REALTEK

Ameba 82 Workshop

瑞昱科技股份有限公司 & 臺灣師範大學產創學院

Outline

Chapter 1 Edge AI

1.1 AIoT

1.2 Edge computing

Chapter 2 AMB82-MINI

2.1 AMB82-MINI Introduction

2.2 LoopPostProcessing

2.3 Audio Classification

2.4 Face Recognition

2.5 Image Classification

2.6 MQTT ON AMB82

Outline

Chapter 3 Object Detection

3.1 YOLO(You Only Look Once)

3.2 YOLOv7 Gesture Detection

Chapter 4 Application of Object Detection

4.1 Parking Cars

4.2 Tango Dance

4.3 Obstacle course racing

Chapter 1
Edge - AI

1.1 AIoT

Definition of AIoT

- AIoT aims to improve the efficiency and intelligence of IoT devices and systems through smart analysis and automated decision-making.

Key Features of AIoT

- **Data Collection and Analysis: IoT devices send large data via sensors, and AI analyzes it to extract valuable insights and patterns.**

Key Features of AIoT

- **Smart Decision-Making: AI can automatically make decisions based on data analysis and implement them through IoT devices.**

Key Features of AIoT

- **Prediction and Prevention: Using machine learning models, AIoT systems can predict future events, aiding in preventive maintenance and resource optimization.**

Key Features of AIoT

- **Self-learning and Optimization: AIoT systems learn from past data to improve performance and decision-making, adapting to changes for more efficient services.**

Layers of AIoT

- **Device Layer: Sensors and actuators are physical devices that collect data from the environment and perform actions.**
- **Connectivity Layer： This layer includes communication between processing devices and other layers. Ex: WiFi, Bluetooth.**

Layers of AIoT

- **Edge Computing Layer: Computing devices close to sensors and actuators perform real-time data processing and analysis.**
- **Data Management Layer：Systems and databases that store large amounts of data from IoT devices, including local and cloud storage.**

Layers of AIoT

- **AI Analytics Layer: AI algorithms and models are used to analyze processed data for predictions and automated decision-making.**
- **Application Layer：Allow users to interact with the AIoT systems, monitor its status, and control devices.**

1.2 Edge Computing

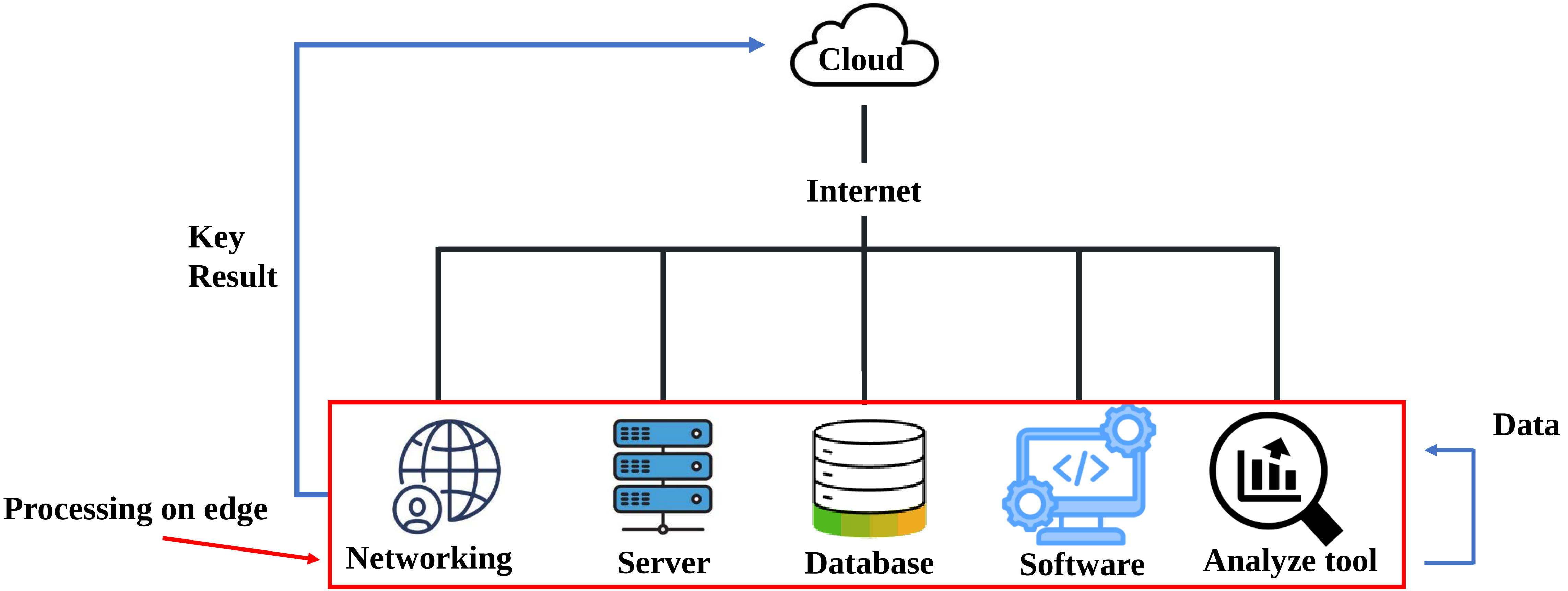
Edge Computing

- **Definition: shift the data processing and analysis from centralized cloud architectures to edge devices that are closer to the data generation source.**

Edge Computing

- **Purpose: By performing real-time processing and analysis on edge devices, edge computing aims to reduce latency, decrease bandwidth requirements, and enhance system reliability as well as data security.**

Edge Computing



Why edge computing matters?

- Reduce Latency / Improve Speed
- Enhanced Data Security
- Increased Productivity
- Ease of Integration
- Cost Reduction

Edge Computing use cases

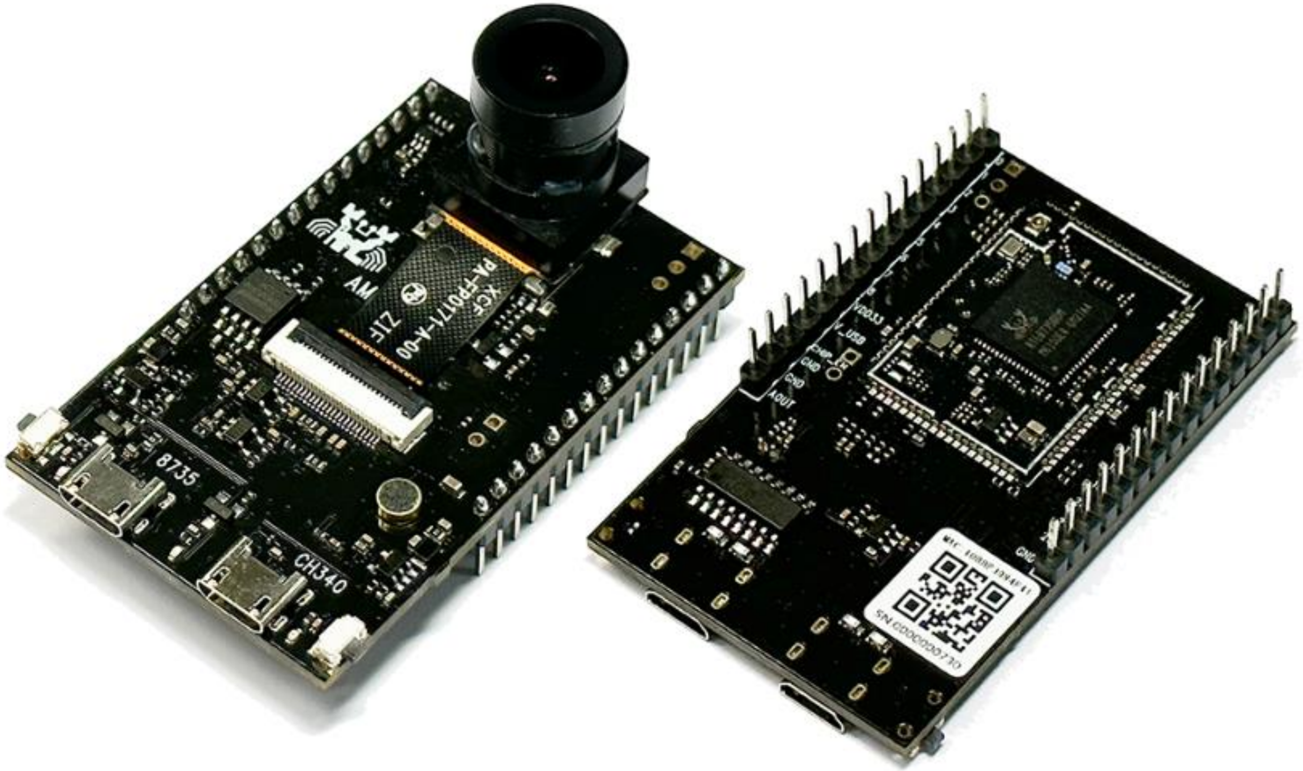
- Retail industry
- Autonomous driving
- Healthcare
- Education

Chapter 2

AMB82-MINI

2.1 AMB82-MINI Introduction

2.1 AMB82-MINI Introduction



https://www.amebaiot.com/zh/amebapro2/

What can AMB82 MINI do?

- WiFi/BLE
- GPIO/PWM
- E-Paper
- Audio/Video
- AI Neural Network

2.2 LoopPostProcessing

2.2 LoopPostProcessing

What is Motion Detection?

- **Definition: Dynamically read a video to detect changing positions.**

2.2 LoopPostProcessing

How do Motion Detection works?

Here are some answers given by ChatGPT

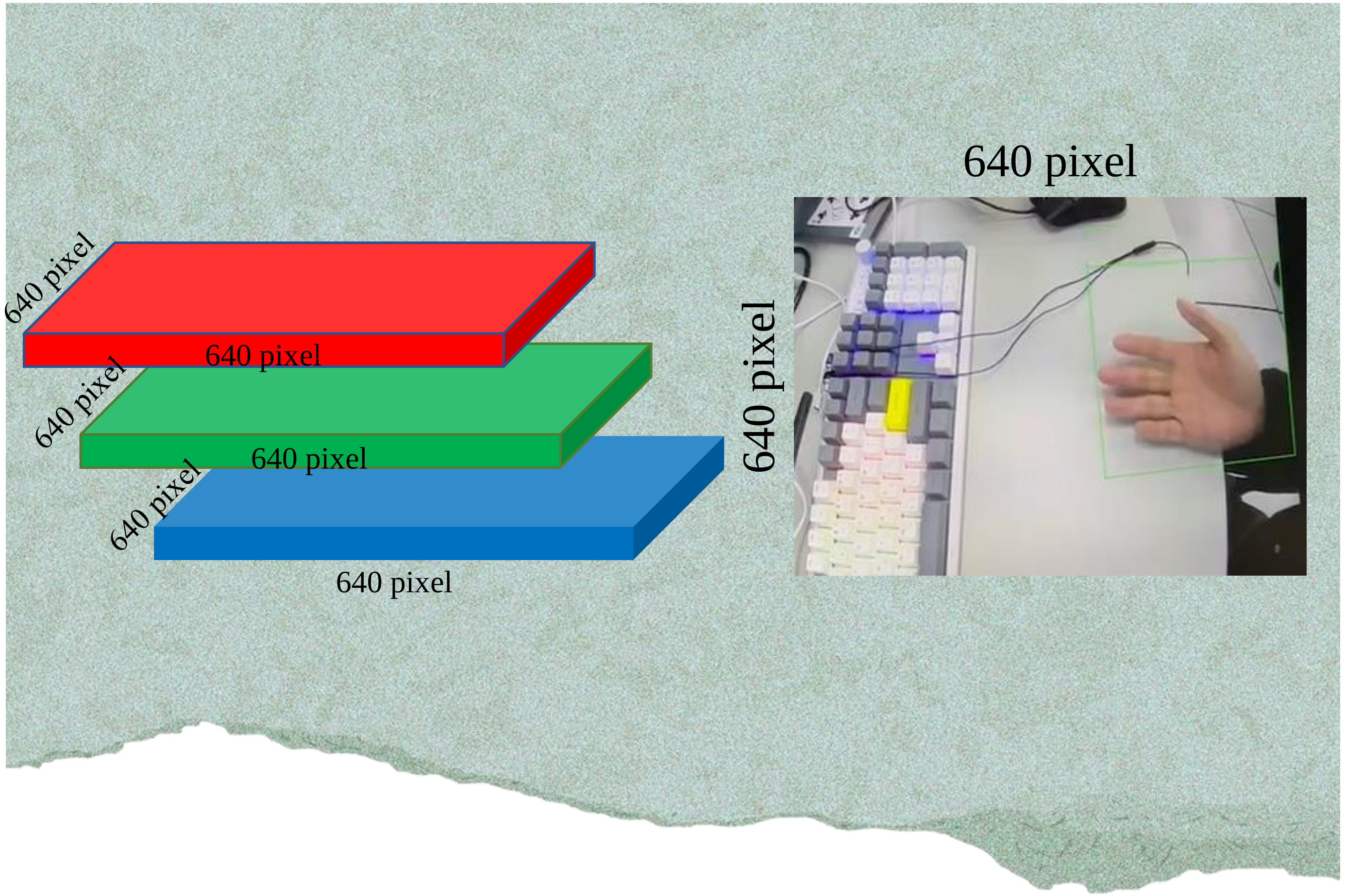
1. Color Detection
2. Depth Camera
3. Machine learning
4. Optical flow

How do AMB actually works?

- Calculate the RGB differences between two adjacent frames and use a threshold to determine if there is any motion change.

2.2 LoopPostProcessing

RGB channel



2.2 LoopPostProcessing

How to define a difference?

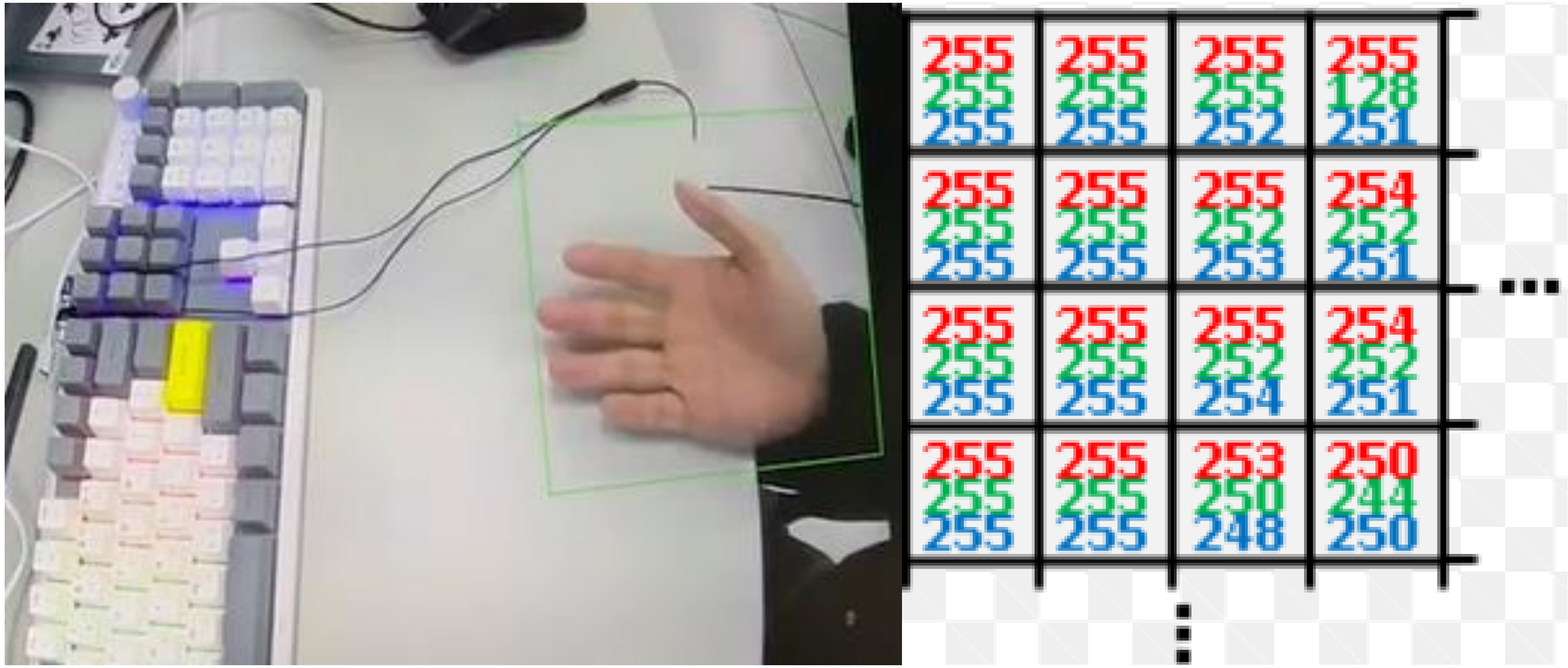
Observe the **RGB value changes** of all pixels in each frame.

Assuming RGB value of two different frames as

(R1, G1, B1) and (R2, G2, B2).

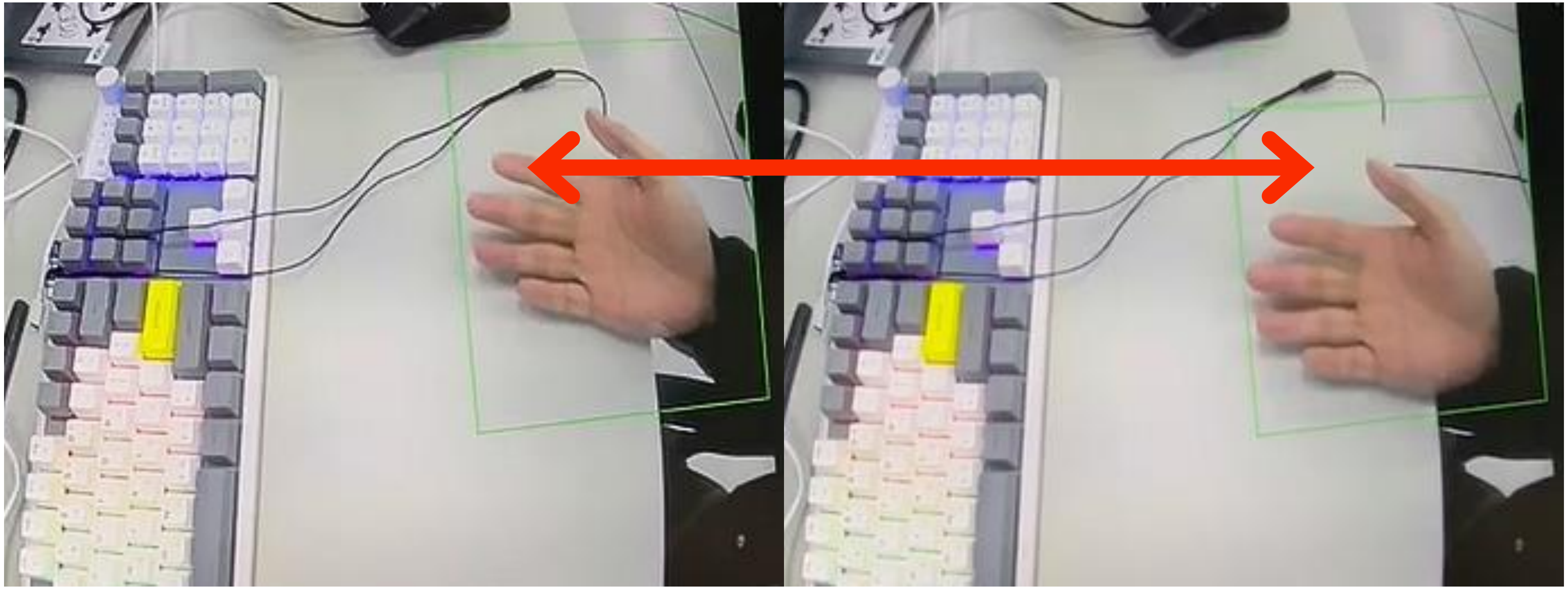
$$\text{diff} = \sqrt{(R2 - R1)^2 + (G2 - G1)^2 + (B2 - B1)^2}$$

2.2 LoopPostProcessing



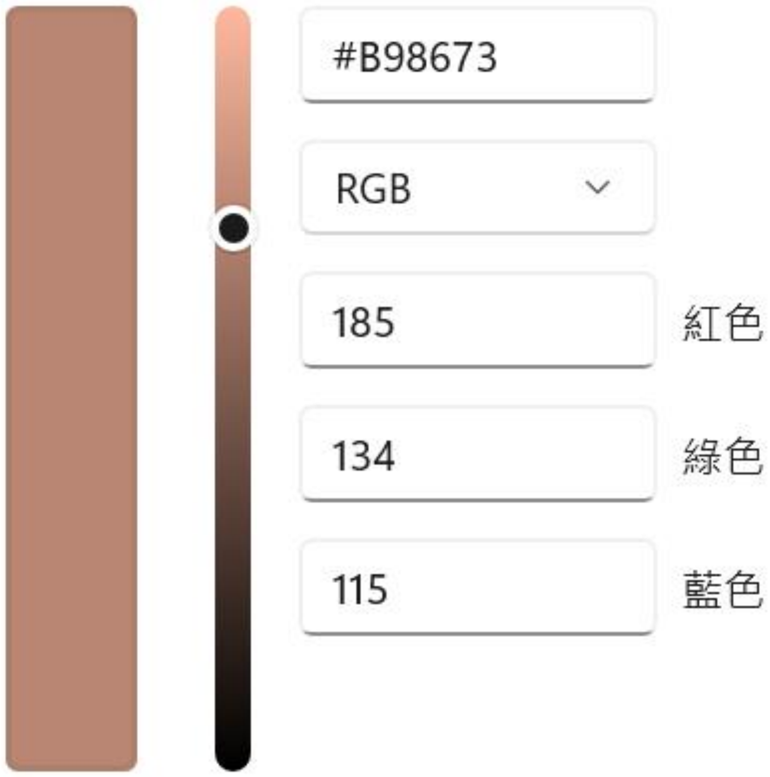
255 255 255	255 255 255	255 255 252	255 128 251
255 255 255	255 255 255	255 252 253	254 252 251
255 255 255	255 255 255	255 252 254	254 252 251
255 255 255	255 255 255	253 250 248	250 244 250

2.2 LoopPostProcessing



2.2 LoopPostProcessing

$$\mathrm{diff} = \sqrt{(255-185)^2 + (255-134)^2 + (255-115)^2}$$



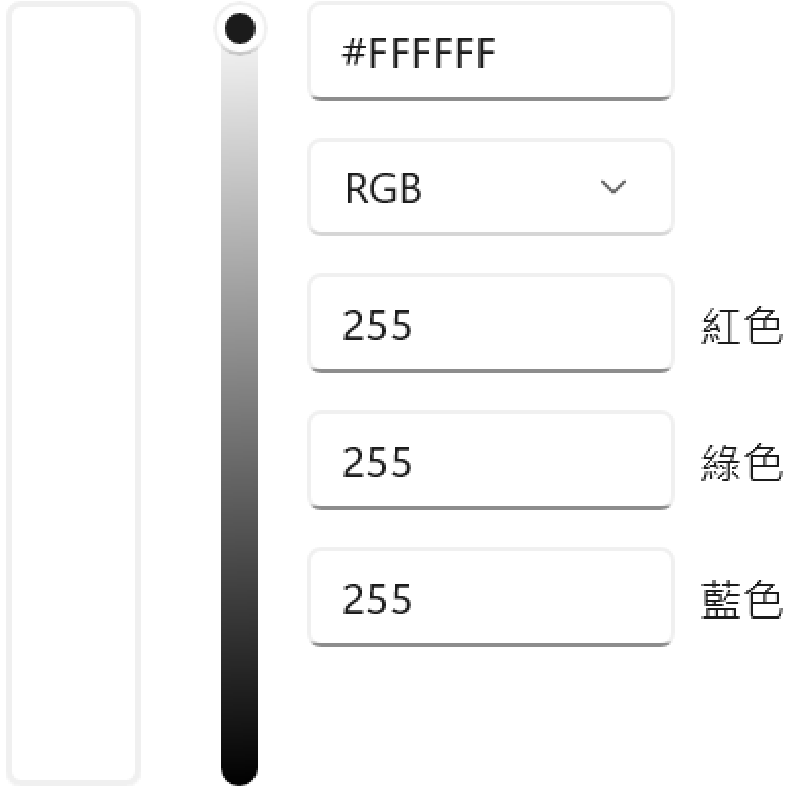
#B98673

RGB

185 紅色

134 綠色

115 藍色



#FFFFFF

RGB

255 紅色

255 綠色

255 藍色

2.2 LoopPostProcessing

$$\text{diff} = \sqrt{(255 - 185)^2 + (255 - 134)^2 + (255 - 115)^2}$$

$$\text{diff} = \sqrt{70^2 + 121^2 + 140^2}$$

$$\text{diff} = \sqrt{4900 + 14641 + 19600}$$

$$\text{diff} = \sqrt{39141}$$

$$\text{diff} \approx 197.84$$

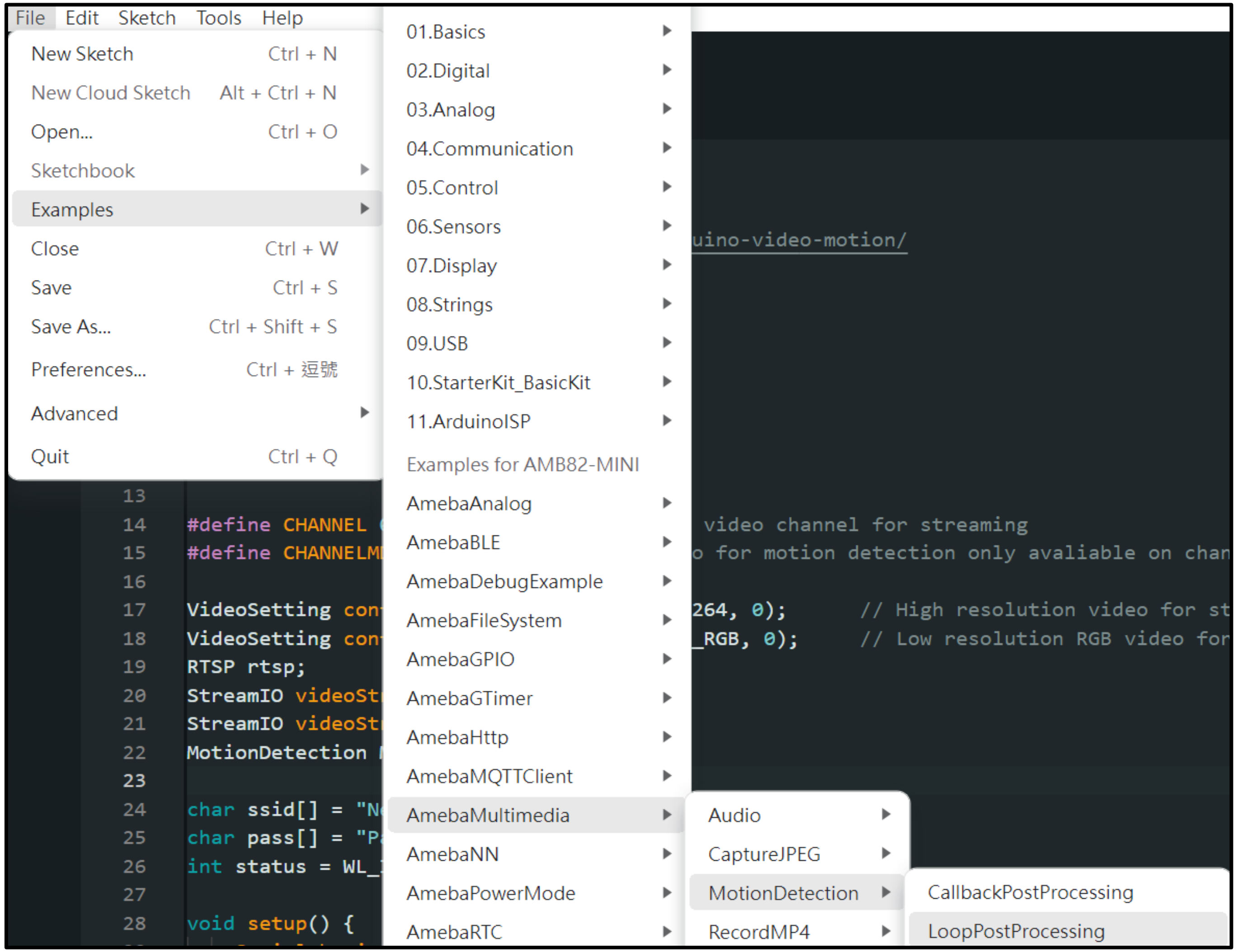
Implementation

2.2 LoopPostProcessing

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaMultimedia**
4. **MotionDetection**
5. **LoopPostProcessing**



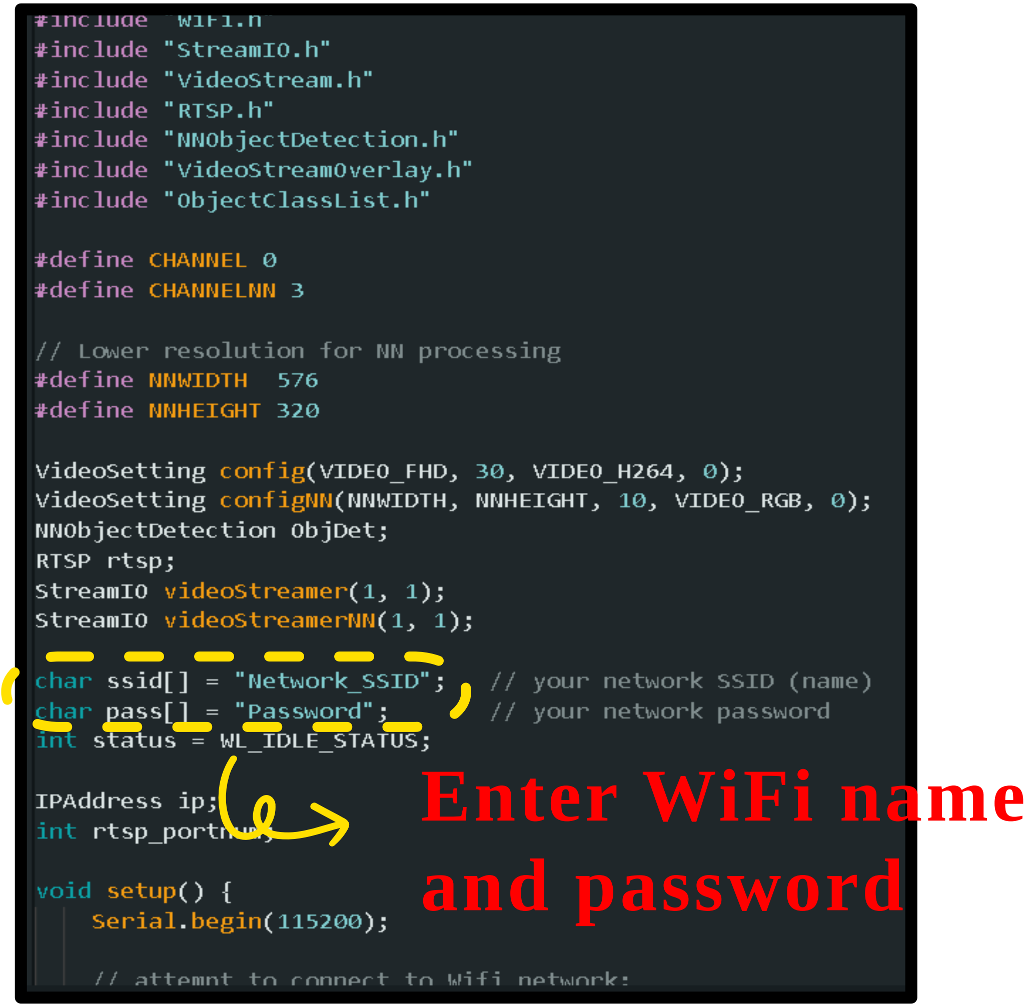
```
uino-video-motion/

#define CHANNEL                  video channel for streaming
#define CHANNELMD               o for motion detection only avaliable on chan

VideoSetting con               264, 0);      // High resolution video for st
VideoSetting con               _RGB, 0);     // Low resolution RGB video for
RTSP rtsp;
StreamIO videoSt
StreamIO videoSt
MotionDetection 

char ssid[] = "N
char pass[] = "P
int status = WL_

void setup() {
```

2.2 LoopPostProcessing

Step 2.

Enter the WiFi name and password

to the corresponding place in the code.

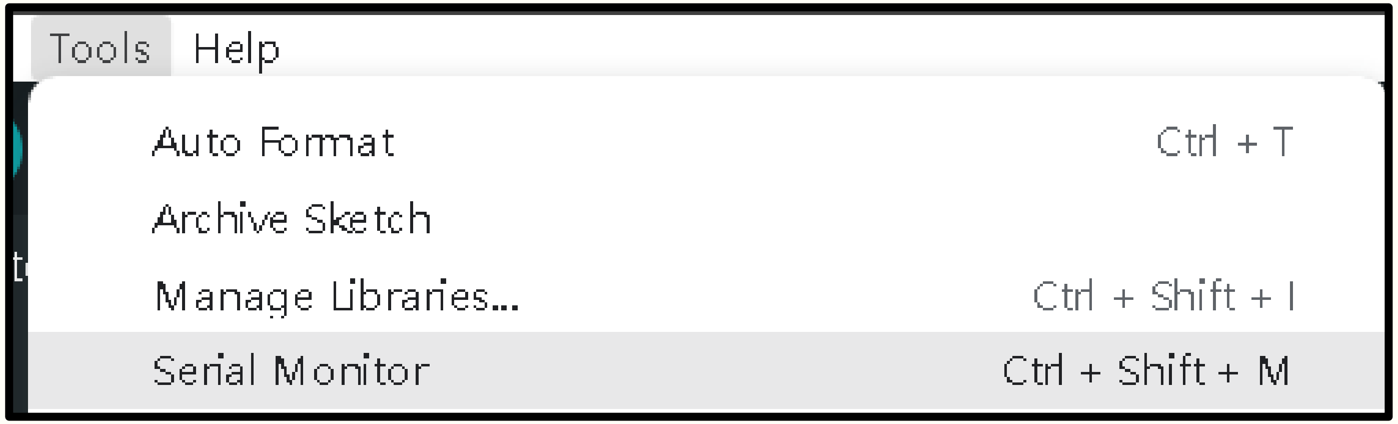


```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNObjectDetection.h"
#include "VideoStreamOverlay.h"
#include "ObjectClassList.h"

#define CHANNEL 0
#define CHANNELNN 3

// Lower resolution for NN processing
#define NNWIDTH  576
#define NNHEIGHT 320

VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
RTSP rtsp;
StreamIO videoStreamer(1, 1);
StreamIO videoStreamerNN(1, 1);

char ssid[] = "Network_SSID";    // your network SSID (name)
char pass[] = "Password";        // your network password
int status = WL_IDLE_STATUS;

IPAddress ip;
int rtsp_portnum;

void setup() {
    Serial.begin(115200);

    // attempt to connect to Wifi network:
```

Enter WiFi name and password

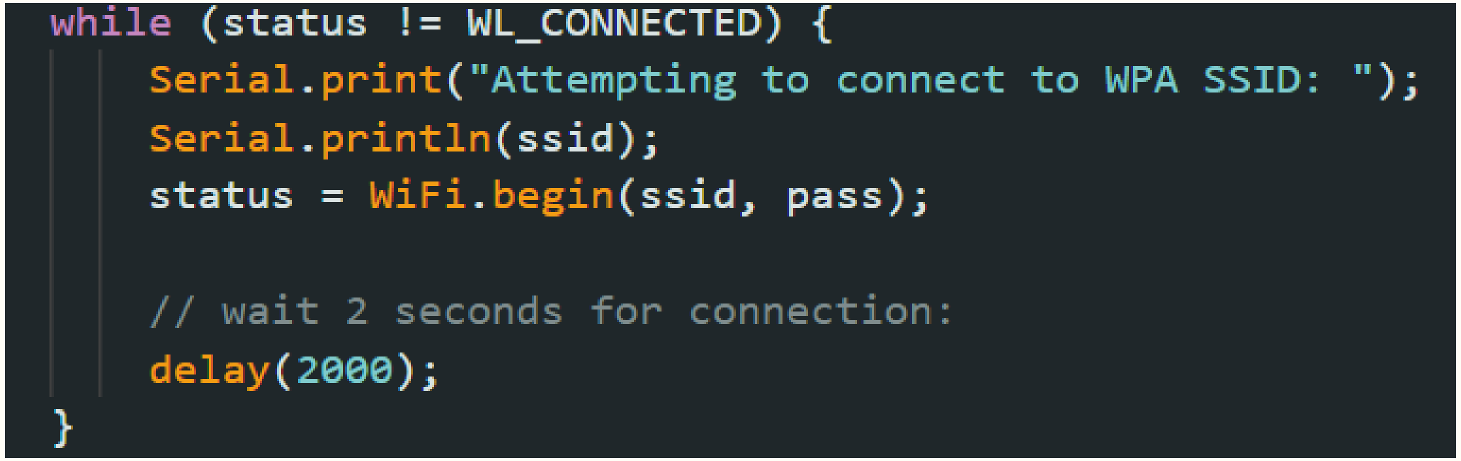
2.2 LoopPostProcessing

Open the Serial Monitor



2.2 LoopPostProcessing

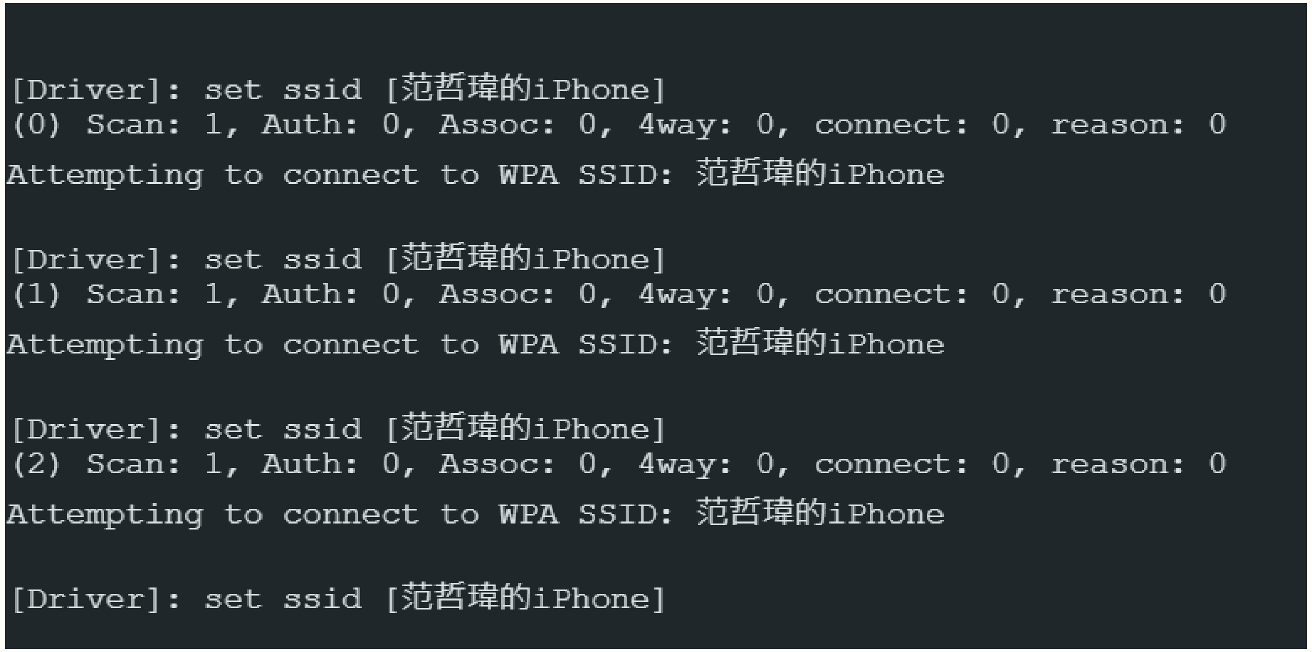
Check the WiFi connection repeatedly



```
while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    status = WiFi.begin(ssid, pass);

    // wait 2 seconds for connection:
    delay(2000);
}
```

2.2 LoopPostProcessing

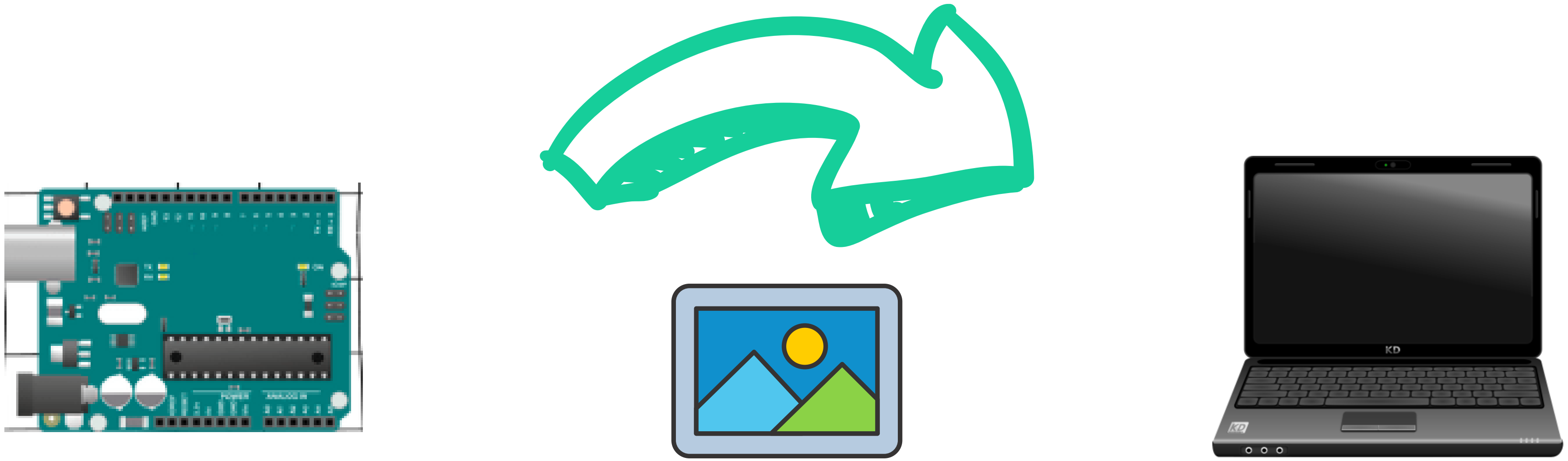
Check the WiFi connection repeatedly



```
[Driver]: set ssid [范哲瑋的iPhone]
(0) Scan: 1, Auth: 0, Assoc: 0, 4way: 0, connect: 0, reason: 0
Attempting to connect to WPA SSID: 范哲瑋的iPhone

[Driver]: set ssid [范哲瑋的iPhone]
(1) Scan: 1, Auth: 0, Assoc: 0, 4way: 0, connect: 0, reason: 0
Attempting to connect to WPA SSID: 范哲瑋的iPhone

[Driver]: set ssid [范哲瑋的iPhone]
(2) Scan: 1, Auth: 0, Assoc: 0, 4way: 0, connect: 0, reason: 0
Attempting to connect to WPA SSID: 范哲瑋的iPhone

[Driver]: set ssid [范哲瑋的iPhone]
```

2.2 LoopPostProcessing

RTSP-Real Time Streaming Protocol



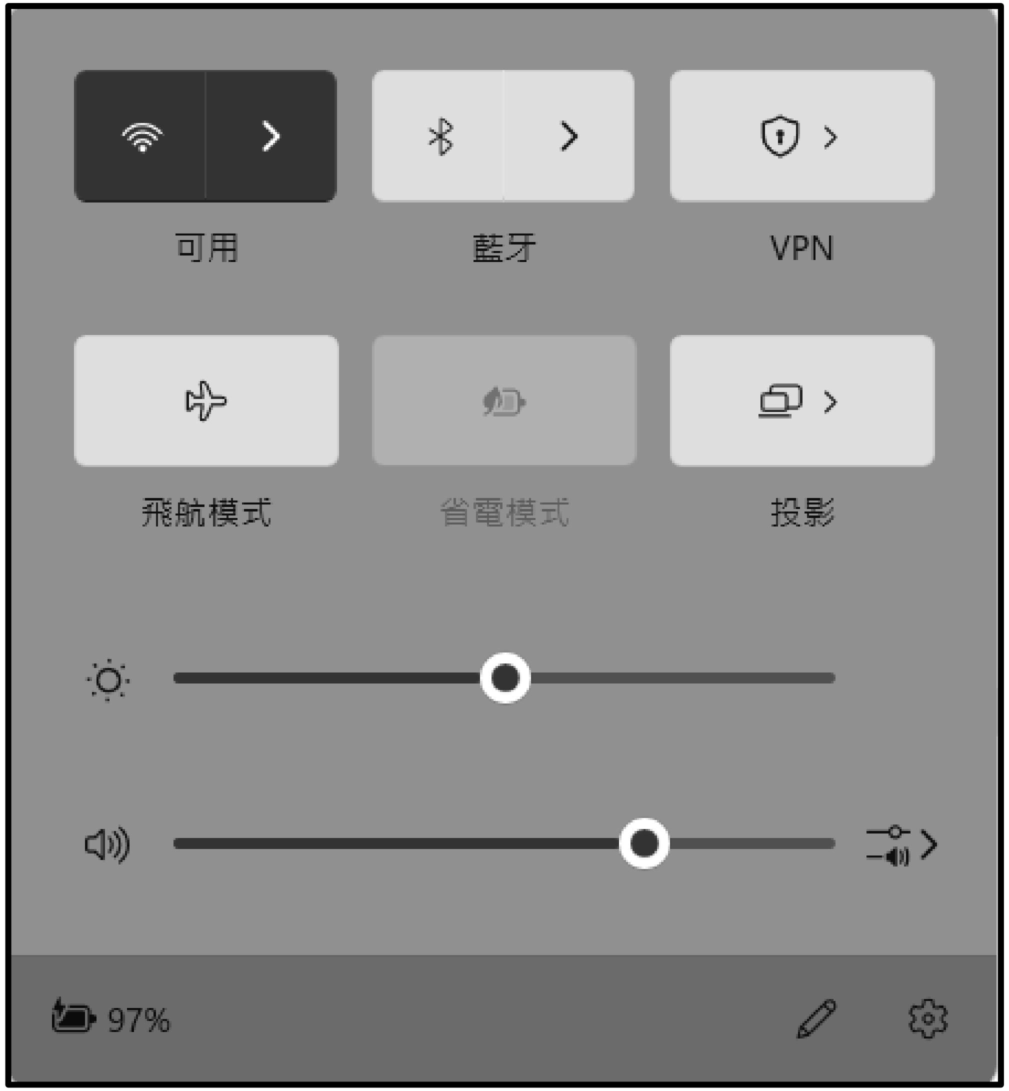
RTSP-Real Time Streaming Protocol

- A network application protocol specifically designed for entertainment and communication systems to control streaming media servers.

2.2 LoopPostProcessing

Step 1.

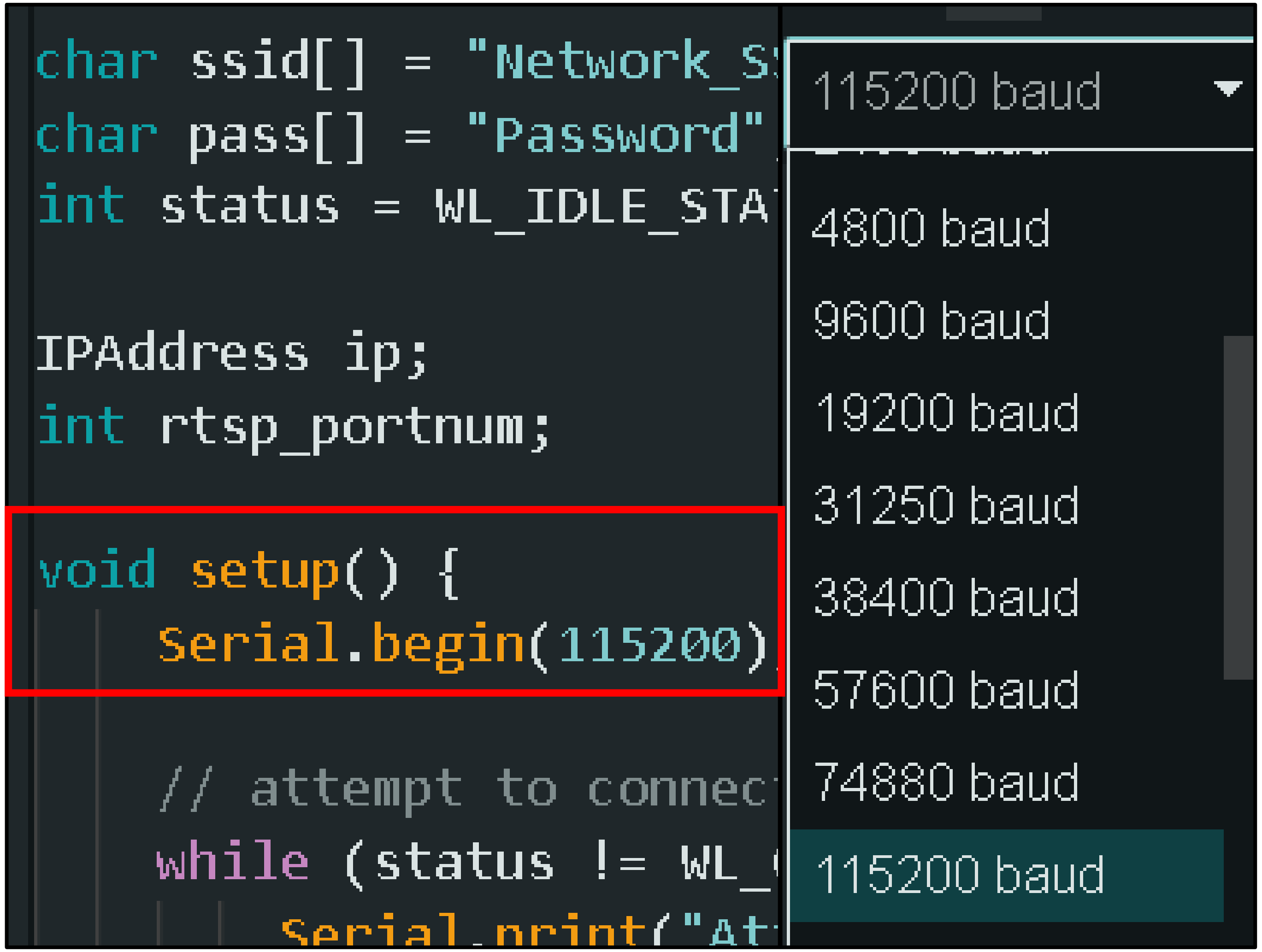
Make sure that Computer and the AMB82 connect to the same WiFi network.



2.2 LoopPostProcessing

Step 2.

Set the Baud of the serial monitor to 115200. As same as in the code.



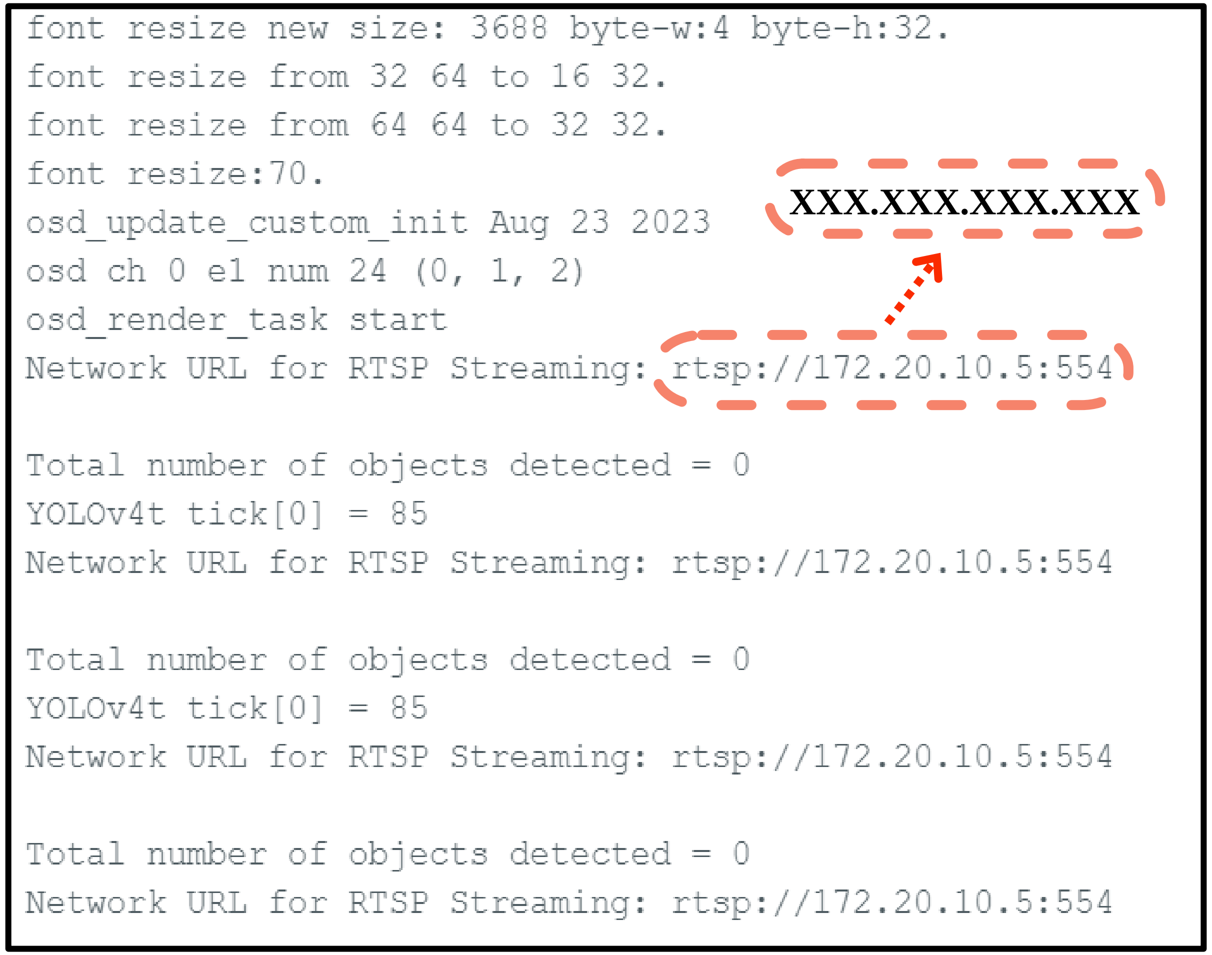
```
char ssid[] = "Network_S
char pass[] = "Password"
int status = WL_IDLE_STA

IPAddress ip;
int rtsp_portnum;

void setup() {
    Serial.begin(115200)

    // attempt to connec
    while (status != WL_
        Serial.print("At
```

115200 baud

2.2 LoopPostProcessing

Step 3.

Press the reset button on the AMB82 and find the IP address in serial monitor. Then copy it.



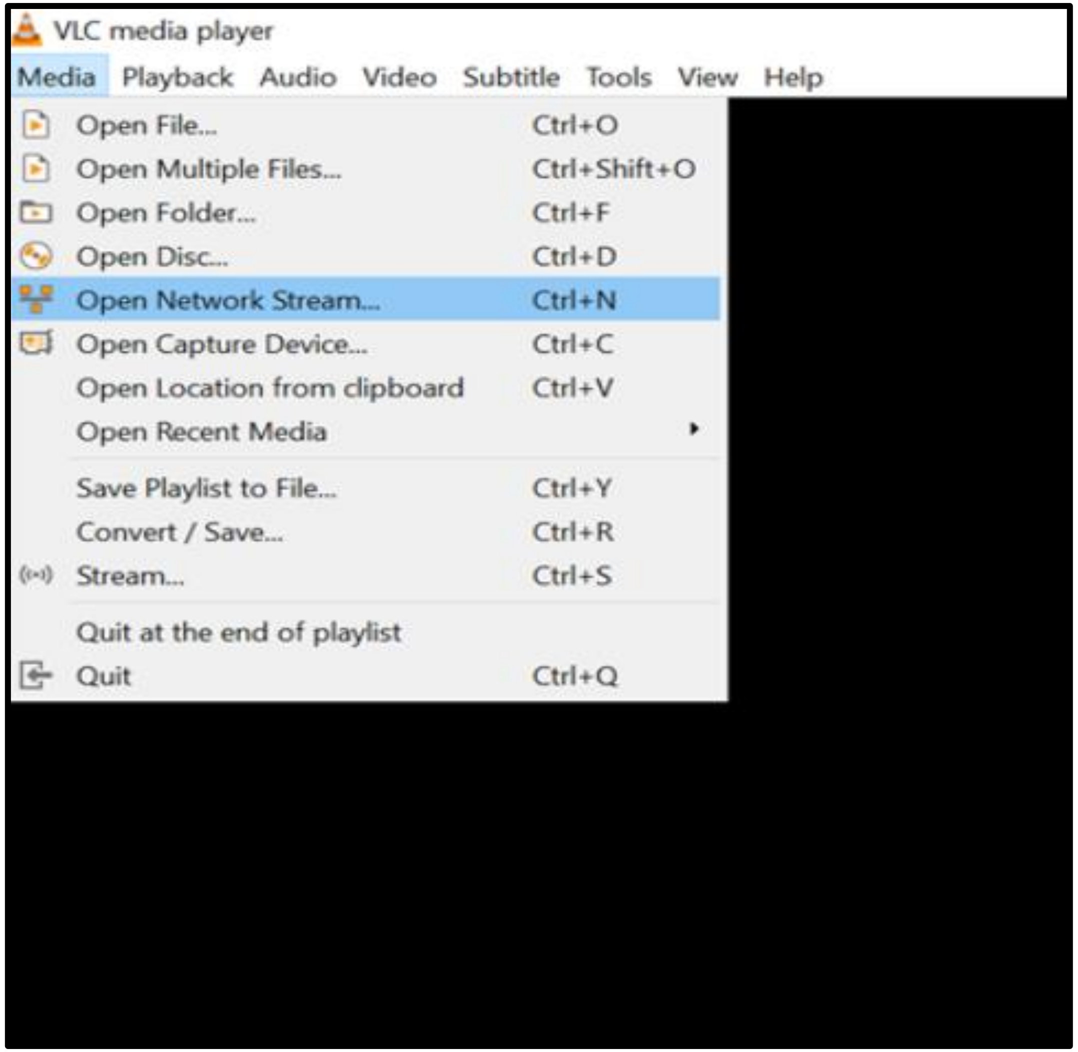
```
font resize new size: 3688 byte-w:4 byte-h:32.
font resize from 32 64 to 16 32.
font resize from 64 64 to 32 32.
font resize:70.
osd_update_custom_init Aug 23 2023
osd ch 0 el num 24 (0, 1, 2)
osd_render_task start
Network URL for RTSP Streaming: rtsp://172.20.10.5:554

Total number of objects detected = 0
YOLOv4t tick[0] = 85
Network URL for RTSP Streaming: rtsp://172.20.10.5:554

Total number of objects detected = 0
YOLOv4t tick[0] = 85
Network URL for RTSP Streaming: rtsp://172.20.10.5:554

Total number of objects detected = 0
Network URL for RTSP Streaming: rtsp://172.20.10.5:554
```

2.2 LoopPostProcessing

Step 4.

Follow the path below in VLC media player to start streaming.

1. **Media**
2. **Open Network Stream**



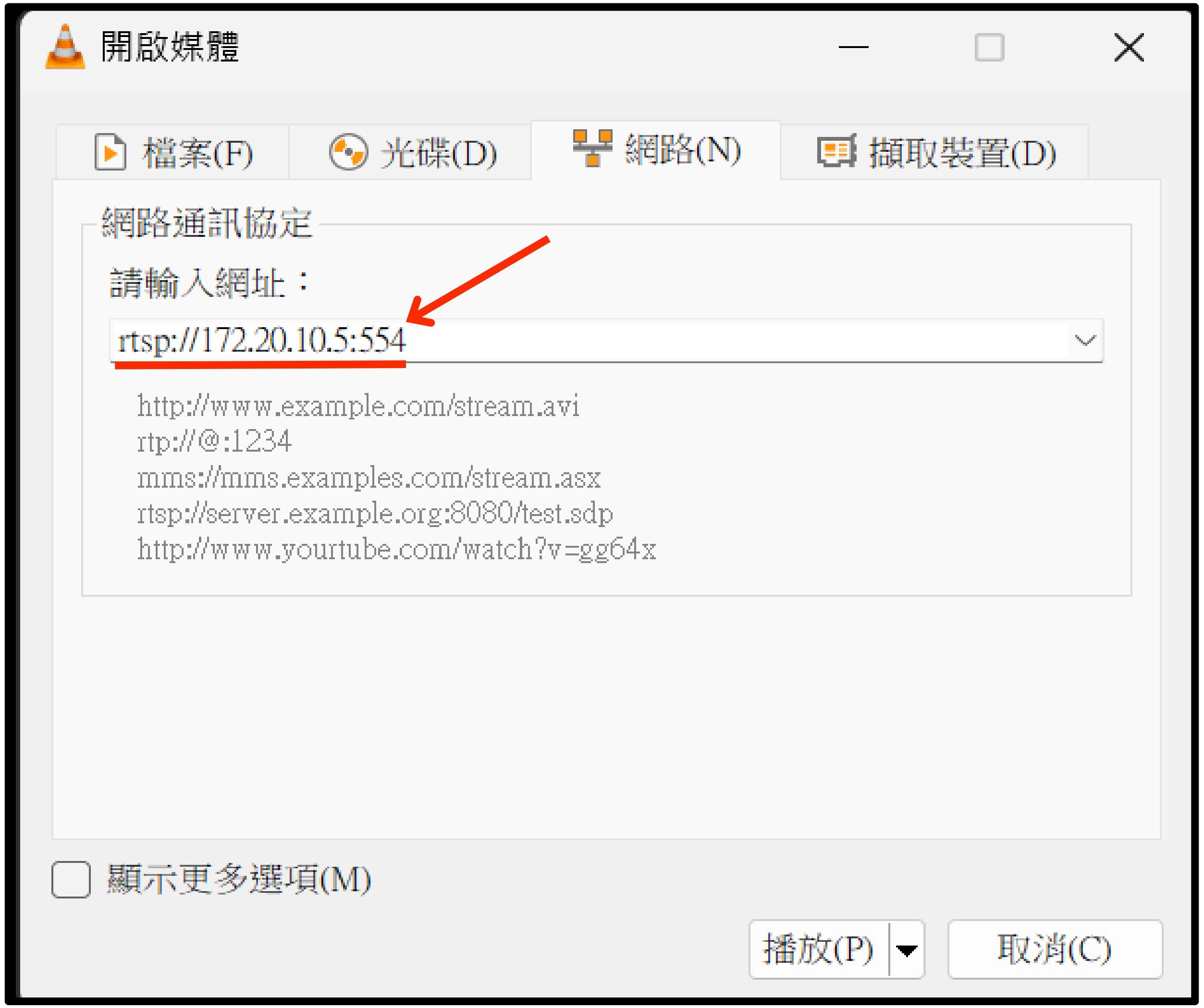
2.2 LoopPostProcessing

Step 5.

Past the copied IP address to VLC.

It must follow the format below.

(rtsp://XXX.XXX.XXX.XXX:554)



rtsp://172.20.10.5:554

http://www.example.com/stream.avi
rtp://@:1234
mms://mms.examples.com/stream.asx
rtsp://server.example.org:8080/test.sdp
http://www.yourtube.com/watch?v=gg64x

LoopPostProcessing

Mask

What does PostProcessing do?

- **Purpose: Post-processing of detected results**
- **Use Mask to remove unnecessary parts.**

Application of Motion Detection?

- **Smart Home：**
  - **Turn on the lights automatically**
- **Outdoor environment monitoring：**
  - **Motion detection take place in parking lots、factories etc.**
- **Office：**
  - **Marking abnormal behavior**

Application of the Massk

- **Perform motion detection on specific areas and ignore other areas. For example, only care about the dynamic changes of doorways or windows, but not the changes in the background.**
- **A private desk in the office or a private area at home. Set a mask to exclude these areas from the motion detection range.**

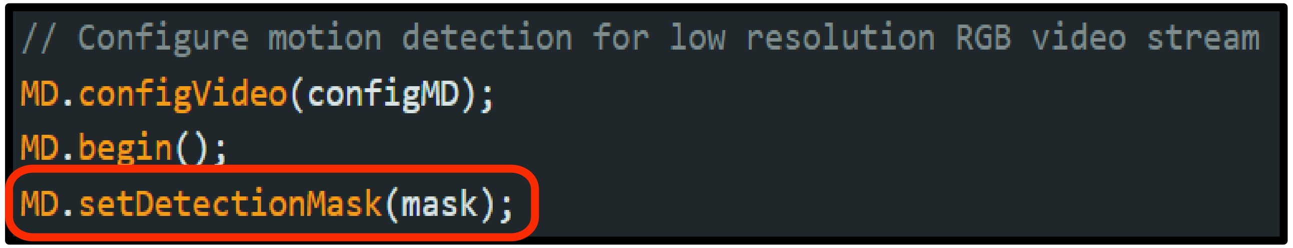
2.2 LoopPostProcessing

Programming

Add the code marked at below to the Arduino code.

In order to use the default mask for application.

Results will look like



```
// Configure motion detection for low resolution RGB video stream
MD.configVideo(configMD);
MD.begin();
MD.setDetectionMask(mask);
```

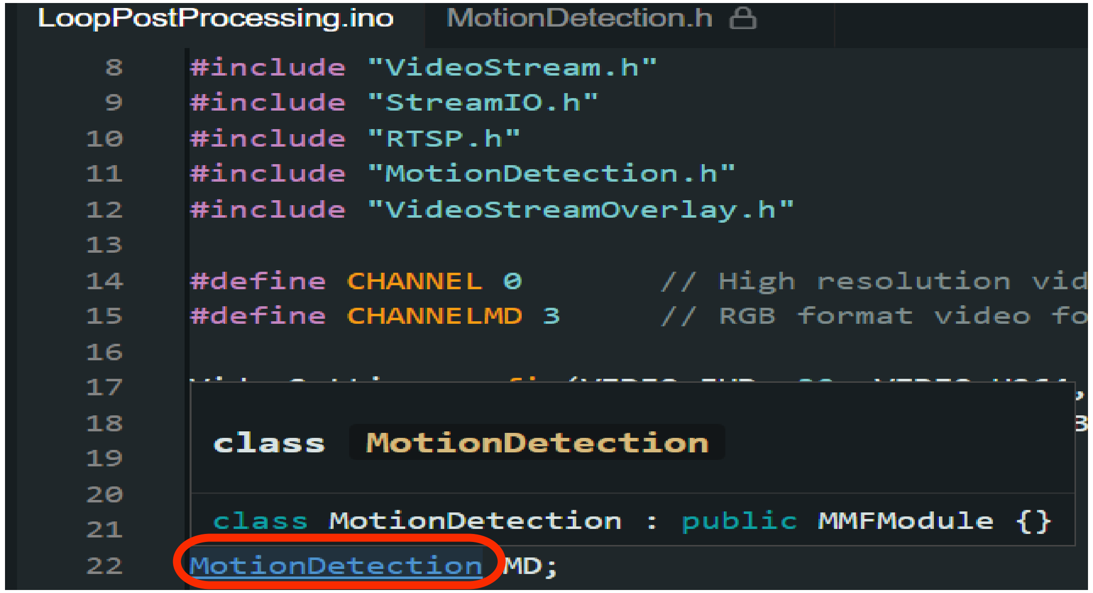
2.2 LoopPostProcessing

Programming

Keep pressing the Ctrl button, and then click the MotionDetection.

Then you will see the default mask format. Shown as below.

Results will look like



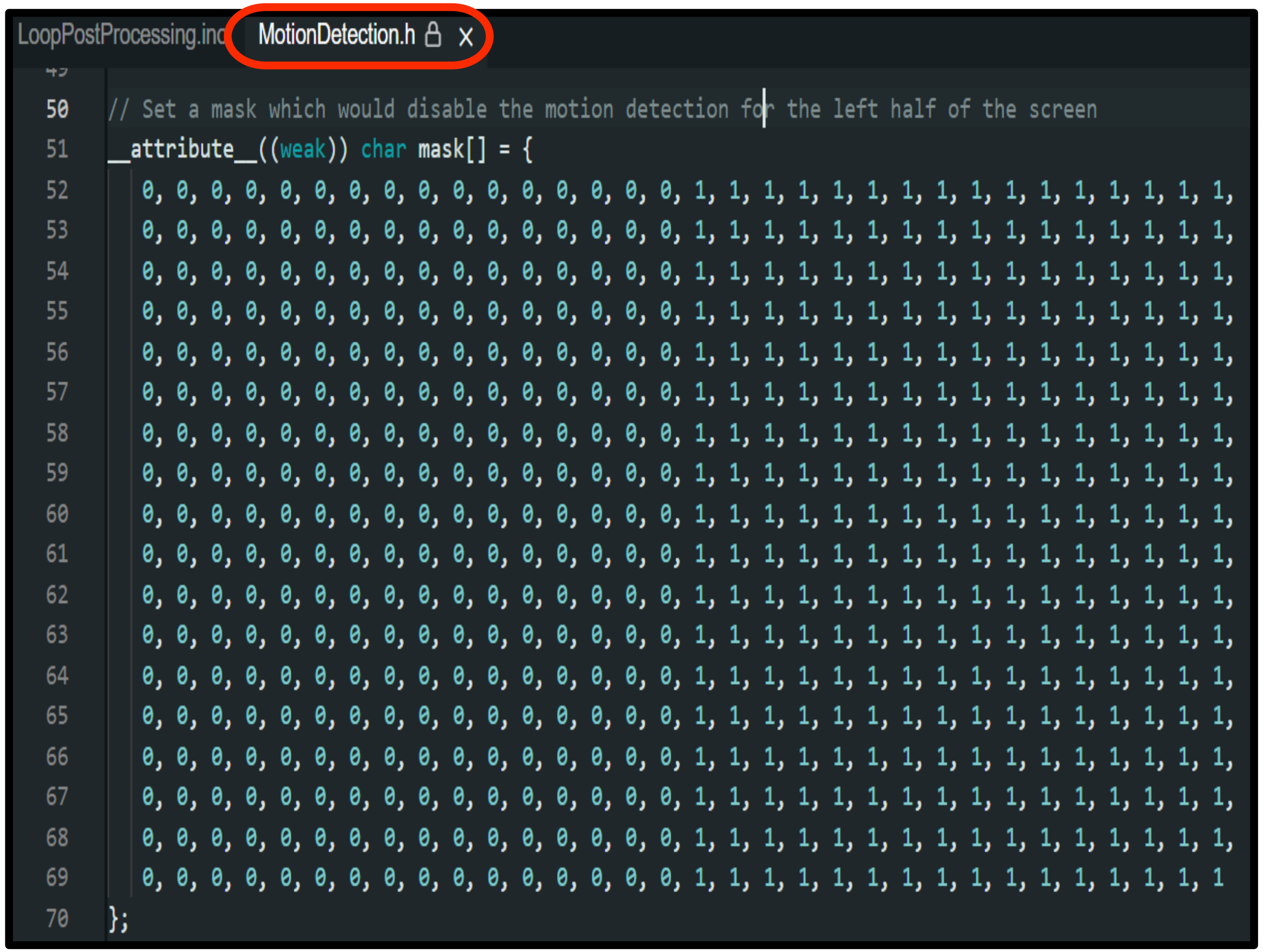
```
#include "VideoStream.h"
#include "StreamIO.h"
#include "RTSP.h"
#include "MotionDetection.h"
#include "VideoStreamOverlay.h"

#define CHANNEL 0        // High resolution vid
#define CHANNELMD 3      // RGB format video fo

MotionDetection MD;
```

```
class MotionDetection

class MotionDetection : public MMFModule {}
```

2.2 LoopPostProcessing

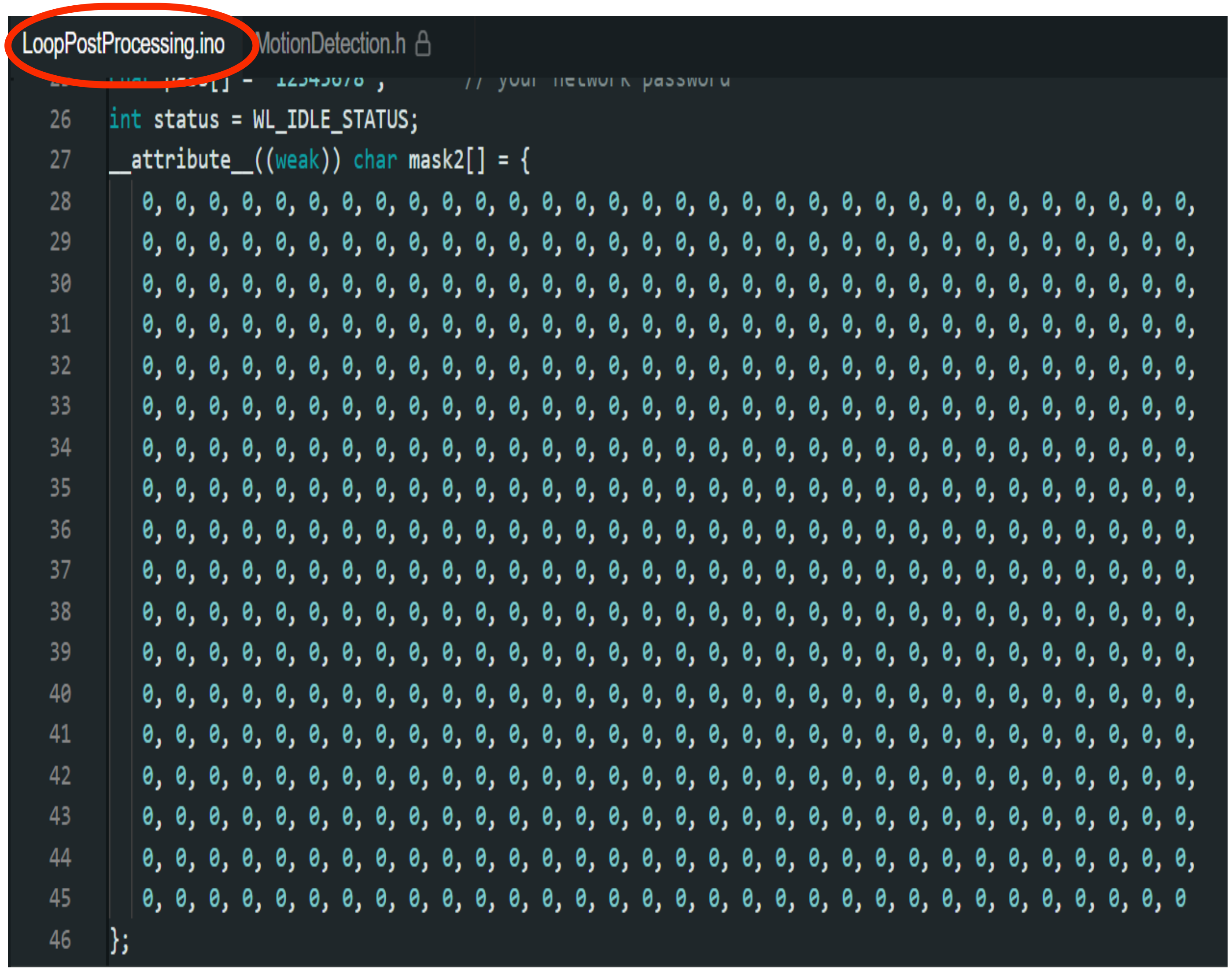
The default mask can be seen in .h file.

This file is unmodifiable.



```
// Set a mask which would disable the motion detection for the left half of the screen
__attribute__((weak)) char mask[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
};
```

2.2 LoopPostProcessing



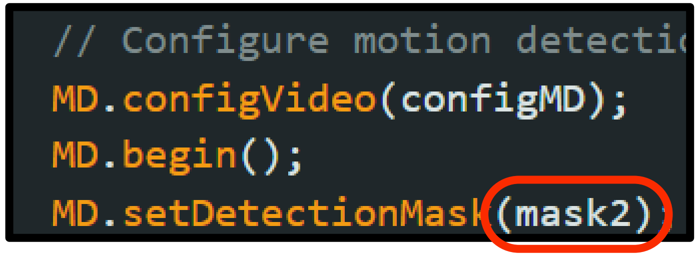
```
char pass[] = "12345678";        // your network password
int status = WL_IDLE_STATUS;
__attribute__((weak)) char mask2[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};
```

Copy the default mask from the .h file,

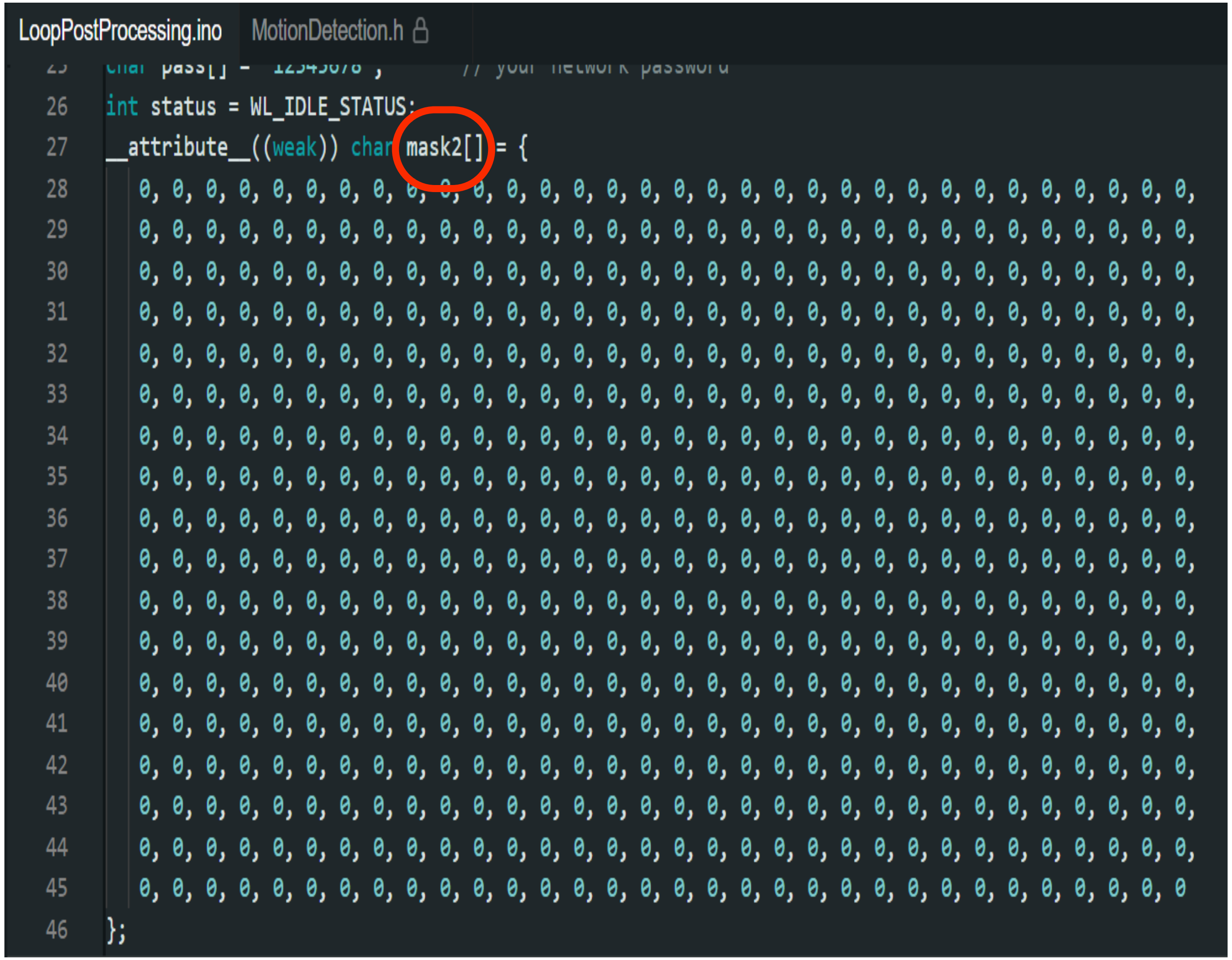
then past it onto the .ino file.

2.2 LoopPostProcessing

Change the name of mask to set the customized mask.



```
// Configure motion detecti
MD.configVideo(configMD);
MD.begin();
MD.setDetectionMask(mask2);
```



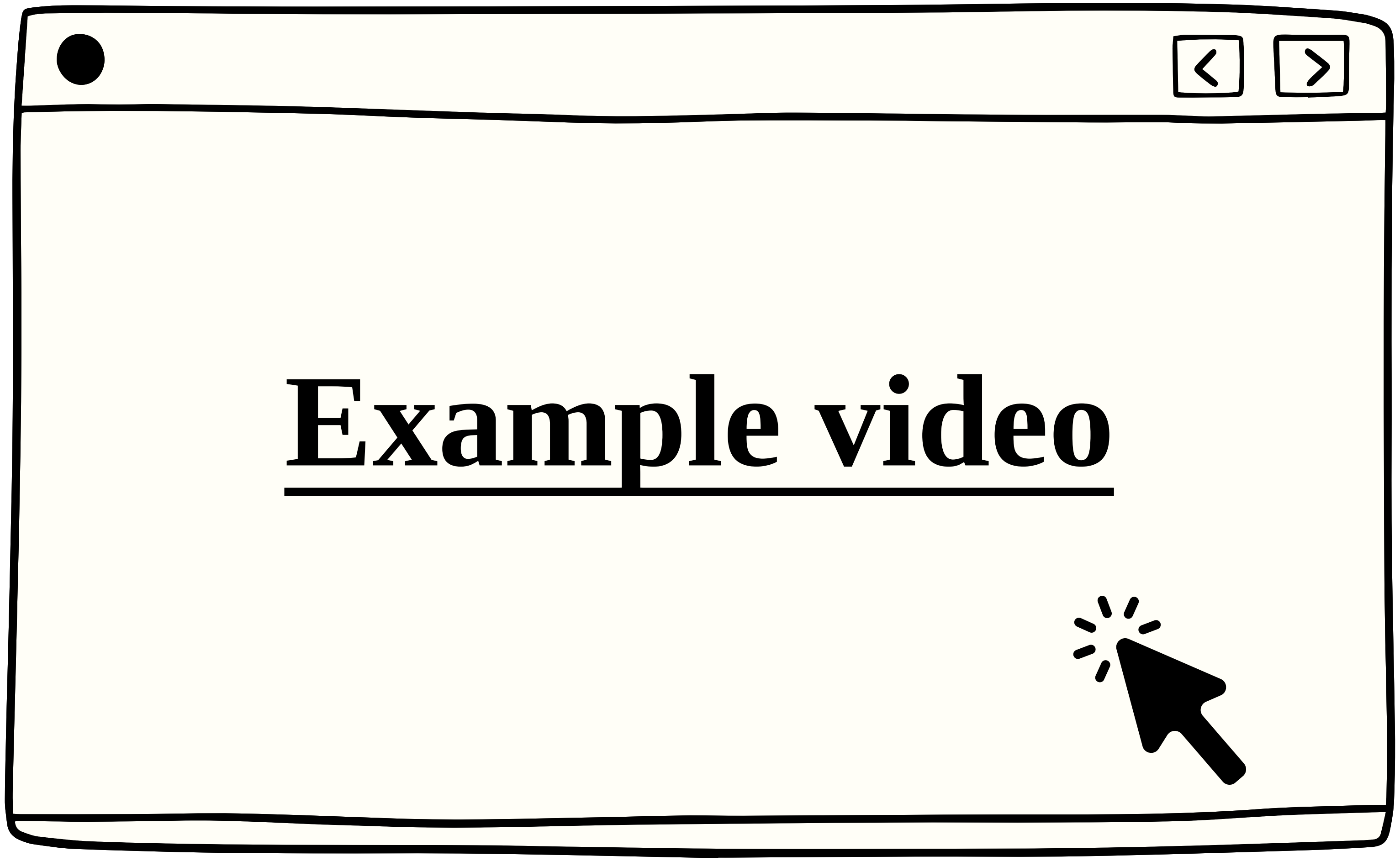
```
char pass[] = "12345678";        // your network password
int status = WL_IDLE_STATUS;
__attribute__((weak)) char mask2[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
};
```

2.3 Audio Classification

Application of Audio Classification

- **Smart home：**
  - **Recognize voice commands such as "turn on the lights" and "turn off the lights“**
- **Health care：**
  - **The patient's abnormal breathing sounds, coughing sounds, etc.**

2.3 Audio Classification



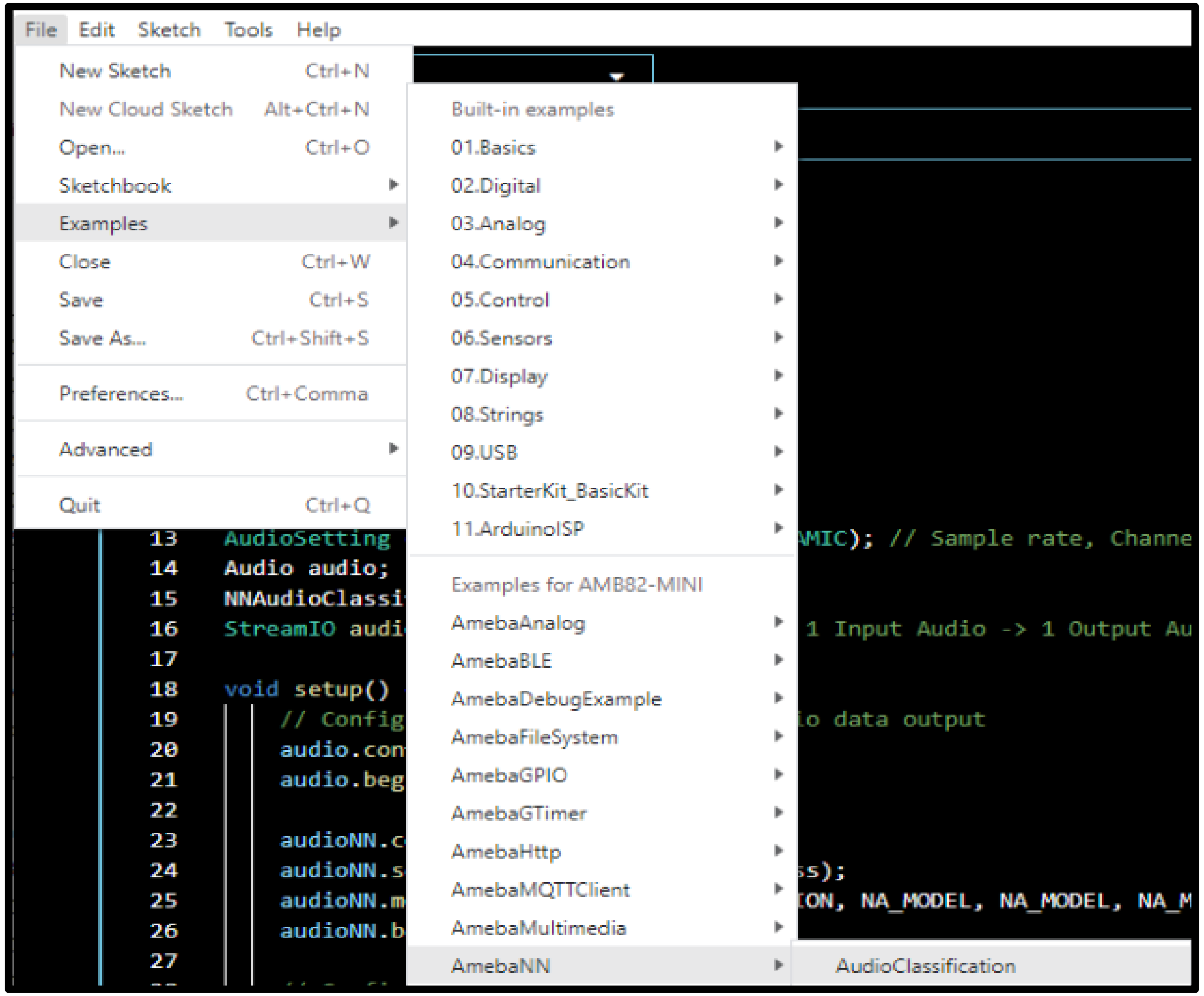
[Example video]

2.3 Audio Classification

Step 1.

Follow the path below in Arduino IDE to open the example.

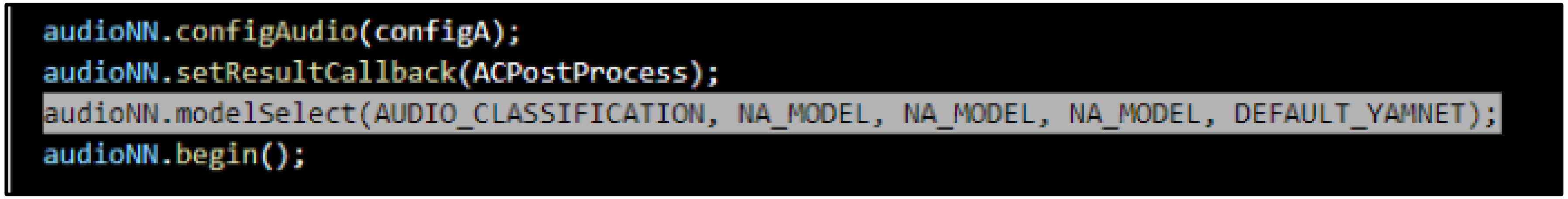
1. **File**
2. **Examples**
3. **AmebaNN**
4. **AudioClassification**



```
AudioSetting                              MIC); // Sample rate, Channe
Audio audio;
NNAudioClassi
StreamIO audi                              1 Input Audio -> 1 Output Au

void setup()
    // Config                              io data output
    audio.con
    audio.beg

    audioNN.c
    audioNN.s                              ss);
    audioNN.m                              ION, NA_MODEL, NA_MODEL, NA_M
    audioNN.b

```

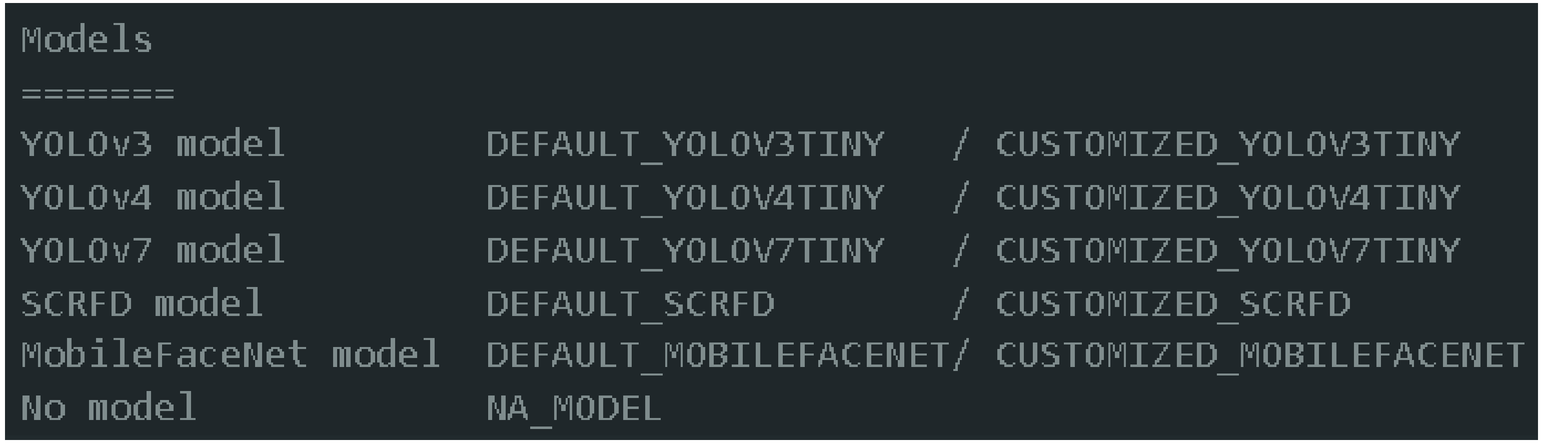
2.3 Audio Classification

Step 2. Model choosing(optional)



```
audioNN.configAudio(configA);
audioNN.setResultCallback(ACPostProcess);
audioNN.modelSelect(AUDIO_CLASSIFICATION, NA_MODEL, NA_MODEL, NA_MODEL, DEFAULT_YAMNET);
audioNN.begin();
```

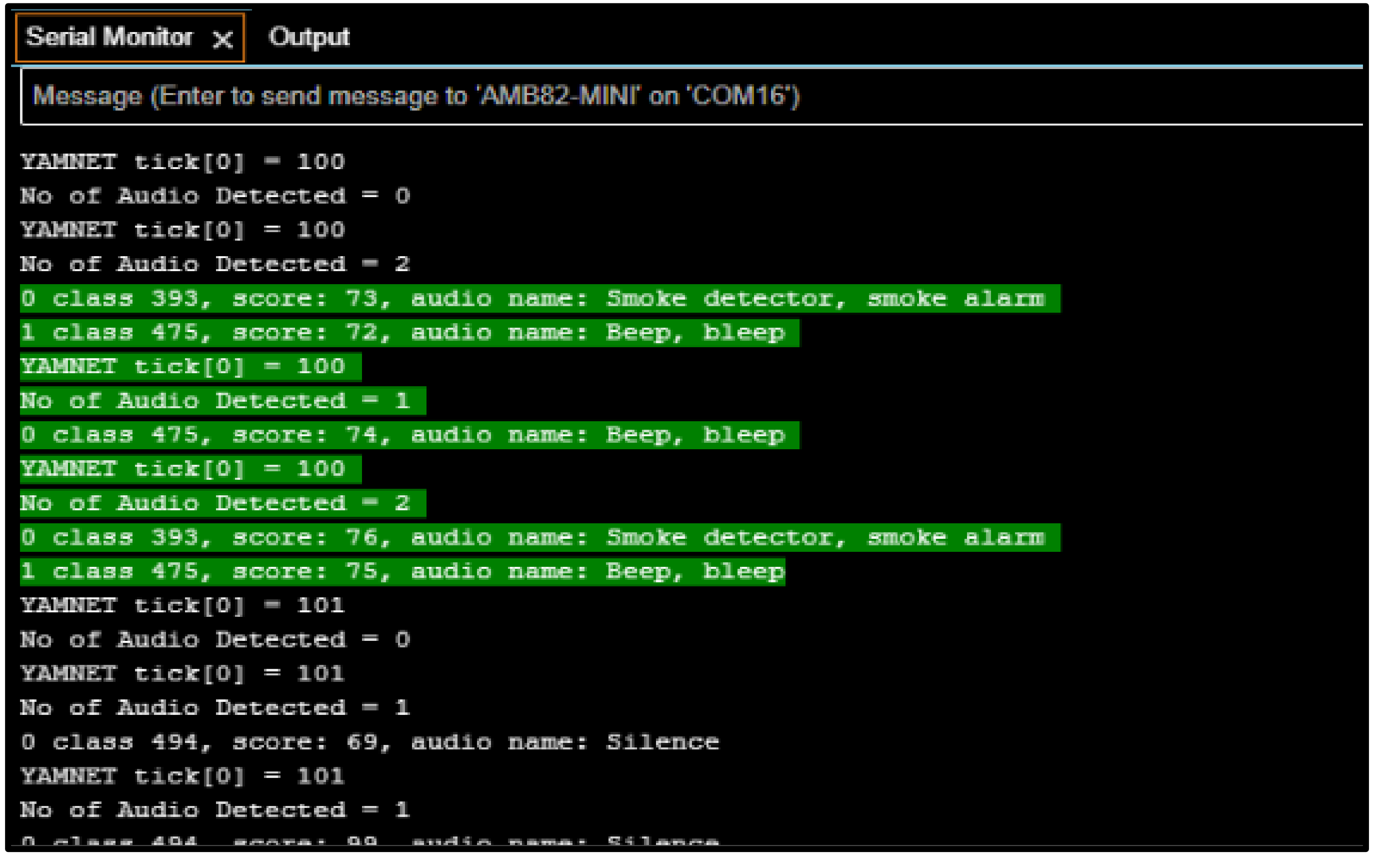
List of models for different tasks



```
Models
=======
YOLOv3 model         DEFAULT_YOLOV3TINY   / CUSTOMIZED_YOLOV3TINY
YOLOv4 model         DEFAULT_YOLOV4TINY   / CUSTOMIZED_YOLOV4TINY
YOLOv7 model         DEFAULT_YOLOV7TINY   / CUSTOMIZED_YOLOV7TINY
SCRFD model          DEFAULT_SCRFD        / CUSTOMIZED_SCRFD
MobileFaceNet model  DEFAULT_MOBILEFACENET/ CUSTOMIZED_MOBILEFACENET
No model             NA_MODEL
```

2.3 Audio Classification

Results: Observe the detected sound category in Serial Monitor.



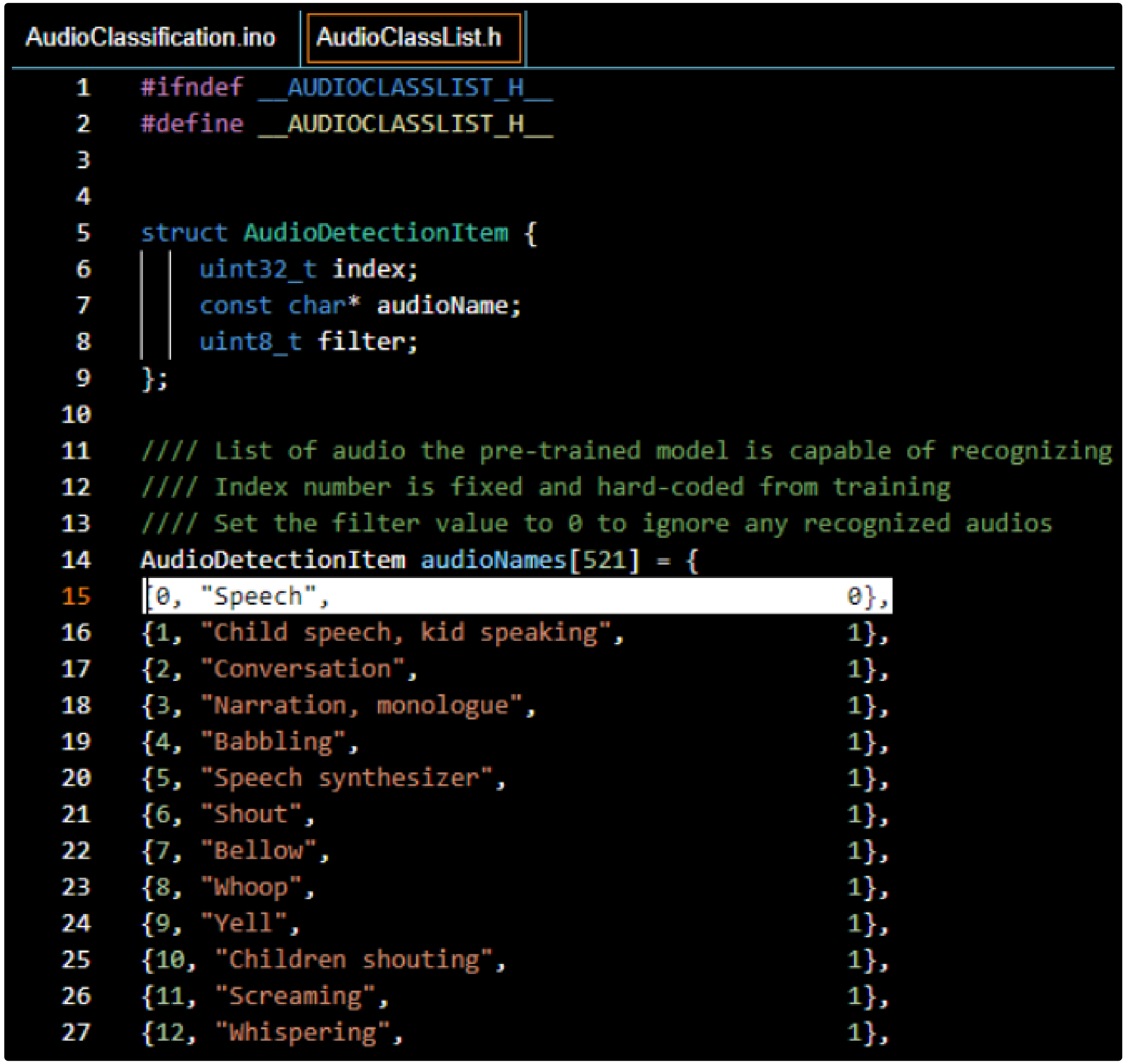
Message (Enter to send message to 'AMB82-MINI' on 'COM16')

```
YAMNET tick[0] = 100
No of Audio Detected = 0
YAMNET tick[0] = 100
No of Audio Detected = 2
0 class 393, score: 73, audio name: Smoke detector, smoke alarm
1 class 475, score: 72, audio name: Beep, bleep
YAMNET tick[0] = 100
No of Audio Detected = 1
0 class 475, score: 74, audio name: Beep, bleep
YAMNET tick[0] = 100
No of Audio Detected = 2
0 class 393, score: 76, audio name: Smoke detector, smoke alarm
1 class 475, score: 75, audio name: Beep, bleep
YAMNET tick[0] = 101
No of Audio Detected = 0
YAMNET tick[0] = 101
No of Audio Detected = 1
0 class 494, score: 69, audio name: Silence
YAMNET tick[0] = 101
No of Audio Detected = 1
0 class 494, score: 88, audio name: Silence
```

2.3 Audio Classification

- In total, the pre-trained model can recognize **521 different types of audio**.

- To disable recognition of certain audios, **set filter to 0**.



```
#ifndef __AUDIOCLASSLIST_H__
#define __AUDIOCLASSLIST_H__


struct AudioDetectionItem {
    uint32_t index;
    const char* audioName;
    uint8_t filter;
};

//// List of audio the pre-trained model is capable of recognizing
//// Index number is fixed and hard-coded from training
//// Set the filter value to 0 to ignore any recognized audios
AudioDetectionItem audioNames[521] = {
{0, "Speech",                                  0},
{1, "Child speech, kid speaking",               1},
{2, "Conversation",                             1},
{3, "Narration, monologue",                     1},
{4, "Babbling",                                 1},
{5, "Speech synthesizer",                       1},
{6, "Shout",                                    1},
{7, "Bellow",                                   1},
{8, "Whoop",                                    1},
{9, "Yell",                                     1},
{10, "Children shouting",                       1},
{11, "Screaming",                               1},
{12, "Whispering",                              1},
```

2.3 Audio Classification

Add results display

2.3 Audio Classification

Implementation must include the following three points in the code.

1.At the beginning of the code :

define the PIN

```
int output0 = 0 ;
int output1 = 1 ;
int output2 = 2 ;
int output3 = 3 ;
int output4 = 4 ;
```

2.3 Audio Classification

Implementation must include the following three points in the code.

2. Add the following in the function void setup():

Assign the output to the defined pin

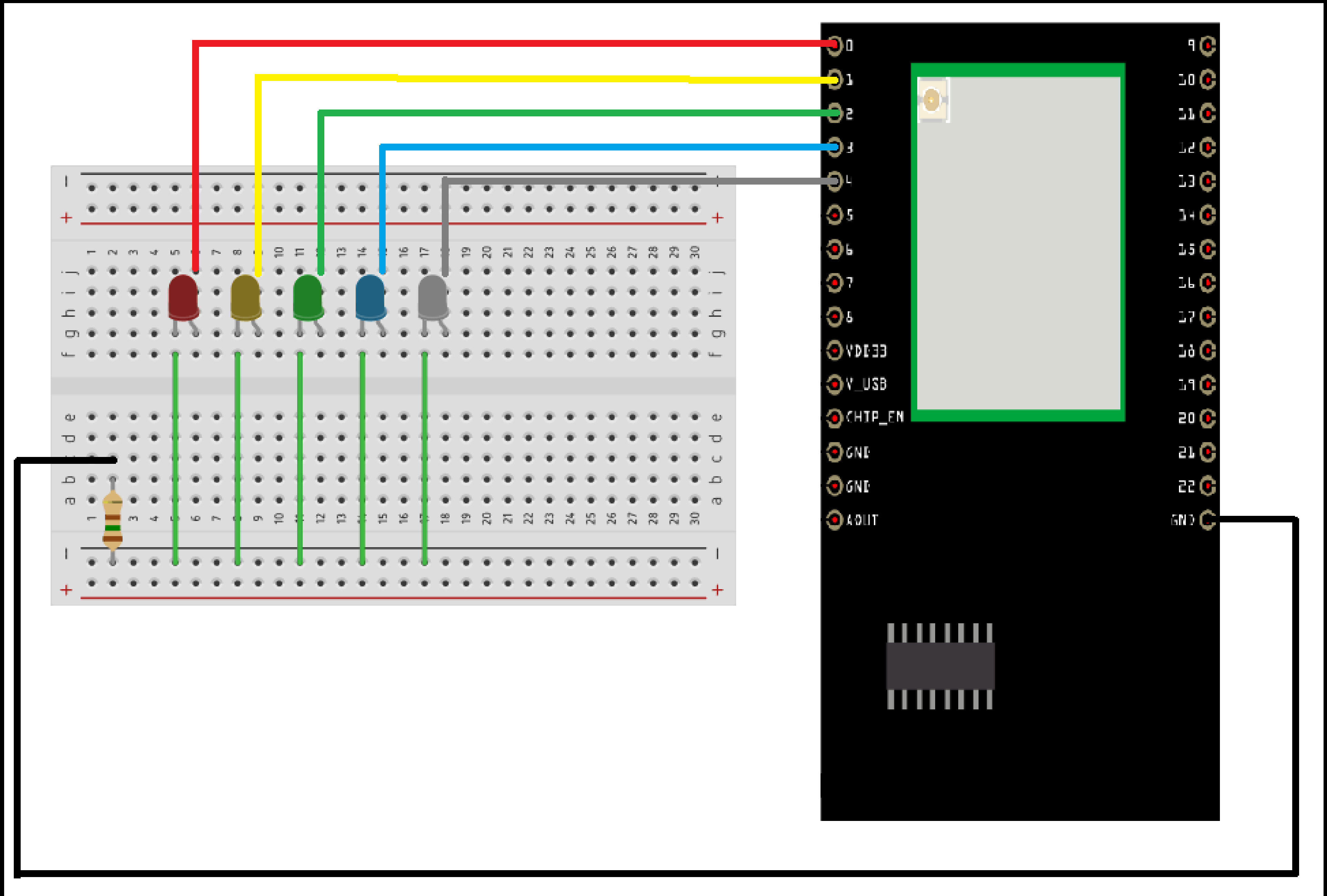
```
pinMode(output0, OUTPUT);
pinMode(output1, OUTPUT);
pinMode(output2, OUTPUT);
pinMode(output3, OUTPUT);
pinMode(output4, OUTPUT);
```

3. Add the following in the function void loop(): Determine which finger was detected

```
 if(obj_type==0) //speech
                {
                  digitalWrite(output0, HIGH);
                  delay(1000);
                  digitalWrite(output0, LOW);
                  delay(1000);
                }

                else if(obj_type==1) //child
speech
                {
                  digitalWrite(output1, HIGH);
                  delay(1000);
                  digitalWrite(output1, LOW);
                  delay(1000);
                }
```

```
else if(obj_type==2)//conversation
{
  digitalWrite(output2, HIGH);
  delay(1000);
  digitalWrite(output2, LOW);
  delay(1000);
}
else if(obj_type==3) //Narration
{
  digitalWrite(output3, HIGH);
  delay(1000);
  digitalWrite(output3, LOW);
  delay(1000);
}
else if(obj_type==4) //Babbling
{
  digitalWrite(output4, HIGH);
  delay(1000);
  digitalWrite(output4, LOW);
  delay(1000);
}
```

Circuit Diagram



2.4 FaceRecognition

2.4 Face Recognition

Example Video

FaceRecognition Technical basis

1. Face detection：
   - Detecting the face areas in images or videos
2. Features extraction：
   - These features can include the contours of the face, eye position, nose shape, etc.
3. Features matching：
   - The extracted features will be compared with known facial features to evaluate the similarity between the two feature vectors.

Application of FaceRecognition

- Access control system：
  - Home access control system
  - Clock in system in companies or schools
- Security monitoring：
  - Blacklist database combination
  - Identity authentication

2.4 Face Recognition

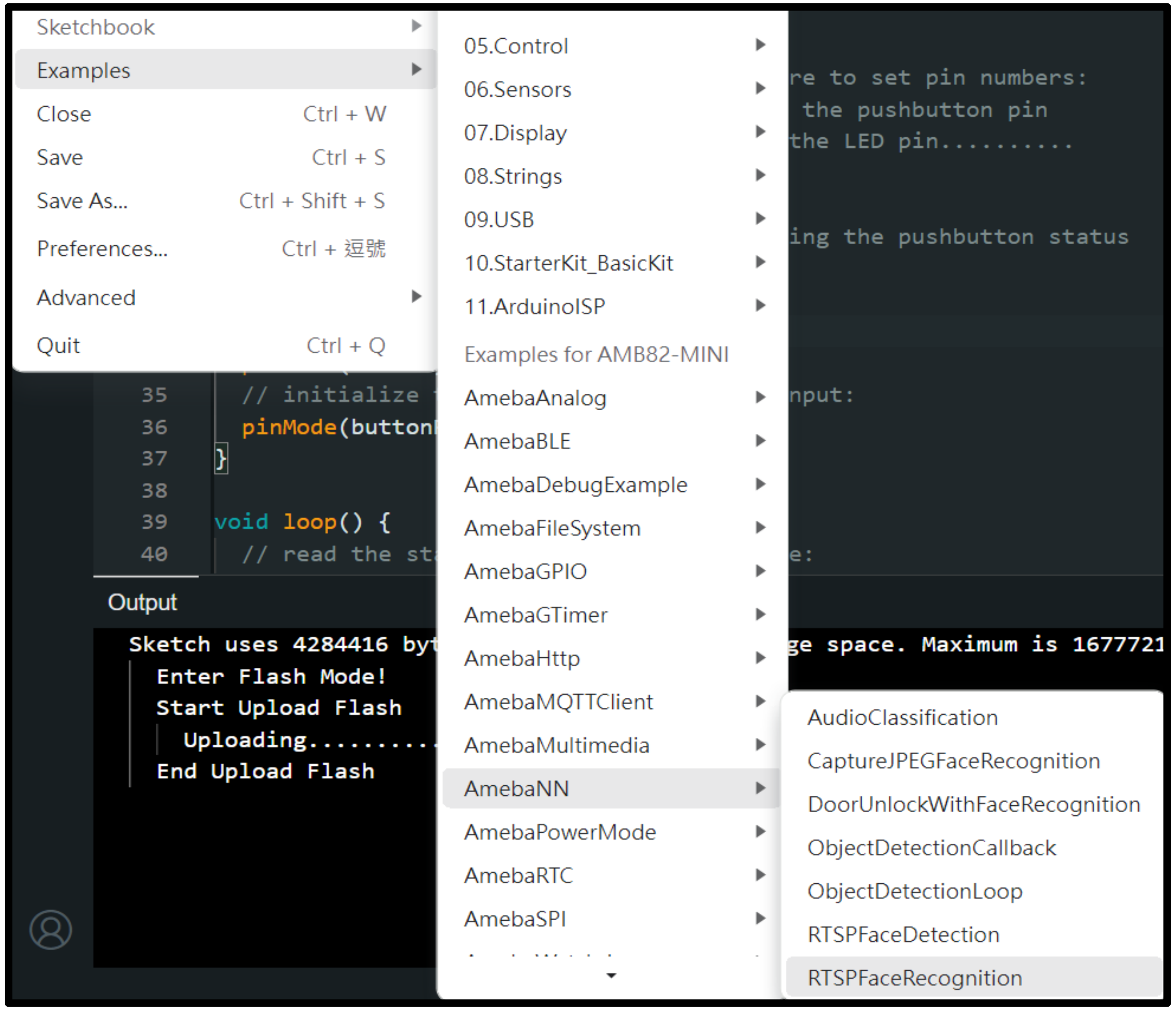
Implementation

2.4 Face Recognition

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaNN**
4. **RTSPFaceRecognition**



```
re to set pin numbers:
 the pushbutton pin
the LED pin.........

ing the pushbutton status

  // initialize                    nput:
  pinMode(button
}

void loop() {
  // read the st                   e:
```

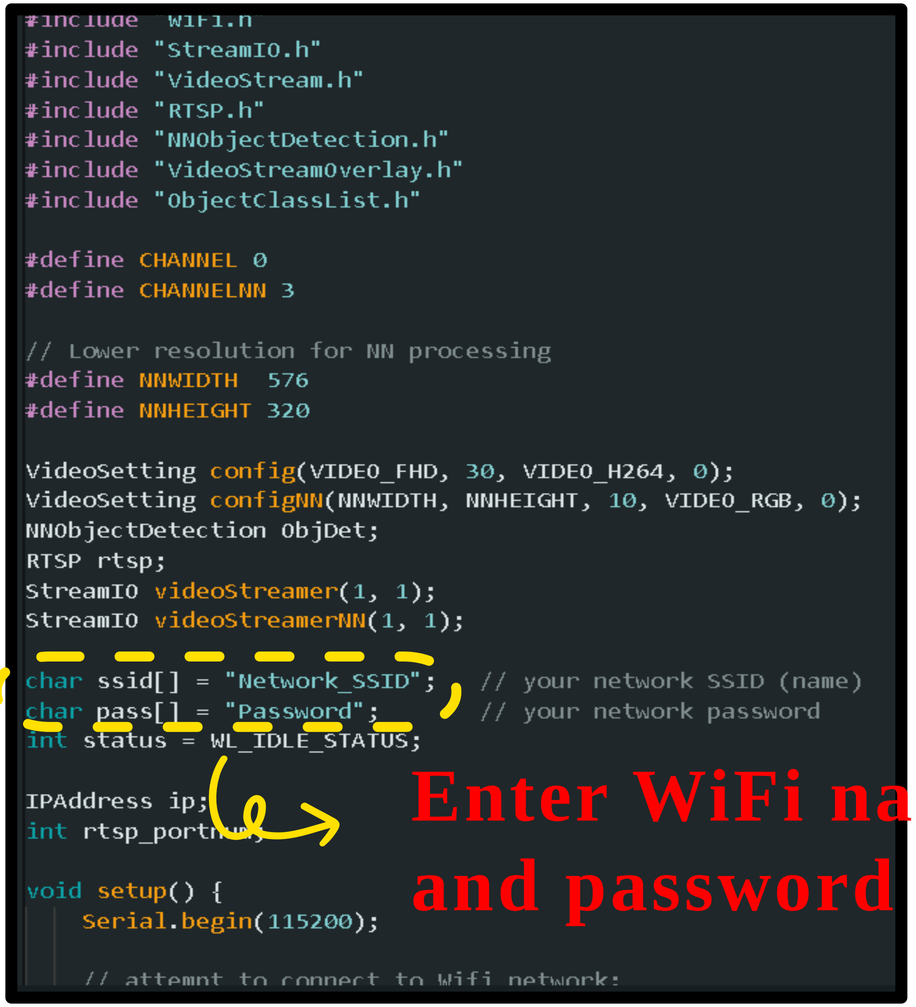
```
Sketch uses 4284416 byt          ge space. Maximum is 1677721
 Enter Flash Mode!
 Start Upload Flash
   Uploading.........
 End Upload Flash
```

2.4 Face Recognition

Step 2.

Enter the WiFi name and password

to the corresponding place in the code.

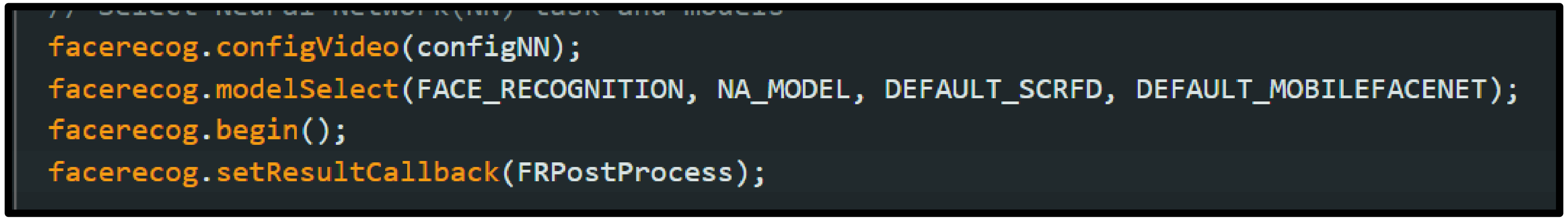


```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNObjectDetection.h"
#include "VideoStreamOverlay.h"
#include "ObjectClassList.h"

#define CHANNEL 0
#define CHANNELNN 3

// Lower resolution for NN processing
#define NNWIDTH  576
#define NNHEIGHT 320

VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
RTSP rtsp;
StreamIO videoStreamer(1, 1);
StreamIO videoStreamerNN(1, 1);

char ssid[] = "Network_SSID";   // your network SSID (name)
char pass[] = "Password";       // your network password
int status = WL_IDLE_STATUS;

IPAddress ip;
int rtsp_portnum;

void setup() {
    Serial.begin(115200);

    // attempt to connect to Wifi network:
```

Enter WiFi name and password

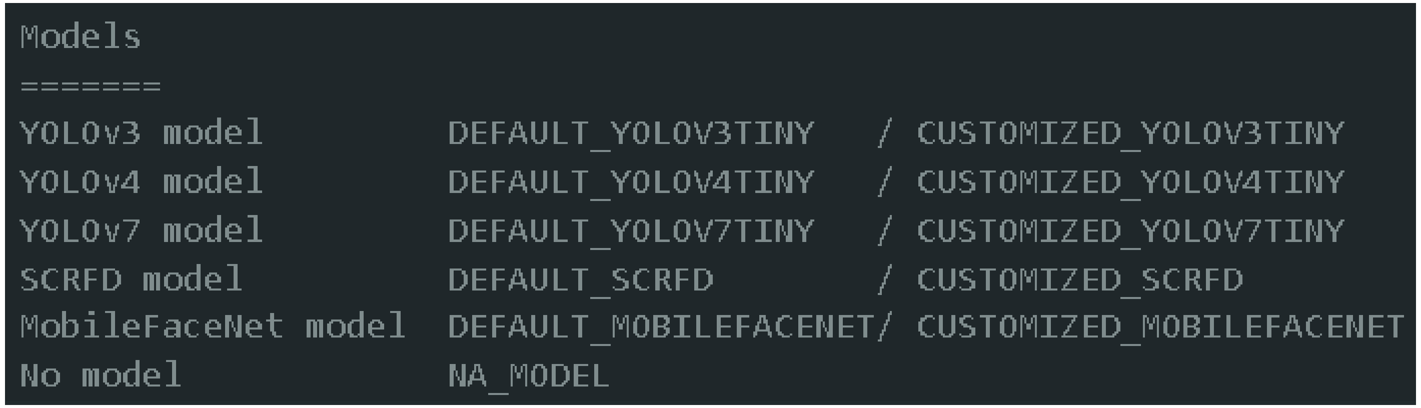
2.4 Face Recognition

Step 3. Model choosing(optional)



```
// Select Neural Network(NN) task and models
facerecog.configVideo(configNN);
facerecog.modelSelect(FACE_RECOGNITION, NA_MODEL, DEFAULT_SCRFD, DEFAULT_MOBILEFACENET);
facerecog.begin();
facerecog.setResultCallback(FRPostProcess);
```

List of models for different tasks



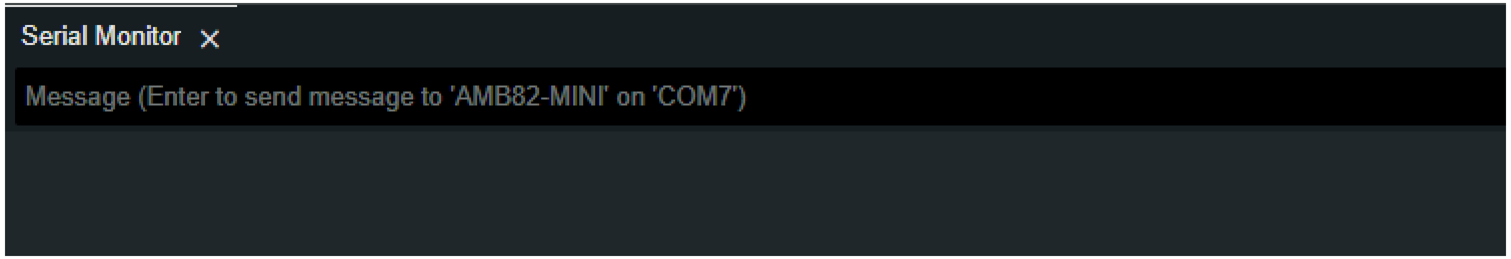
```
Models
=======
YOLOv3 model         DEFAULT_YOLOV3TINY    / CUSTOMIZED_YOLOV3TINY
YOLOv4 model         DEFAULT_YOLOV4TINY    / CUSTOMIZED_YOLOV4TINY
YOLOv7 model         DEFAULT_YOLOV7TINY    / CUSTOMIZED_YOLOV7TINY
SCRFD model          DEFAULT_SCRFD         / CUSTOMIZED_SCRFD
MobileFaceNet model  DEFAULT_MOBILEFACENET/ CUSTOMIZED_MOBILEFACENET
No model             NA_MODEL
```

2.4 Face Recognition

DEMO

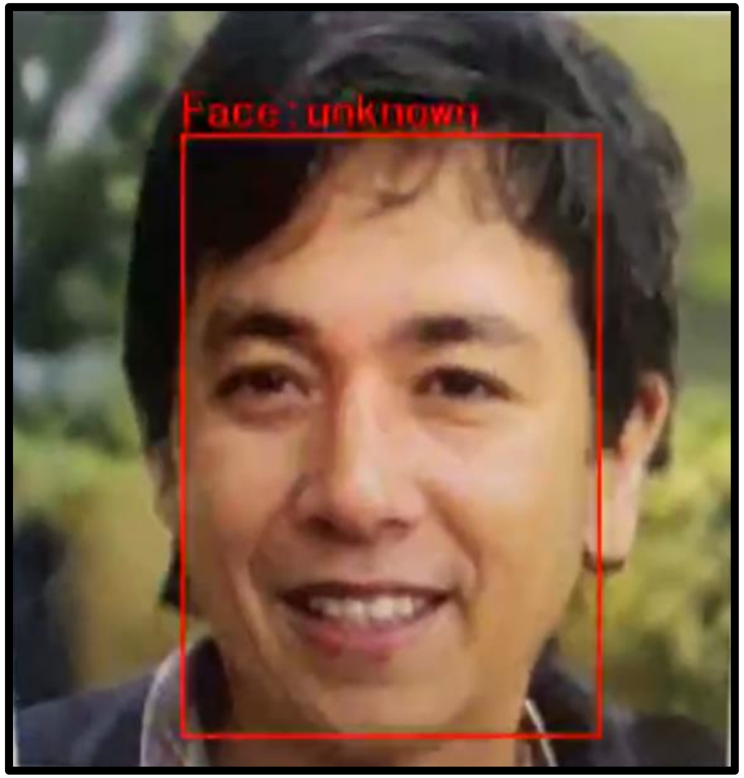
2.4 Face Recognition

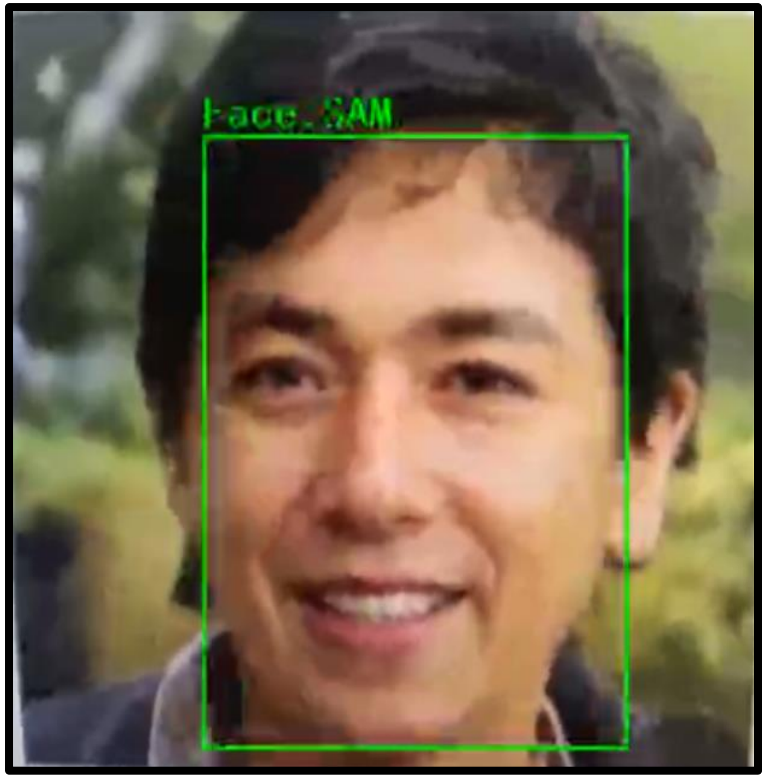
All subsequent operations will be performed in the Serial monitor message box. Shown as below.



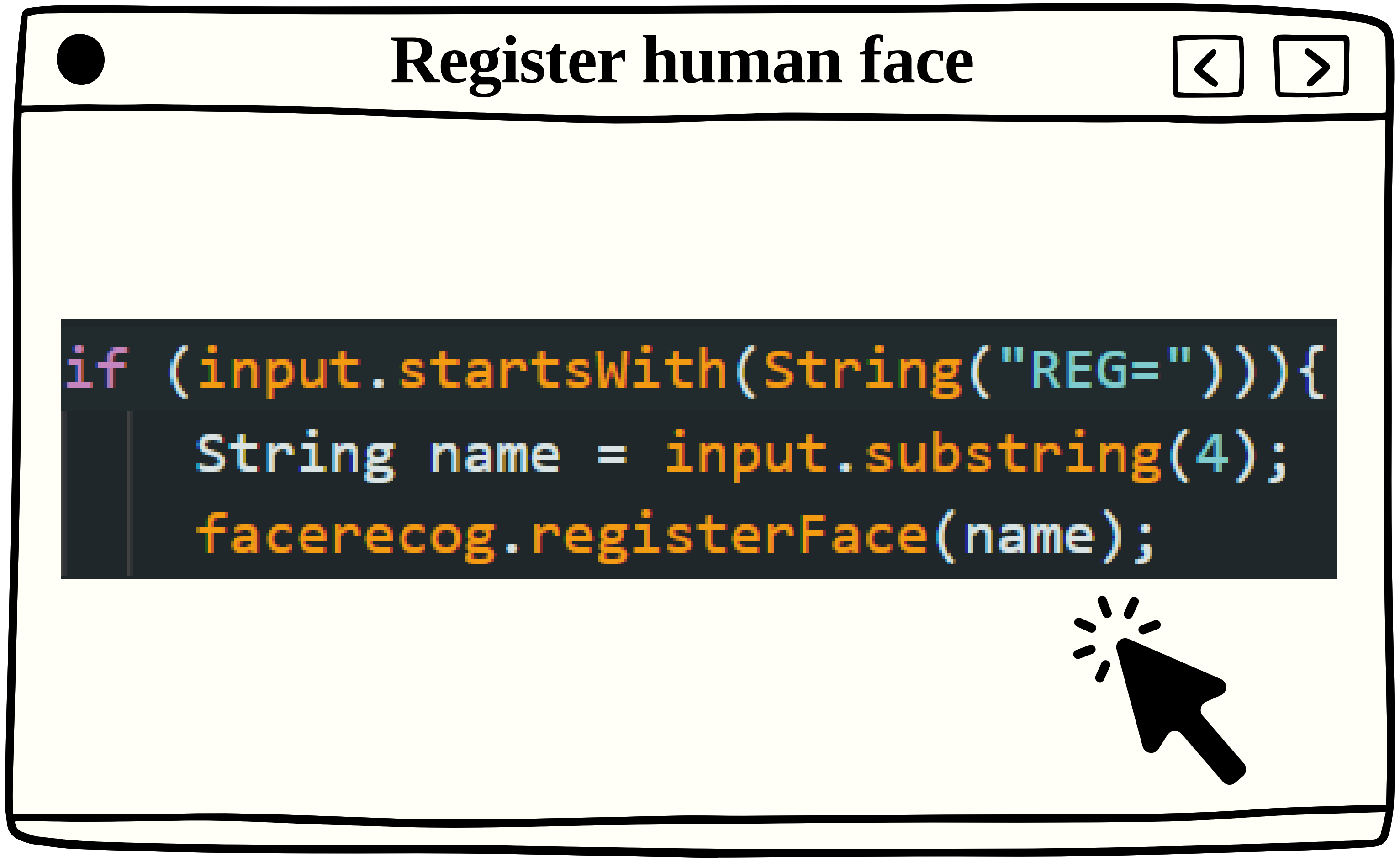
Message (Enter to send message to 'AMB82-MINI' on 'COM7')

2.4 Face Recognition





2.4 Face Recognition

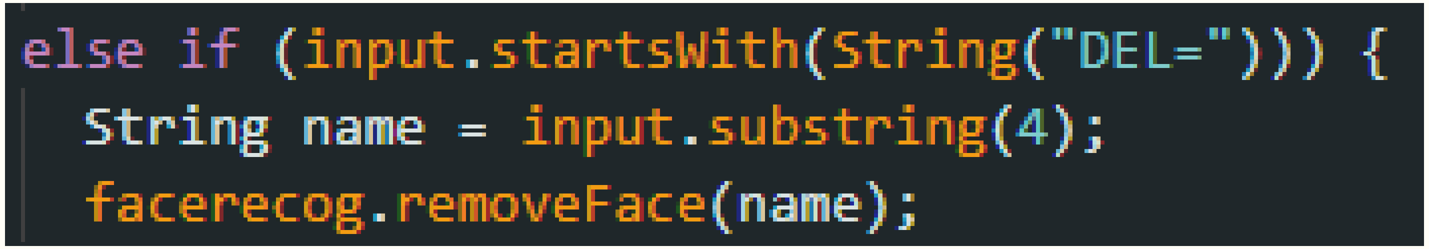


Register human face

```
if (input.startsWith(String("REG="))){
    String name = input.substring(4);
    facerecog.registerFace(name);
```

2.4 Face Recognition

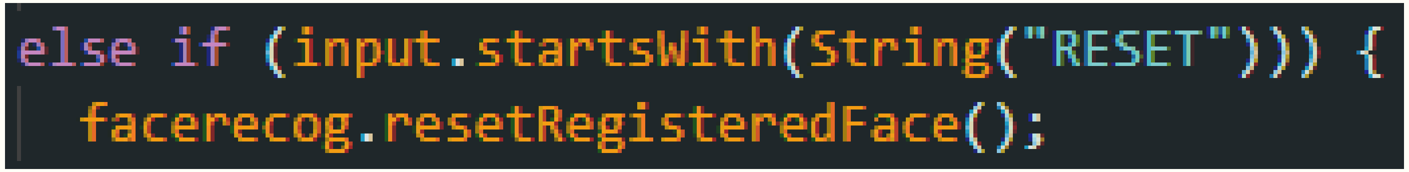
Delete one of the Registered face



```
else if (input.startsWith(String("DEL="))) {
  String name = input.substring(4);
  facerecog.removeFace(name);
```

2.4 Face Recognition

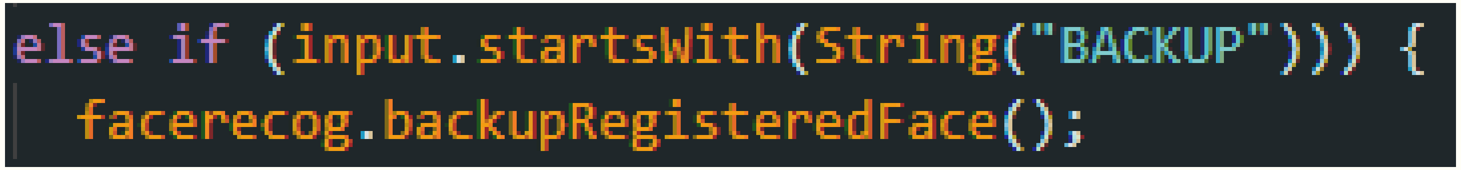
Forget all the Registered face



```
else if (input.startsWith(String("RESET"))) {
  facerecog.resetRegisteredFace();
```

2.4 Face Recognition

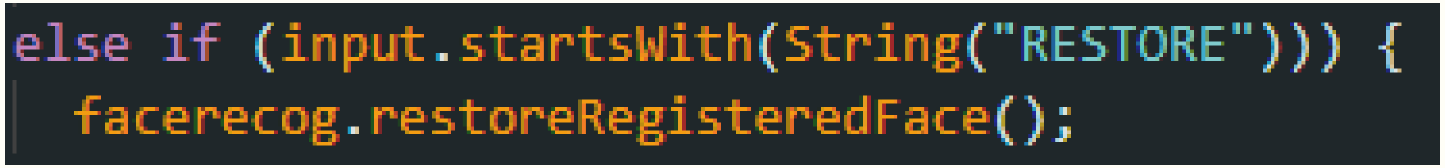
Save registered faces on the board



```
else if (input.startsWith(String("BACKUP"))) {
  facerecog.backupRegisteredFace();
```

2.4 Face Recognition

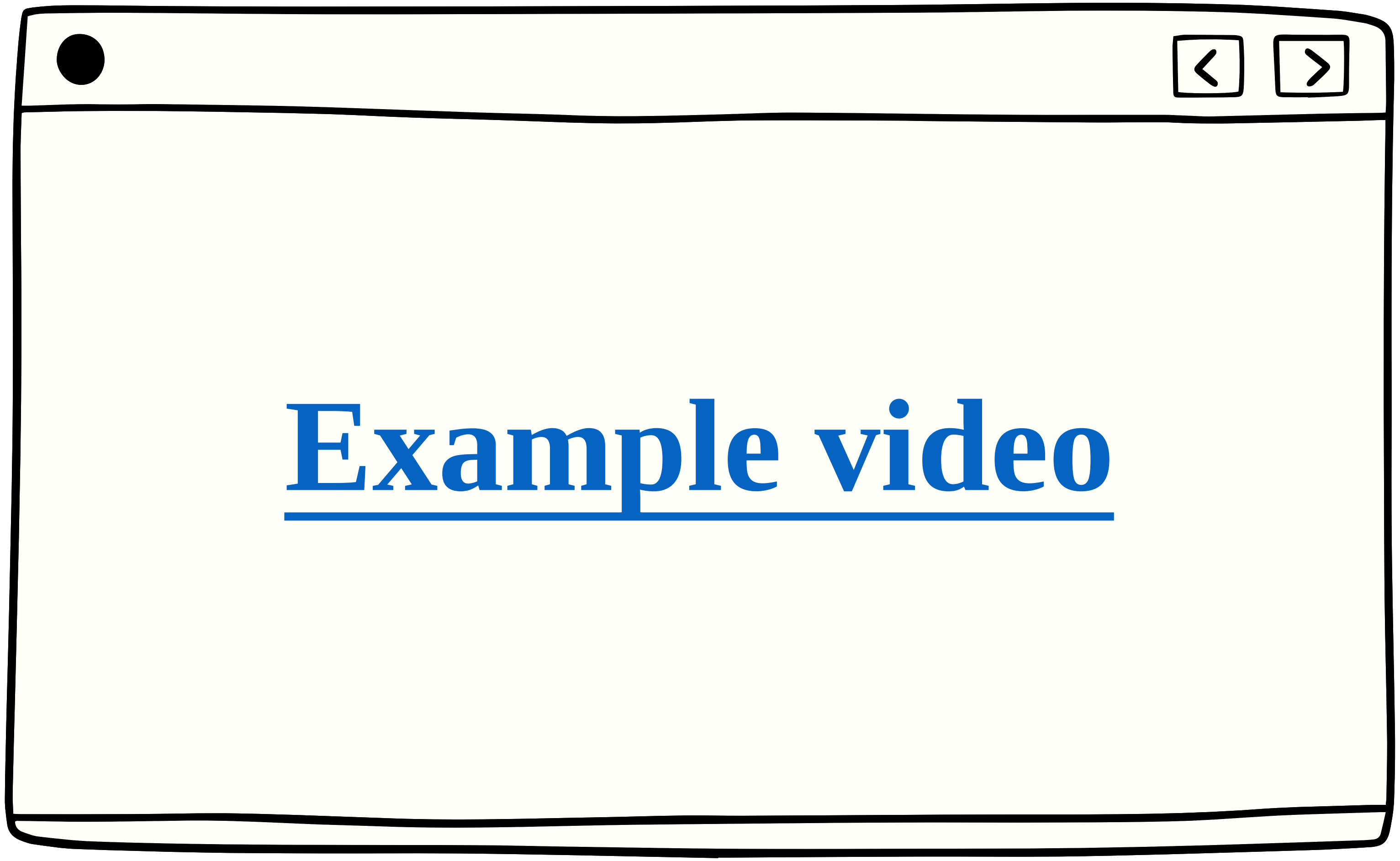
Restore the registered face



```
else if (input.startsWith(String("RESTORE"))) {
  facerecog.restoreRegisteredFace();
```

2.5 Image Classification

2.4 Face Recognition

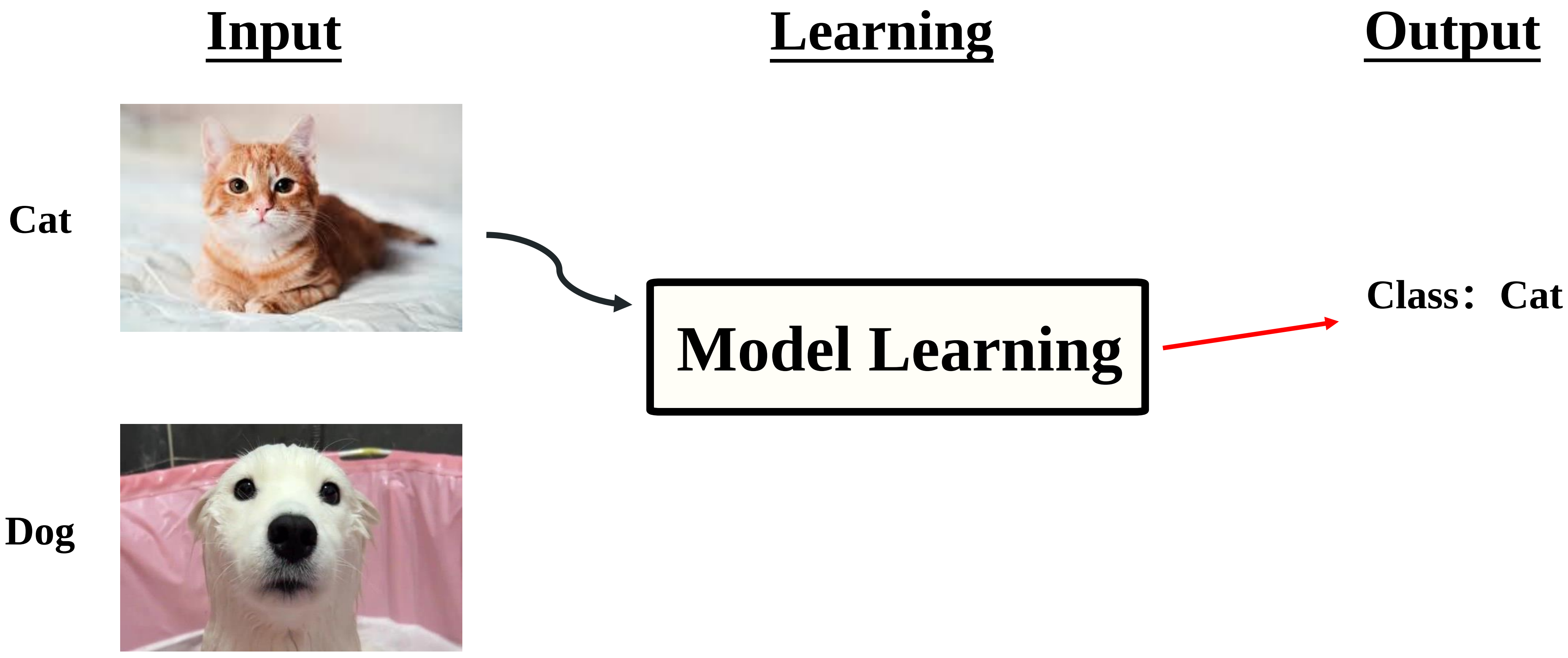


[Example video]

Image Classification basic concept

- Given an image, the model’s task is to determine which predefined category the main object in the image belongs to. For example, in cat and dog classification, the model needs to determine whether the input image is a cat or a dog.

2.5 Image Classification



Input

Learning

Output

Cat

Dog

Model Learning

Class： Cat

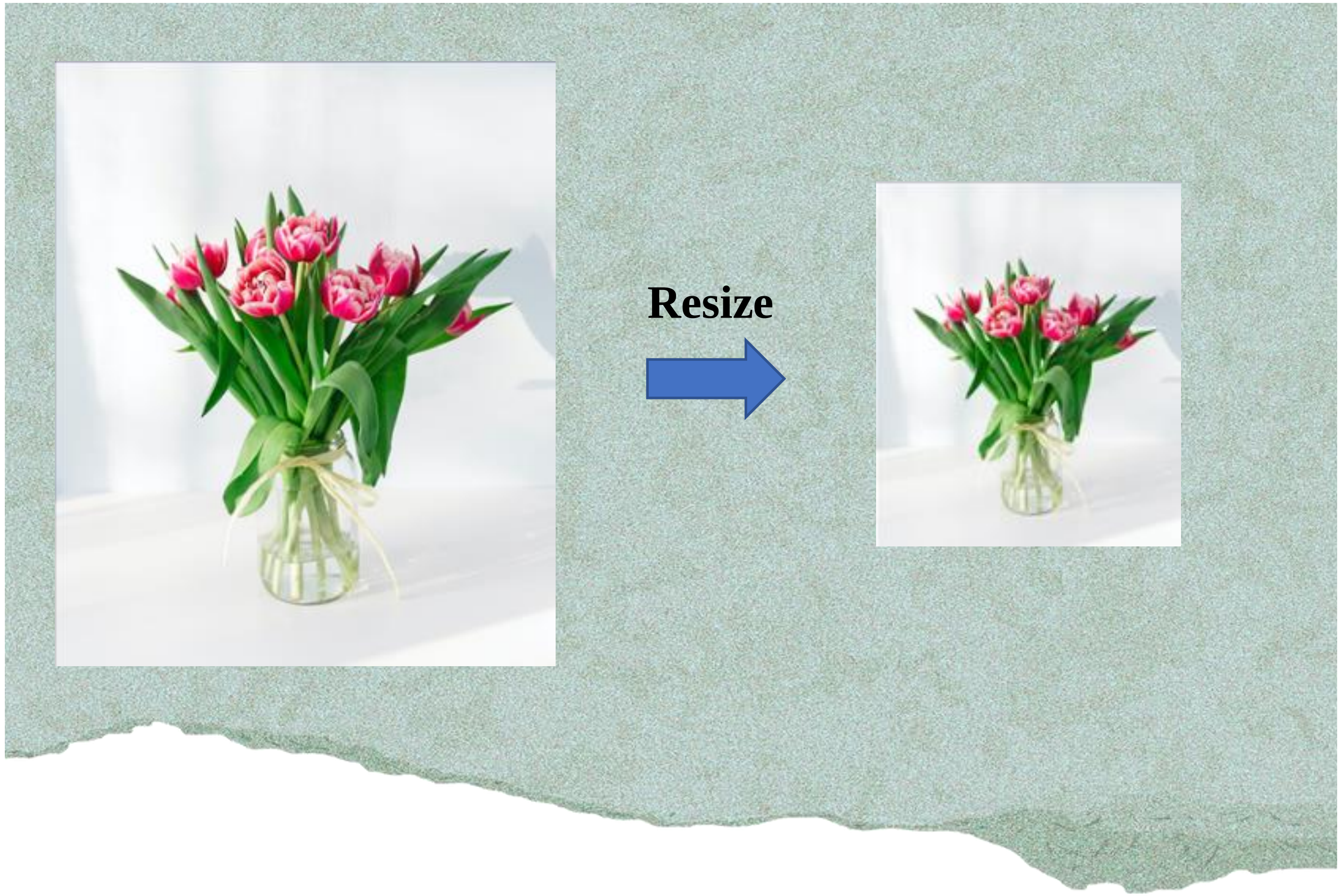
Data Preprocessing

- **Definition: The process of cleaning, transforming, and organizing raw data before performing data analysis, modeling, or machine learning.**
- **Purpose: To ensure the quality and consistency of data, reduce uncertainty, and make the data suitable for subsequent analysis and training.**

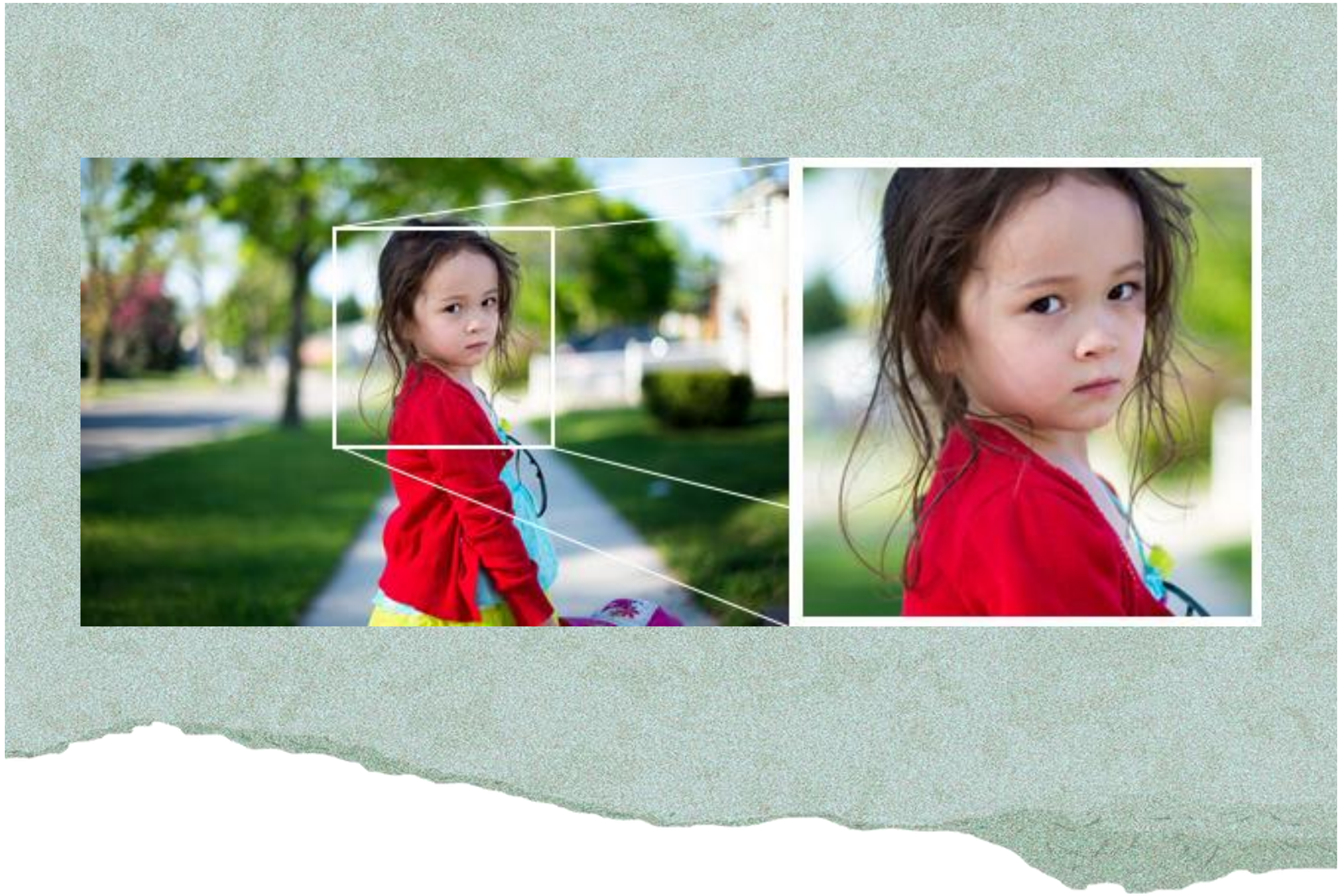
Data Preprocessing

- Resize
- Crop
- Normalization
- Color Conversion

Data Preprocessing - Resize



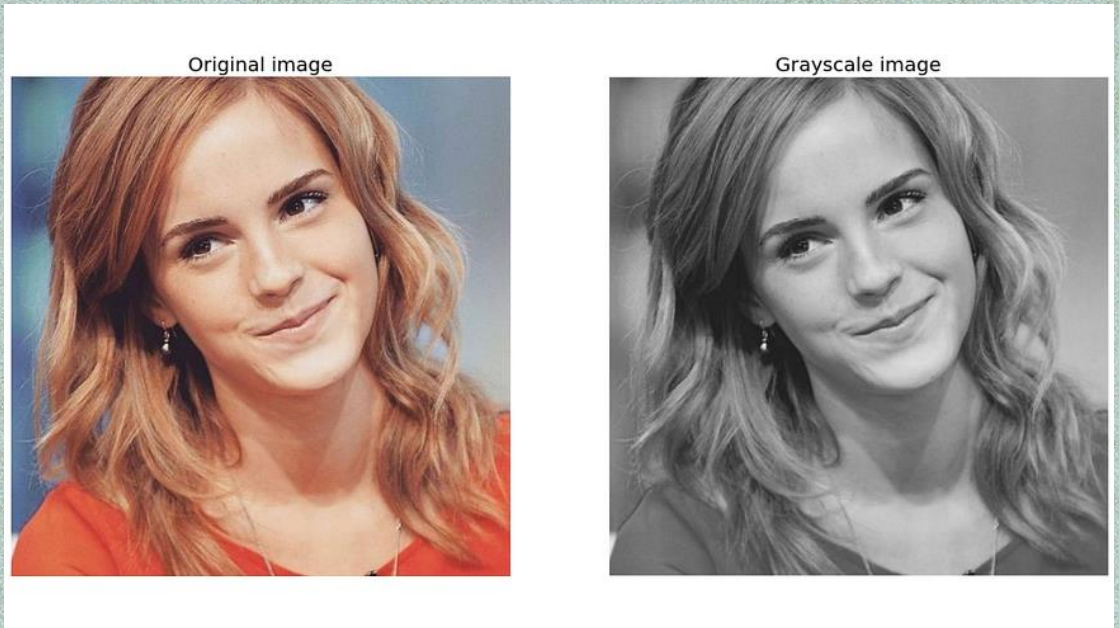
Data Preprocessing - Crop



Data Preprocessing - Normalization

- **Definition: Normalization is to convert the pixel values of an image to a standard range, usually [0,1] or [-1, 1]**

Data Preprocessing - Color Conversion



Data Augmentation

- **Definition: Various random transformations and processing of original data to create more training samples**
- **Purpose: To increase the diversity of data, thereby improving the generalization ability of the model and reducing overfitting.**

Data Augmentation

- Rotation
- Translation
- Flip
- Scaling
- Crop

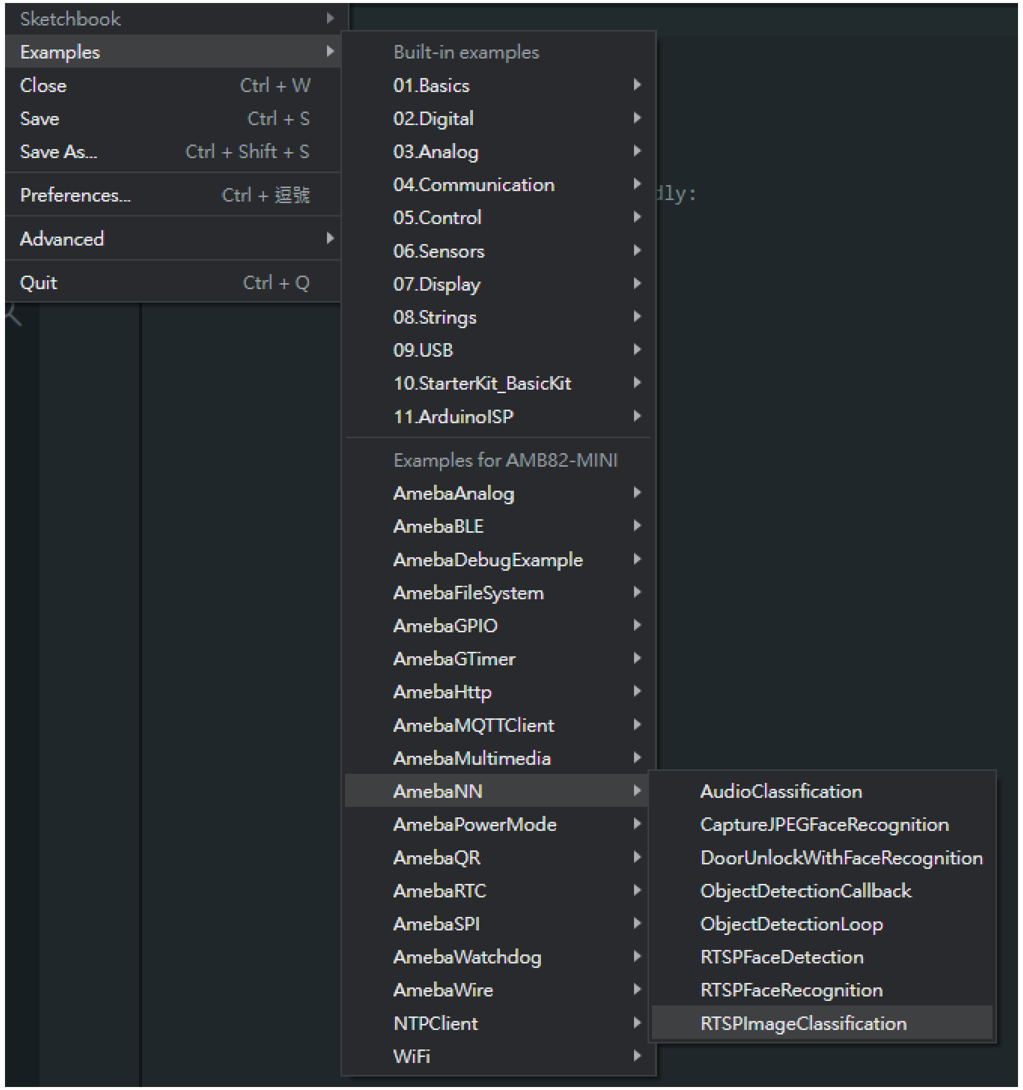
Implementation

2.5 Image Classification

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaNN**
4. **RTSPImageClassification**

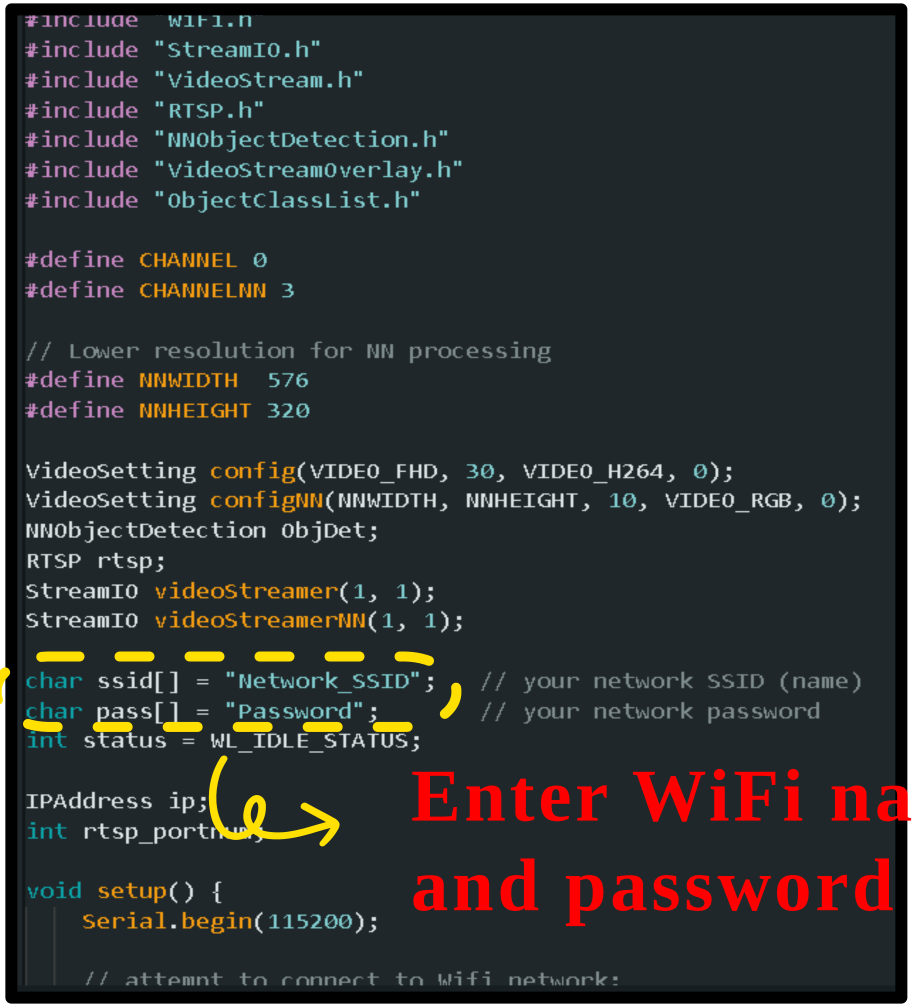


2.5 Image Classification

Step 2.

Enter the WiFi name and password

to the corresponding place in the code.

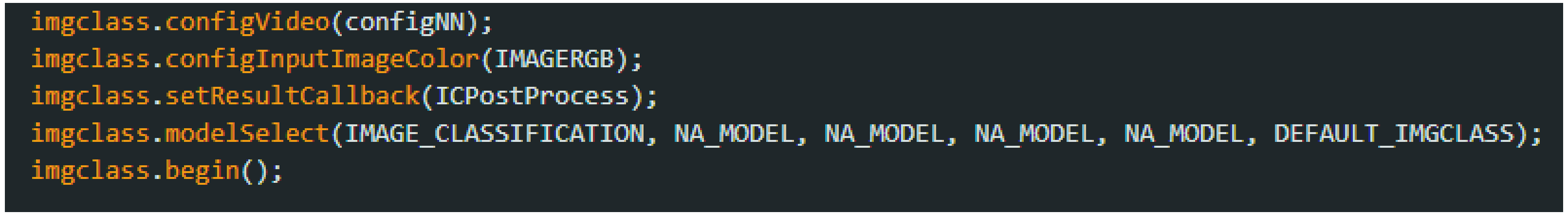


```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNObjectDetection.h"
#include "VideoStreamOverlay.h"
#include "ObjectClassList.h"

#define CHANNEL 0
#define CHANNELNN 3

// Lower resolution for NN processing
#define NNWIDTH  576
#define NNHEIGHT 320

VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
RTSP rtsp;
StreamIO videoStreamer(1, 1);
StreamIO videoStreamerNN(1, 1);

char ssid[] = "Network_SSID";    // your network SSID (name)
char pass[] = "Password";        // your network password
int status = WL_IDLE_STATUS;

IPAddress ip;
int rtsp_portnum;

void setup() {
    Serial.begin(115200);

    // attempt to connect to Wifi network:
```

Enter WiFi name and password

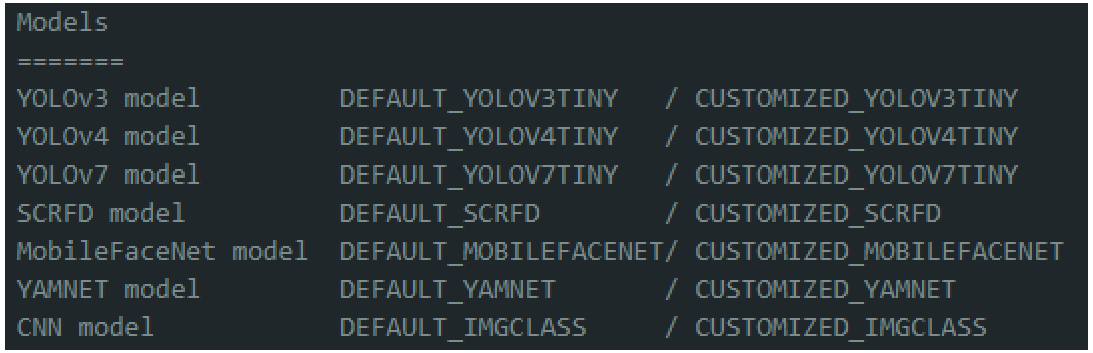
2.5 Image Classification

Step 3. Model choosing(optional)



```
imgclass.configVideo(configNN);
imgclass.configInputImageColor(IMAGERGB);
imgclass.setResultCallback(ICPostProcess);
imgclass.modelSelect(IMAGE_CLASSIFICATION, NA_MODEL, NA_MODEL, NA_MODEL, NA_MODEL, DEFAULT_IMGCLASS);
imgclass.begin();
```

List of models for different tasks



```
Models
=======
YOLOv3 model         DEFAULT_YOLOV3TINY   / CUSTOMIZED_YOLOV3TINY
YOLOv4 model         DEFAULT_YOLOV4TINY   / CUSTOMIZED_YOLOV4TINY
YOLOv7 model         DEFAULT_YOLOV7TINY   / CUSTOMIZED_YOLOV7TINY
SCRFD model          DEFAULT_SCRFD        / CUSTOMIZED_SCRFD
MobileFaceNet model  DEFAULT_MOBILEFACENET/ CUSTOMIZED_MOBILEFACENET
YAMNET model         DEFAULT_YAMNET       / CUSTOMIZED_YAMNET
CNN model            DEFAULT_IMGCLASS     / CUSTOMIZED_IMGCLASS
```

2.6 MQTT ON AMB82

MQTT

- **Definition：Message Queuing Telemetry Transport，which is a lightweight messaging protocol designed specifically for constrained devices and low-bandwidth, high-latency networks.**

Key features of MQTT

- Follow publish/subscribe pattern:
  - Publisher： Publish information to a specified Topic
  - Subscriber： Receive information from a specified topic.
  - Broker： Handle communication between publishers and subscribers.

Key features of MQTT

- **Quality of Service:**

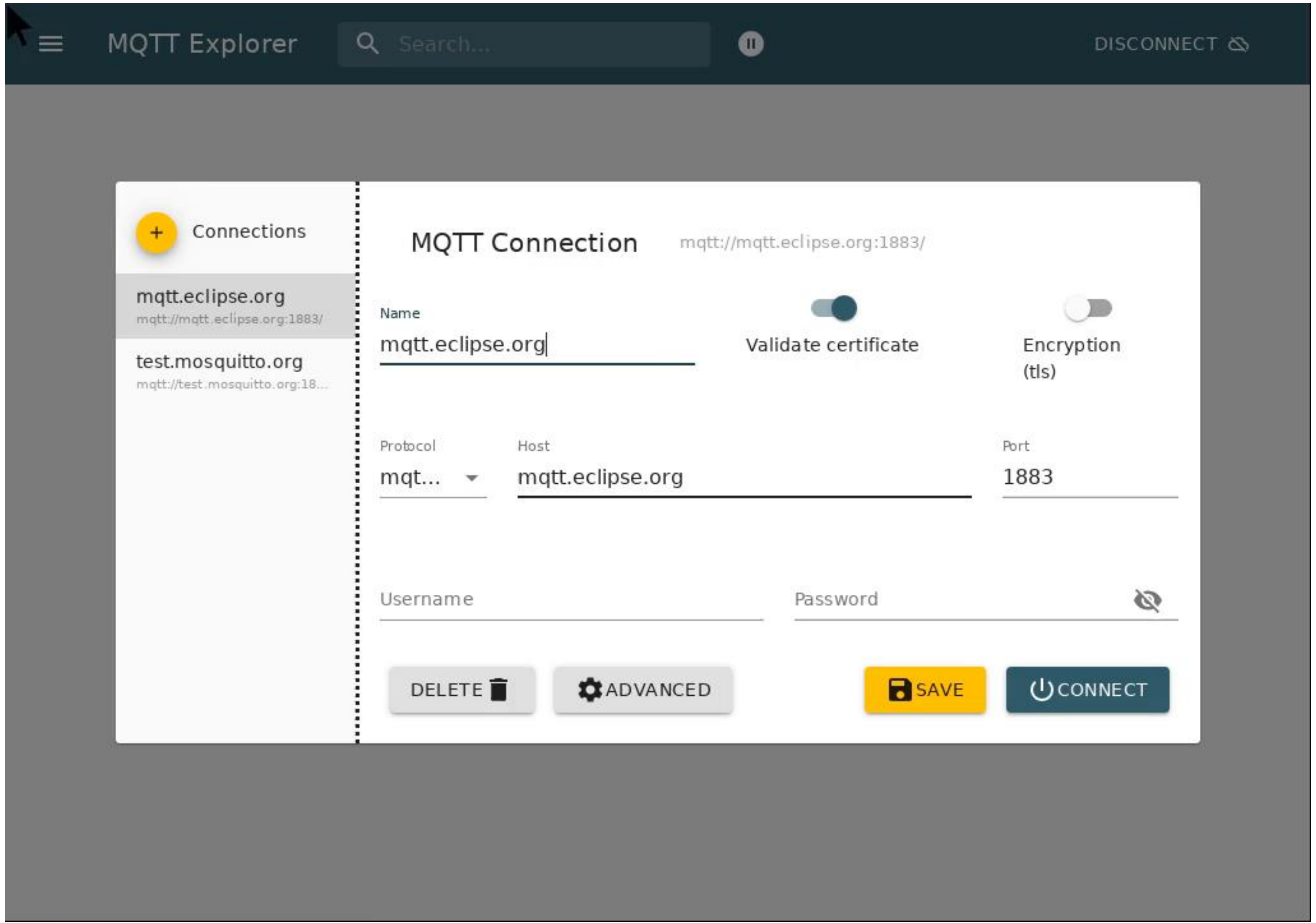
  1. **QoS 0：At most once delivery.**
  2. **QoS 1：At least once delivery.**
  3. **QoS 2：Exactly once delivery.**

Key features of MQTT

- **Last Will and Testament(LWT): When client disconnects, Broker will automatically publish messages.**
- **Persistent Sessions: Ensure the client can retrieve important info from past sessions.**
- **Security: Support TSL/SSL encryption protocols and authentication.**

2.6 MQTT ON AMB82

How to use MQTT Explorer



MQTT Connection mqtt://mqtt.eclipse.org:1883/

Name: mqtt.eclipse.org

Validate certificate: on

Encryption (tls): off

Protocol: mqt...

Host: mqtt.eclipse.org

Port: 1883

Username

Password

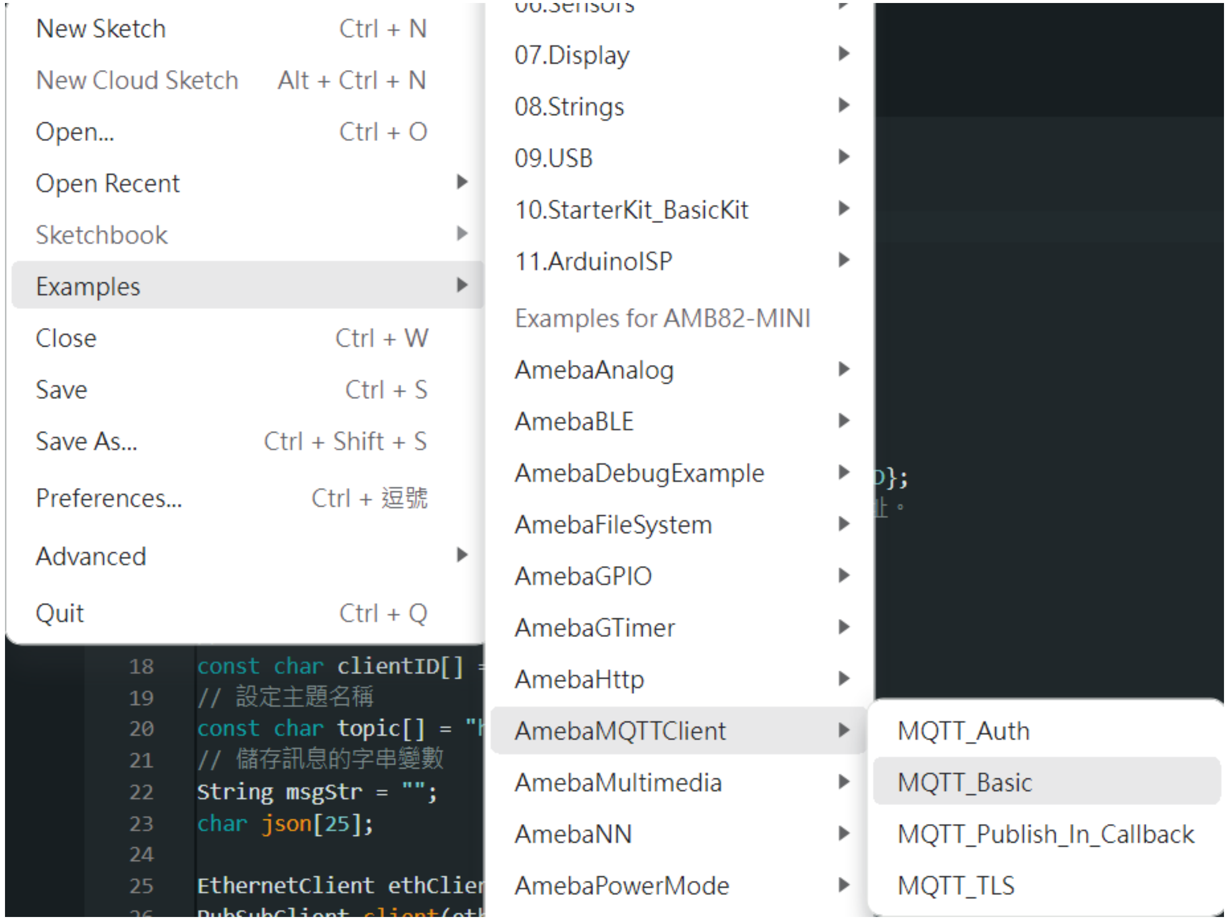
Implementation

2.6 MQTT ON AMB82

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaMQTTClient**
4. **MQTT_Basic**

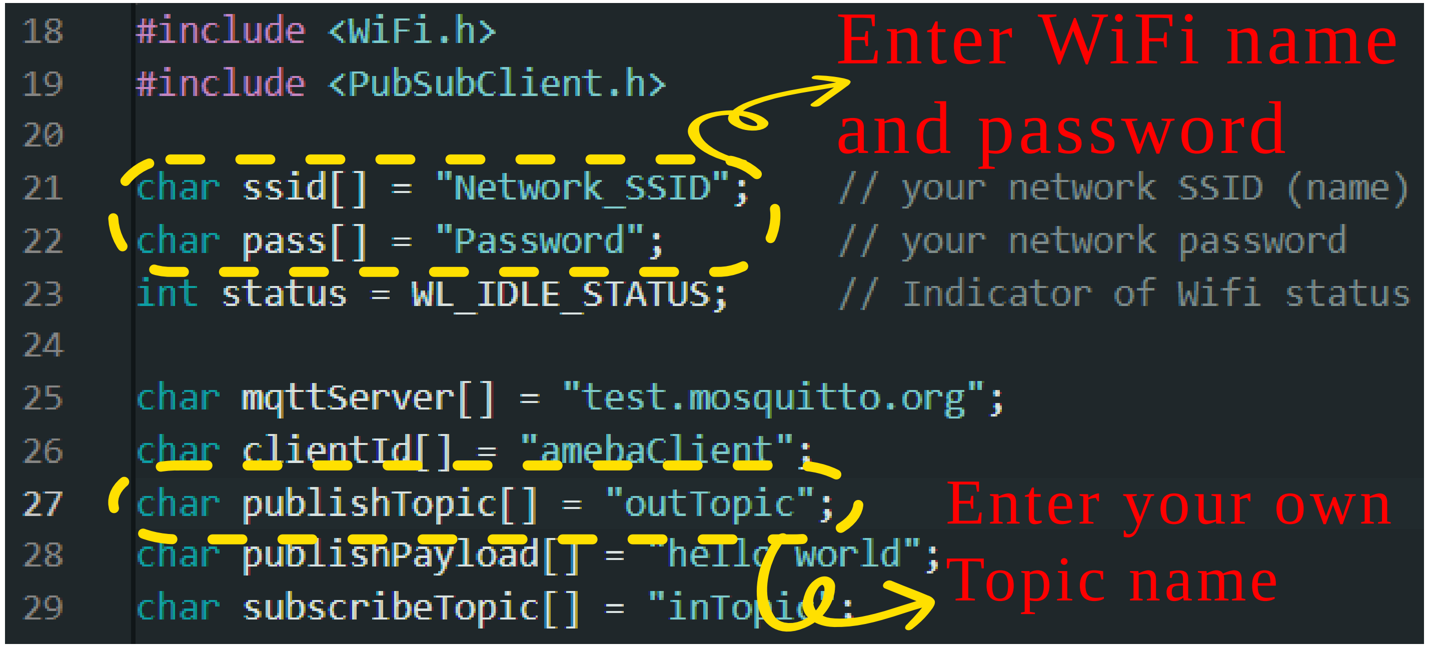


```
const char clientID[] =
// 設定主題名稱
const char topic[] = "I
// 儲存訊息的字串變數
String msgStr = "";
char json[25];

EthernetClient ethClien
PubSubClient client(et
```

2.6 MQTT ON AMB82

Step 2.

Enter the WiFi name, password and publishTopic to the corresponding place in the code.



```
#include <WiFi.h>
#include <PubSubClient.h>

char ssid[] = "Network_SSID";     // your network SSID (name)
char pass[] = "Password";         // your network password
int status = WL_IDLE_STATUS;      // Indicator of Wifi status

char mqttServer[] = "test.mosquitto.org";
char clientId[] = "amebaClient";
char publishTopic[] = "outTopic";
char publishPayload[] = "hello world";
char subscribeTopic[] = "inTopic";
```

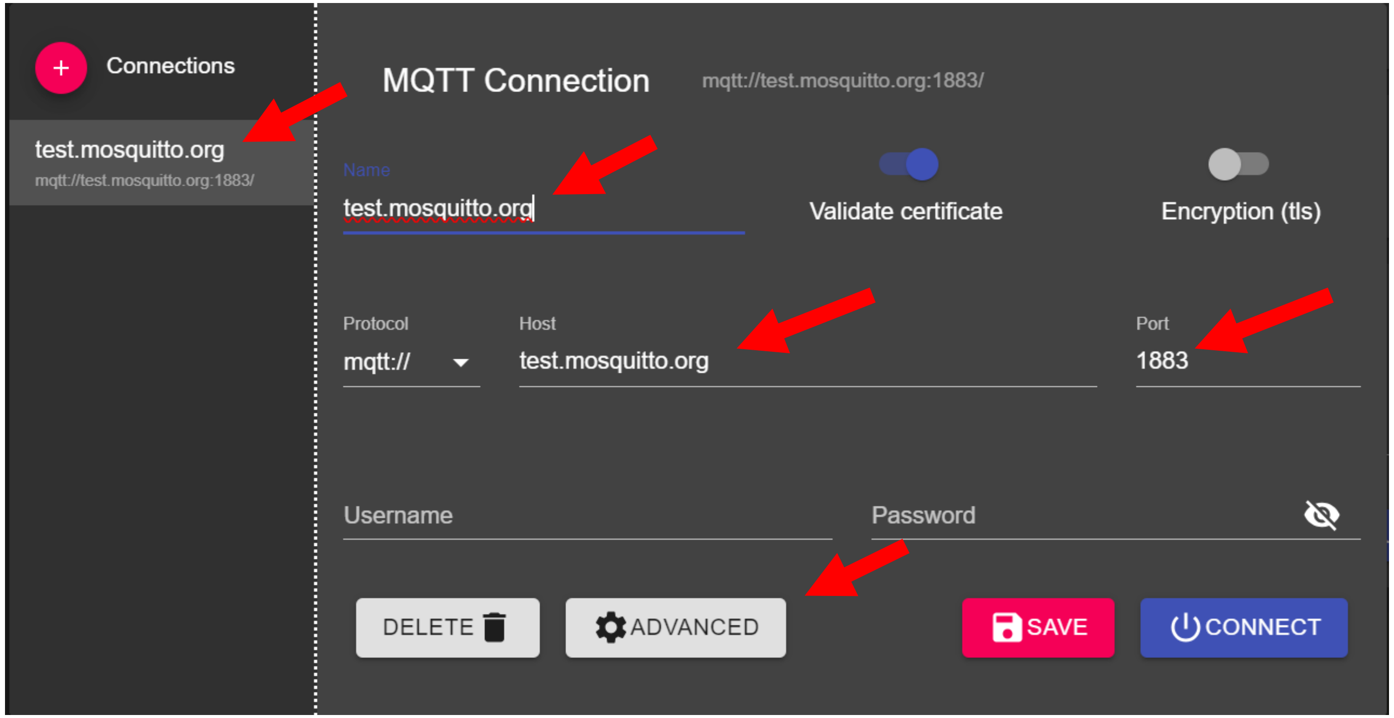
Enter WiFi name and password

Enter your own Topic name

2.6 MQTT ON AMB82

Step 3.

Open MQTT Explorer and make sure open the same connection as the code. Then click the ADVANCED button.



MQTT Connection mqtt://test.mosquitto.org:1883/

Name: test.mosquitto.org

Validate certificate

Encryption (tls)

Protocol: mqtt://

Host: test.mosquitto.org

Port: 1883

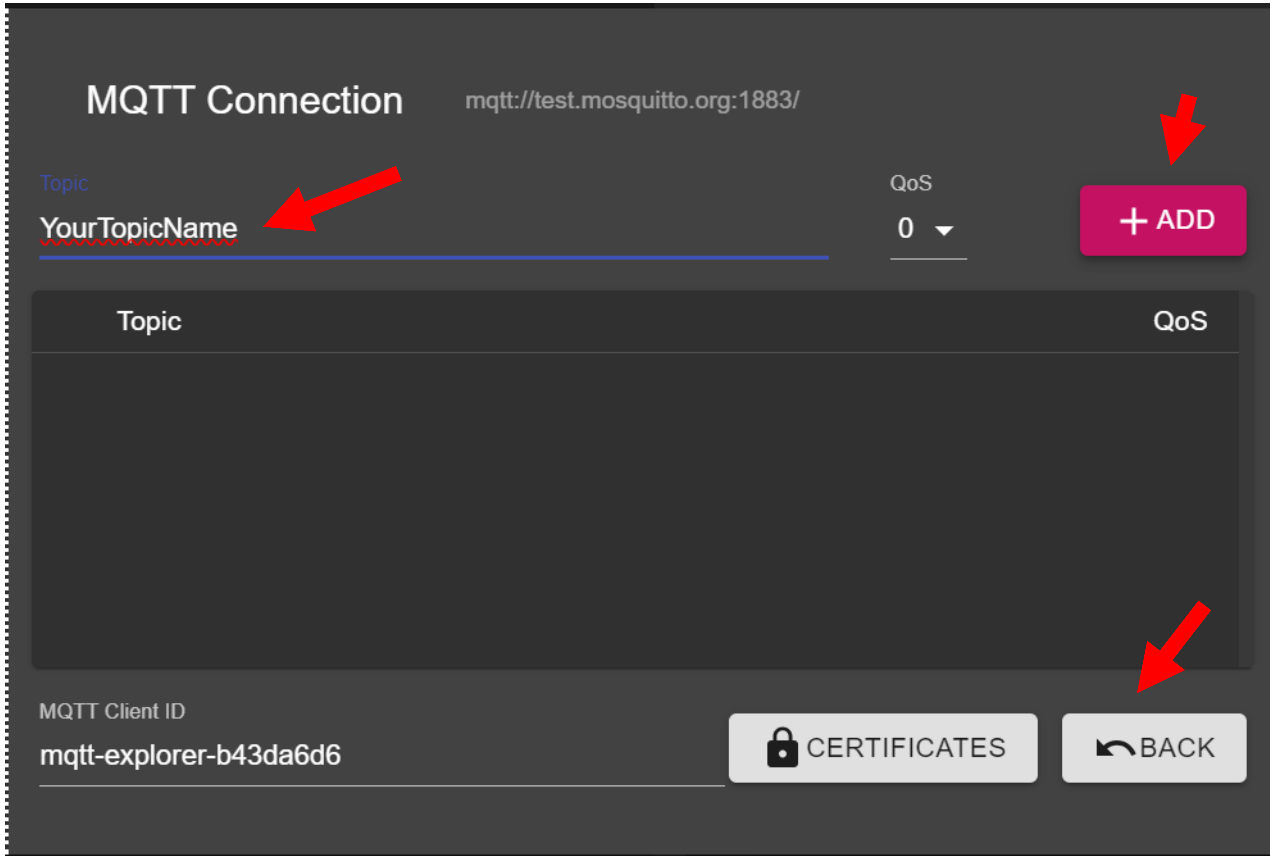
Username

Password

2.6 MQTT ON AMB82

Step 4.

Enter the publishTopic you named in the code to the Topic, then press the +ADD button. Then press BACK button.



MQTT Connection mqtt://test.mosquitto.org:1883/

Topic: YourTopicName

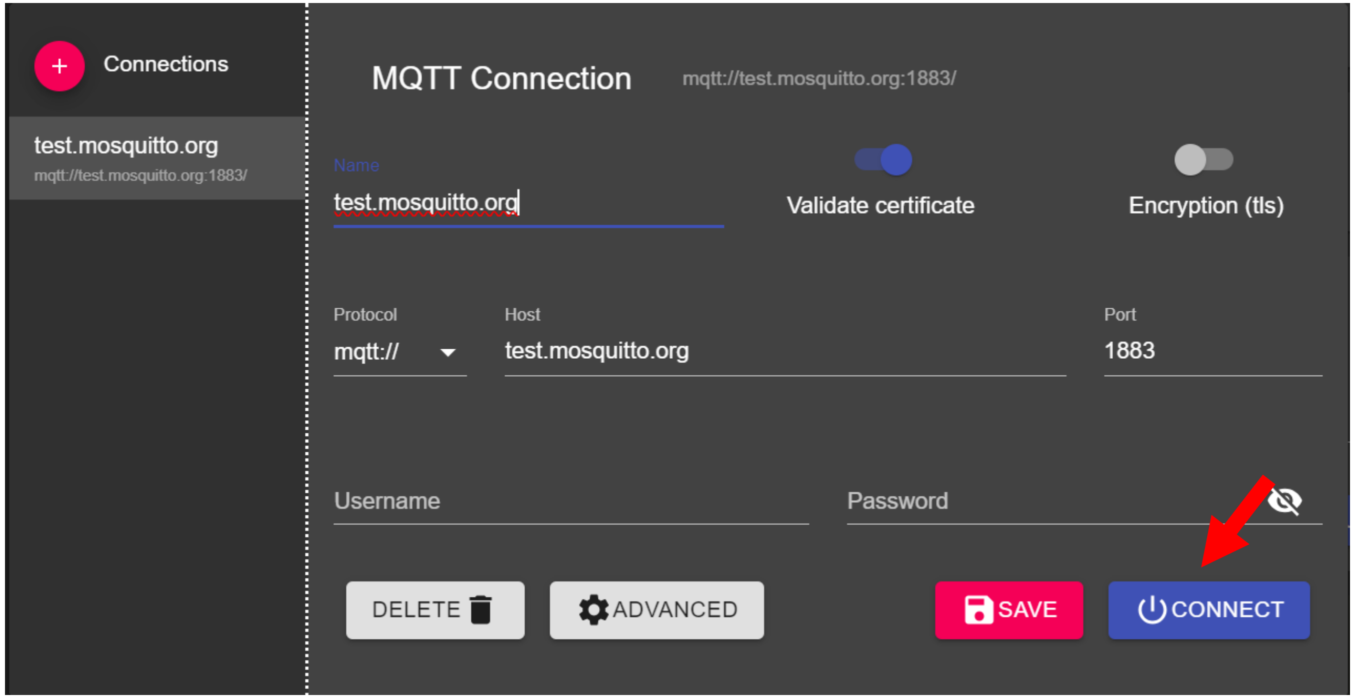
QoS: 0

MQTT Client ID: mqtt-explorer-b43da6d6

2.6 MQTT ON AMB82

Step 5.

After doing all the previous step, press the CONNECT button to start connecting.



Connections

test.mosquitto.org
mqtt://test.mosquitto.org:1883/

MQTT Connection mqtt://test.mosquitto.org:1883/

Name: test.mosquitto.org

Validate certificate: on

Encryption (tls): off

Protocol: mqtt://

Host: test.mosquitto.org

Port: 1883

Username

Password

AIoT Implementation 1

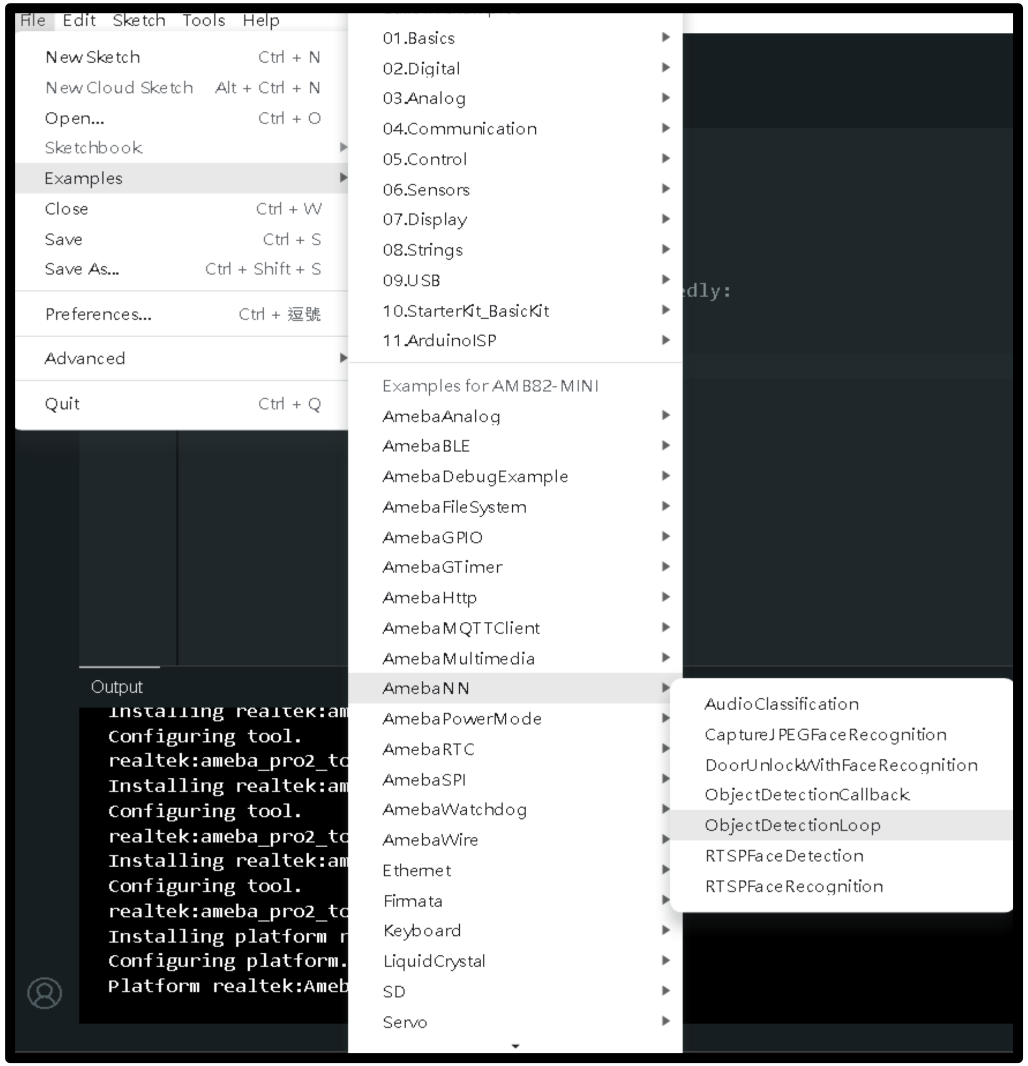
Combine object detection with MQTT to transmit results to the cloud

2.6 MQTT ON AMB82

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaNN**
4. **ObjectDetectionLoop**



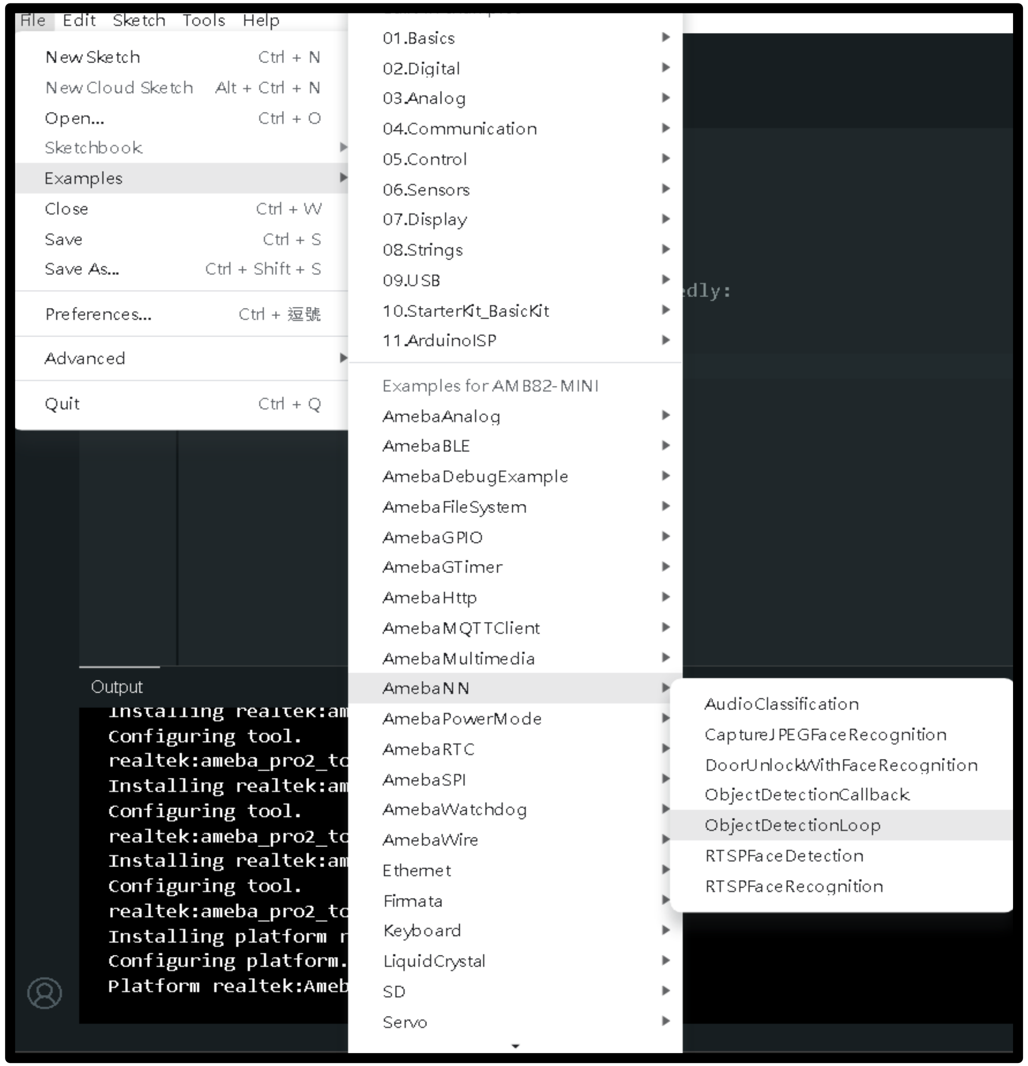
```
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing platform r
Configuring platform.
Platform realtek:Ameb
```

2.6 MQTT ON AMB82

Step 2.

After opening it, copy the code from the following link, then paste the code to Arduino IDE.

https://drive.google.com/file/d/1ABJJTcOY2TODcuO88m8AEMS3zuik4idT/view?usp=sharing

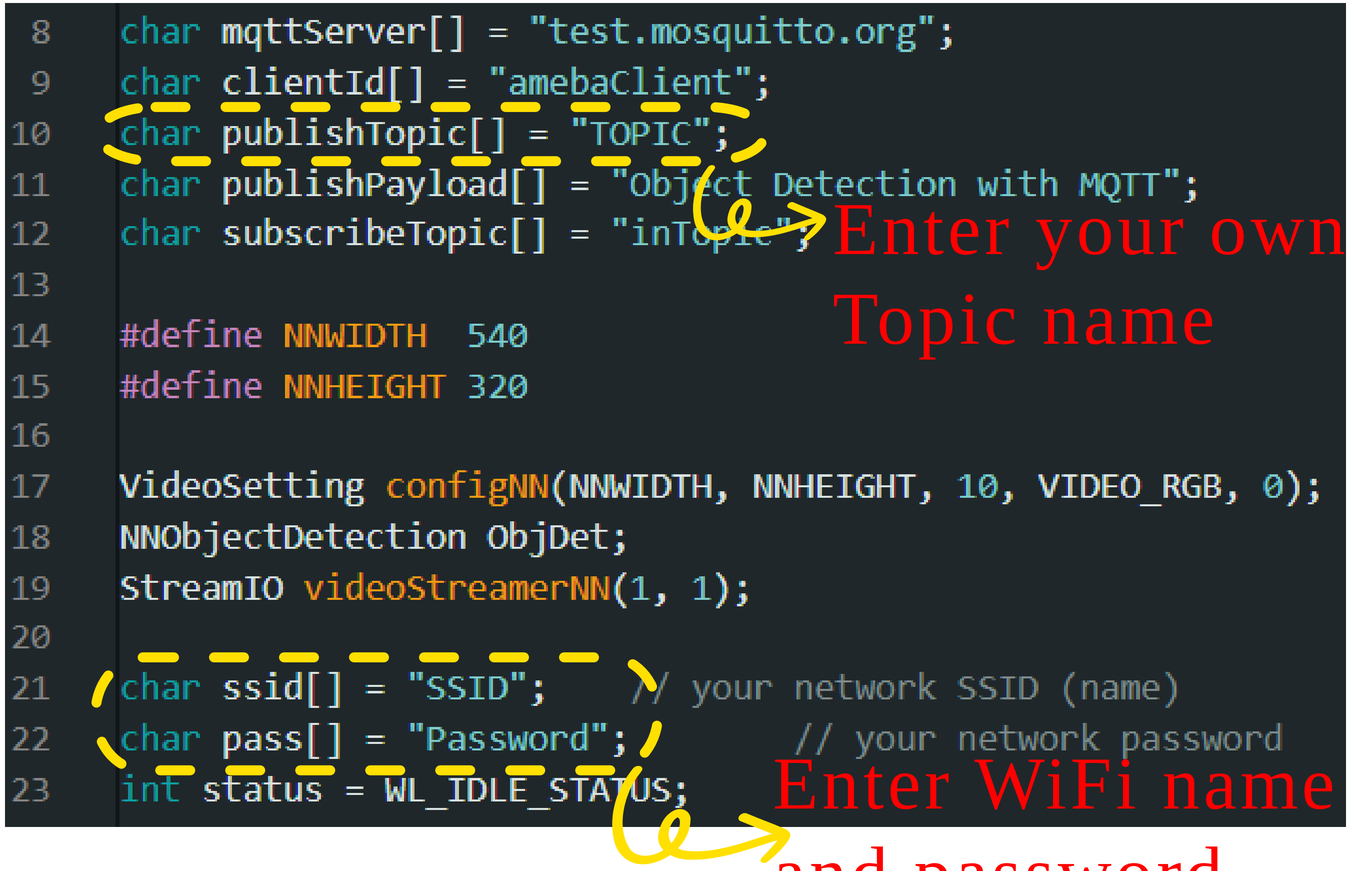


```
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing platform r
Configuring platform.
Platform realtek:Ameb
```

2.6 MQTT ON AMB82

Step 3.

Enter the WiFi name, password and publishTopic to the corresponding place in the code.



```
char mqttServer[] = "test.mosquitto.org";
char clientId[] = "amebaClient";
char publishTopic[] = "TOPIC";
char publishPayload[] = "Object Detection with MQTT";
char subscribeTopic[] = "inTopic";

#define NNWIDTH  540
#define NNHEIGHT 320

VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
StreamIO videoStreamerNN(1, 1);

char ssid[] = "SSID";      // your network SSID (name)
char pass[] = "Password";       // your network password
int status = WL_IDLE_STATUS;
```

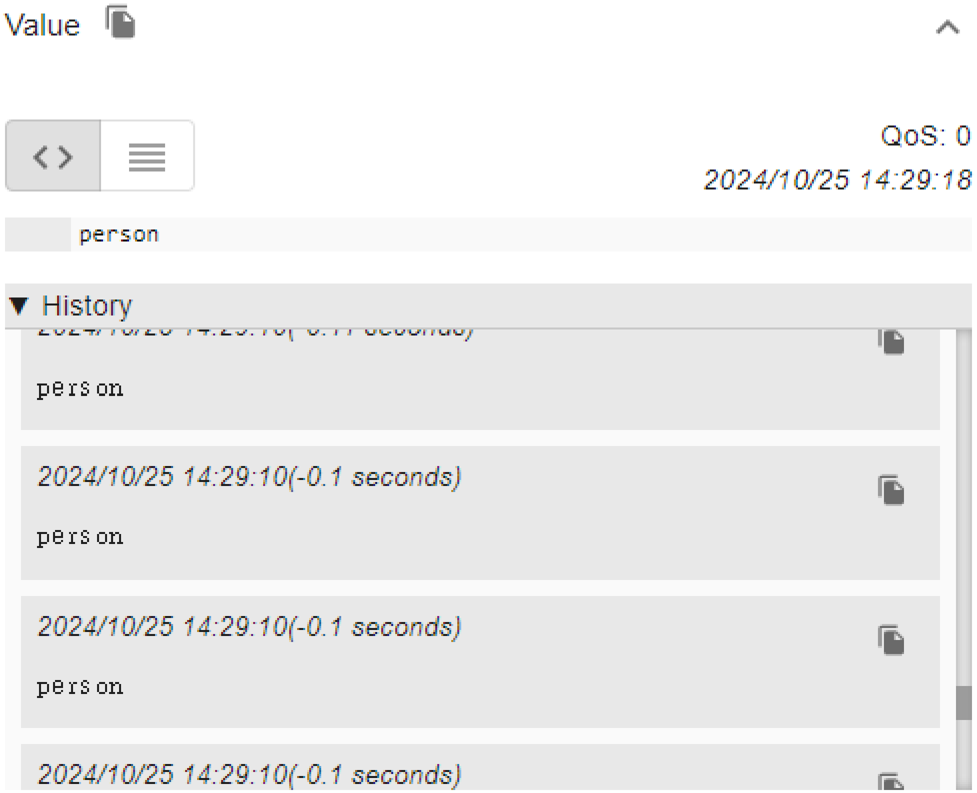
Enter your own Topic name

Enter WiFi name and password

2.6 MQTT ON AMB82

Step 4.

The detection results will be displayed in MQTT Explorer.



2024/10/25 14:29:18

```
person
```

```
person
```

2024/10/25 14:29:10(-0.1 seconds)

```
person
```

2024/10/25 14:29:10(-0.1 seconds)

```
person
```

2024/10/25 14:29:10(-0.1 seconds)

AIoT Implementation 2

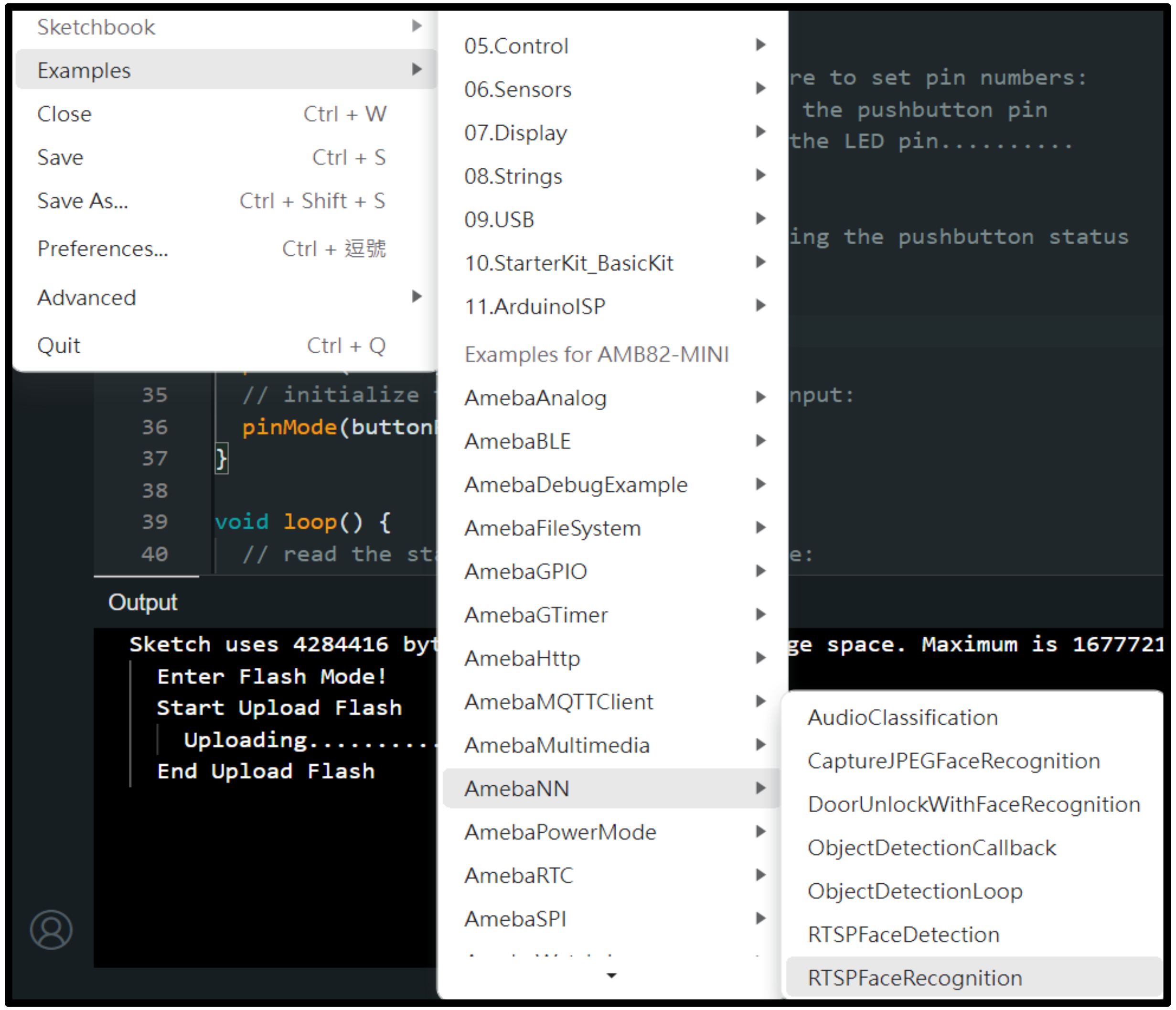
Simple facial recognition clock-in system

2.6 MQTT ON AMB82

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaNN**
4. **RTSPFaceRecognition**



```
re to set pin numbers:
 the pushbutton pin
the LED pin..........

ing the pushbutton status

  // initialize
  pinMode(button
}

void loop() {
  // read the st
```

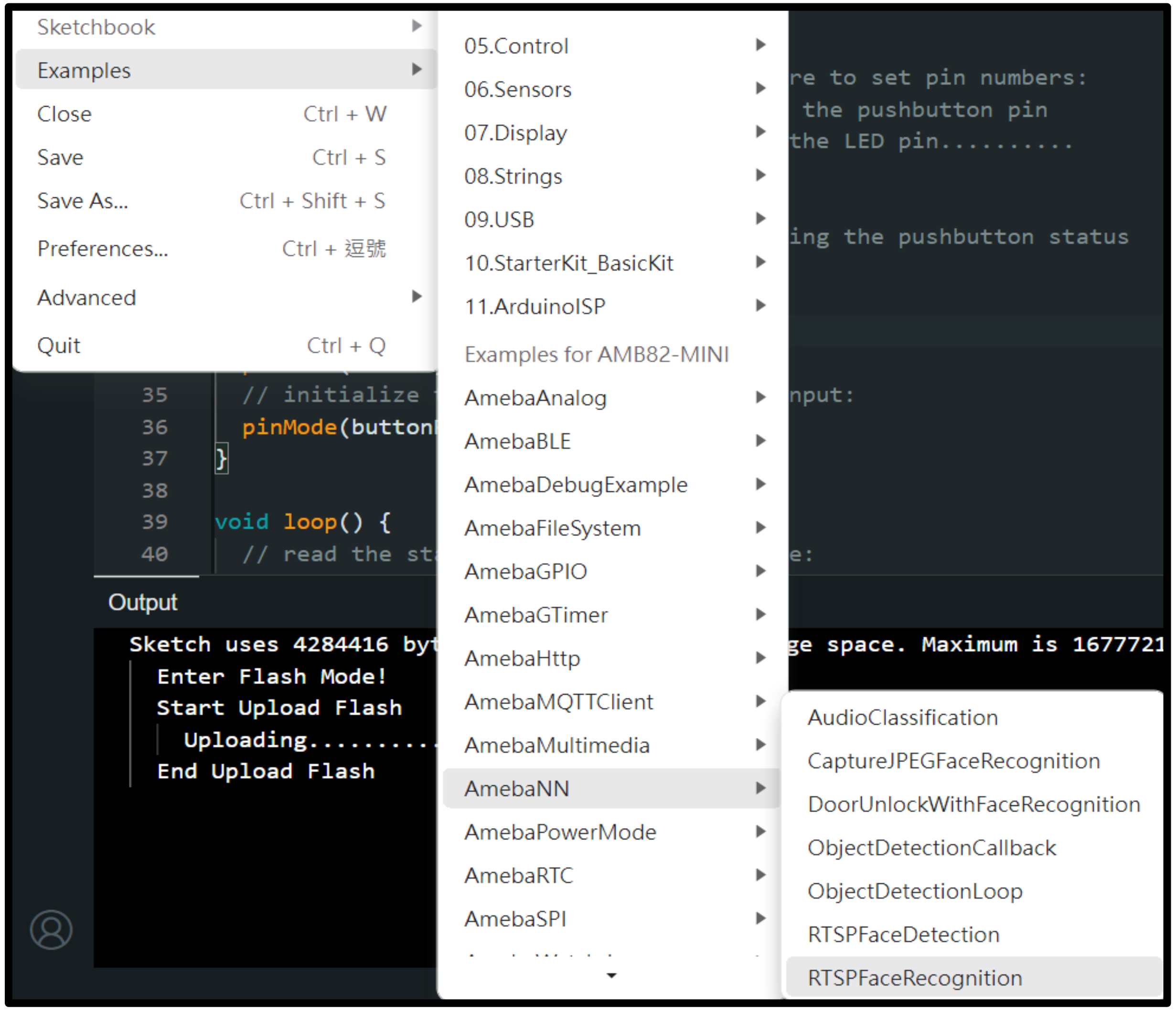
```
Sketch uses 4284416 byt                    ge space. Maximum is 1677721
  Enter Flash Mode!
  Start Upload Flash
    Uploading.........
  End Upload Flash
```

2.6 MQTT ON AMB82

Step 2.

After opening it, copy the code from the following link, then paste the code to Arduino IDE.

https://drive.google.com/file/d/15r02-z5OYz23EaD29MFa0rdimRyMYx8P/view?usp=sharing



```
re to set pin numbers:
 the pushbutton pin
the LED pin.........

ing the pushbutton status

nput:
```

```
  // initialize
  pinMode(button
}

void loop() {
  // read the st
```

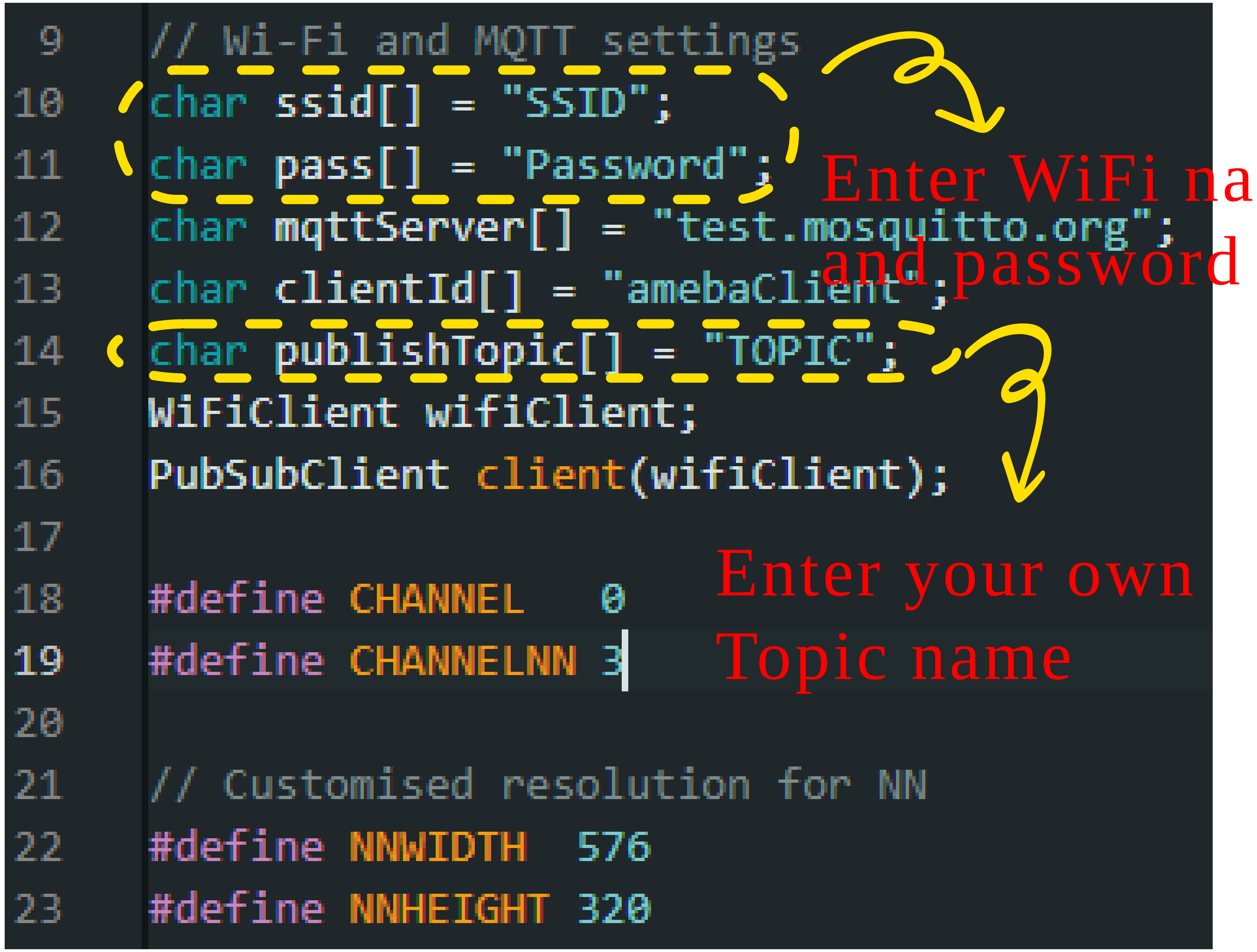
```
e:
```

```
Sketch uses 4284416 byt                 ge space. Maximum is 1677721
  Enter Flash Mode!
  Start Upload Flash
    Uploading.........
  End Upload Flash
```

2.6 MQTT ON AMB82

Step 3.

Enter the WiFi name, password and publishTopic to the corresponding place in the code.



```
// Wi-Fi and MQTT settings
char ssid[] = "SSID";
char pass[] = "Password";
char mqttServer[] = "test.mosquitto.org";
char clientId[] = "amebaClient";
char publishTopic[] = "TOPIC";
WiFiClient wifiClient;
PubSubClient client(wifiClient);

#define CHANNEL   0
#define CHANNELNN 3

// Customised resolution for NN
#define NNWIDTH  576
#define NNHEIGHT 320
```

Enter WiFi name and password

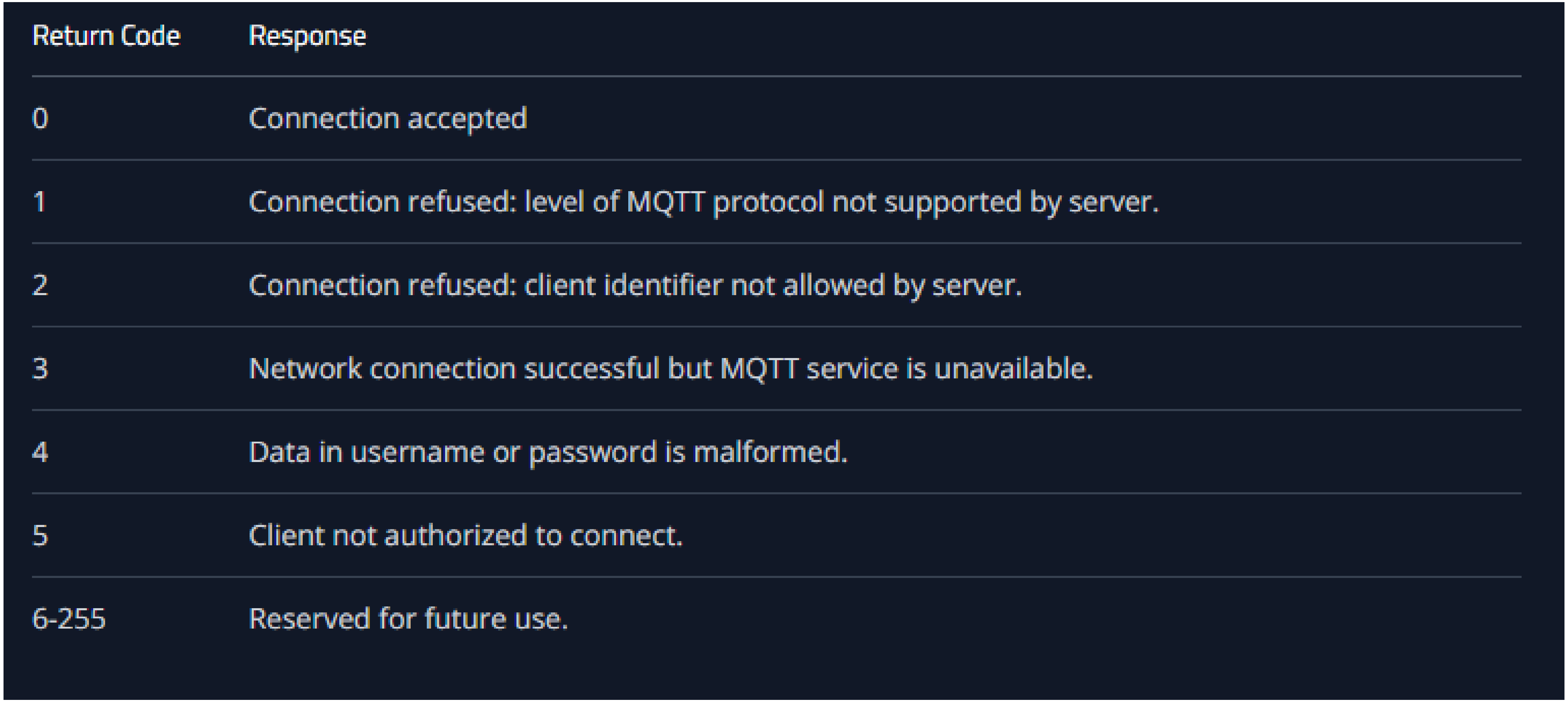
Enter your own Topic name

2.6 MQTT ON AMB82

1. **Open Python (VScode、anaconda)**
2. **Enter pip install paho-mqtt in the terminal**
3. **Create a python (.py) file and copy and paste the code from the following link into the newly created file.**

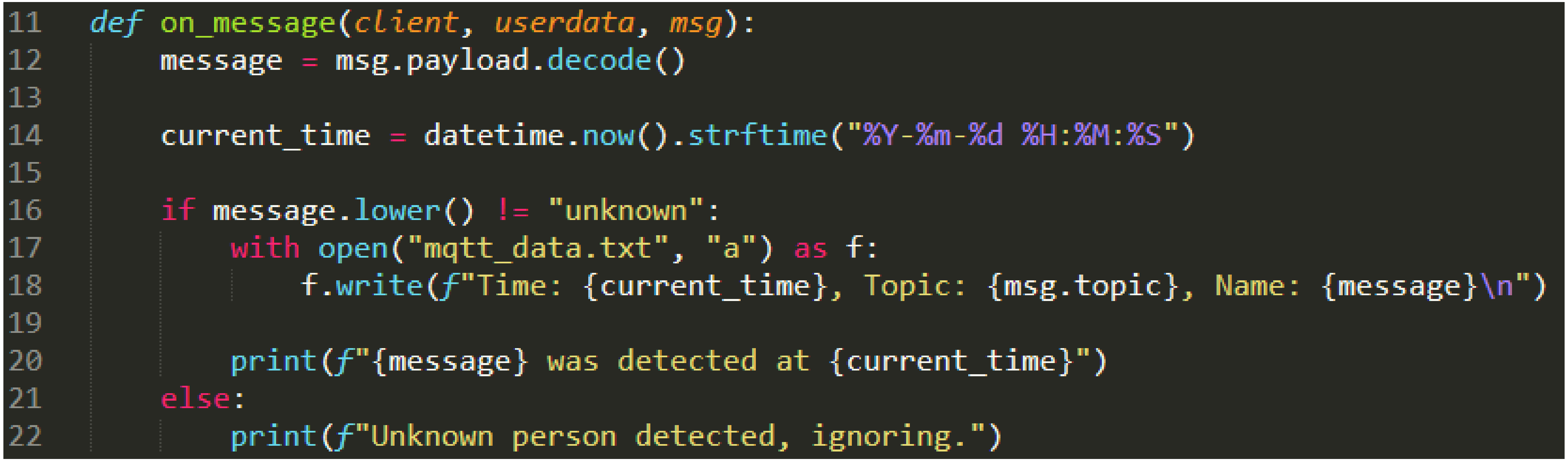
   https://drive.google.com/file/d/1_GRLBpuqgWkN8e_25UPkef-dO_pNCIDe/view?usp=sharing

2.6 MQTT ON AMB82



Return Code	Response
0	Connection accepted
1	Connection refused: level of MQTT protocol not supported by server.
2	Connection refused: client identifier not allowed by server.
3	Network connection successful but MQTT service is unavailable.
4	Data in username or password is malformed.
5	Client not authorized to connect.
6-255	Reserved for future use.

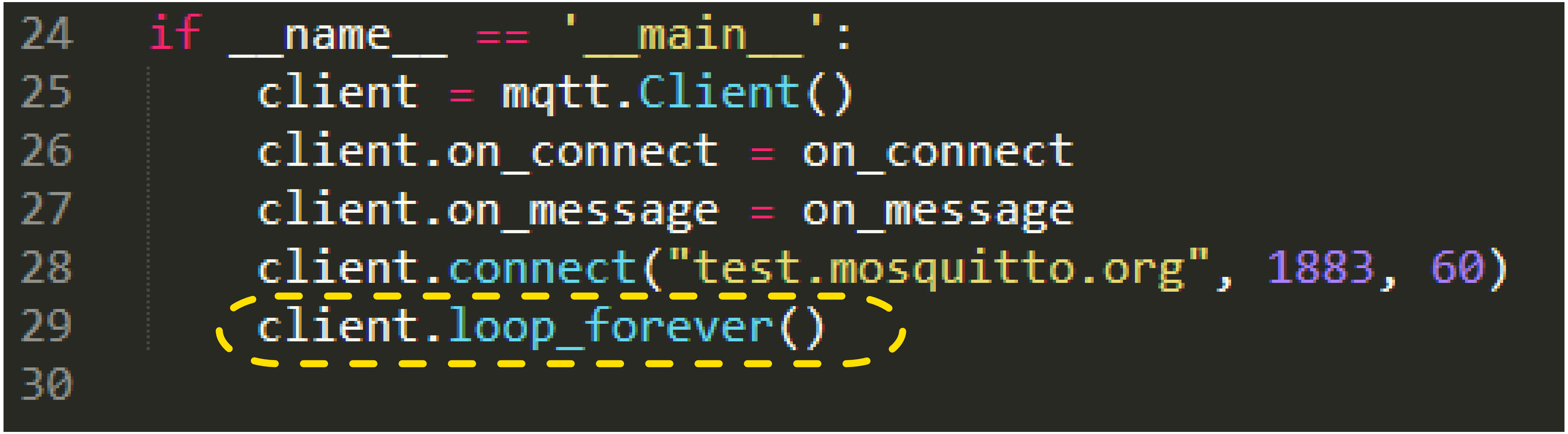
2.6 MQTT ON AMB82



```python
def on_message(client, userdata, msg):
    message = msg.payload.decode()

    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    if message.lower() != "unknown":
        with open("mqtt_data.txt", "a") as f:
            f.write(f"Time: {current_time}, Topic: {msg.topic}, Name: {message}\n")

        print(f"{message} was detected at {current_time}")
    else:
        print(f"Unknown person detected, ignoring.")
```

on_message function explanation

2.6 MQTT ON AMB82

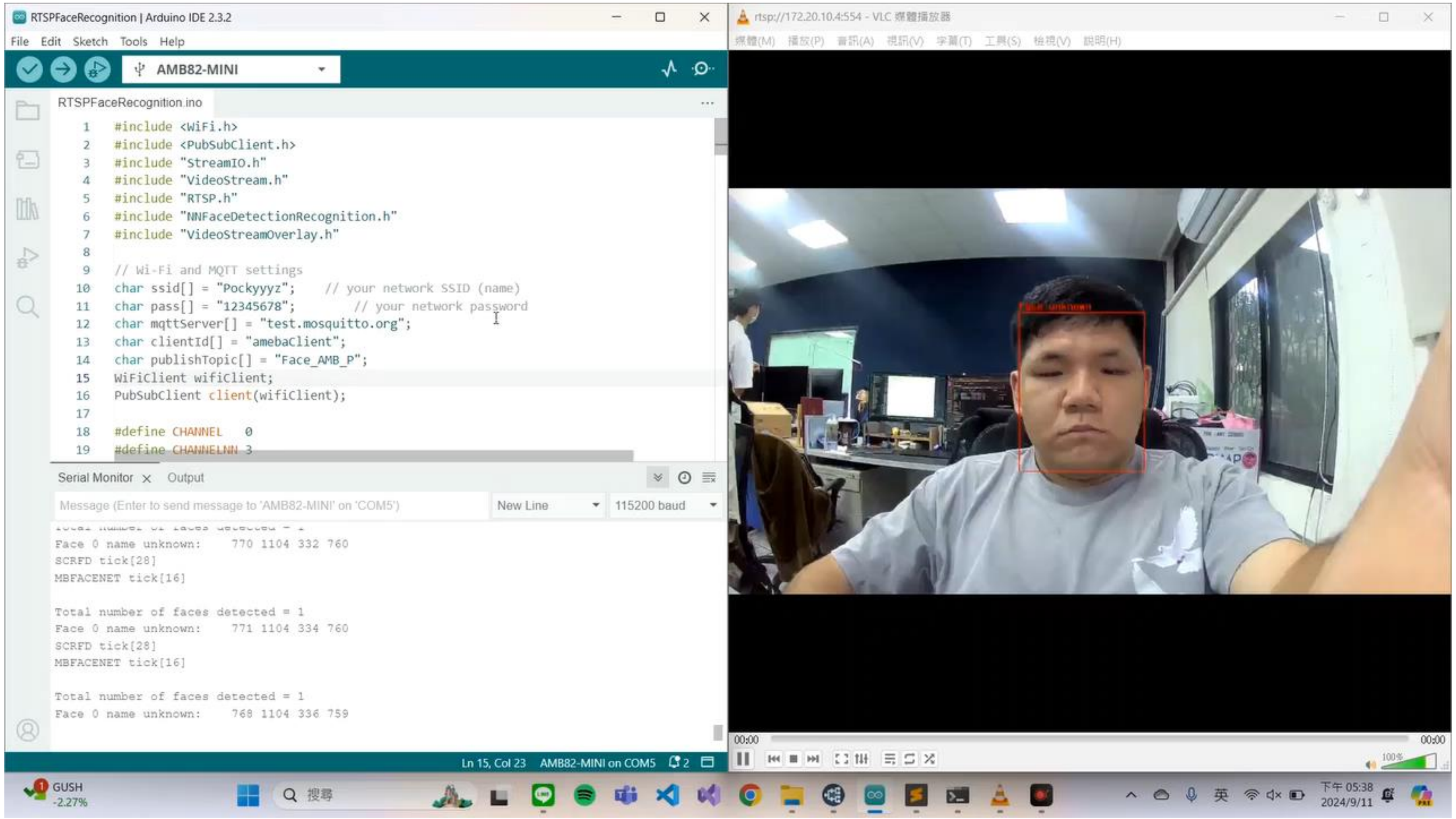


```
if __name__ == '__main__':
    client = mqtt.Client()
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect("test.mosquitto.org", 1883, 60)
    client.loop_forever()

```

main function explanation

2.6 MQTT ON AMB82

DEMO



RTSPFaceRecognition | Arduino IDE 2.3.2
File Edit Sketch Tools Help
AMB82-MINI
RTSPFaceRecognition.ino

```
#include <WiFi.h>
#include <PubSubClient.h>
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNFaceDetectionRecognition.h"
#include "VideoStreamOverlay.h"

// Wi-Fi and MQTT settings
char ssid[] = "Pockyyyz";    // your network SSID (name)
char pass[] = "12345678";        // your network password
char mqttServer[] = "test.mosquitto.org";
char clientId[] = "amebaClient";
char publishTopic[] = "Face_AMB_P";
WiFiClient wifiClient;
PubSubClient client(wifiClient);

#define CHANNEL   0
#define CHANNELNN 3
```

```
Total number of faces detected = 1
Face 0 name unknown:    770 1104 332 760
SCRFD tick[28]
MBFACENET tick[16]

Total number of faces detected = 1
Face 0 name unknown:    771 1104 334 760
SCRFD tick[28]
MBFACENET tick[16]

Total number of faces detected = 1
Face 0 name unknown:    768 1104 336 759
```

Chapter 3
Object Detection

3.1 YOLO

(You Only Look Once)

3.1 YOLO(You Only Look Once)

What is ObjectDetection?

- Object detection is to use an anchor to mark the range of the object in image content such as photos or videos, and classify it into what kind of object it is and the degree of confidence of the attached model in this object
- The most popular and famous object detection model currently is YOLO.

3.1 YOLO(You Only Look Once)

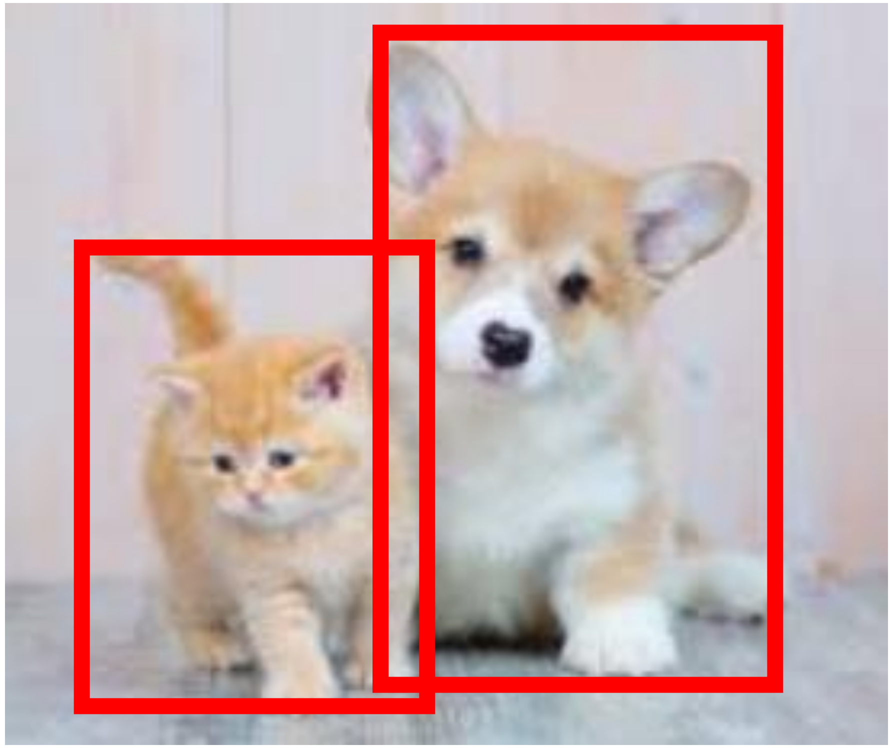
What is ObjectDetection?

- Object detection is to use an **anchor** to mark the range of the object in image content such as photos or videos, and **classify** it into what kind of object it is and the **degree of confidence** of the attached model in this object
- The most popular and famous object detection model currently is **YOLO**.

3.1 YOLO(You Only Look Once)

What is ObjectDetection?

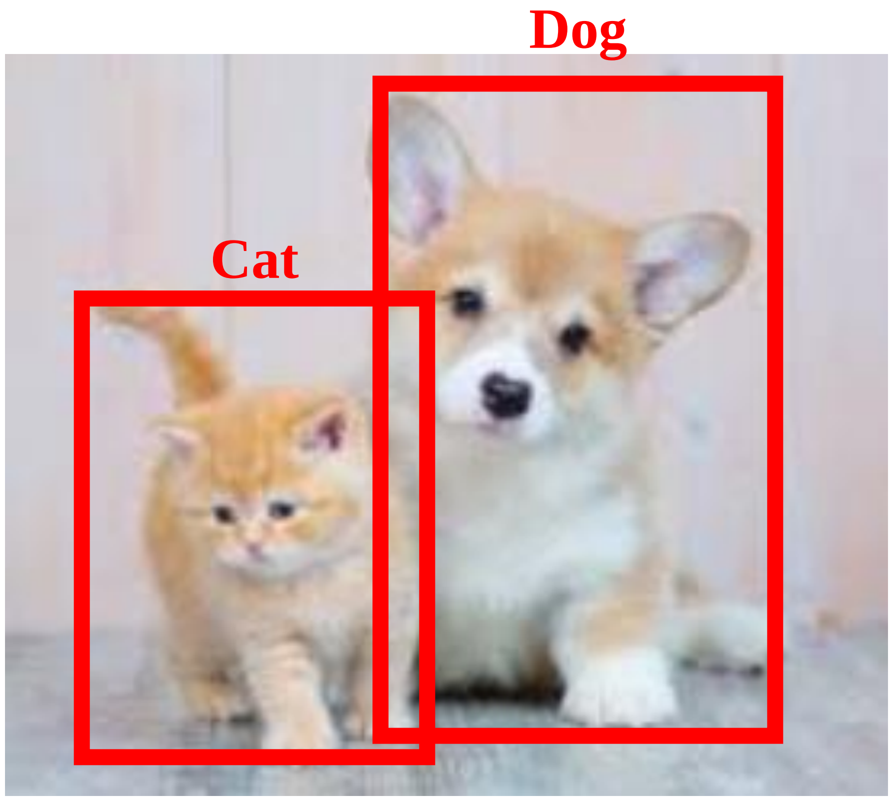
- Object detection is to use an anchor to mark the range of the object in image content such as photos or videos, and classify it into what kind of object it is and the degree of confidence of the attached model in this object
- The most popular and famous object detection model currently is YOLO.



3.1 YOLO(You Only Look Once)

What is ObjectDetection?

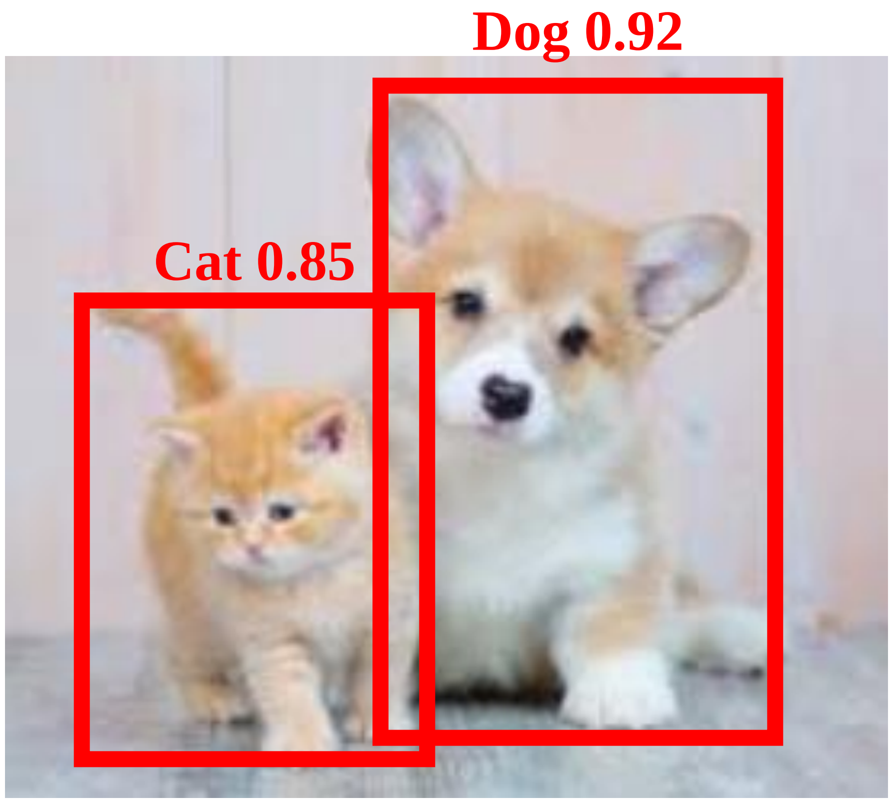
- Object detection is to use an anchor to mark the range of the object in image content such as photos or videos, and classify it into what kind of object it is and the degree of confidence of the attached model in this object
- The most popular and famous object detection model currently is YOLO.



3.1 YOLO(You Only Look Once)

What is ObjectDetection?

- Object detection is to use an anchor to mark the range of the object in image content such as photos or videos, and classify it into what kind of object it is and the degree of confidence of the attached model in this object
- The most popular and famous object detection model currently is YOLO.



3.1 YOLO(You Only Look Once)

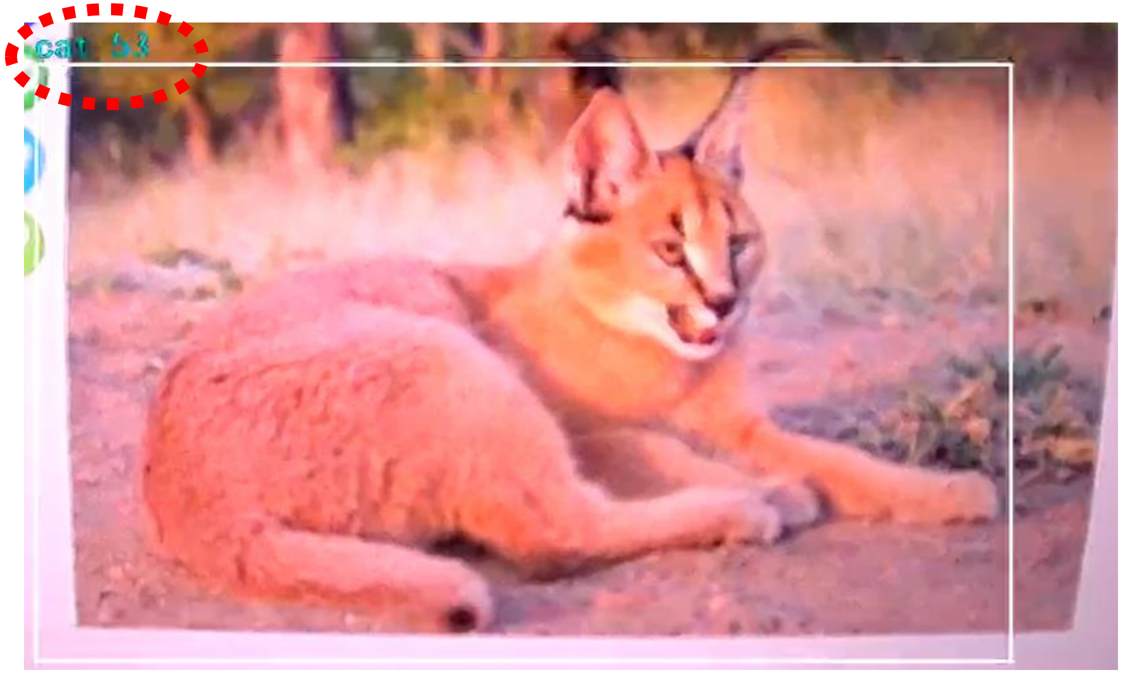
What is ObjectDetection?

- Object detection is to use an anchor to mark the range of the object in image content such as photos or videos, and classify it into what kind of object it is and the degree of confidence of the attached model in this object
- The most popular and famous object detection model currently is YOLO.

3.1 YOLO(You Only Look Once)

What is ObjectDetection?

- Object detection is to use an anchor to mark the range of the object in image content such as photos or videos, and classify it into what kind of object it is and the degree of confidence of the attached model in this object
- The most popular and famous object detection model currently is YOLO.



YOLO(You Only Look Once)

INPUT

640 pixel

640 pixel

YOLO(You Only Look Once)

INPUT

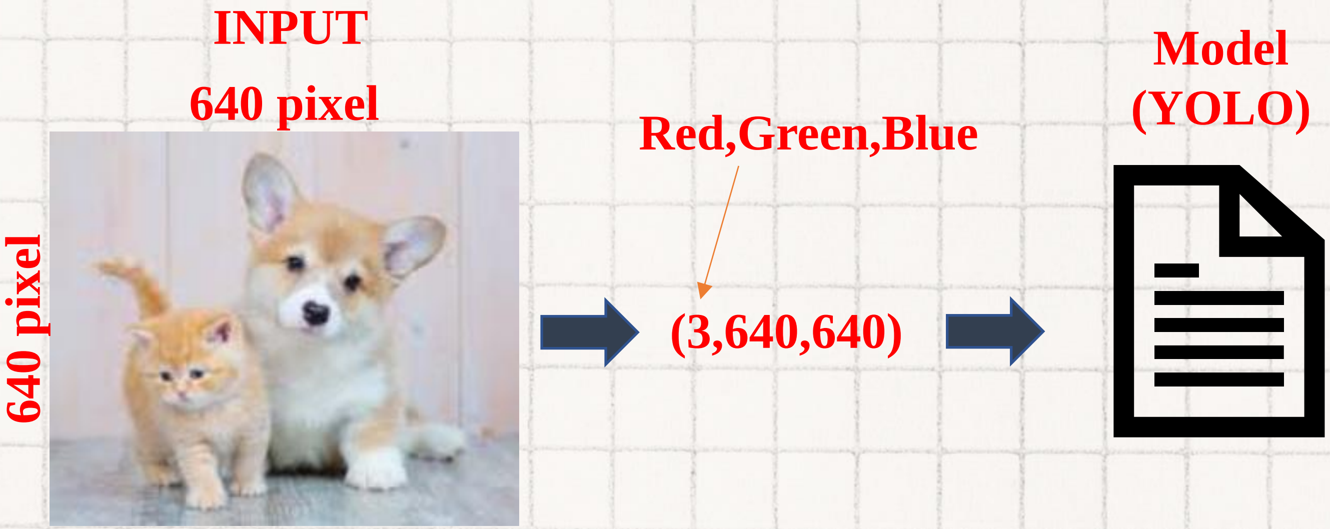
640 pixel

640 pixel

Red,Green,Blue

(3,640,640)

YOLO(You Only Look Once)



INPUT

640 pixel

640 pixel

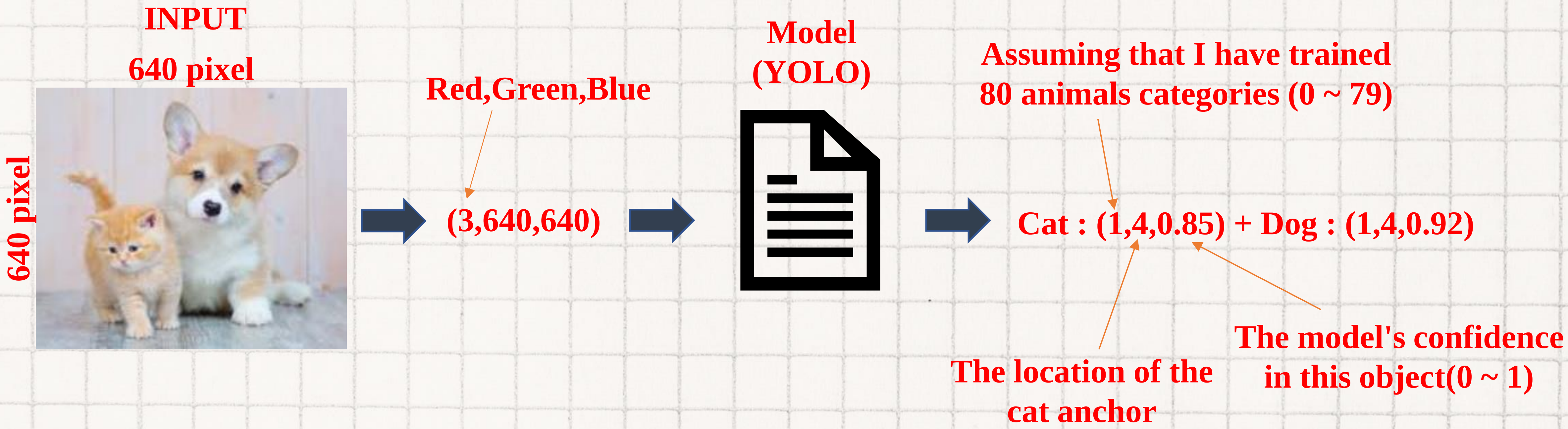
Red,Green,Blue

(3,640,640)

Model

(YOLO)

YOLO(You Only Look Once)



INPUT

640 pixel

640 pixel

Red,Green,Blue

(3,640,640)

Model
(YOLO)

Assuming that I have trained
80 animals categories (0 ~ 79)

Cat : (1,4,0.85) + Dog : (1,4,0.92)

The location of the
cat anchor

The model's confidence
in this object(0 ~ 1)

YOLO(You Only Look Once)

INPUT

640 pixel

640 pixel

Red,Green,Blue

(3,640,640)

Model

(YOLO)

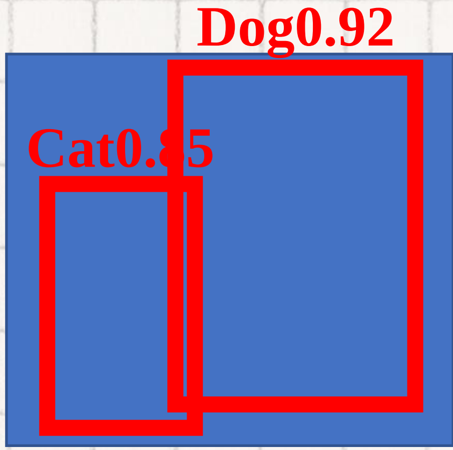
Cat : (1,4,0.85) + Dog : (1,4,0.92)

Assuming that I have trained 80 animals categories (0 ~ 79)

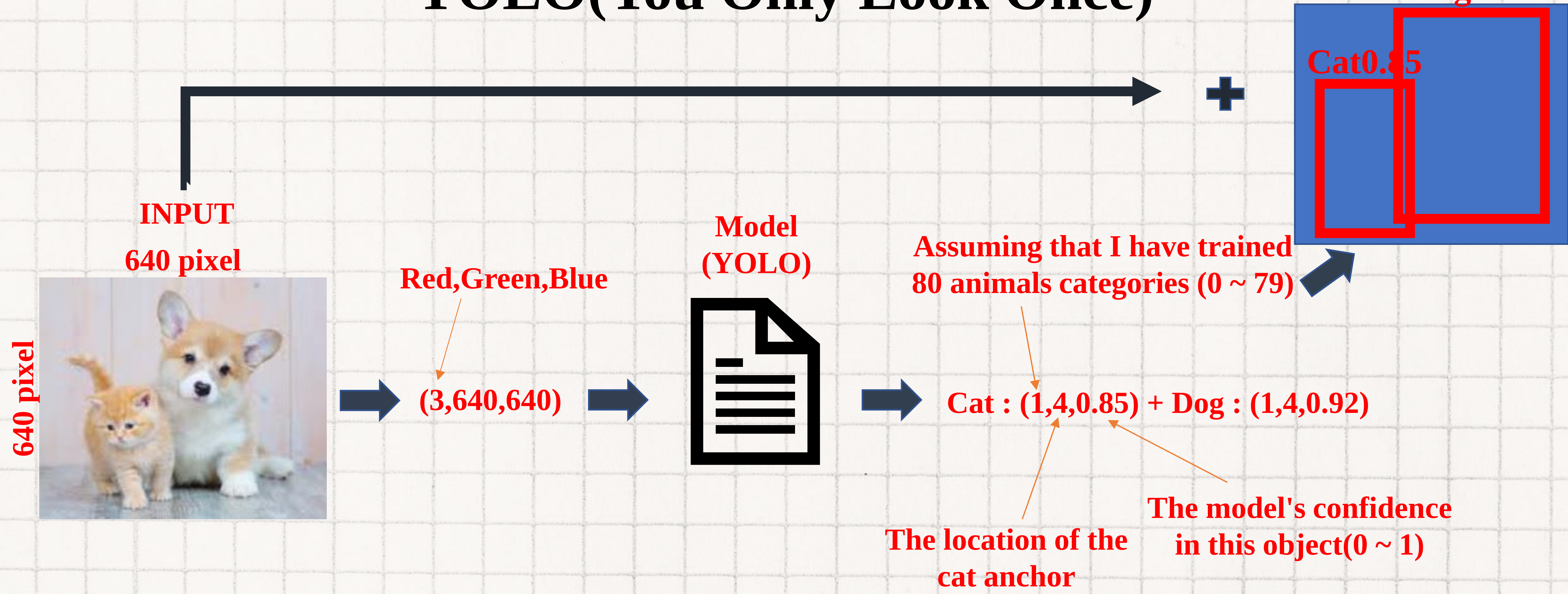
The location of the cat anchor

The model's confidence in this object(0 ~ 1)

OUTPUT



YOLO(You Only Look Once)



OUTPUT

Dog0.92

Cat0.85

INPUT

640 pixel

640 pixel

Red,Green,Blue

(3,640,640)

Model

(YOLO)

Assuming that I have trained 80 animals categories (0 ~ 79)

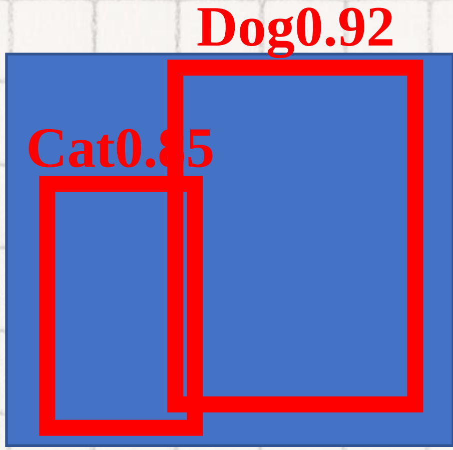
Cat : (1,4,0.85) + Dog : (1,4,0.92)

The location of the cat anchor

The model's confidence in this object(0 ~ 1)

OUTPUT

YOLO(You Only Look Once)



INPUT

640 pixel

640 pixel

Red,Green,Blue

(3,640,640)

Model
(YOLO)

Assuming that I have trained
80 animals categories (0 ~ 79)

Cat : (1,4,0.85) + Dog : (1,4,0.92)

The location of the
cat anchor

The model's confidence
in this object(0 ~ 1)

YOLO(You Only Look Once)

INPUT

640 pixel

640 pixel

Red,Green,Blue

(3,640,640)

Model
(YOLO)

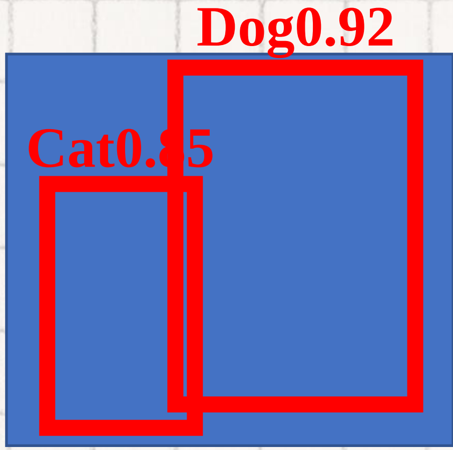
Assuming that I have trained
80 animals categories (0 ~ 79)

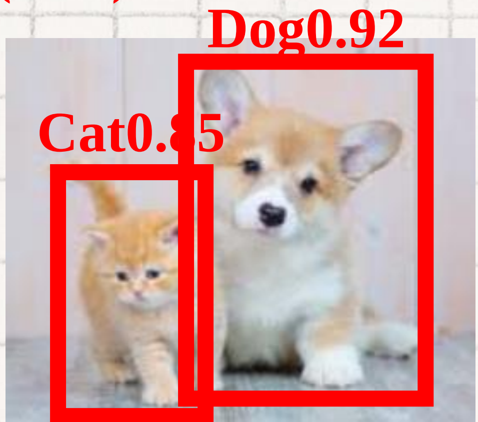
Cat : (1,4,0.85) + Dog : (1,4,0.92)

The location of the
cat anchor

The model's confidence
in this object(0 ~ 1)

OUTPUT





3.1 YOLO(You Only Look Once)

Video of YOLO

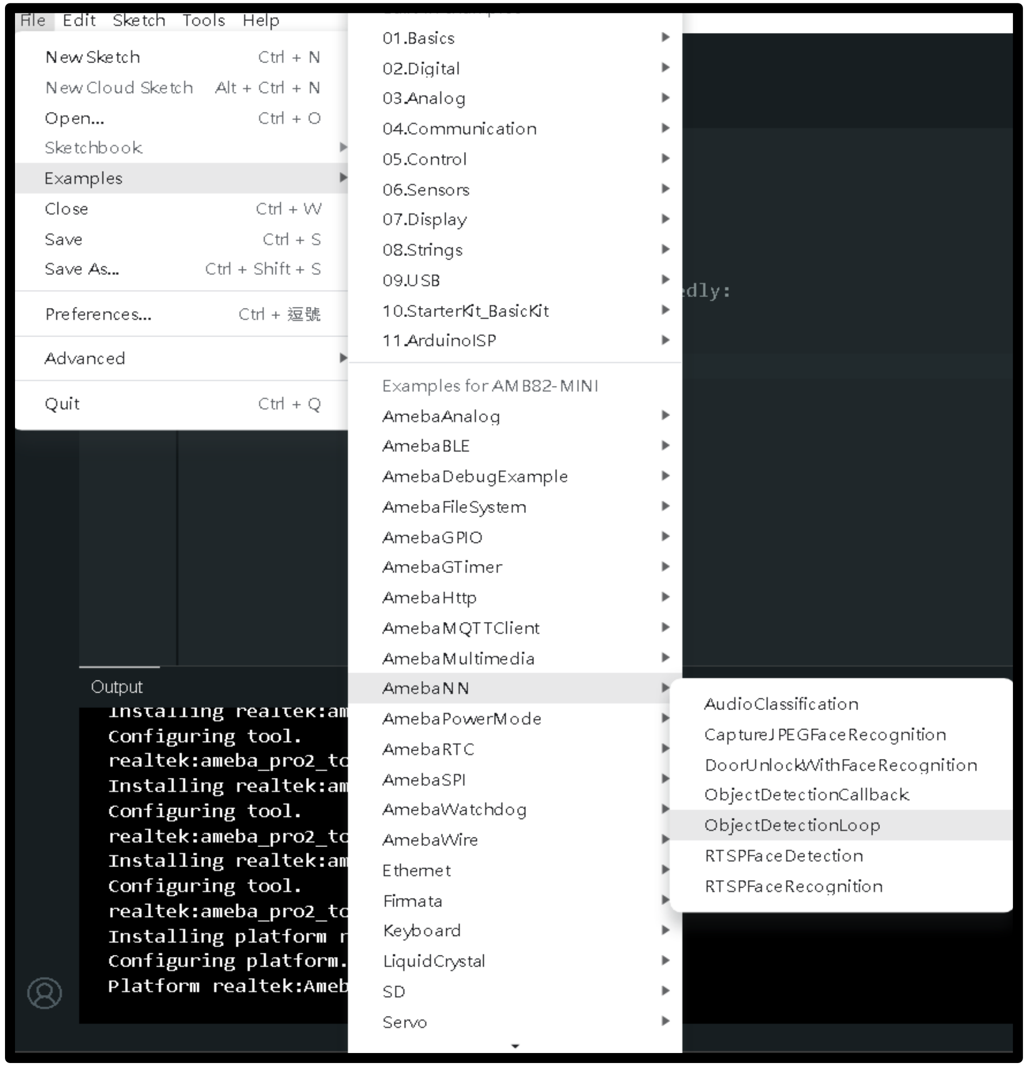
Implementation

3.1 YOLO(You Only Look Once)

Step 1.

Follow the path below in Arduino IDE to open the example.

1. **File**
2. **Examples**
3. **AmebaNN**
4. **ObjectDetectionLoop**



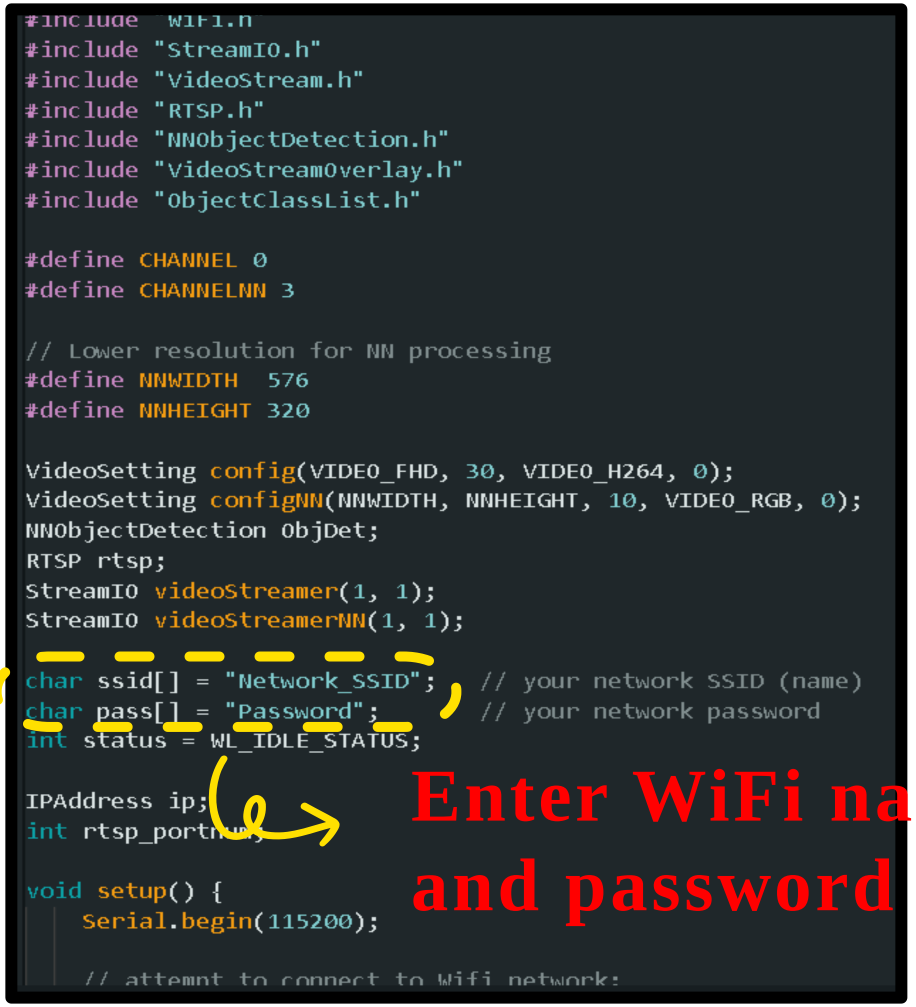
```
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing realtek:am
Configuring tool.
realtek:ameba_pro2_to
Installing platform r
Configuring platform.
Platform realtek:Ameb
```

3.1 YOLO(You Only Look Once)

Step 2.

Enter the WiFi name and password

to the corresponding place in the code.

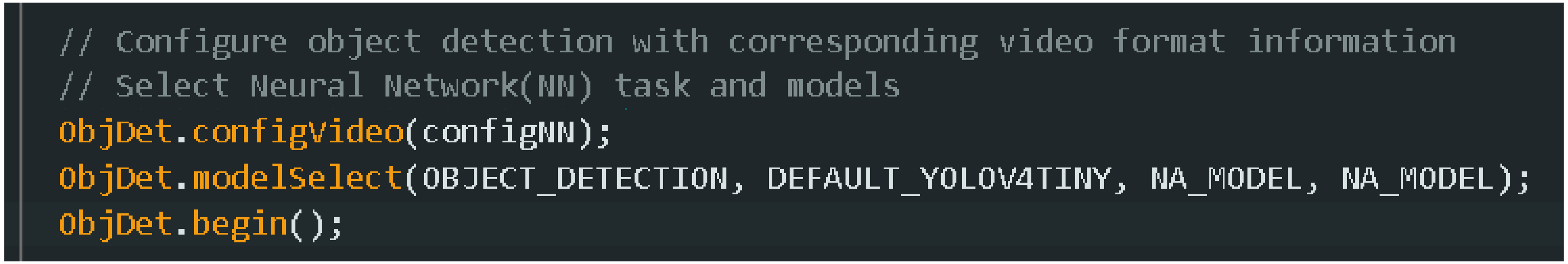


```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNObjectDetection.h"
#include "VideoStreamOverlay.h"
#include "ObjectClassList.h"

#define CHANNEL 0
#define CHANNELNN 3

// Lower resolution for NN processing
#define NNWIDTH  576
#define NNHEIGHT 320

VideoSetting config(VIDEO_FHD, 30, VIDEO_H264, 0);
VideoSetting configNN(NNWIDTH, NNHEIGHT, 10, VIDEO_RGB, 0);
NNObjectDetection ObjDet;
RTSP rtsp;
StreamIO videoStreamer(1, 1);
StreamIO videoStreamerNN(1, 1);

char ssid[] = "Network_SSID";    // your network SSID (name)
char pass[] = "Password";        // your network password
int status = WL_IDLE_STATUS;

IPAddress ip;
int rtsp_portnum;

void setup() {
    Serial.begin(115200);

    // attempt to connect to Wifi network:
```

Enter WiFi name and password

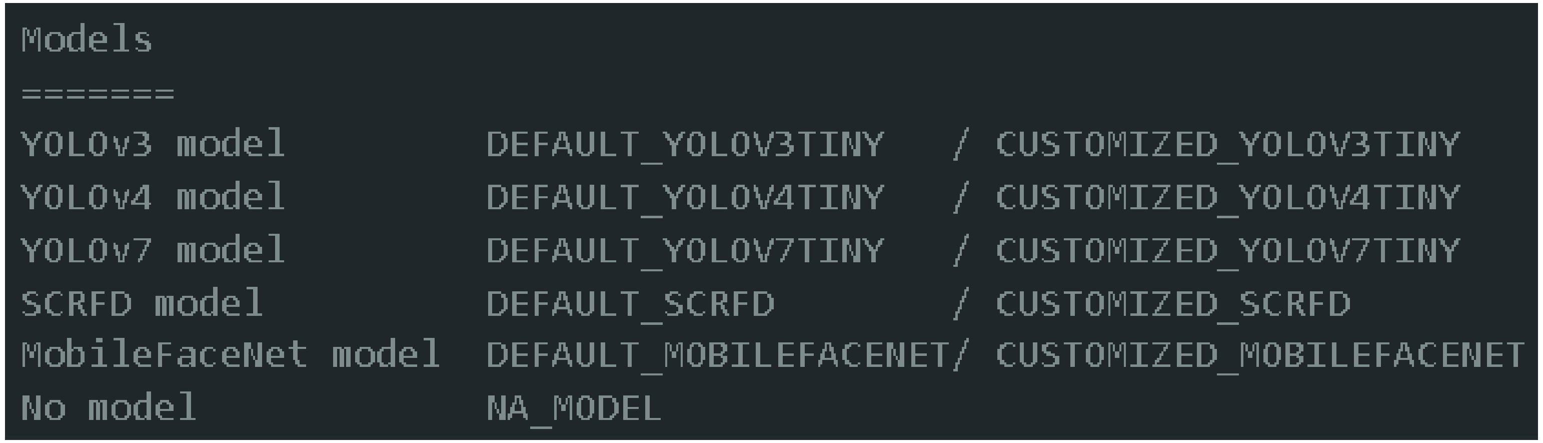
3.1 YOLO(You Only Look Once)

Step 3. Model choosing(optional)



```
// Configure object detection with corresponding video format information
// Select Neural Network(NN) task and models
ObjDet.configVideo(configNN);
ObjDet.modelSelect(OBJECT_DETECTION, DEFAULT_YOLOV4TINY, NA_MODEL, NA_MODEL);
ObjDet.begin();
```

List of models for different tasks

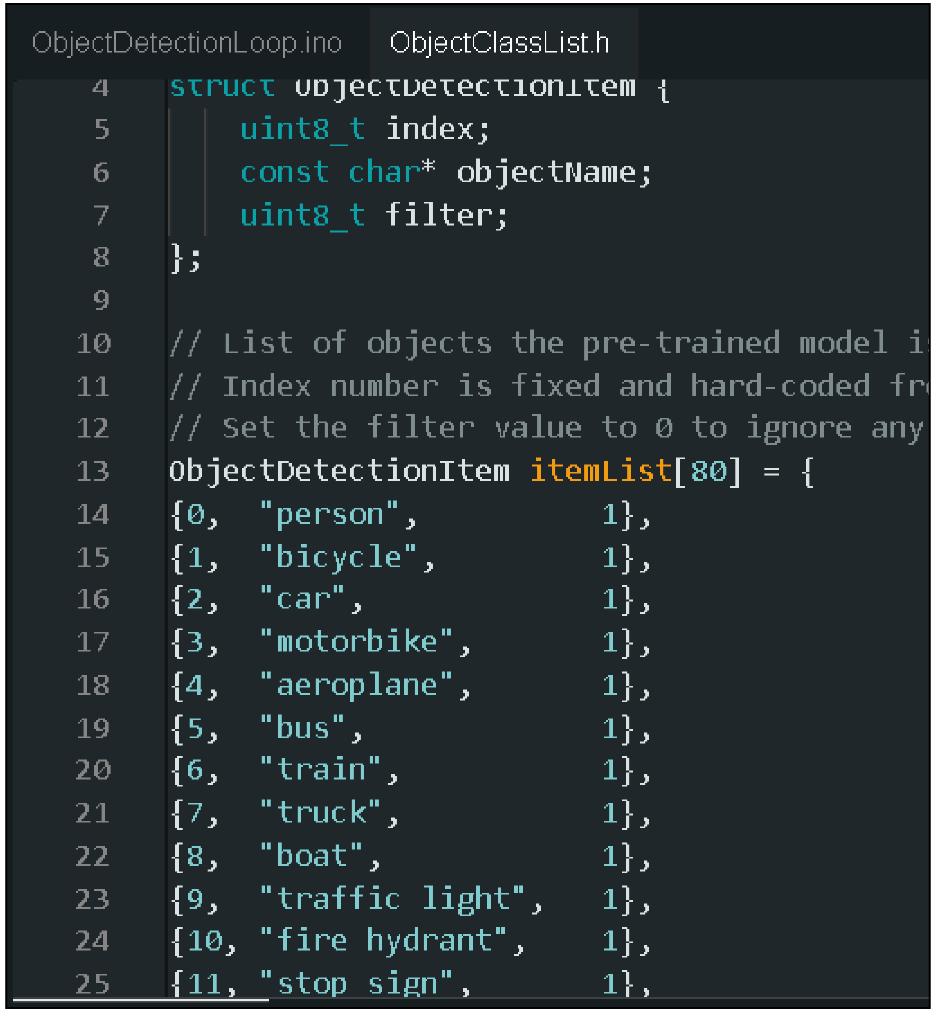


```
Models
=======
YOLOv3 model          DEFAULT_YOLOV3TINY    / CUSTOMIZED_YOLOV3TINY
YOLOv4 model          DEFAULT_YOLOV4TINY    / CUSTOMIZED_YOLOV4TINY
YOLOv7 model          DEFAULT_YOLOV7TINY    / CUSTOMIZED_YOLOV7TINY
SCRFD model           DEFAULT_SCRFD         / CUSTOMIZED_SCRFD
MobileFaceNet model   DEFAULT_MOBILEFACENET/ CUSTOMIZED_MOBILEFACENET
No model              NA_MODEL
```

3.1 YOLO(You Only Look Once)

- The pre-trained model can recognize a total of 80 different types of objects.

- To disable the recognition of certain object, set the filter to 0.



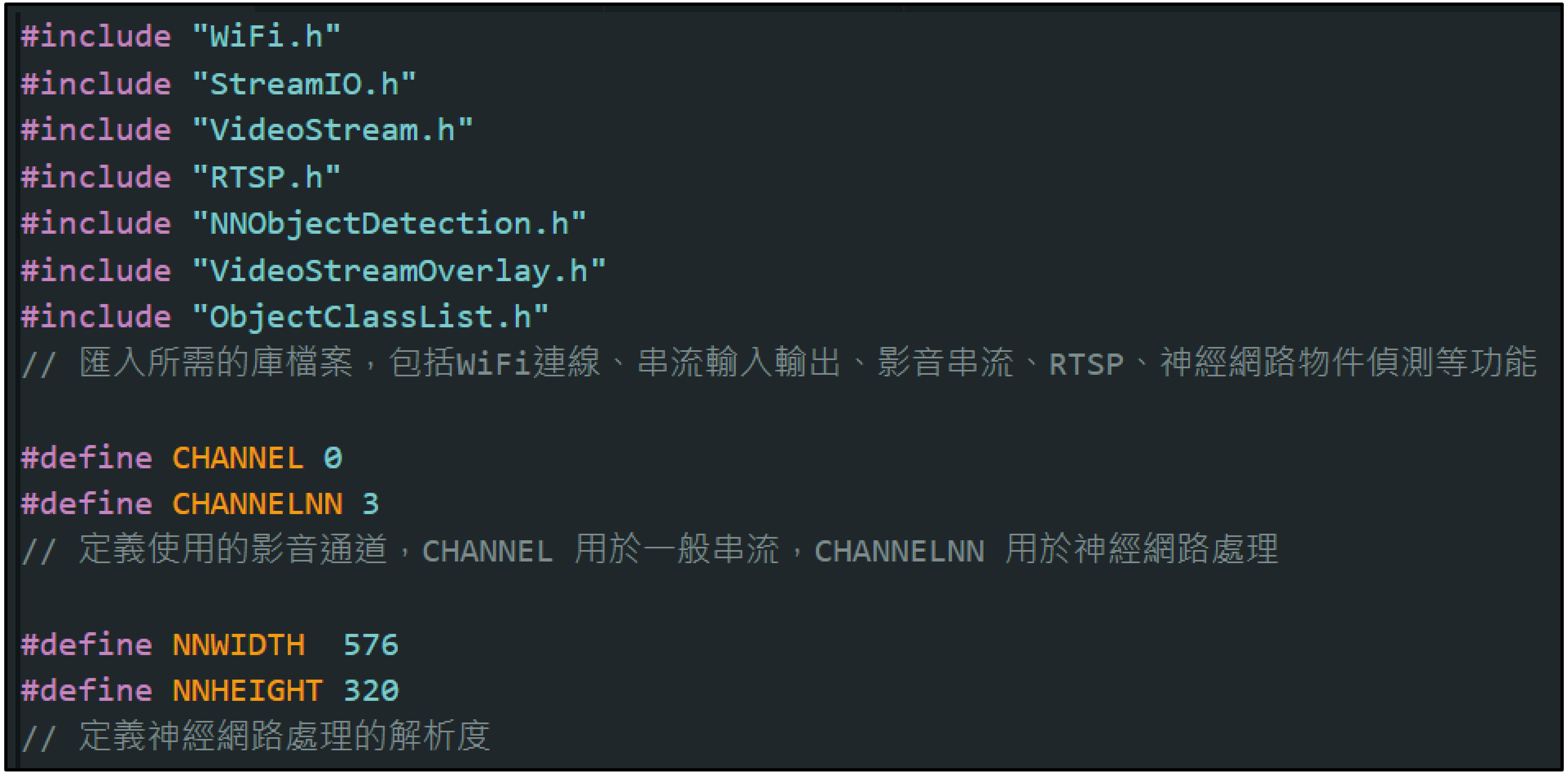
```
struct ObjectDetectionItem {
    uint8_t index;
    const char* objectName;
    uint8_t filter;
};

// List of objects the pre-trained model i
// Index number is fixed and hard-coded fr
// Set the filter value to 0 to ignore any
ObjectDetectionItem itemList[80] = {
{0,  "person",         1},
{1,  "bicycle",        1},
{2,  "car",            1},
{3,  "motorbike",      1},
{4,  "aeroplane",      1},
{5,  "bus",            1},
{6,  "train",          1},
{7,  "truck",          1},
{8,  "boat",           1},
{9,  "traffic light",  1},
{10, "fire hydrant",   1},
{11, "stop sign",      1},
```

Program Explanation

3.1 YOLO(You Only Look Once)

include

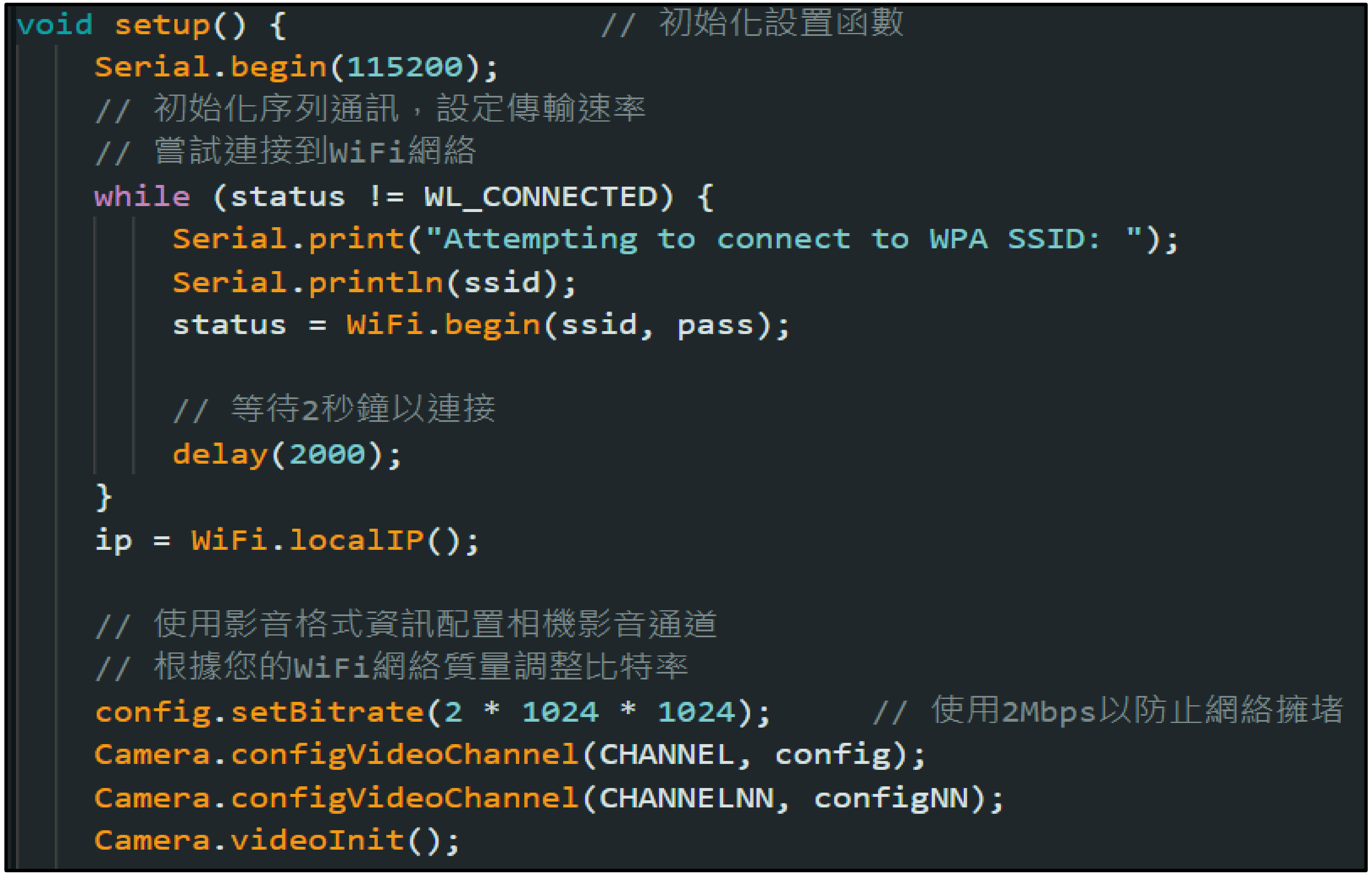
3.1 YOLO(You Only Look Once)



```
#include "WiFi.h"
#include "StreamIO.h"
#include "VideoStream.h"
#include "RTSP.h"
#include "NNObjectDetection.h"
#include "VideoStreamOverlay.h"
#include "ObjectClassList.h"
// 匯入所需的庫檔案，包括WiFi連線、串流輸入輸出、影音串流、RTSP、神經網路物件偵測等功能

#define CHANNEL 0
#define CHANNELNN 3
// 定義使用的影音通道，CHANNEL 用於一般串流，CHANNELNN 用於神經網路處理

#define NNWIDTH  576
#define NNHEIGHT 320
// 定義神經網路處理的解析度
```

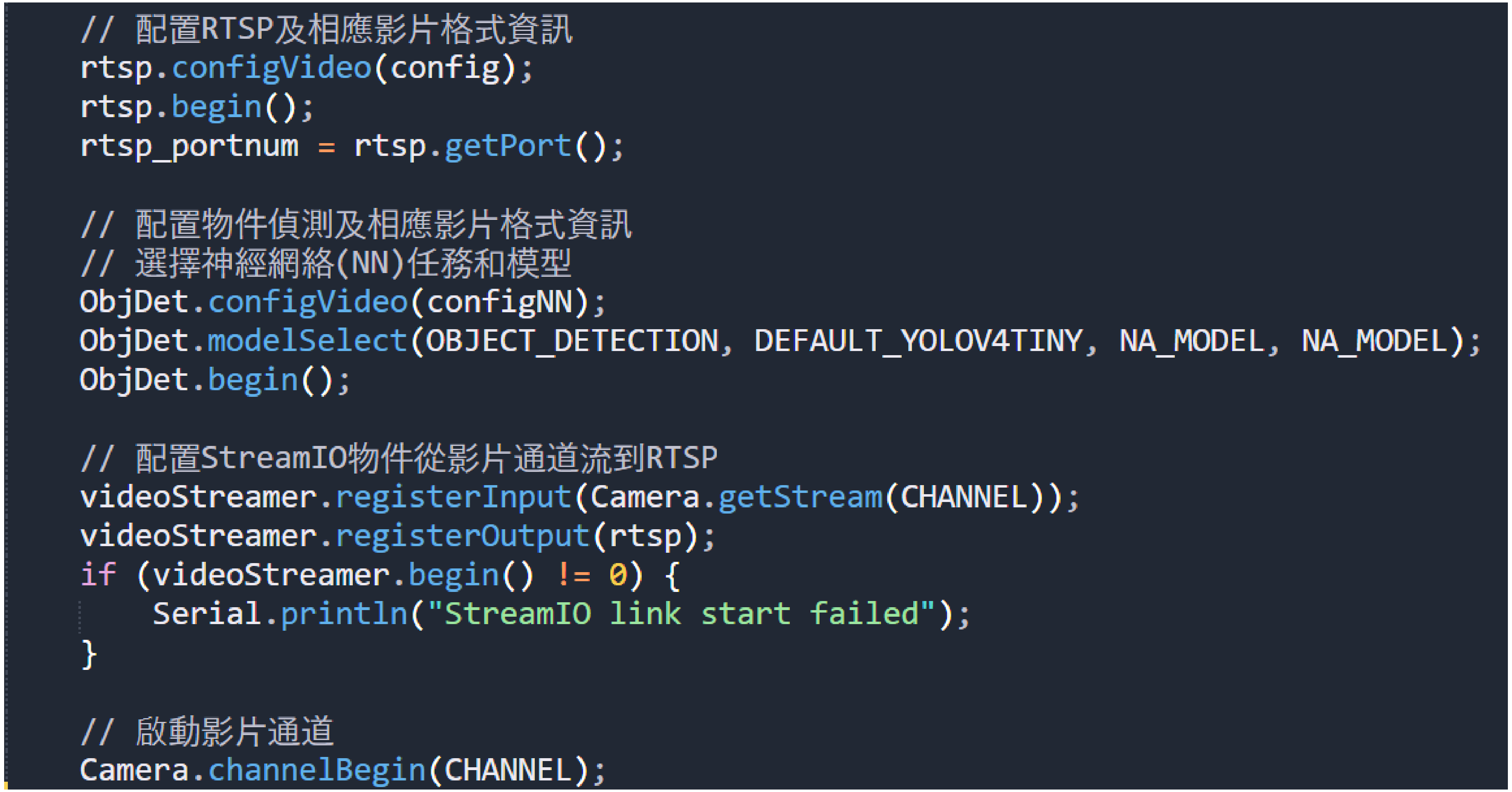
setup()

3.1 YOLO(You Only Look Once)



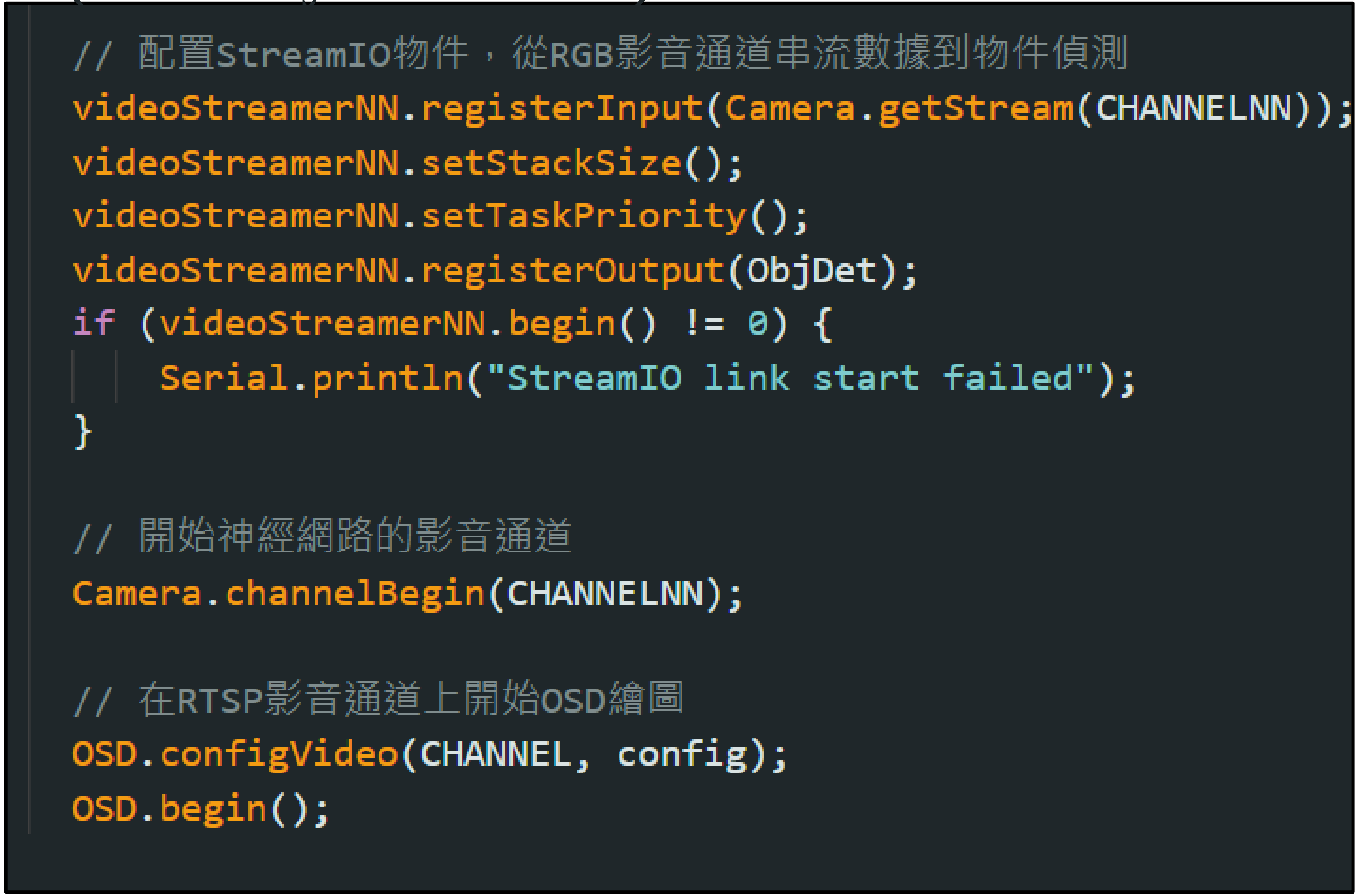
```
void setup() {                // 初始化設置函數
    Serial.begin(115200);
    // 初始化序列通訊，設定傳輸速率
    // 嘗試連接到WiFi網絡
    while (status != WL_CONNECTED) {
        Serial.print("Attempting to connect to WPA SSID: ");
        Serial.println(ssid);
        status = WiFi.begin(ssid, pass);

        // 等待2秒鐘以連接
        delay(2000);
    }
    ip = WiFi.localIP();

    // 使用影音格式資訊配置相機影音通道
    // 根據您的WiFi網絡質量調整比特率
    config.setBitrate(2 * 1024 * 1024);     // 使用2Mbps以防止網絡擁堵
    Camera.configVideoChannel(CHANNEL, config);
    Camera.configVideoChannel(CHANNELNN, configNN);
    Camera.videoInit();
```

3.1 YOLO(You Only Look Once)



```
// 配置RTSP及相應影片格式資訊
rtsp.configVideo(config);
rtsp.begin();
rtsp_portnum = rtsp.getPort();

// 配置物件偵測及相應影片格式資訊
// 選擇神經網絡(NN)任務和模型
ObjDet.configVideo(configNN);
ObjDet.modelSelect(OBJECT_DETECTION, DEFAULT_YOLOV4TINY, NA_MODEL, NA_MODEL);
ObjDet.begin();

// 配置StreamIO物件從影片通道流到RTSP
videoStreamer.registerInput(Camera.getStream(CHANNEL));
videoStreamer.registerOutput(rtsp);
if (videoStreamer.begin() != 0) {
    Serial.println("StreamIO link start failed");
}

// 啟動影片通道
Camera.channelBegin(CHANNEL);
```

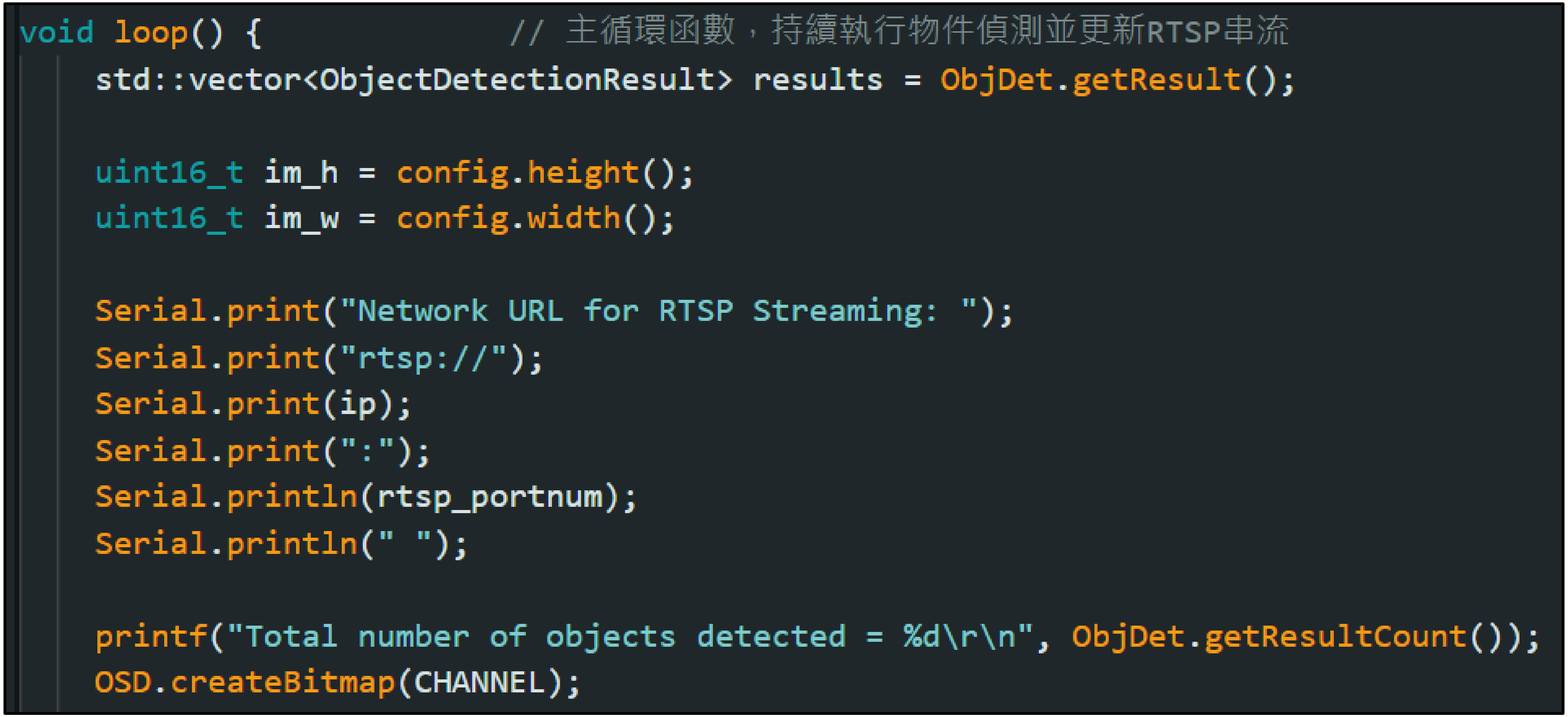
3.1 YOLO(You Only Look Once)



```
// 配置StreamIO物件，從RGB影音通道串流數據到物件偵測
videoStreamerNN.registerInput(Camera.getStream(CHANNELNN));
videoStreamerNN.setStackSize();
videoStreamerNN.setTaskPriority();
videoStreamerNN.registerOutput(ObjDet);
if (videoStreamerNN.begin() != 0) {
    Serial.println("StreamIO link start failed");
}

// 開始神經網路的影音通道
Camera.channelBegin(CHANNELNN);

// 在RTSP影音通道上開始OSD繪圖
OSD.configVideo(CHANNEL, config);
OSD.begin();
```

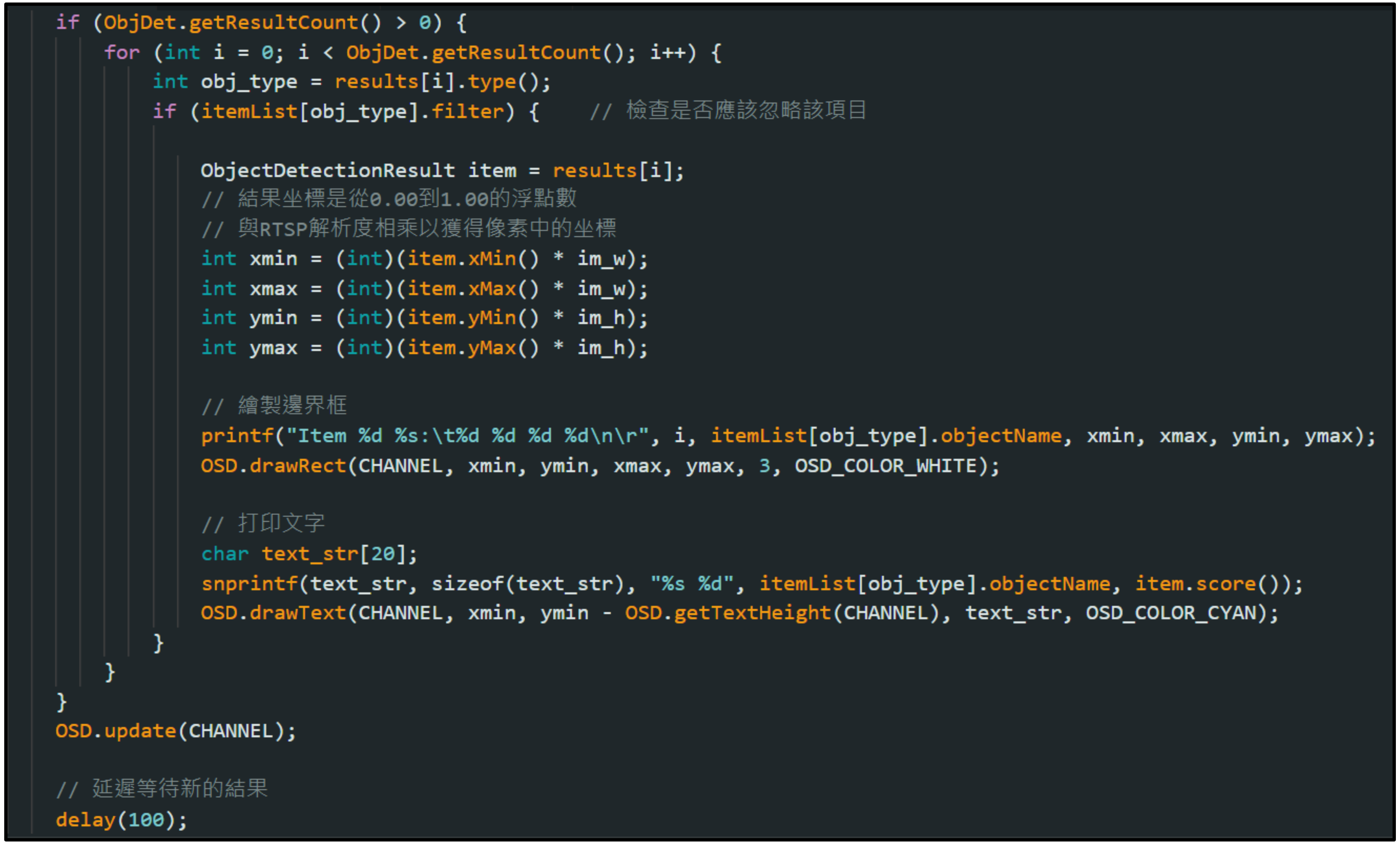
loop()

3.1 YOLO(You Only Look Once)



```
void loop() {                // 主循環函數，持續執行物件偵測並更新RTSP串流
    std::vector<ObjectDetectionResult> results = ObjDet.getResult();

    uint16_t im_h = config.height();
    uint16_t im_w = config.width();

    Serial.print("Network URL for RTSP Streaming: ");
    Serial.print("rtsp://");
    Serial.print(ip);
    Serial.print(":");
    Serial.println(rtsp_portnum);
    Serial.println(" ");

    printf("Total number of objects detected = %d\r\n", ObjDet.getResultCount());
    OSD.createBitmap(CHANNEL);
```

3.1 YOLO(You Only Look Once)



```
if (ObjDet.getResultCount() > 0) {
    for (int i = 0; i < ObjDet.getResultCount(); i++) {
        int obj_type = results[i].type();
        if (itemList[obj_type].filter) {    // 檢查是否應該忽略該項目

            ObjectDetectionResult item = results[i];
            // 結果坐標是從0.00到1.00的浮點數
            // 與RTSP解析度相乘以獲得像素中的坐標
            int xmin = (int)(item.xMin() * im_w);
            int xmax = (int)(item.xMax() * im_w);
            int ymin = (int)(item.yMin() * im_h);
            int ymax = (int)(item.yMax() * im_h);

            // 繪製邊界框
            printf("Item %d %s:\t%d %d %d %d\n\r", i, itemList[obj_type].objectName, xmin, xmax, ymin, ymax);
            OSD.drawRect(CHANNEL, xmin, ymin, xmax, ymax, 3, OSD_COLOR_WHITE);

            // 打印文字
            char text_str[20];
            snprintf(text_str, sizeof(text_str), "%s %d", itemList[obj_type].objectName, item.score());
            OSD.drawText(CHANNEL, xmin, ymin - OSD.getTextHeight(CHANNEL), text_str, OSD_COLOR_CYAN);
        }
    }
}
OSD.update(CHANNEL);

// 延遲等待新的結果
delay(100);
```

Advanced implementation

(Using customized model)

4 0.69

3.1 YOLO(You Only Look Once)

Comparison

The following table compares the computing power of AMB82-MINI and RTX 3090.

Table 1.Comparison of Computing Power

	TOPS(Tera Operations Per Second)
RTX 3090	285
AMB82-MINI	0.4

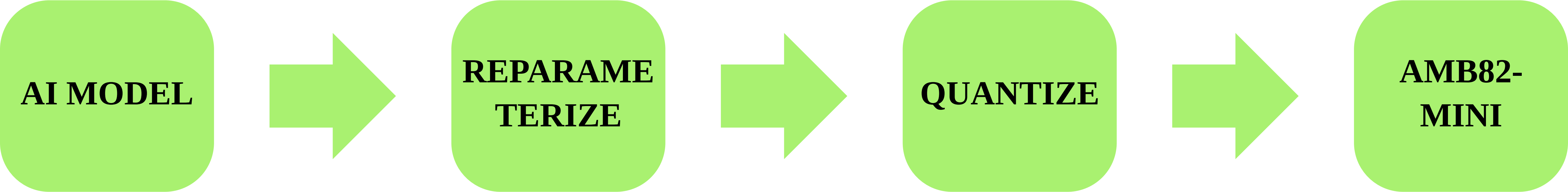
Comparison

The following table compares the capacity of AMB82-MINI and YOLOv7_TINY.

Table 1.Comparison of Capacity

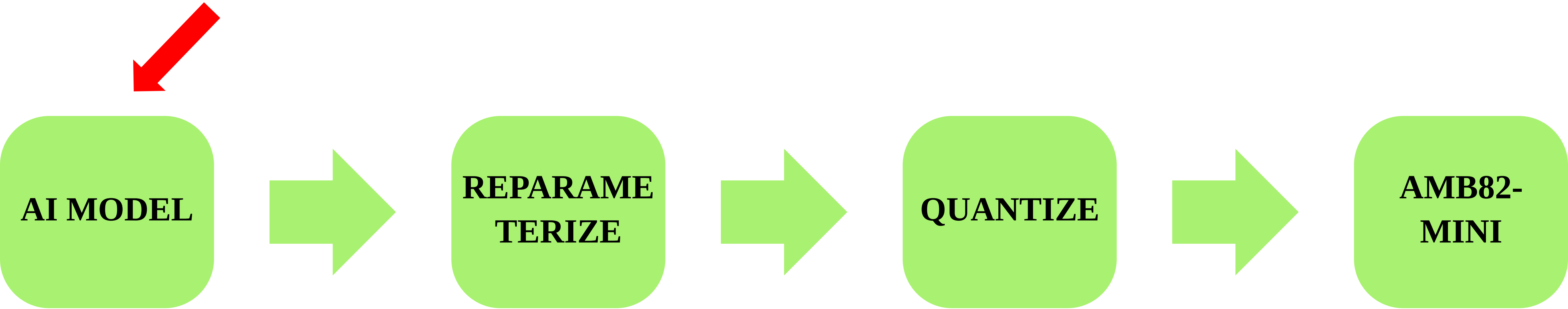
	MB(Megabyte)
YOLOv7_tiny	23
AMB82-MINI	16

3.1 YOLO(You Only Look Once)



AI MODEL → REPARAMETERIZE → QUANTIZE → AMB82-MINI

3.1 YOLO(You Only Look Once)



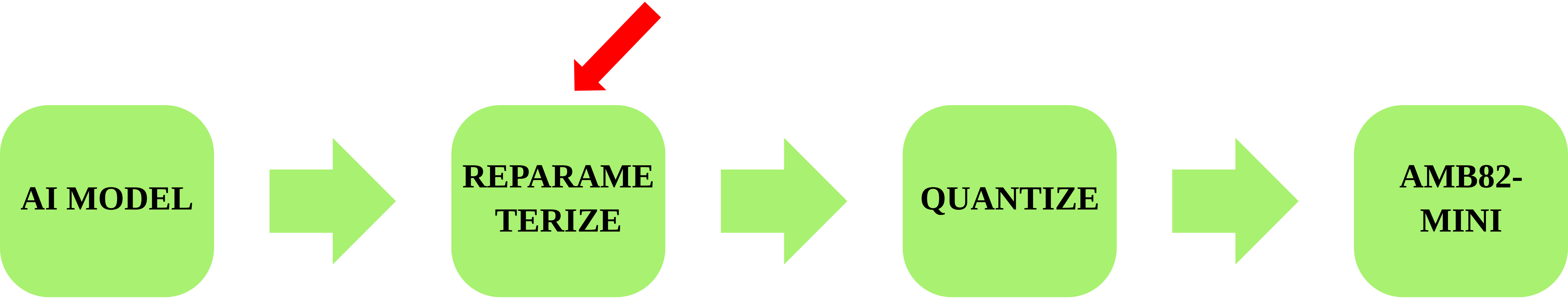
AI MODEL

REPARAMETERIZE

QUANTIZE

AMB82-MINI

3.1 YOLO(You Only Look Once)



AI MODEL

REPARAMETERIZE

QUANTIZE

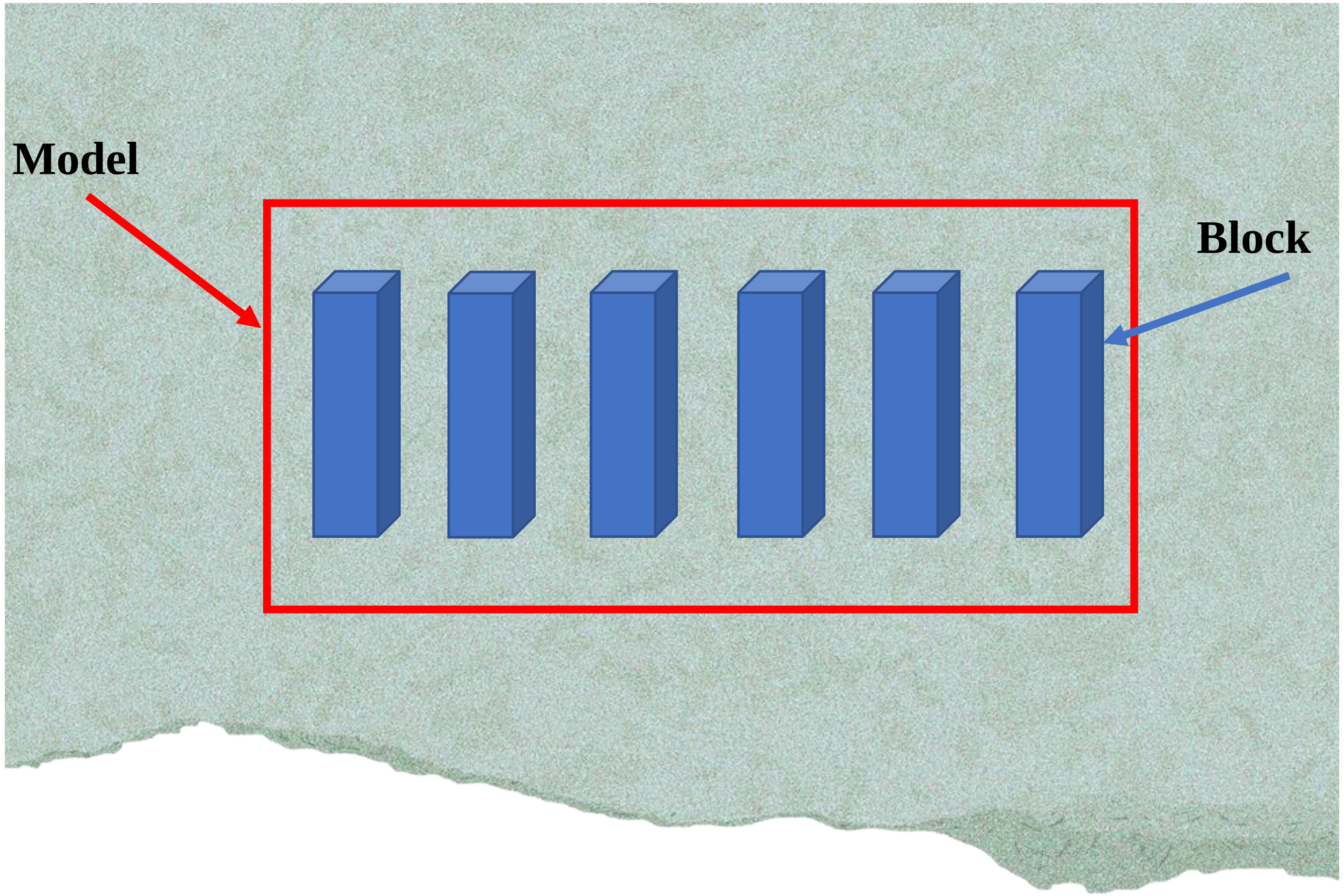
AMB82-MINI

3.1 YOLO(You Only Look Once)

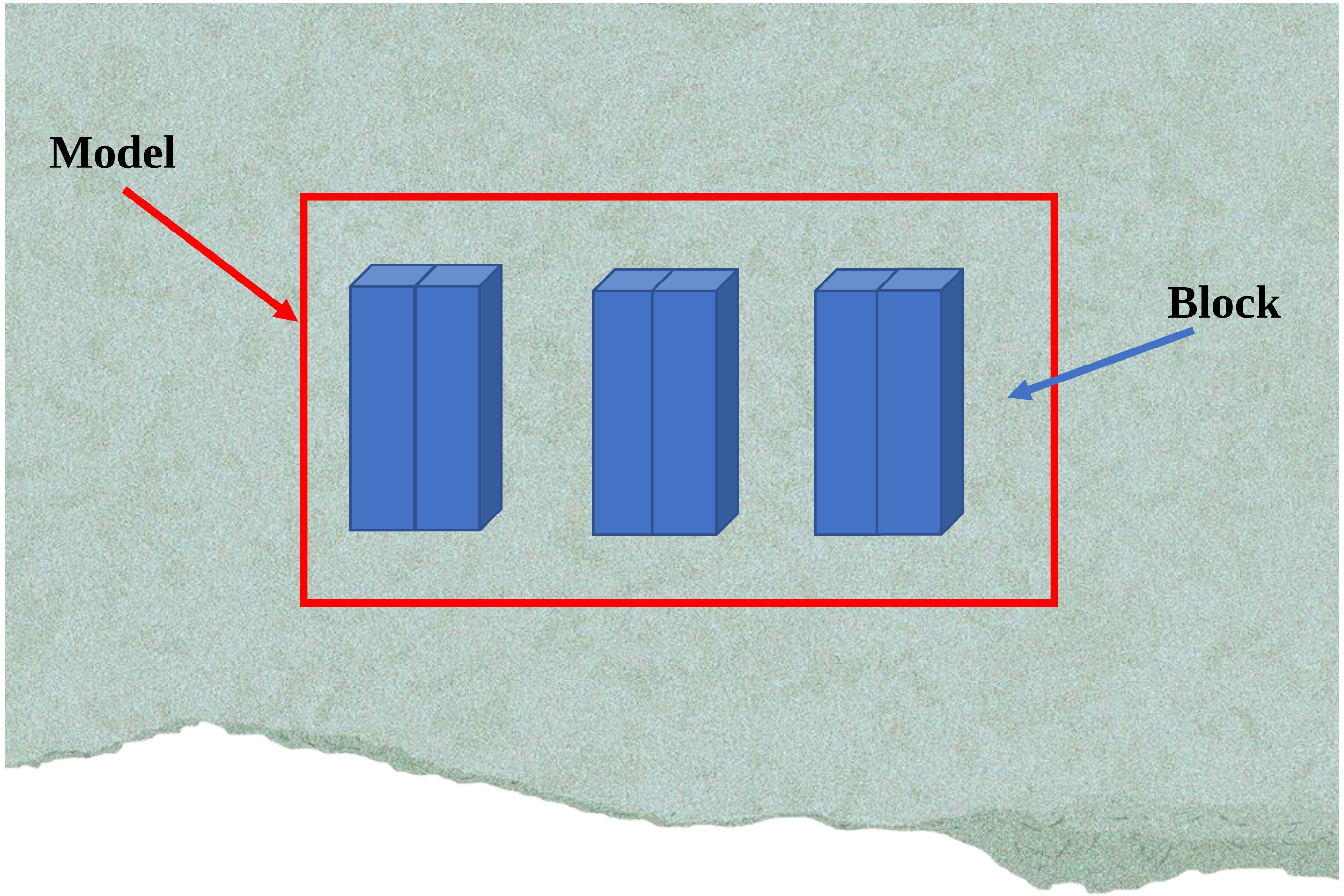
Reparameterize

Definition： By merging multiple blocks and simplifying branches, the model reduces parameters to enhance computational performance. The original model is used solely for training, while only the re-parameterized version is saved and deployed for inference.

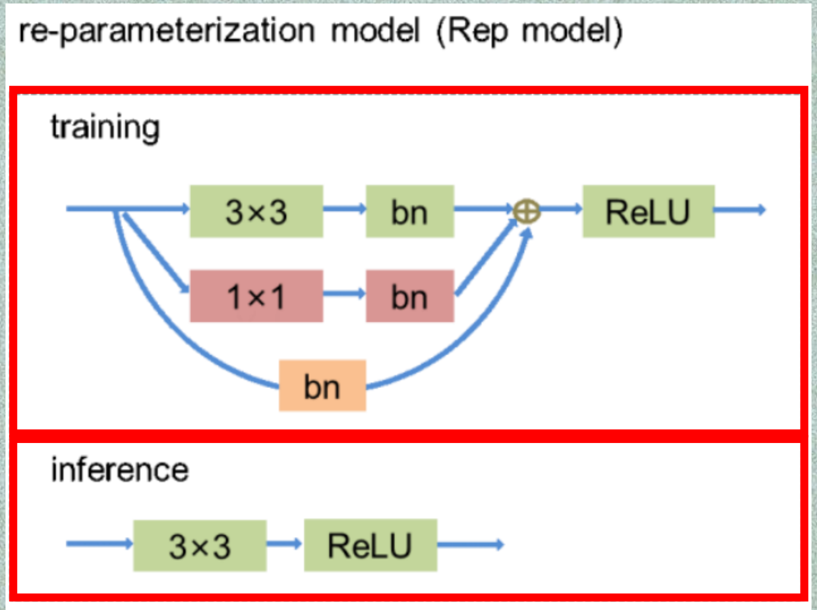
Reparameterize



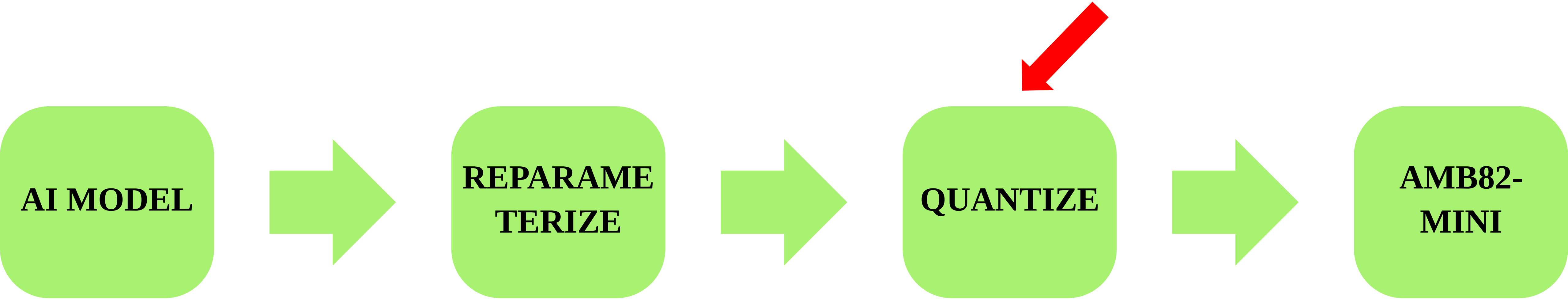
Reparameterize



Reparameterize



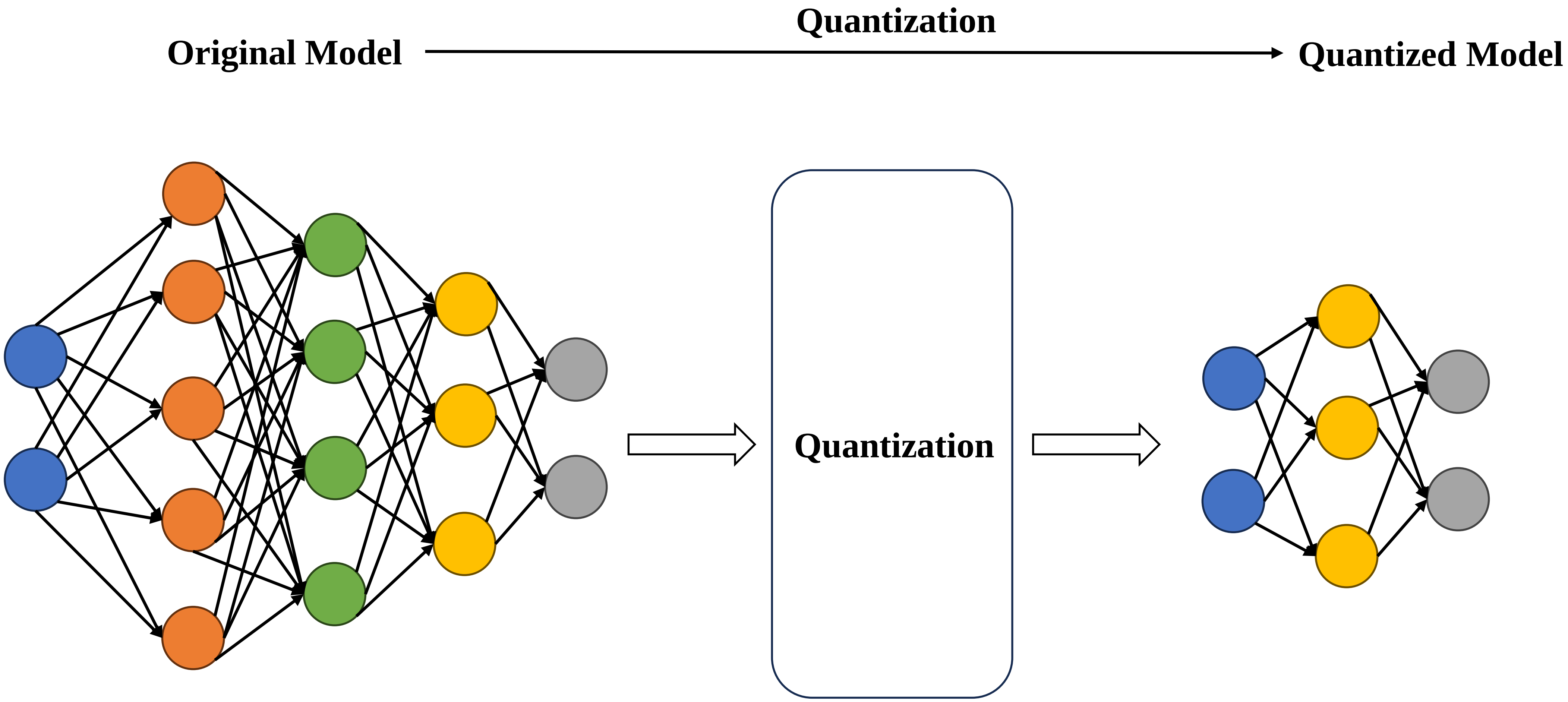
3.1 YOLO(You Only Look Once)



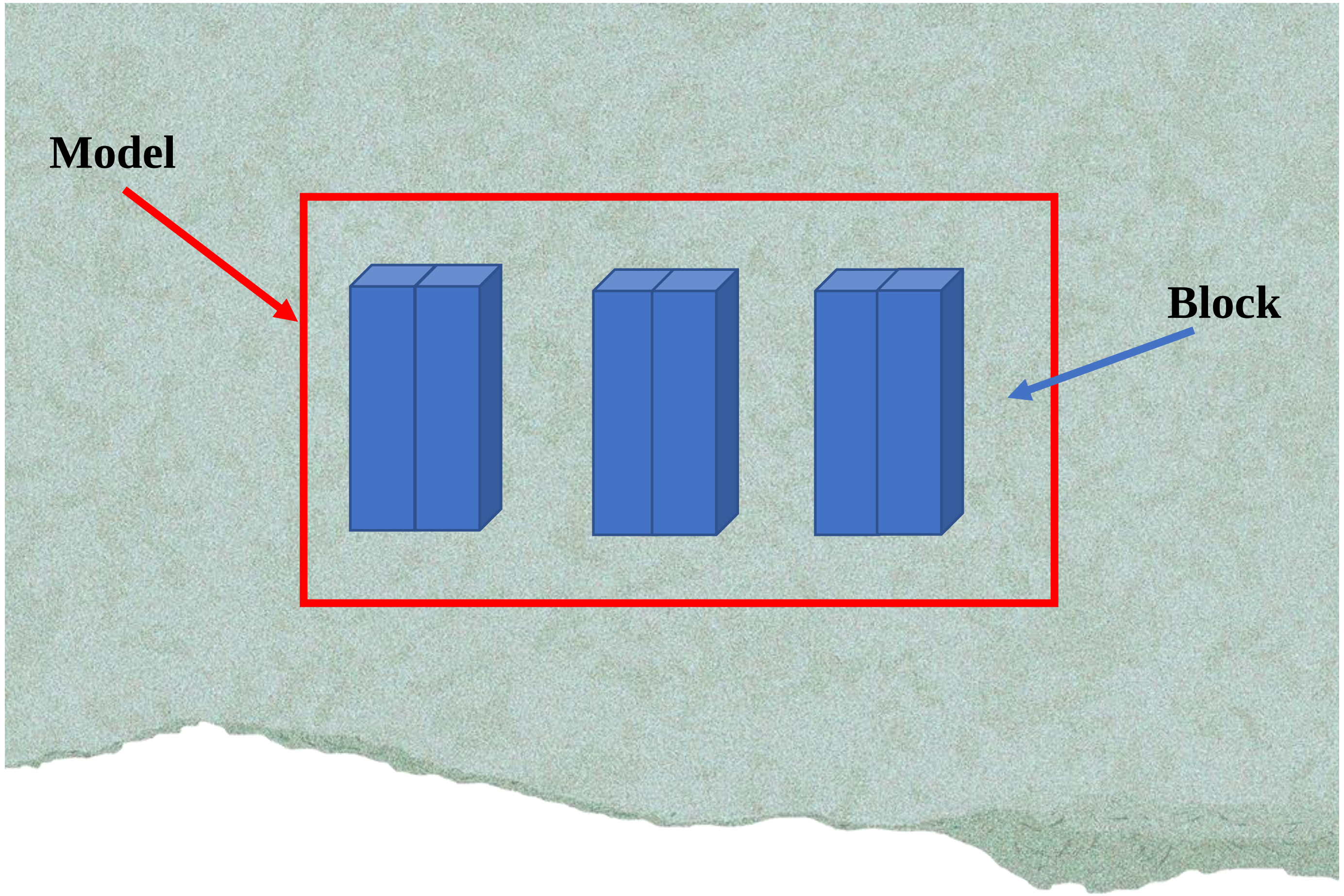
Quantization

- **Definition**： Convert high-precision parameters to low-precision to significantly reduce model size and computational complexity, improving inference speed and efficiency, making it suitable for resource-limited environments like mobile devices.

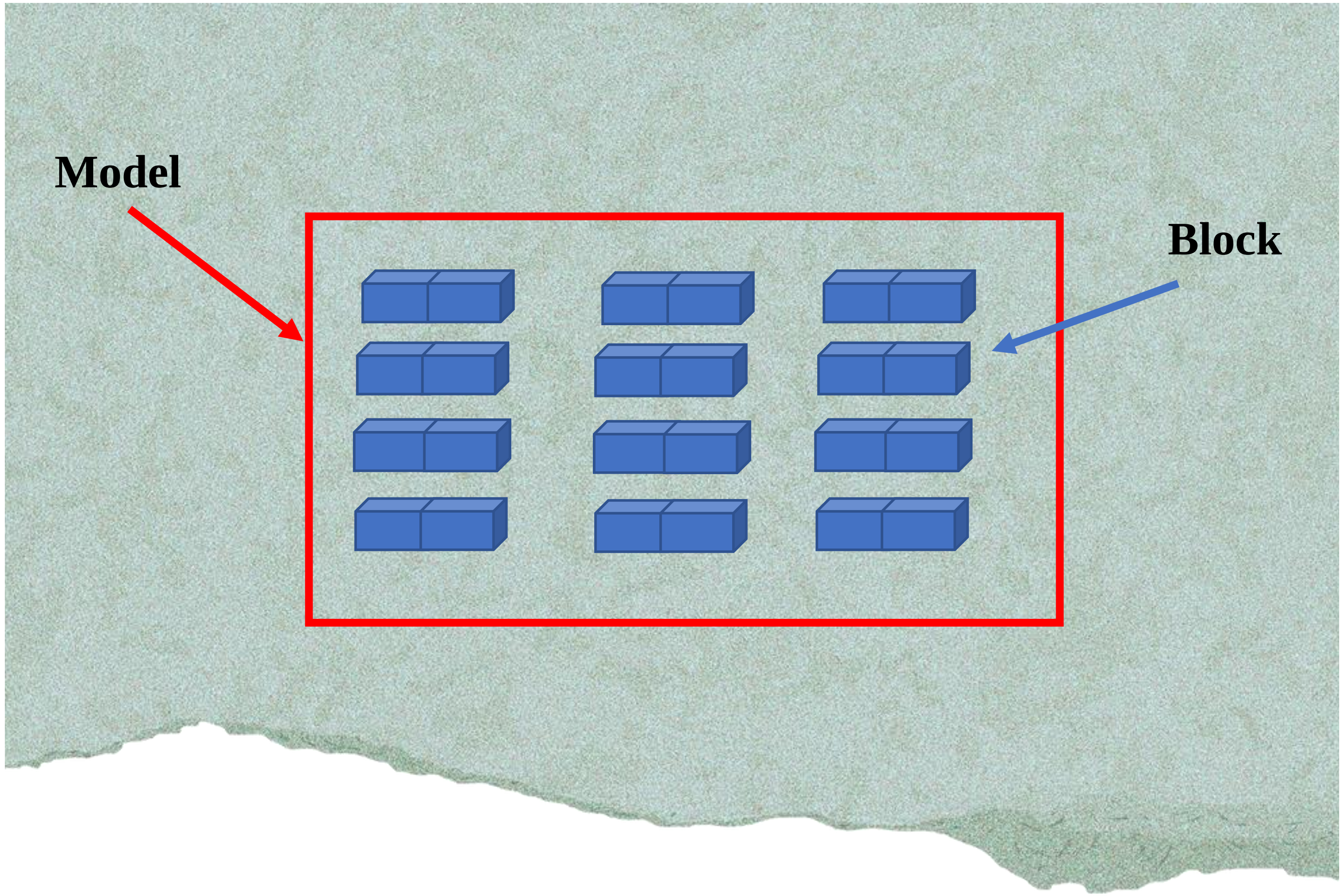
3.1 YOLO(You Only Look Once)



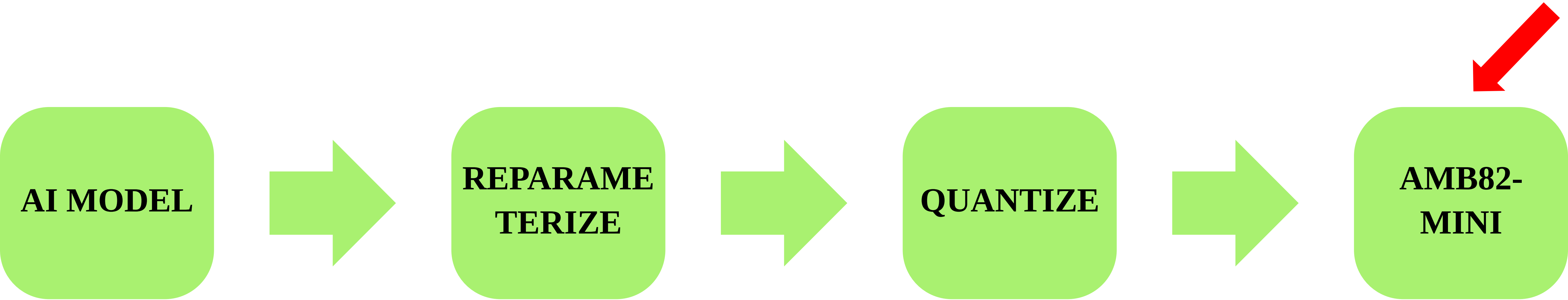
Reparameterize



Quantization



3.1 YOLO(You Only Look Once)

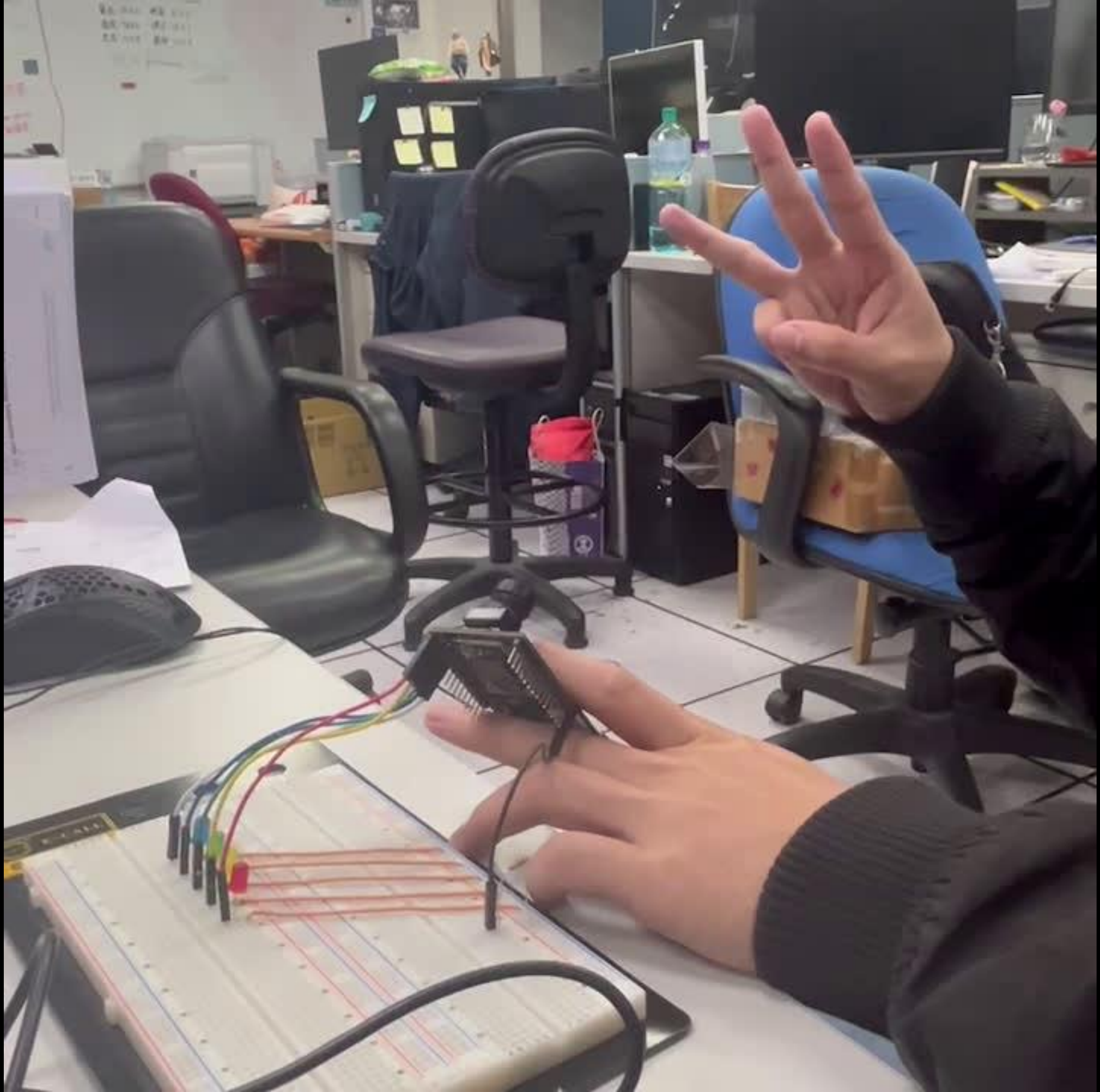


AI MODEL

REPARAMETERIZE

QUANTIZE

AMB82-MINI

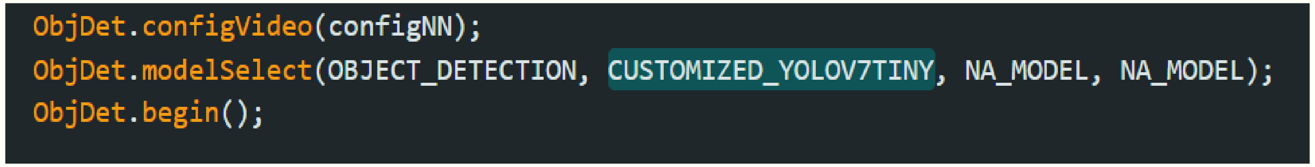


3.1 YOLO(You Only Look Once)

Programming

Switch the model from Default model to Customized model.

Results will look like



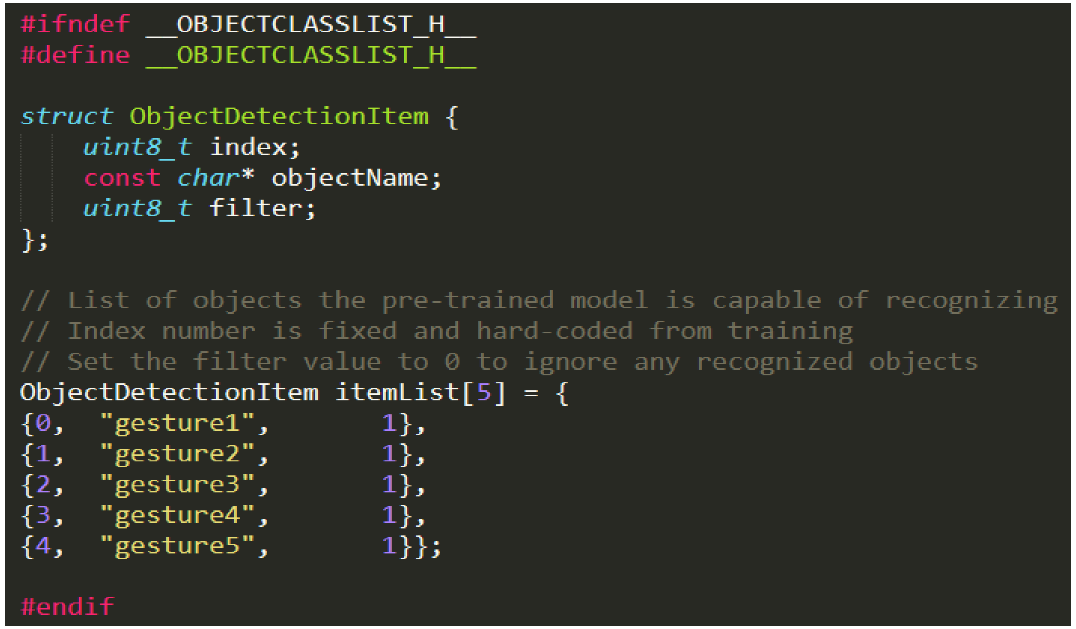
```
ObjDet.configVideo(configNN);
ObjDet.modelSelect(OBJECT_DETECTION, CUSTOMIZED_YOLOV7TINY, NA_MODEL, NA_MODEL);
ObjDet.begin();
```

3.1 YOLO(You Only Look Once)

Programming

The head file (.h) must map the categories to the model's output results

Results will look like



```
#ifndef __OBJECTCLASSLIST_H__
#define __OBJECTCLASSLIST_H__

struct ObjectDetectionItem {
    uint8_t index;
    const char* objectName;
    uint8_t filter;
};

// List of objects the pre-trained model is capable of recognizing
// Index number is fixed and hard-coded from training
// Set the filter value to 0 to ignore any recognized objects
ObjectDetectionItem itemList[5] = {
{0,  "gesture1",        1},
{1,  "gesture2",        1},
{2,  "gesture3",        1},
{3,  "gesture4",        1},
{4,  "gesture5",        1}};

#endif
```

3.1 YOLO(You Only Look Once)

Model Uploading

First, download the converted nb file from the link as below

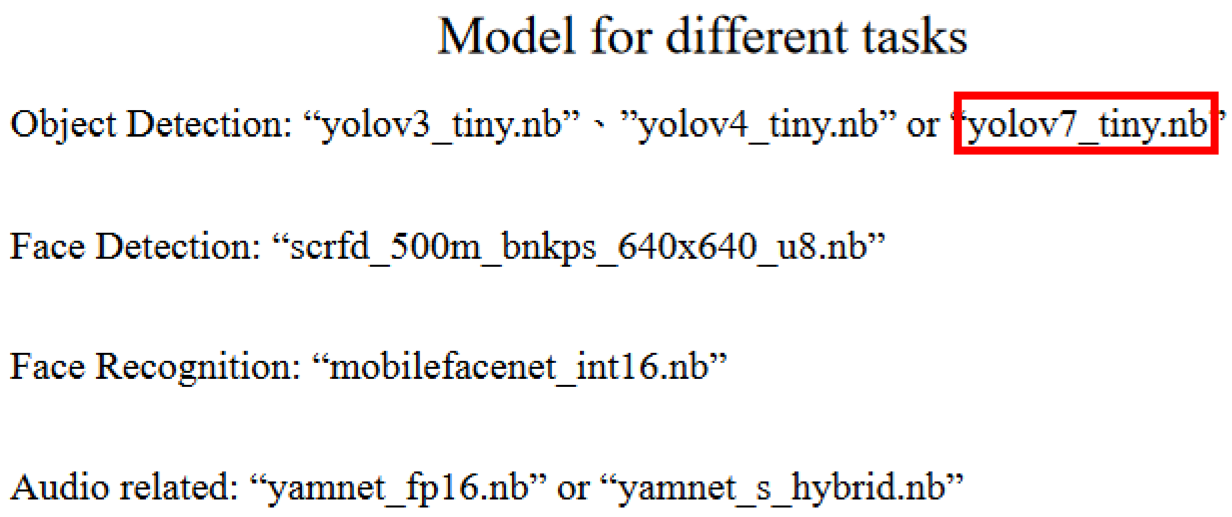
https://drive.google.com/file/d/1Wsa2oWUZ4SdyjZKzTnHtIUJd38ibtltP/view?usp=sharing

3.1 YOLO(You Only Look Once)

Model Uploading

Second, modify the converted nb file to have the same name as the corresponding model.

Corresponding model are shown at below. In our case, change the name to yolov7_tiny.nb.



Model for different tasks

Object Detection: “yolov3_tiny.nb” 、 ”yolov4_tiny.nb” or “yolov7_tiny.nb”

Face Detection: “scrfd_500m_bnkps_640x640_u8.nb”

Face Recognition: “mobilefacenet_int16.nb”

Audio related: “yamnet_fp16.nb” or “yamnet_s_hybrid.nb”

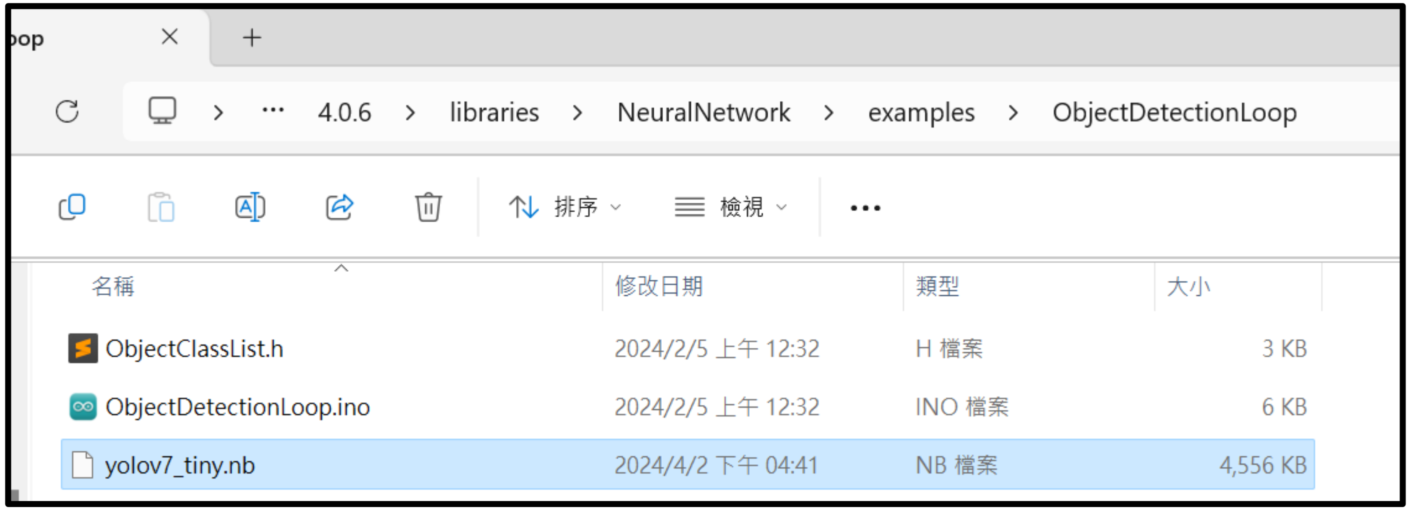
3.1 YOLO(You Only Look Once)

Model Uploading

Finally, find the following path to put the nb file into the folder of the corresponding task

C:\Users\username\AppData\Local\Arduino15\packages\realtek\hardware\AmebaPro2\version\libraries\NeuralNetwork\examples\Corresponding task

Results will look like



… > 4.0.6 > libraries > NeuralNetwork > examples > ObjectDetectionLoop

名稱	修改日期	類型	大小
ObjectClassList.h	2024/2/5 上午 12:32	H 檔案	3 KB
ObjectDetectionLoop.ino	2024/2/5 上午 12:32	INO 檔案	6 KB
yolov7_tiny.nb	2024/4/2 下午 04:41	NB 檔案	4,556 KB

3.1 YOLO(You Only Look Once)

Implementation must include the following three points in the code.

1.Add it at the beginning of the code :

(define the PIN)

```
int gesture1 = 0 ;
int gesture2 = 1 ;
int gesture3 = 2 ;
int gesture4 = 3 ;
int gesture5 = 4 ;
```

3.1 YOLO(You Only Look Once)

Implementation must include the following three points in the code.

2. Add into the function void setup() :

(Give the output to the defined pin)

```
pinMode(gesture1, OUTPUT);
pinMode(gesture2, OUTPUT);
pinMode(gesture3, OUTPUT);
pinMode(gesture4, OUTPUT);
pinMode(gesture5, OUTPUT);
```

3.1 YOLO(You Only Look Once)

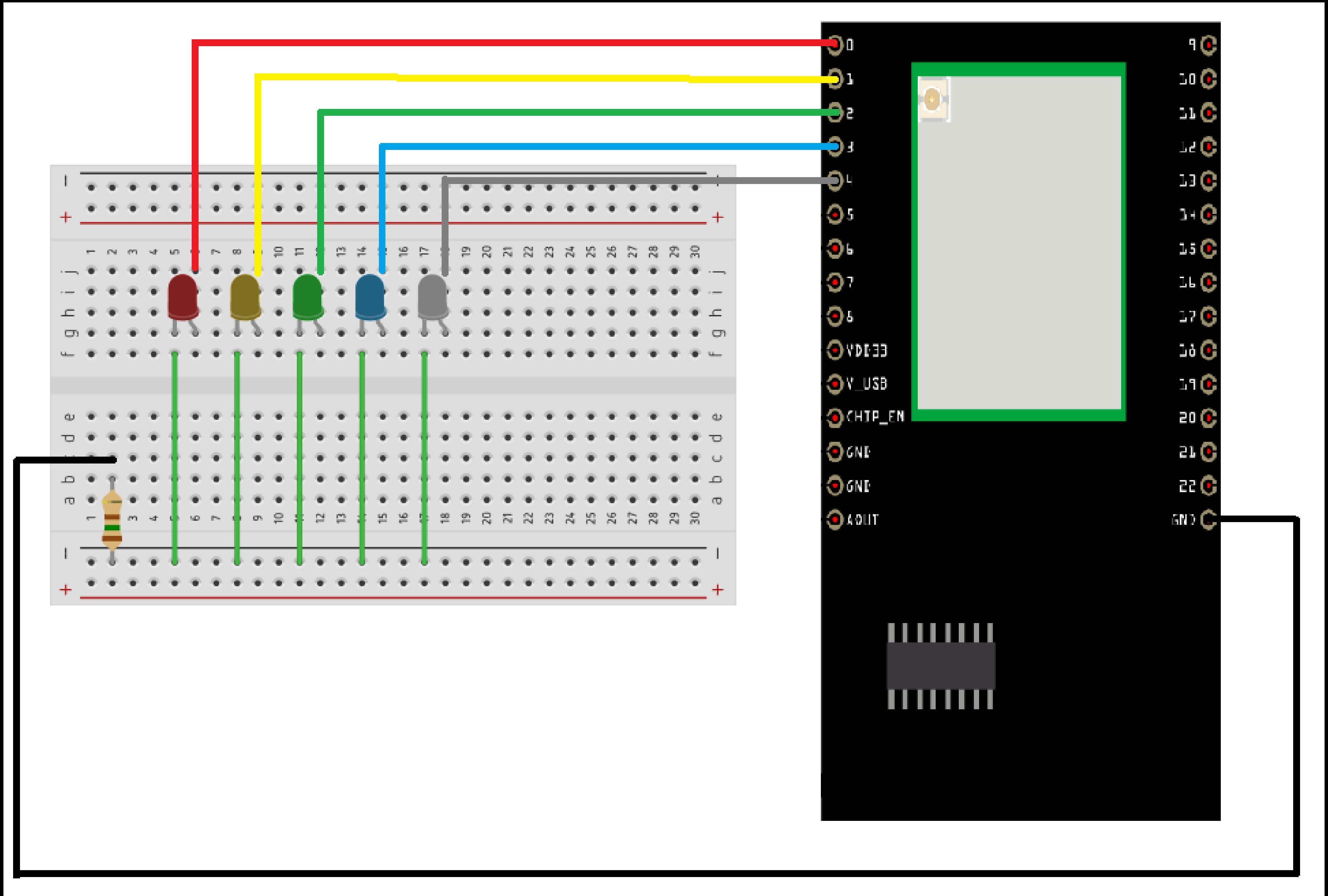
3. Add the following to if(itemList[obj_type].filter) under the function void loop():

(Determine which finger the detected result is)

```
if(obj_type==0) //finger1
          {
            digitalWrite(gesture1, HIGH);
            delay(1000);
            digitalWrite(gesture1, LOW);
            delay(1000);
          }

          else if(obj_type==1) //finger2
          {
            digitalWrite(gesture2, HIGH);
            delay(1000);
            digitalWrite(gesture2, LOW);
            delay(1000);
          }
```

```
else if(obj_type==2)//finger3
          {
            digitalWrite(gesture3, HIGH);
            delay(1000);
            digitalWrite(gesture3, LOW);
            delay(1000);
          }
          else if(obj_type==3) //finger4
          {
            digitalWrite(gesture4, HIGH);
            delay(1000);
            digitalWrite(gesture4, LOW);
            delay(1000);
          }
          else if(obj_type==4) //finger5
          {
            digitalWrite(gesture5, HIGH);
            delay(1000);
            digitalWrite(gesture5, LOW);
            delay(1000);
          }
```

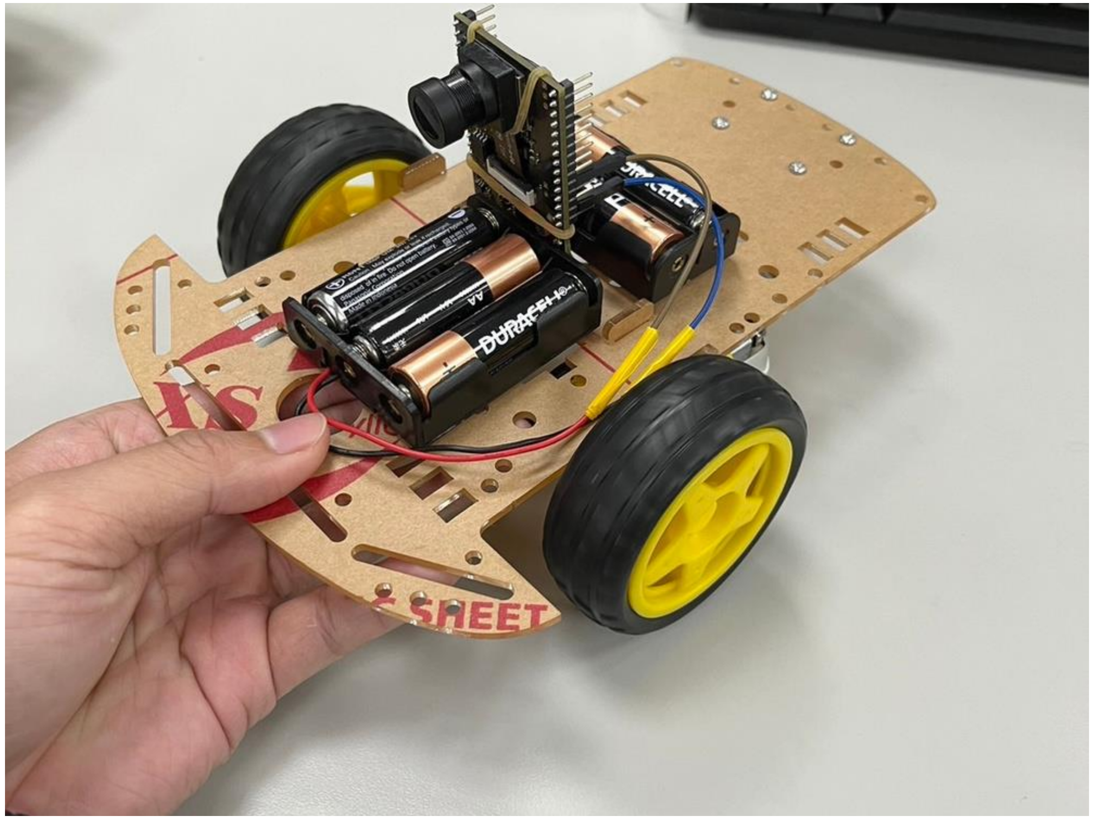
Circuit Diagram



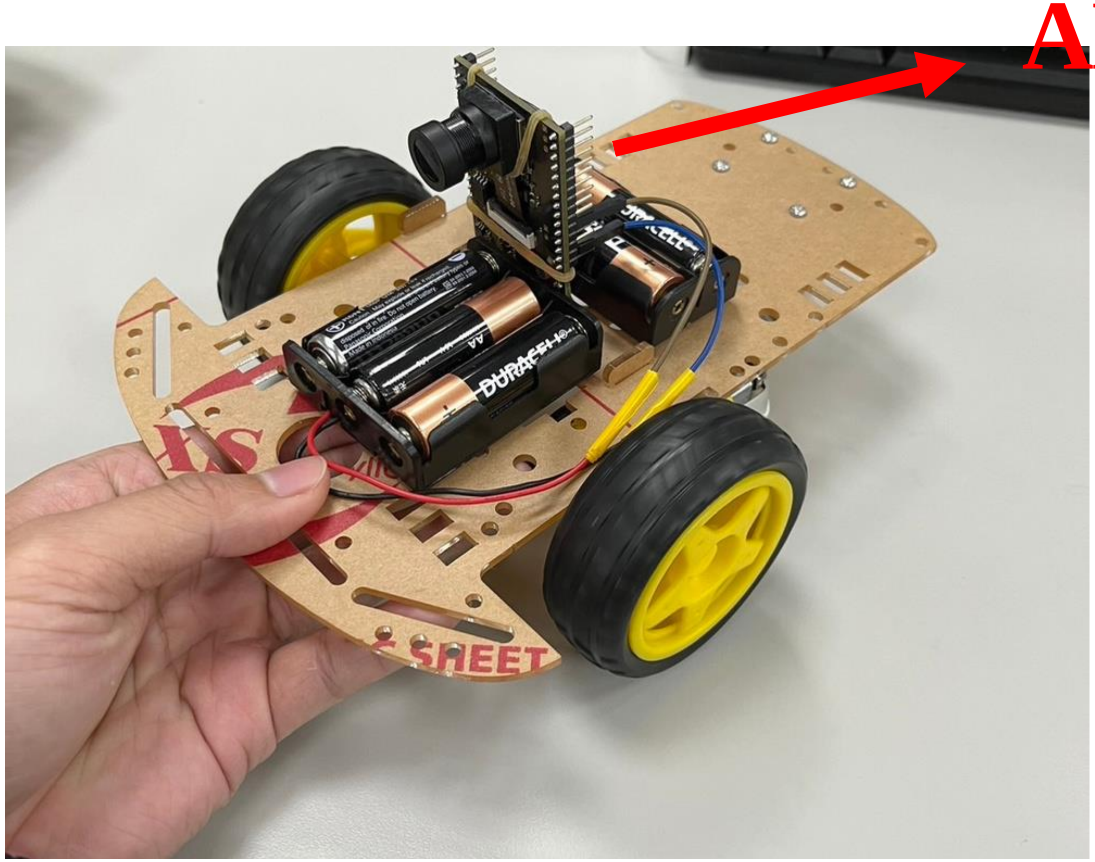
3.2 YOLOv7 Gesture Detection

(Gesture recognition Kart)

3.2 YOLOv7 Gesture Detection

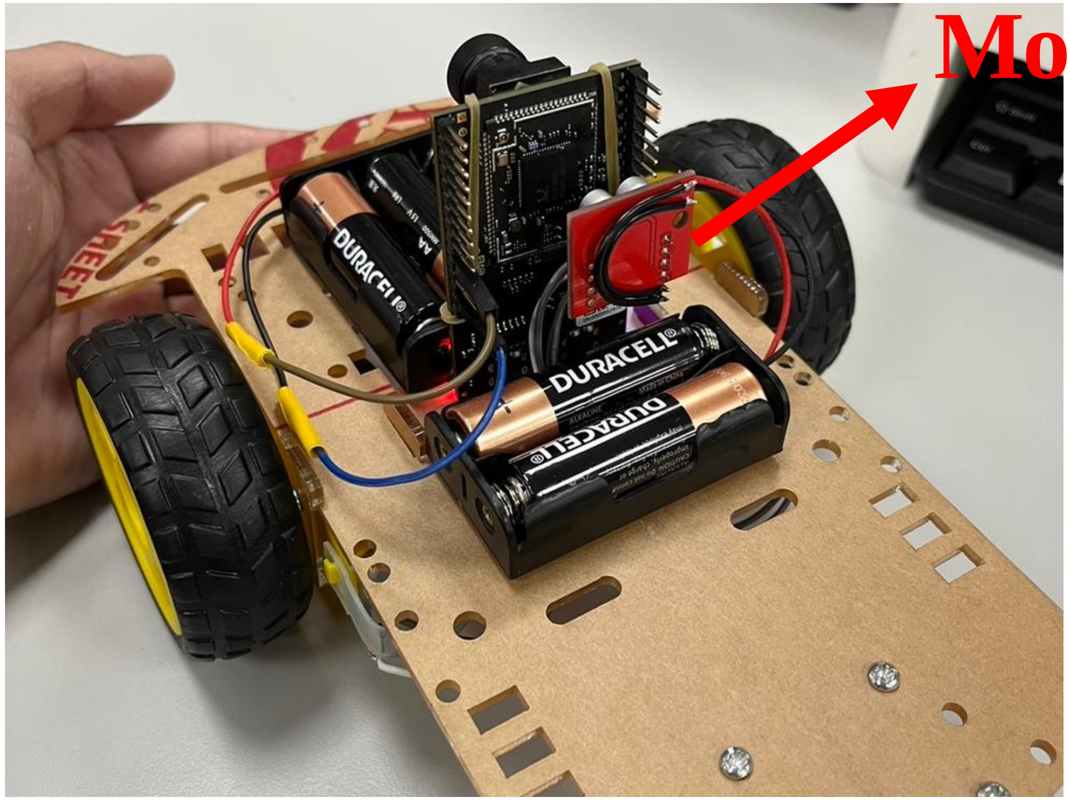


3.2 YOLOv7 Gesture Detection



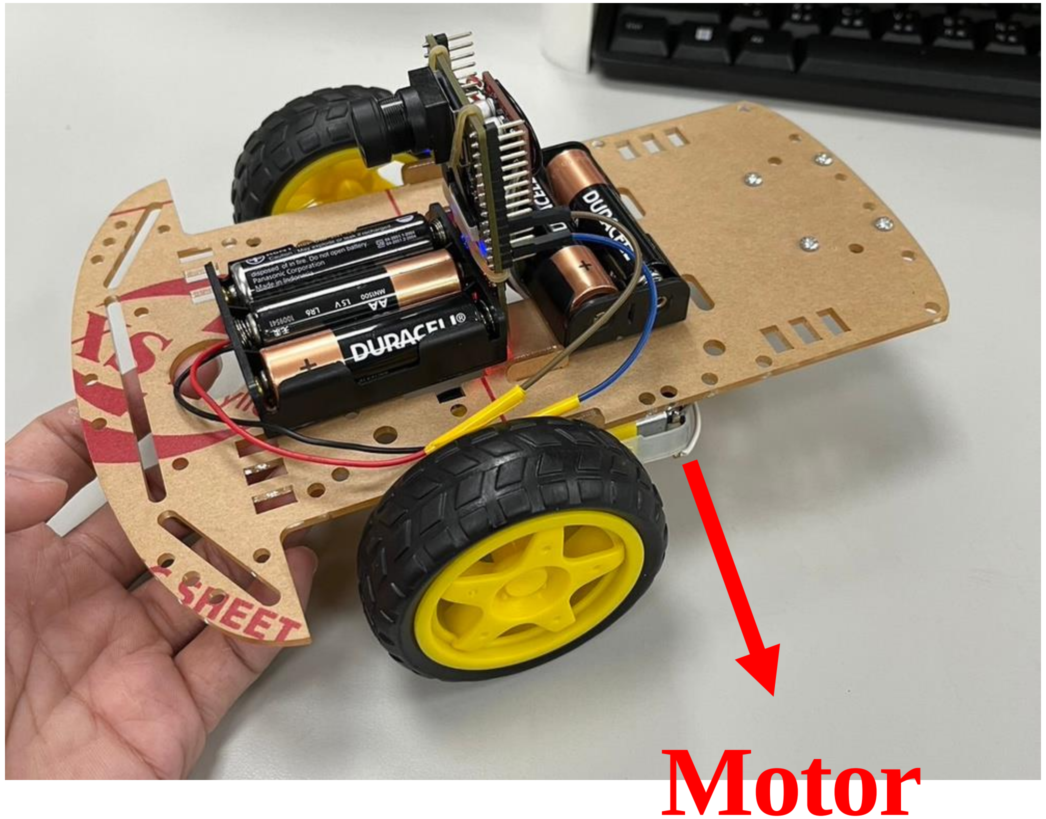
AMB82-MINI

3.2 YOLOv7 Gesture Detection



Motor control board

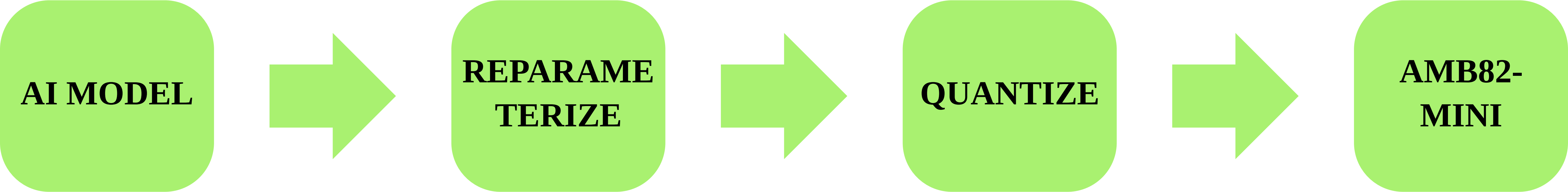
3.2 YOLOv7 Gesture Detection



Motor

Introduction to AI model training

3.2 YOLOv7 Gesture Detection



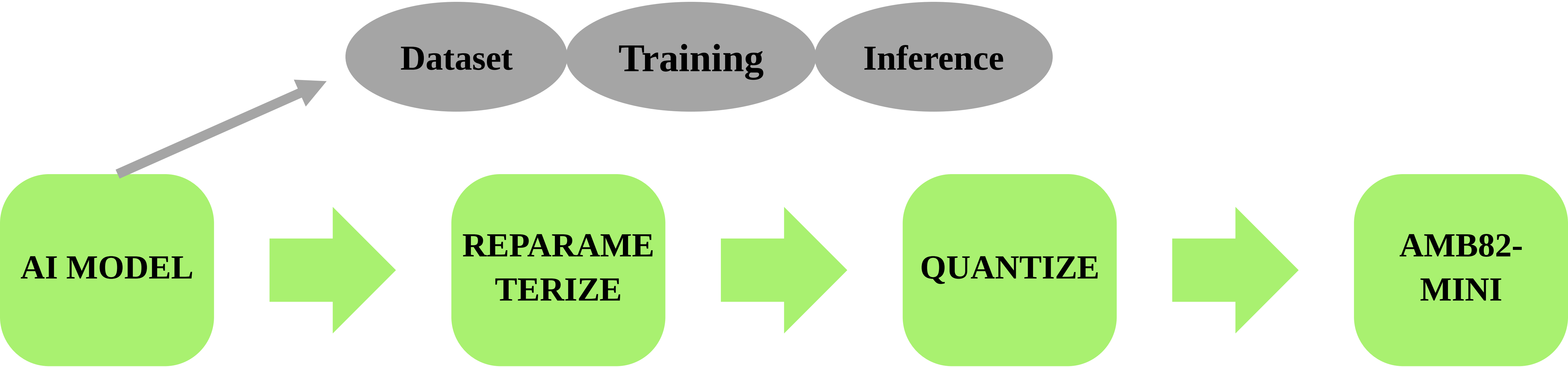
AI MODEL

REPARAMETERIZE

QUANTIZE

AMB82-MINI

3.2 YOLOv7 Gesture Detection

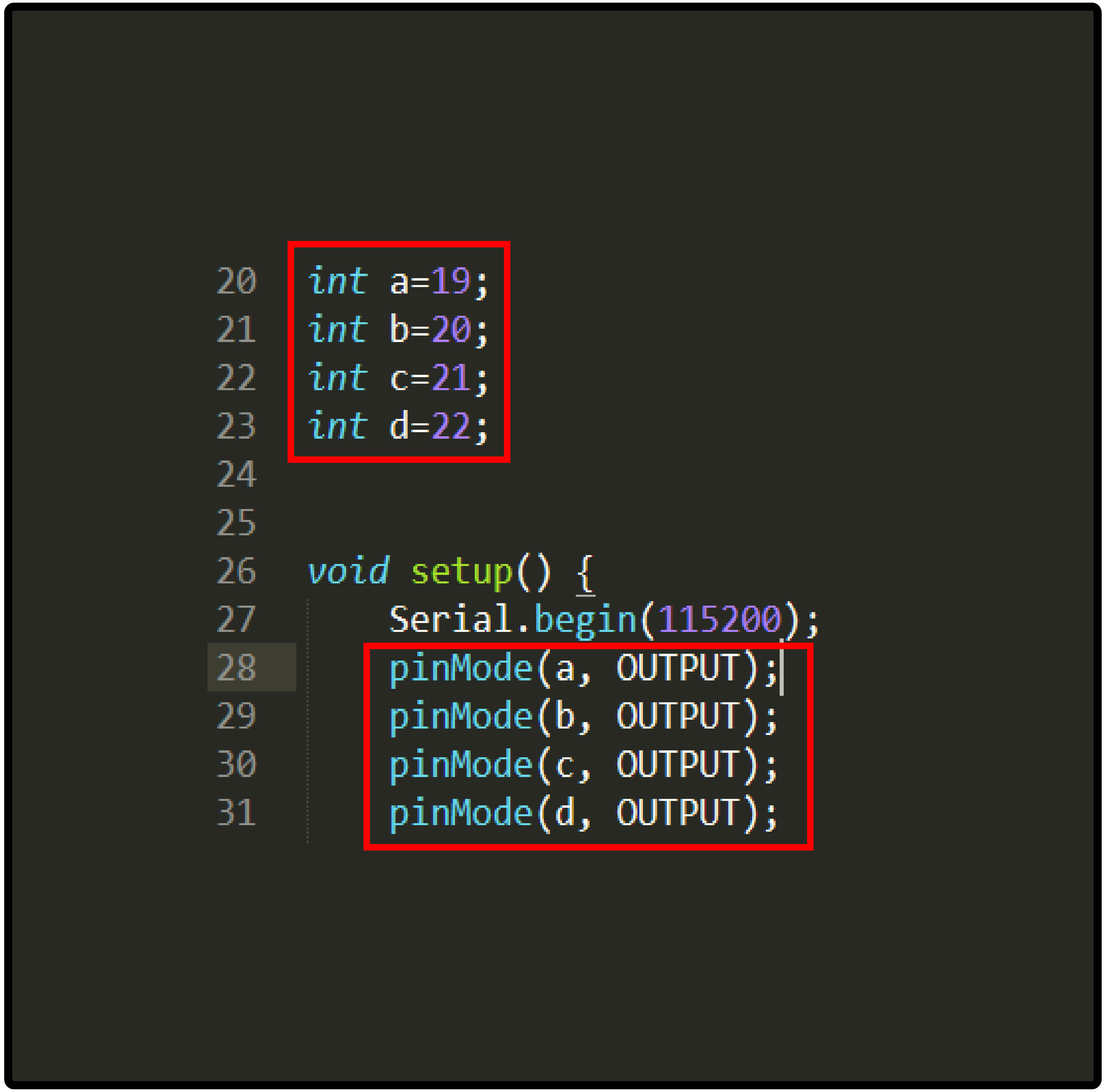


3.2 YOLOv7 Gesture Detection

Programming

01 Setting GPIO pin and the value of GPIO.

02 Assign the output value to the GPIO pin



```
int a=19;
int b=20;
int c=21;
int d=22;


void setup() {
    Serial.begin(115200);
    pinMode(a, OUTPUT);
    pinMode(b, OUTPUT);
    pinMode(c, OUTPUT);
    pinMode(d, OUTPUT);
```

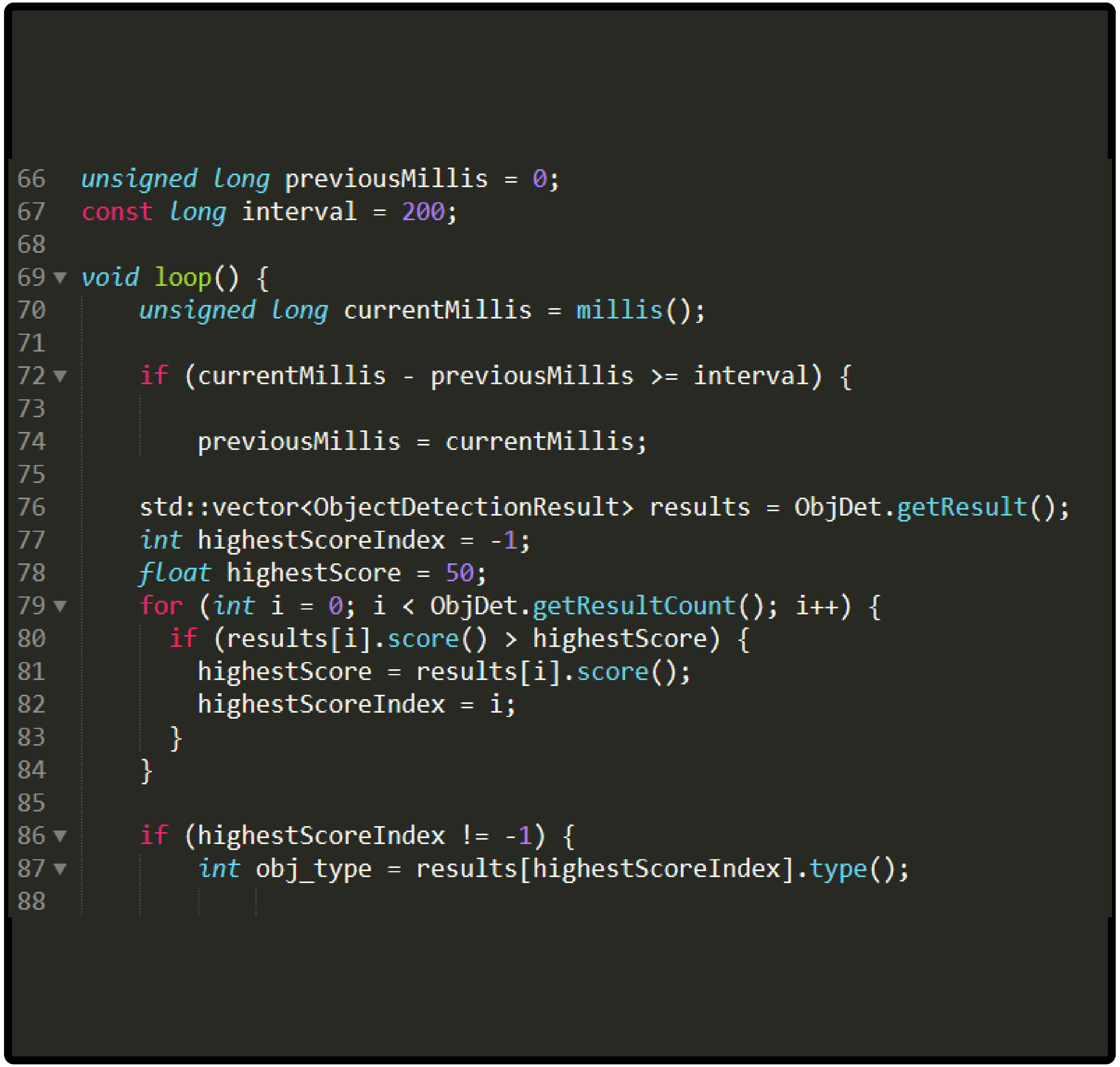
3.2 YOLOv7 Gesture Detection

Programming

01 Setting currentMillis to alleviate latency issues.

02 Classify gestures with a detection result confidence value greater than 50.

03 The result is determined by the category with the highest confidence score.

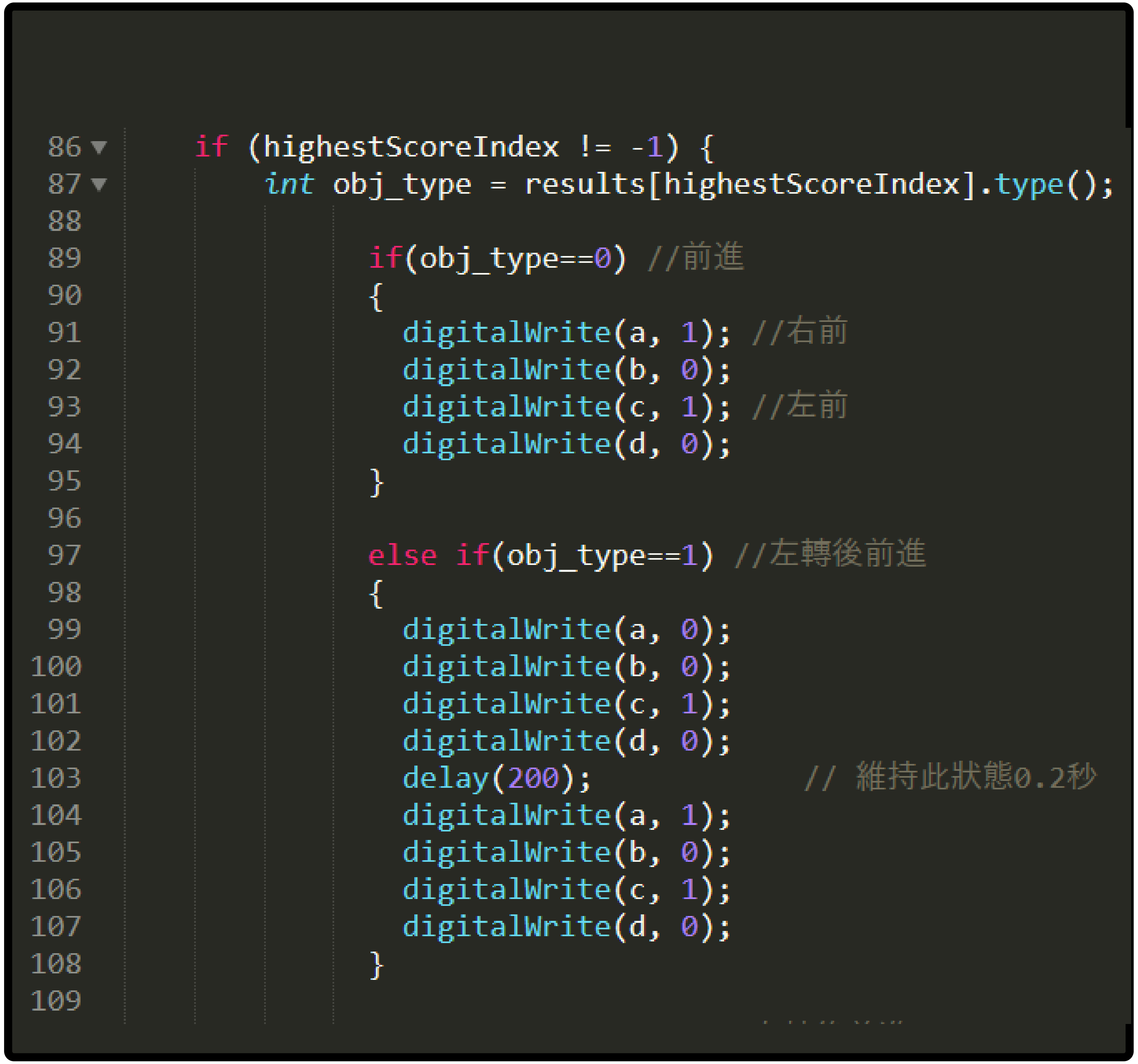


```
unsigned long previousMillis = 0;
const long interval = 200;

void loop() {
    unsigned long currentMillis = millis();

    if (currentMillis - previousMillis >= interval) {

        previousMillis = currentMillis;

    std::vector<ObjectDetectionResult> results = ObjDet.getResult();
    int highestScoreIndex = -1;
    float highestScore = 50;
    for (int i = 0; i < ObjDet.getResultCount(); i++) {
      if (results[i].score() > highestScore) {
        highestScore = results[i].score();
        highestScoreIndex = i;
      }
    }

    if (highestScoreIndex != -1) {
        int obj_type = results[highestScoreIndex].type();

```

3.2 YOLOv7 Gesture Detection

Programming

Match predicted categories to car actions.

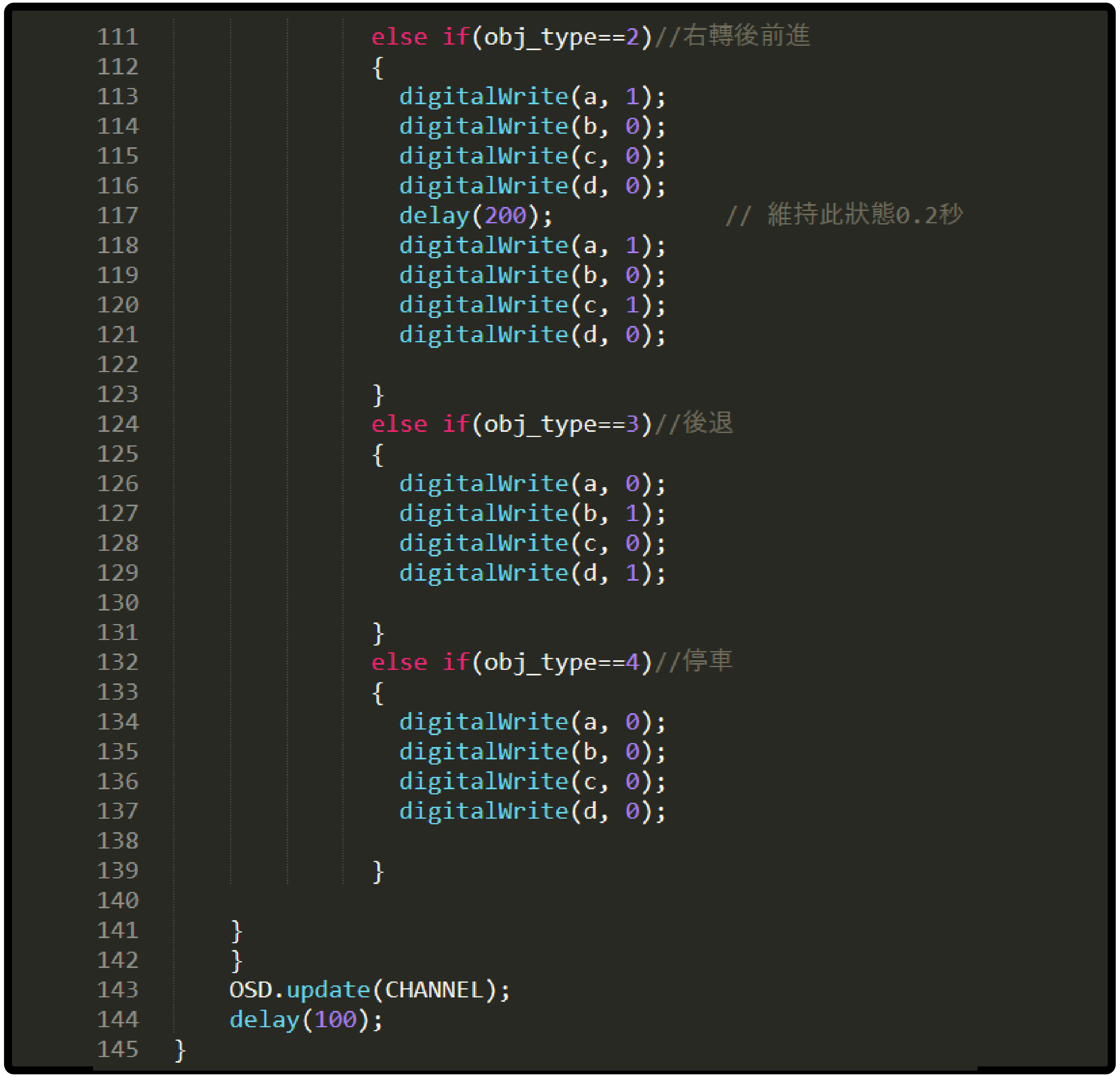


```
if (highestScoreIndex != -1) {
  int obj_type = results[highestScoreIndex].type();

     if(obj_type==0) //前進
     {
      digitalWrite(a, 1); //右前
      digitalWrite(b, 0);
      digitalWrite(c, 1); //左前
      digitalWrite(d, 0);
     }

     else if(obj_type==1) //左轉後前進
     {
      digitalWrite(a, 0);
      digitalWrite(b, 0);
      digitalWrite(c, 1);
      digitalWrite(d, 0);
      delay(200);      // 維持此狀態0.2秒
      digitalWrite(a, 1);
      digitalWrite(b, 0);
      digitalWrite(c, 1);
      digitalWrite(d, 0);
     }

```

3.2 YOLOv7 Gesture Detection

Programming

Match predicted categories to car actions.



```
              else if(obj_type==2)//右轉後前進
              {
                digitalWrite(a, 1);
                digitalWrite(b, 0);
                digitalWrite(c, 0);
                digitalWrite(d, 0);
                delay(200);            // 維持此狀態0.2秒
                digitalWrite(a, 1);
                digitalWrite(b, 0);
                digitalWrite(c, 1);
                digitalWrite(d, 0);

              }
              else if(obj_type==3)//後退
              {
                digitalWrite(a, 0);
                digitalWrite(b, 1);
                digitalWrite(c, 0);
                digitalWrite(d, 1);

              }
              else if(obj_type==4)//停車
              {
                digitalWrite(a, 0);
                digitalWrite(b, 0);
                digitalWrite(c, 0);
                digitalWrite(d, 0);

              }

    }
    }
    OSD.update(CHANNEL);
    delay(100);
}
```

3.2 YOLOv7 Gesture Detection

Code

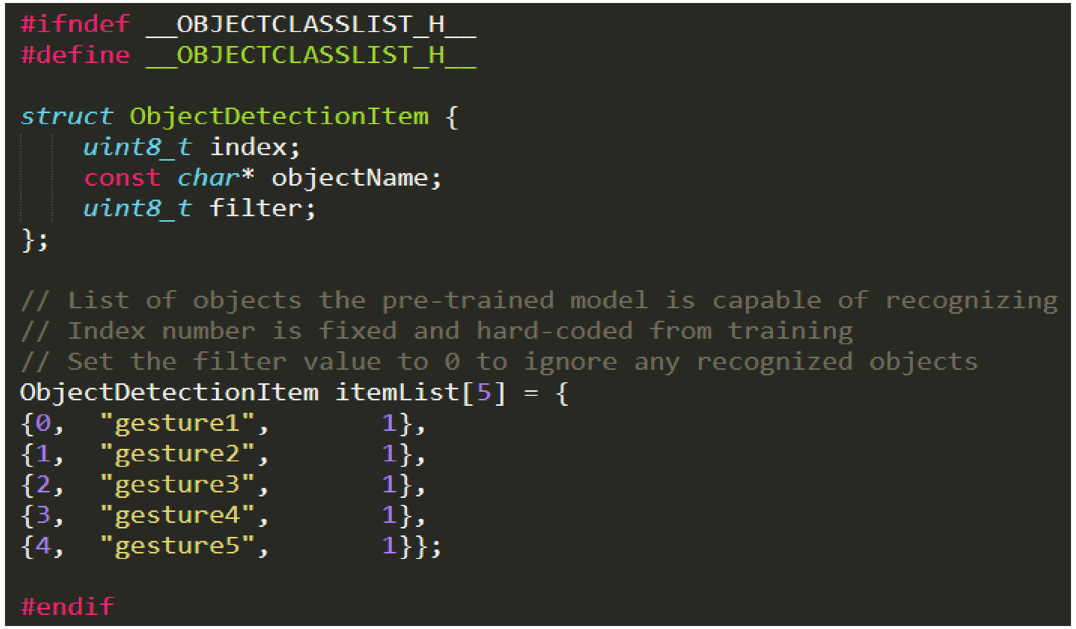
https://drive.google.com/file/d/1AmEI6jfby3BS6mAt2LfuXeCrq86qEFV5/view?usp=drive_link

3.2 YOLOv7 Gesture Detection

Programming

The head file (.h) must map the categories to the model’s output results

Results will look like



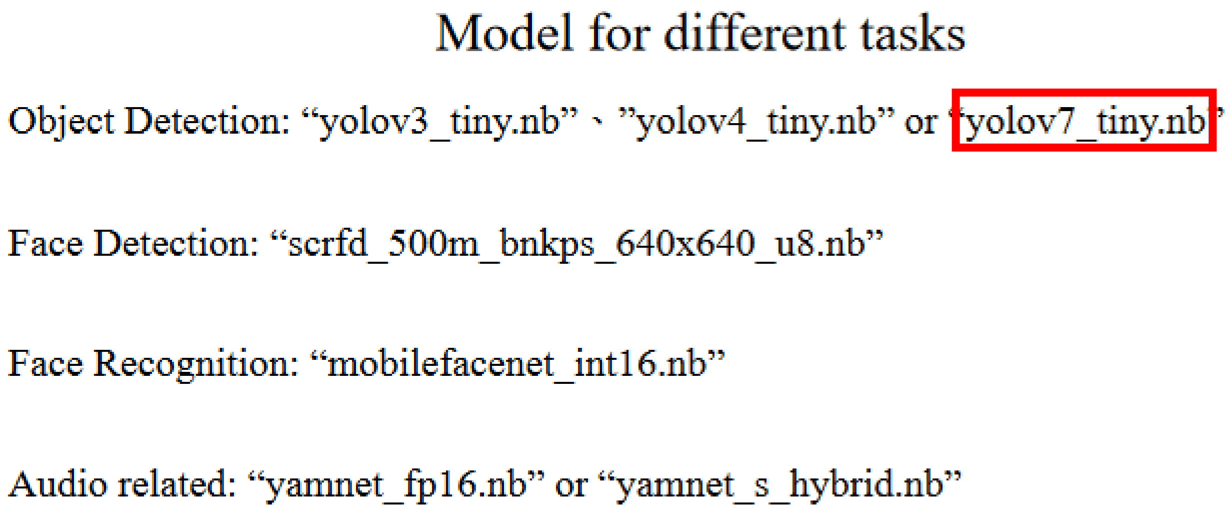
```
#ifndef __OBJECTCLASSLIST_H__
#define __OBJECTCLASSLIST_H__

struct ObjectDetectionItem {
    uint8_t index;
    const char* objectName;
    uint8_t filter;
};

// List of objects the pre-trained model is capable of recognizing
// Index number is fixed and hard-coded from training
// Set the filter value to 0 to ignore any recognized objects
ObjectDetectionItem itemList[5] = {
{0,  "gesture1",       1},
{1,  "gesture2",       1},
{2,  "gesture3",       1},
{3,  "gesture4",       1},
{4,  "gesture5",       1}};

#endif
```

3.2 YOLOv7 Gesture Detection

Model Uploading

First, modify the converted nb file to have the same name as the corresponding model.

Corresponding model are shown at below. In our case, change the name to yolov7_tiny.nb.



Model for different tasks

Object Detection: “yolov3_tiny.nb” 、”yolov4_tiny.nb” or “yolov7_tiny.nb”

Face Detection: “scrfd_500m_bnkps_640x640_u8.nb”

Face Recognition: “mobilefacenet_int16.nb”

Audio related: “yamnet_fp16.nb” or “yamnet_s_hybrid.nb”

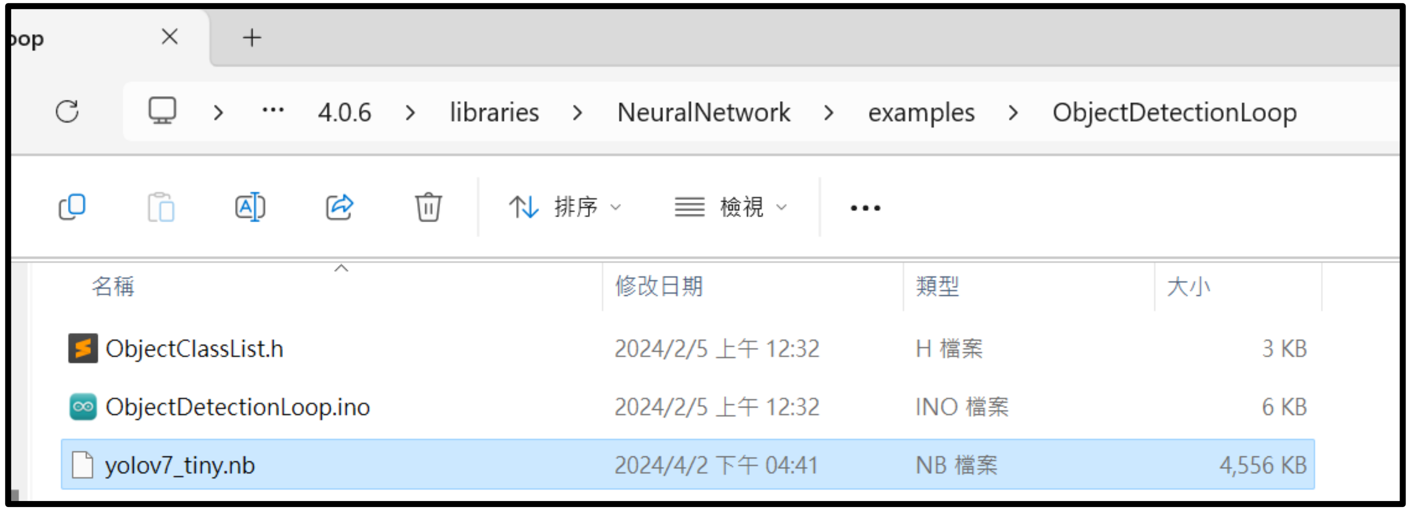
3.2 YOLOv7 Gesture Detection

Model Uploading

Next, find the following path to put the nb file into the folder of the corresponding task

C:\Users\username\AppData\Local\Arduino15\packages\realtek\hardware\AmebaPro2\version\libraries\NeuralNetwork\examples\Corresponding task

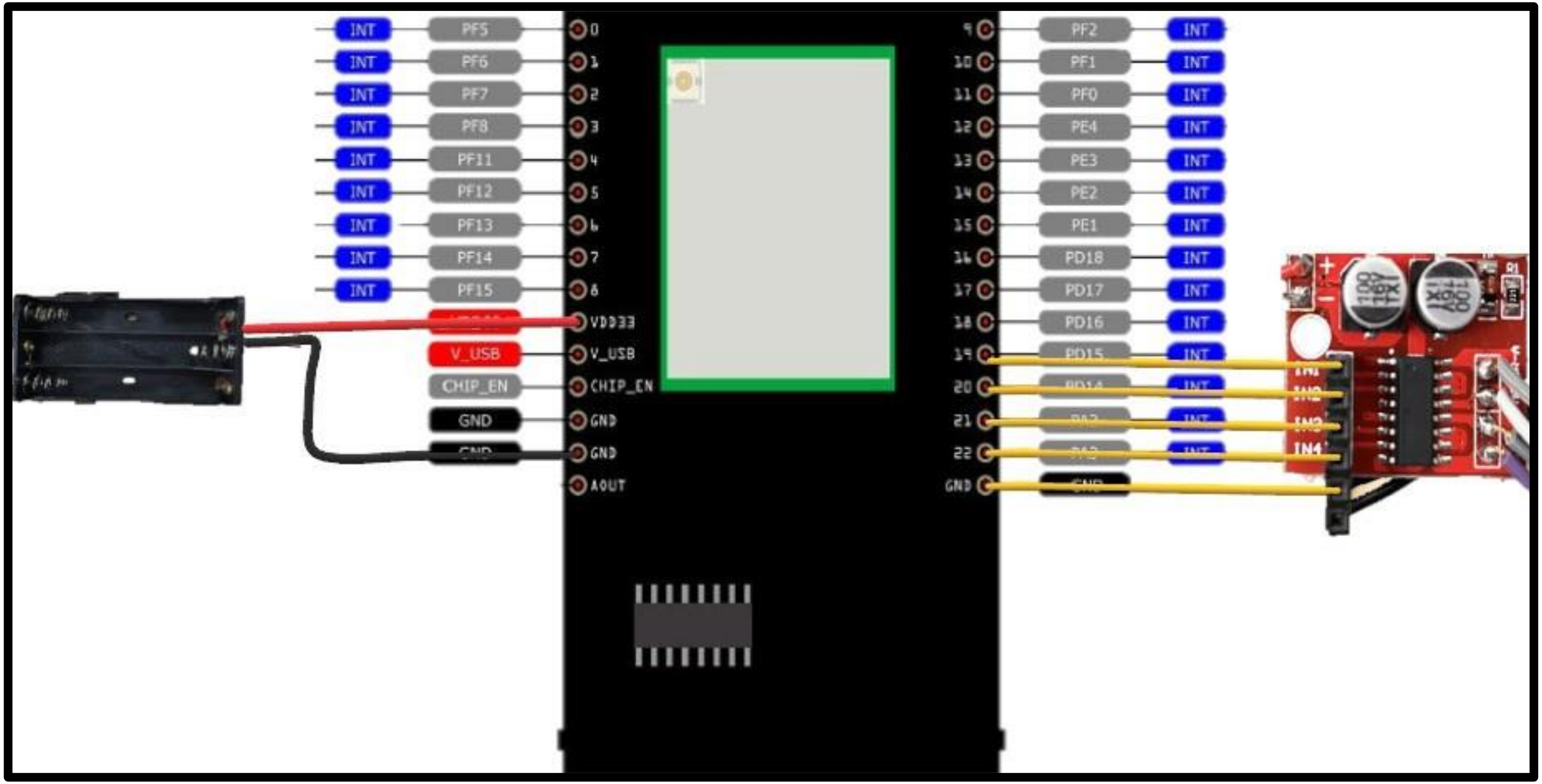
Results will look like



op
4.0.6 > libraries > NeuralNetwork > examples > ObjectDetectionLoop
排序
檢視

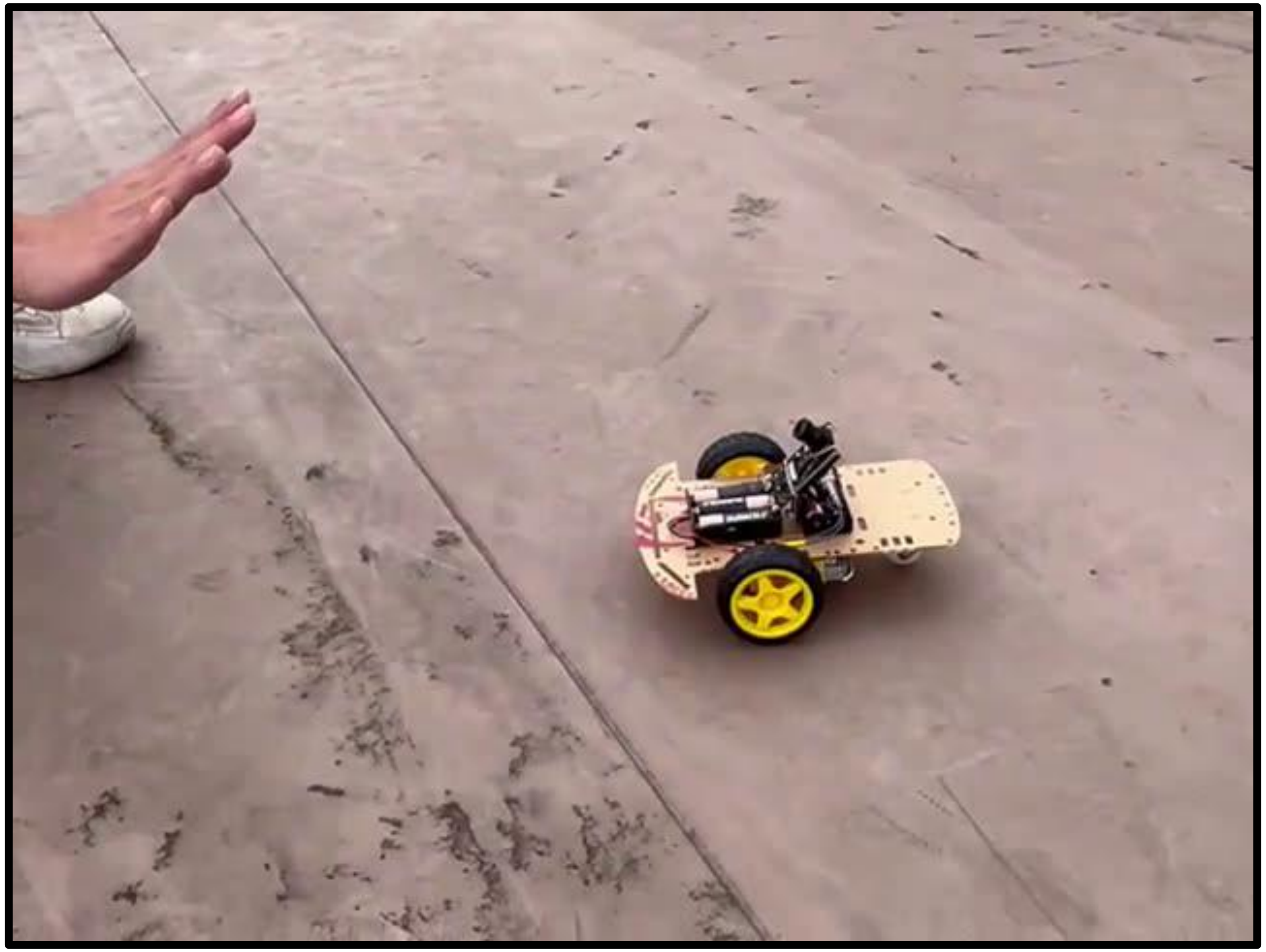
名稱	修改日期	類型	大小
ObjectClassList.h	2024/2/5 上午 12:32	H 檔案	3 KB
ObjectDetectionLoop.ino	2024/2/5 上午 12:32	INO 檔案	6 KB
yolov7_tiny.nb	2024/4/2 下午 04:41	NB 檔案	4,556 KB

3.2 YOLOv7 Gesture Detection



3.2 YOLOv7 Gesture Detection

DEMO Video :



Chapter 4
Application of Object Detection

4.1 Parking cars

4.1 Parking cars

DEMO Video

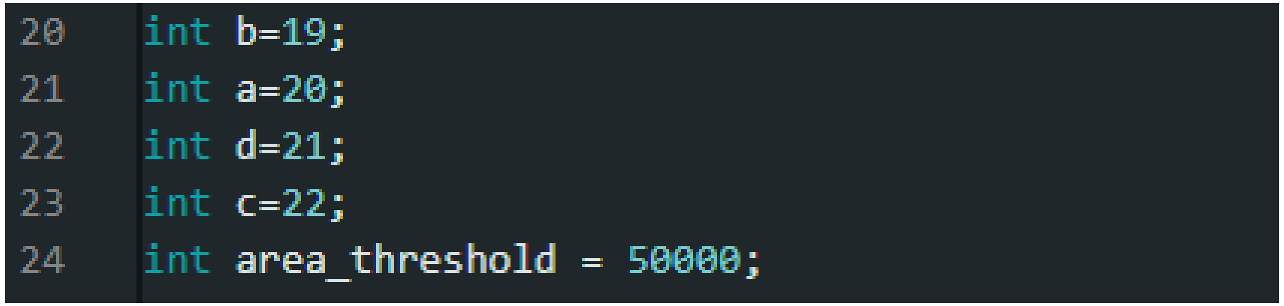
4.1 Parking cars

Code

https://drive.google.com/file/d/1J86lzhAgARXSREn6yvYFWbFO3nOyFQX_/view?usp=sharing

4.1 Parking cars

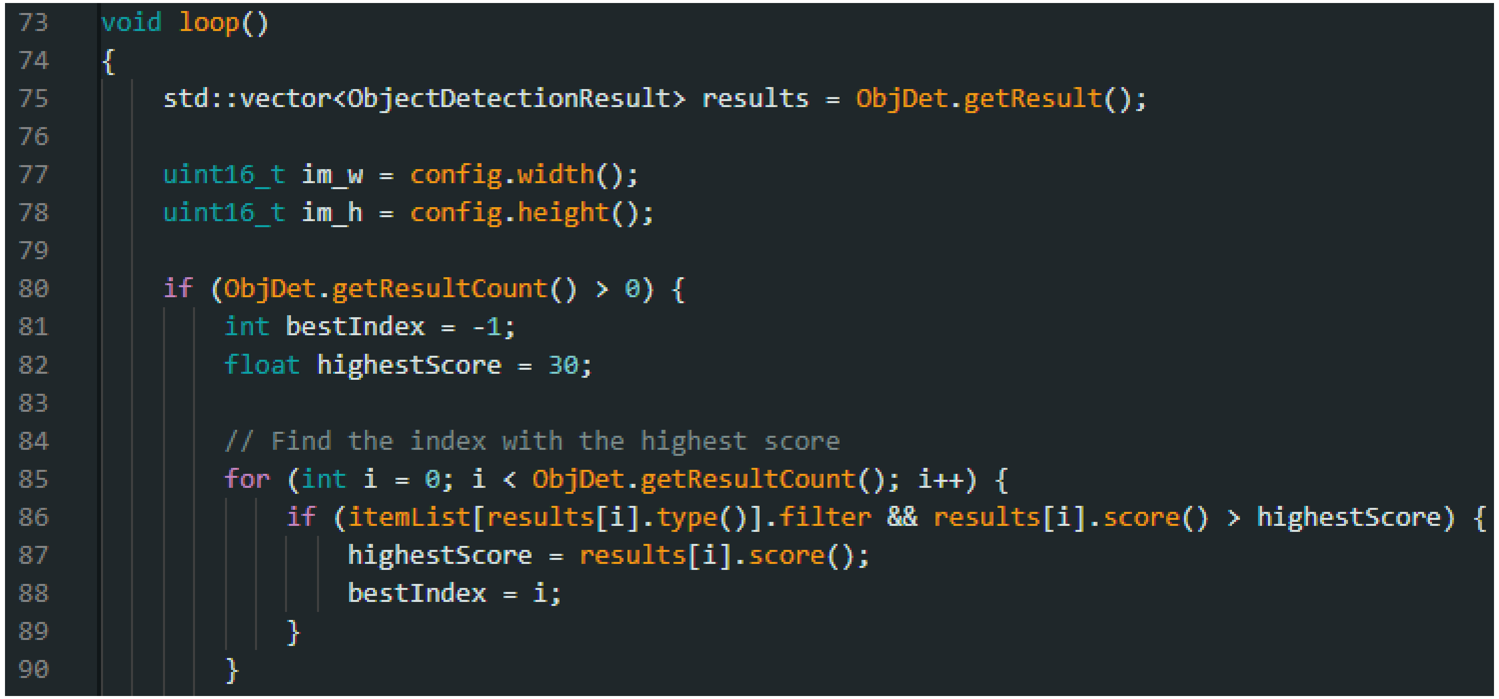
Pin mode & threshold setting



```
int b=19;
int a=20;
int d=21;
int c=22;
int area_threshold = 50000;
```

4.1 Parking cars

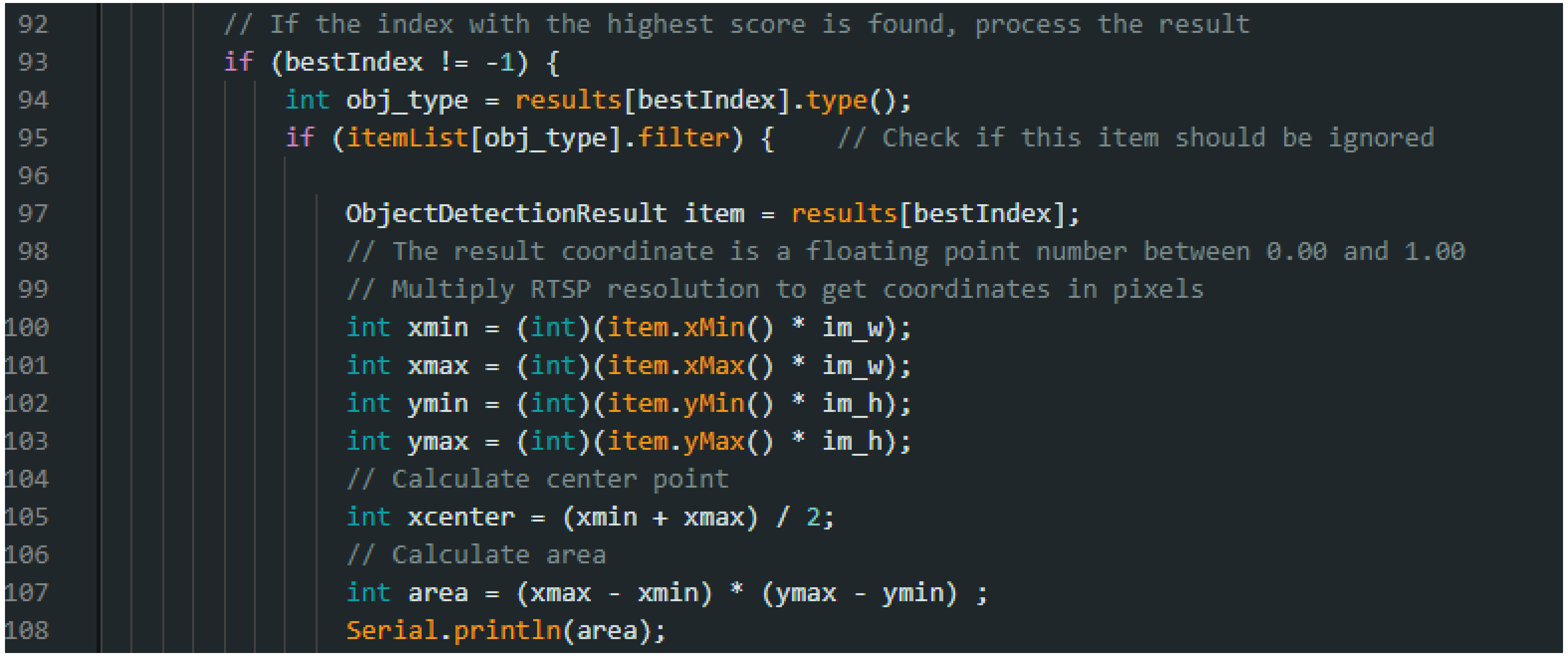
Find the highest confidence score object



```
void loop()
{
    std::vector<ObjectDetectionResult> results = ObjDet.getResult();

    uint16_t im_w = config.width();
    uint16_t im_h = config.height();

    if (ObjDet.getResultCount() > 0) {
        int bestIndex = -1;
        float highestScore = 30;

        // Find the index with the highest score
        for (int i = 0; i < ObjDet.getResultCount(); i++) {
            if (itemList[results[i].type()].filter && results[i].score() > highestScore) {
                highestScore = results[i].score();
                bestIndex = i;
            }
        }
```

4.1 Parking cars

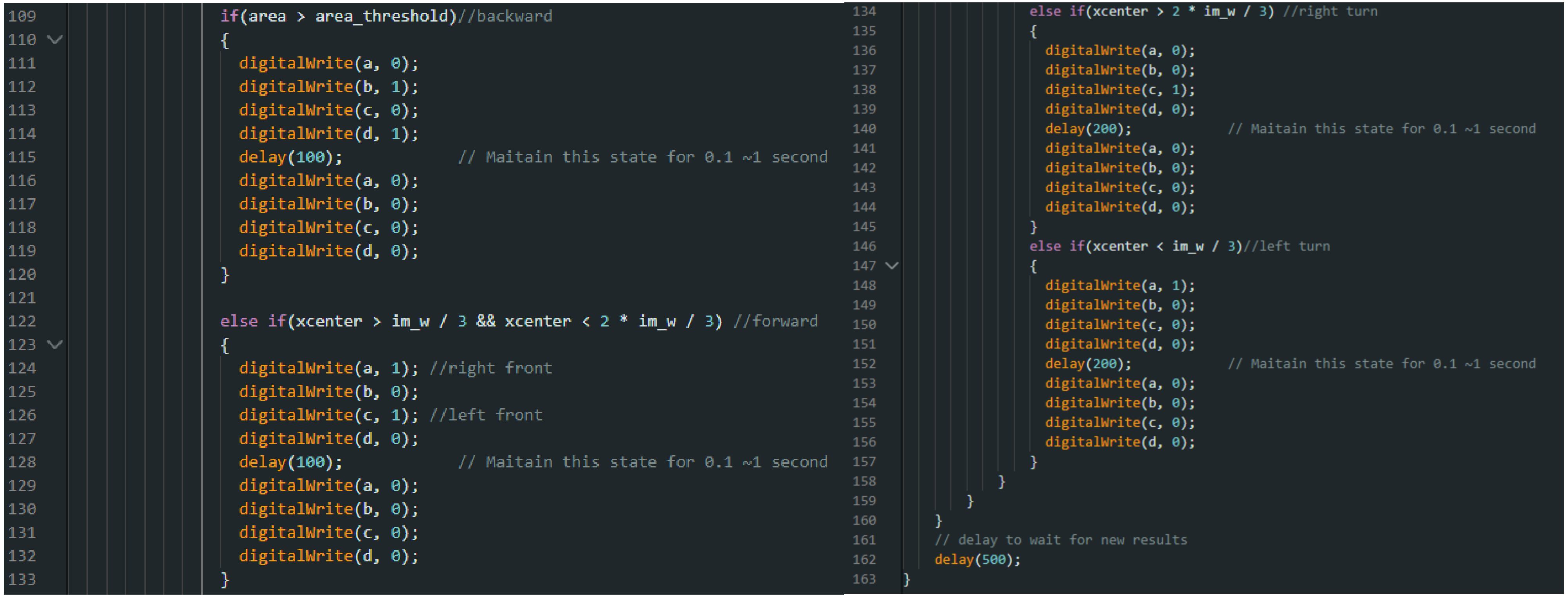
Processing the found object



```
// If the index with the highest score is found, process the result
if (bestIndex != -1) {
    int obj_type = results[bestIndex].type();
    if (itemList[obj_type].filter) {    // Check if this item should be ignored

        ObjectDetectionResult item = results[bestIndex];
        // The result coordinate is a floating point number between 0.00 and 1.00
        // Multiply RTSP resolution to get coordinates in pixels
        int xmin = (int)(item.xMin() * im_w);
        int xmax = (int)(item.xMax() * im_w);
        int ymin = (int)(item.yMin() * im_h);
        int ymax = (int)(item.yMax() * im_h);
        // Calculate center point
        int xcenter = (xmin + xmax) / 2;
        // Calculate area
        int area = (xmax - xmin) * (ymax - ymin) ;
        Serial.println(area);
```

4.1 Parking cars

Motor control

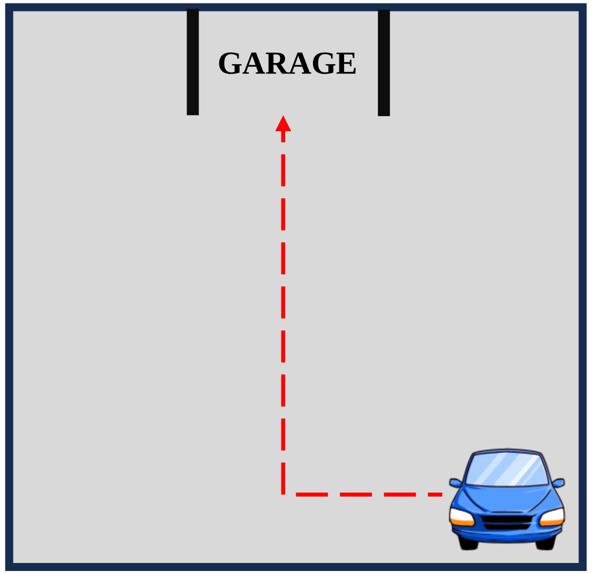


```
        if(area > area_threshold)//backward
        {
          digitalWrite(a, 0);
          digitalWrite(b, 1);
          digitalWrite(c, 0);
          digitalWrite(d, 1);
          delay(100);            // Maitain this state for 0.1 ~1 second
          digitalWrite(a, 0);
          digitalWrite(b, 0);
          digitalWrite(c, 0);
          digitalWrite(d, 0);
        }

        else if(xcenter > im_w / 3 && xcenter < 2 * im_w / 3) //forward
        {
          digitalWrite(a, 1); //right front
          digitalWrite(b, 0);
          digitalWrite(c, 1); //left front
          digitalWrite(d, 0);
          delay(100);            // Maitain this state for 0.1 ~1 second
          digitalWrite(a, 0);
          digitalWrite(b, 0);
          digitalWrite(c, 0);
          digitalWrite(d, 0);
        }
        else if(xcenter > 2 * im_w / 3) //right turn
        {
          digitalWrite(a, 0);
          digitalWrite(b, 0);
          digitalWrite(c, 1);
          digitalWrite(d, 0);
          delay(200);            // Maitain this state for 0.1 ~1 second
          digitalWrite(a, 0);
          digitalWrite(b, 0);
          digitalWrite(c, 0);
          digitalWrite(d, 0);
        }
        else if(xcenter < im_w / 3)//left turn
        {
          digitalWrite(a, 1);
          digitalWrite(b, 0);
          digitalWrite(c, 0);
          digitalWrite(d, 0);
          delay(200);            // Maitain this state for 0.1 ~1 second
          digitalWrite(a, 0);
          digitalWrite(b, 0);
          digitalWrite(c, 0);
          digitalWrite(d, 0);
        }
      }
    }
  }
  // delay to wait for new results
  delay(500);
}
```

4.1 Parking cars

Game description

1. Place your car within the zone
2. Start controlling the car after 3 seconds of countdown
3. The goal is to park the car into the garage
4. The person with the shortest finishing time will be the final winner



4.2 Tango

4.1 Parking cars

DEMO Video

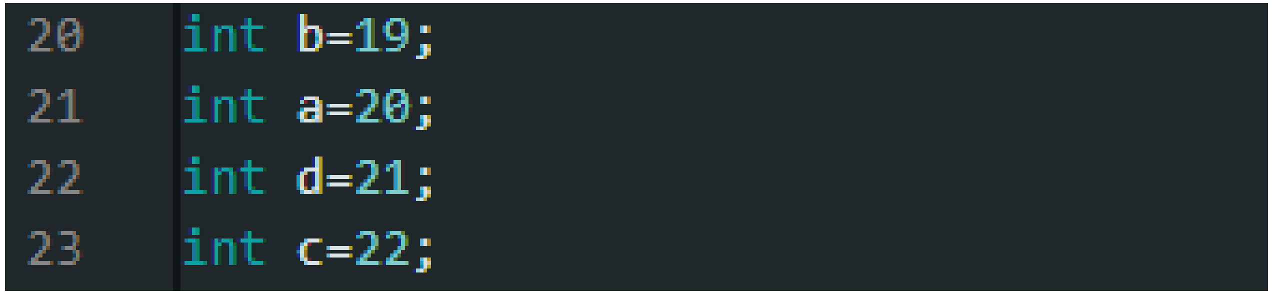
4.2 Tango

Code

https://drive.google.com/file/d/1C9PG8EsXqd25N0eVA_bDtoZ4lByzvtdU/view?usp=sharing

4.2 Tango

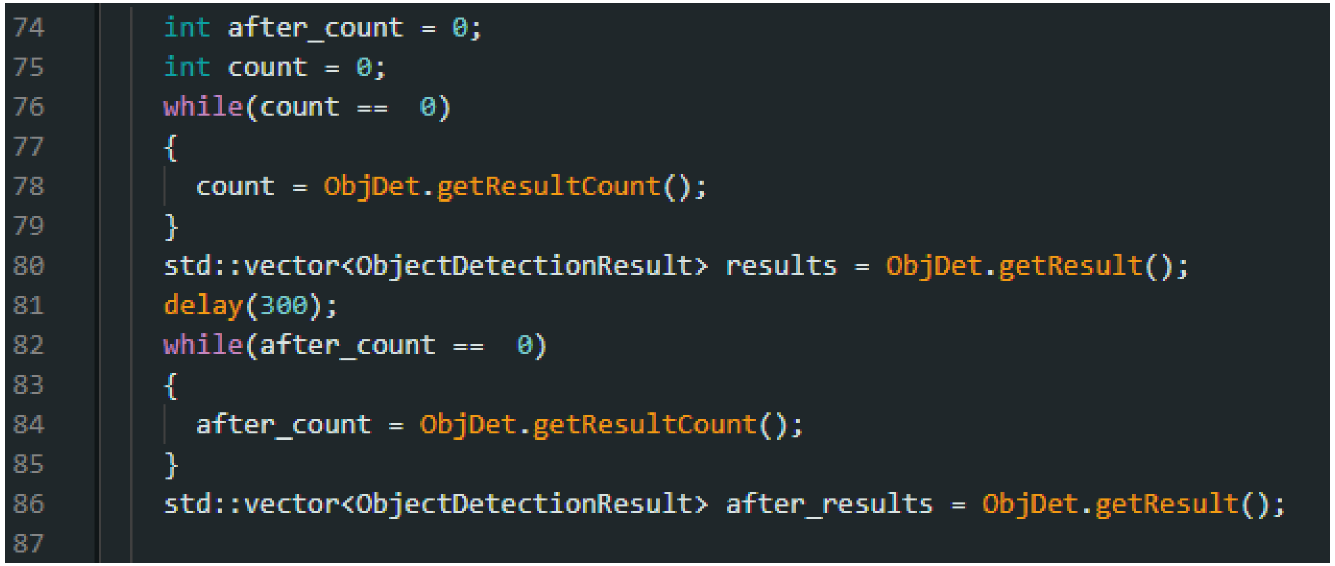
Pin mode setting



```
int b=19;
int a=20;
int d=21;
int c=22;
```

4.2 Tango

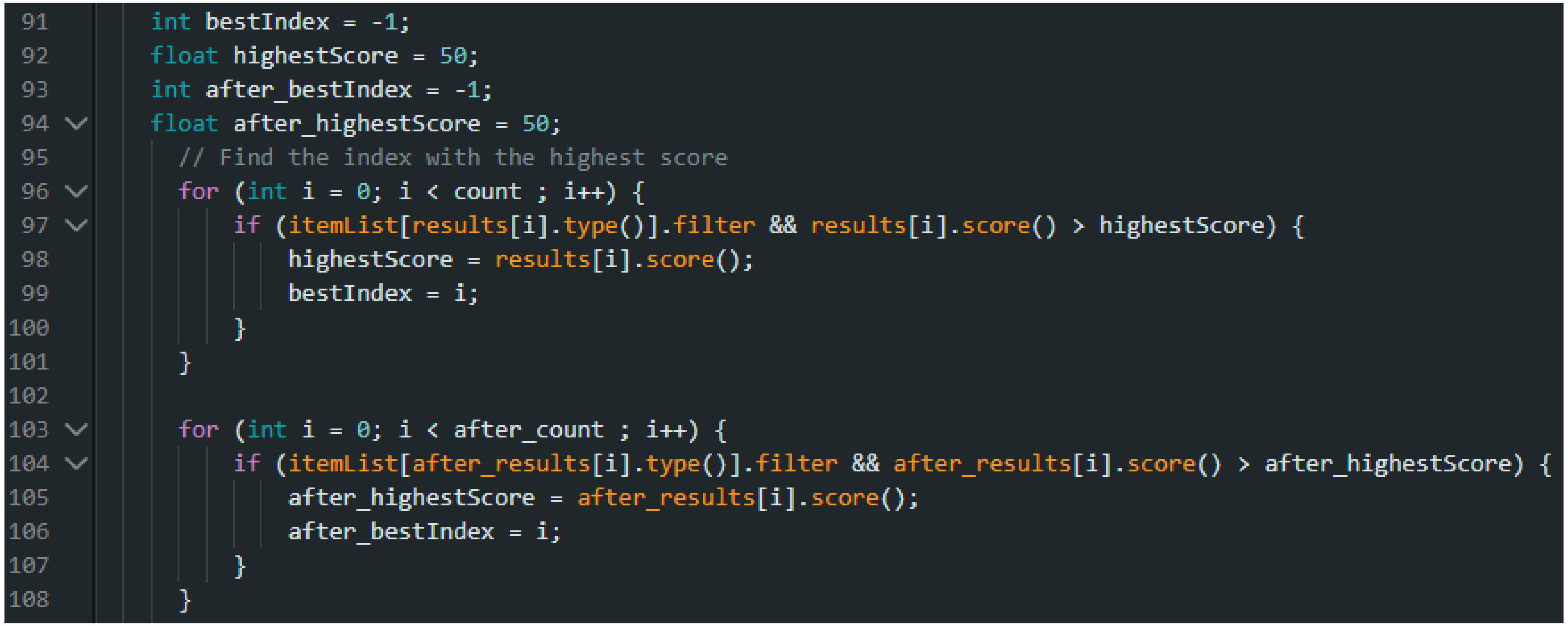
Getting two different frames for comparison



```
int after_count = 0;
int count = 0;
while(count ==  0)
{
    count = ObjDet.getResultCount();
}
std::vector<ObjectDetectionResult> results = ObjDet.getResult();
delay(300);
while(after_count ==  0)
{
    after_count = ObjDet.getResultCount();
}
std::vector<ObjectDetectionResult> after_results = ObjDet.getResult();

```

4.2 Tango

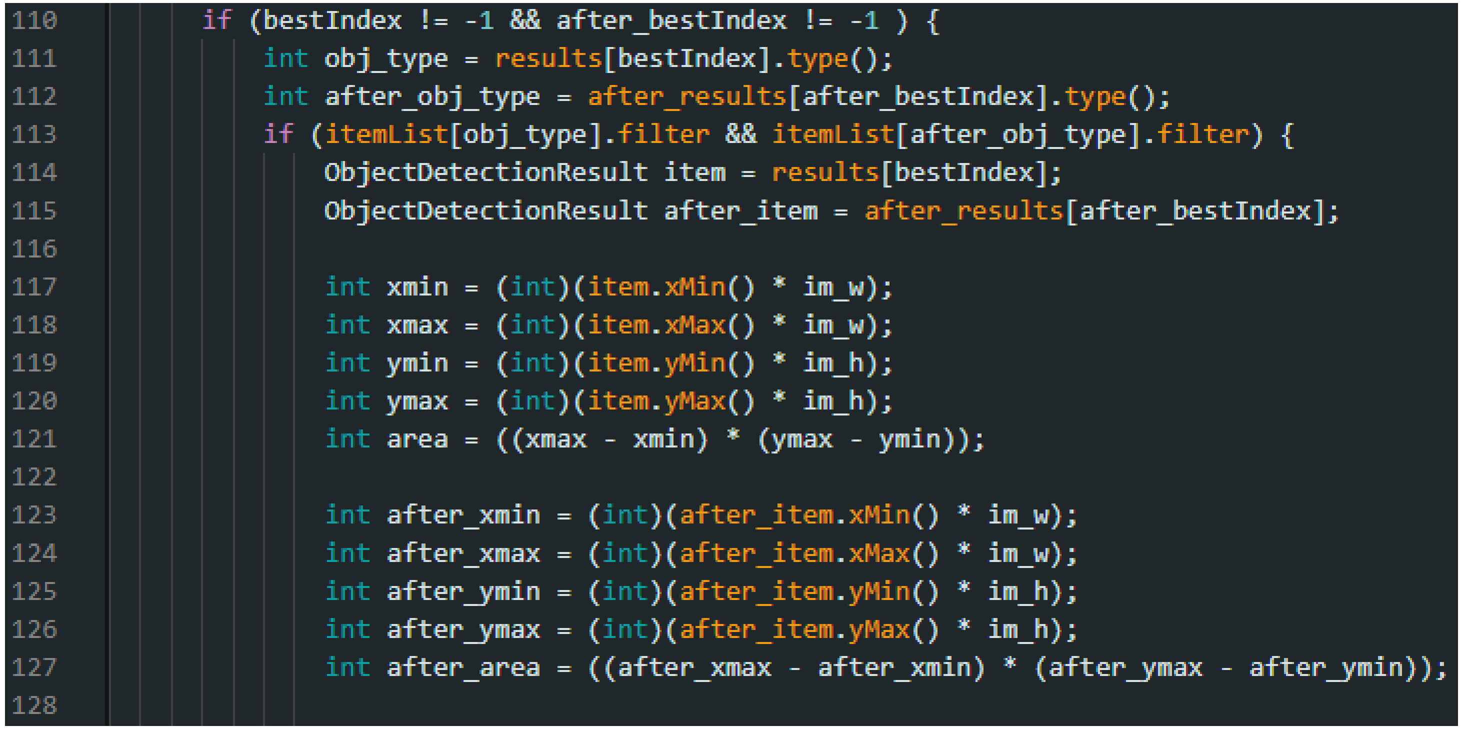
Find the highest confidence score object for each frame



```
int bestIndex = -1;
float highestScore = 50;
int after_bestIndex = -1;
float after_highestScore = 50;
 // Find the index with the highest score
 for (int i = 0; i < count ; i++) {
   if (itemList[results[i].type()].filter && results[i].score() > highestScore) {
     highestScore = results[i].score();
     bestIndex = i;
   }
 }

 for (int i = 0; i < after_count ; i++) {
   if (itemList[after_results[i].type()].filter && after_results[i].score() > after_highestScore) {
     after_highestScore = after_results[i].score();
     after_bestIndex = i;
   }
 }
```

4.2 Tango

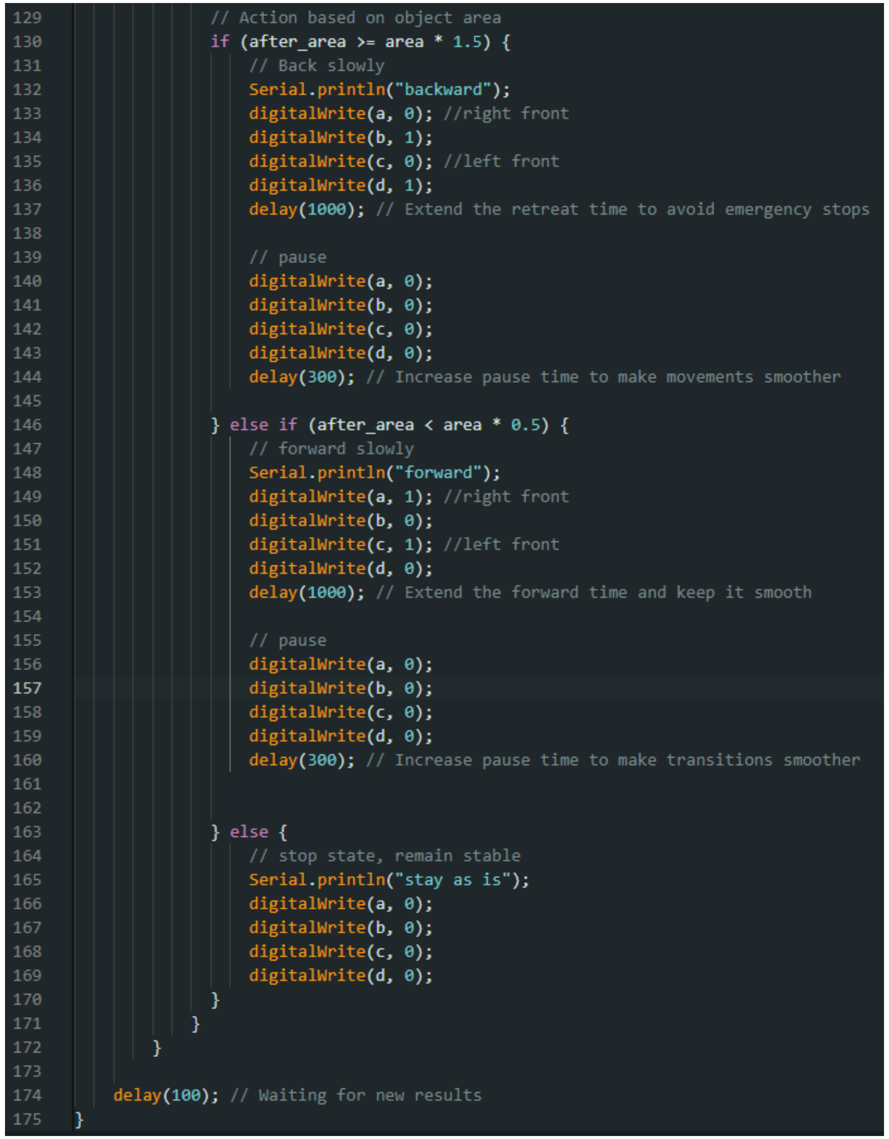
Processing the found object



```
if (bestIndex != -1 && after_bestIndex != -1 ) {
    int obj_type = results[bestIndex].type();
    int after_obj_type = after_results[after_bestIndex].type();
    if (itemList[obj_type].filter && itemList[after_obj_type].filter) {
        ObjectDetectionResult item = results[bestIndex];
        ObjectDetectionResult after_item = after_results[after_bestIndex];

        int xmin = (int)(item.xMin() * im_w);
        int xmax = (int)(item.xMax() * im_w);
        int ymin = (int)(item.yMin() * im_h);
        int ymax = (int)(item.yMax() * im_h);
        int area = ((xmax - xmin) * (ymax - ymin));

        int after_xmin = (int)(after_item.xMin() * im_w);
        int after_xmax = (int)(after_item.xMax() * im_w);
        int after_ymin = (int)(after_item.yMin() * im_h);
        int after_ymax = (int)(after_item.yMax() * im_h);
        int after_area = ((after_xmax - after_xmin) * (after_ymax - after_ymin));

```

4.2 Tango

Motor control

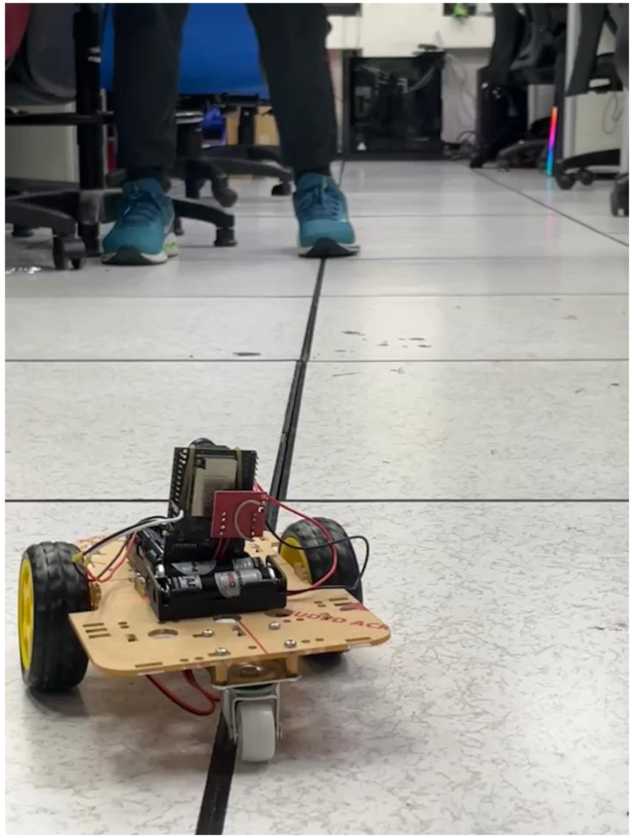


```
              // Action based on object area
              if (after_area >= area * 1.5) {
                  // Back slowly
                  Serial.println("backward");
                  digitalWrite(a, 0); //right front
                  digitalWrite(b, 1);
                  digitalWrite(c, 0); //left front
                  digitalWrite(d, 1);
                  delay(1000); // Extend the retreat time to avoid emergency stops

                  // pause
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
                  delay(300); // Increase pause time to make movements smoother

              } else if (after_area < area * 0.5) {
                  // forward slowly
                  Serial.println("forward");
                  digitalWrite(a, 1); //right front
                  digitalWrite(b, 0);
                  digitalWrite(c, 1); //left front
                  digitalWrite(d, 0);
                  delay(1000); // Extend the forward time and keep it smooth

                  // pause
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
                  delay(300); // Increase pause time to make transitions smoother


              } else {
                  // stop state, remain stable
                  Serial.println("stay as is");
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
              }
            }
        }

    delay(100); // Waiting for new results
}
```

4.3 Obstacle course racing

4.1 Parking cars

DEMO Video



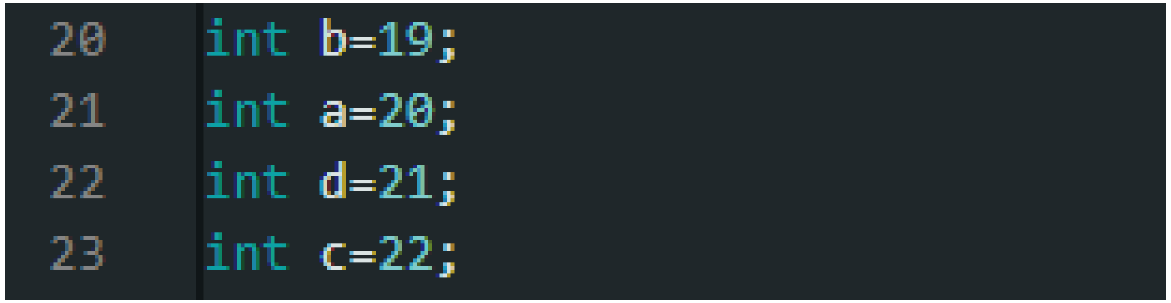
4.3 Obstacle Course racing

Code

https://drive.google.com/file/d/1ZfG3uGIFvKaJXjnabbcEkhDVximjCiDc/view?usp=sharing

4.3 Obstacle Course racing

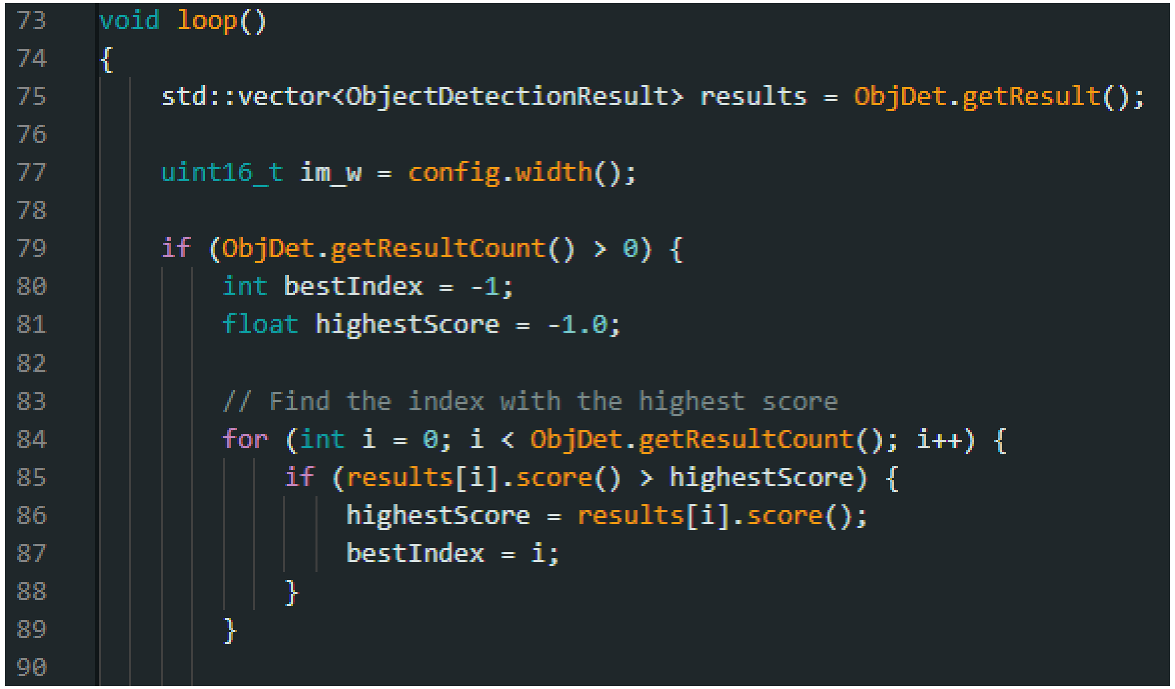
Pin mode setting



```
int b=19;
int a=20;
int d=21;
int c=22;
```

4.3 Obstacle Course racing

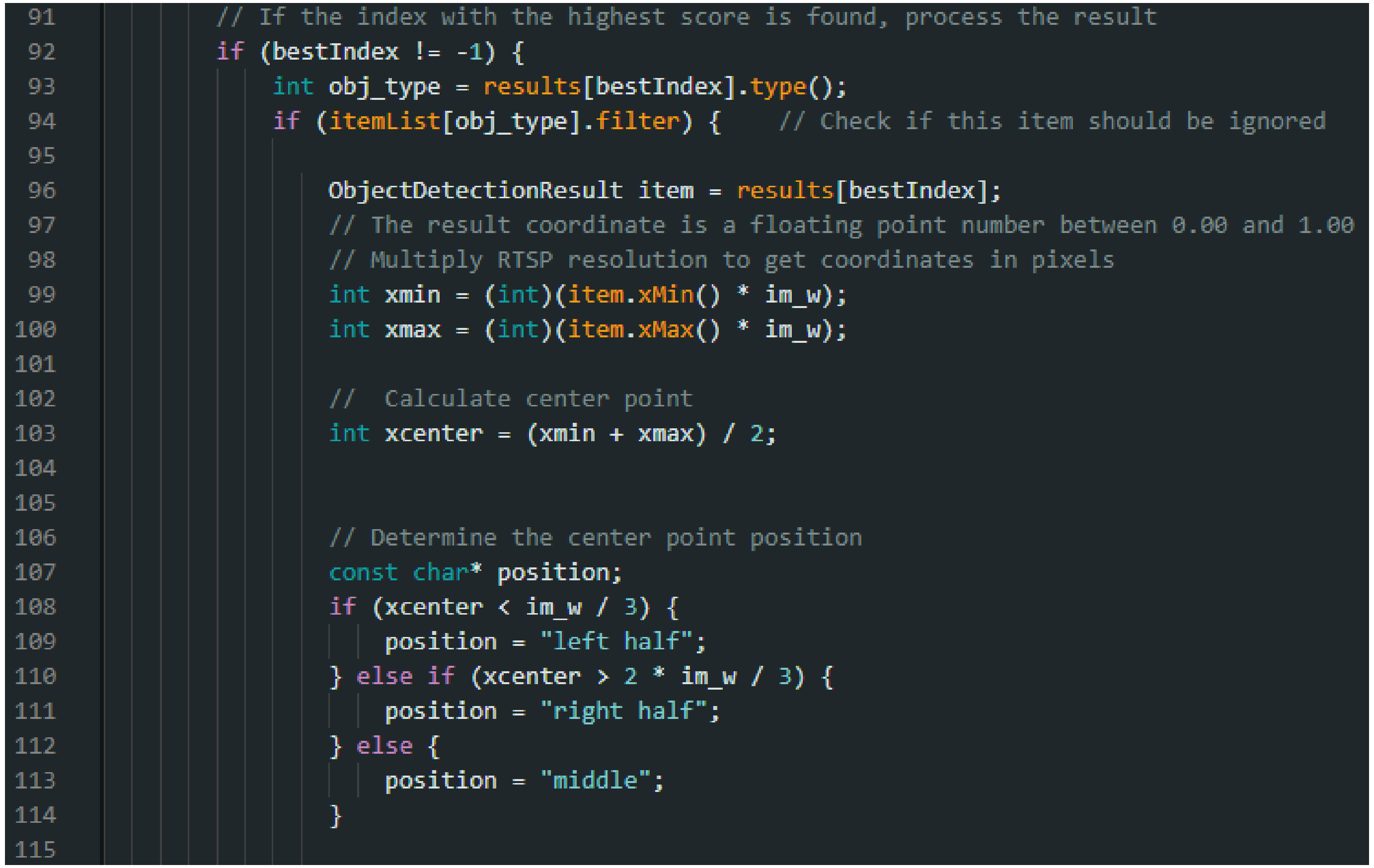
Find the highest confidence score object



```
void loop()
{
    std::vector<ObjectDetectionResult> results = ObjDet.getResult();

    uint16_t im_w = config.width();

    if (ObjDet.getResultCount() > 0) {
        int bestIndex = -1;
        float highestScore = -1.0;

        // Find the index with the highest score
        for (int i = 0; i < ObjDet.getResultCount(); i++) {
            if (results[i].score() > highestScore) {
                highestScore = results[i].score();
                bestIndex = i;
            }
        }

```

4.3 Obstacle Course racing

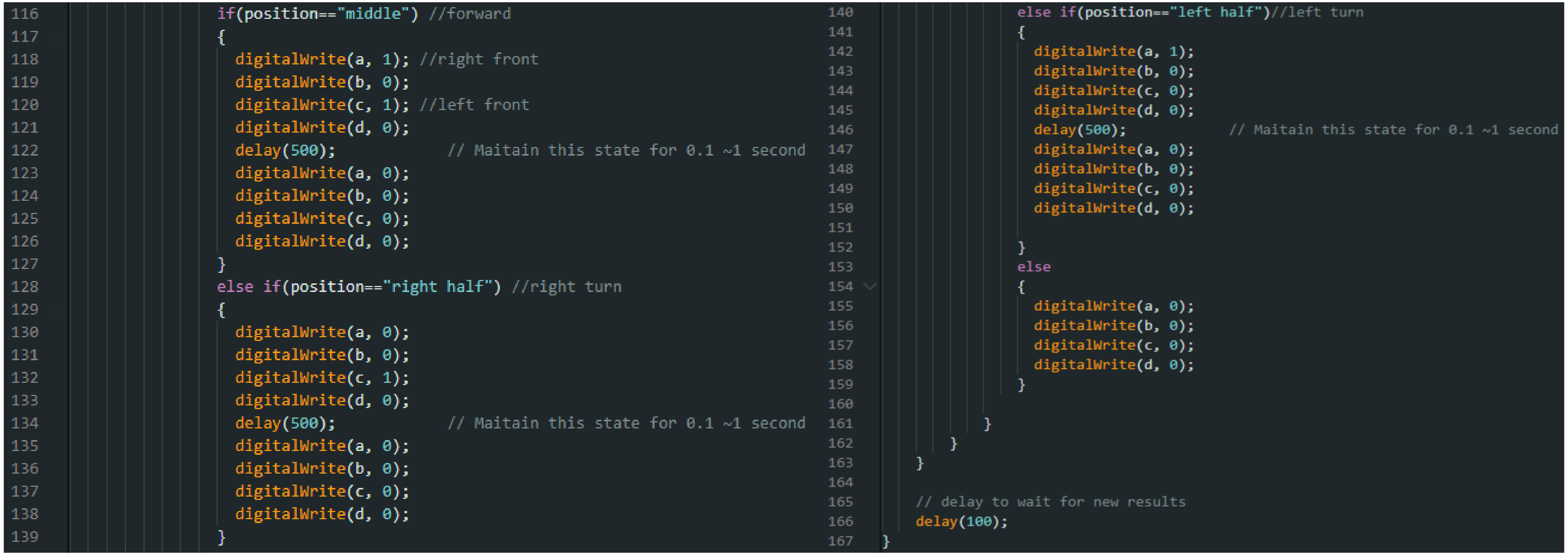
Processing the found object



```
// If the index with the highest score is found, process the result
if (bestIndex != -1) {
    int obj_type = results[bestIndex].type();
    if (itemList[obj_type].filter) {    // Check if this item should be ignored

        ObjectDetectionResult item = results[bestIndex];
        // The result coordinate is a floating point number between 0.00 and 1.00
        // Multiply RTSP resolution to get coordinates in pixels
        int xmin = (int)(item.xMin() * im_w);
        int xmax = (int)(item.xMax() * im_w);

        //  Calculate center point
        int xcenter = (xmin + xmax) / 2;


        // Determine the center point position
        const char* position;
        if (xcenter < im_w / 3) {
            position = "left half";
        } else if (xcenter > 2 * im_w / 3) {
            position = "right half";
        } else {
            position = "middle";
        }

```

4.3 Obstacle Course racing

Motor control

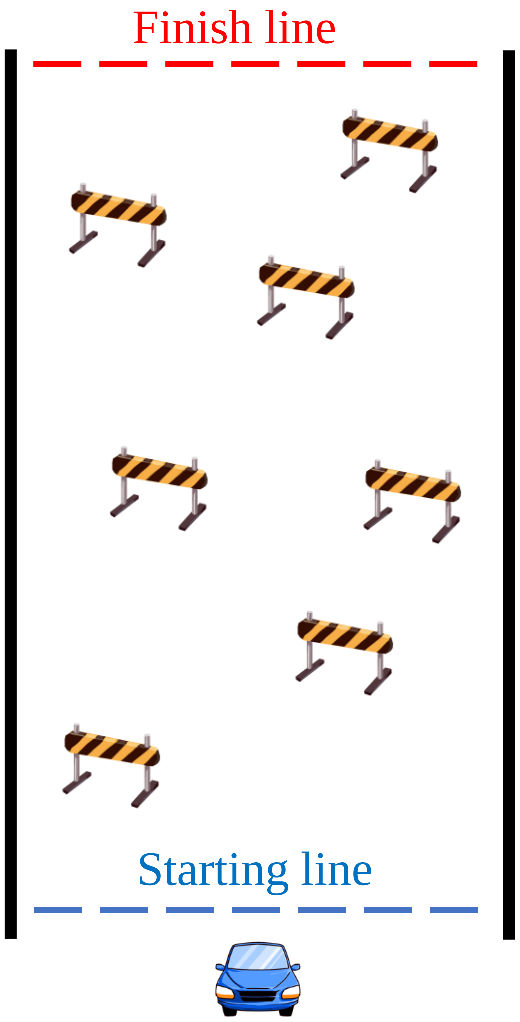


```
                if(position=="middle") //forward
                {
                  digitalWrite(a, 1); //right front
                  digitalWrite(b, 0);
                  digitalWrite(c, 1); //left front
                  digitalWrite(d, 0);
                  delay(500);            // Maitain this state for 0.1 ~1 second
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
                }
                else if(position=="right half") //right turn
                {
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 1);
                  digitalWrite(d, 0);
                  delay(500);            // Maitain this state for 0.1 ~1 second
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
                }
                else if(position=="left half")//left turn
                {
                  digitalWrite(a, 1);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
                  delay(500);            // Maitain this state for 0.1 ~1 second
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);

                }
                else
                {
                  digitalWrite(a, 0);
                  digitalWrite(b, 0);
                  digitalWrite(c, 0);
                  digitalWrite(d, 0);
                }

            }
        }
    }

    // delay to wait for new results
    delay(100);
}
```

4.3 Obstacle Course racing

Game description

1. Place your car at the starting line
2. Start controlling the car after 3 seconds of countdown
3. The goal is to get around the obstacles and reach the finish line
4. The person with the shortest finishing time will be the final winner



4.3 Obstacle Course racing

Paste the code from the following link into the .h file of ObjectDetectionLoop

https://drive.google.com/file/d/1JnzLQff49Q823eubIsf6489V9LYTi7yc/view?usp=sharing