BC7215AC

Arduino Air Conditioner Remote Control Library Application Examples

Index

The **BC7215AC Air Conditioner Remote Control Library** provides 5 example applications, each available in both English and Chinese versions:

- ESP8266 Serial Monitor (Blocking)

- ESP8266 Serial Monitor (Non-blocking)

- ESP32 Serial Monitor

- ESP32 LCD

- ESP32 MQTT

The **Serial Monitor version** is the simplest demo. It only requires connecting any ESP8266 or ESP32 Arduino development board to the BC7215A IR transceiver module, then using the Arduino IDE's built-in Serial Monitor as the human–machine interface to control the air conditioner.

The **LCD** and **MQTT** versions require the LilyGO TTGO T-Display ESP32 board, which comes with a built-in LCD display and two physical buttons. These hardware features allow the demo programs to run independently without a PC.

The **MQTT version** builds on the LCD version by adding MQTT networking support. This enables users to test network-based air conditioner control via a public MQTT broker. The MQTT version still retains local button control and can report the updated air conditioner state to the MQTT server.

The examples use ESP8266 NodeMCU boards and ESP32 TTGO T-Display boards. The hardware connections are as follows:
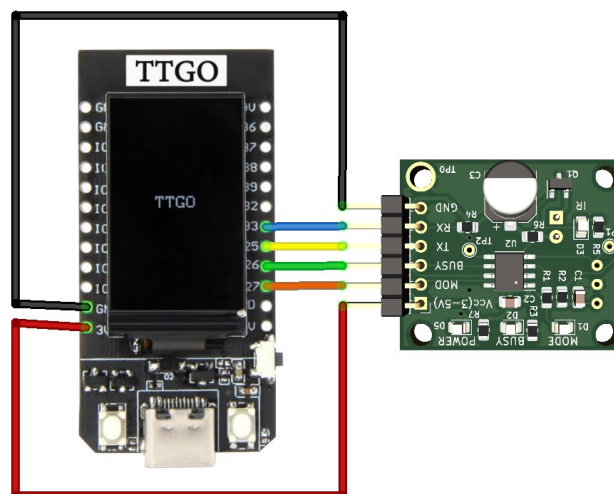
ESP8266:

GPIO5 → BC7215A TX

- GPIO16 → BC7215A RX

- GPIO0 → BC7215A MOD

- GPIO4 → BC7215A BUSY

- 3.3V → BC7215A VCC
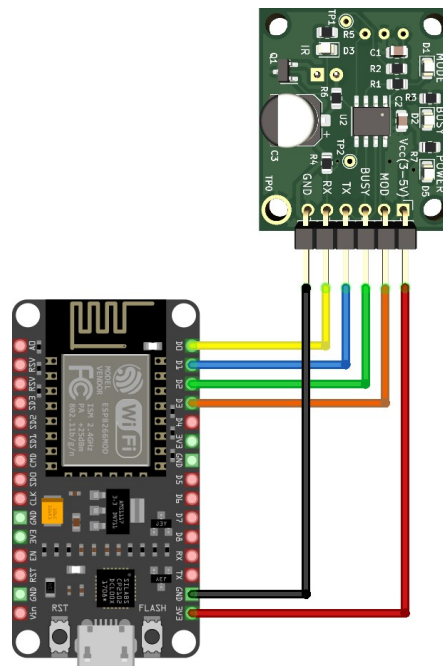
ESP32:

GPIO25 → BC7215A TX

- GPIO33 → BC7215A RX

- GPIO27 → BC7215A MOD

- GPIO26 → BC7215A BUSY

- 3.3V → BC7215A VCC

Serial Monitor Version

This version uses the Arduino IDE Serial Monitor (baud rate: 115200). Three program variants are provided:

- ESP8266 Blocking
- ESP8266 Non-blocking
- ESP32

The **blocking version** is simplest in logic: the program loops indefinitely until a condition is met (e.g., waiting for user input). It's useful for understanding the library's workflow.
The **non-blocking version** uses a state machine so tasks can proceed in parallel without getting stuck.
The **ESP32 version** is a port of the ESP8266 non-blocking version.

After uploading the program, the main menu appears in the Serial Monitor. If not, simply press Enter or reset the Arduino board.
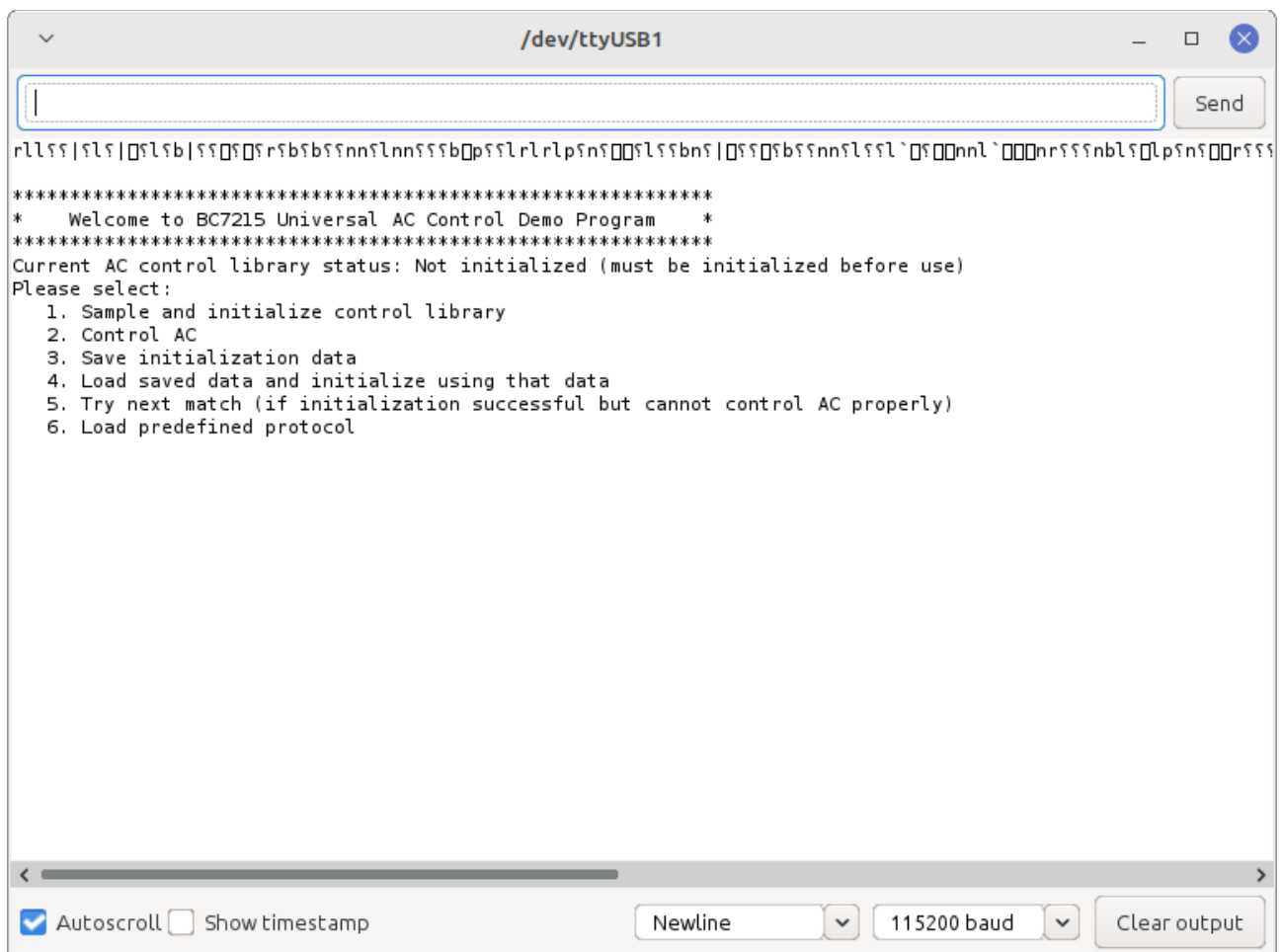
Initialization

First-time use requires **sampling the original air conditioner remote control**. The collected data is used to initialize the library. The process is step-by-step, guided on-screen. If initialization fails repeatedly, the AC model may be one of the rare types that BC7215A cannot directly decode. In this case, you can try using **predefined protocols** to test control.

```
rll⌐⌐|⌐l⌐|∏⌐l⌐b|⌐⌐∏⌐∏⌐r⌐b⌐b⌐⌐nn⌐lnn⌐⌐⌐b∏p⌐⌐lrlrlp⌐n⌐∏∏⌐l⌐⌐bn⌐|∏⌐⌐∏⌐b⌐⌐nn⌐l⌐⌐l`∏⌐∏∏nnl`∏∏∏nr⌐⌐⌐nbl⌐∏lp⌐n⌐∏∏r⌐⌐⌐


*****************************************************************
*     Welcome to BC7215 Universal AC Control Demo Program      *
*****************************************************************
Current AC control library status: Not initialized (must be initialized before use)
Please select:
    1. Sample and initialize control library
    2. Control AC
    3. Save initialization data
    4. Load saved data and initialize using that data
    5. Try next match (if initialization successful but cannot control AC properly)
    6. Load predefined protocol


A few protocols are not supported for direct decoding by BC7215A chip, provided as predefined protocols.
When direct sampling fails or initialization fails, you can try using predefined data to control the AC.
Brand names listed in protocol names are for reference only. If unable to control, please try all predefined p
Please select:
    0. M96b (XIAOMI)
    1. M100b (SHINCO/SAMSUNG/ELECTROLUX)
    2. D131b (RSD/MCQUAY/TICA)
    3. T102b (WHIRLPOOL/BOSCH/AIRWELL)
    4. M128b (FUJITSU/McQUAY)

Selected type 3
66 0 1C 14 0 0 87 5 0 0 21 C1 40
21 10 BD 6 5D 6 5D 2B 9D 1F 19 35 4D 1A A7 72 F9 0 0 22 0 44 0 66 90 D 0 A 0 48 0 39 0
Initialization successful !!! Press any key to continue
```

Once initialized successfully, you can start controlling the AC through a **two-level menu**:

1. Choose control type (parameters such as temperature, or power on/off).

2. Enter parameter values (temperature, mode, fan speed).

```
/dev/ttyUSB1                                    _  □  ⊗

[|                                                    ]  Send

AC Control, please select:
   1. Set AC parameters
   2. Power ON
   3. Power OFF
   4. Return to upper menu

AC Parameter Adjustment: Please enter the temperature, e.g.: 24 ;
or enter 'temperature', 'mode', 'fan level' separated by commas ',', e.g.: 24, 1, 2
 Ranges and meanings:  Temperature(°C)     Mode        Fan Speed
                       Range: 16~30        0 - Auto    0 - Auto
                                           1 - Cool    1 - Low
                                           2 - Heat    2 - Medium
                                           3 - Dry     3 - High
                                           4 - Fan
 * Values outside the above ranges indicate maintaining current status for that item
---------------------------------------------------------------------------------
(Note: Limited to settings supported by controlled AC.
For example: setting heat mode on cooling-only AC will have unpredictable results)
Please enter AC parameter values: (Enter 'exit' to return to upper menu, case insensitive)

18
Sending command to set AC to: 18°C
Sending data: 49 2 20 50 A 0 1 80 4B 12 0 81 13 0 0 4 18
Send complete!

Please continue input
24, 0, 2
Sending command to set AC to: 24°C, Mode: Auto  Fan Speed: Medium
Sending data: 78 8 20 50 A 0 1 0 C6 43 0 81 13 0 0 C 34
Send complete!

Please continue input

☑ Autoscroll □ Show timestamp        Newline  ⌄   115200 baud  ⌄   Clear output
```

ESP32 LCD Version

This version uses the TTGO T-Display Arduino board (135×240 ST7789 LCD). It relies on **Bodmer's TFT_eSPI library**, installable via Arduino IDE Library Manager.

You must configure the library by replacing `User_Setup.h` in TFT_eSPI with the one provided in the `extras/config` directory of the BC7215AC library.

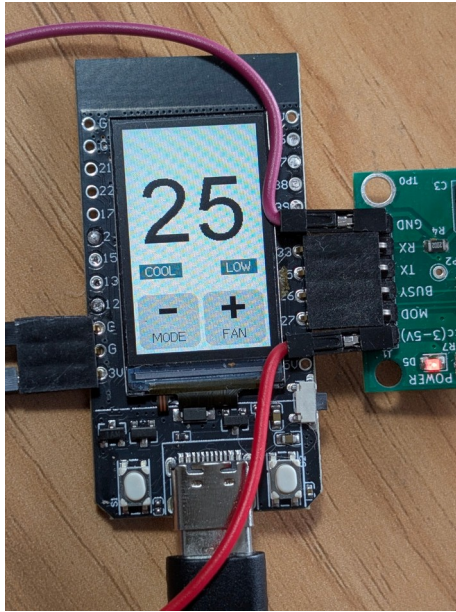When the program runs, the menu appears:

- Left button (**SEL**) selects items.
- Right button (**OK**) confirms.
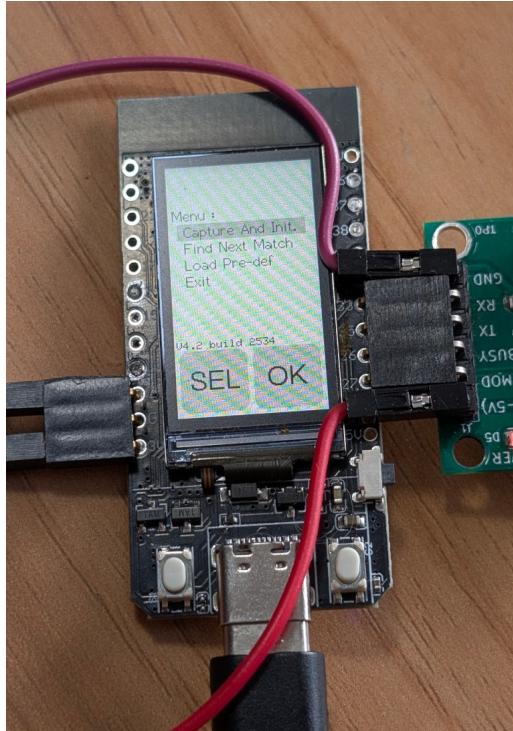
Initialization

First, perform initialization by sampling the AC remote.
If successful, the program enters the **AC control page**:

Button functions:

- Left short press: Decrease temperature

- Right short press: Increase temperature

- Left long press: Switch mode

- Right long press: Switch fan speed

- Double short press: Enter power control page (left = power on, right = power off)

- Double long press: Enter main menu

Whenever an action changes AC state, BC7215A transmits the corresponding IR signal, and an indicator appears at the top-right of the screen.

ESP32 MQTT Version

The MQTT version extends the LCD version with:

1. **MQTT-based network control**
2. **Air conditioner status reporting**

It uses **Nick O'Leary's PubSubClient library** (available in Arduino IDE Library Manager).

1. Preparation Before Compilation

Before compiling, uncomment the following lines in the source and replace with your own values:

```
// WiFi and device configuration - replace with your values
#define MY_WIFI_SSID      "Your WiFi Name"
```

```
#define MY_WIFI_PASSWORD "Your WiFi Password"

#define MY_UUID          "Your UUID"
```

Use a UUID generator to create a unique device ID. If two devices share the same UUID, one will be disconnected by the MQTT server.
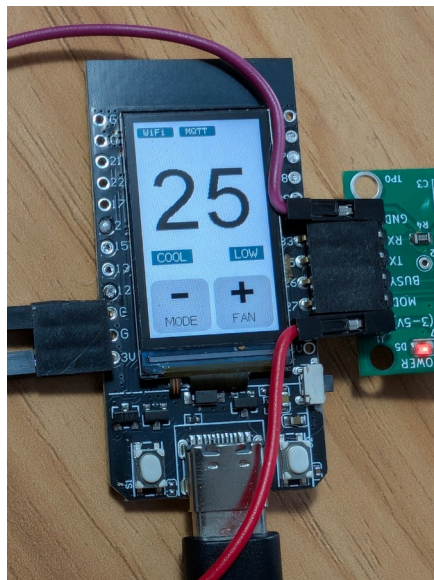Example UUID: `b1225e25-81c8-43d7-8183-6f5793408242`

Default MQTT broker: `broker.hivemq.com`
(You may also use other public brokers such as `broker.emqx.io`).

⚠️ **Note:** This demo uses an unencrypted connection. For production, use TLS-encrypted MQTT.

2. Network-Controlled AC

On startup, the device connects to WiFi, then to the MQTT server. Once connected, WiFi and MQTT icons appear on the screen.

MQTT Topics:

- Temperature: `BC7215A/<UUID>/var/temp`

- Mode: `BC7215A/<UUID>/var/mode`

- Fan speed: `BC7215A/<UUID>/var/fan`

- Power: `BC7215A/<UUID>/var/power`

Example:
`BC7215A/b1225e25-81c8-43d7-8183-6f5793408242/var/temp`

Message formats:

- **Temperature:** `"16"` – `"30"`

- **Mode:**

    - 0 = Auto

- 1 = Cool

- 2 = Heat

- 3 = Dry

- 4 = Fan

- **Fan:**

  - 0 = Auto

  - 1 = Low

  - 2 = Medium

  - 3 = High

- **Power:**

  - 0 = Off

  - 1 = On

MQTT Client

Most free public brokers provide web or desktop MQTT clients. Any client can publish control messages as long as it connects to the same server and topics.

Example HiveMQ web client:
https://www.hivemq.com/demos/websocket-client/

AC Status Reporting

Whenever an IR command is sent (via button or MQTT), the new AC state is also reported to the MQTT server. Clients subscribed to the **report topic** will receive updates:

```
BC7215A/<UUID>/var/report
```