Building and flashing the signals app using VScode and PlatformIO IDE on an Apple Silicon MAC mini M1

1 Introduction

At the ConTEXt 2025 meeting in Poland the ConTEXt Watch was given to the meeting participants. Using colourfull lights, this device gives a visual display of the progress of a ConTEXt run and is particularly usefull when running a large number of files. The ConTEXt Watch is based around a ESP32-S3-WROOM-1 which is connected to a computer via a serial port. More details on the device (otherwise known as a Squid) can be found in the Signals manual in the ConTEXt distribution.

This document gives instructions on how to build the signals software, in this case on a MAC mini M1 which uses an ARM cpu, and flash it to the ESP32 using VScode and the `PlatformIO IDE extention pack`. In addition you will also require the `C/C++ pack`, and it makes no harm to also install the `C/C++ Extention Pack`, `C/C++ Themes`, and `Clang-Format Tools`. Once all the extention packs are installed we can proceed with the building of the signals software.

2 Building the signal software

- Unzip the signals.zip file into your Home folder. You will see a directory named codebase with a number of sub-directories. The sub-directory vscode is the directory where the build files are stored that are used by PlatformIO.

- Open VScode with a `New Window` and click on the `Home icon` in the bottom toolbar to open the PlatformIO IDE. The`PlatformIO Home` will open, then click `New Project`. The `Project Window` will open.

- Give the project a `Name`, then select `Board`, I used `ESP32-S3-DevKitC-1-N8R2`. Then select `Framework: Arduino`. Leave `Location:` checked. Click `Finish` and the new project is created.

- Navigate to the `codebase>vscode` directory. Copy the files in the lib directory to the `PlatformIO>Projects>New Project Name>lib` directory

- Navigate to the `codebase>vscode>src` directory and copy the file `context-lmtx-signal.cpp` to `PlatformIO>Projects>New Project Name>src` directory and delete the file `main.cpp` there.

- Replace the code in the `platformio.ini` file of the new project and save it with the following content:

  ```
  ; PlatformIO Project Configuration File
  ```

```
;
;   Build options: build flags, source filter
;   Upload options: custom upload port, speed and extra flags
;   Library options: dependencies, extra library storages
;   Advanced options: extra scripting
;
; Please visit documentation for the other options and examples
; https://docs.platformio.org/page/projectconf.html

[env:esp32-s3-devkitc-1]
platform = espressif32
board = esp32-s3-devkitc-1
framework = arduino
board_build.arduino.memory_type = qio_qspi
board_build.flash_mode = qio
board_upload.flash_size = 4MB
board_upload.maximum_size = 4194304
board_build.partitions = default.csv
lib_deps =
       stnkl/ESPEssentials@^2.1.5
       tzapu/WiFiManager@^2.0.17
       littlefs
       fastled/FastLED@^3.10.1
build_flags =
       -DARDUINO_USB_MODE=1
       -DARDUINO_USB_CDC_ON_BOOT=1
       -DSIGNAL_USE_DEVICE=ESP32
       -DSIGNAL_USE_BUTTONS=1
```

- We are now ready to perform the build operation. Click on the tick mark in the bottom toolbar. The build operation will commence and after a few seconds, if everything has gone to plan, the build will be successful.

- After the successful build we can now flash the ESP module by clicking the arrow next to the build tick. PlatformIO will automatically find the serial port though which it can upload the code. If it doesn't, click the auto icon on the bottom tool bar and that will show you a list of ports and if your serial port is not listed you will have to download a suitable driver for it. If the flash is successfull take a note of the serial port used as this will be required for the next step, which is configuring signal.

- Navigate to the `tex>texmf-context>context>data>signal` directory and open the `ctxsignals-template.lua file`. Under the line beginning with `--OSX` is the line beginning with `--port`, replace this with the name of the serial port used to flash the ESP module, like `port = "/dev/serialportname"`.

- Rename the file to `ctxsignals.lua`.

3 Testing the module

Testing the module can be done by running a simple file as follows:

```
% signal=squid

\starttext
    \dorecurse {100} {
        \dorecurse {100} {
            \samplefile{ward}
            \par
        }
        \page
    }
\stoptext
```

with the command:

```
context --script signalsTest  --squid  --test
```

A successfull run will have all the lights coloured green

4 Further reading

It goes without saying that the signals manual in the distribution should be your first port of call. On the internet there are may youtube videos showing applications of the ESP32 module, that are worth viewing for background information on the module.

It is likely that the the procedure described is similar on other platforms where PlatformIO is available. The USB set up is of course different then.

Keith McKay, 2025