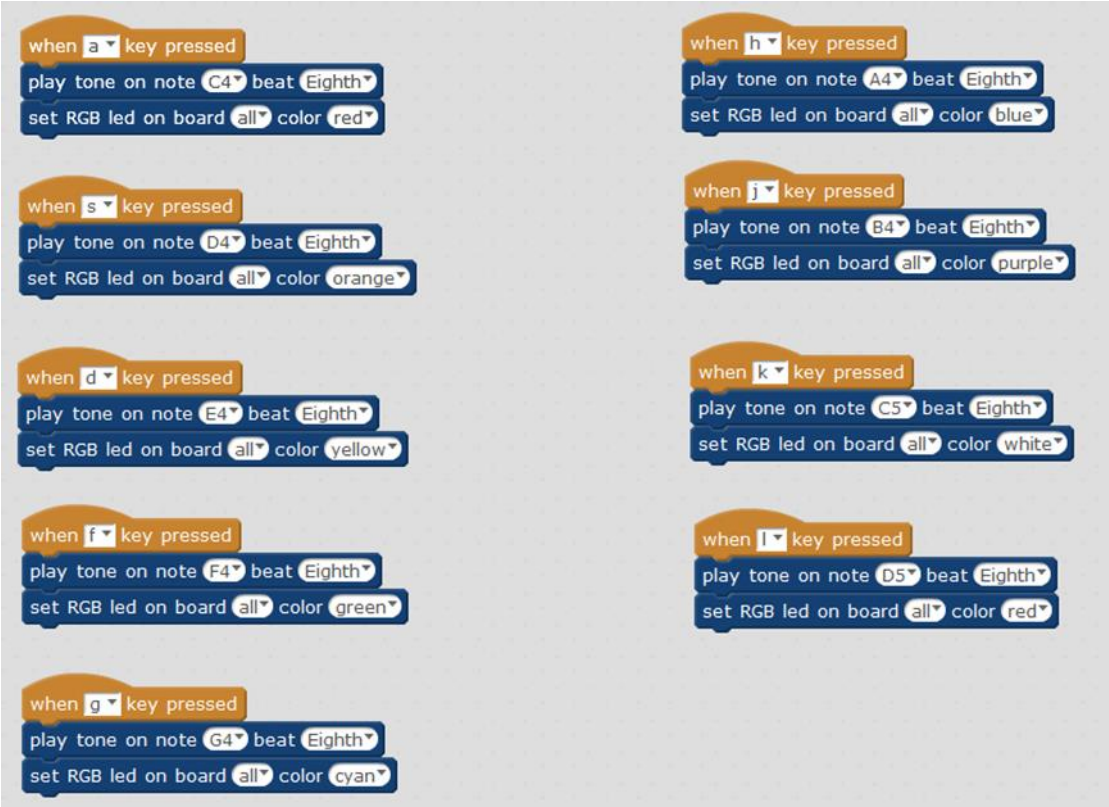


Figure 40 Electronic Keyboard Script

Description:

: Event button. After pressing this button, the following program will run.

Expected Result:

By pressing at each A S D F G J K or L letters on the keyboard, the buzzer emits a different sound, and RGB LED lights will display different colors.

SECTION 4 AMBULANCE SOUND

Script:

Figure 41 Ambulance Sound Script

Expected Result:

Each time a pushbutton is pressed, the onboard buzzer will play an ambulance sound, while the RGB LED light will alternately display the red and blue colors. Ambulance sound consists of high frequency and low frequency sound, being the high frequency sound of 950 hz, and low frequency sound playing at 700 hz. The duration of high frequency is 0.6 seconds, low frequency duration of 0.4 seconds. High and low frequency alternates between them.

SECTION 5 FIRE TRUCK SOUND

In the previous section we simulated an ambulance sound with our buzzer, and in this section we'll simulate the sound effects of fire truck. The sound effects of fire truck are: low-frequency sound that ranges from 650Hz to 750Hz and high-frequency sound in the interval of 1450Hz to 1550Hz, which rises from the low frequency to the high-frequency sound, takes 1.5 seconds, and then the high-frequency sound drops to the low-frequency sound, which takes 3.5 seconds, back and forth. From this, it can be programmed as follows:

The low-frequency sound is set at 700Hz and the high frequency sound is 1500Hz, making the buzzer sound on 3 step intervals (700Hz - >1500Hz - >700Hz). Because time duration of "the low frequency to high frequency", "high frequency to low frequency time" is equal to 1.5:3.5 = 3:7, in under the premise of each sound duration is consistent, need to "increase frequency of the low frequency to high frequency" : "the frequency of high frequency to low frequency drops" to 7:3, again by adjusting the frequency and time every voice increase: decline range, can simulate the fire engine sound.

Script:

Figure 42 Fire Truck Sound Script

Description:

Create a new “f” variable to store the current sound frequency

This program is divided into two parts: Its first part raises low frequency to high frequency.

Set an initial tone frequency value to 700Hz and begins playing it. At each instance of the loop, the frequency increase 35Hz and plays again, continuously increasing to more than 1500Hz, then when it gets out of the loop, it no longer increase frequency.

Similarly, the second part decreases from high frequency down to low frequency. Set the tone frequency initial value to 1500Hz and plays it. At each loop instance, it lowers frequency 25Hz and plays it again, until it drops below 700Hz out of the loop, then no longer reduces its frequency.

: Defined a pitch frequency value, and set each sound for 12 milliseconds. The value 12 here is for reference only, you can adjust the value as you wish

Try:

Try to simulating police car sound effect. The low-frequency sound is in the range of 650Hz to 750Hz, and the high-frequency sound is in the range of 1450Hz to 1550Hz. The low frequency sound reaches the high frequency for 0.23 seconds, and then drops back to the initial low-frequency sound, which takes 0.1 seconds.

CHAPTER 5 RUNNING KITIBOT

In the previous chapters we have learned how to control LED lights and buzzers. As we might presume, you will be eager to know how to control the car and make it move. In this chapter we'll learn how to do so!

SECTION 1 LET'S MOVE THE CAR!

Script:

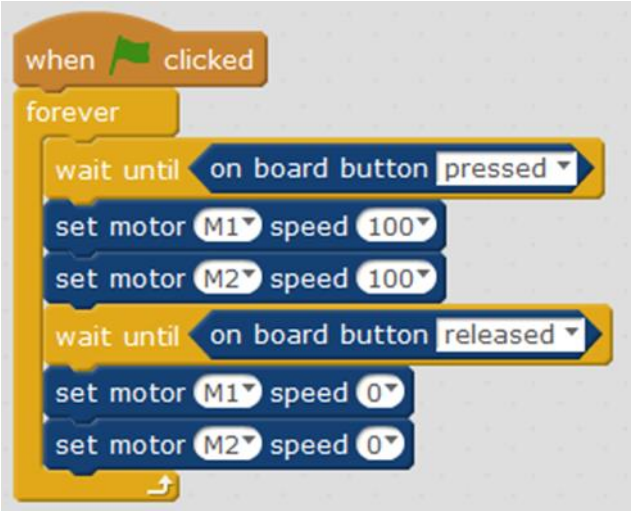


Figure 43 Let's Move the Car Script

Description:

: This block set motor speed, being range value from 0 to 255. 255 stands for faster wheel spin. -255 value represents the reversed wheel rotation (moving backwards) and 0 means stop.

: This block can set the direction of the robot motion. It can be set to move forward, backward, left turn or right turn.

Expected Result:

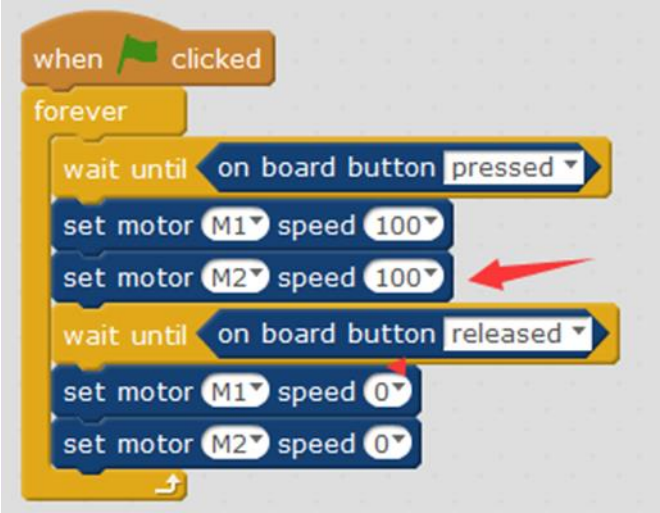
When you click on the green flag, the program starts running. If the program has got a block "cycle" it will continue to run repeatedly. Until the key on the main control board is pressed, the left and right motors will be moving on forwarding rotation speed of 100. When the button is released, the left and right motors will stop rotating. Children can try to press the button on the main board to see how robot is moving.

Try:

1. If the left wheel is turning, the right wheel stops. How will the robot move? If the wheel is turning on the left and the right wheel reverses. How will the robot move?

2. Modify the script so that the robot can move forward, backward, left and right.

Tip: Setting M2 value to 0 will stop the left wheel. Settings it to a negative integer will make the motor turn backwards.



SECTION 2 CONTROL KITIBOT BY KEYBOARD

In the previous section we made the robot moving, but it was not convenient to control the buttons on the main control board, and it was only possible to operate in one direction. It would be nice if you could control the robot to do different exercises. In the last chapter, we can implement the effect of the electronic piano through the computer keyboard, is that the same or can control the robot movement through the computer keyboard? Now let's try the keyboard to control the robot.

Script:

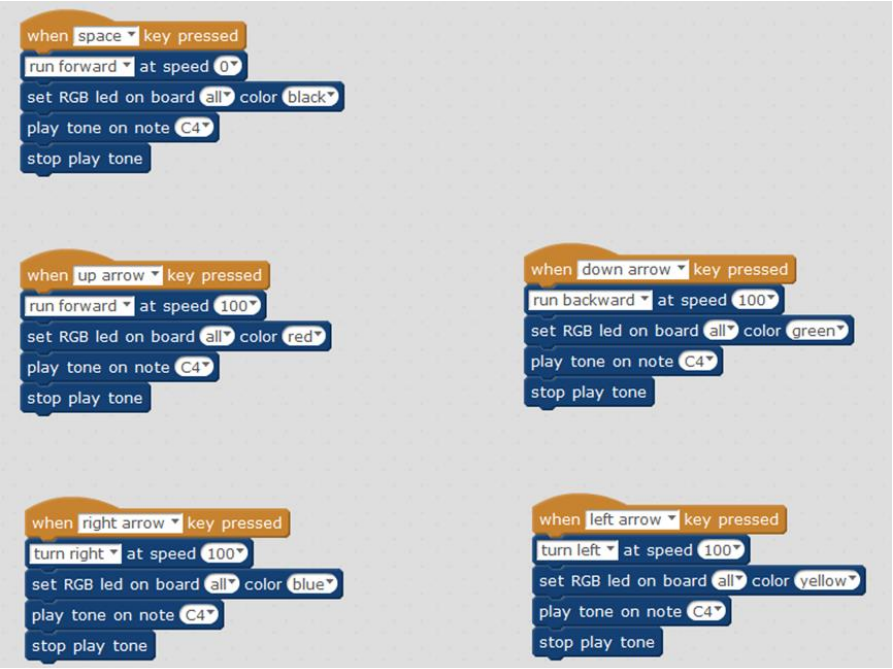


Figure 44 Control KitiBot by Keyboard Script

Description:

This time, the program will have five key events, which are five buttons to control the robot to move around and stop. And at the same time the RGB LED will show a different color, and the buzzer will play.

Expected Result:

Pressing the arrow keys on the computer keyboard will control the robot movement. Pressing the space bar will stop. At the same time, the RGB LEDs will display different colors according to the different movement directions. The buzzer will sound at every key press.

Note: At pressing the button the speed shouldn't be too fast, or it is easy to make a mistake.

Tips:

When the program outcome turns out to be an issue, and is not according to the expected situation, we say that the program has loopholes, also called bugs. In this section of the program if the button is pressed too fast, it will throw a bug error, after we encountered this, if we modify the program to fix any loophole, we called it debugging (fix).

SECTION 3 DEBUGGING

Now let's analyze why the previous program is going to get an error when the button is pressed “too fast”. In the next program we will set five separate key events for a movement (operation). When the button is pressed, the operation corresponding to its event will run. If this operation has not been completed on the press of another button, then two key operations will send a command to the robot. The robot will confusedly think “what event/operation should I run now?” If you want to solve this problem, you only need to ensure that each key event is processed and then determine whether the next button is pressed on the line.

Script:

Figure 45 Debugging Script

Thinking:

Why does this program won't fail on pressing a second key event button too fast as the last program?

CHAPTER 6 OFFLINE MODE

In this chapter we will learn how to use offline mode. And understand the difference between offline mode and online mode.

We have already learned the control of LED lights and buzzer in the previous chapters, as well as how to move the robot. And by setting the blocks, we let the robot move. We don't know if the kids found out that the previous program needed to be connected with a computer to run, which is called online mode. The online mode requires a computer to connect to a robot via a USB cable or a Bluetooth connection. If the computer does not support Bluetooth, then the car will need to connect to the computer via the USB cable, so the car movement will not always be convenient. In this video we'll learn how to use an offline module to control a robot.

SECTION 1 ONLINE MODE

Before learning the offline mode, let's take a look at how the online model works. The online model, by definition, requires a computer to connect to a robot, either via a wired connection or through a wireless connection. In online mode, the AlphaBlock software and the robot are required to communicate continuously. Each block is equivalent to a command, and AlphaBlock software sends different commands to the robot so that the robot can complete the corresponding operation.

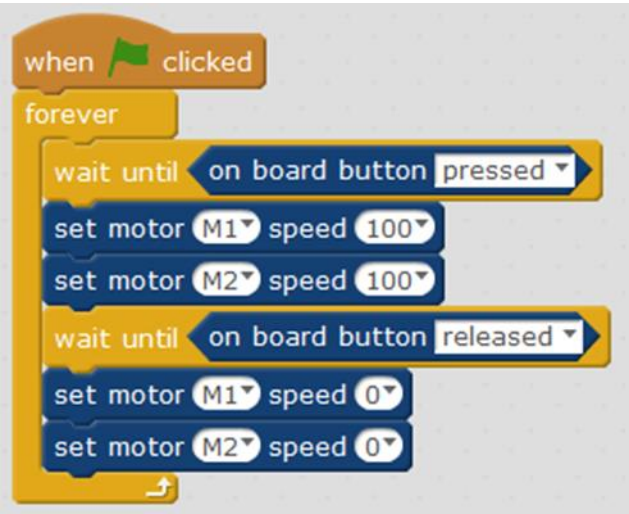


Figure 46 Online Mode

After the program starts, AlphaBlock sends a command to the robot, asking whether the button is pressed, and if the button is pressed, the AlphaBlock sends two more commands to set the left and right motor to make corresponding actions. So the online mode, AlphaBlock and the robot are going to be constantly exchanging data. AlphaBlock will send various commands to the robot, until the robot responds with the corresponding action.

Click "Edit" to select "Arduino mode" and run the program. You then might see that AlphaBlock software and the robot will continue to transmit data after the program is started.

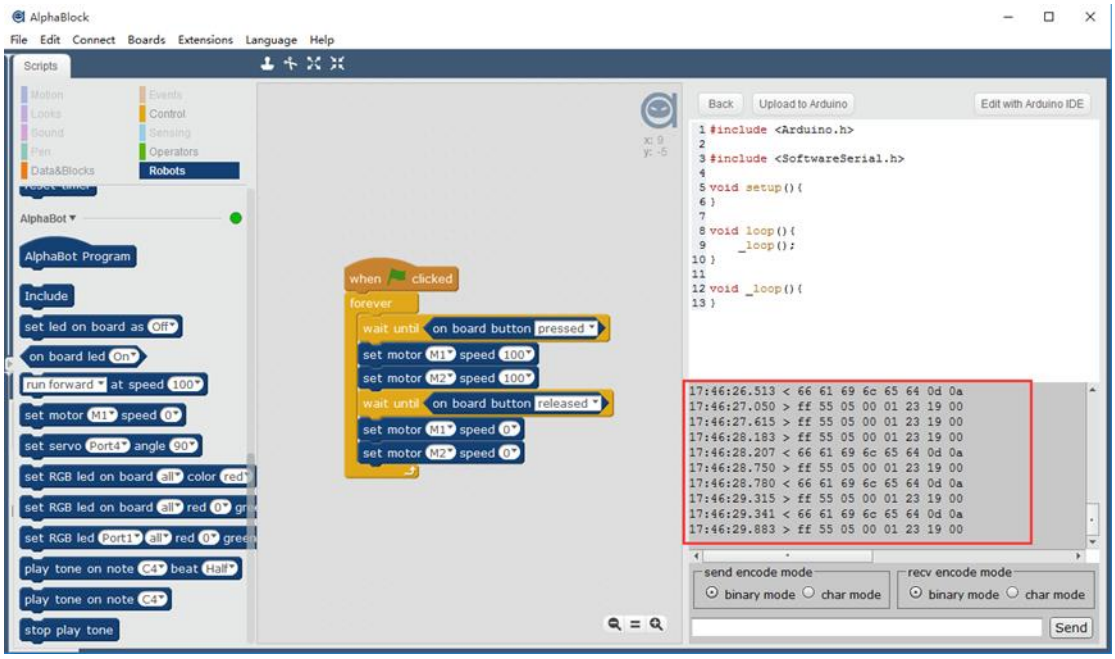


Figure 47 Data Transmission Between AlphaBlock and Arduino

SECTION 2 OFFLINE MODE

In the previous section it has been introduced the working principle of online mode. In this next section we introduce the working principle of offline mode. The online mode requires constant communication between the AlphaBlock and the robot, constantly sending commands to control the robot. So off-line module as the name suggests is not necessary to communicate, the robot itself can complete the appropriate action from the program.

Script:

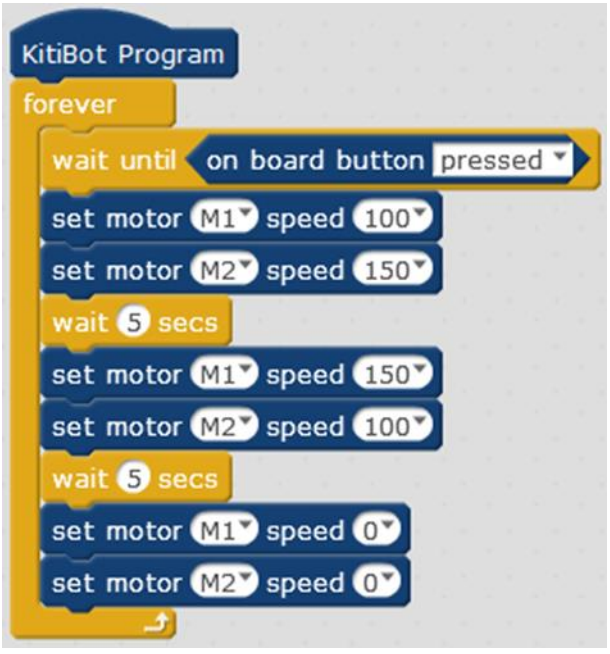


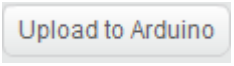
Figure 48 Offline Mode Script

Description:

Let's change the previous program to the offline mode. Change 

to 

as an offline mode.

You can see that the program in the Arduino code editing area has changed, and click 

upload the program to the robot. The following serial port will pop up information, compile the program and upload it to the main control board. Notice that Only once the program is finally uploaded, then program operations will start.

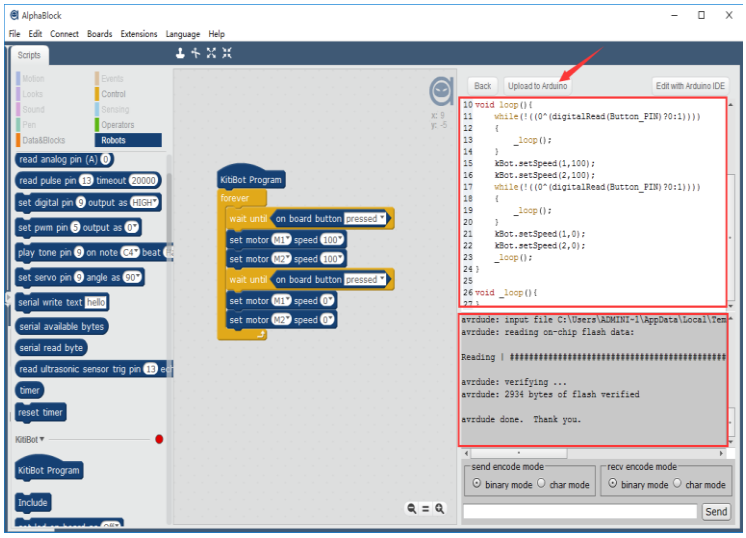


Figure 49 Program Uploading for Offline Mode

Expected Result:

Disconnect any connection between the AlphaBlock and the robot, and you can find that the program can still work. The car can still move when the button is pressed. In offline mode, the program needs to be compiled and downloaded into the main control board. The car is completely controlled by the main control board and is not required any communication from/to AlphaBlock. When the online mode is used again, the main board needs to install the firmware.

SECTION 3 ONLINE MODE VS OFFLINE MODE

ONLINE MODE

Advantages: You can interact with the robot, and it does not need to compile and download the program.

Disadvantages: the robot needs to be connected to the AlphaBlock and is not easy to use, and communication needs to take up more time. Subsequently, the program will run slower.

OFFLINE MODE

Advantages: The robot does not need to connect to AlphaBlock to operate, it can run independently, and the program can operate faster.

Disadvantages: The robot no longer interacts with the AlphaBlock, and most of the bricks cannot run in offline mode.

Try:

What are the programs in the previous section that can be changed to offline mode?

You can give it a try!

SECTION 4 S-SHAPED LINE FOLLOWER

Script:

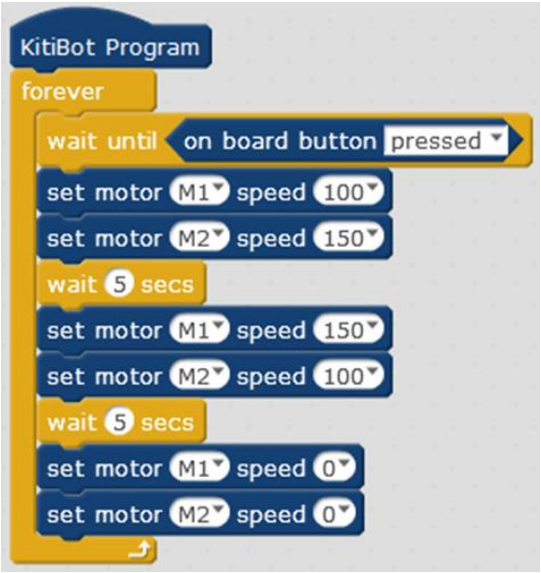
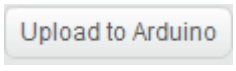


Figure 50 S-shaped Line Follower Script

Description:

Switch to the Arduino code editor area and click 

to upload the program to the robot.

This sample program allows the robot to follow the s-shaped trajectory. Control the robot movement by controlling the speed of the left and right wheels of KitiBot respectively. Use

"differential" speed to control the robot's turn. When the right wheel speed is greater than the left wheel speed, the robot turns left. Conversely, when the left wheel speed is greater than the right wheel speed, the robot turns right. If the program does not achieve the s-type effect, adjust the delay parameters in the program.

Expected Result:

After pressing the button, the robot can run along the s-shaped track.

Try:

1. Let the robot take a circular path.
2. Modify the rotation (wheel spin) speed of the left and right motor (for example, increase the difference between two values) and observe the walking state of the robot.

CHAPTER 7 GENERAL BASIC OF PROGRAMMING

In this chapter we explain the three basic structures of a program, which are usually not used alone, but rather combined. By combining the three basic structures to achieve the program allows the robot to achieve the effect we want.

The three basic structures of our programs are “sequence structure”, “conditional structure”, and “cyclic structure”. We've already used these structures in the first three chapters, and we'll talk about that in more detail.

SECTION 1 SEQUENCE STRUCTURE

The program executes from the first building block, then executes the last block, which is the sequential structure order.

The right picture shows a standard sequence structure. When the program starts, A, B and C are executed in sequence. And finally terminates.

Sequential structure is the basis to follow by any program operations/commands.

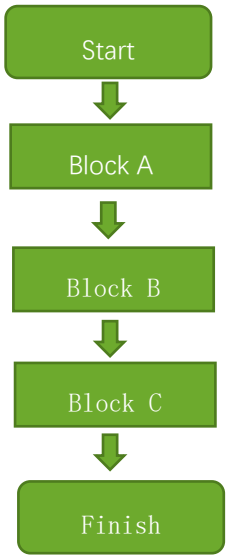


Figure 51 Sequence Structure Flow Chart

Script:

Figure 52 Sequence Structure Script

Expected Result:

After running the program, each block is executed in turn, and the robot moves forward, backward, left and right.

SECTION 2 CONDITIONAL STRUCTURE

In the process of running, the program will judge whether a certain condition is satisfied or not. According to the judgment structure, different procedures are executed. This is called “the conditional structure”.

The one that best represents the condition structure is the following block structure set.

If the <condition> is satisfied, then it executes the first section. Contrary, if the condition is not satisfied then executes the following second section.

On the right is the corresponding flow chart. After the program starts, the condition is determined. If the condition is satisfied, the building block A command is executed.

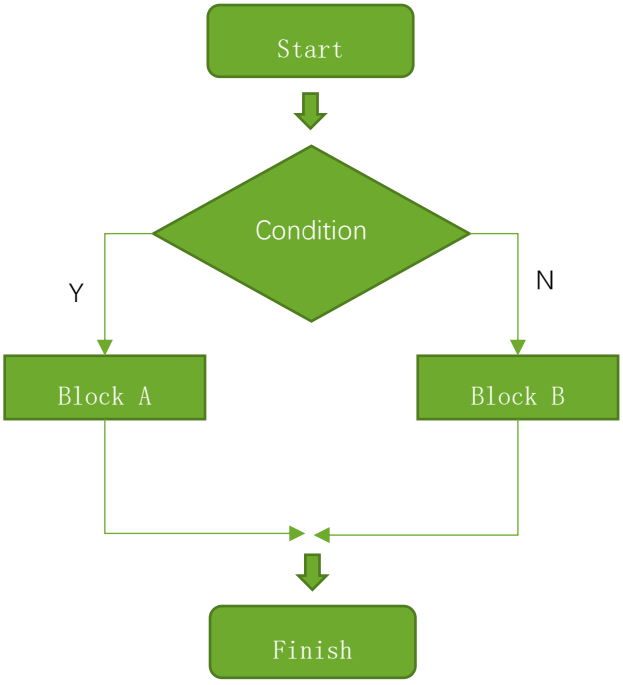


Figure 53 Conditional Structure Flow Chart

Script:

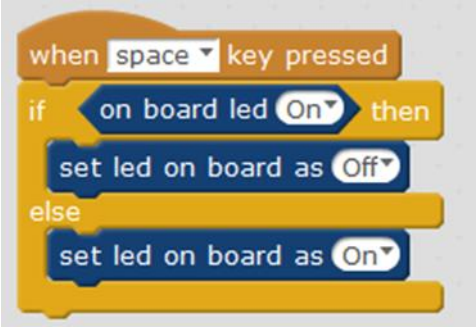


Figure 54 Conditional Structure Script

Expected Result:

After the program starts, it will judge the status of the green LED on the main control board. If it is on, then the above statement will be executed to turn off the LED. If it is off, execute the following statement to turn on the LED. Each time running the script, it will be based on the status of the LED to perform different operations, changing the status of the LED.

SECTION 3 LOOP STRUCTURE

The loop structure, as the name suggests, loops around a certain section. If his condition is not yet satisfied, the loop operation will be performed until his condition is satisfied.

As the flow chart shows in the picture on the right, the program starts by determine its conditions. If the condition is not meet, it will run block A. After A section is over, it will go back again to determine whether its condition is meet. And so on, until its condition expectedly meets for withdrawing from looping, to find the end of the program.

These has to do with the loop structure shown below,

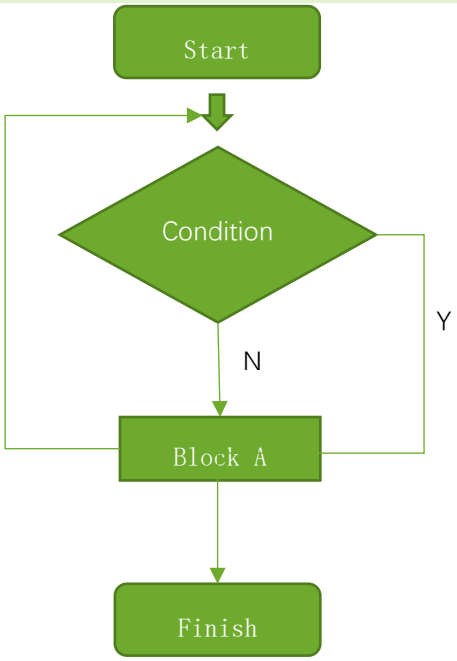
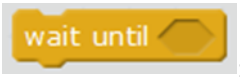


Figure 55 Loop Structure Flow Chart

: Repeat the execution of the enveloping blocks for the specified number of times, to its completion

: The constant repetition of the cycling block is the equivalent of a condition that is always satisfied and never jumps out of the cycle.

: Repeat the execution of the enveloping blocks until the condition is satisfied.



: Keep waiting until the condition is satisfied

Script:

Figure 56 Loop Structure Flow Chart

Expected Result:

After running the program, it will continue executing the section statement of repeated bricks. The RGB LED will flash repeatedly and an alarm sound will be generated, until the key on the main control board is pressed that will break the cycle. Only then, RGB LED will be turned off and the buzzer will stop to sound.

CHAPTER 8 GAME NUMBER

In this chapter we're going to learn how to use math and variables

Tips:

: The four operations include addition, subtraction, multiplication and division, it can be filled in with numbers and variables.

: It can generate random number among the interval.

: With compare operators, you can compare the size of variables and values.

: Logical operators

: Variable is a container for storing data, you can modify the variable value as required.

SECTION 1 ARITHMETIC

Script:

Figure 57 Arithmetic Script

Expected Result:

After running the program, you can see that the cat face will say the calculation results

Try

1. You can change the operation to subtract, multiply, divide, and see what the output looks like.

2. Try another the value to see if the calculation is correct.

SECTION 2 LOGIC OPERATORS

Script 1:

Figure 58 And Logic Operators Script

Description 1:

: When both conditions are satisfied, "and" blocks are formed.

Expected Result:

When the button "a" and "s" are pressed, the buzzer will turn the RGB LED red. If you press "a" or "s" alone, there will be no response.

Script 2:

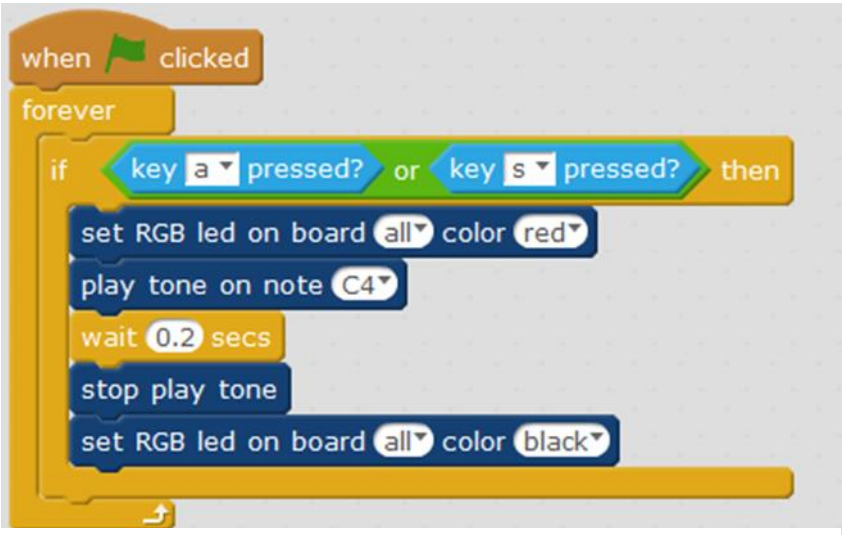


Figure 59 OR Logic Operators Script

Description 2:

: Whenever any of two conditions in a block are satisfied, "or" blocks are formed.

Expected Result 2:

After the program runs, we can find that the button "a" and the "s" button will ring as long as one is pressed, and the RGB LED will turn red.

Script 3:

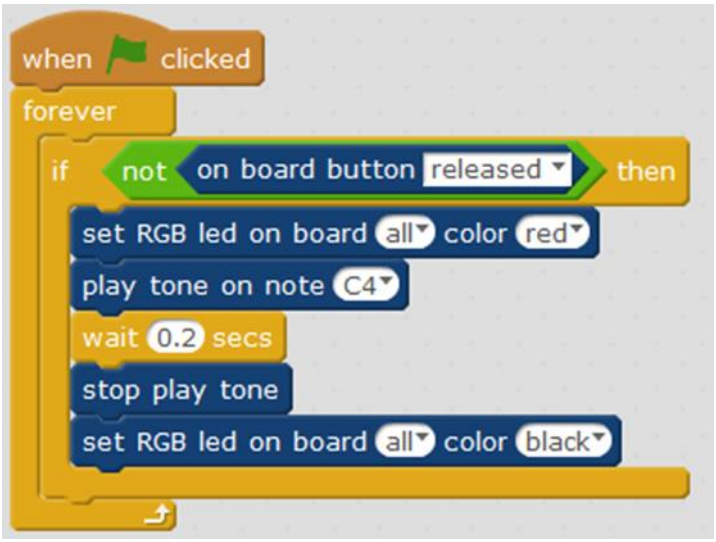


Figure 60 NOT Logic Operators Script

Description 3:

: The function of "not forming" blocks is to reverse the condition, which is, if the condition is not valid, then it run the following program.

Expected Result:

If the onboard button has not been released, then the button is pressed and the program keeps running. As long as the button is pressed, the buzzer will sound and the RGB LED will keep in red light.

SECTION 3 VARIABLES

A variable is a container for storing data, we can modify the value of the variable anytime

Let's take a look at how a variable is used by using a button to control RGB leds to display various colors. Through RGB LED section, we already know that RGB leds display different colors by setting RGB values

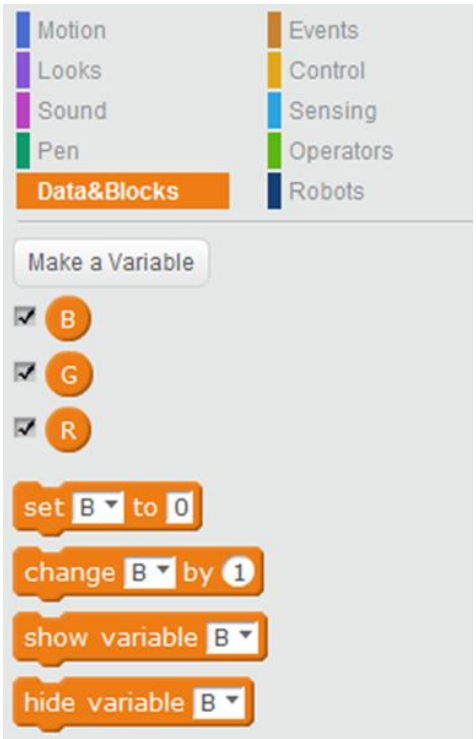
: Set the color of the RGB LED light. Each color range is 0 to 255. To turn it off, you must set all three colors to 0.

Now, the requirement is to control RGB values via the keyboard, the key number "4" will increase the R value by 5, and the key number "1" will be reduced by the R value.

The key number "5" increases the G value by 5, and the number key "2" reduces the G value by 5.

First, click "make a Variable" to create new R, G, and B variables. The values of red, green, and blue are stored separately.

Click on the right side of the block, in the variable field. Its top left corner shows the current value of the variable. The initial value of the variable is 0.



Script:

Figure 61 Variables Script

Expected Result:

Click the green flag to run the program. By pressing the key number "4", you can see the RGB LED displaying in red, if the key number is not "4", then the display of red will be brighter. On the stage you can see that R is getting much and much bigger. You can see the different colors of the RGB LED by other numeric keys, and the values of R, G and B on the stage will changing as well

SECTION 4 COMPARATION OPERATORS

There is a bug in the previous section of the program. The value range of RGB is 0~255. In the previous program, if you hold down the key, then the value of R, G, and B variables may exceed this range. Below we control the values of R, G, and B variables within 0~255 by adding comparison operators. When the variable exceeds 255, the variable value will remain 255. When the variable is less than 0. The variable value stays at 0.

In this way, we can add the following blocks to the program in the previous section to limit the value of the variables to the range of 0~255.

Figure 62 Comparation Operators

SECTION 5 CUSTOMIZING BLOCKS

If you combine the previous two programs, you will find that the program is getting longer and longer, which makes it difficult for us to view the program.

Below we can define the operation in the previous section as a module by customizing the module instructions. This way it will be more convenient for us to use.

Click "new module instruction" to create a new building block and input "Range Control"

Figure 63 Customizing Blocks

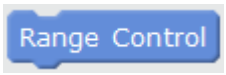
In the script bar there is 

block, and in the code editor will appear 

starting blocks

Script:

Figure 64 Customizing Block Script

After we've defined 

block. We add it as equivalent of adding a script from the left. This way, it works like defining a function for range control, and the function is to adjust the values of R, G, and B variables, so that they don't go beyond the specified range.

This can be done by calling this block in the main program, which is very convenient.

We are already familiar with how to program through the previous chapters. You might be feeling that programming is not as difficult as imagined, aren't you? In the following chapters we will learn how to use various sensors and how to implement various functions for the robot.

CHAPTER 9 LIGHT SENSOR

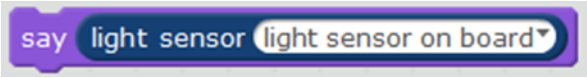
In this chapter, we'll learn how to use light sensor.

SECTION 1 LIGHT SENSOR

Script:

Figure 65 Light Sensor Script

Description:



With 

block, the value of the light sensor is displayed in the form of speaking characters in the stage area.

Expected Result:

The current intensity of light will also be displayed on the stage area. If you approach your hand to the light sensor, you should see that this value is slowly getting smaller. If you illuminate the sensor with a flashlight getting closer to it, it can be found that the value is slowly increasing.

Tips:

Here we taking the light sensor as an example to show how to collect the sensor's value in real time and display it, other sensors values are collected in the same way.

1. Sensor

Sensors are electronic components used to detect events or changes in the environment and send information to other electronic devices. In the process of program operation and debugging, it is often necessary to collect the value of the sensor in real time to help us understand the ambient light, sound, distance and other information.

Value range of light sensor is 0 to 1000. The value when exposed to daylight is over 500, and in the evening is between 0 and 100. Within indoor lighting condition is around 100 to 500

2. The forever repeat block

This block can ensure that the light sensor displays the current sensing value in real time, otherwise it will only continue to display the values that are sensed in the beginning.

3. Wait block

In addition, if the ambient light intensity is not stable (e.g., fluorescent lamp), sensor values can be seen changing rapidly (real time application runs fast, according to the ambient light intensity reading), this block can be added to let the number value change at a slower pace, to be more clear at first sight.

SECTION 2 USING THE LIGHT SENSOR

Script:

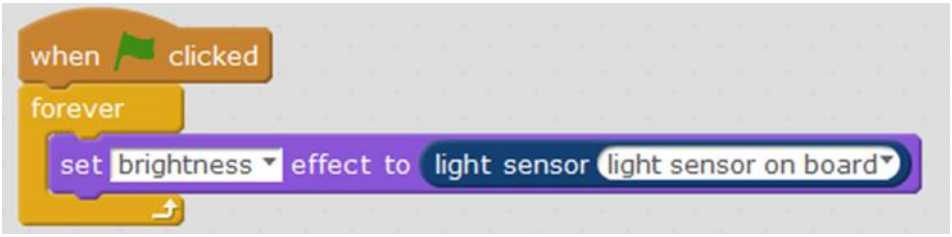
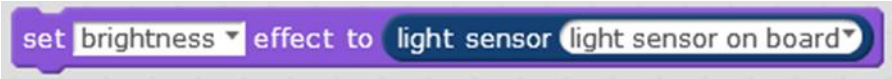


Figure 66 Using The Light Sensor Script

Description:



This block set the stage brightness to the value of the light sensor. The brightness range of the stage background is from negative 100 to 100. When the brightness increases from -100 to 100, the background gradually changes from black to pure white.

The value range of the light sensor is 0 to 65535. The higher is the brightness of the surrounding environment, its value will be bigger. The brightness value of sunny days is about 100 to 200.

Expected Result:

By putting your hand slowly closer to the light sensor you will notive the blue cat's color is slowly getting darker.

Try:

Stop the program a moment. Change "brightness" by selecting other features, such as delay, ultra-wide angle lens, etc. Now, resume the program to observe new effects.

SECTION 3 LIGHT SENSOR CONTROL RGB LED

Script:

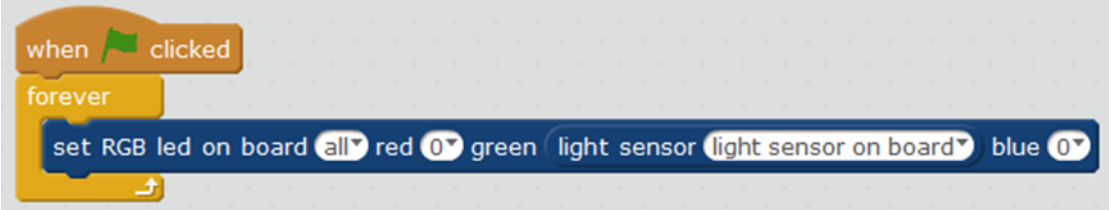


Figure 67 Light Sensor Control RGB LED Script

Expected Result:

Try putting your hand closer again to the sensor. You should see how RGB LED light is slowly dimming until it gets turned off.

CHAPTER 10 IR REMOTE CONTROLLER

In this chapter we are going to learn how to control a robot with an infrared remote control.

infrared remote control is kind of infrared emitting device.

Pressing different buttons, the remote control will send different infrared pulses to the main control board, and the main control board will process the infrared pulse data to get the key value of the corresponding key.

:

Determines if the key of the infrared remote control is pressed,

You can select different buttons in its drop-down menu.

Figure 68 IR Remote Controller

SECTION 1 CONTROLLING LED WITH IR REMOTE CONTROLLER

Script:

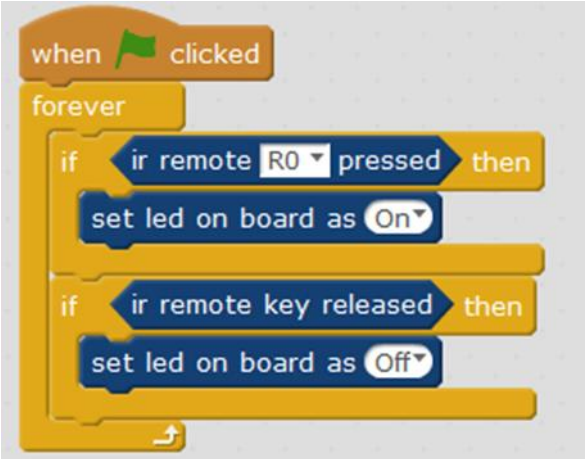


Figure 69 IR Remote Controlling Script

Expected Result:

Use the remote control to control LED lights. When the "0" button is pressed, the LED light will be lit, and when the button is released, the LED light will turn off.

SECTION 2 CONTROLLING RGB LED WITH IR REMOTE CONTROLLER

Script:

Figure 70 IR Remote Controlling RGB LED Script

Expected Result:

At pressing different buttons on the remote control, the RGB LED light will display different colors.

SECTION 3 CONTROLLING ROBOT MOVING

Script:

Figure 71 IR Remote Controlling Robot Moving Script

Description:

After the program runs, the loop structure will be executed to continuously check whether the key of the remote control is pressed.

Expected Result:

Press any key number 2, 8, 4, 6 and 5 of the infrared remote control to control the robot forward, backward, turn left, and turn right and stop. At the same time, RGB LED lights should display different colors, and buzzers should emit different sounds.

Try:

You can arrange the corresponding program for the remote control according to your preference, such as pressing the 0 button to make the robot to produce a sound.

SECTION 4 KEY STATUS OF IR REMOTE CONTROLLER

In a previous program, ever time was pressed a key the robot could complete the full operation. But sometimes, on pressing forward while the robot does not stop yet, the robot can respond moving faster instead. That could make the robot might reach outside the range of the control scope. How to fix all this up?

For the infrared remote control, there is a block that can determine whether the infrared button is released. Below, we solve the problems above by using this block we just mention.

Script:

Figure 72 Key Status of IR Remote Controller Script

Description:

Next, we will obtain the key status information of the infrared remote control. When no button is pressed, the condition is executed; otherwise the condition won´t valid. On the basis of the previous script, the robot will stop moving automatically when the key is released, so that the robot won't run away. This way, if the robot reaches too far of the infrared remote control, it will stop automatically.

Expected Result:

Pressing the button, the robot will make the corresponding action. Then release the button, the robot will stop moving.

SECTION 5 OFFLINE OF IR CONTROLLER

Script:

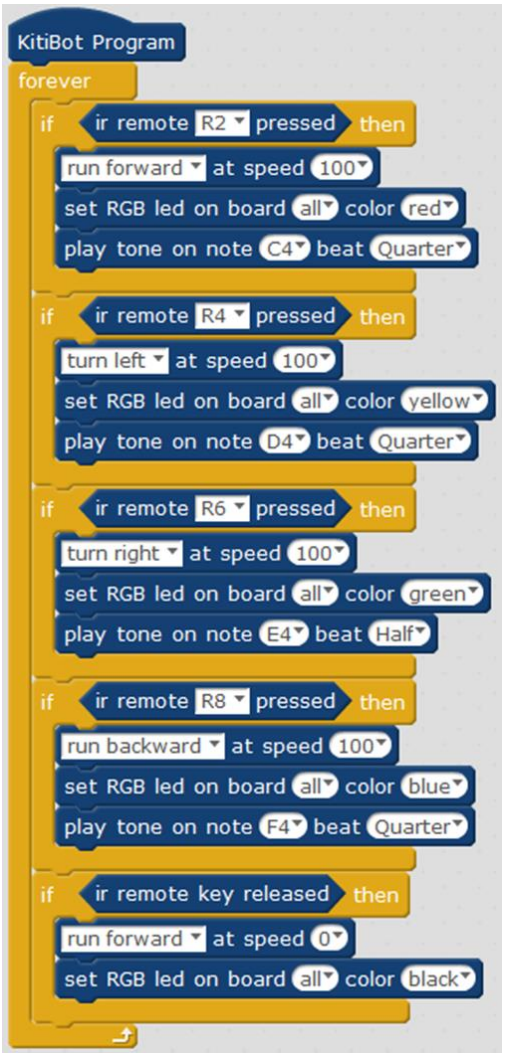


Figure 73 Offline of IR Remote Controlling script

Description:

Here, the program is changed to the offline mode so that the robot can run independently.

CHAPTER 11 MAKING ROBOT SHAKE ITS HEARD

In this chapter we learn how to control the steering gear as well as how to make the robot shake its head.

SECTION I THE STEERING GEAR

The steering gear is a servo control.

Figure 74 Steering Gear

The head of the robot is fitted with a steering gear and he can rotate a certain angle. Below is the block to control the steering gear.

: In this block there are 2 parameters. One is the port number, which the steering gear use to connect to the mainboard. You can choose up to 4 different ports. The other parameter is steering angle. Its range is from 0 to 180 degrees.

At 90 degrees, the robot looks forward;

At 0 degrees, the robot looks to the right;

At 180 degrees, the robot looks to the left.

Script:

Figure 75 Steering Gear Script

Expected Result:

When the green flag is clicked, the robot shakes its head quickly looking to the right. You can try to modify the angle values and control the robot to look in different directions.

SECTION 2 CONTROLLING BY IR REMOTE CONTROLLER

Script:

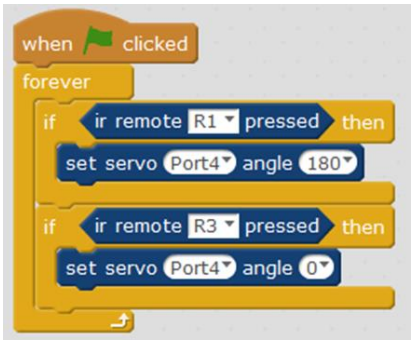


Figure 76 Controlling by IR Remote Controller Script

Expected Result:

Running the next program, when pressing the key number "1", the robot looks to the left, pressing the numeric key "3" and the robot looks to the right.

Try:

Add a few more keys to the program to control the robot to different angles.

SECTION 3 CONTROL THE TURN ANGLE

Maybe some children will ask, how come the robot turns around so fast. Could it slow him down a bit to stop at any Angle? The answer, of course, he can!

Script:

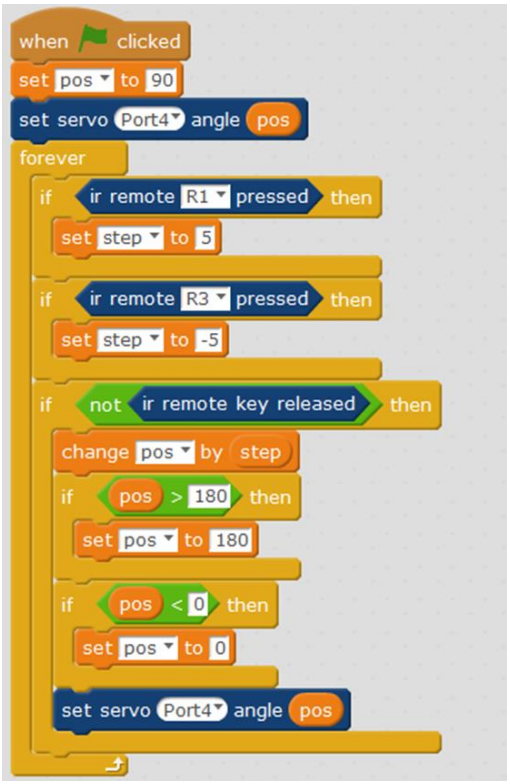
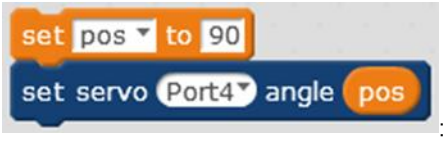


Figure 77 Control The Turn Angle Script

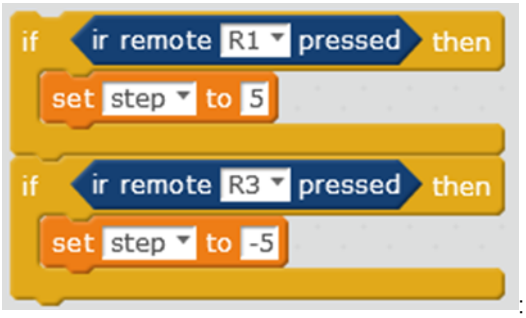
Description:

We divided the range of 0 to 180 degrees into several parts within an average. Pressing the key to turn the head, it will move to its next division. If the button is still pressed it will move to the next division and so on, until it reaches a top.

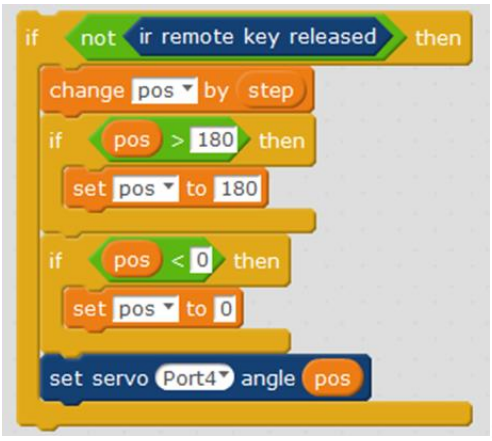
: Here, we first set both “pos” and “step” variables. Pos value represents the current position of the steering gear. Step represents the angle of each rotation. A positive number means a left turn. And a negative number means a right turn.



: The program starts by setting the “pos” to 90 degrees and turning the steering wheel to 90 degrees.That is, making the robot facing forward.



: Then, the program enters into an infinite loop, and determines whether the infrared remote control button is pressed or released. If the key number 1 is pressed, the step is set to 5, which means the robot looks to the left and turns 5 degrees each time. If the key number 3 is pressed, the step is set to -5, which means the robot looks to the right, turning 5 degrees each time.



: If the key of the infrared remote control is not released, it changes the current angle.

: This block increases the current angle “pos” by its current “step” value.

Note: If the step is a positive integer, the value of the POS increases, and if the step is a negative integer, the value of the POS decreases.

For example: when POS equals 90 degrees, then step equals -5. Similarly, 90 degrees “increased” by minus 5 (-5) is 85 degrees making the robot turns right.

The two “conditional blocks” are to determine if the angle is greater than 180 degrees or less than 0 degrees. If it’s about 180 degrees, the angle is set to 180 degrees, and if it’s less than 0 degrees, it always will be set to 0 degrees. Because of this, the robot head can only rotate within an angle range, which is from 0 to 180 degrees.

: This is the most important block, that transfers the steering gear to a current angle. This is the only block that can really turn the robot’s head left and right.

Expected Result:

When the infrared remote control key number 1 is pressed, the robot looks to the left side, increasing the angle 5 degrees each time. When number key 3 is pressed down, the robot looks

to the right side, reducing its angle 5 degrees. When the button is released, the robot stops turning.

Click to check both variables. You should see the current value of the variables on the stage.

Click the small green flag to start the program, and press on the infrared remote control the key number 1. You should see that “step” value is 5, so “pos” value continues to increase y step value, making the robot to look on his left. Release the button, the value won’t change, and the robot also stops turning.

Pressing the key number 1, you can see the step is -5, so “pos” value decreases according to step value. The robot then will look to its right. Releasing the key button, it will show same value, so the robot stops turning.

Thinking：

1. If you remove this block of the example above, will the robot move?

2. What is the effect of increasing and decreasing the number of steps of each change?

SECTION 4 OFFLINE

Script:

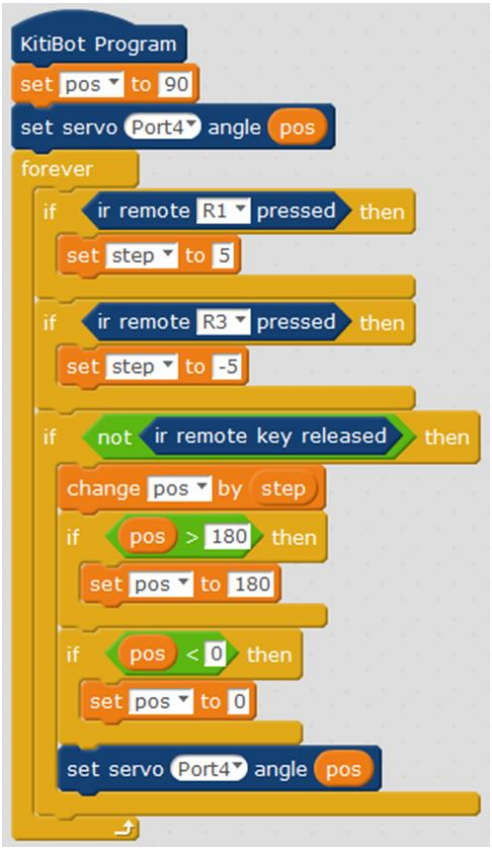


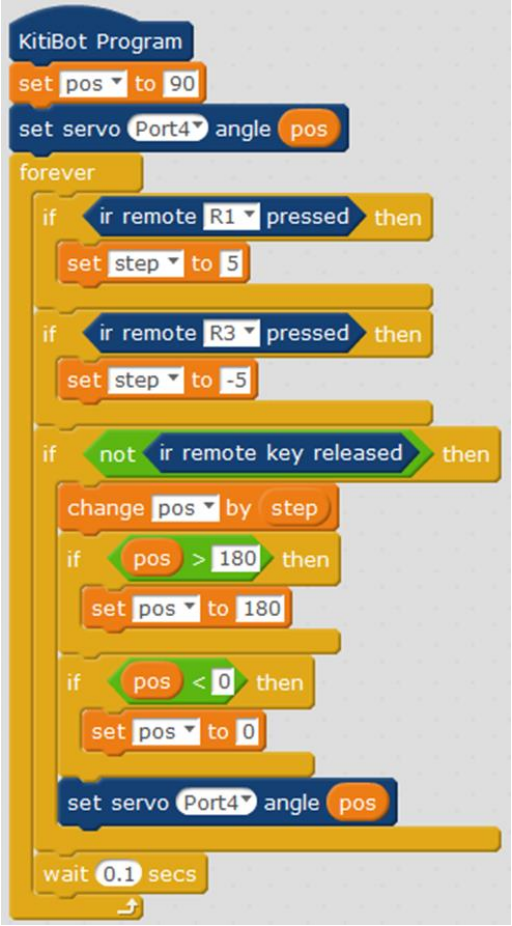
Figure 78 Robot Shake Heard Offline Mode Script

Expected Result:

After uploading the program, it can be found that pressing the key number 1 of the remote control, the steering gear will turn to the left right away, and the steering gear will rotate very quickly. In offline modules, robots do not need to transmit data frequently with AlphaBlock software, which will save most of the time. But at the same time the stage cannot show the current value of the variables.

Extension:

: A short delay of 0.1s can be added to the loop to slow the steering gear down a bit.



Thinking:

1. What parameters can you modify if you want to adjust the speed of the steering gear? Hint: delay time, step length.

2. In the previous chapter, we learned to move the robot through the remote control robot. This chapter we learned about how to use the remote control robot to shake its head. If you want these two functions to work together, how do you put them to work in one group?

CHAPTER 12 DODGING OBSTACLES WITH YOUR ROBOT

In this chapter, we are going to learn about how to avoid obstacles using ultrasonic

Figure 79 Ultrasonic Sensor

Often, our ears do not hear sounds above 20 kHz (we said in the previous section that the sound was generated by vibration, 10 kHz, which is equivalent to 20 thousand vibrations per second, which is 20,000 times). Sonic waves produced above 20 kHz are called "ultrasounds". Due to their amblyopia, bats flying at night emit such ultrasound. Bats rely mainly on ultrasound to locate any avoidable obstacle on their way. Sound waves are emitted by the bat's throat and received by their ears, according to the received sound waves. They can determine the position of the front object. Ultrasound sensor invention is based on the same method.

The two eyes of the robot is an ultrasound sensor module. One eye is sending, and the other one is receiving. This way, the ultrasonic sensor can detect the distance of the front object.

SECTION 1 USING ULTRASONIC SENSOR

Script:

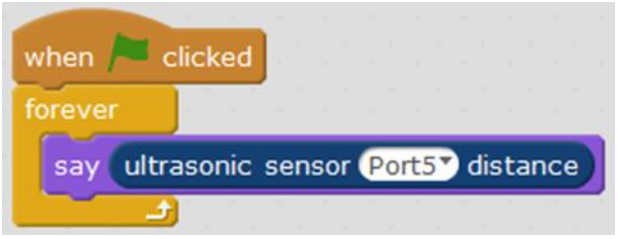


Figure 80 Using Ultrasonic Sensor Script

Description:

: this script can call the ultrasonic block to get the distance of the problem ahead. The unit given is in centimeters.

Expected Result:

After the program is started, it is executing in a cycle continuously. The value of the ultrasonic sensor is displayed in the window. If you place your hand in front of the sensor, you should see this value changing.

SECTION 2 ULTRASONICS OBSTACLES

In the preceding section, we learned how to use ultrasonic sound sensor. In this section, we will use the ultrasonic sensor to detect obstacles in front, so that the robot can avoid obstacles. If the ultrasonic sensor detects the distance in front of the object very close (for example less than 15 centimeters), we will make the robot turn and avoid the obstacle to move on.

Script:

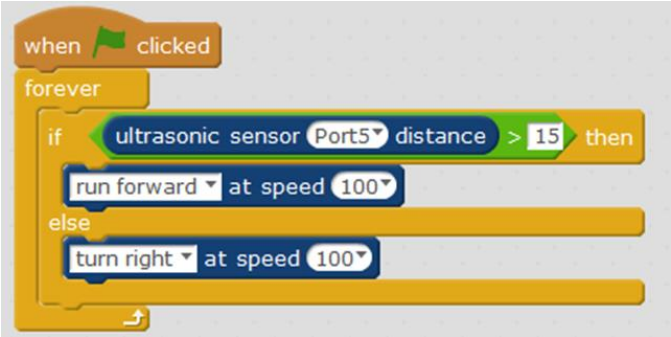


Figure 81 Ultrasonic Obastacles Script

Expected Result:

After the program starts, it will get into the loop structure, to constantly testing whether the ultrasonic distance measurement is larger than 15 centimeters. If the distance is over 15 centimeters, it will continue to moving ahead, otherwise, it will turn right to avoid obstacles.

SECTION 3 ADD CONTROL FUNCTION TO ULTRASONIC OBSTACLE

Script:

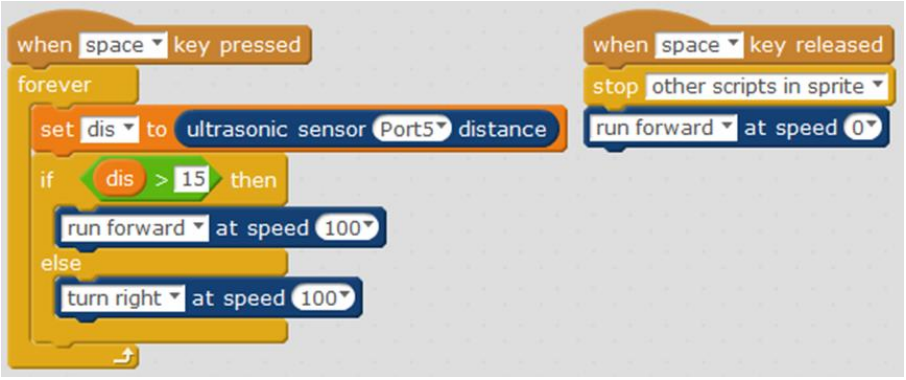


Figure 82 Controlling With Ultrasonic Sensor

Expected Result:

In the previous section, the program moves the robot around into any direction. At encountering obstacles we can turn, but the robot can't control and decided by itself. This time, we build on the previous program adding a space bar key for controlling, when the spacebar is pressed, the robot movement, and when it's released, then it just stops.

SECTION 4 OFFLINE

Of course, we can change the program to offline mode. This will eliminate the need for a computer connection.

Script:

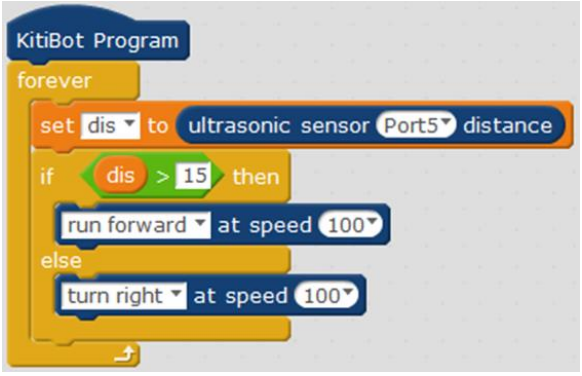


Figure 83 Ultrasonic Sensor Offline Mode Script

SECTION 5 CONTROL THE STEERING GEAR

Each time the robot encounters obstacles in front, the program decides to turn right. This time we add the steering gear control, the robot can determine the situation on both sides before deciding which side to turn.

Script:

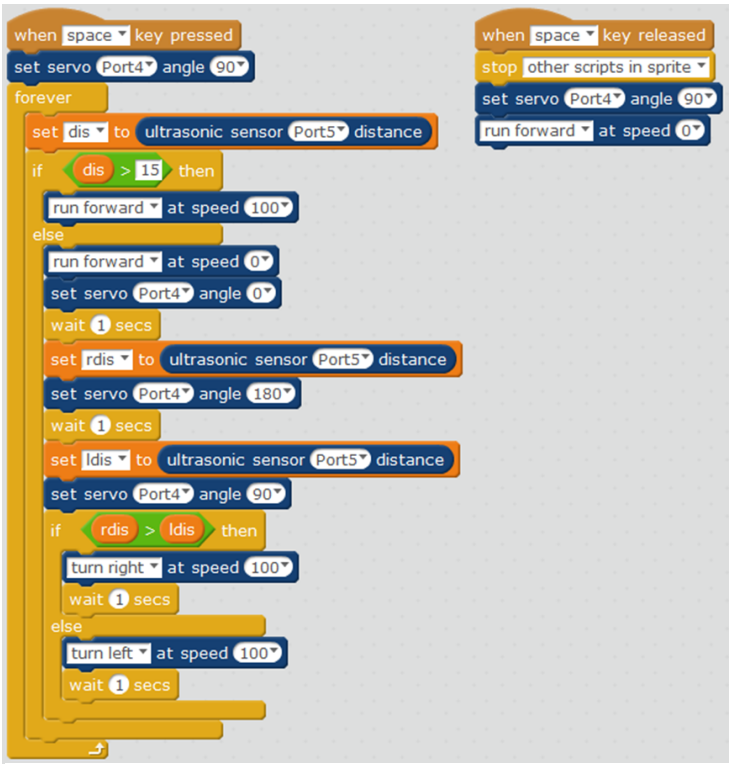


Figure 84 Controlling Steering Gear Script

Expected Result:

When the measuring distance is more than 15cm, the robot will go straight. When the measuring distance is less than 15cm, the program will control the robot to shake the head left and right, measuring again the distance between both left and right, and selecting whether to turn left or right according to the distance between left and right sides.

CHAPTER 13 LINE TRACKING

KitiBot, after seen a high-speed rail moving fast along the track, he hopes to have same track to move along, like a high-speed train. In this chapter we will learn about the “tracing function” of kitiBot. The aim of this chapter is to learn about the tracking sensor, and how to use it to implement the robot patrolling function.

Tracking sensor: the tracking sensor helps the robot move along the black line on the ground. There are five detectors at the bottom of the sensor, and the infrared LED light is used to project infrared light to the ground to detect the deviation of the module relative to the black line.

Figure 85 IR Tracking Sensor

SECTION 1 INFRARED TRACKING SENSOR

Script:

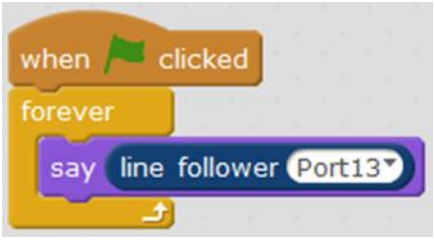


Figure 86 IR Tracking Sensor Script

Description:

: With this block, we can obtain the offset value of the tracking sensor. The numerical range is from 0 to 4000. 0 indicates that the black line is at the far left of the sensor, and 4000 indicates that the black line is at the right of the sensor, and 2000 represents the black line is not deviated, an is moving just in the middle.

Expected Result:

The program will be executed continuously in a loop, checking for the tracking sensor value, which is displayed in the window. Place the car on the black like and shake it left to right to see this value changing.

SECTION 2 TRACKING

When the value of the sensor is 2000, the black line is in the center. At this time, the rotation of the left and right motor should be set to move at the same speed. This way, KitiBot will move only forwards.

When the sensor's value is less than 2000, the robot is far to the right of the line. In this case, to turn sharply left, we set the right motor spin faster and the left motor spin slower.

When the sensor's yields a value more than 2000, the robot is far to the left of the line. We can do reverse the last case procedure, to turn to its right, we set the right motor spin slower and the left motor spin faster.

The larger is the deviation of the patrol sensor, the greater is the speed difference of the motor on the left and right sides. This keeps KitiBot along the black line.

Script:

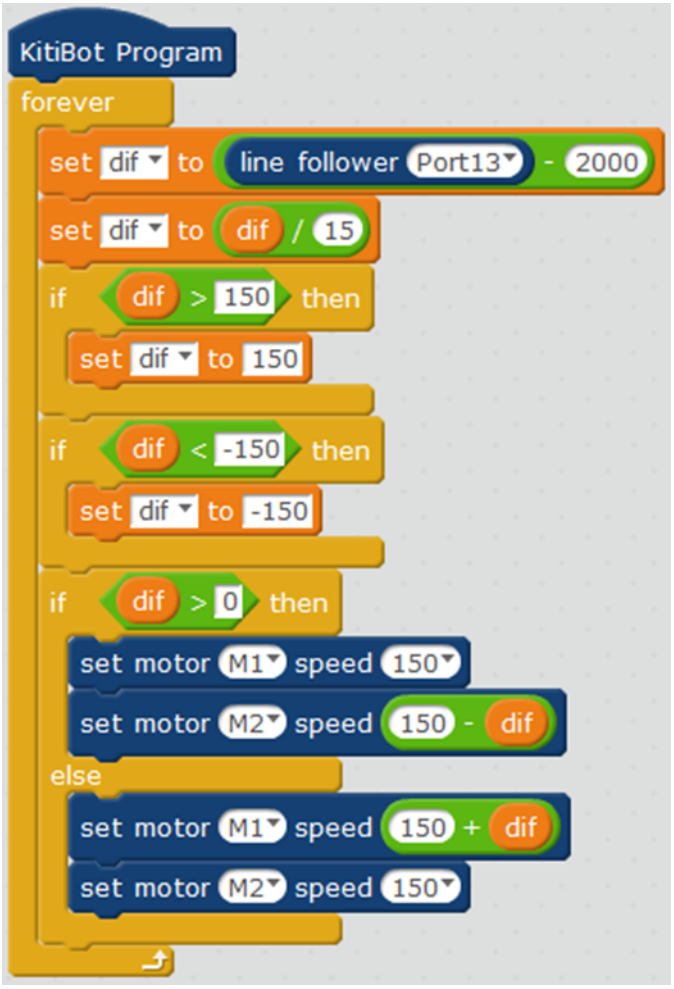


Figure 87 Tracking Script

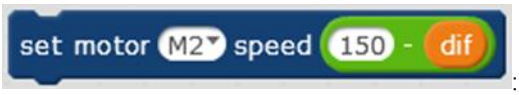
Description:

First, we create a variable called “dif” to store the sensor's deviation value. Since the range of the sensor is 0 to 4000, the value of the sensor is subtracted by 2000 to work with a range of

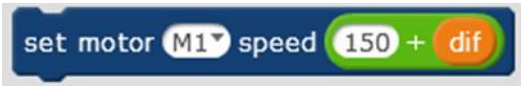
-2000 to 2000. At this point the value 0 means no deviation, a negative number is left, and positive mumbers are biased to the right.

As the maximum robot motor speed is set to 150. It is necessary to divide the deviation by a number to control it within the range of -150 to 150. The value of “dif” is controlled again between negative 150 and 150 by the following statement.

If dif value is greater than 0, then the black line is to the right, and the speed of the motor on the right should be reduced. KitiBot then, will also follow the right.



Similarly, If dif value is smaller than 0, then the black line is to the left, and the speed of the motor on the left should be reduced.



At this point, it should be noted here that “dif” is already negative, and should be added to reduce instead of subtracting.

Expected Result:

Finally, upload the program to the robot and put the robot in the black line. The robot should move along the black line.