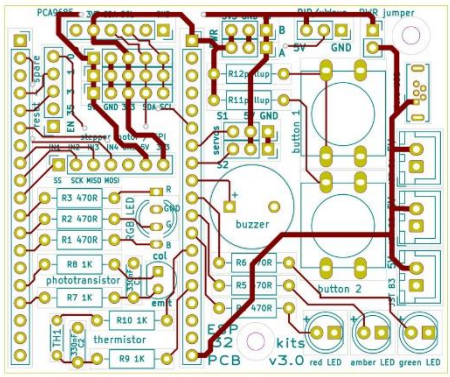
ESP32 Sensor box project



Build and usage documentation

version 1.0

Table of contents

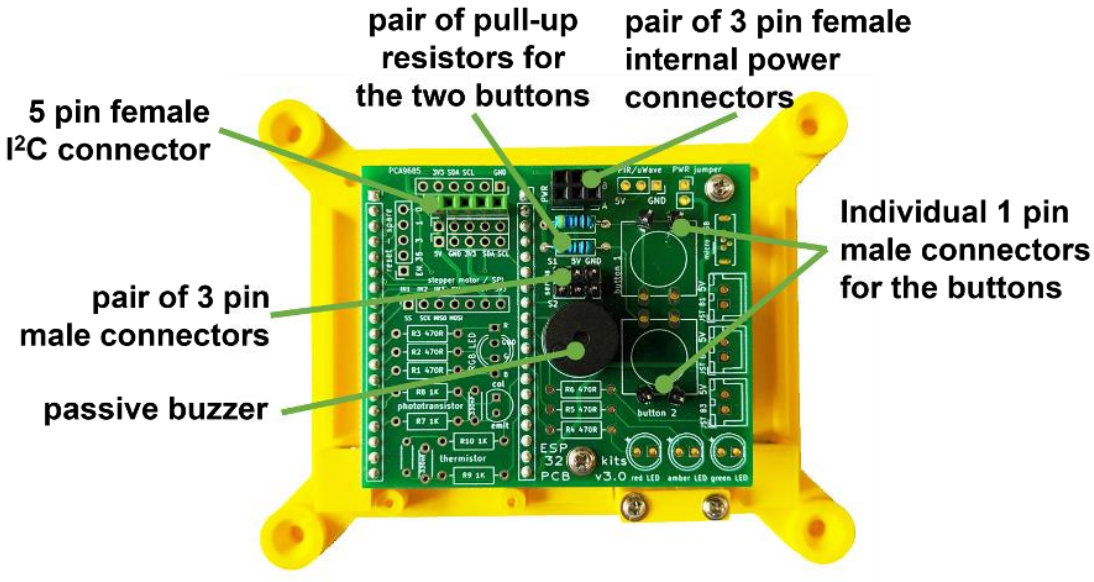
Introduction

This custom ESP32 microcontroller development uses the PCB from the ESP32 Maker Kit and some of the Kit components that connect to the PCB to produce a 'box' to which a number of sensors are connected, as shown on the right.

Whilst this custom development was undertaken for a specific purpose, it is being documented here since the 3D printed 'box' design approach and the software that has been developed for the data collection has some general purpose value.

The 3D print designs are available for download **here** and the software is available for download from **this GitHub repository**.

Using the ESP32 Maker Kit PCB provides an easy way to interconnect the 38-pin ESP32 module to these devices, but as there are not many connections needed, the PCB need only be sparsely 'populated' as shown below.



The connections shown above then support the following devices for the overall system build:

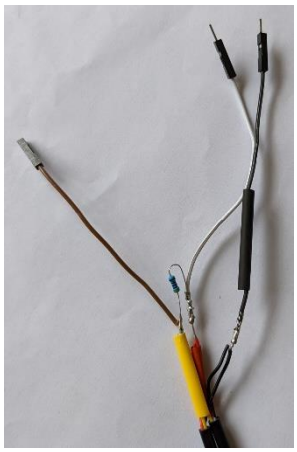
- two metal sheathed DS18B20 sensors (1-wire signal protocol devices) to remotely measure temperature;
- a DHT11 sensor to measure the temperature and humidity at the box top surface; along with
- two tactile buttons for various control purposes;
- a passive buzzer to provide audible alarms; and
- a small OLED display to continuously show the readings and other control/status events.

© 2022 ENMORE GREEN LTD

PCB build and device connections

The following are the various PCB connections:

- a single 5-pin (green) female header is used to connect the 128 x 64 pixel (OLED) display using I^2C;
- the 'signal' pins for the 2x 3-pin (black) male headers (usually used for servo connections) are used for the DHT11 signal connection and the 1-wire connection for the pair of DS18B20 sensors;
- the passive buzzer is connected in its dedicated position on the PCB;
- the pair of 3-pin (black) female PWR headers are used to supply power to the DS18B20 and DHT11 sensors;
- a pair of 4.7kΩ resistors, in their dedicated positions on the PCB, are used as 'pull-up' resistors for the tactile buttons; and
- 4x 1-pin (black) male headers are used to provide connectors for the buttons which are (remotely) installed in the top section of the 'box', i.e., they obviously cannot be installed in their usual place directly on the PCB.



It should also be noted that the 'harness' for the pair of DS18B20 sensors are prepared, as shown on the right, by putting a single 4.7kΩ 'pull-up' resistor between their connected signal wires and the +'ve power lead, and that each of the +'ve and GND leads from the sensors are also paired together so that only single connections are needed on the PCB within the 'box'.

Heat shrink tubing is used to insulate and strain relieve all the various joints.

3D printed 'box' design and build

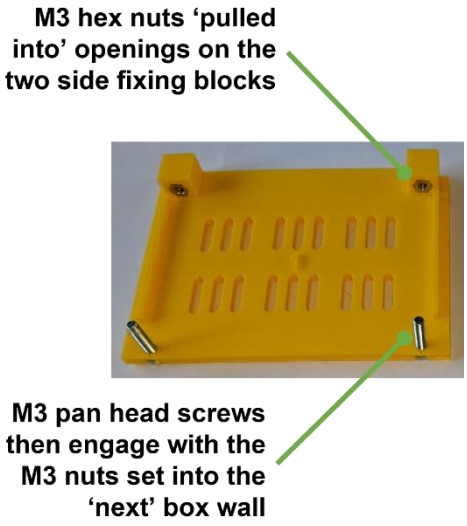
The 3D printed 'box' was designed using FreeCAD, and to allow easier assembly of the internal components as well the 3D printing of the parts to be much simpler, the design approach was for the PCB to be mounted in the bottom of the enclosure and for the top and each of the sides to be printed as separate elements that are then bolted together. All the 3D print designs are available for download **here**.

The base of the 'box' allows the standard ESP32 Maker Kit arrangement with the ESP32 38-pin module 'underslung' beneath the PCB to be attached to the base, with entry areas provided for the ESP32 power supply cable and for the two cables of the pair of DS18B20 1-wire temperature sensors.

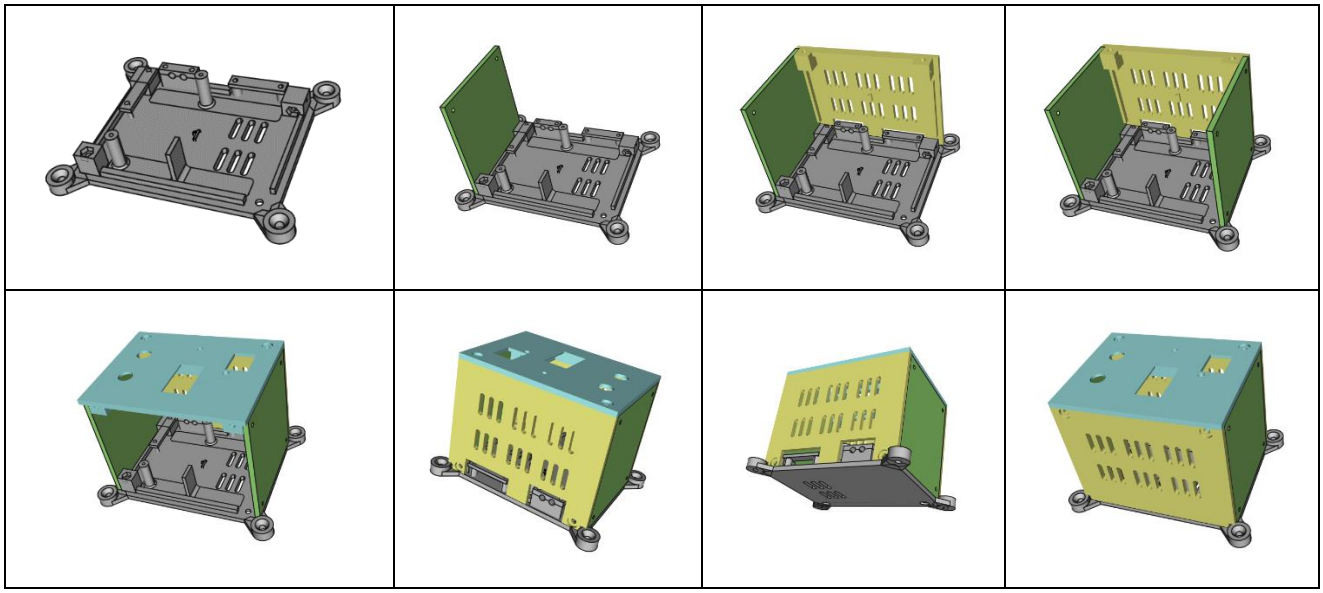
© 2022 ENMORE GREEN LTD

The ESP32 module is attached to the base by embedding into the 3D print design of the base, the same fixings designed to mount the ESP32 Maker Kit assembly (see **here** for details of the original Kit 'mounts').

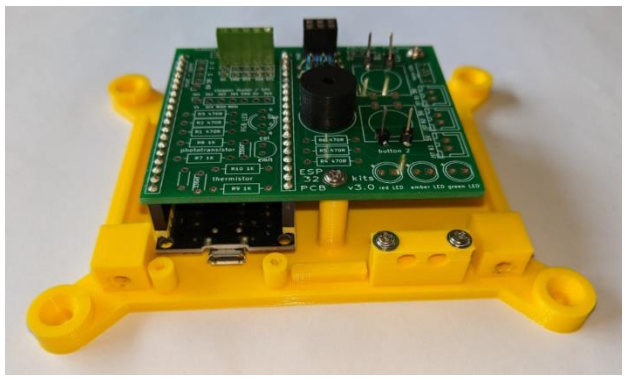
The 'bolting together' of the top, bottom and two main sides of the 'box' is accomplished by creating hex openings at pairs of fixing points on one side of each of the top, bottom and main box sides, that M3 nuts can be 'pulled into', and M3 pan head screws can engage with as shown in the image on the right.



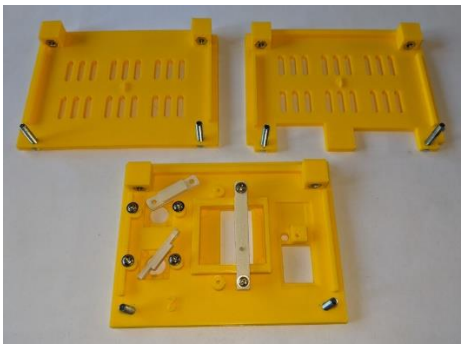
To check that all the components of the 'box' would align correctly, all the FreeCAD designs were 'assembled' into one visual model with various stages of the assembly sequence shown below with FreeCAD generated images.



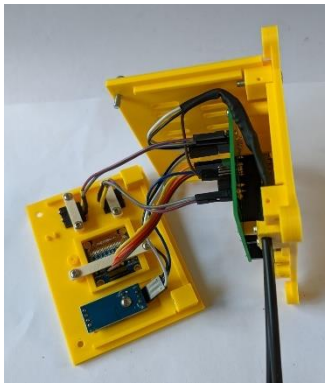
The real physical build sequence starts with the partial population of the PCB as discussed above, and then this is attached to the base of the 'box' as shown on the right.



The sensors, buttons and OLED are then connected to the PCB, the top and main side walls 'prepared' with their M3 nuts being inserted, plus the non-cable entry wall is bolted in place, all as shown below:


The DHT11, buttons and OLED are then attached to the 'box' top using 6mm long M2 self-tap screws, the side wall with the cable entry openings bolted in place, and the top is then bolted to the two main sides, all as shown below.



The final steps are to attach the two 'inserted' side panels using 6mm long M2 self-tap screws to complete the assembly as shown on the right.

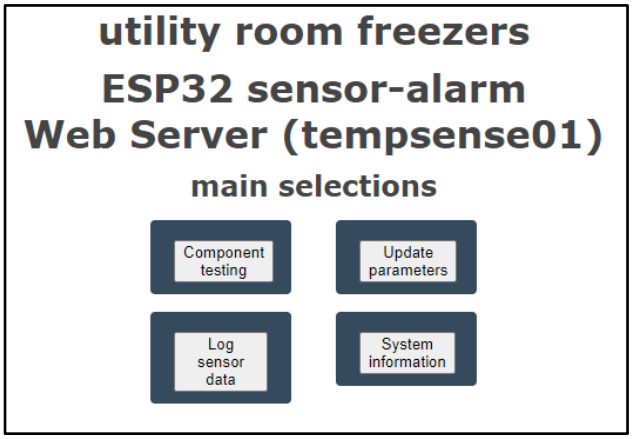
Sensor box software

The software for the ESP32, available for download from **this GitHub repository**, has been developed in C/C++ using the Arduino IDE, with the following overall structure:

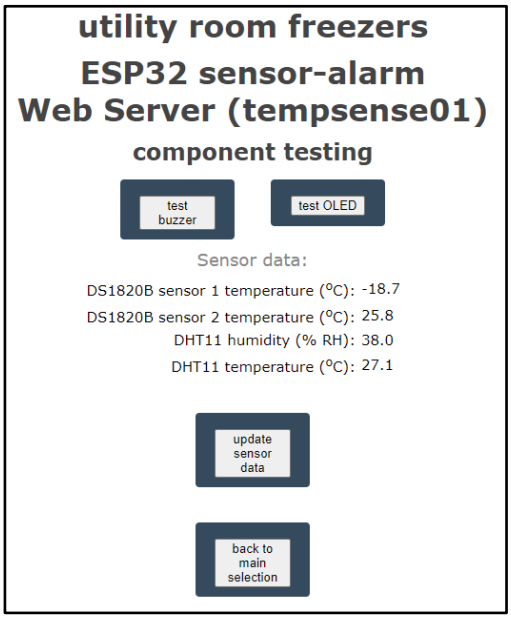
- The OLED provides a continuous output of the sensor data readings and control/status positions during start-up and when a significant change occurs.
- A set of manual controls for the system are provided through tactile button presses with
  - one button (with a red cap!) used to restart the system; and
  - the second button used to 'toggle' the alarm settings through various combinations of ON/OFF, OFF/ON, etc., where these 'state' positions trigger alarm actions when an alarm threshold is 'breached' for either of the two DS18B20 sensors.
- Most of the key operational parameters for the system are stored/read from individual SPIFFS files so the system can be reconfigured for various usage scenarios. These parameters include alarm thresholds for each of the DS18B20 sensors and whether a 'breach' is defined by being 'greater than' or 'less than' its threshold value - these SPIFFS files should be loaded into the microcontroller from a '*data*' folder in the normal Arduino IDE manner.
- Readings are taken on an interval basis (set by a SPIFFS parameter e.g., every 30 seconds), and
- If the alarm setting for an individual sensor is ON and an alarm condition occurs then the buzzer is sounded for 3 seconds, and an email is sent to a SPIFFS designated email address. The buzzer will continue to sound at the measurement interval if an alarm condition persists, but after the initial alarm notification email is sent, this is only repeated 'on the hour' another two times (to avoid email INBOX overload!!).

In addition to the button pressing control options, an extensive web interface has been developed with the following functionality:

- a main/home page with clickable buttons, as shown on the right, allows selection of individual web pages to:
  - carry out some component testing;
  - update all the various system parameters - also re-saving the updated values to SPIFFS;
  - do some basic data logging, i.e., saving the data for a period of time; and to
  - display some overall system information.



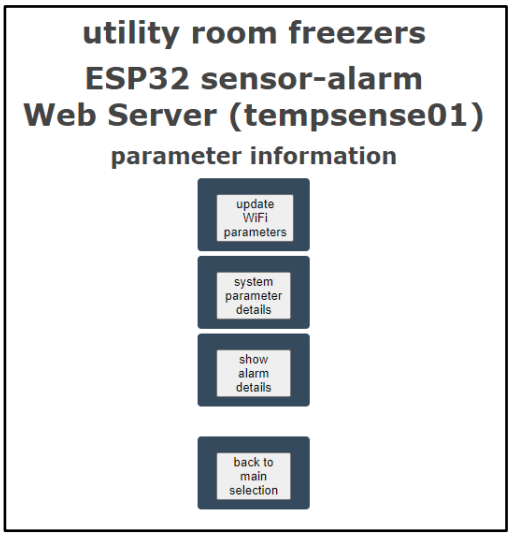
- The component testing options, as shown in the screen shot on the right, allow:
  - the buzzer to be sounded as a test;
  - the OLED to show a wide range of symbols to test its output; and to
  - provide an immediate update and display of the sensor data by clicking a web page button.



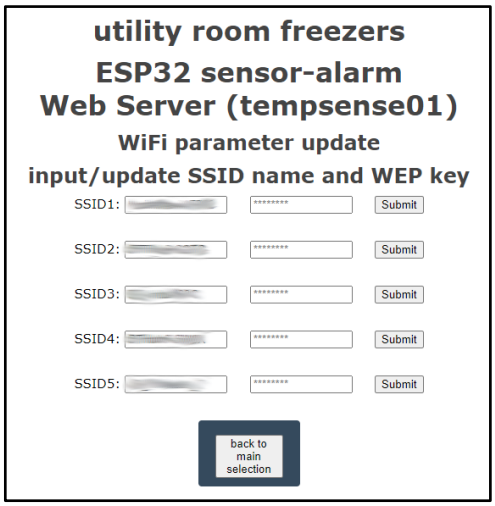
- The system parameters option provides an initial web page, as shown in the screen shot below right, that allows the following ranges of parameters to be selected:
  - WiFi parameters;
  - main system parameters; and
  - a more detailed display of the various alarm settings.

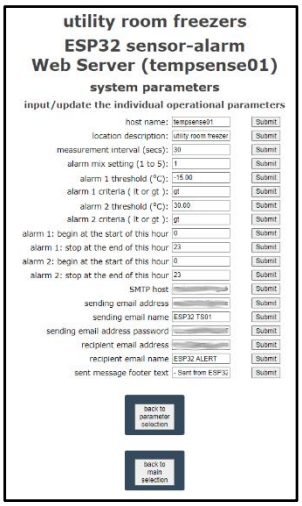
Each individual parameter range web page then allows each of the various parameters to be updated, and also re-saves the updated value to its SPIFFS file.

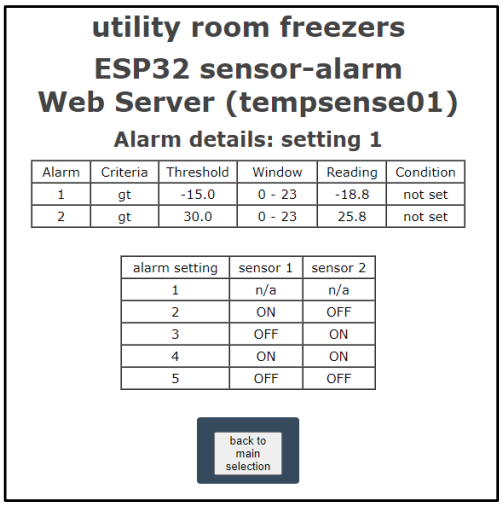
The alarm details page not only provides a reminder of what the individual alarm settings mean but also displays the current status of each alarm, analysing whether it is in an alarm condition or not.



The screen shots below show each of these 'parameter' screens:



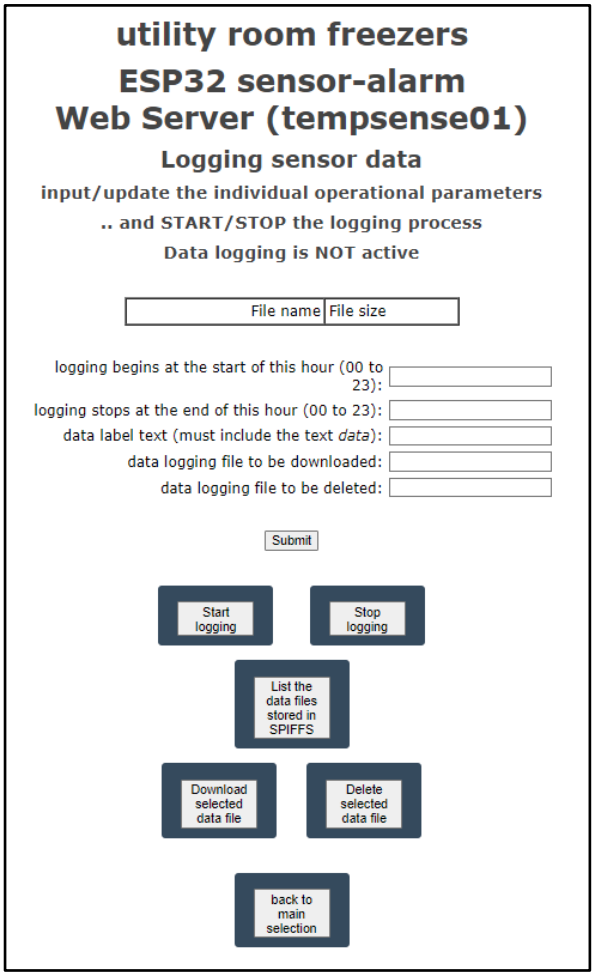




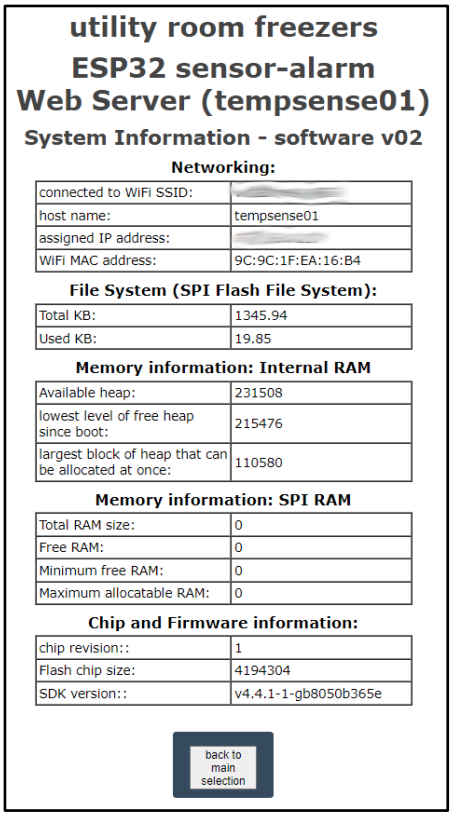
Alarm	Criteria	Threshold	Window	Reading	Condition
1	gt	-15.0	0 - 23	-18.8	not set
2	gt	30.0	0 - 23	25.8	not set

alarm setting	sensor 1	sensor 2
1	n/a	n/a
2	ON	OFF
3	OFF	ON
4	ON	ON
5	OFF	OFF

- The data logging option, shown in the screen shot on the right, allows:
  - a data logging start/stop hours window to be defined;
  - a SPIFFS data file name, where the data is stored, to have an individual text label to which the start/stop hours are also appended - the text label should include the word 'data' as this is used to filter for this type of file;
  - to list all the data logging files currently in SPIFFS i.e., all the files with the word 'data' somewhere within their file name;
  - an individual SPIFFS file to be 'set' which can be downloaded;
  - an individual SPIFFS file to be 'set' which can be deleted;
  - the data logging cycle to be put into operation - i.e., to start logging if within the designated hours window;
  - to stop data logging even within the hours window;
  - to download the currently 'set' download file; and
  - to delete the currently 'set' delete file.



Finally, the system information option shows a table of various system settings such as the IP address, SPIFFS file usage etc., as shown in the screen shot on the right.



END OF DOCUMENT