# MU VISION SENSOR 3 GUIDES

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>MU Vision Sensor 3 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>MU 3 Mixly Programming Guide</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>MU 3 Arduino Programming Guide</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>MU 3 MakeCode Programming Guide</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>MU 3 MicroPython Programming Guide</td>
<td>47</td>
</tr>
<tr>
<td>6</td>
<td>MU Vision Sensor Resource</td>
<td>55</td>
</tr>
<tr>
<td>7</td>
<td>MU Vision Sensor Application</td>
<td>57</td>
</tr>
<tr>
<td>8</td>
<td>MoonBot Kit Introduction</td>
<td>63</td>
</tr>
<tr>
<td>9</td>
<td>MoonBot Kit Hardware Instruction</td>
<td>65</td>
</tr>
<tr>
<td>10</td>
<td>MoonBot Kit Structure Instruction</td>
<td>91</td>
</tr>
<tr>
<td>11</td>
<td>MoonBot Kit MU Bot App Tutorial</td>
<td>103</td>
</tr>
<tr>
<td>12</td>
<td>MoonBot Kit Mixly Tutorial</td>
<td>135</td>
</tr>
<tr>
<td>13</td>
<td>MoonBot Kit Arduino Tutorial</td>
<td>169</td>
</tr>
<tr>
<td>14</td>
<td>MoonBot Kit Extended Structures</td>
<td>205</td>
</tr>
<tr>
<td>15</td>
<td>MoonBot Kit Firmware Upgrade Guide</td>
<td>209</td>
</tr>
<tr>
<td>16</td>
<td>MoonBot Kit Resource</td>
<td>213</td>
</tr>
<tr>
<td>17</td>
<td>MU Self-driving Kit Introduction</td>
<td>215</td>
</tr>
<tr>
<td>18</td>
<td>MU Self-driving Kit Structure</td>
<td>217</td>
</tr>
<tr>
<td>19</td>
<td>MU Self-driving Kit MakeCode Tutorial</td>
<td>219</td>
</tr>
<tr>
<td>20</td>
<td>MU Self-driving Kit Resource</td>
<td>243</td>
</tr>
<tr>
<td>21</td>
<td>Technical Support</td>
<td>245</td>
</tr>
<tr>
<td>22</td>
<td>About</td>
<td>247</td>
</tr>
</tbody>
</table>
MU Vision Sensor 3 (MU 3) is an intelligent vision sensor that can recognize many kinds of objects with deep-learning algorithm inside. For example, it can detect color blocks, balls, human body and cards. Its detect result can be transmitted through UART or I2C interface. MU 3 is compact, low power consumption, process all algorithms locally, and can be widely used in intelligent toys, AI+STEAM lessons, creators and other products or fields.

1.1 Hardware Setup

1.1.1 1. Set Communication Mode

MU supports 4 kinds of communication modes: UART, I2C, WIFI, image transmission. Change mode by switching Output switch on left side of MU.

After choosing mode, communication mode in program should be same as switch to make the codes working. Choose communication mode before coding, and then set other parameters. Every time changing communication mode, MU must be restarted.
<table>
<thead>
<tr>
<th>output mode</th>
<th>switch</th>
<th>number</th>
<th>LED indicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>UART</td>
<td></td>
<td>00</td>
<td>flash red</td>
</tr>
<tr>
<td>I2C</td>
<td></td>
<td>01</td>
<td>flash green</td>
</tr>
<tr>
<td>WiFi</td>
<td></td>
<td>10</td>
<td>flash yellow</td>
</tr>
<tr>
<td>image transmission</td>
<td></td>
<td>11</td>
<td>flash purple</td>
</tr>
</tbody>
</table>

### 1.1.2 2. Set Address

MU supports 4 address: 0x60 (default), 0x61, 0x62, 0x63. Address should be changed when conflicted with other sensors. In I2C modes, several MU sensors can work together with different address.

**Note:** Use default address 0x60 normally.

<table>
<thead>
<tr>
<th>device address</th>
<th>switch</th>
<th>number</th>
<th>device address</th>
<th>switch</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x60</td>
<td></td>
<td>00</td>
<td>0x61</td>
<td></td>
<td>01</td>
</tr>
<tr>
<td>0x62</td>
<td></td>
<td>10</td>
<td>0x63</td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

### 1.1.3 3. Wire Connection

UART/WiFi/image transmission modes

<table>
<thead>
<tr>
<th>MU controller RX</th>
<th>TX</th>
<th>G</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>RX</td>
<td>GND</td>
<td>5V</td>
</tr>
</tbody>
</table>

I2C modes

<table>
<thead>
<tr>
<th>MU controller SCL</th>
<th>SDA</th>
<th>G</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCL</td>
<td>SDA</td>
<td>GND</td>
<td>5V</td>
</tr>
</tbody>
</table>
1.2 Software Setup

Check detailed instructions among platforms below.

1.3 Special modes introduction

1.3.1 WiFi/image transmission mode network distribution

In WiFi/image transmission mode, distribute network by sending AT command to MU. Default serial baudrate is 9600. Send following command to know all AT commands:

AT+HELP

**Attention:** all commands should end with "\r\n" or ' '.

MU supports AP and STA modes to connect to network. Here is difference between two ways:

**AP mode** AP mode is default WiFi mode of MU. In this mode, MU will establish a WiFi hotspot for user to connect. When WiFi connect successfully, LED of MU will turn off.

Default hotspot name is MORPX-MU-AB.

**Note:**

A stands for initial word of LED color on the left, and B stands for initial word of LED color on the right.

For example: left LED is R ed, right LED is **Y**ellow, then default WiFi name is MORPX-MU-RY.

Send following AT command to change WiFi name:

AT+WIFISET=<yourSSID>,<yourPassword>,AP
AT+WIFICON=1

If succeed, returns:

OK
wifi ap mode starting...
OK

**STA mode** STA mode means MU and another device should connect to the same WiFi to get each connected. Send following commands to configure:

AT+WIFISET=<yourSSID>,<yourPassword>,STA
AT+WIFICON=1

**Attention:** <yourSSID> [ ] <yourPassword> should be an available WiFi(case sensitive), or connection failed.
If set successfully, return:

```
OK
wifi sta mode connecting...
OK
```

### 1.3.2 Watch image in image transmission mode

Set MU to image transmission mode and complete *WiFi connect*, images can be watched through website `192.168.4.1`.

### 1.3.3 Wireless Transmission

MU can transmit data in WiFi/image transmission modes. Complete *WiFi connect* and then take following steps:

**Note:** TCP/UDP software are different in PC or mobile devices, here are some common definition:

- **local IP**: IP address of MU
- **target IP**: IP address of target device that MU send to

1. Open TCP/UDP software, choose **UDP**, and change mode to **Unicast**
2. Search local IP by sending command to MU:

```
AT+WIFISIP
```

ReturnMU local IP.
3. SetTCP/IP to MU local IP, and port is **3333**

**Note:** In STA mode router will distribute a random address for MU and target device. Take following steps:

1. Search target IP (MostTCP/IP software will show local IP address)
2. Send command to MU:

```
AT+WIFIUDP=<targetIP>,3333
```

Return:

```
OK
```

NowWiFi configuration is finished, and all the data from TCP/UDP software will show on MU serial port, and all data from MU serial port will show on TCP/UDP software.
This passage introduces how to use MU Vision Sensor 3 with Arduino board and Mixly IDE.

2.1 Import Mixly Library for MU Vision Sensor 3

Open Mixly and choose controller. For example, choose Arduino Uno. If you use MoonBot controller, choose Arduino Mega (atmega 1280) and choose the available COM port.

Click ‘Import’.
Locate the ‘MuVisionSensor3.xml’ file, select and open it.

Then the nether information window prompts “import custom Library successfully”, and you will find ‘MuVisionSensor3’ library in the ‘Blocks’ window.
2.2 Connect to Arduino

MU Vision Sensor 3 peripherals and ports:
2.2.1 I2C Mode (recommended)

2. Connect the output SDA pin of MU to the SDA pin of Arduino, and SCL pin of MU to SCL pin of Arduino.
3. Choose the I2C address of MU by setting Address Switch. Both switches are downwards and the address is set to 0x60 on default. (Changing this setting is not recommended).
2.2.2 Serial Mode*

2. Connect the output RX pin of MU to TX pin of Arduino and TX pin of MU to RX pin of Arduino.
3. Change the UART address of MU sensor by resetting Address Switch. Both switches are downwards and the address is 0x60 on default. (Changing this setting is not recommended)

*Arduino UNO cannot send messages to PC when MuVisionSensor is running in hardware serial mode, due to a communication conflict.
2.2.3 AT Command Mode (For firmware after V1.1.5)

2. Connect the output RX pin of MU to TX pin of Arduino and TX pin of MU to RX pin of Arduino.

2.2.4 Image Transmission Mode (For firmware after V1.1.5)

1. Output Protocol Switch: both switches are upwards.
2. Connect the output RX pin of MU to TX pin of Arduino and TX pin of MU to RX pin of Arduino.

2.3 Block Introduction

2.3.1 Setting Blocks

Initialization

1. Hardware Serial Mode: Vision sensor uses serial mode to initialize the main control when connecting the main control hardware serial port. The serial port is the serial communication between the main control and the computer. When it is used for vision sensor, computer printing characters will be disordered or communication abnormalities.

![Hardware Serial Mode](image)

1. Software Serial Mode: Vision sensor uses serial mode, and the main controller is initialized when connecting the main control software serial port. The main controller can customize RX and TX pins. The speed of soft serial port in real environment may be too fast and unstable. The baud rate is not recommended to exceed 9600.

![Software Serial Mode](image)

1. Hardware I2C Mode: The vision sensor uses I2C mode to initialize the main controller when connecting the main controller I2C pins.

![Hardware I2C Mode](image)
Enable Vision Algorithms

Algorithm performance level

Enable/Disable the High FPS Mode

In high FPS mode, detect speed and power consumption will increase.

Set White Balance Mode

Adjust the image bias caused by the change of external light source.
**LED Settings**

```
setup Mu00

if Mu00 detected then
  LED 1 when detected then
else
  brightness(0-15) 1

if Mu00 detected then
  LED 2 when detected then
else
  brightness(0-15) 1
```

**Restore Default Settings**

Disable all algorithms and restore hardware default settings.

```
setup Mu00
restore default settings
```

**Get Result Blocks**

1. **Ball/Body Detection**

```
if Mu00 detected
  Serial println “ball detected.”
  Serial println “x = ”
  Serial println get Mu00 algorithm Ball Horizontal value
  Serial println “y = ”
  Serial println get Mu00 algorithm Ball Vertical value
  Serial println “width = ”
  Serial println get Mu00 algorithm Ball Width value
  Serial println “height = ”
  Serial println get Mu00 algorithm Ball Height value
  Serial println “label = ”
  Serial println get Mu00 algorithm Ball Label value
else
  Serial println “ball undetected.”
```

1. **Card Detection**
1. Color Block Detection
1. Color Recognition

```cpp
if (Mu00.recognized_coordinate.x == 50 && y == 50) {
    Serial.println("color detected.");
    Serial.print("r = ");
    Serial.print(r);
    Serial.print(" g = ");
    Serial.print(g);
    Serial.print(" b = ");
    Serial.print(b);
    Serial.println("color = ");
}
if (get_Mu00_algorithm(ColorRecognition.color == RED)) {
    Serial.println("red");
} else if (get_Mu00_algorithm(ColorRecognition.color == BLACK)) {
    Serial.println("black");
} else if (get_Mu00_algorithm(ColorRecognition.color == WHITE)) {
    Serial.println("white");
} else if (get_Mu00_algorithm(ColorRecognition.color == GREEN)) {
    Serial.println("green");
} else if (get_Mu00_algorithm(ColorRecognition.color == YELLOW)) {
    Serial.println("yellow");
} else if (get_Mu00_algorithm(ColorRecognition.color == BLUE)) {
    Serial.println("blue");
} else if (get_Mu00_algorithm(ColorRecognition.color == PURPLE)) {
    Serial.println("purple");
} else if (get_Mu00_algorithm(ColorRecognition.color == CYAN)) {
    Serial.println("cyan");
} else if (get_Mu00_algorithm(ColorRecognition.color == UNKNOWN)) {
    Serial.println("unknown");
} else {
    Serial.println("color undetected.");
}
```
2.3.2 Light Sensor Blocks

Enable Light Sensor

Enable light sensor functions. Gesture detect function cannot work together with other functions.

Set Sensitivity

Set light sensor sensitivity, only working in Proximity/Ambient Light Detect.
Get Proximity Value

Read proximity value. Closer the distance, bigger the value.

```
light sensor read proximity detection
```

Read Ambient Light Value

Lighter the ambient light, bigger the value.

```
light sensor read ambient light detection
```

Read Detected Gesture or Not

Read whether detected gesture. If not detected, return 0.

```
detected gesture
```

Judge Detected Gesture

Judge type of the detected gesture.

```
detected gesture = upward
```

### 2.3.3 WiFi Module Blocks

**WiFi AT command Settings**

*Blocks below only works in image transmission or AT command mode.*

**WiFi Initialization**

Initial WiFi port and baudrate.

```
initialize port Serial UART baudrate 9600
```

**Configure WiFi**

Set account and password of WiFi.

```
set ssid " " password " " mode client
```
Connect/Establish WiFi

Try connect to WiFi or establish WiFi and return connection result.

```
MU WiFi connection succeeded?
```

Disconnect WiFi

```
MU WiFi disconnect
```

Set target IP

```
MU WiFi set target IP "192.168.4.2" port 333
```

Read target IP

```
MU WiFi read target IP
```

Read local IP

Get IP of MU.

```
MU WiFi read local IP
```

WiFi Read

Get data that target IP sent to MU.

```
MU WiFi read
```

WiFi Write

Send data to target IP.

```
MU WiFi write 0
```
2.4 Examples

2.4.1 Color Recognition

```java
setup
Serial baud rate 9600
initialize MU00 port I2C
setup Mu00
enable algorithm ColorRecognition
camera white balance LockANB

if (Mu00 recognized coordinate x= 50 y= 50 color)
do
  Serial println "color detected."
  Serial println "r = "
  Serial println get Mu00 algorithm ColorRecognition Channel8 value
  Serial println "g = "
  Serial println get Mu00 algorithm ColorRecognition Channel6 value
  Serial println "b = "
  Serial println get Mu00 algorithm ColorRecognition Channel10 value
  Serial println "color = "
  if (get Mu00 algorithm ColorRecognition color= [red])
do
  Serial println "red"
else if (get Mu00 algorithm ColorRecognition color= [black])
do
  Serial println "black"
else if (get Mu00 algorithm ColorRecognition color= [white])
do
  Serial println "white"
else if (get Mu00 algorithm ColorRecognition color= [green])
do
  Serial println "green"
else if (get Mu00 algorithm ColorRecognition color= [yellow])
do
  Serial println "yellow"
else if (get Mu00 algorithm ColorRecognition color= [blue])
do
  Serial println "blue"
else if (get Mu00 algorithm ColorRecognition color= [purple])
do
  Serial println "purple"
else if (get Mu00 algorithm ColorRecognition color= [cyan])
do
  Serial println "cyan"
else
  Serial println "color undetected."
```

Chapter 2. MU 3 Mixly Programming Guide
2.4.2 Ball Detection

```python
setup
Serial baud rate [9600]
initialize Mu00 port I2C
setup Mu00
enable algorithm Ball

if Mu00 detected Ball:
do
  Serial println "ball detected."
  Serial print "x = "
  Serial println get Mu00 algorithm Ball Horizontal value
  Serial print "y = "
  Serial println get Mu00 algorithm Ball Vertical value
  Serial print "width = "
  Serial println get Mu00 algorithm Ball Width value
  Serial print "height = "
  Serial println get Mu00 algorithm Ball Height value
  Serial print "label = "
  Serial println get Mu00 algorithm Ball Label value
else
  Serial println "ball undetected."
```
2.4.3 Card Detection

```java
setup

Serial baud rate 9600
initialize Mu00 port I2C
setup Mu00

if Mu00 detected ShapeCard

do

    Serial println "card detected."
    Serial println "x = ",
    Serial println get Mu00 algorithm ShapeCard Horizontal value
    Serial println "y = ",
    Serial println get Mu00 algorithm ShapeCard Vertical value
    Serial println "width = ",
    Serial println get Mu00 algorithm ShapeCard Width value
    Serial println "height = ",
    Serial println get Mu00 algorithm ShapeCard Height value
    Serial println "card type = ",

    if get Mu00 algorithm ShapeCard type= Tick
        do
            Serial println "tick"
        else if get Mu00 algorithm ShapeCard type= Cross
            do
                Serial println "cross"
            else if get Mu00 algorithm ShapeCard type= Circle
                do
                    Serial println "circle"
                else if get Mu00 algorithm ShapeCard type= Square
                    do
                        Serial println "square"
                    else if get Mu00 algorithm ShapeCard type= Triangle
                        do
                            Serial println "triangle"
                        else
                            Serial println "card undetected."
                        end
                    end
                end
            end
        end
    end
end
```
2.4.4 Light Sensor - Gesture Detect

```plaintext
setup
Serial baud rate 9600
initialize MUI00 port I2C
MUI00 light sensor enable gesture detection

if MUI00 light sensor detected gesture
  do
    if MUI00 light sensor gesture = upward
        do Serial println "upward"
    else if MUI00 light sensor gesture = downward
        do Serial println "downward"
    else if MUI00 light sensor gesture = leftward
        do Serial println "leftward"
    else if MUI00 light sensor gesture = rightward
        do Serial println "rightward"
    else if MUI00 light sensor gesture = pull
        do Serial println "pull"
    else if MUI00 light sensor gesture = push
        do Serial println "push"
    else if MUI00 light sensor gesture = backward
        do Serial println "backward"
```

2.4.5 Light Sensor - Proximity/Ambient Light Detect

```plaintext
setup
initialize MUI00 port I2C
MUI00 light sensor enable proximity detection
MUI00 light sensor enable ambient light detection
MUI00 light sensor set sensitivity middle

Serial println "proximity = ", toString decimal MUI00 light sensor read proximity detection
Serial println "als = ", toString decimal MUI00 light sensor read ambient light detection
Delay ms 500
```
2.4.6 Connect to WiFi through AT Command

```
setup
SoftwareSerial RX# A5 TX# A4
MU WiFi initialize port SoftwareSerial UART baudrate 9600
MU WiFi set ssid "yourssid" password "yourpassword" mode client
Serial println "connect to WiFi"
if MU WiFi connection succeeded?
do
  MU WiFi set target IP "10.10.1.181" port "3333"
  Serial println "success"
  Serial println "Target IP: " join MU WiFi read target IP
  Serial println "local IP: " join MU WiFi read local IP
else
  Serial println "fail"
  MU WiFi set ssid "MU_WIFI" password "123456" mode hot-spot
if MU WiFi connection succeeded?
do
  Serial println "create WiFi"
  MU WiFi set target IP "192.168.4.2" port "3333"
```

2.5 FAQs

1. Q: What should I do if I can't open programming blocks or blocks are black boxes?
   A: Please download the latest Mixly program file and import the library again.

1. Q: What should I do when I import the library and download the examples correctly, but the vision sensor doesn't respond and the serial port output nothing?
   A: Check whether the cable is correctly and tightly connected.
   Check whether the white light on the back side is on. If this light is not on, this means something wrong with power supply.
   Check whether the output mode switch and address switch are correctly setted.
   It takes MU sensor a period of time to initialize after power-on, we suggested to add a delay of not less than 500 ms before calling 'setup' block in your program.
After reset, the two LEDs on the front side of MU vision sensor will flicker once. Red light indicates that the current mode is serial mode, green light indicates that the current mode is I2C mode. If the color does not go with the setting of output mode switch, please reset this switch.

1. Q: What should I do if I download the program and the serial port output correctly, but the LED does not light?

A: When the color recognition algorithm runs, the LED lights will be shut down in order not to interfere with the recognition results.

Calling the LED setting block and alter the brightness parameter with non-zero number.

1. Q: Why there are some functions which can be found in the datasheet while are not listed in Mixly library?

A: In order to make the library easy to understand and operate, some uncommon functions are removed from the Mixly library, and some parameter setting methods are simplified. If these functions are needed, please email to support@morpx.com.

1. Q: Why the previously downloaded recognition algorithms have impact on the current running program? For example download ball detection algorithm after running color recognition algorithm, you will find the LEDs are still on even if the ball is not recognized, but this will not occur when only ball recognition algorithm was downloaded.

A: Because the previous algorithm is not terminated even if the program ends, you can add ‘setDefault’ block when calling ‘setup’ block or restart MU vision sensor.
This passage introduces how to use MU Vision Sensor 3 with Arduino board and Arduino IDE.

### 3.1 Import Arduino Library for MU Vision Sensor

Download Arduino IDE at Arduino official website

https://www.arduino.cc/en/Main/Software

Download latest MU Vision Sensor library in github

https://github.com/mu-opensource/MuVisionSensorIII

Install the Arduino IDE, and third-part library is located in “Documents\Arduino\libraries”. Unzip the MU Vision Sensor library and drag it into the libraries file to finish importing the library.

Open Arduino IDE, choose “Tools - Board” to change the development board. Arduino Uno is a popular board. If you use the MoonBot controller board, you should choose Arduino Mega 2560, and choose the processor ATmega 1280. Then connect the board and choose the right COM port.

If the library is successfully imported, the examples of MU is shown in "File - Examples - Mu Vision Sensor III”. Open an example and verify it to check whether the library can be compiled.
### 3.2 Connect to Arduino

MU Vision Sensor 3 peripherals and ports:

![MU Vision Sensor diagram]

#### 3.2.1 I2C Mode (recommended)

2. Connect the output SDA pin of MU to the SDA pin of Arduino, and SCL pin of MU to SCL pin of Arduino.
3. Choose the I2C address of MU by setting Address Switch. Both switches are downwards and the address is set to 0x60 on default. (Changing this setting is not recommended).
3.2.2 Serial Mode*

(1) Output Protocol Switch: set both switches downward.
(2) Connect the output RX pin of MU to TX pin of Arduino and TX pin of MU to RX pin of Arduino.
(3) Change the UART address of MU sensor by resetting Address Switch. Both switches are downwards and the address is 0x60 on default. (Changing this setting is not recommended)

Arduino UNO cannot send messages to PC when MuVisionSensor is running in hardware serial mode, due to a communication conflict.
3.2.3 AT Command Mode (For firmware after V1.1.5)

2. Connect the output RX pin of MU to TX pin of Arduino and TX pin of MU to RX pin of Arduino.

3.2.4 Image Transmission Mode (For firmware after V1.1.5)

1. Output Protocol Switch: both switches are upwards.
2. Connect the output RX pin of MU to TX pin of Arduino and TX pin of MU to RX pin of Arduino.

3.3 Examples

ball
body
colorDetect
colorRecognition
getTargetPosition
numberCard
shapeCard
trafficCard
arduino_base_interface
This passage introduces how to use MU Vision Sensor 3 with Micro:bit board and MakeCode IDE.

4.1 Import Extensions


Search `mu vision` or `mu` to find ‘musion’ from the results list. (former link `mu-opensource/pxt-MuVisionSensor3` is abandoned) Click to import the extension into MakeCode.
muvision
MU vision sensor, specifically designed for the AI education.

Want to create your own extension? Login to GitHub
4.2 Connect to Micro:bit

MU Vision Sensor 3 peripherals and ports

4.2.1 I2C Mode (recommended)

1. Output Protocol Switch: set switch 1 downwards and switch 2 upwards
2. Connect the output Pin1(SDA) to the Pin20 of Micro:bit, and Pin2(SCL) to Pin19 of Micro:bit
3. Choose the I2C address of MU sensor by resetting Address Switch. By default, both switches are downward and the address is set to 0x60. (Changing this setting is not recommended)

You may need a shield to connect MU to Micro:bit, as is shown below:
4.2.2 Serial Mode

(1) Output Protocol Switch: both switches are downwards

(2) Connect the output Pin1(RX) to Pin13(TX) of Micro:bit and Pin2(TX) to Pin16(RX) of Micro:bit

(3) Change the UART address of MU sensor by resetting Address DIP Switches. By default, both switches are downward and the address is 0x60. (Changing this setting is not recommended)

*Micro:bit cannot send messages to PC when MuVisionSensor is running in serial mode, due to a communication conflict.

*The default communication baud rate is 9600 and cannot be modified.
4.2.3 AT Command Mode (For firmware after V1.1.5)

2. Connect the output Pin1(RX) to Pin13(TX) of Micro:bit and Pin2(TX) to Pin16(RX) of Micro:bit.

4.2.4 Image Transmission Mode (For firmware after V1.1.5)

1. Output Protocol Switch: both switches are upwards.
2. Connect the output Pin1(RX) to Pin13(TX) of Micro:bit and Pin2(TX) to Pin16(RX) of Micro:bit.

4.3 Block Introduction

4.3.1 MUvision

Initialization

1) Serial Mode

In serial mode, two pins are defined as TX & RX respectively, according to the hardware connection to the MU Vision Sensor P12 and P13 as example

(2) I2C Mode
Enable Vision Algorithms

Seven recognition algorithms are integrated in current firmware (Version A). For detailed information please refer to the datasheet of MU vision sensor.

Set Performance Level

Algorithm performance differs in accuracy and speed. Performance settings can be changed to fit in certain applications. Default setting: Balance level.

Enable/Disable the High FPS Mode

The camera is in high FPS mode by default, which has a higher speed than normal mode. High FPS mode can be turned off to save power.
Set White Balance Mode

Ambient light will influence the detect result of the vision sensor, especially color detection and recognition. In complex light environment or in color recognition mode, it is recommended to lock white balance.

Reset the vision sensor, and put it in front of a white paper to measure brightness. A few seconds later, white balance will be locked.

Set Digital Zoom Level

Larger digital zoom level means longer detectable distance, and view sight is narrower meanwhile.

To get a better detect results, set a proper zoom level for the algorithm and test it.
**On-board LED Settings**

Two on-board LED lights can be programmed to shine different colors when the sensor has detected an object or not. Default setting: When undetected objects, two lights are red and when detected they are blue.

**Restore Default Settings**

Enable light sensor functions

If light sensor enabled, gesture function cannot be used with other functions in the mean time.

Set Light Sensor Sensitivity

Light sensor read proximity value
Light sensor read brightness value

Light sensor read gesture value

Light sensor get gesture result

### 4.3.2 MUvisionAT

WiFi Configure Module can only be used in WiFi and image transmission modes.

**Read local IP**

Read IP of MU.

**Read target IP**

Read IP of target device.

**WiFi Configuration**

Configure WiFi account, password and mode.
WiFi link

Try turn on/off WiFi. Return `true` if succeed.

Configure target IP through WiFi

Only working when successfully connected to WiFi.

WiFi read transmission data

Read data from target device through WiFi.

4.4 Get Result Examples

4.4.1 Ball/Body Detection

Setup program: Init I2C connection, and set algorithm to ball detect. Other settings are default.

Loop program: If MU detected a ball, it will send data to Micro:bit through I2C interface. And PC get the data from Micro:bit through USB serial port. The data contains position and size of the ball.

Actual result: After resetting MU and Micro:bit, LED lights flash red. When MU detected a ball, LED lights flash blue and Makecode console will display the data.
4.4.2 Card Detection

Setup program: Init I2C connection, and set algorithm to traffic card. Other settings are default.

Loop program: If MU detected traffic card, it will send data to Micro:bit through I2C interface. And PC get the data from Micro:bit through USB serial port. The data contains position, size and type of the traffic card.

Actual result: After resetting MU and Micro:bit, LED lights flash red light. When MU detected a traffic card, LED lights flash blue and Makecode console will display the data.
on start
init MU00 ▼ interface ▼ I2C ▼
MU00 ▼ enable ▼ algorithm ▼ Traffic Card ▼

Forever
if MU00 ▼ is detected ▼ Traffic Card ▼ then
serial write line 'card detected'
serial write value 'x' ▼ = get MU00 ▼ Traffic Card ▼ x position ▼
serial write value 'y' ▼ = get MU00 ▼ Traffic Card ▼ y position ▼
serial write value 'width' ▼ = get MU00 ▼ Traffic Card ▼ width ▼
serial write value 'height' ▼ = get MU00 ▼ Traffic Card ▼ height ▼
serial write string 'card type' ▼ =
if MU00 ▼ get ▼ Traffic Card ▼ ▼ forward ▼ then
serial write line 'forward'
else if MU00 ▼ get ▼ Traffic Card ▼ ▼ left ▼ then
serial write line 'left'
else if MU00 ▼ get ▼ Traffic Card ▼ ▼ right ▼ then
serial write line 'right'
else if MU00 ▼ get ▼ Traffic Card ▼ ▼ Turn Around ▼ then
serial write line 'turn around'
else if MU00 ▼ get ▼ Traffic Card ▼ ▼ Park ▼ then
serial write line 'park'
else
serial write line 'card undetected'
4.4.3 Color Recognition

Setup program: Init I2C connection, and set algorithm to color recognition. Lock the white balance to get a higher accuracy. Other settings are default.

Loop program: If MU detected color at (50,50), it will send data to Micro:bit through I2C interface. And PC get the data from Micro:bit through USB serial port. The data contains RGB channel and type of the color.

Actual result: After resetting MU and Micro:bit, LED lights are off. Makecode console will display the data.
4.4.4 Color Block Detection

Setup program: Init I2C connection, and set algorithm to color block. Lock the white balance to get a higher accuracy. Other settings are default.

Loop program: If MU detected a color block, it will send data to Micro:bit through I2C interface. And PC get the data from Micro:bit through USB serial port. The data contains position, size and type of the color block.

Actual result: After resetting MU and Micro:bit, LED lights flash red light. When MU detected a color block, LED lights flash blue and Makecode console will display the data.

4.4.5 Serial Mode Example

Change the output protocol switch to serial mode and connect MU to Micro:bit through serial port. In this mode PC cannot communicate with Micro:bit, so the led dot screen of the Micro:bit is used to display the data directly.

Setup program: redirect the serial port to P14 and P13, and select 'Number Card' algorithm. Other settings are default.

Loop program: If MU detects one number card, it will send data to Micro:bit through serial interface. The Micro:bit LED screen shows the number.

Actual result: After resetting MU and Micro:bit, LED lights flash red light. When MU detected a number 1 card, LED lights flash blue and Micro:bit screen shows 1.
on start
  serial
  redirect to
  TX P14 ↓
  RX P13 ↓
  at baud rate 9600 ↓
  pause (ms) 100 ↓
  init M000 ↓ interface Serial ↓
  M000 ↓ enable ↓ algorithm □ Number Card ↓

forever
  if M000 ↓ is detected □ Number Card ↓ then
  if M000 ↓ get □ Number Card = 1 ↓ then
    show Icds
    □
    □
    □
    □
    □
    □
  else if M000 ↓ get □ Number Card = 2 ↓ then
    show Icds
    □
    □
    □
    □
    □
    □

This passage introduces how to use MU Vision Sensor 3 with Micro:bit board and MicroPython language.

### 5.1 Preparations before Coding

If you want to use Micro:bit to control the MU Vision Sensor, you need to import the MicroPython firmware that contains the 'MuVisionSensor' module. Please follow the steps below:

1. **Download the firmware:**

2. **Flash the firmware:**
   - Connect the micro:bit to computer via USB cable, and drag the downloaded firmware “microbit-micropython-MuVisionSensor-0.8.0.hex” to MICROBIT disk. Micro:bit will flash the new firmware and restart.

3. **Download and install Mu Editor:**
   - Mu Editor is a simple Python code editor for beginner programmers with friendly GUI.
   - It can be downloaded in the main page: [https://codewith.mu/](https://codewith.mu/)

4. **Import module**
   - Open Mu Editor, change the mode to BBC micro:bit, and click the "REPL" button to enter serial reply mode. The micro:bit will send firmware version at first. Type the code below and press enter to run:

   ```python
   >>> from MuVisionSensor import *
   ```

   Now you can call all the public APIs in ‘MuVisionSensor’

   *Key words auto-completion function is only available in REPL mode.*
5.2 Connect to Micro:bit

MU Vision Sensor 3 peripherals and ports:
I2C Mode

1) Output Mode DIP Switch: set switch 1 downwards and switch 2 upwards

2) Connect the output Pin1 (SDA) to the Pin20 of Micro:bit, and Pin2 (SCL) to Pin19 of Micro:bit. Also connecting the ground pin and 3.3v power pin to Micro:bit.

3) Change the I2C address of MU Vision Sensor by resetting Address DIP Switch. In default both switches are downward and the address is 0x60. (Changing this setting is not recommended)

*Only I2C mode is supported now.*

You may need a shield to connect MU to Micro:bit, as is shown below:
5.3 Usage of APIs

5.3.1 Initialize MU Vision Sensor

Two steps to initialize this sensor:

Step1. Call the ‘MuVisionSensor(‘address’)’ to create an object, the value of ’address’ should be consistent with the setting of Address DIP Switch (default is 0x60);

Step2. Call the ‘begin()’ function to start this sensor;
5.3.2 Enable Algorithms

API

```python
MuVisionSensor.VisionBegin(vision_type)
```

All available `vision_type`'s as follows:

- VISION_COLOR_DETECT
- VISION_COLOR_RECOGNITION
- VISION_BALL_DETECT
- VISION_BODY_DETECT
- VISION_SHAPE_CARD_DETECT
- VISION_TRAFFIC_CARD_DETECT
- VISION_NUM_CARD_DETECT
- VISION_ALL

Example

```python
from MuVisionSensor import *  # import the library
...
mu.VisionBegin(VISION_COLOR_DETECT)  #
mu.VisionBegin(VISION_SHAPE_CARD_DETECT | VISION_BALL_DETECT)  # enable card detect and... ball detect algorithms
```

5.3.3 Set Performance Level

API

```python
MuVisionSensor.VisionSetLevel(vision_type, level)
```

`level` can be set to:

- LevelDefault
- LevelSpeed
- LevelBalance
- LevelAccuracy

Example

```python
mu.VisionSetLevel(VISION_BALL_DETECT, LevelSpeed)
```
Get Performance Level

API

```python
mu.VisionSetLevel(vision_type)
```

The return value is between 0~3, which represents the 4 levels

5.3.4 Enable High FPS Mode

API

```python
MuVisionSensor.CameraSetFPS(mode)
```

‘mode’ can be set to:
FPSNormal
FPSHigh

Get FPS Mode

API

```python
MuVisionSensor.CameraGetFPS()
```

return ‘0’(FPSNormal) or ‘1’(FPSHigh)

5.3.5 Set White Balance Mode

Adjust the color cast caused by the changes of external light sources.

API

```python
MuVisionSensor.CameraSetAwb(mode)
```

‘mode’ can be set to:
AutoWhiteBalance
LockWhiteBalance
WhiteLight
YellowLight
Get White Balance mode

API

MuVisionSensor.CameraGetAwb()

The return value is between 0~3, which represents the 4 WB modes.

5.3.6 Set Digital Zoom Ratio

API:

MuVisionSensor.CameraSetZoom(mode)

'mode' can be set to:
ZoomDefault
Zoom1
Zoom2
Zoom3
Zoom4
Zoom5

Get Digital Zoom Ratio Setting

API

MuVisionSensor.CameraGetZoom()

The return value is between 0~5, which represents the 6 zoom levels.

5.3.7 LED Settings

API

MuVisionSensor.LedSetColor(led, detected_color, undetected_color, level)

Explanations of these parameters:
led: the LED you want to configure, the available values
Led1
Led2
detected_color: colors which are detected, the available values as follows
LedClose
LedRed
LedGreen
LedYellow

5.3. Usage of APIs
Morpx Documentation

LedBlue
LedPurple
LedCyan
LedWhite

undetected_color: colors which are not detected, same available values as detected_color.
level: set the brightness level; an integer between 0 and 15; the larger the brighter.

5.3.8 Restore Default Settings

API
MuVisionSensor.SensorSetDefault()

5.3.9 Restart

API
MuVisionSensor.SensorSetRestart()

5.3.10 Get Results of Detection

API
MuVisionSensor.GetValue(vision_type, object_inf)

The available values of 'vision_type' are as mentioned above.
object_inf can be set to:
Status 0 means undetected, 1 means detected
XValue
YValue
WidthValue
HeightValue
Label
RValue
GValue
BValue
6.1 Technical Information

Thanks for purchasing MU Vision Sensor 3, and we would like to provide continuous updating service, please check to our website: www.morpx.com regularly. Updates are subject to change without notice. You can get the latest technical information from the following websites:

GitHub: https://github.com/mu-opensource/

6.2 3D Printing Bracket

For customers bought bare MU board, we provide 3D printing cover and foldable bracket. It can be fixed anywhere and can be adjusted to any angle. Please print by yourself if you have a 3D printer.

MU3 3D Printing Bracket
6.3 Platform Links

MU Vision Sensor can be connected to several opensource software and hardware platforms. Check the websites for detailed information.

Mixly

Mixly Official Website: http://mixly.org/

Arduino

Arduino Official Website: https://www.arduino.cc/

Micro:bit

Micro:bit Official Website: https://microbit.org/
MakeCode: https://makecode.microbit.org/#

MicroPython

MicroPython Official Website: http://micropython.org/

Mu IDE: https://codewith.mu/
CHAPTER
SEVEN

MU VISION SENSOR APPLICATION

7.1 Auto Cannon

7.1.1 Introduction

This project is an auto shoot cannon based on direnjie kit and MoonBot structure parts. MU vision sensor 3 is fixed on top of the cannon that can recognize the coordinates of the target. So that the controller can control the servos and water gun to aim and shoot the target until clear.

7.1.2 Contents

- A water gun
- Servo gimbal
- MU vision sensor 3
- Handbit controller
- Lipo battery + 18650 Power Li battery
- Target with MU number cards
This is the auto cannon. The hardware is from direnjie kit and structure is from MoonBot Kit. Other bracket parts can be 3d printed. Number cards are stuck to the original electric target so that the target can be recognized by MU.

### 7.1.3 Program Example

- **Initial Settings**

Firstly the peripherals should be set, including servos and MU.

```plaintext
setup

Declare servoBlas as int value -12
Declare servoX as int value 90 + servoBlas
Servo PIN# P5
degree (0-180) servoX
Delay(millis) 500
Initialize MU00 port I2C
set up MU00
   enable algorithm NumberCard
   zoom level5
Delay millis 100
Declare detected as int value 0
Declare pos as int value 0
```

- **Search Function**
Move servos in horizontal direction to search card in the vision sight.

- **Aim Function**

  When MU find a card, the program turns into aim function. Move servos slowly to adjust the position between cannon and card.

- **Shoot Function**

  Simply control the P7 pin to shoot for 0.5 seconds. Time can be modified to save the bullet.
• Loop Program

Combine the above functions and make the program repeatly.

• Remote Control

Cannon can be settled by Blynk app through BLE. Put the loop program to blynk blocks. When touch the button on the phone, auto shoot program can be launched.
7.1.4 War Game

Firstly, servos move to central position. The cannon moves from left to right and scan the target in the mean time. When meeting the number card, program turns into aim function. When verify the target position is within 48 to 52 position, shoot for 2 seconds and return to search function again.
MOONBOT KIT INTRODUCTION

MoonBot Kit is a STEAM education kit produced by Morpx. The kit contains various hardware modules, sheet metal parts and plastic shells, used to build all kinds of robots. With programming software in telephone and computer, teenagers can make AI robots, learn STEAM lessons and take steps towards excellent engineers in the future.
MOONBOT KIT HARDWARE INSTRUCTION

MoonBot Kit contains 9 kinds of hardware modules. Users can programme to control modules through Mixly or Arduino platform, or design interactive program after building certain structures. In instructions below, we provide pinout graphs and sample programs of every single module to help users get started with the kit.

Detailed software instructions at:
MoonBot Kit Mixly Instruction

Download all hardware programs here:
MoonBot Kit hardware examples

9.1 Controller Module

9.1.1 Brief Introduction

Controller Module is the programming core of the robot. The main chip is ATmega1280, which is Arduino compatible. On-board servo, motor and GPIO ports can be used to connect to other devices. And there are on-board keys, LED lights, buzzer and IMU that can be used to fast programme.
9.1.2 Specification

Size: 53 x 53 x 17.6 mm

Processor: ATmega1280

Ports: 4 servos, 2 motors, 9 GPIOs

On-board Resources: keys, LED, buzzer, IMU

Pinout
9.1.3 Usage

LED and Button Example

Button and LED light are basic IO device, and can be used to test other devices. So firstly we introduce these to help test others. The following example shows how to control 2 on-board RGB LED lights with 2 programmable buttons.

Code introduction: Loop detect the status of button A and B. LED1 turns red when button A is pressed, while LED2 turns green when button B is pressed. When both of the buttons are pressed, both LEDs turn blue. By default, both LEDs are off.

```plaintext
setup

controller LED brightness(0-255) 20

do

if button A& B pressed
  controller LED show

else if button A pressed
  controller LED show

else if button B pressed
  controller LED show

else
  controller LED show
```

Photo:
**Buzzer Example**

This code shows the way to programme buzzer on controller. Use two methods to make the buzzer beep.

Code introduction: Set the buzzer BPM (beats per minute) to 120, which means 1 beat is 0.5 second. Buzzer can directly play a tone like high G, or play a certain frequency like 194 Hz. A delay should be added when play the frequency, or it will be skipped immediately. Loop play 5 times and end playing. End the whole code by adding an end block or it will play repeatedly.

```
setup
  buzzer set tempo(BPM) 120

count with i from 1 to 5 step 1
  do
    buzzer play high G 1 beat
    buzzer frequency(Hz) 196 time(ms) 1000
    buzzer pause 1/16 beat
  end

buzzer stopplay
end program
```

**IMU Example**

IMU (Inertial Measurement Unit) contains one or more of gyroscope, accelerometer and compass. It is usually used to measure the posture of copters and robots. This IMU on controller contains accelerometer, compass and temperature compensation. Use the feedback to know while the robot is falling, dropped or shaking.

Code introduction: Initialize the compass and adjust the LEDs of controller at first. Loop judge the 4 directions defined by compass and accelerometer and show them on LEDs.

Phenomenon: Reset the controller and draw ∞ in the air to calibrate the compass. The calibration finishes when LEDs shine. Put the controller horizontally and 180° means the right South. Yaw it to the left/right and the left/right LED turns on. Pitch up/down and LED turns green/red.
9.2 Vision Module

9.2.1 Brief Introduction

Vision module is an intelligent vision sensor containing AI algorithms. Vision module can be connected to controller by serial interface, and controlled by preprogrammed instructions. It can also be controlled by mobile phone app through wifi.
9.2.2 Specification

Size: 37 x 37 x 15 mm
Processor: ESP32
Camera: OV7725
Sight: 85°
On-board Resources: keys, LED
Communications: UART wifi
Connector: PH2.0 4P

Pinout

9.2.3 Usage

Serial Communication Example

Vision module can be connected to controller module and receive instructions through serial port.
Hardware connection: Connect the vision module to P9 port, as a UART3 serial port device.
Code introduction: In initial part, Serial 3 is opened and default baudrate is set to 115200. Vision module is connected to serial3 and ball detect algorithm is enabled. In loop part, Controller LEDs are set same as vision module LEDs. That is, LEDs turn blue when detected ball, and turn red when undetected.
Phenomenon: Press the reset button of vision module, wait it to be in receiving status when LEDs turn on. Then reset the controller, and it will send instructions to vision module. Then the vision module is in ball detect mode, and LEDs flash red. The controller will get the data from vision module and show red too. When detect a ball, all LEDs turn blue.

Pay attention that the vision module is a little different from MU 3. Due to wifi function, vision module can only be connected to controller through serial port, and be developed by Arduino IDE or Mixly. The initial code is shown above.

In order to show the connect and initial progress, this program is easy. Learn more blocks and examples at MU 3 Mixly Programming Guide
Connect to App through Wifi

Vision module contains wifi unit, so it can be connected directly to MU Bot App. Programme or remote control it, please check MoonBot Kit MU Bot App Tutorial

9.3 Battery Module

9.3.1 Brief Introduction

Battery module is used to power controller module, driving actuators and sensors. It contains battery voltage converter, electricity management, overload protect and recharge unit. It can be charged directly by USB port (Not for communication).

9.3.2 Specification

Size: 67.6 x 56 x 33.7 mm
Battery Type: Polymer lithium battery
Output Power: 5V 2.8A max
Charging Power: 5V 1A max
Charging Period: 3.5h
Charging Port: USB type-C
9.3.3 Usage

- Short press power button to turn on battery and long press 3 seconds to turn off. Automatically turn off when unconnected for 30 seconds.
- Orange light on the top is charge indicator, while white light is power indicator and red light indicates USB connected. 4 blue lights indicates remaining capacity.
- Battery module power the controller module through PH2.0 5P wire. It can drive high-power actuators like servos and motors. The max output power is 5V 2.8A.
- Battery module can be charged through USB-C port (Not for communication), with 1A max charge current. Whole charge time is about 3.5 hours. It is recommended to use standard charger, while computer USB port can usually provide only 0.6A.

9.3.4 Attention

- Battery module can only be connected to controller module. Do not modify and disassemble it.
- Battery module contains circuit protect unit to prevent overloading. Manually control the output power can maintain a longer battery lifetime.
- LiPo battery is a Flammable and explosive product, so protect it from pressure, falling, water, heat and metal parts.
- Control the remaining power to 50% for long time storage. Prevent touching the button to open it.
9.4 Motor Module

9.4.1 Brief Introduction

Motor module contains a gear motor and an encoder inside. The active wheel is connected to gear motor, while the passive wheel is connected with screw and bearings.

The wheels are attached to track, and two whole motor modules are enough to build a Chassis.

9.4.2 Specification

Size: 109 x 40 x 39.1 mm
Reducer: 120:1
Unloaded Speed: 100rpm
Unloaded Current: 150rpm
Rated Speed: 70rpm
Rated Current: 300mA
Encoder: photoelectric encoder
Connector: PH2.0 2P + PH2.0 4P
9.4.3 Usage

Chassis Control

Under structures as MoonRover, MoonBot and MoonMech, chassis can be controlled to go forward, go back and turn around. Run the following program to test it.

Hardware connection: Connect the motors and encoders to the controller. Motor port M1 corresponds to encoder port P4, and M2 corresponds to P6. Motor module is a high-power device and controller should be connected to battery to drive motor. The controller can be connected to battery and USB at the same time and uses battery as power source.

Code introduction: In setup part, the direction of chassis is set, bias of direction, distance and turning radius is corrected. In loop part, Chassis goes forward, back, turns left and right. Control the distance and angle by setting motor speed. End the program at last.
Morpx Documentation

Chapter 9. MoonBot Kit Hardware Instruction

```plaintext
setup
  tank base reverse direction
  tank base straight offset calibrate(%) 100
  tank base straight distance calibrate(%) 100
  tank base turning angle calibrate(%) 100
  tank base forward(cm) 10 RPM(0-100RPM) 30
  tank base backward(cm) 10 RPM(0-100RPM) 30
  tank base turn left(°) 90 RPM(0-100RPM) 30
  tank base turn right(°) 90 RPM(0-100RPM) 30
  tank base stop
end program
```

Photo:
**Single Motor Control**

Except for controlling the whole chassis, motor 1 or 2 can be directly controlled.

Hardware connection: Same as above.

Code introduction: In initialize code, serial port is opened to send encoder data. Write value and control motor voltage through PWM for 2 seconds. Then write motor speed with encoder feedback for 2 seconds. And the controller send encoder data through serial port. Write 0 to stop the motor.

```
setup
  Serial baud rate 9600

  motor 1 write value(±255) 100
  motor 2 write value(±255) 100
  Delay ms 2000
  motor 1 write RPM(±100RPM) 30
  motor 2 write RPM(±100RPM) -30
  Delay ms 2000
  Serial println "M1 speed:"
  Serial println motor 1 read RPM
  Serial println "M2 speed:"
  Serial println motor 2 read RPM
  motor 1 write RPM(±100RPM) 0
  motor 2 write RPM(±100RPM) 0
end program
```
9.5 Servo Module

9.5.1 Brief Introduction

Servo is an angle-control actuator based on PWM signal. It contains DC motor, reducer, feedback circuit and current control circuit.

9.5.2 Specification

Size: 54 x 20 x 47.2 mm
Servo Type: 55g metal-gear servo
Torque: 9.4kg.cm
Rated Current: 250mA
Blocked Circuit: 1A
Connector: PH2.0 3P

9.5.3 Usage

Servo Calibration

Servo can rotate from 0 to 180 degree. Degree increases clockwise and decreases CCW. Initial degree is 90 degree, with a red mark pointing forward.

Servo is an angle-control device and it uses potentiometer to get the degree. Servo initial degree may has a devistion due to precision of the potentiometer, usually smaller than 10 degrees. Just programme to calibrate the servo.

Code introduction: In initial part, Servo is initialize to 90 degree. Check the actual degree and change the correct value to set the servo to right angle.
Pay attention that this value is the correct value of servo rather than port S1. If the servo changes port, and programme should be changed too.

Servo Rotation

There are two ways to rotate servo. One for setting degree and time, and the other for setting degree and speed and moving together. The first way is often used for single servo rotation, and the second is for multiple servos.

Code introduction: In initial part, servo is calibrated and direction is set. In loop part, control degree and time to move servo for 30 degrees, and then move servos fastly to 150 degree.

Photo:
9.6 Speaker Module

9.6.1 Brief Introduction

Speaker module is a mp3 player controlled by serial instruction. Put sound files to in and control it with controller module.
9.6.2 Specification

Size: 48 x 48 x 11.6 mm
Charging Power: 1W
Supported File: mp3
Memory Space: 16MB

Pinout

9.6.3 Usage

Let the speaker play music, and use the controller button to pause or play.

Connect the speaker to P2 of controller, set play mode to random play and define speaker volume. Detect the state of button A. If it is pressed, play or pause.
setup

speaker init on port 2

speaker set playmode random

speaker volume(0-32) 15

if button A pressed

do

speaker play/pause

repeat while button A pressed

do

else if button B pressed

do

speaker play/pause

repeat while button B pressed

do

Photo:
9.7 Eyes Module

9.7.1 Brief Introduction

Eyes module contains 12 serial RGB LED lights.

9.7.2 Specification

Size: 64 x 32 x 12 mm
LED Type: serial LED
LED number: 12
Color: 16 million colors

Pinout
9.7.3 Usage

Multiple Control

Code introduction: Connect the eyes module to P1 of main controller and define brightness of eyes. Let 12 LEDs show appointed color for 5 seconds, show happy for 5 seconds, and turn off.

Photo:
Single Control

Code introduction: Set LEDs one by one and control the RGB value of every light for 1 second.

```
setup
  eyes init on port 1

count with i from 0 to 11 step 1
  do
    eyes light number i R value 200 G value 155 B value 0
    eyes show
    Delay ms 1000
  eyes clear
  eyes show
end program
```

9.8 Touch Module

9.8.1 Brief Introduction

Touch module is a single touch button.

9.8.2 Specification

Size: 34 x 32 x 9.6 mm

Touch Type: Non-self-locking single touch

Touch Area Size: diameter 14mm
9.8.3 Usage

Code introduction: Connect the touch module to P1 of controller. Detect the module state. Controller LED turn red when touch module is touched, otherwise turn off.

```
setup
    touch sensor init on port 1

if touch sensor read port 1
    do controller LED show
    else controller LED show
```

Photo:
9.9 Infrared Module

9.9.1 Brief Introduction

Infrared module has 2 infrared switch inside, used to detect certain obstacle.
It has short and long modes, which can be used in following line or avoiding obstacle.
9.9.2 Specification

Size: 34 x 32 x 12 mm
Infrared Type: 2 reflective switches
Detect Distance:
short mode 10mm
long mode 110mm

9.9.3 Usage

Code introduction: Connect the infrared module to P3 of controller module and detect 2 switches state.
When both switches detect the obstacle, 2 LEDS of controller turn red. When only one switch detect, turn on 1 LED instead.
9.9. Infrared Module

```plaintext
setup
    IR init on port 3

if
    IR read port 3 pin 1 and
    IR read port 3 pin 2
    do
        controller LED show
    else if
        IR read port 3 pin 2
        do
            controller LED show
    else if
        IR read port 3 pin 1
        do
            controller LED show
    else
        controller LED show
```

Photo:
MOONBOT KIT STRUCTURE INSTRUCTION

MoonBot Kit contains 3 official structures, including MoonRover, MoonMech and MoonBot. Every structure has unique functions, and all parts are packaged in the standard and educational version.

Programme through Mixly, Arduino and MU Bot App, or directly remote control, or even express creative ideas and design your own robot.

10.1 MoonRover Instruction

10.1.1 Introduction

MoonRover is made of sheet metal body and hardware modules. It is driven by a track chassis. The controller module is fixed on top of it, and connected to other devices with wires. Infrared module and vision module can be fixed on the front or bottom of the body, and battery is fixed backwards.

MoonRover can be used to learn applications like avoiding obstacles, following line, auto driving and so on.
10.1.2 Specification

Size: 177 x 157 x 87 mm

Functions
Motion: head, chassis
Sense: vision, infrared, encoder

10.1.3 Build Manual

Download pdf manuals of MoonRover
MoonRover get started guide
MoonRover extended manual
Or watch video guide on youtube
MoonRover video guide
10.1.4 Program Examples

Download MoonBot Mixly examples

MoonRover Examples

Avoid Obstacles

MoonRover becomes a obstacle avoiding car when 2 infrared modules are fixed in front.

Hardware connection: Build the MoonRover with the manual. Infrared sensor on the left is connected to P3 of the controller, and the other is connected to P7. Both sensors should be set to long distance mode.

Code introduction: In initial part, Two infrared sensors are set to the ports, and chassis is calibrated. In loop part, there are three status. When the left/right sensor detect obstacles, the chassis turn right/left. Go forward by default.
Auto Drive

MoonRover becomes a auto driving car when using vision module to navigate.

Hardware connection: Build the MoonRover with the manual. The vision module is connected to P9 of the controller.

Code introduction: In initial part, vision module is connected to serial 3 (P9 port), algorithm is set to traffic card and chassis is calibrated. In loop part, vision module detect traffic card. 5 traffic card refer to 5 status, and MoonRover will move as the card shows.

Phenomenon: After downloading the program, turn on battery. Vision module will shine red after setup. Put a Forward card in front of MoonRover for about 20 centimeters and MoonRover will recognize it and go forward. Change card to change its status. Put different cards on the road to let MoonRover auto drive.
setup
Serial3 baud rate 115200
initialize Mu00 port Serial3

if Mu00 detected TrafficCard
  do
    if get Mu00 algorithm TrafficCard type= Straight
      do
        motor 1 write RPM(±100RPM) 30
        motor 2 write RPM(±100RPM) 30
      do
    end if
  do
  if get Mu00 algorithm TrafficCard type= TurnLeft
    do
      motor 1 write RPM(±100RPM) 30
      motor 2 write RPM(±100RPM) -30
    do
  end if
  if get Mu00 algorithm TrafficCard type= TurnRight
    do
      motor 1 write RPM(±100RPM) -30
      motor 2 write RPM(±100RPM) 30
    do
  end if
  if get Mu00 algorithm TrafficCard type= TurnAround
    do
      tank base turn left(°) 180 RPM(0-100RPM) 30
    do
  end if
  if get Mu00 algorithm TrafficCard type= STOP
    do
      tank base stop
    do
  end if
end if
10.2 MoonMech Instruction

10.2.1 Introduction

MoonMech is a movable mechanical arm. Its body is made by plastic shell and sheet metal frame. The track chassis is driven by motor module, and mech arm is driven by servos. The claw on top of the arm can catch various objects with vision feedback.

MoonMech can be used to learn competitive applications like transpotation and playing basketball.
10.2.2 Specification

Size: 271 x 137 x 244 mm
Functions
Motion: mech arm, claw, chassis
Sense: vision, encoder

10.2.3 Build Manual

Download pdf guide of MoonMech
MoonMech build guide
Or watch video guide on youtube
MoonMech video guide

10.2.4 Program Examples

Download MoonMech Mixly examples
MoonMech Examples

Catch Ball

This example shows how to catch ping-pong ball recognized by vision module.

Code introduction: In initial part, vision module and servos are set to their ports. Vision algorithm is set to ball detect, and servos are set to initial position. In loop part, vision module will detect ball and judge the x offset and move the chassis. Then judge the y offset and move servos. Use the ball width to estimate the distance. When the width is above 27, the claw will close and catch the ball.

Phenomenon: Turn on MoonMech and it will open claw and look forward, with vision module LEDs shining red. Put a ping-pong ball in front of the claw, and the LEDs turn blue. Adjust the position of ball until the claw catch it.
10.3 MoonBot Instruction

10.3.1 Introduction

MoonBot is a semi-humanoid robot with abundant sense and interaction. Its body is made by plastic shell and sheet metal frame. The track chassis is driven by motor module. Head and hands are driven by servos. Eyes and speaker can interact with others with touch, vision and position feedback.

MoonBot can be used to learn service robot applications like reception and patrol.
10.3.2 Specification

Size: 150 x 137 x 216 mm

Functions
Motion: head, hands, chassis
Interaction: eyes, speaker
Sense: vision, touch, encoder

10.3.3 Build Manual

Download pdf guide of MoonBot
MoonBot build guide
Or watch video guide on youtube
MoonBot video guide

10.3.4 Program Examples

Download MoonBot Mixly examples
MoonBot Examples

Shake Body

MoonBot has servos in head and hands, and motors in chassis.Just program to make MoonBot dance.

Code introduction: In initial part, servos are set to head and hands.In loop part, use robot blocks to make MoonBot to move chassis and lands slowly.
Follow people

MoonBot can use vision module to recognize people, and always face people with chassis and head.

Code introduction: In initial part, servos are connected to ports and are adjusted direction according to actual position. Vision module is connected to serial 3(P9), and the algorithm is set to human body. In loop part, when detect human body, MoonBot will move chassis according to x offset of human body, and move head according to y position.

Phenomenon: After downloading the program, put MoonBot on desk and stand in front of it. MoonBot will detect upper body and vision module shines blue when detected. Walk around and MoonBot will rotate to keep the face in front of you all the time.
This article introduces how to connect MoonBot Kit with MU Bot App on mobile phone or tablet.

### 11.1 MoonBot Kit APP Firmware Upgrade Guide

Programming with MoonBot Kit APP requires burning the specified firmware in the master control. This article guides users how to upgrade MoonBot Kit master module to burn firmware needed for APP programming.

#### 11.1.1 Preparation

**Hardware:**
- MoonBot kit
- PC (Windows, Linux, MacOS)

**Software:**
- Arduino Official IDE
- MoonBot Remote control Arduino Source code or firmware

#### 11.1.2 Upgrade by burning HEX files

1. Download MoonBot Firmware of Master Control Remote Controller (.hex File)
2. Download Arduino Hex Upload Tool
3. Burn .hex firmware
   - Windows

   1. Download Arduino Hex Burning tool
   2. Select MoonBot port, Hardware select ‘Genuino ATmega1280’
   3. Click download and wait for download to complete.
11.1.3 Upgrade Arduino source code by compiling Arduino IDE

- 1. Building MoonBot Kit Arduino development environment
- 2. Download MoonBot Kit Master remote control source code (Source.zip File)

- 4. Select the ZIP file downloaded in step 2 and click OK to load the source code of MoonBot Kit master remote control
- 5. Click Arduino File->Example>MoonBotRemote->RemoteWithDemo, Open Source code
• 6. Connect MoonBot Kit Master Control to Computer. Click Arduino Tool->Port, Select the corresponding MoonBot port.

• 7. Click the Download button and wait for the download to complete.
11.1.4 Test

- 1. After restart, press the main control button A, close to the A key LED bright blue light and give a prompt sound.
- 2. After restart, press the main control button B, close to the B key LED bright green light and give a prompt sound.

11.2 APP Remote Control

First select the remote controller, and then chooses the corresponding robot form.

11.2.1 Introduction to Controller Use

1. WiFi Connection
   
   Click on the combination of color blocks that match the color of the LED light of the VisionSensor to start the connection. If the connection fails, press the visual module reset key to re-select the connection.
2. Robot form selection

Choose the corresponding form among the three.

3. Setting Function Button

Click on the Setup Function Key, the circle box below the Function Key is dotted.

Click on the dotted circle box to add, delete and replace each function into the circle box. Click on the Setup Function Key to complete the setup.

4. Wheel control

When WiFi is connected, the motion of the robot can be controlled by the wheel disk.

5. Functional Button

Click on the set function key to make the robot act accordingly. Some functions can be turned on/off.
Morpx Documentation

MoonBotCar: Visual Angle Up/Down, Tracking Ball, Obstacle Avoidance, Card Identification

MoonBotMech: Arm up/Down, Visual Angle Up/Down, Open/Close Claw, Catch Ball, Shoot

MoonBot: Left Arm Up/Down, Right Arm Up/Down, Head Angel Up/Down, Dancer 1/2, Be cute
6. LED Button and Sound Button

There will be lights/sounds when you click on the button.

### 11.2.2 Example

Remote Control.gif
11.3 APP Programing

11.3.1 Introduction to Programming Use

1. WiFi Connect
   
   Click on the combination of color blocks that match the color of the LED light of the VisionSensor to start the connection. If the connection fails, press the visual module reset key to re-select the connection.
2. Exit Button
3. The button to be updated
4. Help Button
Click on the button and there will be instructions for each button.
5. Share Button
Click to share the program with friends / QQ / Wechat / Wechat Friends Circle
6. Project Preservation Button
The program can be named and saved to my project.
7. My Program
Select your own saved project to open
8. Program Block

11.3. APP Programming
Contains various programming blocks

9. Play Button
Execute transmission

11.3.2 Example

Program.gif

11.4 APP Programming Block_Artificial intelligence

11.4.1 Artificial intelligence

Algorithm enable/Disable

Algorithm: ball, body, shape card, traffic card, number card, color block detection, color recognition
Parameter: enable/Disable

Instructions
Ball Algorithm: Identify orange ping-pong (Label:1) and green tennis (Label:2)
Body Algorithm: Detection of upper body characteristics
Shape/Traffic/Number Card: Identify specific cards
Color block detection: Setting a color and detecting its block area

Color recognition: Specify an area to detect its color

One or more algorithms can be open at the same time

Setting Block Detection Colors

Color: Black, white, red, yellow, green, cyan, blue, purple

Instructions
The color block detection algorithm detects red by default. It can change the color of detection by using this function block.

Reading algorithm to detect parameters

Algorithm: ball, body, shape card, traffic card, number card, color block detection, color recognition
Setting parameters: is detected, X, Y, width, height, label
Is detected: True when detected and False when not detected
Morpx Documentation

X coordinate, Y, width, height: Quantify to (0-100)
Classification number: Label

Seeing algorithm block

Algorithm: ball, body, shape card, traffic card, number card, color block detection, color recognition

Algorithm azimuth position block

Algorithm: ball, body, shape card, traffic card, number card, color block detection
Setting parameters: center, up, down, left, right

Algorithm size block

Algorithm: ball, body, shape card, traffic card, number card, color block detection
Setting parameters: big, normal, small

Algorithmic area location block

Algorithm: ball, body, shape card, traffic card, number card, color block detection
x: 1/2/3/4/5
y:1/2/3/4/5

**Shape card block**

![Shape Card Check](image)

Parameters: check, close, circle, square, triangle

**Traffic card block**

![Traffic Card Forward](image)

Parameters: forward, left, right, turn around, park.

**Number card block**

![Number Card 0](image)

Parameters: 0–9

**Ball Recognition block**

![Ball Recognition Ping Pong (orange)](image)

Parameters: ping-pong (orange), tennis (green)
Color recognition block

Parameters: black, white, red, yellow, green, cyan, blue, purple

Detection of gestures block

Parameters: Up, Down, Left, Right, Any

Something Approaching to MU block

Parameters: any, far, middle, near.

Ambient luminance block

Parameters: darkest, dark, good, bright, brightest

Set camera zoom block

Parameters: 1/2/3/4/5

Instructions
If the Zoom value is small, the field of vision is wide and the distance is close.
If the Zoom value is large, the field of vision is narrow and the distance is long.

**White balance mode block**

![White Balance](image)

Parameters: auto, lock, white light, yellow light.

**Instructions**

Automatic mode: suitable for use in environments with good lighting and low color requirements
Lock-in mode: suitable for the environment with high color requirement, let MU calibrate the white balance on the white paper, then lock the white balance parameters, the color will not change with the change of the environment after lock-in.
White light mode: suitable for white light or overcast environment, this mode also belongs to automatic white balance mode;
Yellow light mode: suitable for use in yellow light or sunshine environment, this mode also belongs to automatic white balance mode.

**Lighting settings for MU detection**

![MU If Detect Show](image)

LED lamp color detected: close, blue, green, cyan, red, purple, yellow, white, random
LED lamp color undetected: ibid.
Lighting brightness: 1-10, the greater the value, the brighter
11.5 APP Programming Block_Input

11.5.1 Input

Read button pressed status

Parameter: A, B, A+B
Return: button pressed / not pressed

Initialization of Touch Sensor Port

Parameter: Port3, Port5, Port7, Port8
Initialize the infrared sensor port and read infrared sensors

Port Parameter: Port3, Port5, Port7, Port8
infrared sensor IR1:IR2 Parameter: 0:0, 0:1, 1:0, 1:1
0 means undetected, 1 means detected

Read the compass toward (0-360°)

Return: The compass faces the angle

Read acceleration value (1024/g)

Parameter: X direction, y direction, Z direction, strength value
Return: Acceleration value

Reading rotation angle (°)

Parameters: pitch (x), roll (y), read the main control tilt angle
Return: Angle value (-180° to +180°)
Read thermometer values

Return: Temperature value

Read IMU actions

Parameters: vibration, free falling, X-axis up, X-axis down, Y-axis up, Y-axis down, Z-axis up, Z-axis down, 3g, 6g, 8g

 Calibration compass

Compass calibration module, the main control in calibration needs to be flipped in the shape of "∞"
11.6 APP Programming Block_Light

11.6.1 Light

Eye light setting block

Eye parameters: all, left and right eyes
Color parameters: close, blue, green, cyan, red, purple, yellow, white, random
Luminance parameters: 1-10, the greater the value, the brighter

Expression block

Parameters: happy, sad, angry, blink, turning, flashing, rainbow, closed eyes
Return: Show expression
Eye preset RGB value of each LED lamp

LED light: 1-12 and all lights
Red: 0-255
Green: 0-255
Blue: 0-255

Show eye preset

Main control LED lamp setting block

Main LED parameters: all, 1, 2
Color parameters: close, blue, green, cyan, red, purple, yellow, white, random
Luminance parameters: 1-10, the greater the value, the brighter

MU LED light setting block

MU LED lamp parameters: all, 1, 2
Color parameters: close, blue, green, cyan, red, purple, yellow, white, random
Luminance parameters: 1-10, the greater the value, the brighter
### 11.7 APP Programming Block_music

#### 11.7.1 Music

**Speaker Plays Sound block**

![Speaker Plays Sound block](image)

Play the specified sound: Animal Voice, Greetings, Piano, City, Drum, Custom Voice

**Volume Selection Block**

![Volume Selection Block](image)

Parameters: 0-10, the larger the value, the larger the volume.
Play mode block

Parameters: single play, single loop

Speaker stop block

Stop playing sound

Buzzer play block

Buzzer to play scales in a set beat
Scales: aerial, do1-si7, Do1-Si7
Rhythm: 1/8-4 beats

Buzzer play sound block

Play the specified sound
Parameters: key tone 1-4, alarm 1-2, sound effect 1-4, ambulance sound, siren sound
11.8 APP Programming Block_ motion

11.8.1 motion

Setting steering angle block

Steering port: steering1~steering4
Angle: 0~180°

Read steering angle block

Read the specified steering angle
Parameters: steering gear 1-4
**Forward**

![Move Forward](image)

Distance set to advance at specified gear speed
Execution distance: 0–999cm
Speed parameters: very fast, fast, medium, slow, very slow

**Back off**

![Move Backward](image)

Back-set distance at specified gear speed
Execution distance: 0–999cm
Speed parameters: very fast, fast, medium, slow, very slow

**Turn Left**

![Turn Left](image)

Turn left at the specified gear speed and set the angle
Execution angle: 0–999°
Speed parameters: very fast, fast, medium, slow, very slow

**Turn right**

![Turn Right](image)

Turn right at the specified gear speed and set the angle
Execution angle: 0–999°
Speed parameters: very fast, fast, medium, slow, very slow
Stop Motion block

Writing speed of Motor block

Write a certain speed to the motor (-100~+100R/min)
Parameters: Motor 1, Motor 2

Read motor speed block

Parameters: Motor 1, Motor 2

Calibration of alignment migration block

Calibrate the alignment migration so that it does not migrate in a certain direction.
Parameters: 0-200, > 100 to the right and < 100 to the left

Calibration of alignment distance block

Calibrating the inaccuracy of the direct distance caused by external interference
Parameters: > 100 Increase Distance, < 100 Reduce Distance

**Calibration of rotate angle block**

![Calibrate TankBase Rotate Angle](image)

Incomplete turning angle caused by calibration external interference
Parameters: > 100 Increase turning angle, < 100 Reduce turning angle

**Calibration of steering angle block**

![Calibrate Servo1 Degree](image)

Calibration of angle error in production and installation by rudder calibration module
Parameters: steering gear 1-4
Increase/decrease(-90~+90°)

### 11.9 APP Program Example

#### 11.9.1 Touch Wave

MoonBot has steering engine in its hand and touch sensor in its head. Touch beckoning can be realized by programming.

Explain: Cyclic detection of touch sensor status, when the left side of the head is touched, the robot beckons the left hand. When the right side of the head is touched, the robot waves its right hand.
11.9.2 Simple Algorithm

MoonBot uses VisionSensor and LED Module.

Explain:Cyclic detection of the ball algorithm, when the ball is detected, the eyes turn around the expression, when not detected closed eyes.
11.9.3 Examples of functional modules

MoonBot robot uses button speaker, LED lamp, actuator lamp and mathematic module.

Explain: Cyclic detection of button A/B status. The MoonBot robot makes sound and light arm movements when the button is pressed.
11.9.4 Search sb.

MoonBot Using VisionSensor and Motion Modules

Explain: Turn on Body Detection Algorithms, When the body is not detected, the VisionSensor LED flashes red light, when the body is detected, the VisionSensor LED flashes blue light.

When the body is detected in the center, the robot stops moving, otherwise it turns left/right.
11.9.5 Barrier Avoidance Intelligent Vehicle

Install infrared sensors on left and right sides of intelligent vehicle car

Explain: Calibration chassis, Turn left when the right infrared sensor of the smart car detects obstacles, turn right when the left infrared sensor detects obstacles, and go straight when none of them is detected.
11.9.6 Traffic Intelligent Vehicle

Intelligent Vehicle Cooperative Traffic Card Algorithms

Explain: Calibration chassis, turn on the traffic card algorithm, set the camera zoom level and set the white balance parameters according to the lights.

When traffic card is not detected, the VisionSensor LED flashes red light and blue light when the traffic card is detected.
This article introduces MoonBot Kit developing tutorial with Mixly platform.
Visit Mixly official docs for basic tutorials: Mixly wiki

12.1 MoonBot Mixly Guidelines for Programming Construction

There are two ways to install MoonBot Mixly. Download the full package and unzip, or install independent library if you already installed Mixly.
Instructions are shown below.

12.1.1 Full Installation Package Download

Windows/Linux/Mac Full Edition MoonBot Mixly Package Download Link: https://drive.google.com/drive/folders/1L_FKMIQnrdgi_rLniOYbA9o9q24RGE?usp=sharing

12.1.2 Independent Library Installation

Import and upgrade Mixly-Arduino Library

- 1. Start Mixly-Arduino
  - Windows
    Open $(your_mixly_path)/arduino-1.8.5/arduino.exe file under the Mixly installation path, start Arduino
  - Linux
    Running Arduino files at the terminal, start Arduino

    ```
    $ cd $(your_mixly_path)
    $ ./arduino-1.8.2-linux64/arduino
    ```

- 2. Click on Project -> Load Library -> Manage Library', openLibrary Manager'.
3. Search library **AsyncDelay**, install the relevant library if not installed and update if the library is not the latest edition.
4. Install the library `Software Wire `Adafruit_NeoPixel.Servo' according to the installation method of the third step, ensure that the relevant library is installed to the latest version.

5. Close Arduino, Complete the installation of the base library.

**Import Mixly Library**

1. Click Download `MoonBot/MuVisionSensor3` The latest version of Mixly Library Compression Pack
2. Upzip the downloaded `MoonBot/MuVisionSensor3` compression package
3. Open the Mixly interface, click the import button, find the files at the end of `xml` under the `MoonBot/MuVisionSensor3` folder path.
4. Library installation completed

### 12.2 API Reference resources

The programming blocks of MoonBot Kit and MU Vision Sensor 3 are customized in MoonBot Mixly. This article will explain the program blocks one by one, as well as some complex program examples. It can be combined with previous hardware module examples for learning.

Mixly Basic Tutorial in Mixly Help Document, will not repeat it here.

12.2.1 Input

Input include MoonBot Kit Touch Module Infrared Module Controller Module keys and pin mapping module

**Initialization of Touch Sensor**

- **touch sensor init on port 1**
- **IR init on port 1**
- **button A pressed**
- **touch sensor read port 1**
- **IR read port 1 pin 1**
- **port 1 pin 1**

**Introduction**

Initialize the touch sensor to the corresponding port.

- **port**
  - 1~9
Reading Touch Sensor

**Introduction**  Read the value of the corresponding port of the touch sensor

**Parameters**

- **port**
  - 1~9

**Return**

- **HIGH**: Object Touch Sensor
- **LOW**: Objectless Touch Sensor

Initialization of Infrared Sensors

**Introduction**  Initialize touch sensor to corresponding port

**Parameters**

- **port**
  - 1~9

Reading Infrared Sensor

**Introduction**  Read the corresponding pin value of infrared sensor port

**Parameters**

- **Port**
  - 1~9
- **Pin**
  - 1~2

**Return**

- **HIGH**: Infrared Sensor Triggered
- **LOW**: Infrared sensor not triggered
Reading Button

**Introduction**  Read button status

**Parameters**

- **button**
  - A: Button A
  - B: Button B
  - A&B: Button A and B

**Return**

- **HIGH:** The Button is pressed
- **LOW:** The Button is not pressed

Port pin mapping

**Introduction**  Reading the Arduino pin number corresponding to the MoonBot port

**Parameters**

- **Port**
  - 1~9

- **Pin**
  - 1~2

**Return**

- Corresponding Arduino pin

### 12.2.2 Chassis Control

**Chassis Control** include drives MoonBot Kit *Motor Module* and driving of Encoder in Motor.

By calling these modules, the motor chassis can move.
Reversal direction

**Introduction**  The direction of motion of the flip motor.

**Parameters**

**Reversal direction**

- true: Reversal direction
- false: Default direction
Straight-line offset correction

**Introduction**  Because of friction, installation deviation and other disturbances, the chassis will be offset in a certain direction when it goes straight.

Direct migration caused by external disturbance can be corrected by `direct migration correction` module.

**Introduction**

**Straight-line offset correction** (%)

- 0~200 >100 Correction to the right,<100 Correction to the left

Straight Distance Correction

**Introduction**  Because of friction, installation deviation and other disturbances, chassis traveling a certain distance will have the situation of inadequate direct travel.

Through the `direct distance correction` module, the situation of the out-of-place direct distance caused by external disturbance can be corrected.

Before correcting the straight-line distance, it is suggested that `straight-line migration correction` be carried out first.

**Introduction**

**Straight Distance Correction** (%)

- 0~+∞ >100 Increased direct distance,<100 Straight distance decreases

Turning Angle Correction

**Introduction**  Because of friction, installation deviation and other disturbances, chassis rotating at a certain angle will have the situation that the turning angle is not in place.

Through the `turning angle correction` module, the situation that the turning angle caused by external disturbance is not in place can be corrected.

Before correcting the turning angle, it is suggested that `straight-line offset correction` and `straight-line distance correction` should be carried out first.

**Introduction**

**Turning Angle Correction** (%)
• $0 \rightarrow +\infty > 100$ Increased turning angle, $< 100$ Reduced turning angle

**Forward**

**Introduction**  Control the chassis to move forward at a given speed until it stops at a given distance.

The module ** calls the encoder module ** to ensure that the corresponding encoder has been connected to the corresponding port.

**Parameters**

- **Forward Distance (cm)**
  - Distance value: Given straight distance, Unit: cm
  - Speed value: Given Speed of Direct Motor, Unit: RPM

**Backward**

**Introduction**  The control chassis runs backwards at a given speed until it stops at a given distance.

The module ** calls the encoder module ** to ensure that the corresponding encoder has been connected to the corresponding port.

**Parameters**

- **Backward distance (cm)**
  - Distance value: Given straight distance, Unit: cm
  - Speed value: Given Speed of Direct Motor, Unit: RPM

**Left turn**

**Introduction**  Control the chassis to turn left at a given speed to a given angle and stop.

The module ** calls the encoder module ** to ensure that the corresponding encoder has been connected to the corresponding port.

**Parameters**
Left turn angle(°)
- Angle value: Given a straight angle, Unit: °

speed
- Speed Value: Given Speed of Direct Motor, Unit: RPM

Right turn

**Introduction**  Control the chassis to turn right at a given speed to a given angle and stop.

The module ** calls the encoder module ** to ensure that the corresponding encoder has been connected to the corresponding port.

**Parameters**

Right turn angle(°)
- Angle value: Given a straight angle, Unit: °

speed
- Speed Value: Given Speed of Direct Motor, Unit: RPM

Stop

**Introduction**  The chassis stops turning.

Motor write-in value

**Introduction**  Write the analog to the motor at the corresponding port.

**Parameters**

Motor port
- 1: Motor port 1
- 2: Motor port 2

value
- ±255: Write the value of the analog, >0 Turn Forward, <0 Turn back, =0 Stop turning
Reading motor value

**Introduction**  Read the analog value of the corresponding motor port.

**Parameters**

- **Motor port**
  - 1: Motor port 1
  - 2: Motor port 2

- **Return**
  - `value`: Value of motor analogue

Writing Speed of Motor

**Introduction**  Write the speed to the motor at the corresponding port.

The module calls the encoder module to ensure that the corresponding encoder has been connected to the corresponding port.

**Parameters**

- **Motor port**
  - 1: Motor port 1
  - 2: Motor port 2

- **value**
  - ±60: Write the value of the analog, >0 Turn Forward, <0 Turn back, =0 Stop turning, unit: RPM

Reading motor speed

**Introduction**  Read the speed of the corresponding motor port.

**Parameters**

- **Motor port**
  - 1: Motor port 1
• 2 : Motor port 2

Return

• speed : motor speed , unit: RPM

### 12.2.3 Steering engine

**Steering engine** include MoonBot Kit *Servo Module* drives. It can be used to drive the steering gear connected to the four rudder ports in MoonBot Kit. So the actuator with one or more ports can move simultaneously.

**Setting Angles**

**Introduction** Write the angle to the steering gear connected to the specified steering port.

**Parameters**
steering gear port
• 1~4

angel
• 0~180°

Reading angel

Introduction  Reads the current angle value of the specified steering port.
Parameters
steering gear port
• 1~4

Presupposition angle

Introduction  Preset the steering angle and speed of the specified steering gear port.

The module should be used in conjunction with the `synchronous movement of all steering gear to the preset angle` module.
Parameters
steering gear port
• 1~4
speed
• fast : Set the speed of steering gear to be fast about 150°/s
• mid : Set the steering gear running at medium speed about 100°/s
• slow : Set the speed of steering gear to slow about 50°/s
**Synchronize all steering gear to preset angle**

Introduction  Move all steering gear to the preset angle. The module needs to be used in conjunction with the `preset angle` module.

**Reversal direction**

Introduction  The steering angle is reversed with 90 degree as the median value.

Parameters

- `false`: Default Motion Direction
- `true`: Turn the steering gear in the direction of motion

**Correcting**

Introduction  Errors in production and installation of gears and steering gear disks may cause steering gear to fail to turn at specified angles.

The angle error caused by the above reasons can be corrected by the steering gear calibration module.

Parameters

- `±90°`

**12.2.4 Music**

Music include MoonBot Kit Controller Module buzzer drive and external Drive Speaker Module.

By calling these modules, you can control MoonBot Kit to play music.
Speaker initialization

**Introduction**  Initializes the speaker connected to the specified port.

**Parameters**

- **port**
  - 2, 7, 9
**Speaker Setting Play Mode**

**Introduction**  Set up the playback mode of the speaker.

**Parameters**

- **Play mode**
  - Single Play: Stop playing after playing specified music
  - Single tune circulation: Play specified music in a loop
  - Play all: Play the next music in the music list automatically after playing the specified music
  - Random Play: Play one of the music lists randomly after playing the specified music

---

**Speakers Play Music**

**Introduction**  Play music with a given name.

**Parameters**

- **Music Name**
  - The drop-down menu for the module

---

**Speaker Plays Custom Music**

**Introduction**  Play music with the specified music name.

Users need to put corresponding custom music into loudspeakers before this operation. (*How to save music?*) The first four words of a musical name should be letters or numbers.

**Parameters**

- **Music Name**
  - Customize the first four characters of the music name. Support only **English** or **Numbers**
### Speaker Play Setting

**Introduction**  Set the current speaker playback status.

**Parameters**

**Play settings**

- **Play/pause**: Play or pause current music
- **Next song**: Play the next music in the music list
- **Last song**: Play the last music in the music list
- **Stop**: Stop playing music

### Loudspeaker set volume

**Introduction**  Set the loudspeaker volume.

**Parameters**

**volume**

- 0~32

### Buzzer Plays Scales

**Introduction**  Buzzer to play scales in a set beat

**Parameters**

**Scale**

- High, middle and low levels

**Rhythm**

- 1/16~4 beat: Single beat time can be set by buzzer.
Buzzer pauses play

**Introduction**  The time when the buzzer pauses to play a given beat.

**Parameters**

- **Rhythm**
  - 1/16~4 beat: Single beat time can be set by buzzer.

Buzzer Sets Play Rhythm

**Introduction**  Set the number of beats per minute (BPM) of buzzer.

**Parameters**

- **beats per minute**

Buzzer Play Frequency

**Introduction**  Set up a buzzer to play music at a specified frequency at a given time.

**Parameters**

- **frequency**
  - 0~65535: Frequency Recommendation Setting in the Frequency Range acceptable to the Human Ear[20~20000Hz]

- **time**
  - 0: Continuous broadcasting
  - other: Stop playing for a specified length of time
Buzzer stop playing

**Introduction**  The buzzer stopped playing sound.

### 12.2.5 IMU

IMU include MoonBot Kit *Controller Module* three-axis acceleration, drive of three-axis magnetometer and temperature sensor on board.

By calling these modules, you can get MoonBot Kit master control of current direction, tilt angle and state, etc.
Compass calibration

**Introduction** When calibrating the compass, the master control needs to flip in the shape of "∞".

Acquisition of compass angle

**Introduction** Read the angle between the current direction and the northward direction of the compass Y axis.

**Return**

- 0~360°

Acquisition of acceleration value

**Introduction** Read the acceleration simulation of a given axis.

**Parameters**

- **Directional axis**
  - X, Y, Z

**Return**

- Acceleration analogue

Acquisition of acceleration angle

**Introduction** Acquisition the tilt angle of the master control

**Angle type**

- **Elevation angle**: The angle between Y axis and horizontal plane in the master control coordinates. When the main control tilts upward, the pitch angle is positive and vice versa.

- **Roll angle**: The angle between X axis and horizontal plane in the master control coordinates. When the main control tilts to the right, the roll angle is positive and vice versa.
Return

• ±180°

Read Temperature

Introduction  Read the current temperature

Parameters

Unit of temperature

• °C : Celsius degree
• °F : Fahrenheit degree

Return

• Temperature value

Read the current status

Introduction  Read the current master control state.

Parameters

state

• shock : Whether the master control is in vibration state or not
• Free fall : Whether the master control is in free falling state

Return

• true : The master control is currently in this state
• false : The master control is not currently in this state

12.2.6  Light

Light include MoonBot Kit Controller Module Two on-board LEDs and 12 external LEDs Eyes Module drivers. With these modules, you can easily set the color and brightness of the LED.
Eye Initialization

Introduction  Initialize the eye module to the specified port.

Parameters  

port

• 1~9
Eye display

**Introduction**  Write the color value of the eye LED into the buffer and display it.

**Parameters**

- colour

---

Eyes show expression

**Introduction**  Eye LED displays facial movements.

**Parameters**

- Expression
  - `Module Dropdown Menu`

---

Master control LED display

**Introduction**  Write the color value of the master LED into the buffer and display it.

**Parameters**

- colour

---
LED Setting RGB Value

**Introduction**  Write the RGB color value of a given LED lamp into the buffer.

**Parameters**

- **LED Type**
  - Eye: `eye LED`
  - master control LED: `mastercontrolLED`

- **Lamp number**
  - Eye: 0~11
  - master control LED: 0~1

- **R value**
  - 0~255
  - `Red Channel Analog`

- **G value**
  - 0~255
  - `Green Channel Analog`

- **B value**
  - 0~255
  - `Blue Channel Analog`

LED Setting HSV Value

**Introduction**  Write the HSV color value of a given LED lamp into the cache.

**Parameters**

- **LED Type**
  - Eye: `eye LED`
  - master control LED: `mastercontrolLED`

- **Lamp number**
  - Eye: 0~11
  - master control LED: 0~1

- **H value**
  - 0~360°: `Tone Value`
S value
- 0~255: Saturation value analogue

V value
- 0~255: Luminance value analogue

LED show

**Introduction** Show the color values in a given LED buffer

**Parameters**

**LED Type**
- Eye: Eye LED
- master control LED: master control LED
LED clear

**Introduction**  Clear the cache of the specified LED.

**Parameters**

**LED Type**

- Eye : Eye LED
- master control LED : master control LED

LED brightness

**Introduction**  Set the brightness of the given LED.

**Parameters**

**LED Type**

- Eye : Eye LED
- master control LED : master control LED

**brightness**

- 0~255 : 0 darkest, “255” brightest

12.2.7 Mech

**Mech** include mech Integrative Action in Morphology

By calling these modules, you can easily control the manipulator to catch the ball and so on.
Initialization

Introduction  Initialize the MoonMech manipulator port.

Parameters

MU adress

- MU00  MU address 0x60
- MU01  MU address 0x61
- MU10  MU address 0x62
- MU11  MU address 0x63

MU Port
Setting the position of catch ball

Introduction  Setting up the position of the MoonMech manipulator to grasp the ball. The mechanical gripper can catch the ball by adjusting the X-Y value of the recognition ball.

When the ball is within the given X-Y value range, the mechanical claw closes to grasp the ball.

Parameters

X

• 0~100 : Horizontal position of mechanical claw grip ball. The horizontal position of the mechanical claw relative to the ball can be adjusted by modifying this value.

Y

• 0~100 : Vertical position of mechanical claw catching ball. The vertical height of the gripper can be adjusted by modifying this value.

Setting Shooting Conditions

Introduction  Setting up the conditions of MoonMech mechanical claw shooting. The horizontal position X and width of the card can be adjusted to allow the manipulator to shoot accurately into the basket.

When the card is within the given X-width range, the arm triggers the shooting action to shoot.

Parameters

X

• 0~100 : Horizontal position relative to transverse coordinate X of card when shooting with mechanical claw. The horizontal position of the gripper relative to the card can be adjusted by modifying this value.

Width
• 0~100 The size of the card when the mechanical claw shoots, the distance between MoonMECH mechanical arm and basket (card) can be adjusted by modifying this value.

Claw movement

**Introduction** Set up mechanical claw action. This module can be used to control the horizontal or up-down translation of the mechanical claw.

**Parameters**

- **action**
  - open [Open the mechanical claw 110°]
  - close [Close the mechanical claw 90°]
  - forward [The mechanical claw advances horizontally in a unit.]
  - back [The mechanical claw retreats one unit horizontally]
  - up [The mechanical claw is vertically upward in a unit.]
  - down [A vertical downward unit of a mechanical claw.]

Find the ball

**Introduction** Control MoonMech manipulator to perform ball searching.

**Return**

- true: Find the ball
- false: No ball was found.

Catch the ball

**Introduction** Control MoonMech manipulator to perform catch action.

If the ball manipulator is not found during execution of this block, MoonMech will remain in place and return false.

**Return**

- true: Catch the ball
- false: No ball was found.
Find the card

Introduction The MoonMech manipulator is controlled to perform the search basket (card) action.

Parameters
card type
  • shape card
  • traffic card
  • number card

Return
  • true: Find the given card
  • false: No given card was found

Shoot

Introduction Control MoonMech manipulator to execute shooting action.

If the ball manipulator is not found during execution of this block, MoonMech will remain in place.
12.2.8 MoonBot

MoonBot include MoonBot robot Integrative Action in Morphology.

By calling these modules, you can control the robot to perform nodding, waving and other actions.

**Initialization**

- **Introduction** Initialize the ports of the MoonBot robot.
- **Parameters**
  - Head steering gear
Morpx Documentation

- 1~4

**Left-handed steering gear**
- 1~4

**Right-handed steering gear**
- 1~4

**Wave**

- **Introduction** The arm that drives the robot waves.
- **Parameters**
  - **Arm**
    - Left hand
    - Right hand
    - Both hands
  - **deviation**
    - 0~90°: Deviation Angle of Up and Down Waves of Robot
  - **Speed**
    - fast
    - mid
    - slow

**Swing**

- **Introduction** The head and foot of the robot are swing at the same time.
- **Parameters**
  - **Motor**
    - Left motor
    - Right motor
    - Dual motor
  - **Speed**
    - fast
Shake your body from side to side

**Introduction**  Control the robot motor to sway left and right.

**Parameters**

- **Speed**
  - fast
  - mid
  - slow

- **Time**
  - \(0/2^\infty\) : Single shaking time of motor

Step Forward

**Introduction**  Controlling the robot takes a step forward.

**Parameters**

- **Speed**
  - fast
  - mid
  - slow

- **Time**
  - \(0/2^\infty\) : The longer the motor takes a step forward, the bigger the step is.
Nod

**Introduction**  The control robot nodded once.

**Parameters**

- **deviation**
  - 0~90° : Nodding range

- **Speed**
  - fast
  - mid
  - slow
MOONBOT KIT ARDUINO TUTORIAL

This article introduces developing MoonBot Kit with Arduino IDE.

13.1 MoonBot Kit Arduino Guidelines for Building Development Environment

MoonBot Kit (hereinafter referred to as MoonBot) provide Arduino library functions, support for development programming on Arduino (ATmega1280).

This document aims to guide users to build MoonBot hardware development environment based on Arduino official IDE.

13.1.1 Preparation

Hardware:

- MoonBot Developer Suite
- PC (Windows, Linux or Mac OS)

Software:

- Arduino Official IDE
- MoonBot Arduino library

13.1.2 Detailed installation steps

First step: MoonBot Import Arduino External Dependency Library

- 1. Start Arduino official IDE
- 2. Click on Project -> Load Library -> Manage Library', open Library Manager.'
3. Search library 'AsyncDelay', install the relevant library if not installed, update if the library is updated
• 4. Install the library `Software Wire' Adafruit_NeoPixel' Servo according to the installation method in step 3. Ensure that the relevant libraries are installed in the latest version.

**Step 2: import MoonBot Arduino library**

• 1. Download the latest MuVisionSensor3 Arduino library and MoonBot Arduino library(Source code(zip))

• 2. Click on the 'Project - > Load Library - > Add. zip Library', select the MounBot Arduino Library downloaded in the first step, complete the import of the library.
Step 3: Connecting devices

Now connect your MoonBot to PC, device connection and port configuration

- 1. Click on Tool -> Development Board, select "Arduino/Genuino Mega or Mega 2569"
- 2. Click on Tool -> Processor, select ATmega1280
- 3. Click on Tool -> Port, Select the corresponding MoonBot port

Generally, serial ports display different names under different operating systems:

- **Windows Operating System:** COM1 etc.
- **Linux Operating System:** Start with `dev/tty`
- **MacOS Operating System:** Start with `dev/cu`
Step 4: Compile routines

1. Click on ‘File - > Example - > MoonBot’, select one of the routines

2. Click on the upload button. If everything goes well, the development board will be reset and the corresponding routine will start running after the burning is completed.
13.2 API Reference

There are customized programming blocks in MoonBot Arduino used for MoonBot Kit and MU Vision Sensor 3. This article introduces every block, and shows some complicated program examples. Users can learn programming combining hardware modules.

Check MU Vision Sensor 3 documents here: MU Vision Sensor 3 Guide

13.2.1 Pin Map

Overview

MoonBot Kit Controller Module contains 9 GPIO ports, 4 servo ports, two motor ports and other on board resources. And in Arduino library we provide pin map library and relevant defined pin to guide users requiring corresponding pin number.

Through these functions and macros, we can easily get pin numbers of MoonBot Kit controller.

Get Keys Status

For example, turn on the on-board LEDs by getting button status.

```cpp
#include <MoonBot.h>

int button_a = MOONBOT_PIN_BUTTON_A;    // Get button A pin number
int button_b = MOONBOT_PIN_BUTTON_B;    // Get button B pin number

void setup()
{
  LED.begin();    // On board LEDs begin
}

void loop()
{
  if (!(digitalRead(button_a) && !digitalRead(button_b))) {
    // If button A and B is pressed at mean time, LED 0 and 1 show cyan.
    LED.setPixelColor(0, 0x00ffff);
    LED.setPixelColor(1, 0x00ffff);
    LED.show();
  } else if (!(digitalRead(button_a))) {
    // If only button B is pressed, LED 0 show green
    LED.setPixelColor(0, 0x00ff00);
    LED.setPixelColor(1, 0x000000);
    LED.show();
  } else if (!(digitalRead(button_b))) {
    // If only button A is pressed, LED 1 show blue.
    LED.setPixelColor(0, 0x000000);
    LED.setPixelColor(1, 0x0000ff);
    LED.show();
  } else {
    // If buttons are not pressed, LEDs turn off.
    LED.setPixelColor(0, 0x000000);
    LED.setPixelColor(1, 0x000000);
    LED.show();
  }
}```
Get LED Eyes Pin Number

Initialize LED by getting the status of *Eyes Module* on port 3.

```cpp
moonbot_eyes.setPin(moonbotPortToPin(kPort3, kPortPin1));  // Set port 3 as the
->first pin of LED eyes.
moonbot_eyes.begin();  // Initialize LED eyes
```

Get Touch Module status

Read status of *Touch Module* on the GPIO ports.

```cpp
#include <MoonBot.h>

// connect touch sensor 1 on port 1
uint8_t touch1 = moonbotPortToPin(kPort1, kPortPin1);
// connect touch sensor 2 on port 2
uint8_t touch2 = moonbotPortToPin(kPort2, kPortPin1);

void setup()
{
    // initialize touch sensor 1/2 as INPUT_PULLUP
    pinMode(touch1, INPUT);
    pinMode(touch2, INPUT);
}

void loop()
{
    Serial.println("================");
    Serial.println("touch sensor1: ");
    // read touch sensor 1 state
    if (digitalRead(touch1)) {
        Serial.println("on touch");
    } else {
        Serial.println("not touch");
    }
    Serial.println("touch sensor2: ");
    // read touch sensor 2 state
    if (digitalRead(touch2)) {
        Serial.println("on touch");
    } else {
        Serial.println("not touch");
    }
}
```
Get Infrared Module Status

We can use the same way to read the status of Infrared Module.

```c
#include <MoonBot.h>

// connect ir sensor 1 on port 1
uint8_t ir1[2] = {
    moonbotPortToPin(kPort1, kPortPin1),
    moonbotPortToPin(kPort1, kPortPin2)
};

// connect ir sensor 1 on port 1
uint8_t ir2[2] = {
    moonbotPortToPin(kPort2, kPortPin1),
    moonbotPortToPin(kPort2, kPortPin2)
};

void setup()
{
    // initialize ir sensor 1/2 as INPUT_PULLUP
    pinMode(ir1[0], INPUT);
    pinMode(ir1[1], INPUT);
    pinMode(ir2[0], INPUT);
    pinMode(ir2[1], INPUT);
}

void loop()
{
    Serial.println("" ===============");
    Serial.print(" ir sensor1: ");
    // read ir sensor 1 state
    if (!digitalRead(ir1[0]) || !digitalRead(ir1[1])) {
        Serial.println("triggered");
    } else {
        Serial.println("not triggered");
    }
    Serial.print(" ir sensor2: ");
    // read ir sensor 2 state
    if (!digitalRead(ir2[0]) || !digitalRead(ir2[1])) {
        Serial.println("triggered");
    } else {
        Serial.println("not triggered");
    }
}
```

**Attention:** Infrared module is in low level when touched, and pin mode is LOW; And it is not touched when level is HIGH.
API Reference - Pin Map

Header File

- src/pins_moonbot.h

Enum

defined enum moonbot_servo_t

value: MoonBot Kit servo port

- kServo1 = 0
- kServo2
- kServo3
- kServo4
- kServoNum

- servo port number

defined enum servo_pin_t

value: servo port type

- kSignal
  - servo signal pin
- kShutDown
  - servo power pin
- kState
  - servo status pin

defined enum moonbot_motor_t

value: MoonBot Kit motor port

- kMotor1 = 0
- kMotor2
- kMotorNum

- motor port number

defined enum motor_pin_t

value: motor port type

- kDirection
  - motor direction pin
kSpeed
  • motor speed pin

enum moonbot_port_t
  • MoonBot Kit GPIO port

    value:
    kPort1=0
    kPort2
    kPort3
    kPort4
    kPort5
    kPort6
    kPort7
    kPort8
    kPort9
    kPortNum
    • GPIO port number

enum port_pin_t
  • GPIO port type

    value:
    kPortPin1=0
    kPortPin2
    kPortPinNum
    • port pin number

Macro Definition

MOONBOT_PIN_LED
  • MoonBot Kit controller on-board LED pin

MOONBOT_PIN_BUZZER_SIG
  • MoonBot Kit controller buzzer signal pin

MOONBOT_PIN_BUZZER_SHDW
  • MoonBot Kit controller buzzer power pin

MOONBOT_PIN_BUTTON_A
  • MoonBot Kit controller button A pin

MOONBOT_PIN_BUTTON_B
  • MoonBot Kit controller button B pin
Functions

```c
uint8_t moonbotPortToPin(moonbot_port_t port_num, port_pin_t pin_num);
```

- Get MoonBot Kit controller Arduino pin number of the GPIO port.
  
  ```c
  value
  ```
  - port_num: GPIO port
  - pin_num: port pin number

  ```c
  return
  ```
  - Arduino pin number of the port pin

```c
uint8_t moonbotMotorToPin(moonbot_motor_t motor_num, motor_pin_t pin_type);
```

- Get MoonBot Kit controller Arduino pin number of the motor port.
  
  ```c
  value
  ```
  - motor_num: motor port number
  - pin_type: motor pin type

  ```c
  return
  ```
  - motor port Arduino pin

```c
uint8_t moonbotServoToPin(moonbot_servo_t servo_num, servo_pin_t pin_type);
```

- Get MoonBot Kit controller Arduino pin number of the servo port.
  
  ```c
  value
  ```
  - servo_num: servo port number
  - pin_type: servo pin type

  ```c
  return
  ```
  - servo port Arduino pin

### 13.2.2 Motor

#### Overview

MoonBot Kit Motor Module includes motor and encoder. In Arduino library we provide motor library to drive single motor, and chassis control library to drive dual motors.

We can include `MoonBot.h` header file to call `TankBase` to drive motor chassis, or call `Motor1 Motor2` to control single motor.
Chassis control

How to make the chassis move around? Check this simple example:

```c
#include <MoonBot.h>

void setup()
{
  TankBase.begin();  // enable TankBase, use default setting
}

void loop()
{
  // forward 1s
  TankBase.write(100, 100);
  delay(1000);
  // backward 1s
  TankBase.write(-100, -100);
  delay(1000);
  // turn right 1s
  TankBase.write(100, -100);
  delay(1000);
  // turn left 1s
  TankBase.write(-100, 100);
  delay(1000);
}
```

Through `TankBase.write()` function, write motor speed of left and right motor can make the chassis move.

If you want to control the motor speed accurately, use `TankBase.writeRpm()` function instead.

```c
// Chassis turn left, with left motor speed 30 RPM, and right -30 RPM.
TankBase.writeRpm(30, -30);
```

**Note:**

Use functions that can control accurate speed of the motor like `TankBase.write()` `TankBase.writeDistance()` `TankBase.writeAngle()`, you need to connect the encoder to its port, and initialize function `TankBase.begin()` and change the parameter `enc_enable` to `true` (default is `true`).

You can also make motor move certain angle or distance.

```c
void loop()
{
  TankBase.writeDistance(30, 20);  // Chassis move forward for 20 cm with the...
  while(TankBase.read(kLeftMotor) || TankBase.read(kRightMotor));  // Wait for...
  // the chassis to stop
  delay(100);
  TankBase.writeAngle(30, 180);  // Chassis turn right for 180 degrees with...
  while(TankBase.read(kLeftMotor) || TankBase.read(kRightMotor));  // Wait for...
  // the chassis to stop
  delay(100);
}
```

**Note:** Because of fix deviations or friction force, you will find chassis cannot go straight. You can use following calibration...
functions to correct it.

```c
void setup() {
    TankBase.rpmCorrection(82); // Calibrate speed offset %
    TankBase.distanceCorrection(120); // Calibrate distance offset %
    TankBase.wheelSpacingSet(100); // Calibrate angle offset %
}
```

Control Single Motor

If you only want to control single motor, call `Motor1 Motor2` to achieve.

```c
Motor1.write(100); // Set motor1 analog value to 100
Motor2.write(100); // Set motor2 analog value to 100
Motor1.writeRpm(30); // Set motor1 speed to 30 RPM
Motor2.writeRpm(30); // Set motor2 speed to 30 RPM
```

API Reference - Motor

Header File

- `src/MoonBot_Motor.h`

Enum Type

```c
define moonbot_motor_t
    • motor port type
    value:
      kMotor1=0
      kMotor2
      kMotorNum
```

Class

```c
class Motor
    • Single motor driver
    member function
      Motor(moonbot_motor_t motor_type);
      • constructed function, define port type
      parameter
        • motor_type
        int begin(const bool reverse_dir = false, const bool enc_enable = true);
```
• Initialize motor in the given port
  
  **parameter**
  
  • `reverse_dir` : Reverse motor rotate direction, `false` by default
  • `enc_enable` : Enable encoder, `true` by default

  **return**
  
  • `0` : Initialization succeed
  • `-1` : Can not find the motor port

  ```cpp
  void write(int vol);
  ```

  • Write analog value
  
  **parameter**
  
  • `vol` : Value of the voltage, with range of `±255`. `>0` means CW, `<0` means CCW

  ```cpp
  int read(void);
  ```

  • Read the analog value
  
  **return**
  
  • `±255` : Analog value

  ```cpp
  void writeStep(uint32_t step, int rpm = 30);
  ```

  • Drive motor with certain speed and steps and then stop.
  
  • This function will use encoder, and encoder must be opened in `begin()` function.
  
  **parameter**
  
  • `step` : Rotation step, 240 steps per cycle
  • `rpm` : Motor rotation speed, 30 RPM per cycle

  ```cpp
  void writeRpm(int rpm = 30);
  ```

  • Write motor speed, unit: RPM
  
  • This function will use encoder, and encoder must be opened in `begin()` function.
  
  **parameter**
  
  • `rpm` : Motor rotation speed, 30 RPM by default

  ```cpp
  int readRpm(void);
  ```

  • Read motor speed, unit: RPM
  
  • This function will use encoder, and encoder must be opened in `begin()` function.
  
  **return**
  
  • Motor rotate speed

  ```cpp
  void writeDistance(int rpm, uint32_t distance_cm);
  ```

  • Drive motor with certain speed and distance and then stop. Because of the offsets,
  please use "distanceCorrection()" function to calibrate.
  
  • This function will use encoder, and encoder must be opened in `begin()` function.
  
  **parameter**
• rpm: Motor rotation speed
• distance_cm: Moving distance, unit: cm

`uint32_t readEncoderPulse(void);`
• Read encoder value
• This function will use encoder, and encoder must be opened in `begin()` function.
  return
  • Current encoder value

`void rpmCorrection(uint8_t percent);`
• Motors RPM calibration parameter
  • percent: Calibration percentage. >100 means increase, <100 means decrease

`void distanceCorrection(uint8_t percent);`
• Motors distance calibration parameter
  • percent: Calibration percentage. >100 means increase distance, <100 means decrease

API Reference - Chassis Control

Header File

• `src/MoonBot_TankBase.h`

Enum Type

define motor_type_t
  • motor type
    value:
      kLeftMotor=0
        • left motor
      kRightMotor
        • right motor
Class

class MoonBotTankBase

• Chassis dual motor driver

  member function

  MoonBotTankBase(Motor& left_motor, Motor& right_motor);

  • constructed function, define two motors to the port

  parameter

  • left_motor
  • right_motor

  int begin(const bool reverse_dir = false, const bool enc_enable = true);

  • Initialize chassis motors

  parameter

  • reverse_dir: Reverse rotation direction, false by default
  • enc_enable: Enable encoder, true by default

  return

  • 0: Initialization succeed
  • -1: Can not find the motor port

  int begin(const bool left_reverse_dir, const bool right_reverse_dir, const bool enc_enable);

  • Initialize chassis motors

  parameter

  • left_reverse_dir: Reverse left motor rotation direction
  • right_reverse_dir: Reverse right motor rotation direction
  • enc_enable: Enable encoder

  return

  • 0: Initialization succeed
  • -1: Can not find the motor port

  void write(int left_vol, int right_vol);

  • Write analog value to motors

  parameter

  • left_vol: Left motor voltage, range of ±255, >0 means CW, <0 means CCW
  • right_vol: Right motor voltage, range of ±255, >0 means CW, <0 means CCW

  int read(motor_type_t motor_type);

  • Read analog value from motors
parameter
  • motor_type

return
  • ±255: Analog value

uint32_t readEncoderPulse(motor_type_t motor_type);
  • Read motor encoder value
  • This function call encoder, and should be opened after begin() function

parameter
  • motor_type

return
  • Current encoder value

void writeRpm(int left_rpm, int right_rpm);
  • Write motors speed, unit: RPM
  • This function will use encoder, and encoder must be opened in begin() function.

parameter
  • left_rpm: Left motor rotation speed
  • right_rpm: Right motor rotation speed

int readRpm(motor_type_t motor_type);
  • Read motors speed, unit: RPM
  • This function will use encoder, and encoder must be opened in begin() function.

parameter
  • motor_type

return
  • Motor rotation speed

void writeDistance(int rpm, uint32_t distance_cm);
  • Drive motors with certain distance and stop. Because of the offsets, please use rpmCorrection() and distanceCorrection() function to calibrate.
  • This function will use encoder, and encoder must be opened in begin() function.

parameter
  • rpm: Motor rotation speed
  • distance_cm: Moving distance, unit: cm

void writeAngle(int rpm, uint32_t angle);
• Drive motors with certain distance and stop. Because of the offsets, please use `rpmCorrection` and `wheelSpacingSet()` function to calibrate.

• This function will use encoder, and encoder must be opened in `begin()` function.

  parameter
  • `rpm`: Motor rotation speed
  • `angle`: Rotation angle, unit: degree(°)

`void wheelSpacingSet(int correct, float space_cm = 0);`

• Set wheel spacing and calibrate turn angle. This function can calibrate that wheel can not turn the accurate angle

  parameter
  • `correct`: Correction value, >100 means angle increase, <100 means decrease
  • `space_cm`: Motor spacing

`void rpmCorrection(int percent);`

• Calibrate the speed of left and right motor

  parameter
  • `percent`: Calibration value, >100 means calibrate to right, <100 means calibrate to left

`void distanceCorrection(int percent);`

• Calibrate moving distance

  parameter
  • `percent`: Calibration value, >100 means increase distance, <100 means decrease

`void forward(unsigned int step, unsigned int rpm = 30);`

• Chassis move forward for certain distance and stop

  parameter
  • `step`: Forward distance, unit: cm
  • `rpm`: Motor rotate speed, 30RPM by default

`void backward(unsigned int step, unsigned int rpm = 30);`

• Chassis move backward for certain distance and stop

  parameter
  • `step`: Back distance, unit: cm
  • `rpm`: Motor rotate speed, 30RPM by default

`void turnLeft(unsigned int step, unsigned int rpm = 30);`

• Chassis turns left for certain degrees and stop

  parameter
• step: Left turn angle, unit: degree(°)
• rpm: Motor rotate speed, 30 RPM by default

void turnRight(unsigned int step, unsigned int rpm = 30);

• Chassis turns right for certain degrees and stop

parameter
• step: Right turn angle, unit: degree(°)
• rpm: Motor rotate speed, 30 RPM by default

void stop(void);

• Chassis stop moving

13.2.3 Music

Overview

MoonBot Kit provide two sound devices, buzzer on Controller Module and Speaker Module. We can use Arduino basic functions tone() and noTone() to control the buzzer. Use Speaker library to control the speaker.

By including MoonBot.h header file in program, we can call speaker driver to drive the speaker module.

On-board Buzzer Driver

We can use macro definition MOONBOT_PIN_BUZZER_SIG to get Arduino pin of the buzzer, and control the voltage of MOONBOT_PIN_BUZZER_SHDW to open or close the buzzer.

```c
#include <MoonBot.h>

void setup()
{
  pinMode(MOONBOT_PIN_BUZZER_SIG, OUTPUT);  // Initialize buzzer signal pin
  pinMode(MOONBOT_PIN_BUZZER_SHDW, OUTPUT);  // Initialize buzzer power pin to
  digitalWrite(MOONBOT_PIN_BUZZER_SHDW, LOW);  // Pull down buzzer power pin to
  tone(MOONBOT_PIN_BUZZER_SIG, 1000, 2000);  // Let buzzer play with 1000Hz
  for 2000ms
}
```

**Attention:** In 7th line of the example:

```c
digitalWrite(MOONBOT_PIN_BUZZER_SHDW, LOW);  // Initialize buzzer power pin to
```

Pull MOONBOT_PIN_BUZZER_SHDW voltage LOW to open it, and HIGH to close. The default voltage is LOW.
Morpx Documentation

Speaker Module Driver

MoonBot Kit Speaker Module use a WT2003S MP3 decoder chip. Call speaker library to control it as a MP3 player. There is a simple MP3 player tutorial:

```c
#include <MoonBot.h>

void setup()
{
    speaker.begin(Serial2); // Initialize speaker module to Arduino serial port
    speaker.setVolume(20); // Set volume to 20. Max volume is 32
}

void loop()
{
    if (!digitalRead(MOONBOT_PIN_BUTTON_A)) {
        speaker.playNext(); // Play the next music
    } else if (!digitalRead(MOONBOT_PIN_BUTTON_B)) {
        speaker.playPrevious(); // Play the last music
    }
}
```

Check Official terminal MP3 player examples for more detailed information.

API Reference - Speaker

Header File

- src/MoonBot_WT2003S_MP3_Decoder.h

Class

class WT2003S

- WT2003S MP3 player driver

  Member function

  ```
  void begin(SoftwareSerial &serialPort);
  ```

  Use software serial port to initialize speaker parameter

  ```
  void begin(HardwareSerial &serialPort = Serial);
  ```

  Use hardware serial port to initialize speaker parameter

  ```
  serialPort: hardware serial port, Serial by default
  ```
```c
uint8_t play(char* fileName);
• play music with the file name
  parameter
  • fileName: 4 bytes of the file name
  return
  • 0 means the command is right, other return means wrong

uint8_t setVolume(uint8_t volumeLevel);
• Set volume of the speaker
  parameter
  • volumeLevel: volume level, with range of 0~32
  return
  • 0 means the command is right, other return means wrong

uint8_t stop(void);
• Stop playing current music
  return
  • 0 means the command is right, other return means wrong

void pause(void);
• Pause when playing, play when pausing

uint8_t playPrevious(void);
• Play the last music. Play the final music when on the first
  return
  • 0 means the command is right, other return means wrong

uint8_t playNext(void);
• Play the next music. Play the first music when on the final
  return
  • 0 means the command is right, other return means wrong

uint8_t setPlayMode(uint8_t mode);
• Set play mode
  parameter
  • mode:  
    |   |      |
    |---|------|
    | 0 | single play |
    | 1 | single cycle |
    | 2 | list loop |
    | 3 | random play |

  return
  • 0 means the command is right, other return means wrong
```
uint16_t getSongCount(void);

• Get the music order number in the list

    return

• current music order number in the list

void getSongName();

• Get first 9 bytes of the song name. Read
  WT2003S::songName[MP3_NUM_NAME_BYTES] to get the
  name

uint8_t playTrackNumber(uint8_t trackNumber);

• Play music with the given order number
  parameter

    • trackNumber: music order number in the list

    return

• 0 means the command is right, other return means wrong

uint8_t getVolume(void);

• Get current volume level of the speaker

    return

• 0~32: volume level of the speaker

uint8_t getPlayStatus(void);

• Get the current play status

    return

     1 play
     2 stop
     3 pause

13.2.4 IMU

Introduction

MoonBot Kit Controller Module integrates three functions of triaxial magnetometer, triaxial acceleration and temperature sensor into IMU module. In the Arduino library, we also provide ref:IMU < api-ref-imu > library to facilitate users to access the master control of the current attitude, direction, temperature and other states.

By calling `IMU`, we can quickly obtain the current compass angle, pitch angle, roll angle, gravity acceleration, temperature and other state values.
Read master control current direction

By reading the angle of the compass, we can know the direction of the current master control:

```c
#include <MoonBot.h>

void setup()
{
    IMU.enable();  // IMU Enable
    IMU.calibrateMag();  // IMU magnetometer calibration, the master control needs to...
                           // flip in the shape of "∞"
}

void loop()
{
    Serial.print("compass:");
    // Obtain the compass angle(0~360°). When pointing north, the value is 0 or 360
    Serial.println(IMU.getMagAngle());
}
```

**Note:**

When the main control is flat, the return value is Y axis (see the master control front silk mark) and the northward clip.

When the main control is erected, the return value is the angle between Z axis and North direction.

---

Obtain Pitch angle or Rolling angle

```c
// Obtain Pitch angle±180° When the main control is up, the angle is positive and...
// the downward angle is negative.
int pitch = IMU.getAccAngle(kAccPitch);
// Obtain Rolling angle±180° The main control right deviation is positive and the...
// left deviation is negative.
int roll = IMU.getAccAngle(kAccRoll);
```

**Note:** MoonBot Kit The main control direction is Y axis (see the main control front silk mark) Angles are calculated on this premise.

---

Acquisition of current acceleration

```c
// Acquisition of acceleration unit g The value at rest is 1.0.
float acceleration = IMU.getAcceleration();
```
Obtain the current motion state

```c
void loop()
{
    if (IMU.on(kIMUShake)) {
        // If the current master is shaking
        // bright red LED
        LED.setPixelColor(0, 0xff0000);
        LED.setPixelColor(1, 0xff0000);
        LED.show();
    } else if (IMU.on(kIMUFreeFall)) {
        // If the current master is in free fall
        // bright green LED
        LED.setPixelColor(0, 0x00ff00);
        LED.setPixelColor(1, 0x00ff00);
        LED.show();
    } else {
        // If the main control is stationary
        // close LED
        LED.setPixelColor(0, 0x000000);
        LED.setPixelColor(1, 0x000000);
        LED.show();
    }
}
```

API Reference - IMU

Header file

- `src/LSM303AGR_IMU_Sensor.h`

enumeration

```c
enum lsm303_axes_t
{
    // IMU Directional axis type
    value:
    kDirX
    kDirY
    kDirZ
}
enum lsm303_acc_angle_t
{
    // IMU Attitude Angle type
    value:
    kAccRoll
    kAccPitch
}
enum imu_state_t
{
    // IMU Special state type.
    value:
```
value:

**kIMUShake**
- IMU Is it in a sloshing state

**kIMUFreeFall**
- IMU Is it in a free falling state

**Class**

class LSM303AGR_IMU_Sensor
- IMU Drive.

**group function**

```c
int enable(void);
```
- enable IMU

**Return**
- 0 enable success, unable failure

```c
int advGetMagAngle(lsm303_axes_t main_axes, lsm303_axes_t sub_axes);
```
- Get the plane where the specified spindle and vice-spindle are located, and the angle between the spindle and the North side.

**parameters**
- main_axes Spindle
- sub_axes Countershaft

**Return**
- Angle between the spindle and the North

```c
int getMagAngle(void);
```
- Obtain the compass angle When the main control is placed horizontally, the angle between the spindle and the North side is returned.
- when the main control is placed vertically, the angle between the positive direction of Z axis and the north is returned.

**Return**
- Angle between the spindle and the North

```c
int getAccAngle(lsm303_acc_angle_t angle_type);
```
- Obtain the main control angle.

**parameters**
- angle_type angle type

**Return**
- angle

```c
float getAcceleration(void);
```
- Acquisition of acceleration value

**Return**
• Acceleration value unit g

bool on(imu_state_t imu_state);
  • Get whether the master control is in some state
    parameters
    • imu_state IMU state
  Return
    • true IMU In this state, Otherwise, it is not in this state.

bool calibrateMag(void);
  • Calibration of Magnetometer
  Return
    • Whether the calibration is completed or not

int16_t temperature(void);
  • Obtain the original temperature value
  Return
    • Primitive value of temperature

float temperatureC(void);
  • Get the current temperature unit Celsius degree
  Return
    • Current temperature unit Celsius degree

float temperatureF(void);
  • Current temperature unit Fahrenheit degree
  Return
    • Current temperature unit Fahrenheit degree

group variable

  LSM303AGR_ACC_Sensor Acc;
  • Acceleration drive

  LSM303AGR_MAG_Sensor Mag;
  • Magnetometer drive

13.2.5 Servo

Overview

MoonBot Kit Controller Module can be connected up to four Servo Module. In Arduino library, we provide Servo library. Through this library, you can control one or more servos to move.

Servo library inherit Arduino basic servo driver class Servo. Except for basic Servo class function, we also provide functions like servo calibration, several servos move together. In MoonBot.h header file, we provide four variables m_servo[kServo1] m_servo[kServo2] m_servo[kServo3] m_servo[kServo4] to drive corresponding servo ports in controller module.
Basic Application

There is a basic application of servos.

```c
#include <MoonBot.h>

int pos;

void setup() {
    m_servo[kServo1].attach(kServo1, true); // attaches servo on servo port
    // 1, and reverse directions
}

void loop() {
    for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
        m_servo[kServo1].write(pos); // tell servo to go to position
        // in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the
        // position
    }
    for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
        m_servo[kServo1].write(pos); // tell servo to go to position
        // in variable 'pos'
        delay(15); // waits 15ms for the servo to reach the
        // position
    }
}
```

Note: Initial function of servos is changed to `attach(moonbot_servo_t servo_port, bool reverse)`, and original function `uint8_t attach(int pin)` is not supported anymore.

Servos move together

We provide `void MoonBotServo::setTargetAngle()` and `MoonBotServo::moveAllServoToTarget()` functions to make servos move together.

```c
#include <MoonBot.h>

void setup() {
    for (int i = 0; i < kServoNum; ++i) {
        m_servo[i].attach((moonbot_servo_t)i); // attaches servo
    }
}

void loop() {
    // in steps of 1 degree
    for (int i = 0; i < kServoNum; ++i) {
        m_servo[i].setTargetAngle(180, 1); // set all servo to go to position in
        // variable '180', speed 1 degree per pulse(20ms)
    }
    MoonBotServo::moveAllServoToTarget(); // move all servo to target angle
    for (int i = 0; i < kServoNum; ++i) {
        m_servo[i].setTargetAngle(0, 1); // set all servo to go to position in
        // variable '0', speed 1 degree per pulse(20ms)
    }
}
```

(continues on next page)
MoonBotServo::moveAllServoToTarget(); // move all servo to target angle

Note: When using MoonBotServo::moveAllServoToTarget(); default parameter, the function will wait for all servos finish moving and stopping. When parameter is not 0, it will stop when time is over, and feed back whether moving is finished.

Function `bool isMoving(void);` can be used every certain time to check the status.

```cpp
while (!MoonBotServo::moveAllServoToTarget(0)) {
    // Check whether servos are moving.
    for (int i = 0; i < kServoNum; ++i) {
        if (!m_servo[i].isMoving()) {
            // when servos stop, print the status.
            Serial.print("Servo");
            Serial.print(i);
            Serial.println(" Stopped.");
        }
    }
}
Serial.println("All Servo Stopped.");
```

When using COM monitor, information will be received as below.

```
Servo1 Stopped.
...
Servo1 Stopped.
Servo2 Stopped.
...
Servo2 Stopped.
Servo3 Stopped.
...
Servo3 Stopped.
All Servo Stopped.
```

Servo Calibration

MoonBot Kit Servo library provide servo calibration function that can correct the offset of servos.

```cpp
m_servo[kServo1].correction(-2); // Calibrate servo 1 downwards for 2°
```

API Reference - Servo

Header File

- src/MoonBot_Servo.h
Enum

enum moonbot_servo_t
  • servo port type

  value:
    kServo1
    kServo2
    kServo3
    kServo4
    kServoNum
      • servo port number

Class

class MoonBotServo
  • MoonBot Kit servo driver library

Member function

  uint8_t attach(moonbot_servo_t servo_port, bool reverse = MOONBOT_SERVO_REVERSE);

  • Initialise servo to servo ports.

  Parameter
    • servo_port
    • reverse

  Return
    • NOT_A_PORT Servo port is invalid, and other initialization is right.

  uint8_t attach(moonbot_servo_t servo_port, int min, int max, bool reverse = MOONBOT_SERVO_REVERSE);

  • Initialise servo to servo ports, and set its moving range.

  Parameter
    • servo_port • servo port
    • min • minimum degree of servo
    • max • max degree of servo
    • reverse • reverse servo direction

  Return
    • NOT_A_PORT Servo port is invalid, and other initialization is right.

  void detach(void);
  • Detach servo and port
void write(int value);
• Write servo angle
  parameter
  • value: angle value range 0~180°

int read(void);
• Read current servo degree
  Return
  • current degree

void reverse(bool state);
• Reverse servo direction
  parameter
  • state: Status true Direction is reversed

void setTargetAngle(int angle, unsigned int speed = 1);
• Initialise servos. It should be used together with “static bool moveAllServoToTarget()”.
  parameter
  • angle: Initialised angle
  • speed: degree of every pulse

void stop(void);
• stop servos

void power(bool state);
• open or close servo power
  parameter
  • state: Status of servo power, true means open

void correction(int angle_offset);
• Servo calibration
  parameter
  • angle_offset: Calibrate the angle. Range: ±90°

bool isMoving(void);
• Read moving status
  Return
  • true Servo is moving

bool isPowerOverload(void);
• Detect whether current is overload
  Return
  • true Power is overload
Static member function

    static bool moveAllServoToTarget(unsigned long timeToWait_ms = 0xFFFFFFFF);

- Move all servo to set angle

  Parameter

  - timeToWait_ms: Default time is infinite, until servo move to target angle.

  Return

  - true Finish all movement.

    static void stopAllServo(void);

- Stop all servo movements.

13.2.6 Light

Overview

MoonBot Kit contains two sets of light modules. Respectively located Controller Module and Eyes Module. We can use Adafruit NeoPixel library to drive these two sets of light modules.

We can drive two on-board LED lights by calling `LED'. Drive 12 LED eyes by calling `moonbot_eyes'. At the same time, through the call: ref: `LED eye movement < api-ref-led-action >` let the eyes make a rich expression.

LED Foundation driven

```cpp
#include <MoonBot.h>

void setup() {
  // enable main control LED
  LED.begin();
  moonbot_eyes.begin();
  // clear LED color
  LED.clear();
  LED.show();
  moonbot_eyes.clear();
  moonbot_eyes.show();
}

void loop() {
  if (digitalRead(MOONBOT_PIN_BUTTON_A) == LOW
      && digitalRead(MOONBOT_PIN_BUTTON_B) == LOW) {
    // If A&B is pressed at the same time
    // LED and eye lights display cyan
    LED.fill(0x003030);
    LED.show();
    moonbot_eyes.fill(0x003030);
    moonbot_eyes.show();
  } else if (digitalRead(MOONBOT_PIN_BUTTON_A) == LOW) {
    // If key A is pressed
    // LED0® The right eye display green.
```
LED.setPixelColor(0, 0x003000);
LED.setPixelColor(1, 0);
LED.show();
moonbot_eyes.clear();
moonbot_eyes.fill(0x003000, 0, 6);
moonbot_eyes.show();
}
else if (digitalRead(MOONBOT_PIN_BUTTON_B) == LOW) {
    // If key B is pressed
    // LED1 The left eye display green.
    LED.setPixelColor(0, 0);
    LED.setPixelColor(1, 0x000030);
    LED.show();
    moonbot_eyes.clear();
    moonbot_eyes.fill(0x000030, 6, 6);
    moonbot_eyes.show();
} else {
    // LED Eye lights off
    LED.clear();
    LED.show();
    moonbot_eyes.clear();
    moonbot_eyes.show();
}

LED Eye Lighting Action

MoonBot Kit provides abundant Eye action to be used:

    colorWipe(moonbot_eyes, 0x40, 200);  // LEDs turn green one by one
    theaterChase(moonbot_eyes, 0xFF00, 50);  // Eyes turn around
    MoonBotEyesCircle(moonbot_eyes, 50);  // Eyes turn around gradually
    rainbow(moonbot_eyes, 5);  // Eyes show rainbow color
    rainbowCycle(moonbot_eyes, 5);  // Eyes show rainbow color one by one

API Reference - Adafruit_NeoPixel

Check Adafruit official website for detailed information:  https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library-use

API Reference - LED Eyes Action

Header File

- src/MoonBot_Eyes.h
Enum

enum moonbot_eyes_t
  • eyes type
    value:
    kEyesLeft
      • left eye
    kEyesRight
      • right eye
    kEyesBoth
      • both eyes

enum moonbot_look_t
  • direction that eyes look at
    value:
    kEyesLookUp
      • Eyes look up
    kEyesLookDown
      • Eyes look down
    kEyesLookLeft
      • Eyes look left
    kEyesLookRight
      • Eyes look right

enum moonbot_eyes_scroll_t
  • eyes scroll type
    value:
    kEyesScrollUp
      • eyes scroll up
    kEyesScrollDown
      • eyes scroll down
    kEyesScrollLeft
      • eyes scroll left
    kEyesScrollRight
      • eyes scroll right
Functions

void colorFade(Adafruit_NeoPixel& led, uint8_t r, uint8_t g, uint8_t b, uint8_t wait);

• LED eyes turn to target color gradually

parameter

• led: LED type
• r: value of red channel
• g: value of green channel
• b: value of blue channel
• wait: time to wait

void colorWipe(Adafruit_NeoPixel& led, uint32_t c, uint8_t wait);

• LED change color one by one

parameter

• led: LED type
• c: REG color of the LED
• wait: time to wait for action

void rainbow(Adafruit_NeoPixel& led, uint8_t wait);

• LED light shine rainbow color

parameter

• led: LED type
• wait: time to wait

void rainbowCycle(Adafruit_NeoPixel& led, uint8_t wait)

• LED change light of rainbow color gradually

parameter

• led: LED type
• wait: time to wait for changing

void theaterChase(Adafruit_NeoPixel& led, uint32_t c, uint8_t wait);

• LED turn around with target color

parameter

• led: LED type
• c: RGB color of LED light
• wait: time to wait for action

void MoonBotEyesLook(Adafruit_NeoPixel& led, moonbot_look_t look_type, uint32_t color);

• LED eyes look to the direction

parameter

• led: LED type
• look_type: the direction that eyes looks to
void MoonBotEyesScroll(Adafruit_NeoPixel& led, moonbot_eyes_scroll_t scroll_tpye, uint32_t color, uint8_t wait = 50);

- LED eyes scroll to the direction
  
  parameter
  
  - led: LED type
  - scroll_tpye: direction that eyes scroll to
  - color: LED color type
  - wait: time to wait, 50ms by default

void MoonBotEyesCircle(Adafruit_NeoPixel& led, uint32_t color, moonbot_eyes_t eyes_type = kEyesBoth, uint8_t wait = 50);

- LED eyes turn around gradually
  
  parameter
  
  - led: LED type
  - color: eyes color
  - eyes_type: eyes type
  - wait: circle time, 50ms by default
MOONBOT KIT EXTENDED STRUCTURES

Except for official structures, makers can use MoonBot Kit hardware modules and other common materials to build creative robots.

14.1 Lego Car

Hardware modules of MoonBot Kit can be connected to lego parts and used to build lego robots.

Basic connection: There are two main types of lego parts: technic series and traditional series. Horizontally placed modules are compatible with traditional series, while vertically placed modules are compatible with technic series. Traditional parts need to be transferred to a 2X2 lego block, and technic parts need black bolts to connect to others, as shown below.

Demount the screw of active wheel, remove the active wheel and mount an adapter of TT motor to Lego. Then MoonRover can use Lego wheels to become a traditional two wheel car. And a universal wheel is needed to make the car horizontal, as is shown below. A big wheel will make the car drive twice faster than before.
14.2 Biped Robot

MoonBot controller can be connected to 4 servos, and made a biped dancing robot with some sheet metal parts.
Shell of MoonBot can be modified by yourself. Just use 3d printing, paper and wood boards to change shell or inside parts of MoonBot. For example, a cool dragon from Teacher Ma is shown below.
MOONBOT KIT FIRMWARE UPGRADE GUIDE

This doc will guide users to upgrade firmware of MoonBot Kit Controller Module and Vision Module.

15.1 Preparation

Hardware:

- MoonBot Kit
- PC (Windows)

Software:

- Arduino IDE
- MoonBot Arduino Library
- MU Vision Sensor 3 upgrade software
- MU (for MoonBot) latest firmware (.bin file)

15.2 Upgrading Steps

15.2.1 Step One: Upgrade firmware of MoonBot Kit Controller

Refer to MoonBot Kit APP Firmware Upgrade Guide

15.2.2 Step Two: Upgrade firmware of MU Vision Sensor 3

1. Connect Vision Module to MoonBot Kit Controller Module port P9, and connect the controller to PC.
2. Press and hold function button of Vision Module on the left, and then press Reset button once. Then release function button, and the vision sensor is in upgrading mode now.
3. Open MU Vision Sensor 3 upgrade software flash_download_tools_vx.x.x.exe
4. Choose ESP32 DownloadTool
5. Change Settings
15.2. Upgrading Steps
Morpex Documentation

Note:

• SPI SPEED: 40MHz
• SPI MODE: DIO
• FLASH SIZE: 32Mbit
• BAUD: 115200
• COM port: connect to the right COM port, which can be found in Windows device manager.

6. Choose ... button and choose firmware file, and choose √ to activate the file.

7. Input address of the firmware behind @, which is 0x10000.

Attention: Do not forget to input the address or modify it. Otherwise you will damage the firmware order of the vision sensor. If it happens, please contact to Morpx support to solve it.

Phone number: (0571)8195 8588
E-mail: support@morpx.com

8. Press START button on the left-bottom corner, and click continuously button B of MoonBot Kit Controller Module until the software start burning firmware. LED on the right side of the controller turns green.

9. When the software progress bar is full, and shows FINISH, firmware downloading is complete.
MOONBOT KIT RESOURCE

16.1 Technical Information

Thanks for purchasing MoonBot Kit, and we would like to provide continuous updating service, please check to our website: www.morpx.com regularly. Updates are subject to change without notice. You can get the latest technical information from the following websites:

Official Website: http://mai.morpx.com/page.php?a=moonbot-kit
GitHub: https://github.com/mu-opensource/

16.2 3D Assembly Models

MoonBot Kit 3D assembly files can be downloaded here:

MoonBot Kit 3D assembly

STP file is a universal 3D file format, which can be opened by popular CAD software like solidworks and CREO. The models can be used to watch the details of each MoonBot Kit structure, measure dimension, render in Keyshot and so on.

Plastic and sheet metal parts in models are optimized for manufacturing, and are not recommended to be used for FDM 3D printing.
16.3 Projects

There are extended projects of MoonBot Kit. Watch the latest updates at https://www.hackster.io/tianli.

16.4 Platform Links

MoonBot Kit is compatible with Arduino opensource platform. Check related website to learn basic knowledge.

Mixly

Mixly Official Website: http://mixly.org/

Arduino

Arduino Official Website: https://www.arduino.cc/
CHAPTER
SEVENTEEN

MU SELF-DRIVING KIT INTRODUCTION

MU Micro:bit

Micro:bit

MU
CHAPTER EIGHTEEN

MU SELF-DRIVING KIT STRUCTURE

18.1

18.2
CHAPTER NINETEEN

MU SELF-DRIVING KIT MAKECODE TUTORIAL

MakeCode

19.1

19.1.1

19.1.2
Download to your micro:bit

1. Connect the micro:bit with a USB cable.
2. Use the microUSB port on the top of the micro:bit.
3. Drag the downloaded file to the MICROBIT drive.

---

**micro:bit Flashing Heart**

Hex: [micro:bit Flashing Heart.hex](data:microbit-flashing-heart.hex;base64,OjAyMDAw)

---

**MakeCode Tutorial**
19.1.3

Want to create your own extension? Login to GitHub.
19.2 Tips

19.2.1 Motor

222 Chapter 19. MU Self-driving Kit MakeCode Tutorial
19.2.2 

```plaintext
[servo] PWM 0.10-180
```

19.2.3 

```plaintext
LED
```

Chapter 19. MU Self-driving Kit MakeCode Tutorial
micro:bit

```plaintext
无限循环
LED灯 左侧 ▼ 打开 ▼
暂停 (ms) 1000 ▼
LED灯 左侧 ▼ 关闭 ▼
LED灯 右侧 ▼ 打开 ▼
LED灯 右侧 ▼ 关闭 ▼

当开机时
将 myImage ▼ 设置为
创建大图像

无限循环
滚动图像 myImage ▼ 偏移为 1 且间隔 (ms) 为 100
```
19.2.4

当启用时，将 strip 设为 NeoPixel at pin P15 with 4 leds as RGB (GR8 format)。

无限循环

对于从 0 至 3 的 index 执行：
- strip set pixel color at index to red 255 green 0 blue 0
- strip show
- 延时 (ms) 100

19.2.4

micro:bit P0
19.3 ??

?кро?sensor?

19.3.1 ??

??micro?????
19.3.2

-micro:bit (E)(W)(S)(N)
19.3.3

IMU 当前方向判断与显示

19.3.
19.3.4 久了

```plaintext
100cm ≤ distance < 20cm
```

```plaintext
20cm ≤ distance ≤ 90cm
```
19.4.1 Serial Communication

19.4.2 micro:bit

micro:bit

micro:bit
19.4.3

19.5
19.5.1 基本

repeat
while
for

19.5.2 循环

repeat
while
for
for of

19.5.3 逻辑

if

19.5.4 值

x = x + n

19.5.5 数学

19.5.6 函数
19.5.7 (array)

19.5.8 (text)

19.6 GPIO

19.6.1 GPIO
Chapter 19. MU Self-driving Kit MakeCode Tutorial
19.6.2 ADC-DAC

ADC (Analog) to DAC (Digital) conversion:

\[ \text{Volt} = 3.3 \times \frac{\text{Value}}{1024} \]

DAC (Digital) to ADC (Analog) conversion:

\[ \text{Volt} = 3.3 \times \frac{\text{Value}}{1024} \]

---

DAC conversion flowchart:

\[ \text{Volt} = 3.3 \times \frac{\text{Value}}{1024} \]

---

ADC conversion flowchart:

\[ \text{Volt} = 3.3 \times \frac{\text{Value}}{1024} \]
19.6.3 PWM

PWM

19.6.4 IIC SPI

IIC SPI

19.7 MU3

MU3

19.7. MU3
19.7.1 WiFi
19.7.2 PID
19.8
19.8.1
19.8.2
19.8.3
CHAPTER TWENTY

MU SELF-DRIVING KIT RESOURCE

20.1 ?? ??

http://mai.morpx.com/page.php?a=sensor-support

GitHub https://github.com/mu-opensource/

20.2 ?? ??

2mm

20.3 3D ?? ??

MU 3D
20.4 🎁🎁🎁

MU Self-driving Kit Resource

[wiki](http://wiki.dfrobot.com.cn/index.php?title=(SKU:ROB0148)_micro:Maqueen(V2.0)%E6%9C%BA%E5%99%A8%E4%BA%BA%E5%B0%8F%E8%BD%A6)

**Micro:bit**

[https://microbit.org/zh-CN/](https://microbit.org/zh-CN/)

[MakeCode](https://makecode.microbit.org/#)
TECHNICAL SUPPORT

Thanks for purchasing our products, and we would like to provide continuous updating service, please check to our website: www.morpx.com regularly. Updates are subject to change without notice. You can get the latest technical information from the following websites:

Official Website: http://mai.morpx.com/page.php?a=moonbot-kit
GitHub: https://github.com/mu-opensource/
Phone Number: 0571-81958588
Email: support@morpx.com
Wechat ID

![Wechat QR Code]

QQ ID
This website is document of products produced by Morpx Inc. The website can be opened in computer and mobile devices.

22.1 /Page Usage Notes

Click the Read the Docs button in the left-bottom corner to open the sidebar pannel. Functions are shown below.

- /Change Language

Most documents have Chinese and English version. English version is the translation of the Chinese. There is no corresponding page if the page does not exist when changing language. Just click the left-top button to return to the homepage and change again. If there is any problem, please let us know.
• **Version Control**

The newest document is in latest branch. Other stable branch will be shown when published.

• **Download the Docs**

There are pdf, html and Epub files available for downloading.
23.1 MU Vision Sensor Disclaimer & Copyright

- The information in this manual applies to the MU Vision Sensor III is produced by Morpx Inc. Please check the Morpx Inc's website http://www.morpx.com for the latest version of the firmware and library functions. Updates are subject to change without notice.

- Please read this manual carefully before using MU Vision Sensor and make sure you understand it. Incorrect operation may cause the device stopping working, getting worse detection results, or even damaging the device.

- Morpx Inc will not warrant the damage caused by unauthorized repair or modification of electronic components on the product.

- The technical solution, vision algorithms and communication protocol mentioned in this manual is developed by Morpx and protected by intellectual property rights. Organizations or individuals should not copy or plagiarize the technical achievements of Morpx Inc. In case of any infringement, Morpx will take legal actions to protect its rights.

- Morpx is the registered trademark of Morpx, Inc., and MU is the registered trademark of MU Vision Sensor. All trademarks (names and patterns) presented here in the text or figures belong to the holders of the marks.

Copyrights © 2019 Morpx Inc. All rights reserved.
CHAPTER TWENTYFOUR

SOFTWARE LICENSES

24.1 Opensource Software

24.1.1 Arduino

• Arduino is an open-source physical computing platform based on a simple I/O board and a development environment that implements the Processing/Wiring language. Arduino can be used to develop stand-alone interactive objects or can be connected to software on your computer (e.g. Flash, Processing and MaxMSP). The boards can be assembled by hand or purchased preassembled; the open-source IDE can be downloaded for free at https://www.arduino.cc/en/Main/Software

• Arduino is an open source project, supported by many.

• The Arduino team is composed of Massimo Banzi, David Cuartielles, Tom Igoe and David A. Mellis.

• Arduino uses GNU avr-gcc toolchain, GCC ARM Embedded toolchain, avr-libc, avrdude, bossac, openOCD and code from Processing and Wiring.

• Icon and about image designed by ToDo.

24.1.2 MicroPython

• MicroPython is written in C99 and the entire MicroPython core is available for general use under the very liberal MIT license. Most libraries and extension modules (some of which are from a third party) are also available under MIT or similar licenses.

• You can freely use and adapt MicroPython for personal use, in education, and in commercial products.

• MicroPython is developed in the open on GitHub and the source code is available at the GitHub page, and on the download page. Everyone is welcome to contribute to the project.
class LSM303AGR_IMU_Sensor, 193
class MoonBotServo, 197
class MoonBotTankBase, 184
class Motor, 181
class WT2003S, 188

eenum imu_state_t, 192
eenum lsm303_acc_angle_t, 192
eenum lsm303_axes_t, 192
eenum moonbot_eyes_scroll_t, 201
eenum moonbot_eyes_t, 201
eenum moonbot_motor_t, 177, 181
eenum moonbot_port_t, 178
eenum moonbot_servo_t, 177, 197
eenum motor_pin_t, 177
eenum motor_type_t, 177, 183
eenum port_pin_t, 178
eenum servo_pin_t, 177

MOONBOT_PIN_BUTTON_A, 178
MOONBOT_PIN_BUTTON_B, 178
MOONBOT_PIN_BUZZER_SHDW, 178
MOONBOT_PIN_BUZZER_SIG, 178
MOONBOT_PIN_LED, 178

uint8_t moonbotMotorToPin(moonbot_motor_t motor_num, motor_pin_t pin_type), 179
uint8_t moonbotPortToPin(moonbot_port_t port_num, port_pin_t pin_num), 179
uint8_t moonbotServoToPin(moonbot_servo_t servo_num, servo_pin_t pin_type), 179

void colorFade(Adafruit_NeoPixel& led, uint8_t r, uint8_t g, uint8_t b, uint8_t wait), 202
void colorWipe(Adafruit_NeoPixel& led, uint32_t c, uint8_t wait), 202
void MoonBotEyesCircle(Adafruit_NeoPixel& led, uint32_t color, moonbot_eyes_t eyes_type = kEyesBoth, uint8_t wait = 50), 203
void MoonBotEyesLook(Adafruit_NeoPixel& led, moonbot_look_t look_tpye, uint32_t color), 202
void MoonBotEyesScroll(Adafruit_NeoPixel& led, moonbot_eyes_scroll_t scroll_tpye, uint32_t color, uint8_t wait = 50), 203
void rainbow(Adafruit_NeoPixel& led, uint8_t wait), 202
void rainbowCycle(Adafruit_NeoPixel& led, uint8_t wait), 202
void theaterChase(Adafruit_NeoPixel& led, uint32_t c, uint8_t wait), 202