

### HomeKit32

A modular smart-home automation framework for the Keystudio Smart Home Kit (KS5009), powered by ESP32, BLE, MQTT and B4X.

#### Overview

**HomeKit32** is a modular smart-home control system based on the Keystudio Smart Home Kit (KS5009) and the Keystudio ESP32 Plus microcontroller.

It provides real-time communication via BLE and MQTT, integrates 13 sensors and actuators, and exposes a clean, structured protocol for external applications.

The firmware is written in B4R, with performance-critical functionality implemented in wrapped C++ libraries.

Client applications currently exist for B4J, B4A and Python to complete the cross-platform ecosystem.

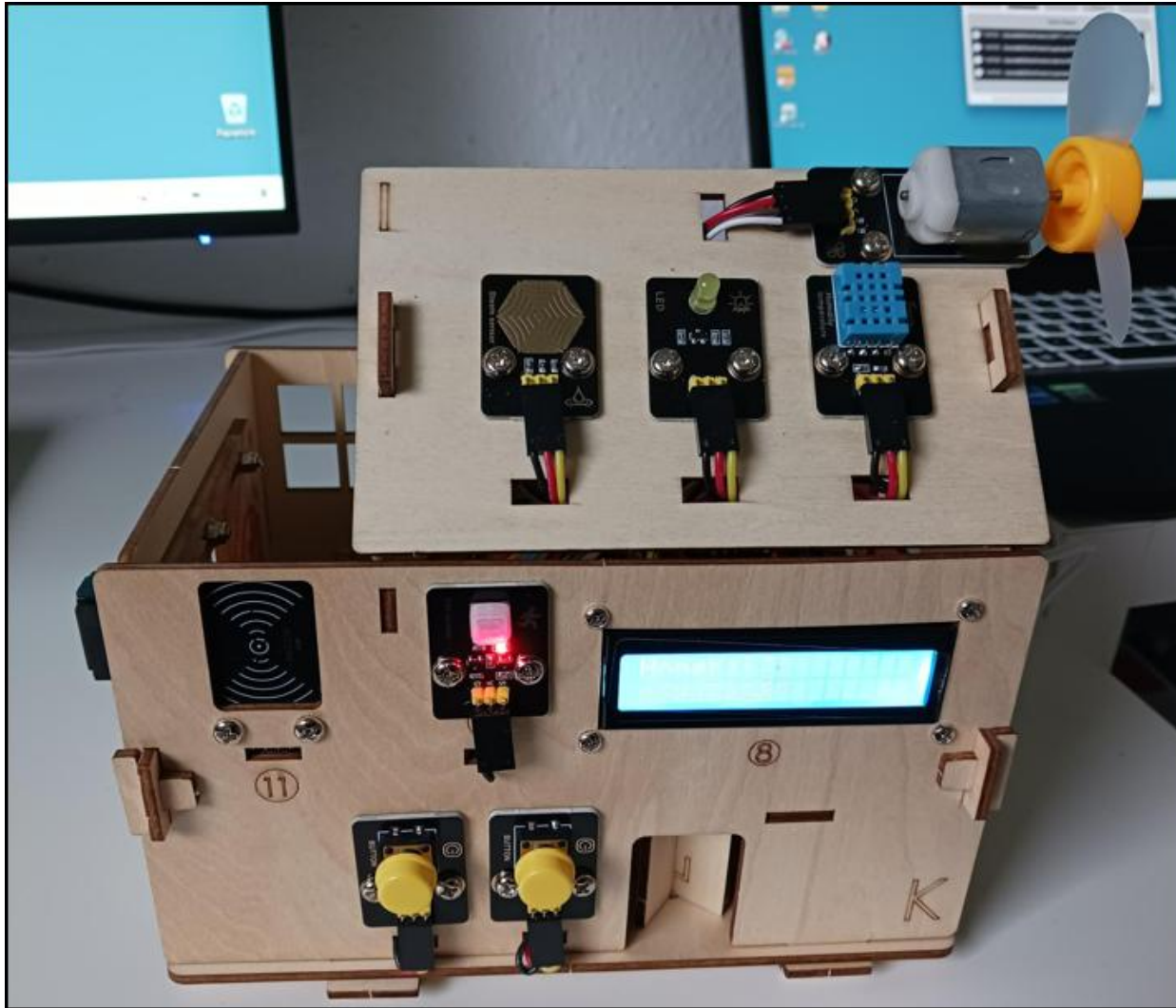
**HomeKit32** is ideal for education, prototyping, IoT experiments, and as a reference architecture for a clean, extensible, multi-protocol smart-home system.

### **IMPORTANT**

This example is for educational & personal use only.

It is NOT meant to be used for commercial purposes, nor to replace the original device.

# Keystudio Smart Home Kit ESP32



Kit assembled.

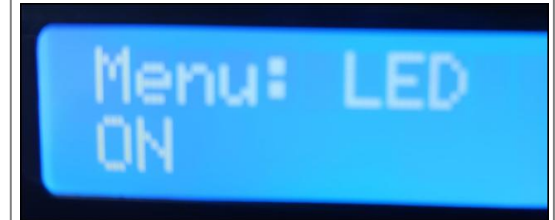
## Menu

Button left select menu item displayed on the LCD1602 (top row).

Button right select menu item state (bottom row).

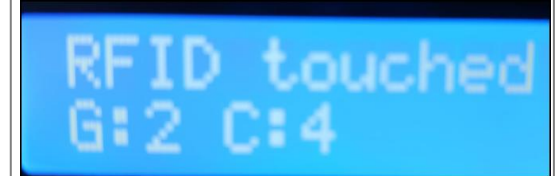
Example:

Menu: LED, state ON or OFF

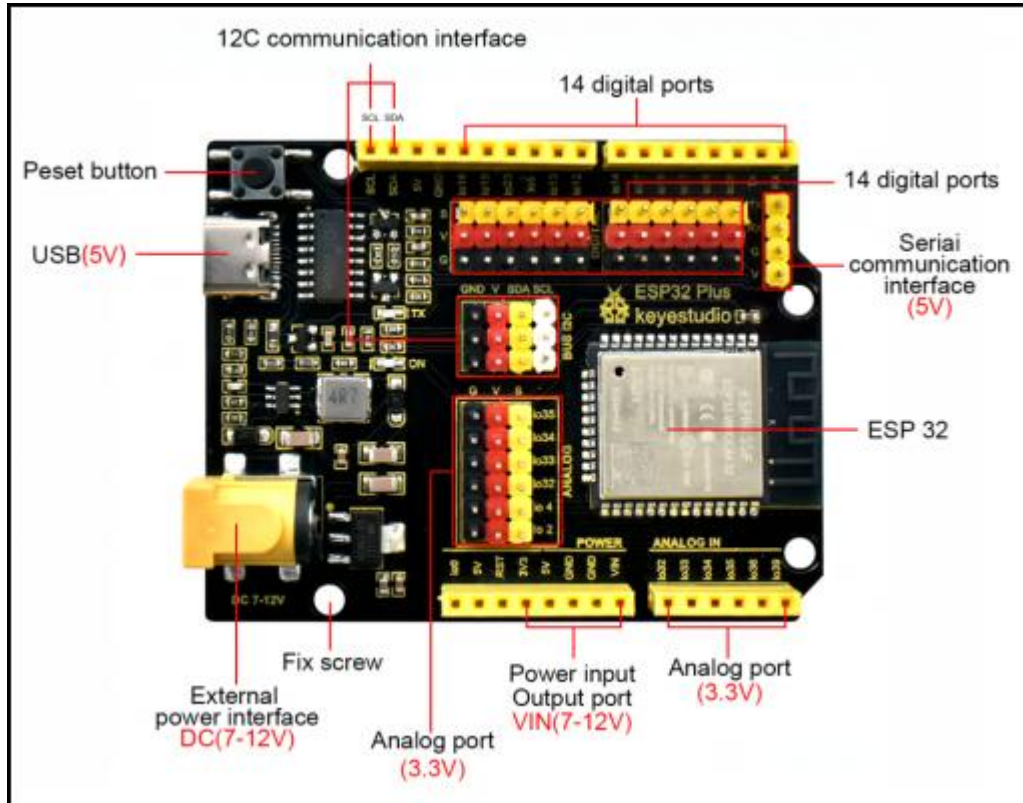


## RFID

Open or closed door when touched with tag holding group 2 and command 4.



# Keyestudio ESP32 Plus Pinout



## Pin mapping

===== LEDs =====

Public ONBOARDLED\_PIN As Byte = 2

Public YELLOW\_LED\_PIN As Int = 12

Public RGB\_LED\_PIN As Int = 26

===== Buttons =====

Public BTN\_LEFT\_PIN As Int = 16

Public BTN\_RIGHT\_PIN As Int = 27

===== PIR Sensor (Motion) =====

Public PIR\_SENSOR\_PIN As Int = 14

===== DHT11 Temp + Hum =====

Public DHT11\_PIN As Int = 17

===== Moisture Sensor (Analog) =====

Public MOISTURE\_SENSOR\_PIN As Int = 34

===== Gas Sensor (Analog) =====

Public GAS\_SENSOR\_PIN As Int = 23

===== Audio =====

Public BUZZER\_PIN As Int = 25

===== Fan =====

Public FAN\_DIRECTION\_PIN As Int = 19

Public FAN\_SPEED\_PIN As Int = 18

===== Servos =====

Public SERVO\_WINDOW\_PIN As Int = 5

Public SERVO\_DOOR\_PIN As Int = 13

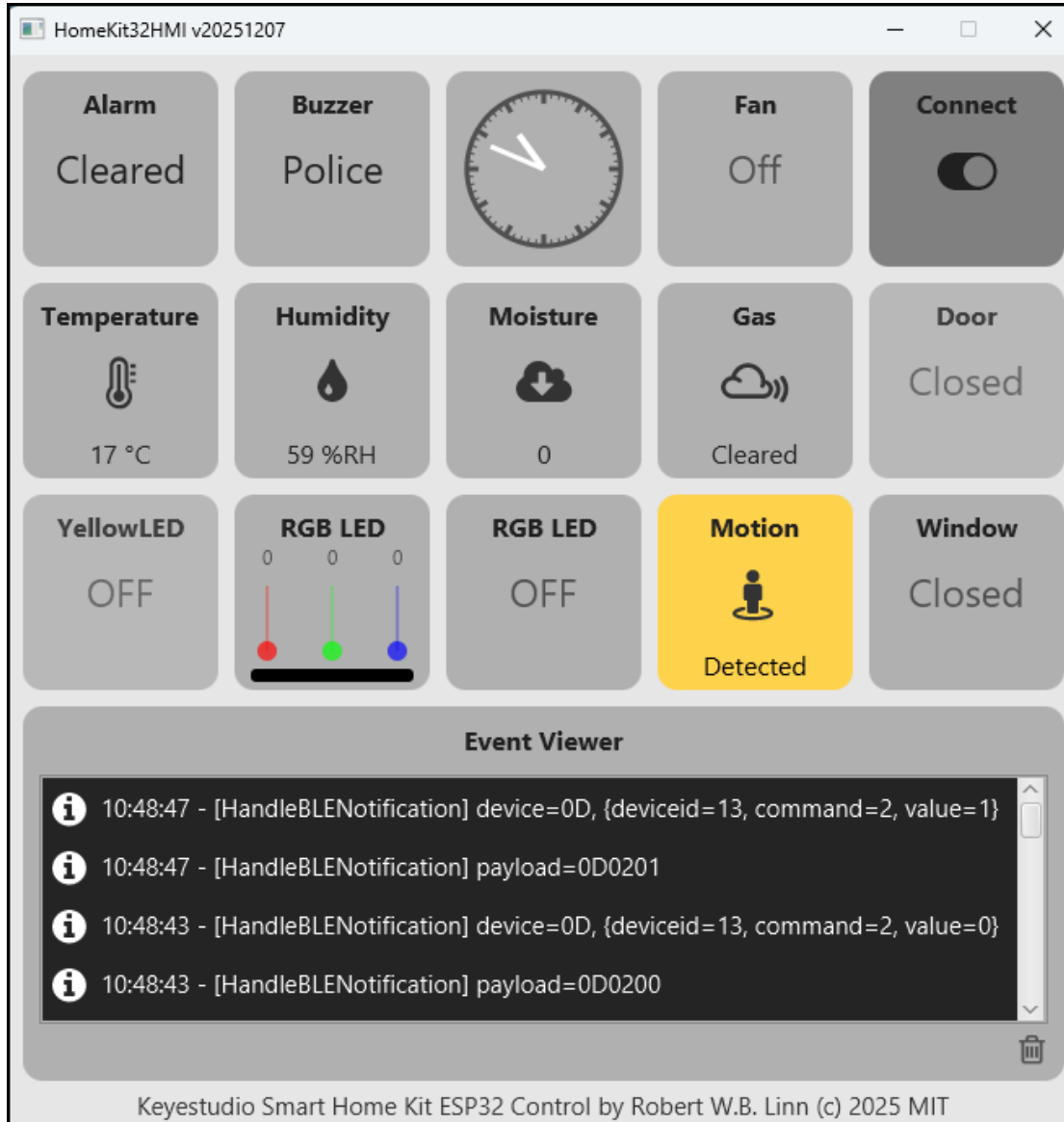
===== I2C Devices =====

Public RFID\_I2C\_ADDRESS As Byte = 0x28 ' RFID Mifare

Public LCD\_I2C\_ADDRESS As Byte = 0x27 ' LCD1602

[Source Keyestudio](#)

# B4J Client HK32HMI



## HMI to control devices over BLE.

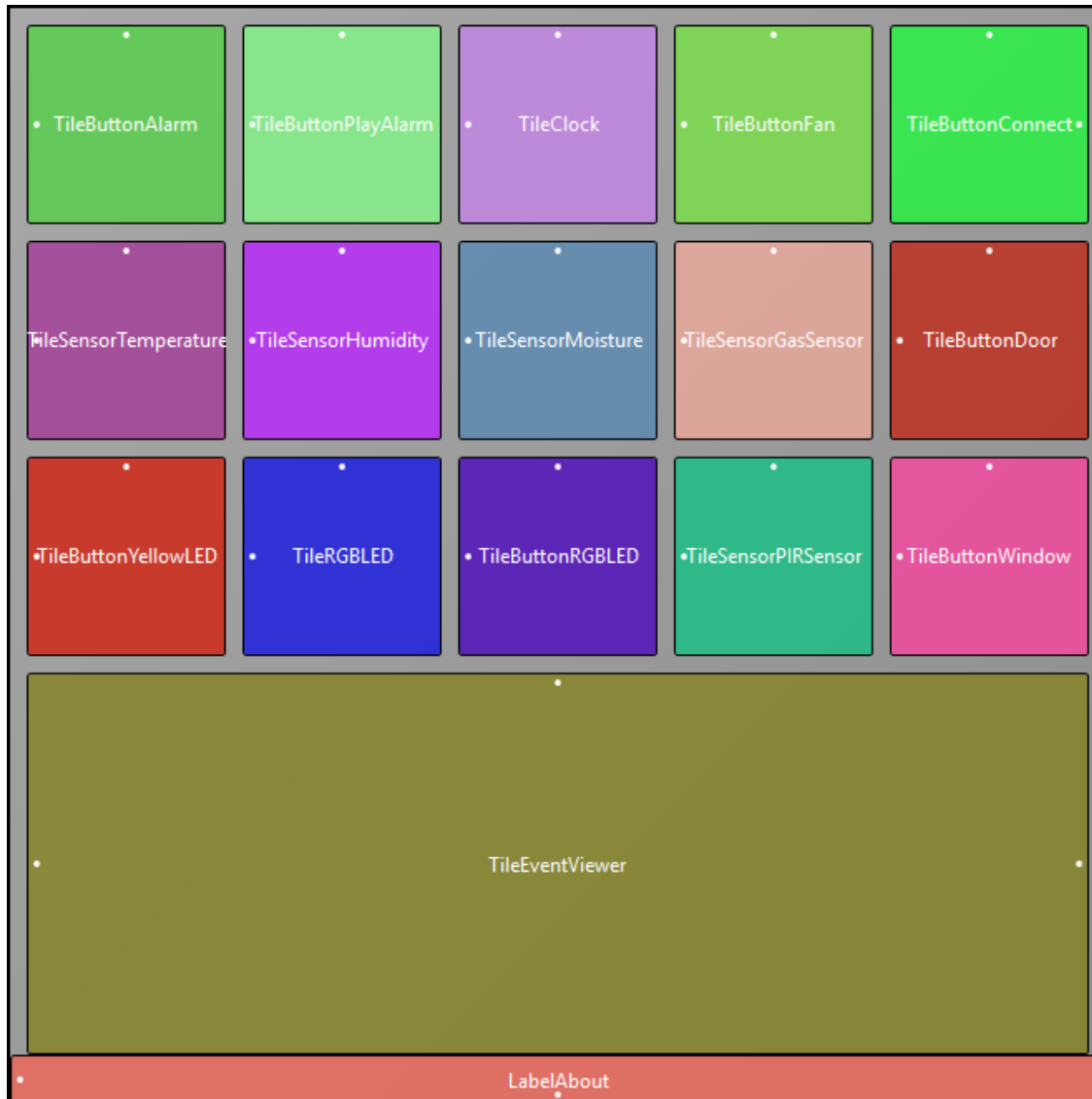
### B4J Project

- B4X Pages Project
- B4J Layout Single Page
- HMI Tiles (Custom Views) (Human-Machine Interface) Default 120px x 120px
- Event log (CustomListView)
- ISA-101 Guidelines aligned (Human-Machine Interface Design)

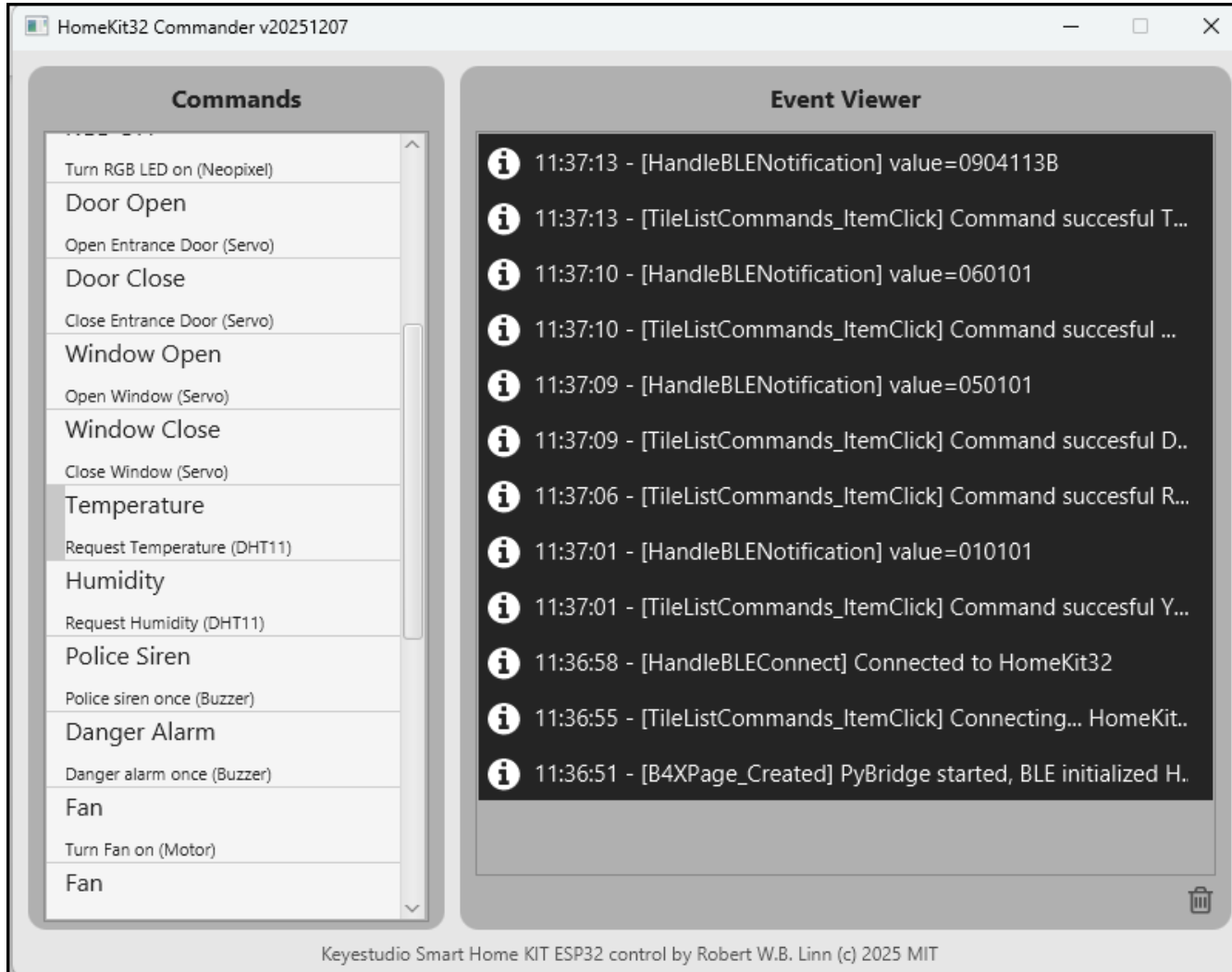
### Communication Mode

- BLE
- BLE Frame:  
[DeviceID][Command][Payload...]  
Optional response  
(GET/request commands):  
[DeviceID][Command][Response...]
- Example Payload switch YellowLED ON:  
Byte Array with 3 bytes:  
0x01 0x01 0x01  
Device-ID Command-ID Payload

# HK32HMI – B4J Layout Single Page



# B4J Client HK32Commander

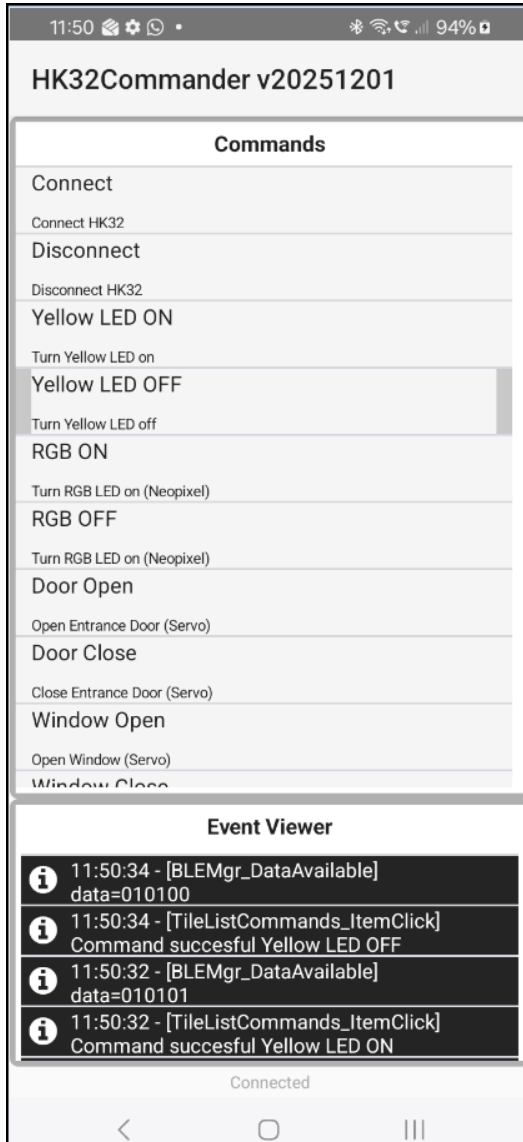


**Test Commands to control devices over BLE.**

## **B4J Project**

- B4X Pages Project
- B4J Layout Single Page
- Commands List (CustomListView)
- Event Viewer (CustomListView)

# B4A Client HK32Commander



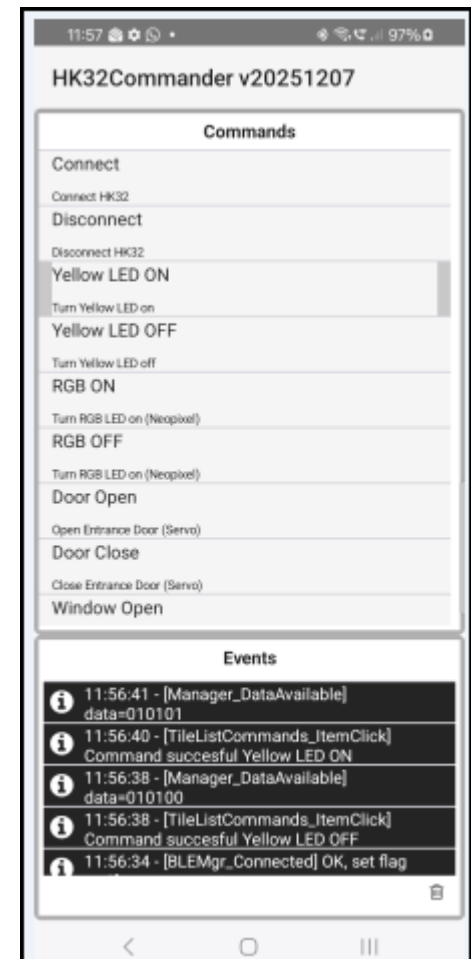
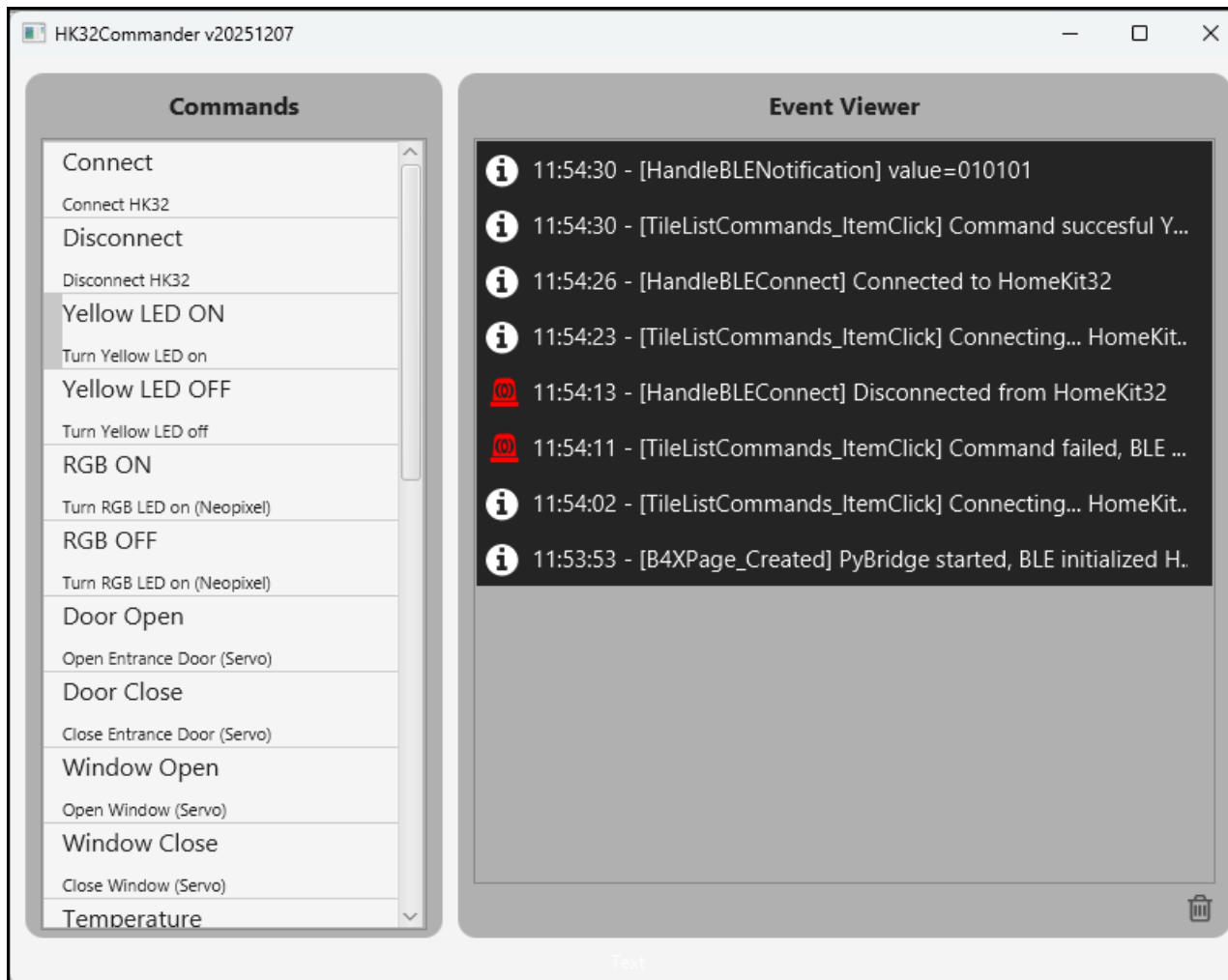
## Test Commands to control devices over BLE.

### B4A Project

- B4X Pages Project
- B4A Layout Single Page
- Commands List (CustomListView)
- Event Viewer (CustomListView)

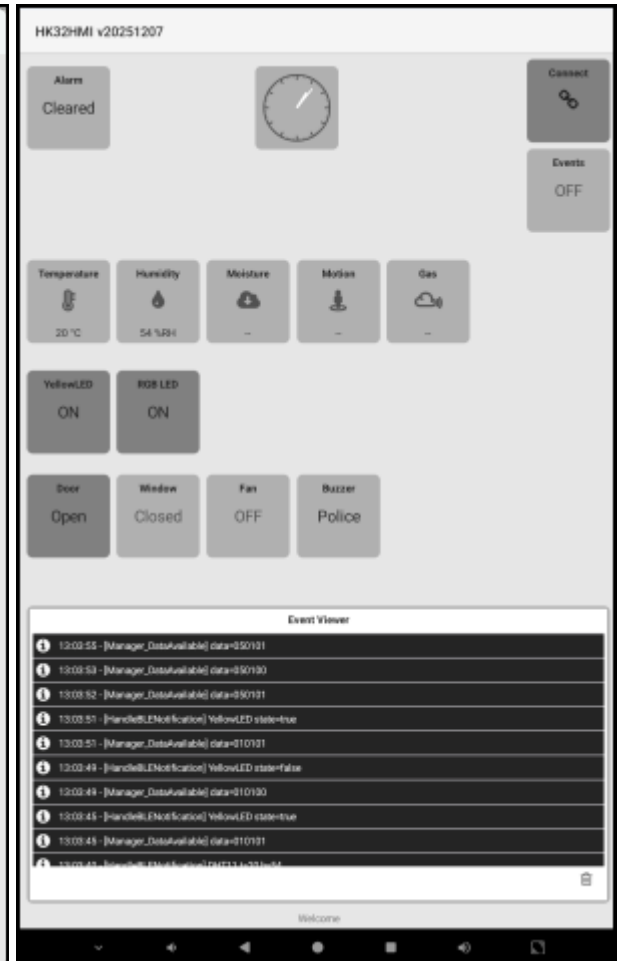
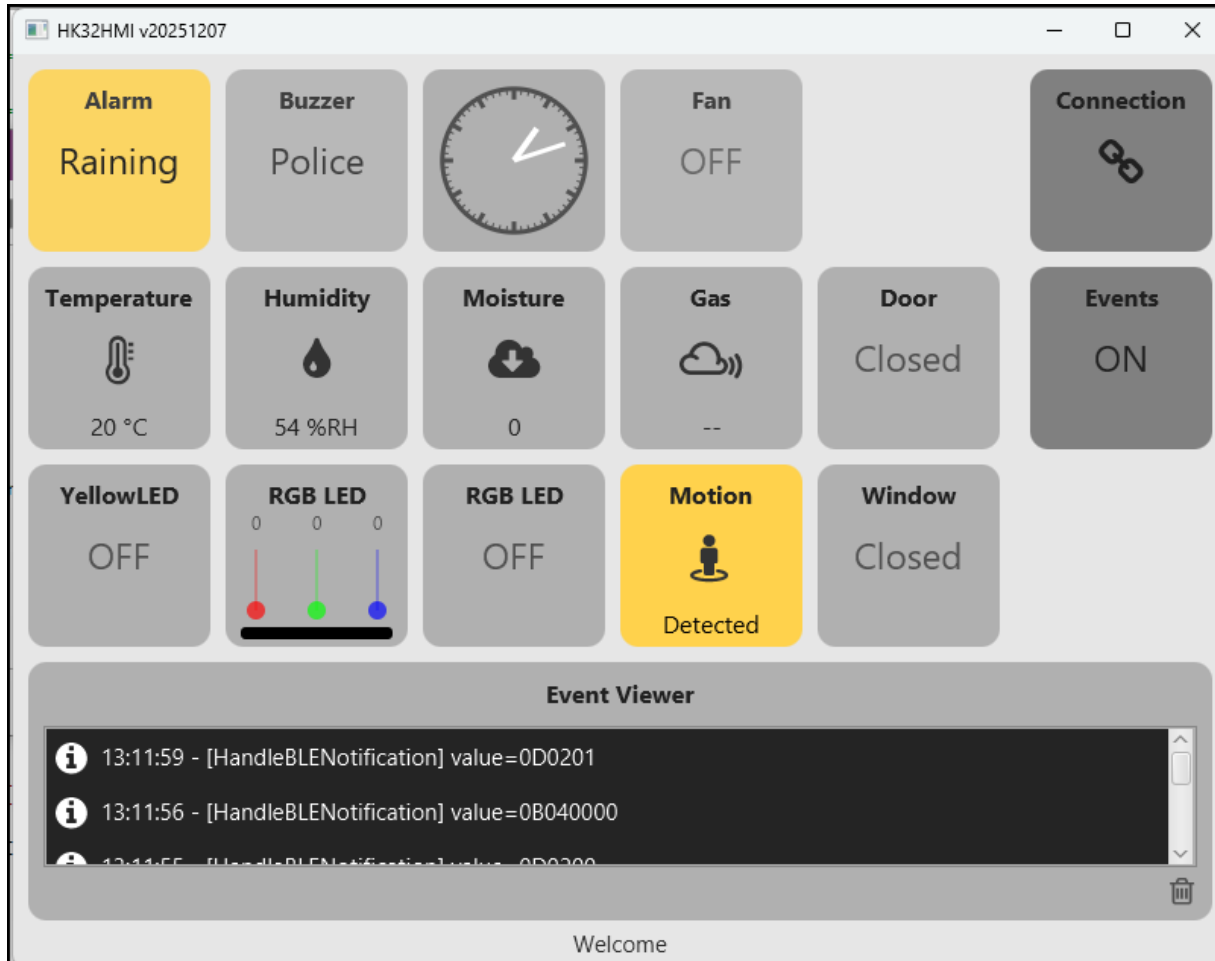
# B4X Clients HK32Commander

B4A & B4J BLE clients (B4X project) to test direct commands to control devices.



# B4X Clients HK32HMI

B4A & B4J BLE clients (B4X project) HMI.



# Python Client HK32HMI

Python with PySide6 framework example using same HMITile concept as B4X – selective TileButton & TileReadout.

TileButton

TileReadout

