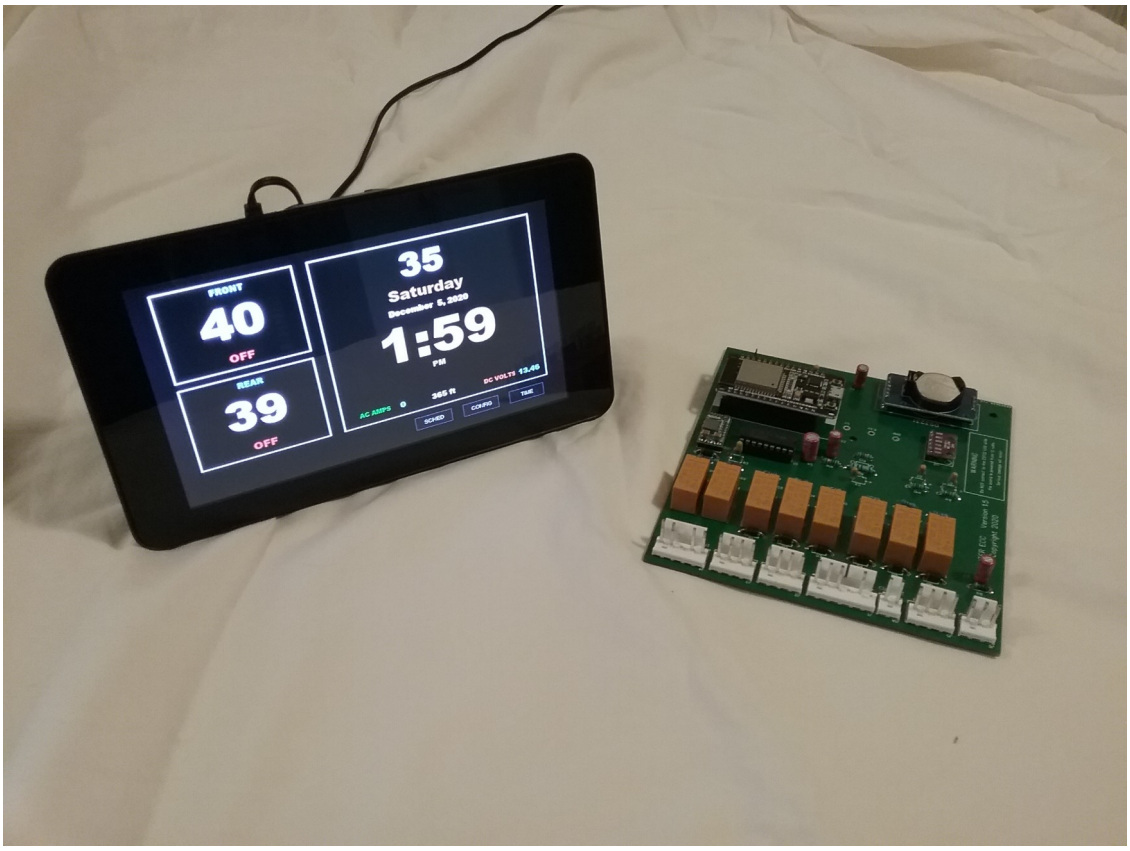
Waiter ECC / NWR

Electronic Climate Controller
(MQTT)

August 22, 2022



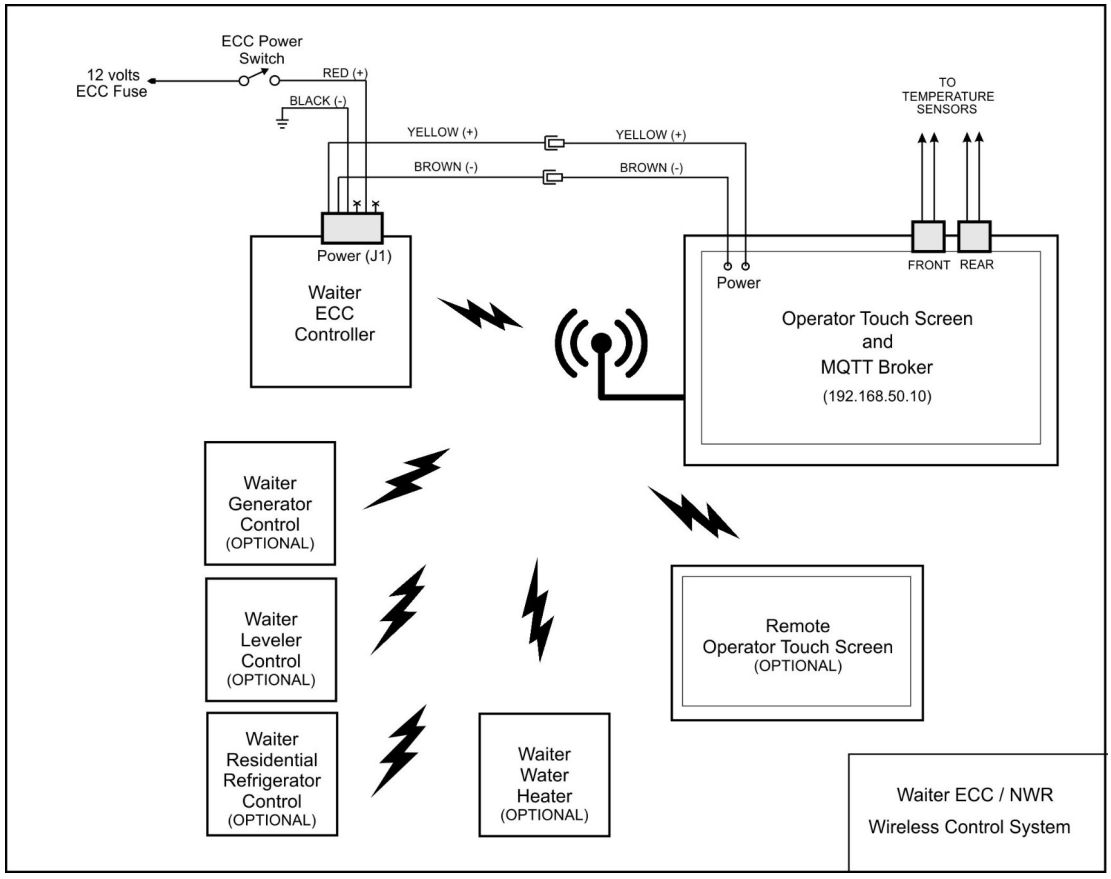
TECHNICAL DETAIL

www.WaiterECC.com

INTRODUCTION

The WaiterECC system provides similar functionality as the original Intellitec system, and then some. All logic and functionality is controlled within the Controller module. The Operator panel serves as an interface to the Controller module, it receives status from the module and sends commands to the module via MQTT messages.

The NWR stands for No Wiring Required. This was implemented in version 1.6 controller module and version 1.4 Rpi hat



VERSION INFORMATION

CONTROLLER CIRCUIT BOARD

ESP32:	MELIFE ESP-32S
Waiter ECC Circuit board	1.11
Waiter ECC Firmware:	2.10.4
Arduino IDE (ESP32):	1.8.12
Current draw:	0.04 amps (no relays energized)

RASPBERRY TOUCH SCREEN

Hardware:	Raspberry pi 3A, 3B, 3B+
Linux Rasbian:	10 (buster)
Mosquitto MQTT Broker:	3.1
Waiter ALM/PWR Circuit:	1.7 (hat)
Waiter ECC Software:	5.0.3 MQTT (rpi3)
Mono:	6.8.0.105
Visual Studio 2019:	16.5.4
(reference) Raspberry.IO	3.1.1.0
Current draw:	0.25 amps (booted, minimum brightness)
	0.45 amps (booted, maximum brightness)

CONTROLLER BOARD

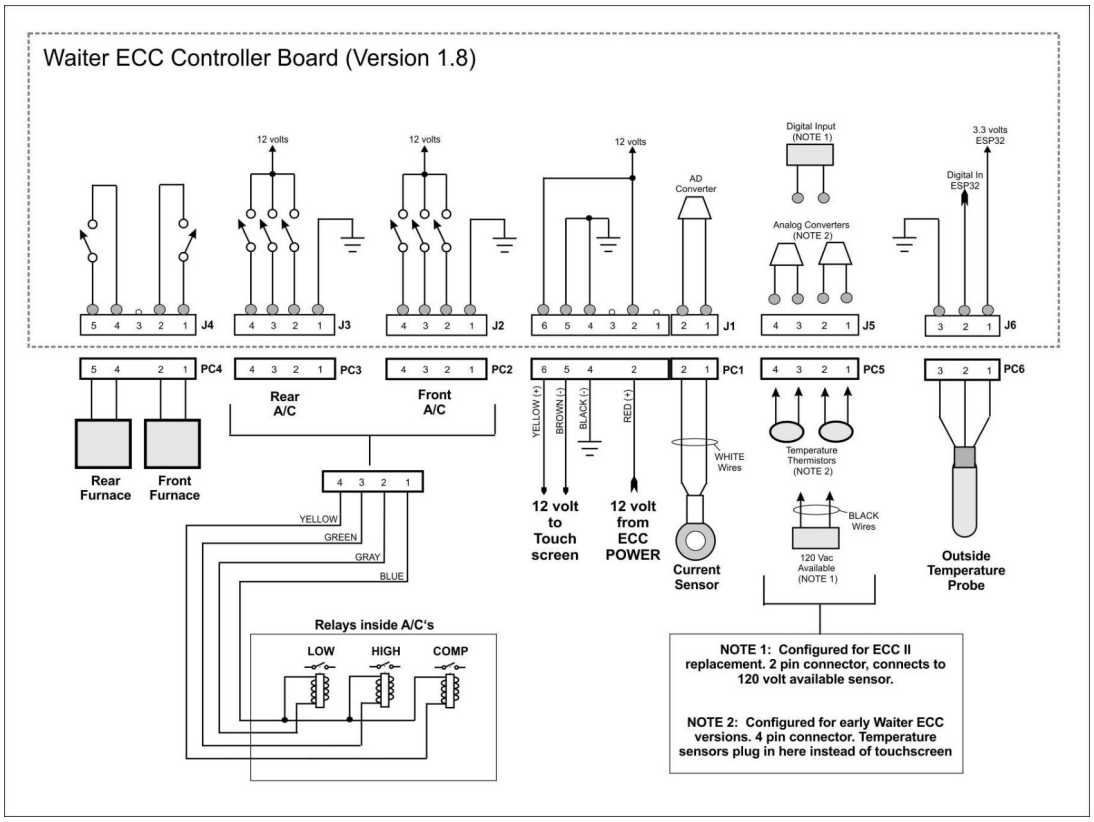
Version 1.11

The brains behind the controller is an ESP32 micro-controller module. The micro-controller and additional components are assembled on a custom circuit board. The new Waiter ECC circuit board (left) is designed to replace the original Intellitec circuit board (right). The new circuit board fits inside the existing Intellitec enclosure.

The controller board controls two furnaces and two A/C units. It monitors current flow (amps) into the 115 volt distribution panel and has the ability to SHED loads in order to prevent a system overload (30 amps). It also monitors the coach 12 volt system and reports the voltage to the display. The controller receives signals from the temperature and current sensors. It then uses this information to control 8 relays. The relay contacts in turn control the furnaces and A/C units.

The controller also contains a real time clock with battery backup for keeping track of date/time information and a pressure sensor for reporting pressure altitude.

The controller transmits / receives status messages via MQTT over a dedicated WiFi network. (see Appendix A).



Temperature sensors:

The controller uses three temperature sensors to calculate temperatures. All temperatures are reported in degrees Fahrenheit.

NOTE – The blue LED on the controller boards ESP32 module will illuminate if a temperature sensor isn't reading correctly. Before panicking, verify that the sensors are configured correctly on the CONFIG screen.

In a normal installation, the outside air temperature (OAT) will use a DS18B20 temperature probe and the two inside temper sensors will use the original Intellitec 10k thermistors. On the operators CONFIG screen this would be set as "1" for the OAT sensor, and "4" for the front and rear sensors if they're plugged into the back of the touchscreen, or "0" if they are plugged into the control module.

NOTE: When the "source" is set to #4, The temperature is supplied to the control module by an MQTT message. See note #9 in MQTT messages, Appendix A. In the normal Waiter ECC, the touchscreen has the necessary circuits to read the sensors and then supplies the temperatures to the control module via a MQTT message.

The DS18B20 is a very accurate electronic probe that sends a digital temperature reading directly to the controller. The sensor probe has a 16 ft long cable to allow it to be routed to the motor-home exterior. A good location for the probe would be on the underside, away from exhaust

pipes, open to the air, and out of direct sunlight. On the prototype install, the probe was placed in the propane tank compartment.

Very early model Intellitec ECCs used a 2.2k thermistor (control module part number 00-00329-XXX). These are not compatible with the Waiter ECC system.

Later model Intellitec system came equipped with two 10k thermistors to sense temperature. These are an analog device that changes resistance as the temperature changes. At 77 deg F the resistance of the device will be almost exactly 10k ohms. As the resistance changes, the analog to digital (AD) converters read the voltage across the thermistors and convert this to a temperature reading. This happens on either the control module (source set to “0”) or on the back of the touchscreen (source set to “4”)

A temperature to resistance chart is at the end of this manual, APPENDIX C.

The thermistor sensors are very accurate. However, any minor errors that may be introduced by cable length, wiring, or A/D circuit deviations can be corrected using a “Correction Factor” (-10 to +10) on the CONFIG screen of the operator panel. The Front and Rear correction factors are applied when the source is either “0” or “4”. There is no correction factor for the OAT when its source is “4”.

Note that if an installer wishes, the two thermistor sensors could be replaced with DS18B20 probes (will need to rewired). This change would then need to be entered into the CONFIG screen so the controller knows the address (1, 2, or 3) of the probe sensors to use for what particular location.

If using the DS18B20 probes for all three sensors, you’ll need to determine the correct index number for each probe; 1, 2, or 3. and assign it to the correct location; OAT, front, and rear.

When the DS18B20s are manufactured, each one has a unique serial number. The Waiter controller module’s firmware doesn’t know what serial number is assigned to what probe, so instead, it reads all the probes connected to it and assigns an “index” number to each probe it finds based on the probes serial number. i.e. the lowest serial number is #1, the next is #2 and the highest is #3.

To assign the correct index number to the correct probe location follow this procedure:

1) On the CONFIG screen, assign the OAT to #1, the front to #2, and the rear to #3.

2) Go to the front sensor and change its temperature by blowing on it. On the operator panel, see what the location is for the temperature that changed and note the current index number you assigned on the CONFIG screen.

Example, When you blew on the front sensor, but on the operator panel the OAT temperature was changing, then you know that the front sensors should really be assign as #1, not #2.

Repeat this for the rear sensor.

3) Reassign the index numbers to the correct locations and re-verify the index numbers are

reading correct for the location.

Voltage Sensor:

The controller board measures the boards supply voltage (12 volts) and reports this in an MQTT message topic. Its important to note that because of various wiring configurations, component tolerances, and A/D converter variances, the DC volts will need to be calibrated. This is done before the circuit board is shipped, but the end user can also perform a calibration using the CONFIG page on the touch screen.

The A/D converter in the ESP32 is a 12 bit converter providing 4095 bits of resolution. One of the documented shortcomings of the ESP32 A/D converters is the inconsistent conversion from unit to unit, the same signal applied to two different ESP32's will produce two slightly different A/D values.

A resistive voltage divider is configured for 0 to 20 volts, This produces a 0 to 3.33 volt signal to the A/D converter. Although the A/D converters are notoriously inaccurate because of a non-linear response, the response curve for the area we're measuring (8 – 15 volts) is fairly linear, but the curve is offset from a straight line.

Correction for this offset is made by adding a value. We've found that this presents a fairly accurate voltage reading from 8 to 15 volts.

The A/Ds volts per bit is set at 0.004884 = 20 / 4095. There are 30 steps of correction for the offset, each step adds 0.1 volts to the calculated value. This is normally enough to correct for the non-linear curve and also the protection diode in version 1.9 boards

NOTE – On version 1.9 and UP, the 12 volt test point goes through the reverse protection diode, so it will read approximately 0.6 volts lower than the 12 volt supply.

Prior to a controller board being shipped, part of the bench test is to calibrate the A/D converter so that when 13.42 volts is applied, it reads 13.42, plus / minus 0.05 volts

AS OF IO VERSION 2.10.4 Uses an ESP32 internal calibration called Vref. and outputs the A/D value across the 2k resistor directly as millivolts. The voltage divider (10k and 2k resistors) calculation is then made to normalize this to the 12 volt reading. The correction factor on the CONFIG page is still applied to correct for wiring differences and also a steering diod that may not be present prior to version 1.9 board.

Current Sensor:

The controller board uses the OEM current sensor. This is basically an AC transformer that feeds a small AC voltage to one of the A/D converters on the ESP32 through a 10 ohm burden resistor. As more current flows, a higher voltage is dropped across the burden resistor. The ESP32 AD converter reads the voltage and calculates a corresponding current. This current value (amps) is

then used by the controller and also displayed to the operator.

Each board is tested using a 1500 watt electric heater. The test nominally indicates 11-12 amps

Atmospherics Sensor:

A BMP280 Atmospheric sensor has been added in controller circuit board Version 1.5. This sensor reports the sensors atmospheric pressure, altitude, and temperature.

Altitude and temperature are converted and reported in feet and degrees Fahrenheit. The pressure is reported in metric units, hPa. These are reported in the MQTT packets. See the MQTT in Appendix A

NOTE: The BMP280 was disabled in firmware version 2.3.0 because of a possible issue with its driver causing the WiFi to lose connection with the touchscreen. We're looking into this and will re-enable the BMP280 when we're confident we have a solution to this problem.

RE-ENABLED as of IO Board 1.11

Real Time Clock:

Normally, a WiFi connection to the internet could provide time and clock functions. However, since an internet connection cannot be counted on, a battery backup Real Time Clock (RTC) module (DS3231) is mounted on the control module to provide time / date data. The RTC has a battery backup (CR2032) with a life of 5 -6 years. The RTC doesn't maintain time zones or daylight savings time. If these change during your travel, simply program the correct date and time from the touch screen controller.

SSID Selection:

By using jumper DI-5, the control circuit board can be configured to connect to either the "WaiterControl" SSID (default - unjumpered) or the "WaiterControl2" SSID (jumpered). Circuit board 1.8 or higher and firmware version 2.6.0 or higher are required.

SSID selection on the Rpi touchscreen was removed as of 5.0.0. If the SSID is to be changed, it must be ordered from Waiter ECC or it can be changed using the PC version of the Waiter Control. (SD card must be removed from the Rpi and placed in the PC where its changed)

The SSID on the Rpi touchscreen MUST match the jumper settings on the control circuit card or the two cannot talk to each onther.

WiFi Watchdog:

The Waiter ECC WiFi access point is very robust and has performed trouble free since being implemented. However, as with any WiFi connection, outside forces could interfere with it.

The WaiterControl WiFi access point MUST be up and operational 100% of the time. A

watchdog routine is written into the screen software to monitor the WiFi connection. In the unlikely event the connection is lost, the watchdog may reboot the touchscreen if the watchdog has been enabled by the certain criteria.

The watchdog is enabled only after a WiFi connection has been established for at least 30 consecutive minutes. After that, if the connection goes down for over 10 consecutive minutes, the watchdog will do one of three things: 1) a channel change and reboot (CHANGE CHAN), 2) a reboot but stay on the same channel (ON), or 3) do nothing (OFF). The default is CHANGE CHAN.

Changes to the SSID, channel and mode are performed on the CONFIG > WiFi screen. HOWEVER, to reduce the possibility of accidentally making changes, changes on this screen can only be done within the first 120 seconds of a reboot. After that no changes to the watchdog or any other options on the screen can be made.

There are three Watchdog modes; CHANGE CHAN, ON, and OFF. The status of the internal MQTT broker and the control module is shown with two small indicators on the WiFi / SSID CONFIG screen.

MQTT (left indicator) monitoring is performed by subscribing to the “SET_Temp_FR” message. This message originates from a Python script that's reading the temperature A/D converter on the touchscreen. The script sends the MQTT message directly to the MQTT broker, then the MQTT broker sends the message to the touchscreen software. The entire path for this message is within the touchscreen so it doesn’t rely on any WiFi or other outside equipment. If the touchscreen is receiving the message, then this is a very good indicator that the MQTT broker is functional.

ECC Communications (right indicator). The status of the control module is derived by monitoring two MQTT messages that originate from the control module, the “ECC/Watchdog” and the “ECC/IP” messages. If the touchscreen is receiving these messages at least once every 90 seconds, then its assumed the WiFi connection is good. If the watchdog fails to see these messages, then its assumes the WiFi connection is bad and if enabled, the watchdog will reboot the touchscreen.

Software:

Controller board (ESP32) software is written in C++ and authored on the Arduino IDE platform. The authoring computer is a Windows 10 computer with the Arduino IDE platform and the appropriate ESP32 libraries installed. Updates and changes to the ESP32 are made via the Arduino IDE with a USB serial connection between the ESP32 and the computer.

WARNING

Do NOT apply USB power to the ESP32 module while the 12 volt power is applied. Serious damage to the ESP32 module will result.

Currently, any updates to an ESP32 controller firmware would need to be accomplished using the Arduino IDE and a USB cable connected to the ESP32. There are no PUSH updates or updates available via the internet.

Control Board WATCHDOG:

The control board uses the ESP32's implementation of an "Interrupt Watchdog". The watchdog is reset every time the MQTT message "SET_Temp_FR" is received. If this message isn't received in a 5 minute period (300 seconds) the Watchdog initiates a ESP32 reboot.

The SET_Temp_FR message is sent from the Rpi touchscreen at least once every 75 seconds.

IO TEST MODE:

The IO) TEST jumper has two functions, 1) Allows you to perform a factory reset of all parameters, and 2) places the control module in a continuous self test loop.

1) Restoring factory defaults To restore all parameters to their factory defaults, momentarily short the test pins to each other with coin or other metal object, At the same time, apply power to the control module board. Remove the short jumper.

The SIG RX and ESP32 LEDS will illuminate for 2 seconds, then blink 5 times to indicate a reset is in progress.

The factory defaults have been saved and the board will reboot.

2) To enter self test mode, momentarily short the two test pins together. Read the cautions below.

CAUTION

When the controller board is placed in IO TEST mode, each of the 8 relays is energized in sequence for one second. If the A/C units are plugged into the controller board during this test, the blowers and compressor will attempt to start. Its recommended to NOT allow the compressors to start more than once or twice while in test mode, this could cause an overload.

Either unplug the A/C units from the control module, turn the A/C circuit breakers OFF, or exit the IO TEST feature.

There are two ways to enter the IO TEST mode:

1) Momentarily short across the IO TEST pins on the controller circuit board. If your controller board has a IO TEST push button, then simply press this button.

2) Send the following topic/payload to the MQTT broker

mosquitto_pub -h localhost -t WaiterECC/ECC/SET_TEST_IO -m "1"

There are three ways to exit the IO TEST mode, Click on the FRONT or REAR mode OFF buttons on the operators touchscreen, cycle power OFF then back ON, send the MQTT command to exit the IO TEST mode.

Send the following topic/payload to the MQTT broker:

mosquitto_pub -h localhost -t WaiterECC/ECC/SET_TEST_IO -m "0"

RASPBERRY MICRO COMPUTER

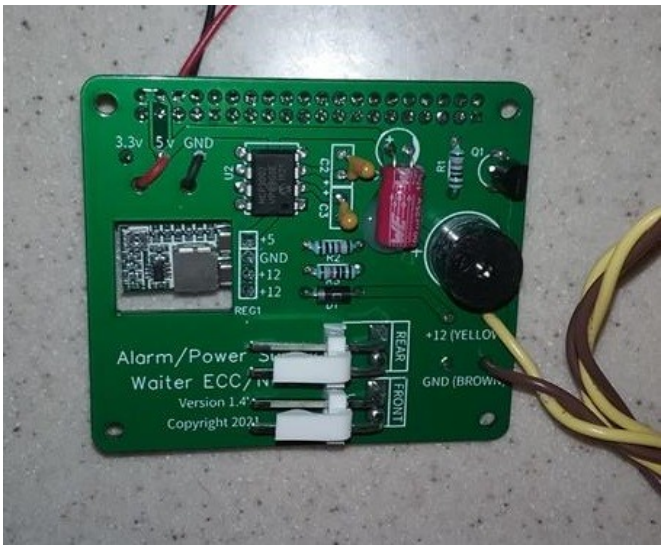
The Operator Panel (OP) is built around a standard off the shelf Raspberry pi 3 microcomputer that's attached to an official Raspberry 7 inch touch screen, running the Linux operating system.

There are three different versions of the Rpi 3 that are used, 3A, 3B and 3B+.

The 3A is the preferred model because of lower power consumption, but the selection is based more on whats available from our component distributors.

The 3A doesn't have the multi USB hubs or an RJ45 Ethernet connection, hence the lower power consumption. Other than that its functionally identical to the 3B and 3B+.

An additional custom circuit card (hat) is added to the Rpi (see below). This custom circuit board contains a 12 volt to 5 volt power supply that powers the Rpi and touchscreen, Analog to Digital converters for the two temperature sensors, and the alarm clock buzzer and supporting driver circuits.



The Rpi touch screen serves four purposes:

1. WiFi access point for Waiter devices (SSID = WaiterControl or WaiterControl2).
2. MQTT broker for status messages to/from the controller and the controlled devices.
3. The Operator control panel.
4. The Analog to Digital converters for the two temperature sensors.

<u>WiFi Access Point</u>

The Raspberry is set up to operate as a WiFi access point. This access point doesn't currently support external internet connections, its dedicated solely to the Waiter control system. Waiter control devices will automatically search for and connect to this SSID.

SSID = WaiterControl or WaiterControl2
Password = ECC123456
Channel 6 or 11
IP = 192.168.50.10
DHCP = for control modules

Using Remote Desktop Connection (RDC) and WinSCP from a laptop

Connect to the WaiterControl WiFi access point. As described above.

The Username / password for the RDC and WinSCP will be

USER > pi
Password > waiterecc

OFFICIAL TOUCHSCREEN

We use the Rpi Official touchscreen and case bezel. These have been proven to be very robust and they look good.

On request, there is another supplier for a bezel, VonLinder. We recommend the VonLinder bezel for our REMOTE touchscreen display

NOTE – Raspberry Pi Foundation added an additional vendor for their Official Touchscreen. The screen from this vendor was slightly thicker and we noted issues with the glass pulling away from the frame when the touchscreen was installed in the official case. Raspberry Pi Foundation has been notified of this issue and is working to resolve it.

In the meantime to mitigate this issue and be able to continue production. When we're forced to use touchscreens from this vendor, we install a small spacer washer between the case and the frame, and place stickers on the inside of the case to advise anyone that the washers must be in place. The spacer washers reposition the frame so as not to pull the glass away from the frame.

ANALOG to DIGITAL CONVERTER

A two channel A/D converter chip (MCP-3002) reads the two 10k thermistor temperature sensors. A Python script runs continuously reading the two channels, converts the A/D values to degrees Fahrenheit, and publishes the two readings to the on-board MQTT server.

The Python script reads the A/D converter once a second. If the temperature changed its reported immediately via MQTT message. If there hasn't been a change, the temperature gets reported every 90 seconds, regardless if its changed or not. If there is an error reading the temperature, i.e. open sensor, it gets reported as a -99.

The two temperatures are then available to the control module via a MQTT message if the temperature source is set to "4" on the CONFIG screen.

ALSO – The MQTT message for the Front temperature "SET_Temp_FR" is sent to the software that runs the operators panel. It uses this message as a watchdog to indicate the status of the MQTT broker. If it receives the message at least once every 90 seconds, then it marks the MQTT broker status as OK.

MQTT Broker

An MQTT broker is installed and running (Mosquito) on the Raspberry The MQTT broker acts like a switchboard for incoming messages. It receives messages from all devices connected to it via WiFi, it then sends the messages to the devices that have subscribed to receive them.

Multiple devices are currently planned / supported to communicate with the MQTT broker. All of these devices will utilize the MQTT topics with the prefix "WaiterECC". Example, Topics to/from the Controller module are identified with "ECC" plus the parameter name (topic).

Example: “WaiterECC/ECC/SET_FR_MODE”

MQTT topic assignments for “WaiterECC” topics:

- Operator control screen (main) – runs on the same Raspberry as the MQTT broker.
- Operator control screen (remote) – runs on remote Raspberry.
- ECC - controller module for heat / A/C control.
- REFRIG – Waiter Refrig – controls residential refrigerator.
- GEN – Control Generator start / stop / quite times, etc.
- LEVEL – Control the Power Gear leveler system.
- WTR – Control temperature in a water heater

Operator Control Panel:

The WaiterECC control screen software is a standard Windows Form application written in C# on a windows 10 computer running Visual Studio 2019 IDE. The compiled applications EXE file and supporting DLL files are then loaded onto the Raspberry using WinSCP.

The windows form application file (the .exe file) is then run under a program called “MONO”. When the Raspberry is re-booted, it runs a script file that basically configures the touch screen, turns off the mouse cursor, and then starts MONO running the WaiterECC_MQTT.exe application.

The Windows exe application could be run on any computer. However, in order to run properly on the Raspberry with the programmed functionality, these four items are peculiar to the Raspberry pi. These are configured within the .exe program before its compiled to run on the raspberry.

1) Controlling the Raspberry display brightness. This is done using the linux file system as the interface between our windows application and the linux operating system. Our application creates a file and writes it to the Linux file system. Linux reads the file and then sends commands to the display to turn brightness up/down.

This is the procedure and location of how our .exe writes to the linux file system:

StreamWriter sw = new StreamWriter("/sys/class/backlight/rpi_backlight/brightness")

2) In a similar fashion, Our windows application uses the file system to interface to the Raspberries GPIO pins to turn the alarm clock buzzer on and off. This is the DLL that we need to include as a “resource” in our IDE. And need to place the DLLs in the same linux folder as our application.

Utilize Raspberry.IO.GeneralPurpose as a resource.

3) Our program reads and writes a file CONFIG.XML to the linux file system. This is configuration information that's used by our program. The file path is peculiar to the Raspberry.

On the Raspberry, the file will be located in the root folder. On a Windows machine the file is located in the same folder as the .exe file.

4) EXE and DLL locations

The EXE and dlls are all located in the program folder, /home/pi/WaiterECC:

WaiterECC_MQTT.exe
M2Mqtt.Net.dll
Raspberry.IO.dll
Raspberry.IO.GeneralPurpose.dll
Raspberry.IO.Interop.dll
Raspberry.System.dll

MQTT testing

Incoming MQTT messages can be seen, and messages can be sent by using the Mosquito's built in test feature. Use Remote Deskptop connection and log into the Raspberry. Open two command line windows. Use one window to subscribe to MQTT messages, and the other window to send MQTT messages.

To subscribe to all messages coming from the ECC:

mosquitto_sub -h localhost -t "WaiterECC/ECC/#" -v

This will show all messages that the broker receives from the ECC controller.

To send a message to the broker:

mosquitto_pub -h localhost -t WaiterECC/ECC/SET_FR_MODE -m "1"

This is used to send messages to the Broker. Note the topic isn't quoted (WaiterECC/ECC/SET_FR_MODE) , but the payload is ("1").

See ECC MESSAGES for a complete list of topics / payloads that are sent and received by the control module

MQTT TESTING

Use the Android Application "MyMQTT" to monitor / interact with the MQTT Server.

The MQTT Server will be on the WaiterControl SSID WiFi network.

In the MyMQTT, set the URL to 192.168.50.10.

“Subscribe” to “WaiterECC/ECC/#” to receive all topics.
To “Publish”, enter the entire topic, the messabe, then click “Publish”.

Example: To turn the IO test ON / OFF. The topic will be “WaiterECC/ECC/SET_TEST_IO”.
The message will be “1” to turn the IO test ON, and the message will be “0” to turn the IO Test OFF.

UPDATING Display and controller software

Over time, users submit ideas and suggestions for features and enhancements they'd like to see in the system. As these features are added, existing users may be interested in updating their systems to accommodate these new features.

Touchscreen display software and controller firmware can be upgraded locally by a technically savvy user. For those who aren’t quite as technical, we offer an exchange service for the microSD chip for the touchscreen and a ESP32 controller for the control board.

Contact us for access to the update files and procedures.

APPENDIX A

ECC MQTT MESSAGES (Furnace / A/C unit controller)

The following pages describe the MQTT messages exchanged between the operator control panel and the controller module. Any device that is capable of sending and receiving these MQTT messages could conceivably control the controller module.

All MQTT messages are two parts TOPIC and PAYLOAD

With the exception of “Version”, “VerDate”, and “IP” all WaiterECC payloads will be an integer value (a number)

MQTT messages to/from the ECC control module will begin with “WaiterECC/ECC/”

Example: To set the front system to OFF we’d send the following topic and payload to the control module:

WaiterECC/ECC/SET_FR_MODE, 0

When the control module received this message, it sets the front mode to OFF. If this was a change (i.e. it was on HEAT) the control module will send a response to show the new front mode setting:

WaiterECC/ECC/FR_MODE, 0

Below is a list of MQTT topics that the control module sends to the broker, and also a list of MQTT messages that the control module responds to:

Using MyMQTT for testing

One method we used in testing is to use an application called MyMQTT on a smart phone.

Connect to the WaiterControl SSID. In MyMQTT, connect to the MQTT broker at URL 192.168.50.10.

Subscribe to WaiterECC/ECC/+. This will show all topics to/from the ECC. You’ll see these messages under the Dashboard.

To send a message, use Publish. If you Publish the message that we used in the example, you’ll see the mode get changed.

ECC Controller module PUBLISHs these to MQTT Broker WaiterECC/ECC/

NOTE	TOPIC	PAYLOAD
1	FR_MODE	current front MODE (0 - 7)
1	RR_MODE	current rear MODE (0 - 7)
	HEAT_FR_SP	Front heater set point (40 - 90)
	HEAT_RR_SP	Rear heater set point (40 - 90)
	AC_FR_SP	front AC set point (60 - 100)
	AC_RR_SP	rear AC set point (60 - 100)
	FR_COR	Front thermistor correction (-10 to 10)
	RR_COR	Rear thermistor correction (-10 to 10)
	FR_AC_TEST	1 = A/C test switch detected in test position
	RR_AC_TEST	1 = A/C test switch detected in test position
7	VOLTS_COR	Voltage correction (-15 to +15)
2	Temp_OAT_Source	Source for OAT (0,1,2,3,4)
2	Temp_FR_Source	Source for FR (0,1,2,3,4,99)
2	Temp_RR_Source	Source for RR (0,1,2,3,4,99)
3	FR_ACUNIT	BTU for Front AC unit (0, 9, 11, 13, 15, 130, 150)
3	RR_ACUNIT	BTU for Rear AC unit (0, 9, 11, 13, 15, 130, 150)
	NUM_FUR	Number furnaces installed (1, 2)
5	T_MONTH	Current RTC time (1 - 12)
5	T_DAY	Current RTC time (1 - 31)
5	T_YEAR	Current RTC time (2020 - 2040)
5	T_HOUR	Current RTC time (0 - 23)
5	T_MIN	Current RTC time (0 - 59)
5	T_DOW	Current RTC time (0 - 6)
	Version	ESP32 version #
	VerDate	ESP32 version date
	Watchdog	minutes since last reboot (0 – 562,600 one year)
	Volts_12_Supply	12 volt supply (0.00 > 20.00)
	Temp_OAT	current outside temperature (deg F)
	Temp_FR	current front temperature (deg F)
	Temp_RR	current rear temperature (deg F)
	ShoreAmps	current shore amps
10	AMPS_AVAILABLE	Amps avail for SHED 0, 1, 2, 3, 4 (see note 10)
11	SHED_PRIORITY	Who gets SHED first (0, 1, 2) (see note 11)
12	SHED_TIME_LIMIT	Shed limit time in seconds 0 - 3600 (see note 12)
	FR_Furnace	Status - item OFF (0) or ON (1)
	FR_Compressor	Status - item OFF (0) or ON (1)
	FR_FanHI	Status - item OFF (0) or ON (1)
	FR_FanLO	Status - item OFF (0) or ON (1)
	FR_HardStartWait	Status - item OFF (0) or ON (1)
	FR_SHED	Status - item OFF (0) or ON (1)
	FR_PRESHED	Status - item OFF (0) or ON (1)
	FR_HP_AVAIL	Status - item NO (0) or YES (1)
	FR_HP_VALVE	Status - item OFF (0) or ON (1)
	RR_Furnace	Status - item OFF (0) or ON (1)
	RR_Compressor	Status - item OFF (0) or ON (1)
	RR_FanHI	Status - item OFF (0) or ON (1)
	RR_FanLO	Status - item OFF (0) or ON (1)
	RR_HardStartWait	Status - item OFF (0) or ON (1)
	RR_SHED	Status - item OFF (0) or ON (1)
	RR_PRESHED	Status - item OFF (0) or ON (1)

RR_HP_AVAIL	Status - item NO (0) or YES (1)
RR_HP_VALVE	Status - item OFF (0) or ON (1)
280_ALTITUDE	Altitude in feet (DISABLED)
280_TEMPERATURE	Temperature of 280 sensor (deg F) (DISABLED)
280_PRESSURE	Pressure in hPA (xxxx.xx) (DISABLED)
IP	xxx.xxx.xxx.xxx IP address
DATASAVED	1 = Data has been written to EEprom
SOUND_ALARM	1 = pulse a remote buzzer / alarm
SET_MAIN_CONFIG	Payload is comma delimited format of the configuration date. The first character defines who's configuration it is: M = Main screen R = Remote screen P = PC or laptop screen
GET_MAIN_CONFIG	Retrieve the Correct config file from the SPIFFS memory and send it out via MQTT.Payload indicates whos file should be retrieved and sent: M = Main screen R = Remote screen P = PC or laptop screen

Heat Pump Controller module PUBLISHs these to MQTT Broker WaiterECC/FR_HP/ or WaiterECC/RR_HP/

NOTE	TOPIC	PAYLOAD
	FR_REVERSE_VALVE	0 = OFF, 1 = ON
	RR_REVERSE_VALVE	0 = OFF, 1 = ON

NOTE – The main Control module must receive a reverse valve status at least once every 60 seconds, this acts as a watchdog. If the status isn't received, the FR or RR system is marked as not having a heat pump available.

ECC TOUCHSCREEN PUBLISHs these to MQTT broker WaiterECC/ECC/

NOTE	TOPIC	PAYLOAD
1	SET_FR_MODE	Set Front MODE (0 to 7)
1	SET_RR_MODE	Set Rear MODE (0 to 7)
4	SET_HEAT_FR_SP	Set/change Front heater SP (-1 to 90)
4	SET_HEAT_RR_SP	Set/change Rear heater SP (-1 to 90)
4	SET_AC_FR_SP	Set/change Front AC SP (-1 to 100)
4	SET_AC_RR_SP	Set/change Rear AC SP (-1 to 100)
	SET_FR_COR	increment source (1 = increment)
	SET_RR_COR	increment source (1 = increment)
7	SET_VOLTS_COR	increment source (1 = increment)
2	SET_Temp_OAT_Source	increment source (1 = increment)
2	SET_Temp_FR_Source	increment source (1 = increment)
2	SET_Temp_RR_Source	increment source (1 = increment)
9	SET_Temp_OAT	Temperature from external source (deg F)
9	SET_Temp_FR	Temperature from external source (deg F)
9	SET_Temp_RR	Temperature from external source (deg F)
3	SET_FR_ACUNIT	increment source (1 = increment)
3	SET_RR_ACUNIT	increment source (1 = increment)
	SET_NUM_FUR	increment source (1 = increment)
5	SET_MONTH	Set time (1 - 12)
5	SET_DAY	Set time (1 - 31)
5	SET_YEAR	Set time (2020 - 2040)
5	SET_HOUR	Set time (0 - 23)
5	SET_MIN	Set time (0 - 59)
5	SET_DOW	Set time (0 - 6)
5	SET_CLOCK	Set Time = 5 (sets RTC to the values supplied)
6	SET_TEST_IO	initiates IO output test, 1 starts test, 0 aborts
10	SET_SHORE_CAP	increment source (1 = increment)
11	SET_SHED_PRIORITY	increment source (1 = increment)
12	SET_SHED_TIME_LIMIT	increment source (1 = increment)
	SET_SENDALL	Resend all of MQTT topics and data (1)
	SET_SAVENOW	1 = instructs control module to perform a save.
	SET_DEFAULT	Restores configurations to factory defaults, reboots
8	SET_MAN_FR_FURNACE	Set IO control (0, 1, 2)
8	SET_MAN_RR_FURNACE	Set IO control (0, 1, 2)
8	SET_MAN_FR_COMPRESSOR	Set IO control (0, 1, 2)
8	SET_MAN_RR_COMPRESSOR	Set IO control (0, 1, 2)
8	SET_MAN_FR_FANHI	Set IO control (0, 1, 2)
8	SET_MAN_RR_FANHI	Set IO control (0, 1, 2)
8	SET_MAN_FR_FANLO	Set IO control (0, 1, 2)
8	SET_MAN_RR_FANLO	Set IO control (0, 1, 2)
	WaiterECC/OPR/RUNTIME	(from main touchscreen)
	WaiterECC/OPR/IP	(from main touchscreen)
	WaiterECC/RMT/RUNTIME	(from Remote touchscreen)
	WaiterECC/RMT/IPTIME	(from Remote touchscreen)
13	SET_MAIN_CONFIG	Payload is comma delimited MQTT text. The first character defines who's configuration it is:
13	GET_MAIN_CONFIG	Retrieve the Correct config file (M, R, P)

NOTE 1 - MODE indicates / sets the mode of operation:

0 = OFF
1 = HEAT
2 = FAN HIGH
3 = FAN LOW
4 = A/C FAN HIGH
5 = A/C FAN LOW
6 = A/C FAN automatically shifts speed
7 = A/C and HEAT full automatic mode

NOTE 2 - Temperature sensor source:

The original Intellitec system used two 10k ohm thermistors, one for the front and one for the rear. The Waiter ECC can use the original thermistors for front and rear, and also comes with a third electronic sensor that can be used as an outside temperature sensor. The OAT sensor is a very accurate digital sensor that can be used during initial installation to determine calibration factors for the original thermistors. As an option, instead of using the original thermistors for the front and rear, Digital sensors can be installed and configured for the Front and Rear temperatures. When more than one digital sensor is installed, each sensor has an internal address, i.e. 1,2, and 3.. Each display (OAT, FRONT, REAR) is assigned an address as to which sensor to use.

The Temperature Sensor Source settings determine what sensors are used, and/or where the controller should get its sensor temperature from.

The default settings are 1,4,4:

Temp_OAT_Source (DEFAULT = 1)
Temp_FR_Source   (DEFAULT = 4)
Temp_RR_Source   (DEFAULT = 4)

0 = Use OEM thermistor plugged into the controller board.

1, 2, 3 = Electronic sensor address.

4 = Sensor temperature will be sent to the controller via MQTT message. I.e. OEM thermistors are plugged into back of touchscreen. Touchscreen then sends MQTT message to the control module telling it what the temperatures are.

99 = Faulty sensor, for FRONT and REAR only. Use the temperature from the good sensor to control both FRONT and REAR. i.e. If the FRONT sensor is bad, set the FRONT TEMP SOURCE to 99, The FRONT system will now be controlled by the REAR sensor.

NOTE 3 - A/C btu capacity.

Sets the default current draw (amps) of a compressor. This is used to pre-test to see if turning on a compressor will cause a load shed and subsequent hard start delay. The compressor amps is added to the present shore amps, if this would exceed the shore capacity, then the compressor is not turned on and is marked as Pre-SHED, HOWEVER, since the compressor was not started, the hard start delay isn’t initiated.

This does NOT impact the normal shore amps SHED monitoring function. If a compressor is started and running, and the shore amps exceeds the capacity, the compressor is shut down, marked as SHED, and the hard start timer initiated. The compressor cannot restart until the 2 minute timer counts down. This allows compressor pressures to bleed down so the compressor isn't locked up when starting.

```
15  15k btu = 11 amps
13  13k btu = 9 amps   (Default for front and rear
11  11k btu = 6 amps
9   9k btu = 4 amps
0   0k btu = 0 amps (no pre-test performed)
```

NOTE 4: SET POINTS

If the set point message payload is a 1 or a -1, then the set point is incremented or de-incriminted by 1.

If the HEAT payload message is >= 40 and <= 90, the heater set point is set to that value.

If the A/C payload message is >=60 and <=100, the A/C set point is set to that value.

The controller limits the set points: Heater 40 - 90, A/C 60 - 100

NOTE 5: Time

The controller board comes with a battery backup Real Time Clock (RTC) module. Its function is to provide date/time functions to any other Waiter Control interfaces, including the operator panel touch screen.

To set / change the date time, the values must first be sent to the controller using the "SET_" topics and payload. Once the date/time values have been sent, the new values are loaded into the RTC by sending the SET_CLOCK:3.

Note that the Day of Week (DOW) is deprecated for an older time date function. The current RTC module computes the DOW based on date values.

NOTE 6: TEST_IO

A "1" enters the IO test mode, Each of the eight IO control relays will be energized for one second. To stop the test, either send a "0" with the topic, or send a topic and payload to set the front or rear system to OFF (see note 1).

The sequence will be:

REAR FURNACE
FRONT FURNACE
REAR COMPRESSOR
REAR FAN HIGH

REAR FAN LOW
FRONT COMPRESSOR
FRONT FAN HIGH
FRONT FAN LOW
Send MQTT message (SOUND_ALARM) to pulse the alarm chirp

NOTE 7: Volts Correction

To correct for minor deviations in the A/D converter and associated voltage divider network. The Volts correction factor makes small changes to the A/D scaling factor.

The A/D converter is supplied a 0-3.3 volt signal from a 0-20 volt voltage divider. The A/D converter is 12 bits (4095).

BEFORE 2.7.0

SET_VOLTS_COR = 1 increments the Volts Correction by one, between -15 to +15. When the Volts Correction reaches +15, the next step will cause it to revert back to -15. The Volts Correction is multiplied by 0.000025 and added to the A/D converters scale factor of 0.000505. As can be seen, a minus Volts Correction (i.e. -15) will make the A/D scaling factor smaller 0.0004675, and a Larger Volts Correction (plus 15) will increase the scaling factor (0.0005425.

The raw A/D value is then multiplied by the scaling factor to determine the voltage. VOLTS_COR = (-15 to +15) The Control Module reports what its current Volts Correction value is (-15 to + 15)

2.7.0 and up

The ESP32 A/D converter response curve isn’t linear and is know to have issues. The previous algorithm addressed a non-linear response.

However, we’ve found that the area of the curve that we use to measure 12 volts (8-15volts) is reasonably linear but has an offset that varies from chip to chip.

We multiply the output by 0.004884 then add an offset correction 0 – 3.0 in 0.1 increments. This seems to work better than the original algorithm we used.

NOTE 8: IO control

The 8 IO relays are normally in an AUTO state and controlled by temperature and features. There is a TEST IO feature (see note 6) that cycles through each control relay, turning them ON and OFF for approximately one second.

Another way to control the IO relays is using MQTT command that tells each individual relay how to respond, AUTO, ON, or OFF. These commands can be used to test the IO or even control the IO relays from an external program like MyMQTT or NodeRED. There are three possible payloads that would be included with the topic, 0, 1, or 2.

0 = AUTO. The IO relay is controlled by the controllers firmware, temperature, mode, and features. This would be the normal setting.

1 = ON. The IO relay is turned ON,

2 = OFF. The IO relay is turned OFF.

Resetting the IO relays back to AUTO can be done in two ways, Individually with the topic and a payload of “0”, or by topic and payload to set the front or rear system to OFF (see note 1).

NOTE 9: External Temperature source

Also see note 2. Temperature (-99 to +185) supplied by external source i.e. Raspberry pi touchscreen, NodeRED, etc, If used (source set to #4), must be sent at least every 60 seconds or will set an error in controller.

Corrections (-10 to +10) are applied to the Front and Rear temperatures, but not the OAT temperature.

NOTE 10 – Set shore capacity amps:

A “1” increments the Shore Amps Available. SHED will occur if shore amps exceed about 90% of the available (shed point)

0 = 10 (9 amps shed point)
1 = 15 (14 amps shed point)
2 = 20 (18 amps shed point)
3 = 25 (23 amps shed point)
4 = 30 (28 amps shed point) (Default)

NOTE 11 – SET_SHED_PRIORITY:

A “1” increments the Shed Priority. This determines who gets SHED first if a potential overload exists.

0 = TOGGLE (Default) toggle back and forth between the front and rear
1 = REAR A/C unit gets SHED first during an overload.
2 = FRONT A/C unit gets SHED first during an overload.

NOTE 12 – SET_SHED_TIME_LIMIT:

A “1” increments the ShedTimeLimit Times are 0, 900, 1800, 2700, 3600.   A time limit of zero disables the time limit.

SHED_TIME_LIMIT is sent from the control module to report the times as above.

NOTE 13 – SET_MAIN_CONFIG, GET_MAIN_CONFIG, MAIN_CONFIG:

As of Version 5.0.0, the Rpi's SD card is set to READ ONLY w/File System Overlay. This means we can't store configuration information on the Rpi's SD card. To work around this, configuration information is sent to the control module via MQTT message, where its stored inside the ESP32's processors read/write memory.

There are currently three configurations that can be stored in the ESP32; M=Main Screen, R=Remote screen, and P=PC or Laptop. The format for each configuration is identical, a 54 field, comma delimited text file. The letter in the first field (M, R, or P) indicates who's configuration this is, and is used by the ESP32 to store the file in the correct location.

Topic = SET_MAIN_CONFIG, Payload = A text string that contains 54 comma delimited fields. The first field contains a M, R, or P to indicate who's configuration this is.

Topic = GET_MAIN_CONFIG, Payload = M, R, or P. Tells the ESP 32 who's config file should be retrieved (M, R, or P) and sent as a MAIN_CONFIG message

The above two topics originate from a touchscreen or PC. They tell the ESP32 to either store the information, the "SET" command, or to retrieve the information, the "GET" command, for either the Main, Remote, or Pc.

Topic = MAIN_CONFIG, Payload = 54 comma delimited text string

This topic originates from the control module (the ESP32) in response to a "GET" command

UPDATED: Nov 24, 2021

APPENDIX B

VERSION INFORMATION

Controller Circuit Board artwork:

1.0 – Prototype
1.1 – Correct error on 12 volt feed to Q1
1.2 – Add 12 volt voltage divider for A/D converters
1.3 – Add diodes to A/C unit relay outputs. Re-arrange components for better fit
1.4 - Correct ground test point error. Relocate relay test points for better access, add silk screen messages to improve assembly.
1.5 – Add BMP280 sensor. Add reverse protection for 12 to 5 volt power supply. Clean up real estate. Make land areas larger for connectors
1.6 – Add traces to route 12 volts to the old ISX wires through a 3 amp fuse.
1.7 – Clean up traces and positions for easier assembly.
1.8 – Add configurable options for power available (ECC II)
Do not install components that are not needed.
Removed 3 amp fuse, 12 volts gets routed straight to yellow.
Remove BMP280
1.8A Replace several capacitor values
1.9 - Added SPST Panasonic relay Adlp112 as alternate relay
Removed DI2 jumper and replaced with DI3 (DI2 was on board blue LED and can't be used)
Increased spacing of resistor pads to make assembly easier
Decreased spacing of Tantalum pads to make assembly easier
Changed polarity orientation of C6 to make assembly easier
Moved 12 volt A/D sensing to protected circuit (after diode D7)
Increased clear area (no ground plane) for WiFi antenna
Moved connectors out 0.2mm to increase clearance with enclosure cover
Fixed connector orientation silkscreen to reduce chance of assembly error
1.10 Artwork changes, added BMP280 back in
1.11 Added SIG RX LED support on 4 pin IO connector.
Reposition components for easier assembly
Added support for single 5 volt power supply

Controller firmware:

2.10.4 Add Vref to read temps and 12 volts
Blink SIG RX when doing factory reset.
2.10.3 Reset defaults (write to SPIFFS) with MQTT command or EST button held at bootup
2.10.2 Blink IO23 with MQTTmessage received (SIG RX)
Add 50ms delay after issuing READ TEMP command
2.10.1 correct rear A/C hard start delay pointing at front A/C unit
2.10.0 Add pressure altitude back into firmware if the sensor is present.
2.9.2 Fix error in dtrostrf that was introducing spaces in MQTT messages
2.9.0 Impliment ESP32 interupt watchdog
2.8.2 Relocate WDT reset to inside loop
Correct Watchdog Counter from Integer to Long
Removed seriel prints
Adjust char lengths of dtostrf to ensure no memory issues
2.8.1 Send MQTT temperatures (FR, RR, and OAT) at least every 75 seconds.
2.8.0 Incorporate SPIFs to allow Rpi config files to be saved on the ESP32
Rpi CONFIG is sent via MQTT to control module where its stored
2.7.4 Add ability to FLIP screen (Removed this in next update)

2.7.3 Added SAVENOW to MQTT message (don’t wait for 30 second save.
Removed OFF from A/C SHEdding . Will always SHED.
2.7.2 Modify Temp sensor source “0” to ensure backward compatible with very early model control modules.
2.7.1 Add code to tolorate different manufactures of DS18B20
2.7.0 Modify 12 volt A/D converter algorithm to be more accurate
Add adjustable Shore Power amps
Add adjustable SHED Sequence
Add adjustable timer for alternating back and forth on SHED units.
Send messages for SHED, Hard start, Pre-SHED, and RUN
2.6.1 Implement task watchdog using MQTT messages as the reset. 5 minutes.
2.6.0 Add configurable SSID using jumper IO-5. WaiterControl = no jumper. WaiterControl2 = jumper
2.5.2 Tweeked the A/D converter correction factors
2.5.1 Add failed sensor back into firmware (99 sets a sensor as failed)
2.5.0 Add HVAC full control (heat-A/C at same time). Add faulty sensor bypass.
2.4.0 Fix rear A/C control method. Sound Alarm with MQTT message. Fix unsigned integer for correction.
2.3.4 Freeze protection changed from 32 to 35 degrees
2.3.3 Set external temperature error on startup
2.3.2 Allow corrections if source set to 4 and temp <> -99
2.3.1 Fix unsigned integer on VoltCOR. Would not remember VoltCOR if it was a negative value. (Voltage Correction)
2.3.0 Fix possible issue with BMP280 causing lockup.
Add IP address reporting.
Fix rear temperature source lockup on #3.
2.2.3 Add ability to use external temperature sources instead of sensors. Temperatures come to controller via MQTT message.
2.2.2 Add delay after issuing Temp read command to allow sensor to complete its conversions before attempting a read.
2.2.1 Extend Temperature error lockout from 5 seconds to 30 seconds
2.2.0 Add individual IO Relay control, AUTO, ON, OFF
2.1.4 Add BMP280 sensor data to MQTT stream
fix temperature reporting when in IO self test mode.
2.1.3 Fix Freeze protection time sequence
2.1.2 Add Volts correction factor -15 to +15 - Multiply this number by .000025 and add that to the raw AD conversion factor of .00505
2.1.1 Decrease Delay times for MQTT send messages from 200 to 50 ms
2.1.0 Add testio switch to pin 13. short to ground to start test. reset power (or send MQTT command) to stop
2.0.9 fix furnace set point min/max
2.0.8 Add error counter to the temperatures. modify test to cycle through all the outputs
2.0.7 modify how ECC configurations is set / and displayed
2.0.5 Send hour every time minute is sent (er need this for the Generator control
2.0.4 fix COR values, fix rear furnace MQTT reporting, add 12 volts reporting
2.0.3 In mode 6, turn blower off 30 seconds after compressor gets turned off delay start of compressor if blower turned off 5-7 seconds
2.0.2 add WiFi autoreconnect
2.0.1 Set up MQTT and watchdog
1.0.8 Tweek overload checking
1.0.7 Change Shedding - we now attempt to alternate shedding compressors
1.0.6a Rework SHEDing and Hard Start delay, added PRESHED
1.0.5 Disable shedding if A/C unit BTU set at zero.
1.0.4 Hysteria for mode 6 blower speed

```
1.0.3   fixed the software clock (rarely used) and added ability to simulate
          temps and shore power if temp sensor = -99 set temp error flag and
          do not allow furnace to run.
1.0.2  Added TEST_IO to allow testing the relays.+-
1.0.1  Add Bluetooth serial RX and Tx
1.0.0  RELEASE
```

Waiter ECC Rpi hat (touchscreen power supply and alarm board)

1.7 Add 0.1uf and 120uf caps Reposition components
1.6A Replace 100uf with 470uf capacitor.
0.1uf cap on A/D converter changed to 1.0uf
1.6 Reposition power supply to better clear ribbon cable.
Clean up artwork
1.5 Reposition Temp Sensor plugs for use with rear cover
Move A/D converter to 5 volts instead of 3.3
remove 3.3 volt power traces
1.4 NWR – Temp sensors and A/D converter added
PS moved to underside of board
Removed 3 pin connectors that supplied 12 volts
Added Yellow / Brown pigtails for 12 volt supply.
1.3 Vent hole for heat sink
Relocate 3 pin power connector to clear Rpi mount tab
1.2 Power supply added
1.1 Initial – No power supply, alarm only

Waiter ECC touch screen Application

5.0.3 Add WD reset to altitude and amps reporting
Add reset factory default to wifi screen]
5.0.2 Fix TRIM() that was accidentally introduced in ESP32 2.8.2
Fix wifi display screen
Increase watchdog time for remote display
5.0.1 Fix Alarm clock, ONCE would not write OFF to config
Initial setting of DIMs could cause crash.
Config type P (PC version) wasn't storing config info properly
PC version to act as tool for changing microSD card parameters
5.0.0 MAJOR OPERATING SYSTEM CHANGE – READ ONLY and FILE SYSTEM OVERLAY
Touch screen config now saved on the control module ESP32 SPIF
Screen flip and SSID changes are no longer allowed on Rpi due to
READ ONLY and File Overlay configuration
4.7.4 Add ability to flip screen 180 degrees
Add NITE MAX brightness
Shorten return to dim to 7 seconds
Call return to dim from RETURN button
4.7.3 Add SAVEALL and SAVE status to MQTT messages
Shore Amps selected shows on main screen
Shore Amps warning (yellow) text color
Saving Changes notification added to screen.
4.7.2 Change RunTime display from WiFi runtime to time since last boot
4.7.1 REMOTE ONLY Don’t allow CHAN CHANGE as WiFi option
4.7.0 Add adjustable Shore Power amps
Add adjustable SHED Sequence
Add adjustable timer for alternating back and forth on SHED units.
Fix status displays for SHED, Hard start, Pre-SHED, and RUN
re-arrange real estate on CONFIG page
4.6.4 Fix watchdog would always reboot to WaiterControl Ch 6.
4.6.3 Add IO runtime to WiFi screen
4.6.2 Add CHANGE CHAN (6 > 11 or 11 > 6) to the WiFi watchdog options.
4.6.1 Add WiFi uptime display, clean up displays, keep WiFi screen up for 10
minutes.
4.6.0 WiFi watchdog implemented.
4.5.1 Change inactivity timer for the MODE screen to 10 seconds.

4.5.0 Add support for two SSIDs selection.
4.4.1 if OAT source set to “0”, blank the OAT temperature display on main screen.
4.4.0 Disable WiFi power management in OS. Remove 2.2k thermistor support. Remove tabs display option selet
4.3.2 NOT RELEASED TO PUBLIC – Support for 2.2k thermistor.
4.3.1 Add furnace indicator to left side – clears up mode 7 use.
4.3.0 Add Auto heat-A/C control (mode 7). Add user ability to control display dim times. Add faulty sensor bypass.
4.2.0 Add MQTT message for sounding alarm
4.1.8 hard code path to CONFIG.XML depending on compile configurations
4.1.7 Add RUNTIME minutes to MQTT message
4.1.6 Add IP addresses to MQTT watchdog
4.1.5 Fix Refrigerator SP reporting
4.1.4 Add Try/Catch to MQTT to trap errors in the message format/content
4.1.3 Working on generator / Water heater – no changes
4.1.2 Add BMP280 Altitude atmospheric altitude readout
allow user config to blank/not blank screen tabs
4.1.1 blank tabs and screens after 30 seconds
4.1.0a Work on Water Heater
Add Water heater control
4.1.0 PRODUCTION RELEASE. add deg C conversion to schedule
4.0.3b disable MQTT screen.
4.0.3a Add Schedules
4.0.3 Fixed screen blanking. All screens now will go back to clock if no activity for 30 seconds
4.0.2a Sunup - Sundown dimming SU minus 15 to SD plus 60 will use day.
4.0.2 turn off IO test when either OFF buttons are pressed
4.0.1 Add Voltage correction. ABility to modify the volts per bit for the A/D converter
4.0.0 RaspberryPi 4 confirm compatable
2.1.4 Add remote screen capability
2.1.3 allow brightness to go down to 5
2.1.2 set screen brightness to max is not connected to controller
2.1.1 changed wording for SHORE AMPS to AC AMPS
* added MQTT back in
2.1.0 Release - Removed MQTT screen
2.0.5 MQTT display
2.0.4 Add temperature display F or C, fix number of funaces display
2.0.3 update ECC config page
2.0.2 Work on Leveler
2.0.1 Work on Generator page. change Sched stop to sched duration
2.0.0 MQTT for Refrigerator
1.0.9 Add refreig screen , getting ready to do multiple screens
1.0.8 Add PRESHED warning to display
1.0.7 Fix UP/DOWN buttons for mode6. fix SHED, RUN, DELAY indicator colors
1.0.6 chirp alarm
1.0.5 fix day/night, turn SHED off, reconfigure A/C BTU to disable shedding (change in ESP32 firmware
1.0.4 Add day min brightness, compute approx sunrise/sunset
1.0.3 Add IO to alarm buzzer on P1Pin11
1.0.2 fix error display, time display
1.0.1 add adjustable min brightness, fix Alarm functions. (still need IO)
1.0.0 INITIAL RELEASE – Carry over from Arduino Mega proof of concept

APPENDIX C

Temperature vs Resistance Conversion Chart for 10k Thermistors

°F	OHMS	°F	OHMS	°F	OHMS	°F	OHMS	°F	OHMS	°F	OHMS	°F	OHMS
-50	491,142	0	85,387	50	19,900	100	5,827	150	2,044	200	829	250	378
-49	472,642	1	82,719	51	19,377	101	5,697	151	2,005	201	815	251	373
-48	454,909	2	80,142	52	18,870	102	5,570	152	1,966	202	802	252	367
-47	437,907	3	77,656	53	18,377	103	5,446	153	1,929	203	788	253	362
-46	421,602	4	75,255	54	17,899	104	5,326	154	1,892	204	775	254	357
-45	405,965	5	72,937	55	17,435	105	5,208	155	1,856	205	763	255	352
-44	390,966	6	70,696	56	16,985	106	5,094	156	1,821	206	750	256	347
-43	376,577	7	68,535	57	16,548	107	4,982	157	1,787	207	738	257	342
-42	362,770	8	66,447	58	16,123	108	4,873	158	1,753	208	726	258	337
-41	349,522	9	64,428	59	15,711	109	4,767	159	1,720	209	714	259	332
-40	336,804	10	62,479	60	15,310	110	4,664	160	1,688	210	702	260	327
-39	324,597	11	60,595	61	14,921	111	4,563	161	1,657	211	691	261	323
-38	312,876	12	58,774	62	14,543	112	4,464	162	1,626	212	680	262	318
-37	301,622	13	57,014	63	14,173	113	4,368	163	1,596	213	669	263	314
-36	290,813	14	55,313	64	13,820	114	4,274	164	1,567	214	658	264	309
-35	280,433	15	53,669	65	13,473	115	4,183	165	1,538	215	648	265	305
-34	270,460	16	52,078	66	13,136	116	4,094	166	1,509	216	637	266	301
-33	260,878	17	50,541	67	12,809	117	4,007	167	1,482	217	627	267	296
-32	251,670	18	49,054	68	12,491	118	3,922	168	1,455	218	617	268	292
-31	242,821	19	47,616	69	12,182	119	3,839	169	1,428	219	607	269	288
-30	234,316	20	46,225	70	11,882	120	3,758	170	1,402	220	598	270	264
-29	226,138	21	44,879	71	11,589	121	3,679	171	1,377	221	588	271	280
-28	218,276	22	43,577	72	11,305	122	3,602	172	1,352	222	579	272	276
-27	210,716	23	42,318	73	11,029	123	3,527	173	1,328	223	570	273	273
-26	203,445	24	41,099	74	10,761	124	3,454	174	1,304	224	561	274	269
-25	196,451	25	39,919	75	10,500	125	3,382	175	1,281	225	553	275	265
-24	189,722	26	38,777	76	10,246	126	3,312	176	1,258	226	544	276	262
-23	183,248	27	37,671	77	9,999	127	3,244	177	1,235	227	536	277	258
-22	177,019	28	36,601	78	9,758	128	3,177	178	1,213	228	527	278	255
-21	171,023	29	35,565	79	9,525	129	3,112	179	1,192	229	519	279	251
-20	165,251	30	34,561	80	9,297	130	3,049	180	1,171	230	511	280	348
-19	159,696	31	33,590	81	9,076	131	2,987	181	1,150	231	503	281	244
-18	154,347	32	32,648	82	8,861	132	2,926	182	1,130	232	496	282	241
-17	149,197	33	31,737	83	8,651	133	2,867	183	1,110	233	488	283	238
-16	144,236	34	30,853	84	8,447	134	2,809	184	1,091	234	481	284	235
-15	139,458	35	29,998	85	8,249	135	2,752	185	1,072	235	473	285	232
-14	134,855	36	29,169	86	8,056	136	2,697	186	1,054	236	466	286	229
-13	130,420	37	28,365	87	7,867	137	2,643	187	1,035	237	459	287	225
-12	126,147	38	27,587	88	7,684	138	2,591	188	1,017	238	452	288	223
-11	122,030	39	26,832	89	7,506	139	2,539	189	1,000	239	445	289	220
-10	118,061	40	26,100	90	7,333	140	2,489	190	983	240	439	290	217
-9	114,235	41	25,391	91	7,164	141	2,440	191	966	241	432	291	214
-8	110,547	42	24,704	92	6,999	142	2,392	192	950	242	426	292	211
-7	106,991	43	24,037	93	6,839	143	2,345	193	933	243	420	293	208
-6	103,561	44	23,391	94	6,683	144	2,299	194	918	244	413	294	206
-5	100,254	45	22,764	95	6,530	145	2,254	195	902	245	407	295	203
-4	97,063	46	22,156	96	6,382	146	2,210	196	887	246	401	296	200
-3	93,986	47	21,566	97	6,236	147	2,167	197	872	247	395	297	198
-2	91,017	48	20,993	98	6,097	148	2,125	198	857	248	390	298	195
-1	88,152	49	20,438	99	5,960	149	2,084	199	843	249	384	299	193
												300	190

APPENDIX D

Touch screen CONFIG.XML file

<items>
<DISPLAY_UNITS>F</DISPLAY_UNITS>
<MIN_BRIGHT>12</MIN_BRIGHT>
<DAY_BRIGHT>100</DAY_BRIGHT>
<ALARM_HOUR>6</ALARM_HOUR>
<ALARM_MINUTE>15</ALARM_MINUTE>
<ALARM_DURATION>150</ALARM_DURATION>
<ALARM_AM>1</ALARM_AM>
<ALARM_MODE>0</ALARM_MODE>
<SHOW_BUTTONS>YES</SHOW_BUTTONS>
<INSTALL_LEVEL>NO</INSTALL_LEVEL>
<INSTALL_REFRIG>NO</INSTALL_REFRIG>
<INSTALL_GENER>NO</INSTALL_GENER>
<INSTALL_WTRHTR>NO</INSTALL_WTRHTR>
<XML_FR_SCHED_TIME_1>7:00 AM</XML_FR_SCHED_TIME_1>
<XML_FR_SCHED_TIME_2>9:30 AM</XML_FR_SCHED_TIME_2>
<XML_FR_SCHED_TIME_3>4:30 PM</XML_FR_SCHED_TIME_3>
<XML_FR_SCHED_TIME_4>10:00 PM</XML_FR_SCHED_TIME_4>
<XML_FR_SCHED_SETPOINT_1>70</XML_FR_SCHED_SETPOINT_1>
<XML_FR_SCHED_SETPOINT_2>65</XML_FR_SCHED_SETPOINT_2>
<XML_FR_SCHED_SETPOINT_3>80</XML_FR_SCHED_SETPOINT_3>
<XML_FR_SCHED_SETPOINT_4>65</XML_FR_SCHED_SETPOINT_4>
<XML_FR_SCHED_SETPOINT_1A>85</XML_FR_SCHED_SETPOINT_1A>
<XML_FR_SCHED_SETPOINT_2A>85</XML_FR_SCHED_SETPOINT_2A>
<XML_FR_SCHED_SETPOINT_3A>80</XML_FR_SCHED_SETPOINT_3A>
<XML_FR_SCHED_SETPOINT_4A>85</XML_FR_SCHED_SETPOINT_4A>
<XML_FR_SCHED_MODE_1>----</XML_FR_SCHED_MODE_1>
<XML_FR_SCHED_MODE_2>----</XML_FR_SCHED_MODE_2>
<XML_FR_SCHED_MODE_3>----</XML_FR_SCHED_MODE_3>
<XML_FR_SCHED_MODE_4>----</XML_FR_SCHED_MODE_4>
<XML_RR_SCHED_TIME_1>7:00 AM</XML_RR_SCHED_TIME_1>
<XML_RR_SCHED_TIME_2>9:30 AM</XML_RR_SCHED_TIME_2>
<XML_RR_SCHED_TIME_3>4:30 PM</XML_RR_SCHED_TIME_3>
<XML_RR_SCHED_TIME_4>10:00 PM</XML_RR_SCHED_TIME_4>
<XML_RR_SCHED_SETPOINT_1>70</XML_RR_SCHED_SETPOINT_1>
<XML_RR_SCHED_SETPOINT_2>65</XML_RR_SCHED_SETPOINT_2>
<XML_RR_SCHED_SETPOINT_3>80</XML_RR_SCHED_SETPOINT_3>
<XML_RR_SCHED_SETPOINT_4>65</XML_RR_SCHED_SETPOINT_4>
<XML_RR_SCHED_SETPOINT_1A>85</XML_RR_SCHED_SETPOINT_1A>
<XML_RR_SCHED_SETPOINT_2A>85</XML_RR_SCHED_SETPOINT_2A>
<XML_RR_SCHED_SETPOINT_3A>80</XML_RR_SCHED_SETPOINT_3A>
<XML_RR_SCHED_SETPOINT_4A>85</XML_RR_SCHED_SETPOINT_4A>
<XML_RR_SCHED_MODE_1>----</XML_RR_SCHED_MODE_1>
<XML_RR_SCHED_MODE_2>----</XML_RR_SCHED_MODE_2>
<XML_RR_SCHED_MODE_3>----</XML_RR_SCHED_MODE_3>
<XML_RR_SCHED_MODE_4>----</XML_RR_SCHED_MODE_4>
<XML_WTR_SCHED_TIME_1>11:00 am</XML_WTR_SCHED_TIME_1>
<XML_WTR_SCHED_TIME_2>9:00 pm</XML_WTR_SCHED_TIME_2>
<XML_WTR_SCHED_TIME_3>6:00 am</XML_WTR_SCHED_TIME_3>
<XML_WTR_SCHED_TIME_4>6:00 am</XML_WTR_SCHED_TIME_4>
<XML_WTR_SCHED_MODE_1>140</XML_WTR_SCHED_MODE_1>
<XML_WTR_SCHED_MODE_2>OFF</XML_WTR_SCHED_MODE_2>

<XML_WTR_SCHED_MODE_3>ON</XML_WTR_SCHED_MODE_3>
<XML_WTR_SCHED_MODE_4>----</XML_WTR_SCHED_MODE_4>
<XML_MINUTE_SR_POINTER>14</XML_MINUTE_SR_POINTER>
<XML_MINUTE_SD_POINTER>46</XML_MINUTE_SD_POINTER>
</items>