

how android app communicate with the raspberry pi

Initially, we try to use database as a bridge to transmit message between pi and android. However, it fail for some reason:

1. some data like photo, video and audio need to be encoded into other format to be stored in database and need to be decode on android side after transmission.
2. when somebody press the button, the message need to be transmit to android immediately, which is not possible through database.

Therefore, in order to achieve real time message transmission, we use another way to communicate between android and pi. We use a light-weight python web frame work call flask to set up a http server on raspberry pi and provide several api for the android app to get access to data on pi as a http client. The android and pi need to be in the same LAN.

flask-socketio

With the help of flask web framework, the android app can trigger action on raspberry pi, like open the lock or record an audio, by sending a http get request to pi server. However, under some situation, we want the server to initiate connection with android client and send message to it, which is quite difficult because once one communication is over, the connection is cut off.

Therefore, we use a Socket.IO integration for Flask applications call flask-socketio to set up long-life, bi-directional connection between pi and android. SocketIO is an event-driven library for real-time web applications. It enables real-time, bi-directional communication between web clients and servers.

On the android side, we use the java implementation of SocketIO to set up a socket client to receive message. Here we can see that the SocketIO dependency is added to android project and a SocketManager java class is written to handle with socket object.

```
dependencies {  
  
    implementation("androidx.appcompat:appcompat:1.6.1")  
    implementation("com.google.android.material:material:1.11.0")  
    implementation("androidx.constraintlayout:constraintlayout:2.1.4")  
    testImplementation("junit:junit:4.13.2")  
    androidTestImplementation("androidx.test.ext:junit:1.1.5")  
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")  
    implementation("com.squareup.okhttp3:okhttp:4.12.0")  
    implementation("io.socket:socket.io-client:2.0.0")  
}
```

```

1 package com.example.smartdoorbell;
2
3 > import ...
  2 usages  ⚡ coryTu
7 public class SocketManager {
  3 usages
8     private static Socket mSocket;
  1 usage  ⚡ coryTu
9     private SocketManager(String SERVER_URL) {
10         try {
11             mSocket = IO.socket(SERVER_URL);
12         } catch (URISyntaxException e) {
13             e.printStackTrace();
14         }
15     }
  1 usage  ⚡ coryTu
16     public static Socket getInstance(String SERVER_URL) {
17         if (mSocket == null) {
18             new SocketManager(SERVER_URL);
19         }
20         return mSocket;
21     }
22 }
23

```

In the MainActivity, a function is written to handle with the message receiving from server.

```

socket.on( event: "message", args -> {
    String message = (String) args[0];
    switch (message) {
        case "Some one is at the door!":
            System.out.println("Some one is at the door!");
            new Thread(() -> {
                try {
                    int t_sleep = 15000;
                    getCameraPhoto(t_sleep);
                    runOnUiThread(() -> txtStatus.setText(message));
                    Thread.sleep(t_sleep);
                    runOnUiThread(() -> txtStatus.setText(""));
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            }).start();
            break;
        case "The door is opened":
            System.out.println("The door is opened");
            runOnUiThread(() -> {
                txtDoorStatus.setText("OPEN");
            });
            break;
        case "The door is closed":
            System.out.println("The door is closed");
            runOnUiThread(() -> {
                txtDoorStatus.setText("CLOSE");
                imgBtnLock.setImageResource(R.drawable.ic_lock);
            });
            break;
        default:
            System.out.println("incorrect message");
            break;
    }
});

socket.connect();

```

Depends on different message, the app do different operation. For example, if the message from raspberry pi is "some body is at the door!", then the app will send a request to ask pi to take a picture of the visitor and show the picture together with the message on the home page for 15 seconds.

On the server side, we have two function to listen to the status of the door and button, and we let them work on another thread by adding these two function to background.

```

110 def detect_input_change():
111     def input_change():
112         print(button.value)
113         print("Some one is at the door!")
114         socketio.send("Some one is at the door!")
115     button.when_activated = input_change
116     while True: pass
117
118 def detect_door_open():
119     def door_open():
120         print("door is opened")
121         socketio.send("The door is opened")
122
123     def door_close():
124         print("door is closed")
125         lock_door()
126         socketio.send("The door is closed")
127
128     magnet.when_activated = door_open
129     magnet.when_deactivated = door_close
130     while True: pass
131
132 if __name__ == '__main__':
133     socketio.start_background_task(detect_input_change)
134     socketio.start_background_task(detect_door_open)
135     socketio.run(host='0.0.0.0', port='80', app=app)

```

User Information
