M-Vision Tutorial

Contents:

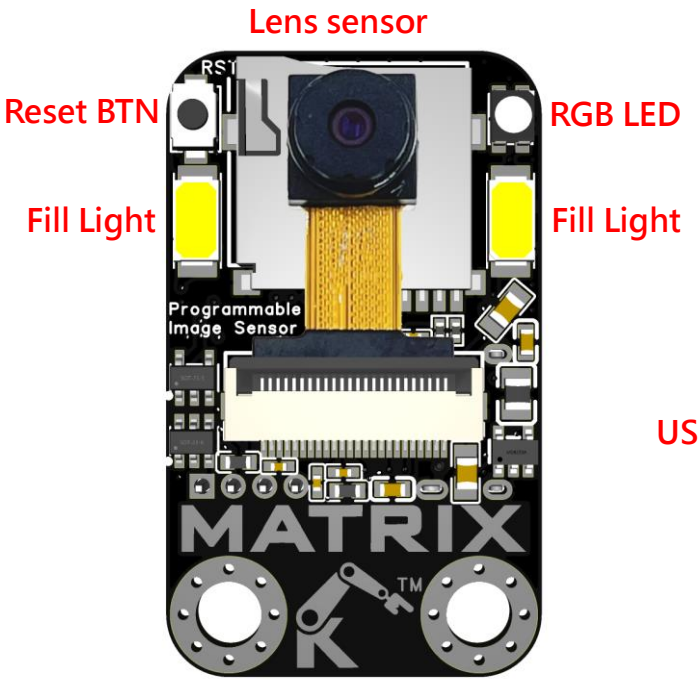
Chapter 1 Introduction to M-Vision

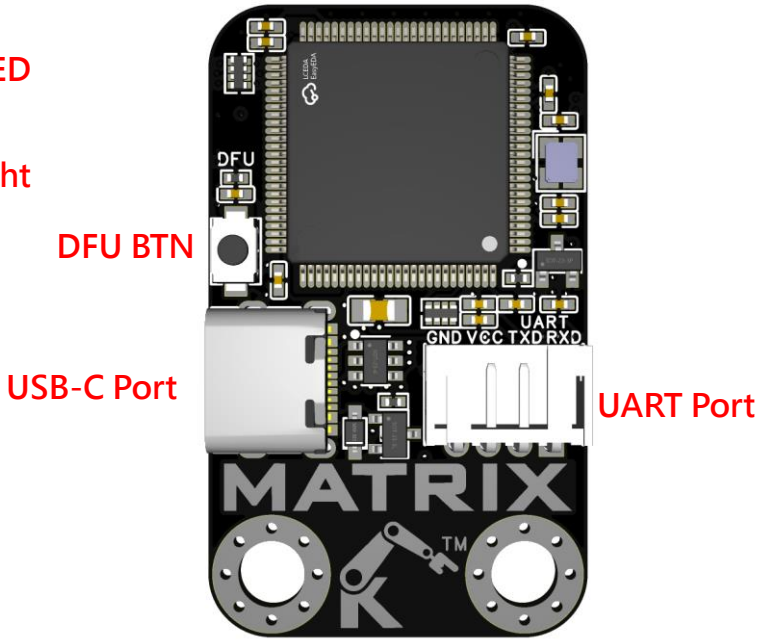
1-1 Camera Introduction

M-Vision is a programmable smart camera primarily used for embedded machine vision and image processing. Equipped with a microcontroller and a vision module, it enables various vision processing functions with low power consumption, making it suitable for applications in robotics, IoT, and AI-based machine learning tasks.

M-Vision uses the MicroPython language, allowing users to easily get started with developing vision applications. Programming is done via the OpenMV IDE, which enables real-time monitoring of the image feed while writing code and adjusting parameters.

Appearance introduction:





Function introduction:

- Lens Sensor: The core component of M-Vision, which converts light into electronic signals.

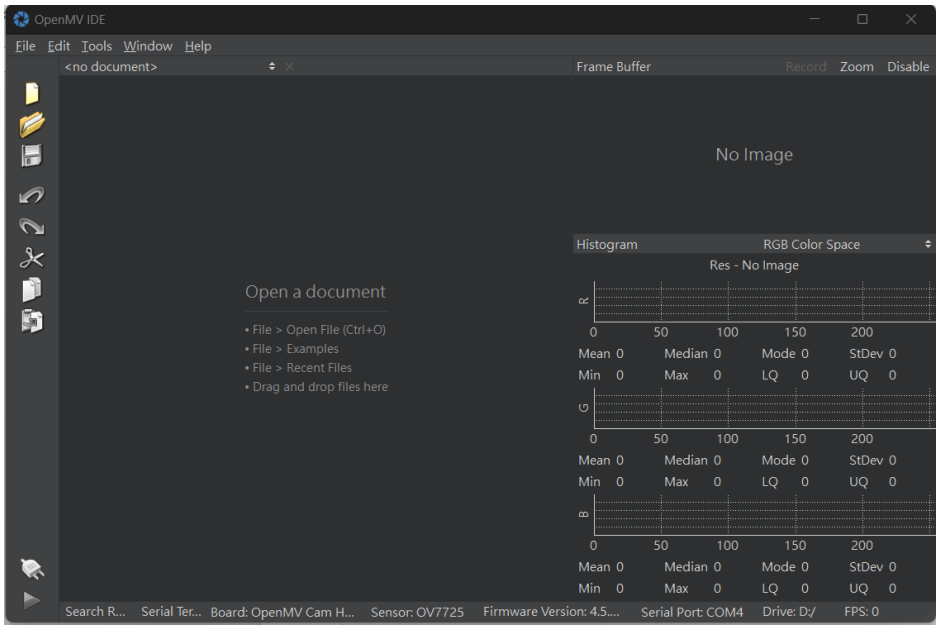
- Reset Button: Restarts the main program.
- RGB LED: Programmable RGB LED light.
- Fill Light: A light source that can be turned on to increase the environment's brightness when there is insufficient light.
- DFU Button: Device Firmware Upgrade. Press this button and connect the power supply to enter firmware update mode when the device is powered off.
- USB-C Port: Used for connecting to a computer via a USB data cable for programming using OpenMV IDE.
- UART Port: Used for UART communication to connect with other controllers.

1-2 Setup and Operating Environment – OpenMV IDE

Visit the official OpenMV website to download and install the version suitable for your operating system.

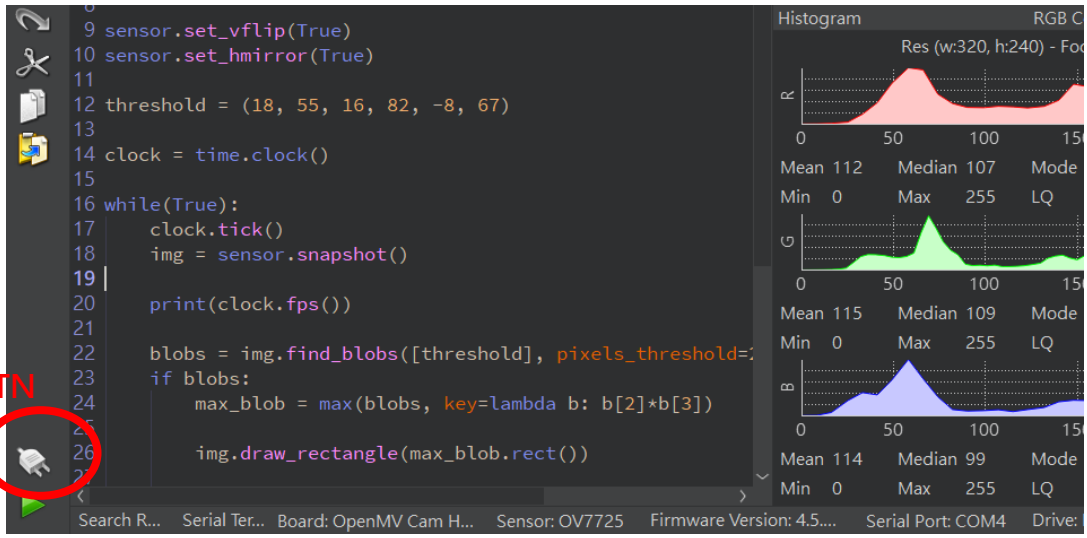
- Link: https://openmv.io/pages/download

After installation is complete, opening the OpenMV IDE will display the following screen, indicating that the installation was successful.

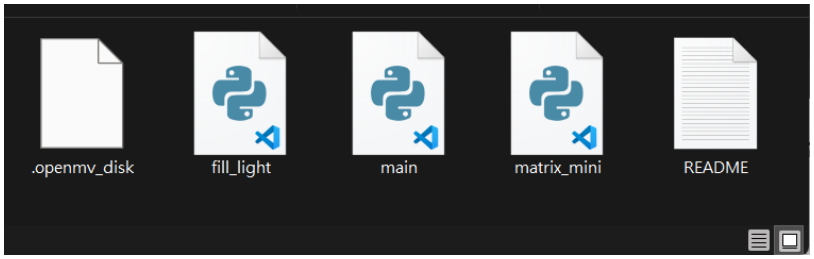


Use the USB data cable included in the package to connect your computer to M-Vision, and click the connect button in the lower left corner of the screen.

Notice: Must use USB-A to USB-C data cable to connect.



Once M-Vision is successfully connected to the computer, you can find M-Vision's storage space in the File Explorer and place three basic programs inside it (main.py, matrix_mini.py, fill_light.py).



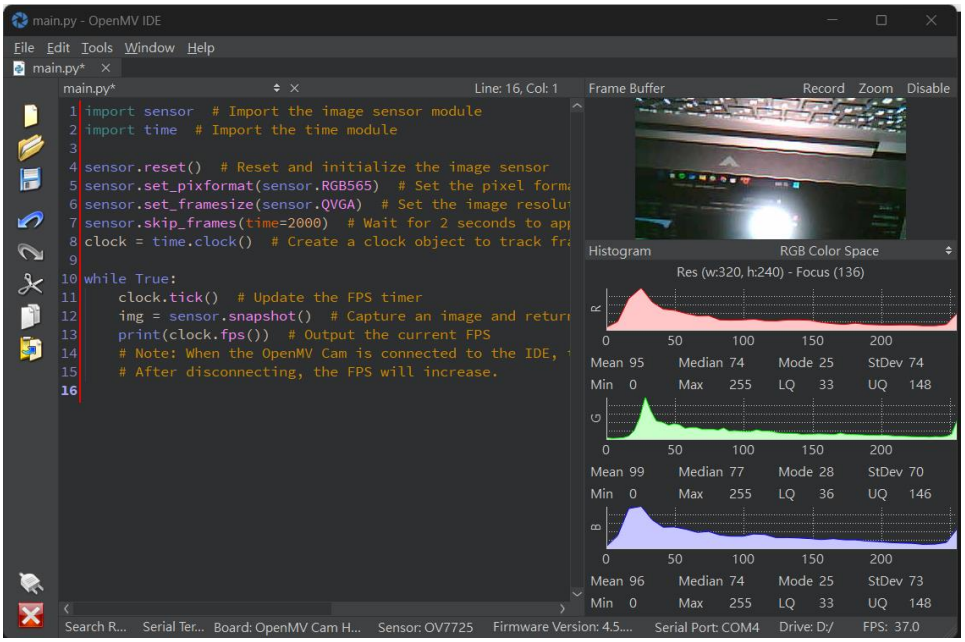
- main.py: The main program, which will be automatically executed after M-Vision is powered on.
- matrix_mini.py: The communication program between OpenMV and the Matrix controller.
- fill_light.py: The fill light driver program.

Test:

Open main.py in OpenMV IDE and copy the following code into the IDE to execute it. If it runs correctly, you will see the M-Vision screen in the upper right corner. If it fails, the screen will not appear.

```
import sensor  # Import the image sensor module
import time  # Import the time module

sensor.reset()  # Reset and initialize the image sensor
sensor.set_pixformat(sensor.RGB565)  # Set the pixel format to RGB565 (or set it to GRAYSCALE)
sensor.set_framesize(sensor.QVGA)  # Set the image resolution to QVGA (320x240)
sensor.skip_frames(time=2000)  # Wait for 2 seconds to apply the settings
clock = time.clock()  # Create a clock object to track frames per second (FPS)

while True:
    clock.tick()  # Update the FPS timer
    img = sensor.snapshot()  # Capture an image and return the image object
    print(clock.fps())  # Output the current FPS
    # Note: When the OpenMV Cam is connected to the IDE, the execution speed will be reduced by about half.
    # After disconnecting, the FPS will increase.
```

Successful execution screen of the program.



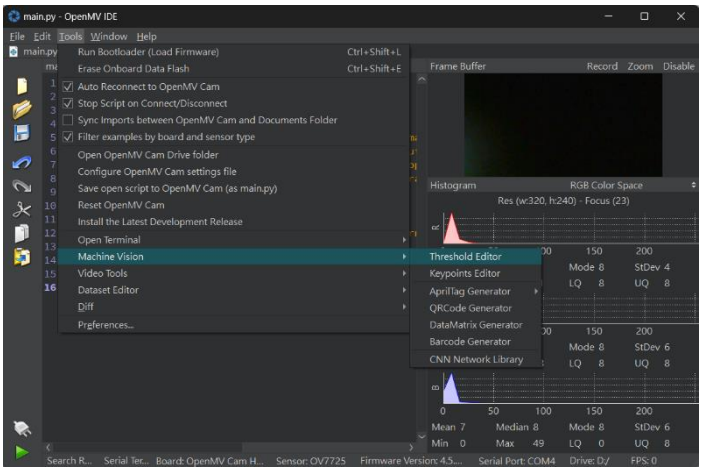
1-3 Example

To find blocks, copy the following program into OpenMV IDE, run it, open the threshold editor, and adjust the threshold.

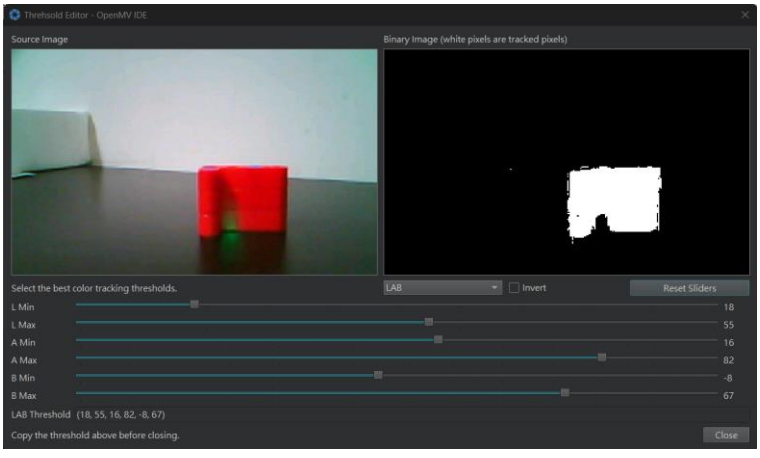
```
import sensor, image, time  # Import image sensor and time modules
from matrix_mini import send_data  # Import the send_data function from the matrix_mini module

# Initialize the sensor
sensor.reset()  # Reset the sensor
sensor.set_pixformat(sensor.RGB565)  # Set the pixel format to RGB565
sensor.set_framesize(sensor.QVGA)  # Set the image resolution to QVGA (320x240)
sensor.skip_frames(time=2000)  # Wait for 2 seconds to stabilize the sensor

# Set image flipping
sensor.set_vflip(True)  # Vertically flip the image
sensor.set_hmirror(True)  # Horizontally mirror the image

# Define color threshold range (HSV), suitable for detecting the target color
threshold = ( 0, 0, 0, 0, 0, 0)  # Hue, Saturation, and Value range

clock = time.clock()  # Create a clock to calculate frames per second (FPS) of image processing

while(True):
    clock.tick()  # Start timing
    img = sensor.snapshot()  # Capture an image

    print(clock.fps())  # Output the current FPS
```

```
24.
25.     # Find blobs (color regions) in the image that match the threshold
26.     blobs = img.find_blobs([threshold], pixels_threshold=200, area_threshold=200)
27.     if blobs:  # If blobs are found
28.         # Find the largest blob by area
29.         max_blob = max(blobs, key=lambda b: b[2]*b[3])
30.
31.         img.draw_rectangle(max_blob.rect())  # Draw a rectangle around the largest blob
32.
33.         # Calculate the center coordinates of the blob
34.         x_center = max_blob.cx()
35.         y_center = max_blob.cy()
36.
37.         # Calculate the blob's area and round it to an integer
38.         blob_area = round(max_blob.area() / 2)
39.
40.         # Send the center coordinates and area to an external system
41.         send_data([x_center, y_center, blob_area])
42.
43.         # Mark the center point and coordinates text on the image
44.         img.draw_cross(x_center, y_center)  # Draw a cross at the center
45.         img.draw_string(0, 0, str(x_center) + ", " + str(y_center))  # Display the coordinates
```

Open the threshold editor (Tools / Machine Vision / Threshold Editor / Frame Buffer).



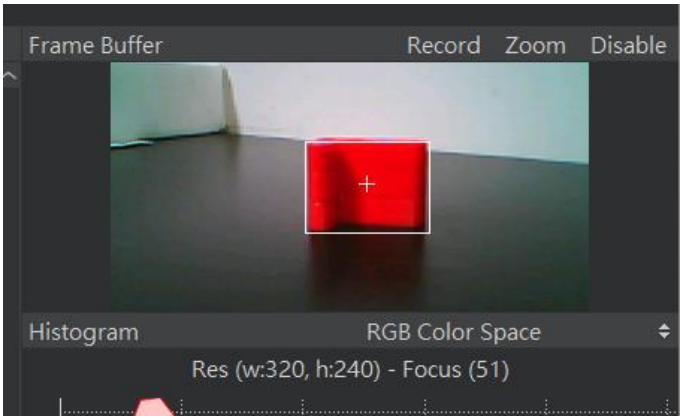
After adjusting the LAB threshold to the appropriate position, copy the threshold.



After changing the threshold to the threshold variable, save the file and rerun the program.

```
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 2000)

sensor.set_vflip(True)
sensor.set_hmirror(True)

threshold = (18, 55, 16, 82, -8, 67)

clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
```

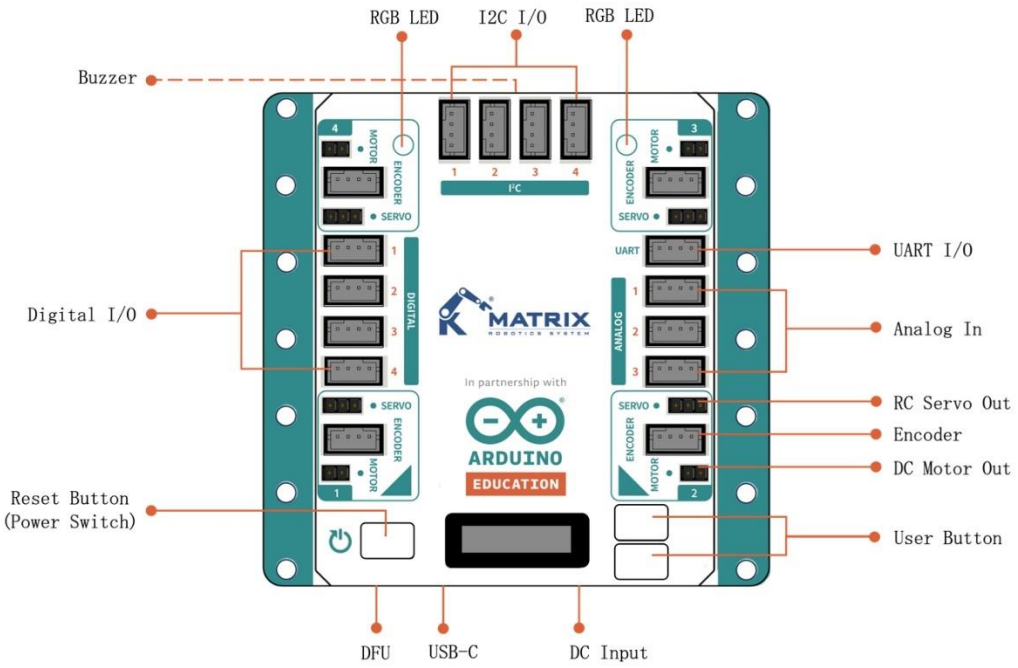
Successfully found the block.



Chapter 2 Introduction to Matrix Mini R4

2-1 Introduction to the Matrix Mini R4 Controller

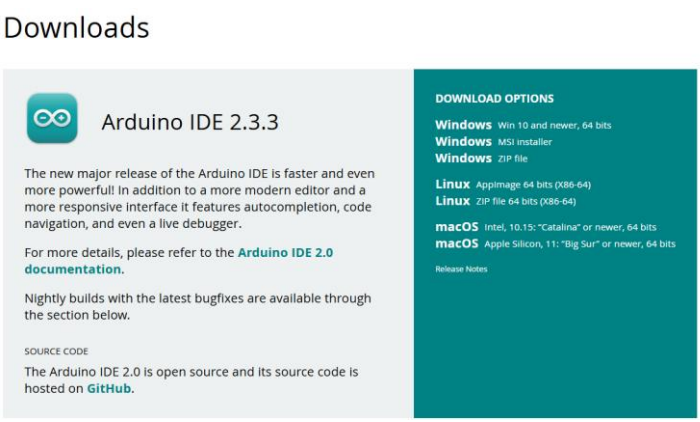
The Matrix Mini R4 Controller is based on the Arduino UNO R4 Wi-Fi, extended and integrated with common robot functions to make it easier for users to write programs and connect circuits more intuitively.



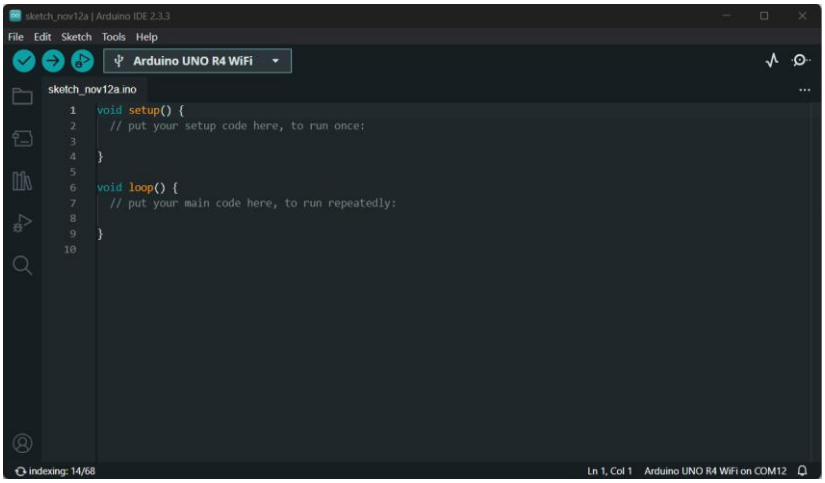
2-2 Setup and Operating Environment – Arduino IDE

Visit the official Arduino website to download and install the version suitable for your operating system.

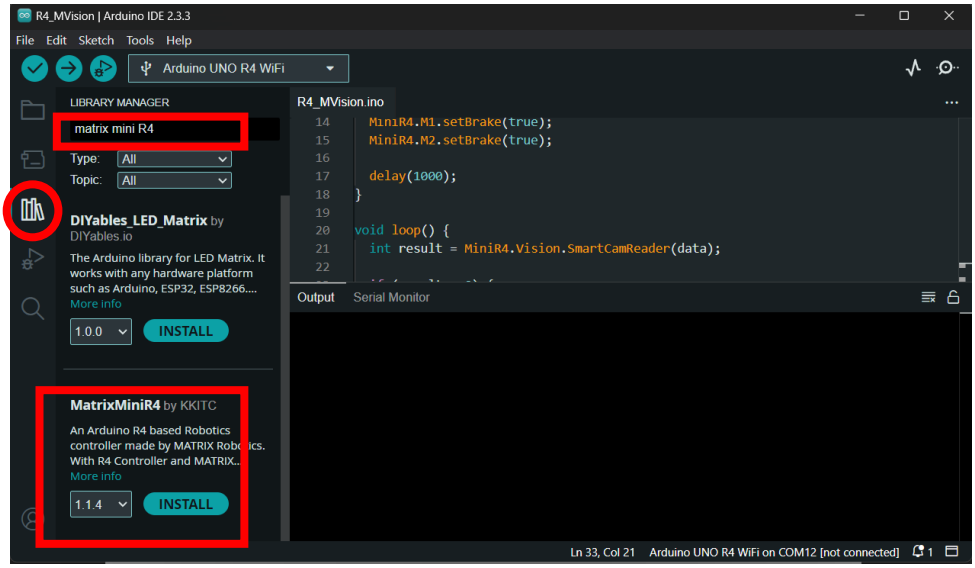
- Link: https://www.arduino.cc/en/software



After installation is complete, opening the Arduino IDE will display the following screen, indicating that the installation was successful.

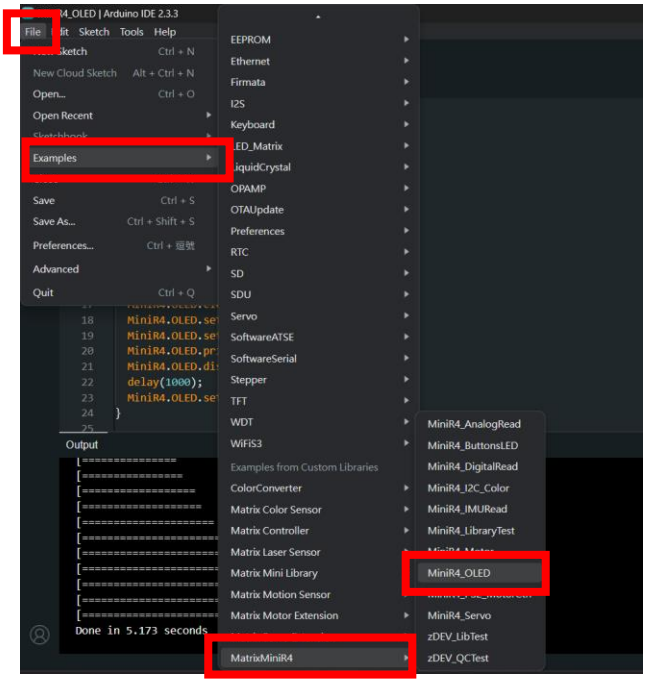


Use the Library Manager in Arduino IDE to install the dedicated library for Matrix Mini R4. Click 'Library Manager,' search for 'Matrix Mini R4,' find 'MatrixMiniR4 by KKITC,' and click Install.

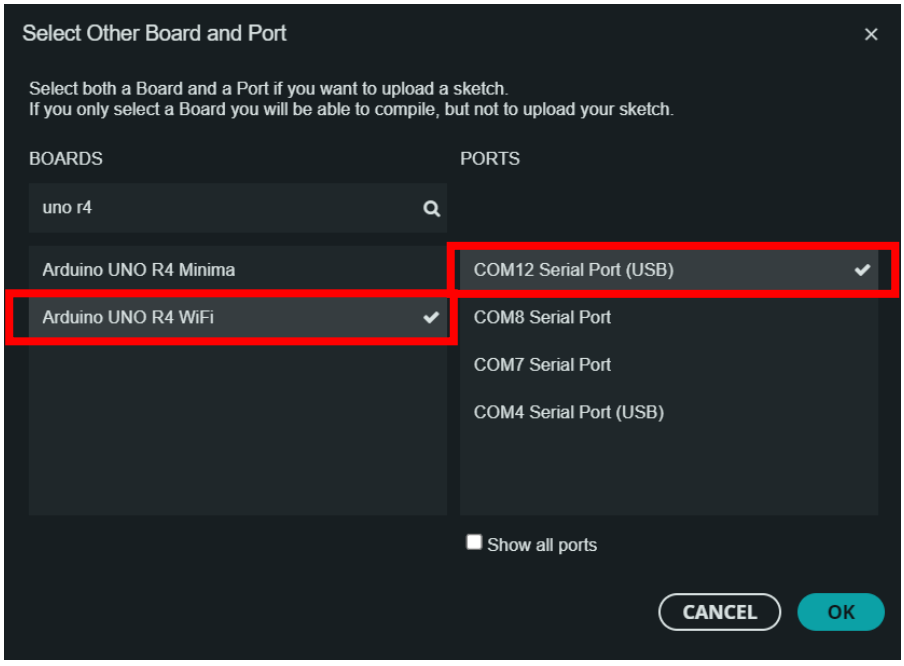


Test:

Open the built-in example 'MiniR4_OLED' from the Matrix Mini R4 library in the IDE (File / Examples / MatrixMiniR4 / MiniR4_OLED.ino).



Select the board model, choose the correct port in Arduino IDE, and select 'Arduino UNO R4 WiFi' as the board.



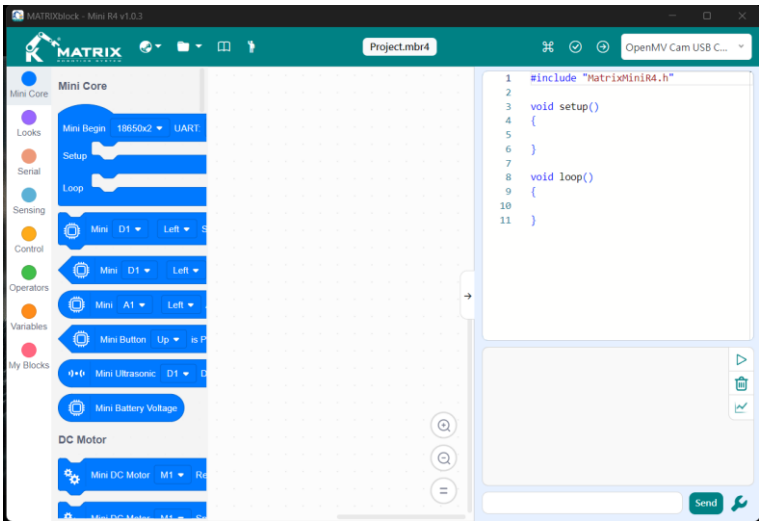
After setup is complete, press and hold the power button to turn on the controller, then click to upload the program to the controller. If the upload is successful, you will see the voltage and battery percentage displayed on the controller's OLED screen.

2-3 Setup and Operating Environment – MATRIXblock

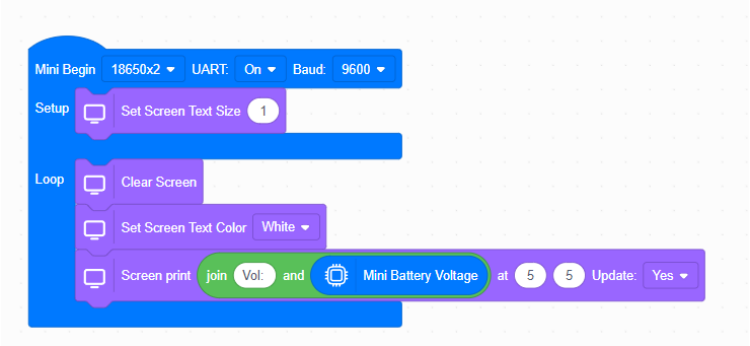
Go to the Matrix Robotics official website to download and install the version suitable for your operating system.

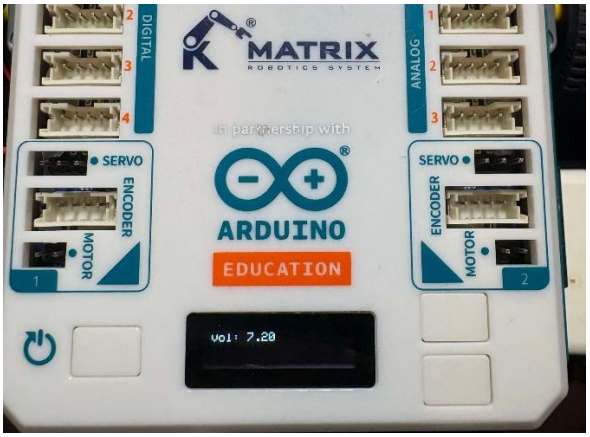
- Link: https://www.matrixrobotics.com/matrixblock-software

After the installation is complete, open MATRIXblock. If you see the following screen, it indicates that the installation was successful.



Write the following program. Press and hold the power button on the controller to turn it on, then click to upload the program to the controller. If the upload is successful, the voltage will be displayed on the controller's OLED screen.

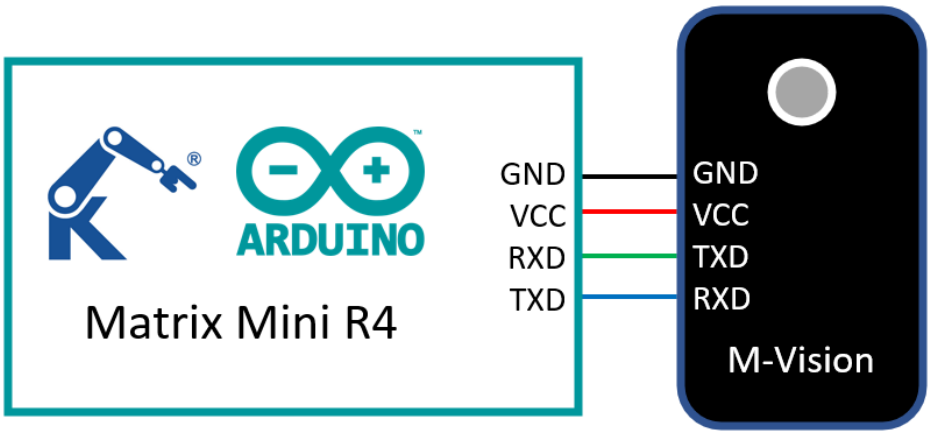




Chapter 3 Communication between Matrix Mini R4 and M-Vision

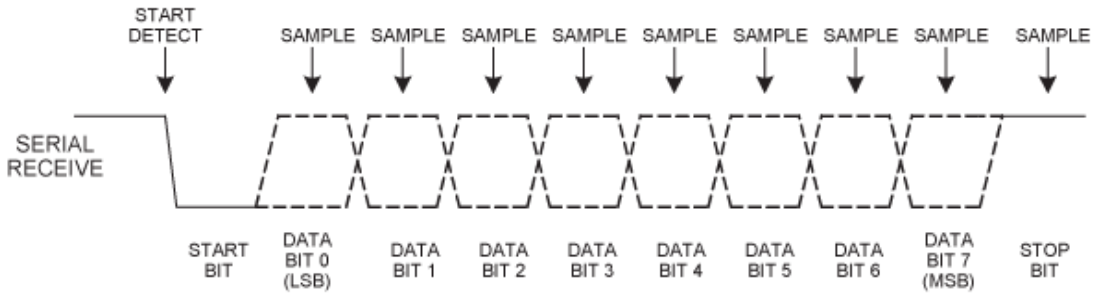
3-1 Wiring Instructions

The Matrix Mini R4 controller and M-Vision communicate through the UART communication protocol. The wiring for UART is as follows:



3-2 UART Communication Protocol

UART (Universal Asynchronous Receiver/Transmitter) is a serial communication protocol used for full-duplex communication between two devices. UART is one of the common communication methods in embedded systems, microcontrollers, and other electronic devices. It allows serial data transmission between devices without the need to share a clock signal.



3-3 Example

Object tracking robot, Arduino IDE example program:

```
#include <MatrixMiniR4.h>  // Include the Matrix Mini R4 controller library

unsigned int data[20];  // Array to store data received from SmartCam
unsigned int cX, cY = 0;  // Variables to store the X and Y coordinates of the object in the image

void setup() {
  Serial.begin(115200);  // Set the serial communication speed to 115200 bps
  MiniR4.begin();  // Initialize the Matrix Mini R4 controller
  MiniR4.Vision.Begin();  // Start the image processing module

  MiniR4.M2.setReverse(true);  // Set motor M2 to reverse direction

  MiniR4.M1.setBrake(true);  // Enable brake for motor M1
  MiniR4.M2.setBrake(true);  // Enable brake for motor M2

  delay(1000);  // Wait for 1 second to ensure initialization is complete
}

void loop() {
  int result = MiniR4.Vision.SmartCamReader(data);  // Read image data from SmartCam and store it in the data array
  if (result > 0) {  // If valid data is received
    cX = data[0];  // Get the X coordinate of the object
    cY = data[1];  // Get the Y coordinate of the object

    int error = cX - 160;  // Calculate the horizontal error of the object relative to the center of the image (image width is 320)
    Move(error);  // Move the motors based on the error
  }
}
void Move(int str) {
  // Set the power for motors M1 and M2 to achieve tracking effect
  MiniR4.M1.setPower(60 + str);  // Increase M1 motor power with the error
  MiniR4.M2.setPower(60 - str);  // Decrease M2 motor power with the error
}
```

MATRIXblock example program: