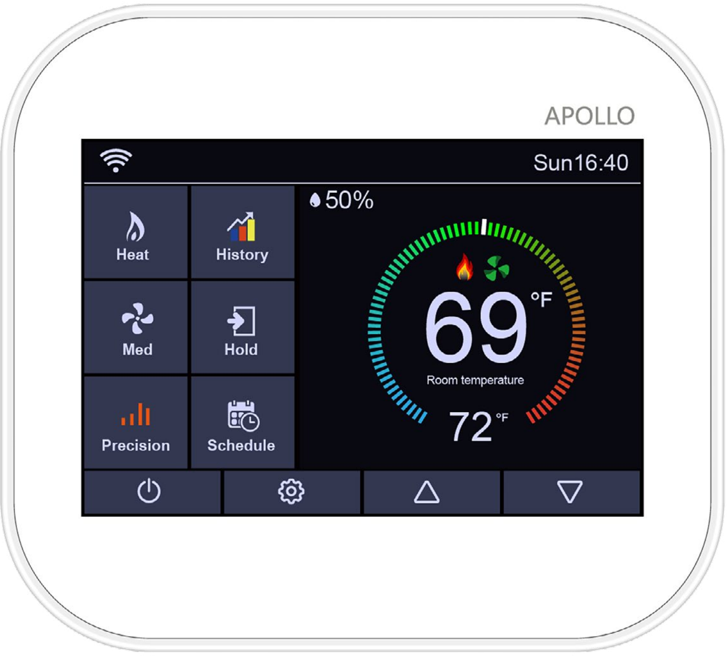
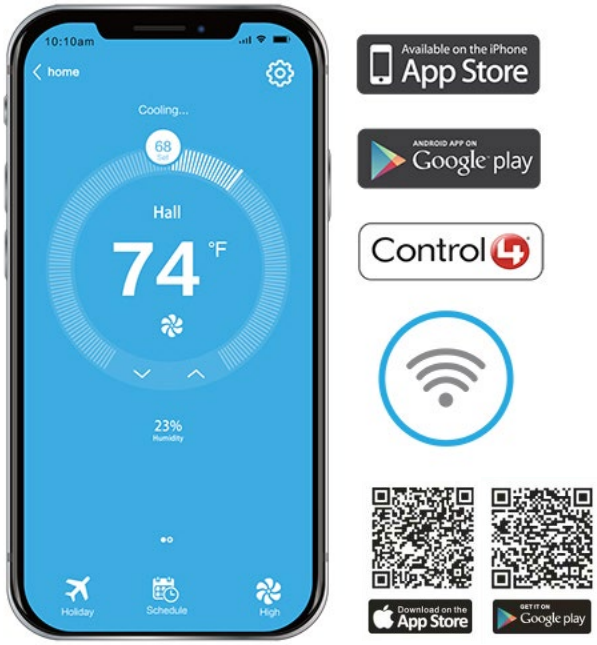
APOLLO Fan Coil Thermostat

FAN COIL THERMOSTAT





MAIN FEATURE

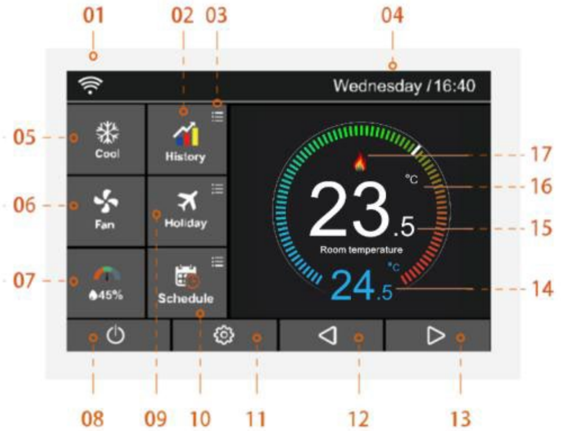
- 12 to 24VAC/DC or 110 to 240VAC
- 3.5" TFT color touch screen
- Ambient light sensor
- Humidity sensor
- Wi-Fi connective(option)
- GEO-fence
- Boost output
- 3 speed or 0-10VAC fan control
- On/Off relay or 0-10VAC valve control
- 2/4 pipes HVAC system
- On/off or 0-10v valve for heating and cooling.
- Manual or auto changeover
- 7 day, weekday/weekend, 24hours schedule
- Four or six events per day
- Auto sync time and date
- Automatically adjusts to daylight saving time
- Defrost protection
- Modbus RS485, modbus TCP(option)
- Holiday mode
- Multi-language (9 languages)
- C/F temperature display
- Touch sceen lockout

TECHNICAL DATA

Purpose Of Control	2/4 Fan Coil Unit
Supply Voltage	12 to 24VAC/DC or 110 to 240VAC
Relay Rating	5 Amp maximum per relay
Output Relay	SPST – NO
Display	320x480 pixels TFT, Capacitive touch screen
Communications	Modbus RTU, Modbus TCP
Displayed Temperature Resolution	0.1°C
Control Temperature Step	0.2°C (default = 0.5°C)
Wire Size, Terminals	Current ≤5 A – 1.5 mm^2, solid core wire
Smart Schedule Type	7days, weekday/weekend, 24hours
Schedule Per Day	Up to 7 different 4- or 6-event program
Standby Consumption	≤0.5 w
Wifi(Option)	IEEE 802.11 b/g/n – 2.4ghz
Security	WPA/WPA2
Operating Temperature	1° c - 85° c
Dimensions (W/H/D)	96 * 86 * 13.8 (mm)
Build-In Depth	24.5 mm
Weight	≤200 g
Enclosure Rating	IP 21
Enclosure Material	PC + ABS plastic
Alexa	Optional
Google Home	Optional
Approval	CE, FCC, RoHS,CA65

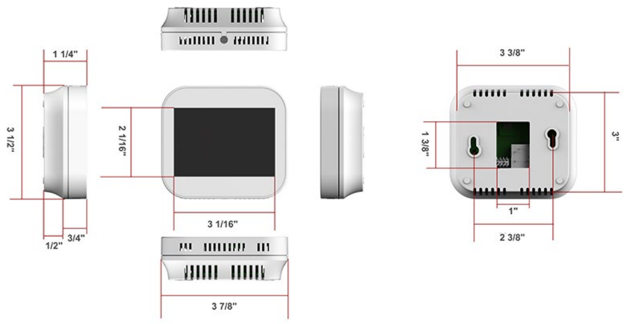
FAN COIL THERMOSTAT

MAIN FEATURE



1. Wi-Fi indicator 2. History data diagram 3. Adjusting the optional settings 4. Day indicator & clock 5. System mode: heat, cool ,auto, off 6. Fan :auto, high, med, low 7. Humidity 8. Power on/off 9. Holiday 10. Schedule 11. Configuration settings 12-13. Adjusting the setting temperature 14. Setting temperature display 15. Room temperature display 16. "C" or "F"display 17. "heating/ cooling/vent" symbol

OUTLINE DIMENSION



CONFIGURATION SETTINGS

Language	English, Chinese, Spain, Italian, Russian, Polish, Czech, German, French, Slovenski, Swedish, Portuguese
Date & Time	01) Time Zone 02) Date & Time 03) Daylight saving Time(DST)
Display Brightness	Ambient Light Diming
Screen Saver	Standby Screen 01) Time: 3min, 10min, Just night 02) Type: Clock , default, Off screen
Lock screen	4 number PIN; Disable/enable
Network Settings	01) WiFi Setup 02) Pair to App(Smart Config) 03) QR Code 04) Mac Address 05) Modbus Address(Option)
Feature Settings	01) Temperature Format 02) Temperature Limit 03) Switch differential 04) Output delay 05) Defrost: Defrost temperature; Enable/disable 06) Build-in sensor calibration 07) Humidity sensor calibration 08) Optimum Start 09) Energy Tariff: KW/H, $/kwh 10) Open Window function 11) Fan setting: 0-10v Fan; 3 speed Fan; EC, Fan work mode 12) Valve setting: Switch on/off valve 0-10v valve 13) Input signal setting: Disable/enable input, Input signal type
Schedule	Weekday/Weekend, 7 Days, 24 Hours, None
Hold Temperature	Hold time, Hold setpoint
System Information	Version number, QR code
Factory reset	Reset to factory settings

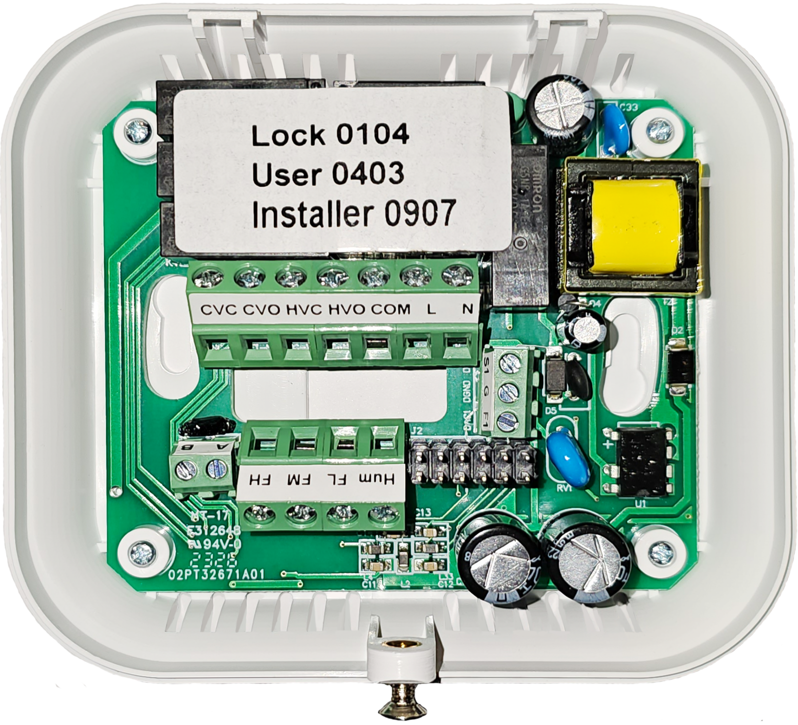
PRODUCT MODEL

White

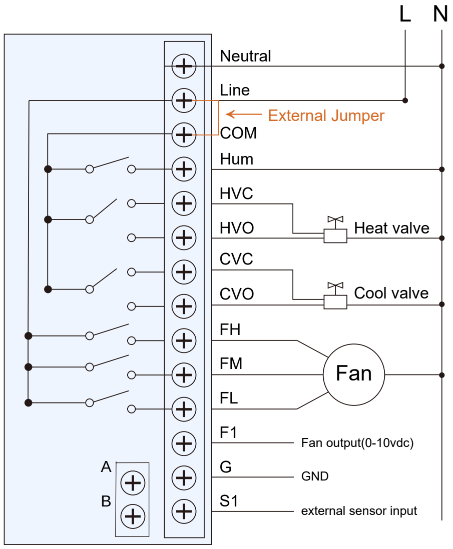
Black on special order

FAN COIL THERMOSTAT

CONNECTION BOARD



WIRING DIAGRAM

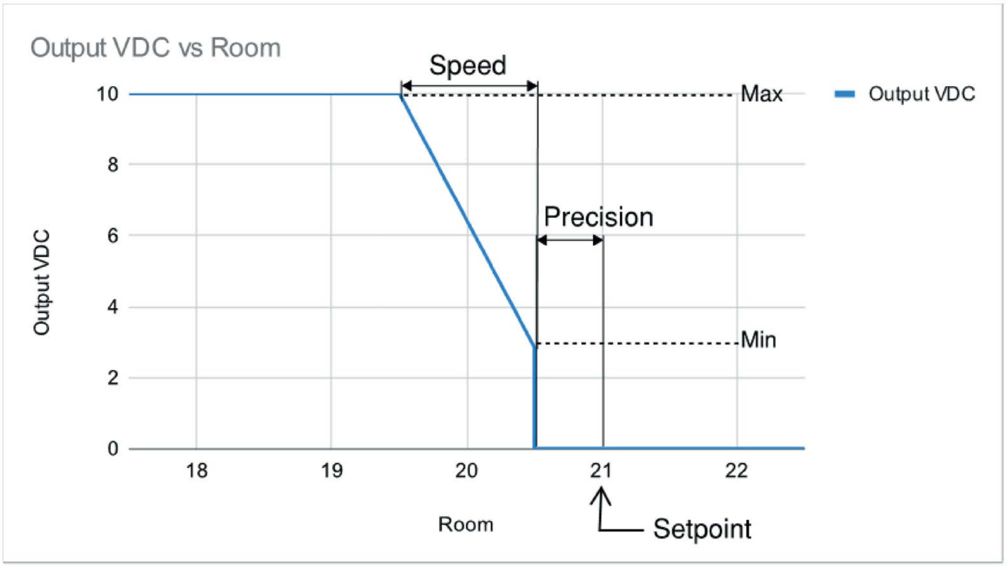


Terminal	Description	Terminal	Description
L	Line	CVO	Cool Valve NO
N	Neutral	CVC	Cool Valve NC
FL	Fan Low	Hum	Humidity relay
FM	Fan Med	COM	Common for valve
FH	Fan High	F1	Fan 0-10vdc
HVO	Heat Valve NO	G	Ground
HVC	Heat Valve NC	S1	External sensor
A	Bacnet Terminal	B	Bacnet Terminal

*Make sure external jumper L & COM to power valves

OPERATION EXAMPLE

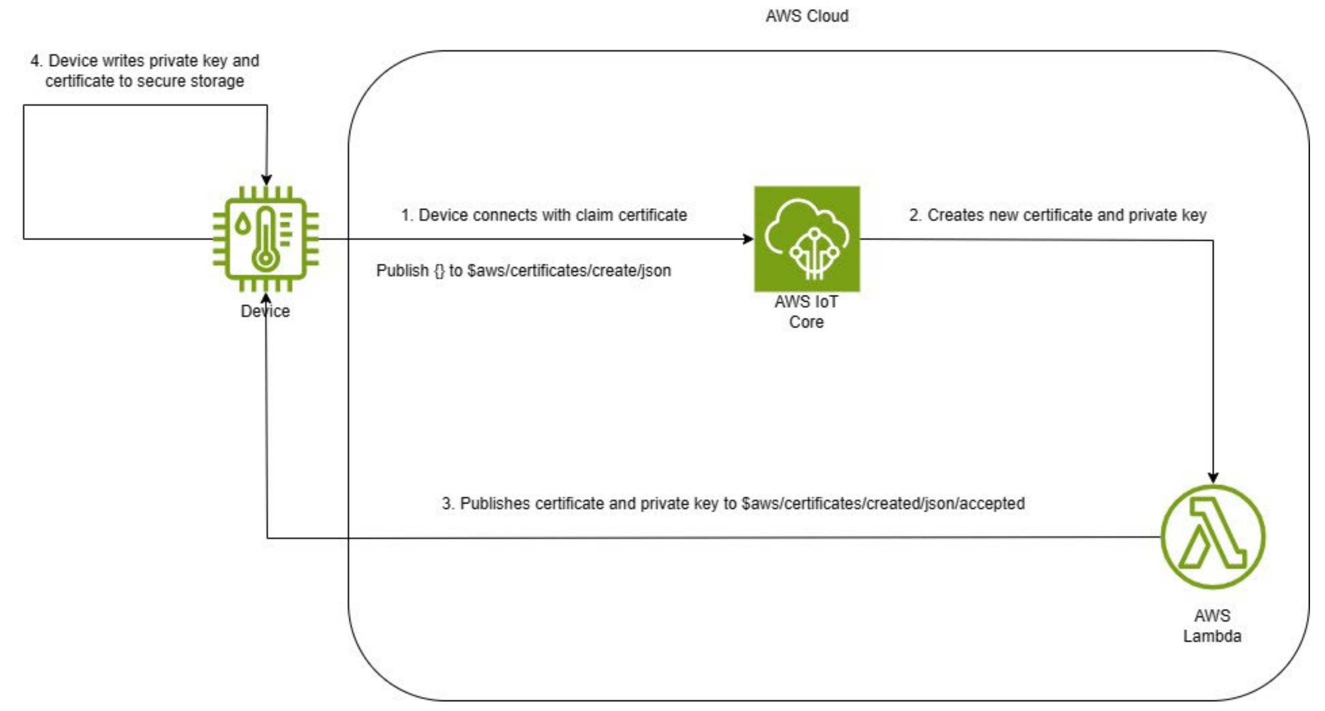
Mode	Min Fan	Max Fan	Precision	Speed	Setpoint	Room	Output VDC
Heating	30%	100%	0.5C	1C	21C	22.5	0
						22	0
						21.5	0
						21	0
						20.5	3
						20	6.5
						19.5	10
						19	10
						18.5	10
						18	10
						17.5	10



Device connection process

We are going to use TLS securely sockets for connection between devices and server part as a transport for MQTT protocol. Amazon recommends this. So, before we can connect our devices to server, we have to receive private key and certificate. For doing this we just publish a special kind of message - {} in the topic with predefined name: “$aws/certificates/create/json” . Message Broker from AWS IoT Core has already subscribed to this topic, so it reads message and starts Lambda. Lamdba uses AWS IoT fleet provisioning. It generates device certificate and private key that signed by the Amazon Root certificate.

This mechanism we use when device don’t have its own device certificates. This mechanism can be extended, for instance Lambda can implement some additional device verification logic.
When certificate and private key are ready Lambda publish them in predefined topic with name “$aws/certificates/created/json/accepted”. Device subscribed to this topic, so it receives key and certificate and stores it in secure storage. That’s all, now our device can work with server.

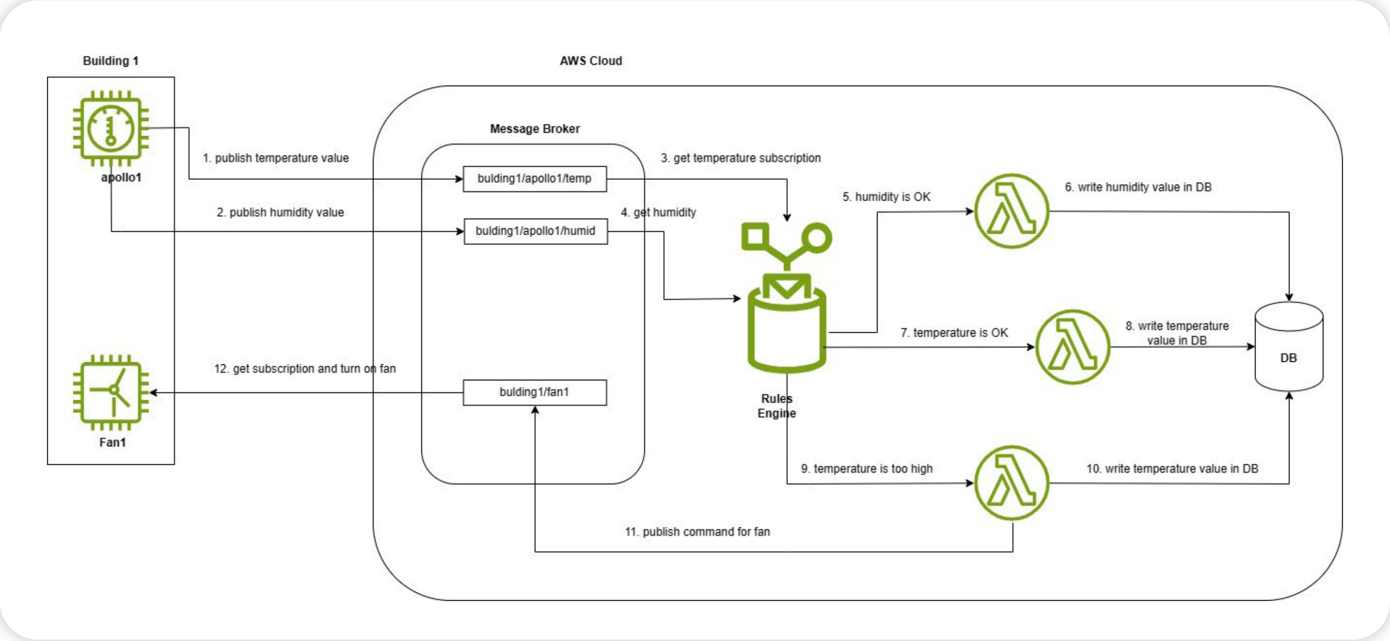

AWS Cloud
4. Device writes private key and certificate to secure storage
Device
1. Device connects with claim certificate
Publish {} to $aws/certificates/create/json
AWS IoT Core
2. Creates new certificate and private key
3. Publishes certificate and private key to $aws/certificates/created/json/accepted
AWS Lambda

How it works

As an example, for the architecture explanation, we will use a building with 1 Apollo device and 1 Fan. We executed connection process for both devices.

Our apollo device sends temperature and humidity values to the server every 10 seconds, for instance. It publishes them to the topics named “building1/apollo1/temp” and   “building1/apollo1/hymid” respectively. Rules Engine, another AWS IoT Core component is subscribed to both these topics. Rules Engine reads humidity value and starts Lambda that writes this value to Database. In this example we don’t have rule for humidity value. Rules Engine reads temperature value and check if it belongs to some interval. If temperature value higher than high bound of the interval Rules Engine will start Lambda that publish message with command for Fan in appropriate topic. Fun device is subscribed to this topic, it receives command and turns on. In any case temperature value will be written in DB.

We don’t specify which DB will be used. It depends on what you are going to do with gathered data.


Building 1
AWS Cloud
apollo1
Fan1
Message Broker
1. publish temperature value
2. publish humidity value
bulding1/apollo1/temp
bulding1/apollo1/humid
bulding1/fan1
12. get subscription and turn on fan
3. get temperature subscription
4. get humidity
Rules Engine
5. humidity is OK
6. write humidity value in DB
7. temperature is OK
8. write temperature value in DB
DB
9. temperature is too high
10. write temperature value in DB
11. publish command for fan

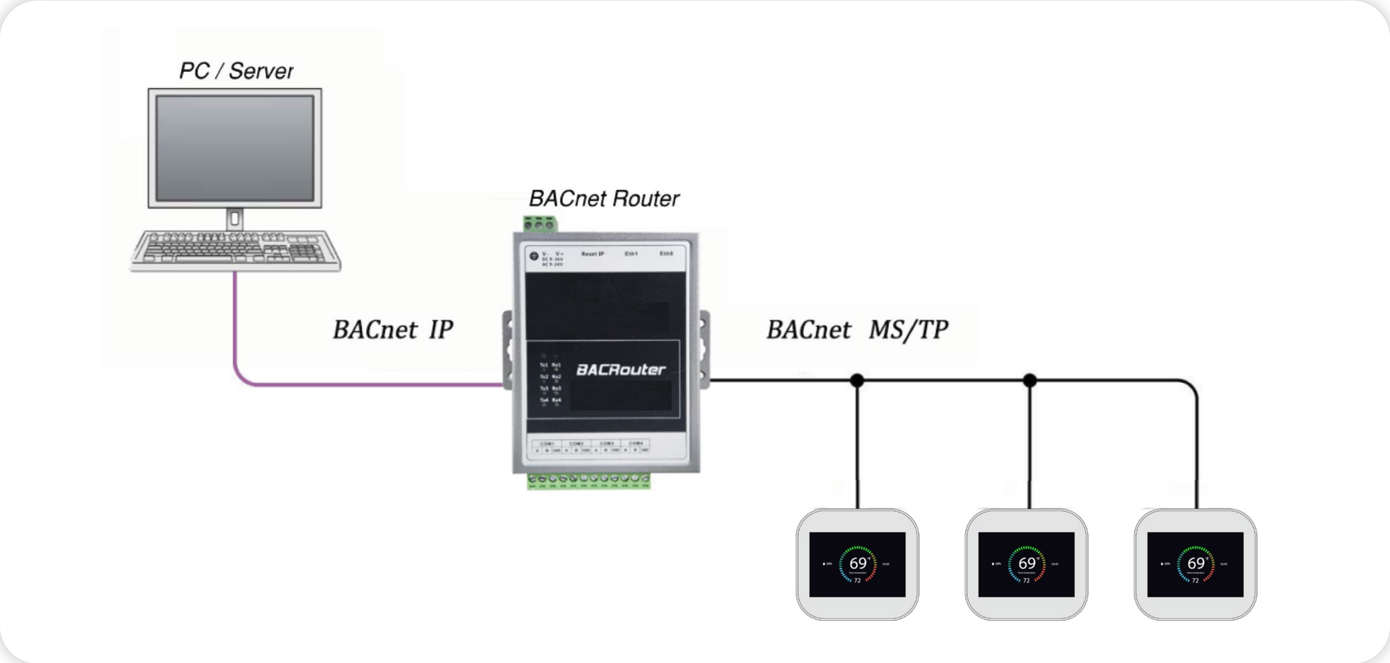
Thermostat integration with MQTT server

1)Unit publish topic: updData/MacAddress
2)Unit subscibe topic: MACaddress
server sends command(setpoint, turn on/off) from cloud server to MQTT broker:
server should publish msg(settings) to this topic is only the mac address , so that thermostat could subscribe msg(settings) from this topic “mac address”

The red part is controlable keys
{“msgid”:4,”mac”:”308398abea7d”,”version”:5,”temp”:210,”humi”:350,”settemp”:260,”mode”:5,”onoff”:1,”frost”:70,”delay”:0,”diff”:10,”holdtemp”:260,”holdtime”:0,”kb”:1,”kbkey”:”0000”,”cf”:0,”holiday”:0,”holiday_startime”:0,”holiday_endtime”:0,”standby”:2,”fan”:4,”timezone”:13,”prog”:0,”temp_prog”:{}}

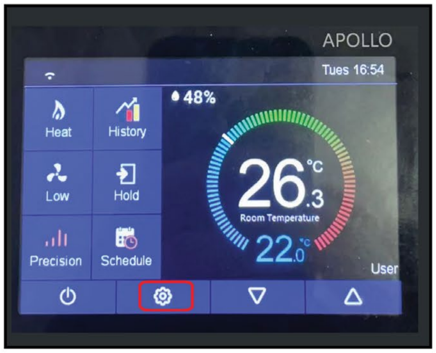
KEY	COMMENT	TYPE	SAMPLE		
mac	mac address	String	3 ways		
temp	display temperature(upload after multiplying by 10)	int	0-10v	{“temp”:200} means the current room temperature is 20, all the multiplied by 10 are the similar case	
humi	display humidity(upload after multiplying by 10)	int	0-10v		
settemp	setting temperature(upload after multiplying by 10)	int	2 ways		
mode	mode(1 cool, 2 heat, 3 vent, 4 auto)	int	2 ways		
onoff	status (1 on, 2 off)	int	2 ways cool / electric heat		
frost	frost temperature(upload after multiplying by 10)	int	2 ways		
delay	output delay time	int	0-10v		
diff	switch difference(upload after multiplying by 10)	int	2 ways		
holdtime	hold temperature time	int	2 ways		
holdtemp	hold temperature(upload after multiplying by 10)	int	2 stage compressor 1 stageHeating		
kb	screen lock(1 no, 2 yes)	int			
kbkey	lock pin(four numbers)	String			
cf	temperature format(0 °C, 1 °F)	int			
holiday	holiday (0 no, 1 yes)	int			
standby	standby (1 no, 2 yes)	int			
fan	fan speed(1 high, 2 medium, 3 low, 4 auto)	int			
prog	schedule(0 none, 1 weekday/weekend, 2 7 days, 3 24 hours)	int			
opt	optimum start(1 on, 2 off)	int			
timezone	time zone	int			
version	version	int			
temp_prog	Program mode: t-the time in the period, 1440 means that the period is not used; s-setting temperature	String[]	Taking 2 7-day mode as the example, where the temperature is multiplied by 10, and the time unit is minute, such as “t”: 360, “s”: 160; it means that the setting temperature is 16 at 06:00 o’clock. In the 7-day mode, not all data are uploaded from Monday to Sunday, and the data of only one or several days may be uploaded, however, the data of all four time periods in the day must be uploaded. { “temp_prog”: { “sun”: [{ “t”: 280, “s”: 250 }, { “t”: 360, “s”: 160 }, { “t”: 420, “s”: 250 }, { “t”: 720, “s”: 220 }], “mon”: [{ “t”: 160, “s”: 250 }, { “t”: 850, “s”: 160 }, { “t”: 680, “s”: 250 }, { “t”: 1440, “s”: 220 }], “tues”: [{ “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 160 }, { “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 220 }], “wed”: [{ “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 160 }, { “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 220 }], “thur”: [{ “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 160 }, { “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 220 }], “fri”: [{ “t”: 480, “s”: 210 }, { “t”: 1440, “s”: 160 }, { “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 220 }], “sat”: [{ “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 160 }, { “t”: 1440, “s”: 250 }, { “t”: 1440, “s”: 220 }] } }	In the 1 weekday/weekend mode, not all data are uploaded on Saturday and Sunday, and it may upload one day data, however, the data of all four time periods in the day must be uploaded. { “temp_prog”: { “weekday”: [{ “t”: 1230, “s”: 820 }, { “t”: 1380, “s”: 630 }, { “t”: 480, “s”: 700 }, { “t”: 570, “s”: 610 }], “weekend”: [{ “t”: 990, “s”: 790 }, { “t”: 1380, “s”: 620 }, { “t”: 480, “s”: 700 }, { “t”: 570, “s”: 610 }] } }	Take 3 24 hours mode as the example. { “temp_prog”: { “24hrs”: [{ “t”: 480, “s”: 210 }, { “t”: 570, “s”: 160 }, { “t”: 990, “s”: 220 }, { “t”: 1380, “s”: 170 }] } }
timer_prog	Hot water mode: n-on time, 1440 means not used in this period; f-off time	String[]	Taking 2 7-day mode as the example, and the time unit is minute, for example, “n”: 360, “f”: 385; it means that the timer starts at 06:00 and ends at 06:25. In the 7-day mode, not all data are uploaded from Monday to Sunday, and the data of only one or several days may be uploaded, however, the data of all four time periods in the day must be uploaded. { “timer_prog”: { “sun”: [{ “n”: 420, “f”: 480 }, { “n”: 540, “f”: 560 }, { “n”: 1020, “f”: 1050 }, { “n”: 1320, “f”: 1400 }], “mon”: [{ “n”: 420, “f”: 480 }, { “n”: 540, “f”: 560 }, { “n”: 1020, “f”: 1040 }, { “n”: 1320, “f”: 1380 }] } }	In the 1 weekday/weekend mode, not all data are uploaded on Saturday and Sunday, and it may upload one day data, however, the data of all four time periods in the day must be uploaded. { “timer_prog”: { “weekday”: [{ “n”: 120, “f”: 125 }, { “n”: 0, “f”: 0 }, { “n”: 420, “f”: 540 }, { “n”: 960, “f”: 1200 }], “weekend”: [{ “n”: 425, “f”: 545 }, { “n”: 960, “f”: 1200 }, { “n”: 1440, “f”: 1440 }, { “n”: 1440, “f”: 1440 }] } }	Take 3 24 hours mode as the example. { “timer_prog”: { “24hrs”: [{ “n”: 480, “f”: 560 }, { “n”: 570, “f”: 580 }, { “n”: 990, “f”: 1020 }, { “n”: 1380, “f”: 1400 }] } }

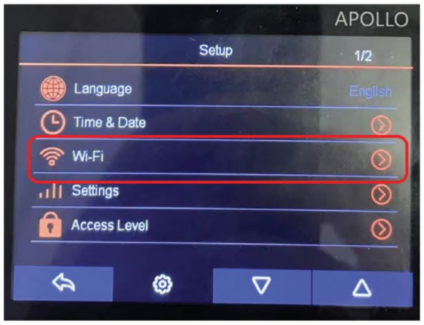
BACNET MS/TP INTEGRATION



Set BACnet parameters in MC6

(1) Home sreen->Configuration menu ->the APOLLO thermostat




APOLLO

Setup 1/2

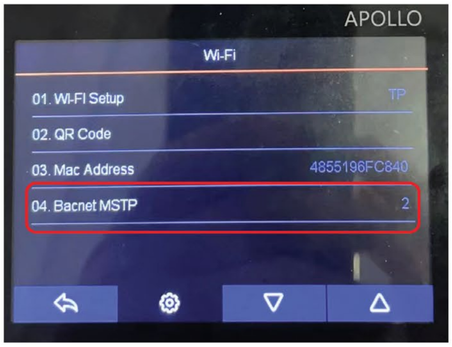
Language English

Time & Date

Wi-Fi

Settings

Access Level


APOLLO

Wi-Fi

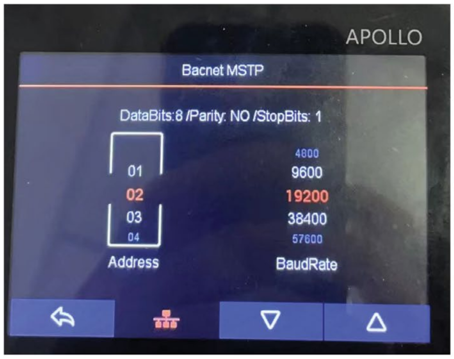
01. Wi-Fi Setup TP

02. QR Code

03. Mac Address 4855196FC840

04. Bacnet MSTP 2

(2) BACnet MSTP communication port default settings: **Data Bits: 8; Parity: No; Stop Bits: 1;**
Below three items could be changed on MC6 :
Address: 1 to 127 , 01(default);
Data Rate : 1200, 2400, 4800, 9600, 19200, 38400, 57600. 19200 (default)

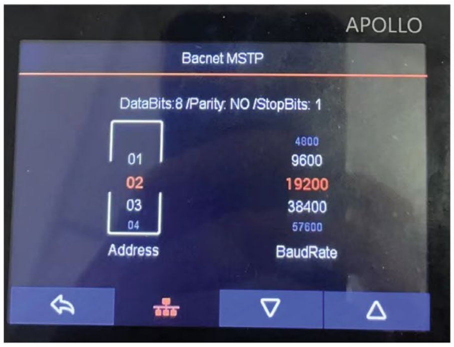

APOLLO

Bacnet MSTP

DataBits:8 /Parity: NO /StopBits: 1

Address: 01 / 02 / 03 / 04

BaudRate: 4800 / 9600 / 19200 / 38400 / 57600


APOLLO

Bacnet MSTP

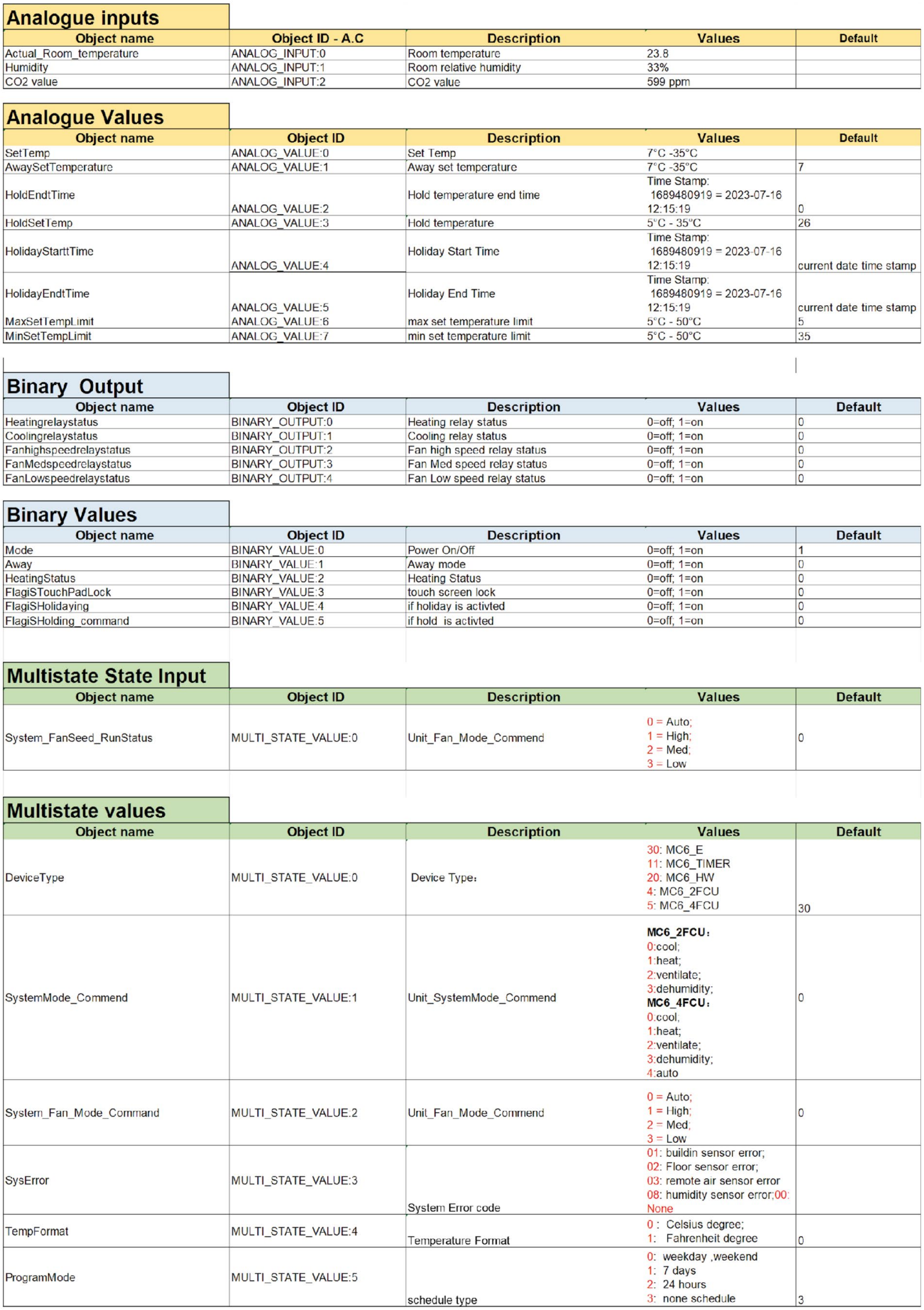
DataBits:8 /Parity: NO /StopBits: 1

Address: 01 / 02 / 03 / 04

BaudRate: 4800 / 9600 / 19200 / 38400 / 57600

BACNET MS/TP INTEGRATION

BACNET DATA FORMAT



Analogue inputs

Object name	Object ID - A.C	Description	Values	Default
Actual_Room_temperature	ANALOG_INPUT:0	Room temperature	23.8	
Humidity	ANALOG_INPUT:1	Room relative humidity	33%	
CO2 value	ANALOG_INPUT:2	CO2 value	599 ppm	

Analogue Values

Object name	Object ID	Description	Values	Default
SetTemp	ANALOG_VALUE:0	Set Temp	7°C -35°C	
AwaySetTemperature	ANALOG_VALUE:1	Away set temperature	7°C -35°C	7
HoldEndtTime	ANALOG_VALUE:2	Hold temperature end time	Time Stamp: 1689480919 = 2023-07-16 12:15:19	0
HoldSetTemp	ANALOG_VALUE:3	Hold temperature	5°C - 35°C	26
HolidayStarttTime	ANALOG_VALUE:4	Holiday Start Time	Time Stamp: 1689480919 = 2023-07-16 12:15:19	current date time stamp
HolidayEndtTime	ANALOG_VALUE:5	Holiday End Time	Time Stamp: 1689480919 = 2023-07-16 12:15:19	current date time stamp
MaxSetTempLimit	ANALOG_VALUE:6	max set temperature limit	5°C - 50°C	5
MinSetTempLimit	ANALOG_VALUE:7	min set temperature limit	5°C - 50°C	35

Binary Output

Object name	Object ID	Description	Values	Default
Heatingrelaystatus	BINARY_OUTPUT:0	Heating relay status	0=off; 1=on	0
Coolingrelaystatus	BINARY_OUTPUT:1	Cooling relay status	0=off; 1=on	0
Fanhighspeedrelaystatus	BINARY_OUTPUT:2	Fan high speed relay status	0=off; 1=on	0
FanMedspeedrelaystatus	BINARY_OUTPUT:3	Fan Med speed relay status	0=off; 1=on	0
FanLowspeedrelaystatus	BINARY_OUTPUT:4	Fan Low speed relay status	0=off; 1=on	0

Binary Values

Object name	Object ID	Description	Values	Default
Mode	BINARY_VALUE:0	Power On/Off	0=off; 1=on	1
Away	BINARY_VALUE:1	Away mode	0=off; 1=on	0
HeatingStatus	BINARY_VALUE:2	Heating Status	0=off; 1=on	0
FlagiSTouchPadLock	BINARY_VALUE:3	touch screen lock	0=off; 1=on	0
FlagiSHolidaying	BINARY_VALUE:4	if holiday is activted	0=off; 1=on	0
FlagiSHolding_command	BINARY_VALUE:5	if hold is activted	0=off, 1=on	0

Multistate State Input

Object name	Object ID	Description	Values	Default

System_FanSeed_RunStatus	MULTI_STATE_VALUE:0	Unit_Fan_Mode_Commend	0 = Auto;
			1 = High;
			2 = Med;
			3 = Low
				0

Multistate values

Object name	Object ID	Description	Values	Default
DeviceType	MULTI_STATE_VALUE:0	Device Type:	30: MC6_E 11: MC6_TIMER 20: MC6_HW 4: MC6_2FCU 5: MC6_4FCU	30

SystemMode_Commend	MULTI_STATE_VALUE:1	Unit_SystemMode_Commend	**MC6_2FCU:**
			0:cool;
			1:heat;
			2:ventilate;
			3:dehumidity;
			MC6_4FCU:
			0:cool,
			1:heat;
			2:ventilate;
			3:dehumidity;
			4:auto
				0

System_Fan_Mode_Command	MULTI_STATE_VALUE:2	Unit_Fan_Mode_Commend	0 = Auto;
			1 = High;
			2 = Med;
			3 = Low
				0

SysError	MULTI_STATE_VALUE:3	System Error code	01: buildin sensor error;
			02: Floor sensor error;
			03: remote air sensor error
			08: humidity sensor error;00: None	

TempFormat	MULTI_STATE_VALUE:4	Temperature Format	0 : Celsius degree; 1: Fahrenheit degree	0
ProgramMode	MULTI_STATE_VALUE:5	schedule type	0: weekday ,weekend 1: 7 days 2: 24 hours 3: none schedule	3

APOLLO

APOLLO Heat Pump
Apolloheatpumps.com
sales@apolloheatpumps.com
www.apolloheatpumps.com
888-301-0737

2411 N. American Street
Philadelphia. PA 19133, USA