

coffee-sensor

Table of contents

- [Parts](#)
- [Library dependencies](#)
- [Setting up](#)
- [Build-time configuration](#)
- [Signing binaries](#)
- [OTA](#)
- [Configuration mode](#)
 - [Scale calibration](#)
- [Config server assets](#)

Parts

- Microcontroller LOLIN D1 mini
 - (https://docs.wemos.cc/en/latest/d1/d1_mini.html)
- Generic 128x64 OLED display module SSD1306 with I2C pinout
 - <https://www.addicore.com/OLED-128x64-Monochrome-p/ad304.htm>
 - OLED (probably): <https://www.datasheets.com/en/datasheet/ug-2864hlbeg01-wisechip-semiconductor-inc-71567627>
 - Controller: https://ucfdd013f8f3954b509bffd132fa.d1.dropboxusercontent.com/cd/0/inline2/A4CA0xJ8f02-H03uKl9I0W970m2HBDD-uvBdftsJ0r6Q4jT60h9DzDuakn0IxYsPIvIZUcggU4ewIC0A-an-4x0cfXqz83CmyeS5Yf5wIrb64ga4vBNiGS8cxLuF3Y2fuNysT5eLj07XdH0P1ns819F30Lgbi7T0H9wbG0Hu9TG6HhUkOEbE0BzpNVjhwiVjrBF9Am9EkZfcSADpTmXG8QHn-CkfJKggIEltSnojN9AkIV1CoRrNNYiy6njT-x5hfanwtqbgwg-2JfniB5_BCKD38WKwb8Cixw6oNP-4xus-67W4-ch2URQ1v1r0e6hu6LKfspuWbV-fgZLus65ctF/file#
- HX711 module
 - <https://www.amazon.com/DIYmall-Weighing-Conversion-Sensors-Microcontroller/dp/B010FG9RX0>
 - Controller: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf

Library dependencies

Library	Header	Description
Adafruit SSD1306 by Adafruit	Adafruit_SSD1306.h	Used to interface with the 128x64 OLED display. Chosen due to solid documentation and previous experience with the library.
Adafruit GFX Library by Adafruit	Adafruit_GFX.h	Dependency of Adafruit_SSD1306.
ArduinoJson by Benoit Blanchon	ArduinoJson.h	Used to serialize and deserialize JSON. Used for the system configuration in the flash memory, and for the configuration web server's data endpoint. Chosen because it's the de-facto library on the Arduino platform and it has extensive documentation and tools.
ArduinoWebsockets by Gil Maimon	ArduinoWebsockets.h	Used to communicate with the Node-RED server over WebSockets. Chosen because it seemed to have the most commits and activity on Github out of the alternatives.
DoubleResetDetect by Jens-Christian Skibakk	DoubleResetDetect.h	Used to detect a double-reset which enables the configuration mode of the sensor.
HX711 Arduino Library by Bogdan Necula, Andreas Motl	HX711.h	Used to interface with the HX711 load cell module. Chosen because of its simplicity.

Setting up

1. Obtain a [LOLIN/WeMos D1 Mini](#). The D1 Mini is a small ESP8266 development board with on-board flash memory, USB-to-serial converter, and a 5V to 3V3 regulator.
2. Follow the [schematic](#) and wire up the project. If you just want to test the system, you don't need to connect anything, but you might want to edit the code to send dummy values instead of reading from the load cell
3. Install the esp8266 Arduino board using the Boards Manager. Follow [these instructions](#).
4. Connect your device, open the Arduino IDE.
5. From the Arduino IDE, choose the right board:
 - Tools -> Boards -> LOLIN(WEMOS) D1 R2 & mini
 - If you obtained the D1 mini Pro variant which has the same footprint as the D1 mini but which features a ceramic on-board antenna and a U.FL connector for an external antenna, choose LOLIN(WEMOS) D1 mini Pro
6. Install the [required libraries](#) from the Library Manager (Tools -> Manage Libraries...)
7. Update the relevant build-time configuration
 - If you're running the infrastructure locally, you'll probably want to replace `FIRMWARE_UPDATE_URL`
8. Compile and upload to your D1 Mini

If you're building firmware to serve via the OTA server:

9. Generate cryptographic keys for binary signing
 - note: steps only for linux and macos
 - 1. Open this directory on the command line
 - 2. Run `openssl genrsa -out private.key 2048`
 - Leave passphrase empty
 - 3. Run `openssl rsa -in private.key -outform PEM -pubout -out public.key`
 - 4. Now you have the private and public keys. The private key is **private** it is used to sign the firmware binary, which is verified with the public key. The public key can be shared, and in fact needs to be updated in the next step
 - 5. Copy the contents of `public.key` into the `_pubkey` variable in `SigningKey.cpp`
10. Sign the firmware. If you're using Arduino IDE and you have both `private.key` and `public.key` in the same directory as the `.ino` file, the [IDE will generate a signed copy](#) of the firmware for you. Enable verbose logging during compilation to see where the file is output. Ignore the *legacy* signed binary.
 - If you're on Windows or "Enabling binary signing" did not appear in the output, follow [these steps](#) to sign it manually.

More in [Signing binaries](#) .

Build-time configuration

The sensor has multiple build-time macros which can be modified.

File	Macro	Default	Description
coffee-sensor.ino	SENSOR_MEASUREMENT_INTERVAL	2000 (2 s)	How often the sensor takes load cell measurements and sends them. Milliseconds.
	SENSOR_FW_UPDATE_INTERVAL	5*60*1000 (5 min)	How often the sensor polls for firmware updates. Milliseconds. During updates, measurements are not taken or sent.
	DRD_TIMEOUT	2	How long, in seconds, to wait for a double-reset during boot to enter configuration mode.
	FIRMWARE_UPDATE_URL	"https://ota.ele2.cxcorp.systems:443/update"	The HTTP/HTTPS URL of the firmware update endpoint.
	FIRMWARE_UPDATE_HTTPS	1	Whether HTTPS is used for the firmware update. If <code>FIRMWARE_UPDATE_URL</code> starts with <code>https://</code> , keep this defined. If not, comment out (<code>//</code>) this macro so it is not defined.
	firmwareUpdateHttpsFingerprint		The fingerprint of the SSL (the S in HTTPS) certificate served by the update server. You can find this via your browser , or via command line . If <code>FIRMWARE_UPDATE_HTTPS</code> is not defined, that is, if you're not using HTTPS, this is not used.
	CONFIG_AP_SSID	"OH sensor admin"	Configuration mode's WiFi access point's SSID.
	CONFIG_AP_PASSPHRASE	"12345678"	Configuration mode's WiFi access point's passphrase.
	CHANGE_WINDOW_INTERVAL	10000 (10 s)	How often the configuration mode's

Configuration mode

If the sensor's reset button is clicked twice within two seconds, the server starts in configuration mode. In maintenance mode, the sensor starts its own WiFi network named OH sensor admin with the passphrase 12345678. Connecting to this WiFi access point and navigating to <http://192.168.1.1/> opens the configuration portal, from which the WiFi client credentials and WebSocket URL can be specified.

Scale calibration

The scale calibration can be accessed from the configuration portal. The portal has instructions how to calibrate the sensor as seen below.

Scale calibration

1. Set **Scale multiplier** to **1.0** and press Save.

2. Press **Tare**.

3. Place a known weight on the scale, wait a second, and refresh this page.

4. Divide the **Current weight** by your known weight. This is the approximate value for **Scale multiplier**.

5. Adjust the parameter in step 4 until you get an accurate reading.

Current weight

Scale multiplier (set_scale)

Save

The HX711 module uses this multiplier to scale the calculated values.

Scale offset (set_offset)

Save

The current raw tare offset. This value is added to the reading to offset the weight.

Tare

tare()

Let sensor calculate the **Scale offset** from the current weight.

Config server assets

The config server's assets (index.html, bootstrap.css) are served directly from the ESP's flash memory. The bootstrap.css file is pre-compressed with gzip, and the gzipped bytes are compiled directly into PROGMEM (see ConfigWebAssets.cpp). The index.html file is simply ran through a minifier.

These files' unminified or uncompressed versions can be found from the assets directory.

To modify index.html, edit the HTML, then minify it [here](#) (from the right, untick *Remove optional tags!*). If the minifier doesn't minify the JavaScript code inside the <script> tags, use [this](#) JS minifier to minify it and replace it with the minified version. Copy this minified output to your text editor of choice, and do the following replacements with Replace All: \ -> \\ and " -> \". These escapes are needed so that the string can be copy pasted into the ConfigWebAssets.cpp file.

As for bootstrap.css, should there be a need to modify it, get the minified version (bootstrap.min.css), then run `gzip -k9 index.min.html && xxd -i index.min.html.gz > copyme.txt` to compress the HTML with gzip and generate almost copypasteable C code. Replace the new length and bytes in the ConfigWebAssets.cpp file. Remember to keep the static const char and PROGMEM words that are present in the code currently.