coffee-sensor

Table of contents

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.dl.dropboxusercontent.com/cd/0/inline2/A4CAQxJ8f02-H03uKL9I0W97Qm2HBDD-uvBdftsJ0r6Q4jT60h9DzDuaknOIxYsPIvIZUcggU4ewIC0A-an-4xQcfXqz83CmyeS5Yf5wIrb64qa4vBNiGS8cxluF3Y2fuNysT5eLj07XdHOP1ns819F30Lgbi7TQH9wbGOHu9TG6HpUkOEbE0BzpNVjhwijvjrBF9Am9EkZfCSADpTmXG8QHn-CkfJKgqIEltSnojN9AkIV1CoRrNNYiY6njT-x5hfanwtqbqwg-2JfniB5_BCKD38WKwb8Cixw6oNP-4xus-67W4-cb2URQ1v1rOe6hu6lKfspuWbV-fgZLus65ctF/file#
- HX711 module
  - https://www.amazon.com/DIYmall-Weighing-Conversion-Sensors-Microcontroller/dp/B010FG9RXO
  - 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.

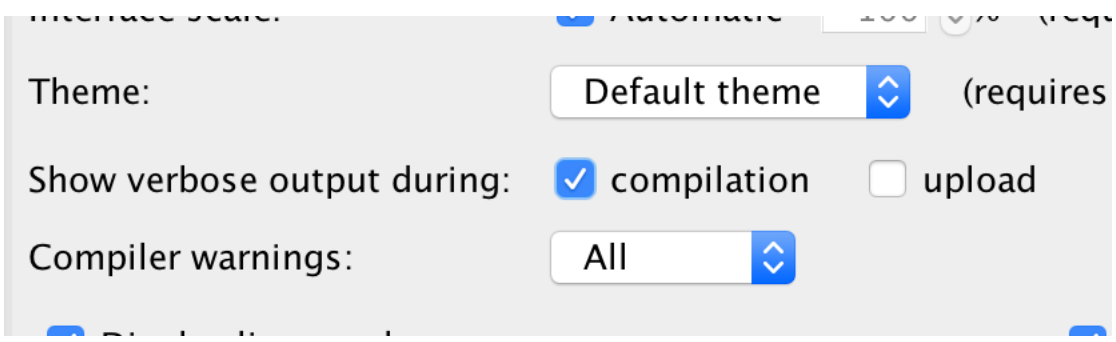
File	Macro	Default	Description
coffee-sensor.ino	SENSOR_MEASUREMENT_INTERVAL	`2000` (2 s)	How often the sensor takes load cell measurements and sends them. Milliseconds.
	SENSOR_FW_UPDATE_INTERVAL	`5*60*1000` (5 min)	How often the sensor polls for firmware updates. Milliseconds. During updates, measurements 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 `FIRMWARE_UPDATE_URL` starts with `https://`, keep this defined. If not, comment out (`//`) 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 `FIRMWARE_UPDATE_HTTPS` 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 display modes change.

ConfigPersistence.h	DEFAULT_SCALE_MULTIPLIER	-230000	The default multiplier value passed to the HX711 library if device has not been calibrated yet.
	DEFAULT_SCALE_OFFSET	-117878	The default weight offset.
SigningKey.cpp	_pubkey		The `public.key` signing key. Used to verify firmware from the OTA update server. Firmware should be signed with a private key matching this public key.
Version.h	_FW_VERSION	`"0.8.2"` or whichever the current version is	The version number of the firmware. Sent to the firmware update server when asking for new firmware. The format is semver, so the version should be `number.number.number`

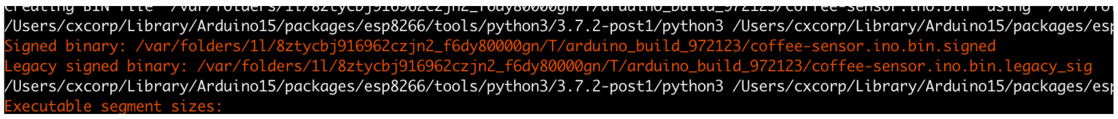
Signing binaries

The OTA system requires all produced firmware binaries to be signed with our private key. To produce signed binaries, copy `private.key` into this directory. If `public.key` does not exist, generate it from the private key with `openssl rsa -in private.key -outform PEM -pubout -out public.key`. If both of these files are present, Arduino IDE will sign the binary after compiling.

To see where the signed binary is produced, enable Show verbose output during compilation from Arduino IDE's preferences:


Theme: Default theme (requires
Show verbose output during: [✓ compilation] upload
Compiler warnings: All

After compilation, you should see the following in the console:


```
/Users/cxcorp/Library/Arduino15/packages/esp8266/tools/python3/3.7.2-post1/python3 /Users/cxcorp/Library/Arduino15/packages/esp
Signed binary: /var/folders/1l/8ztycbj916962czjn2_f6dy80000gn/T/arduino_build_972123/coffee-sensor.ino.bin.signed
Legacy signed binary: /var/folders/1l/8ztycbj916962czjn2_f6dy80000gn/T/arduino_build_972123/coffee-sensor.ino.bin.legacy_sig
/Users/cxcorp/Library/Arduino15/packages/esp8266/tools/python3/3.7.2-post1/python3 /Users/cxcorp/Library/Arduino15/packages/esp
Executable segment sizes:
```

The path after "Signed binary:" is where the signed binary is.

OTA

The sensor checks for firmware updates over HTTP over fixed intervals. This check happens intelligently so that the sensor sends its current version to the server, and the server either responds with a 304 Not Found if the version is up to date, or the updated binary if an update as found.

The binaries are signed via public key cryptography, and the sensor validates the signature of a downloaded binary so that only authentic updates are applied.

The public key used for verification is configured through `SigningKey.cpp`. A copy of this key is found in the file `public.key`. The Arduino IDE supposedly handles this if both `private.key` and `public.key` is found, but testing showed that the public key still needs to be applied manually on the ESP8266 Arduino board v2.6.3, which is why the public key needs to be copied to `SigningKey.cpp`.

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.

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.