

[This document contains detailed information about the project and was created as a go-to place to find answers to all of your HOWs and WHYs, so the software support and further project development are easier.]

[ August 2024 ]

[42 prague smart sigN.]

[technical documentation.]

[Table of contents.]

[SUMMARY.]

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | About the project  General program runtime description  Hardware description  Hardware decisions explained  Circuit diagrams and schematics  Graphics  Description of the program files  Description of the program functions  Description of the program libraries and their functions  Description of the program constants  Architectural decisions explained  Communication protocols  Power management  Safety considerations  The Don’ts of changing the program  Development tools  Microcontroller Flashing process  Suggestions for dealing with confidential information  Known bugs  Future development suggestions  Sources |

[ABOUT THE PROJECT.]

[**General description of the program run**.]

Let's take a look at one day of the 42 Prague Smart Sign‘s life. Let’s say that on this particular day there is an exam scheduled for 13:00 and it will take 3 hours.

It is still night and the Sign is showing the cluster number with the default icons from yesterday, while still being asleep. Time passes by. Now, it is 6:00 in the morning. The Sign wakes up to check if there are exams today. It goes to the Intra server and sees that there will be an exam starting at 13:00 and ending at 16:00. The Sign replaces the default icons with a note “The cluster will be reserved for an exam today at 13:00” while still displaying the cluster number. Since there is nothing more for the Sign to do, it sets its alarm clock for 12:00 (an hour before the exam) and goes to sleep.

It is 12 o'clock and the Sign wakes up again. It checks Intra to make sure the exam was not canceled during its sleep and replaces the cluster number with a big warning sign that says: “RESERVATION! The cluster is reserved for an exam. Please, vacate it in due time. You have XX minutes left”. Instead of XX it first says 50 minutes, then 25 minutes and finally 5 minutes left.

Finally, it is 13:00. The exam begins. The Sign changes the previous warning sign for a new one, saying “DO NOT ENTER! Exam in progress! The exam will end at 16:00”. At this point the Sign has nothing else to do, so again it sets its alarm clock for 16:00 and goes to sleep.

At 16 o'clock the Sign wakes up, checks Intra, finds no more exams for today, replaces the warning sign with a cluster number with the default icons, sets its alarm clock until the next scheduled wake up - in this case 18:00 - and goes back to sleep.

At 18 o'clock the Sign wakes up, checks Intra and finds no more exams. Its work for today is over. It goes to sleep to wake up again the next morning at 6.

[PROGRAM RUN STEP-BY-STEP.]

**НАЧАЛО РАБОТЫ ПРОГРАММЫ**

1. включение
2. инициализация Серийного порта

* вывод тире в серийный порт даёт время синхронизовать передачу данных с компьютером и избежать потерю важных данных

1. инициализация файловой системы

* при ошибке инициализации файловой системы не будут доступны следующие функции: восстановление последнего использованного чата Телеграм, значения Секрет и значение флага ОТА после переустановки программы, после отключения электропитания или после программного ресета (т.е. после всех случаев когда данные памяти RTC теряются); запись значения последнего использованного чата Телеграм, значения Секрет и значение флага ОТА

1. инациализация кнопок (отключено из-за багов)
2. инициализания ADC для замеров аккумулятора
3. инициализация SPI порта дисплея
4. проверка причины перезагрузки

* также восставливает значение последнего использованного чата Телеграм, значение Секрет и значение флага ОТА после переустановки программы, после отключения электропитания или после программного ресета (т.е. после всех случаев когда данные памяти RTC теряются)
* также уводит устройство в сон на 24 часа если срабатывает BROWN OUT детектор. Он срабатывает если заряда аккумулятора недостаточно для продолжения работы. В таком случае **[РАБОТА УСТРОЙСТВА ЗАКАНЧИВАЕТСЯ ЗДЕСЬ]**, пока аккумулятор ни будет заряжен.

1. проверка аккумулятора
   * + из-за технических особенностей устройства мы можем определить уровень заряда аккумулятора только когда он почти разряжен. Точные замеры аккумулятора можно проводить ориентировочно между 3% и 0% заряда аккумулятора. В показаниях ADC это соответствует 800 и 400.
     + берём 5 замеров уровня заряда и высчитываем их среднее значение
     + все показатели выше 800 означают, что аккумулятор достаточно заряжен и сообщать о разряженном аккумуляторе не требуется – выходим из функции
     + подключаемся к Wi-Fi чтобы сообщить о состоянии аккумулятора в Телеграм
     + все показатели ниже 400 означают полностью разряженный аккумулятор. Ни смотря на то, что не сработал BROWN OUT детектор в предыдущем шаге, с таким низким зарядом нельзя продолжать работу. Сообщаем о разряженном аккумуляторе в Телеграм, выводим на дисплей сообщение "Low battery" и уводит устройство в сон на 24 часа. В таком случае **[РАБОТА УСТРОЙСТВА ЗАКАНЧИВАЕТСЯ ЗДЕСЬ]**, пока аккумулятор ни будет заряжен.
     + показатели между 700 и 600 могут означать что устройство заряжается
     + если мы до сих пор не вышли из функции, но показатели ниже 800, значит аккумулятор уже разряжен, но ещё можно продолжить работать. Сообщаем о разряженном аккумуляторе в Телеграм, выводим на дисплей сообщение "Low battery" и продолжаем исполнение программы.
2. инициализация функции OTA (отключено из-за блокировки файерволом)
3. переход в режим OTA (отключено из-за блокировки файерволом)
4. делаем выбор в каком режиме продолжать работу: в режиме номера кластера (дефолтный режим) или в режиме экзамена
   * + Режим номера кластера отображает номер кластера + пиктограммы (в обычный день) или сообщение-предупреждение об экзамене (в день экзамена) или сообщения об ошибке (невозможность получить данные экзаменов, истёкший срок действия Секрета, низкий заряд аккумулятора). Этот режим активен 99% времени.
     + Режим экзамена включается 1 час заранее до начала экзамена, показывает предупреждение о скором начале экзамена, потом переключается в предупреждение о проходящем экзамене и после окончания экзамена переходит обратно в Режим номера кластера. Этот режим активен только в день экзамена, 1 час до начала экзамена + всё время проведения экзамена.
     + Двойная проверка флага состояния экзамена в этой функции необходима для перехода из одного режима в другой. Не менять!

**В РЕЖИМЕ НОМЕРА КЛАСТЕРА**

1. подключение к сети Wi-Fi
2. проверка входящих сообщений в Телеграм чате
   * + Через чат Телеграм может прийти новый Секрет, который будет полезен далее при запросе данных экзаменов
3. синхронизация времени, даты и летнего/зимнего режима времени с NTP сервером
   * + Без данных времени нельзя обеспечить надёжную работу устройства. Если поле нескольких попыток получить данные времени не удалось, на дисплей выводится ошибка а само устройство уводится в сон до следующего запланированного пробуждения. В таком случае **[РАБОТА УСТРОЙСТВА ЗАКАНЧИВАЕТСЯ ЗДЕСЬ]**, пока ни удастся получить данные времени при одном из будущих запланнированных пробуждений.
4. получаем данные экзаменов на актуальный день из Интры
   * + Без данных экзаменов нельзя обеспечить надёжную работу устройства. Если поле нескольких попыток получить данные экзаменов не удалось, на дисплей выводится ошибка а само устройство уводится в сон до следующего запланированного пробуждения. В таком случае **[РАБОТА УСТРОЙСТВА ЗАКАНЧИВАЕТСЯ ЗДЕСЬ]**, пока ни удастся получить данные экзаменов при одном из будущих запланнированных пробуждений.
     + заходим на сайт Интры
     + логинимся на сайте Интры
     + заходим на страницу расписания на сегодняшний день
     + читаем полученный HTML код до обнаружения данных об экзамене
     + очищаем данные от лишнего мусора
     + сверяем полученные данные с имеющимися данными в памяти
     1. если данные отличаются, то изменяем их на дисплее;
     2. если нет, то оставляем как есть
5. настраиваем время следующего включения
6. выключение питания дисплея
7. выключение

[PROGRAM RUN STEP-BY-STEP.]

STARTING THE PROGRAM

1. Turning on

2. Initializing the Serial Port

• Outputting a dash to the serial port gives time to synchronize data transfer with the computer and avoid losing important data

3. Initializing the file system

• If the file system initialization fails, the following functions will not be available: restoring the last used Telegram chat, the Secret value, and the OTA flag value after reinstalling the program, after a power outage, or after a software reset (i.e. after all cases when RTC memory data is lost); record the value of the last used Telegram chat, the Secret value and the OTA flag value

4. initialize the buttons (disabled due to bugs)

5. initialize ADC for battery measurements

6. initialize the SPI port of the display

7. check the reboot reason

• also restores the value of the last used Telegram chat, the Secret value and the OTA flag value after reinstalling the program, after a power outage or after a software reset (i.e. after all cases when RTC memory data is lost)

• also puts the device to sleep for 24 hours if the BROWN OUT detector is triggered. It is triggered if the battery charge is insufficient to continue operation. In this case, [DEVICE OPERATION ENDS HERE] until the battery is charged.

8. battery check

• due to the technical features of the device, we can determine the battery charge level only when it is almost discharged. Accurate battery measurements can be taken approximately between 3% and 0% of the battery charge. In ADC readings, this corresponds to 800 and 400.

• take 5 measurements of the charge level and calculate their average value

• all readings above 800 mean that the battery is sufficiently charged and there is no need to report a low battery - exit the function

• connect to Wi-Fi to report the battery status to Telegram

• all readings below 400 mean a completely discharged battery. Despite the fact that the BROWN OUT detector did not work in the previous step, you cannot continue working with such a low charge. We report a low battery in Telegram, display the message "Low battery" on the display and put the device to sleep for 24 hours. In this case, [DEVICE OPERATION ENDS HERE] until the battery is charged.

• indicators between 700 and 600 may mean that the device is charging

• if we still haven't exited the function, but the indicators are below 800, then the battery is already discharged, but we can still continue working. We report the discharged battery in Telegram, display the message "Low battery" on the display and continue executing the program.

9. initializing the OTA function (disabled due to blocking by the firewall)

10. switching to OTA mode (disabled due to blocking by the firewall)

11. choosing which mode to continue working in: in cluster number mode (default mode) or in exam mode

• Cluster number mode displays the cluster number + icons (on a normal day) or a warning message about the exam (on the day of the exam) or error messages (inability to receive exam data, expired Secret, low battery). This mode is active 99% of the time.

• The exam mode is activated 1 hour before the exam, shows a warning about the exam starting soon, then switches to a warning about the exam in progress and after the exam is over, switches back to the Cluster Number Mode. This mode is active only on the exam day, 1 hour before the exam + the entire exam time.

• Double checking the exam status flag in this function is necessary to switch from one mode to another. Do not change!

IN CLUSTER NUMBER MODE

1. connecting to a Wi-Fi network

2. checking incoming messages in the Telegram chat

• A new Secret may arrive via Telegram chat, which will be useful later when requesting exam data

3. synchronizing time, date and summer/winter time with the NTP server

• Without time data, reliable operation of the device cannot be ensured. If after several attempts to receive time data it was not possible, an error is displayed on the display and the device itself is put to sleep until the next scheduled awakening. In this case, [DEVICE OPERATION ENDS HERE], until we manage to get the time data during one of the future scheduled awakenings.

4. get exam data for the current day from Intra

• Without exam data, it is impossible to ensure reliable operation of the device. If after several attempts to get exam data it was not possible, an error is displayed on the display and the device itself is put to sleep until the next scheduled awakening. In this case, [DEVICE OPERATION ENDS HERE], until we manage to get exam data during one of the future scheduled awakenings.

• go to the Intra website

• log in to the Intra website

• go to the schedule page for today

• read the received HTML code until we find exam data

• clear the data from unnecessary garbage

• compare the received data with the existing data

[GETTING READY TO MAINTAIN AND DEVELOP THE PROJECT.]

## Needed hardware:

* Computer with any OS,
* USB to USB-C data cable compatible with your computer.

## Needed software:

* Arduino IDE with installed ESP-IDF plug-in,
* Telegram (smartphone app or its desktop version).

## Preparing the software tools

1. Install Arduino IDE and add the ESP-IDF extension. User-friendly instructions on how to do it may be found here:

<https://randomnerdtutorials.com/getting-started-with-esp32/#esp32-arduino-ide>

The online instruction suggests to download and install the latest versions, but the project was built using Arduino IDE version **1.8.19** and the „esp32“ board version **2.0.16**. Compatibility with the later versions was not tested, this is why it is recomended to use this, even though outdated, versions of the tools.

1. Create a folder for Arduino IDE projects. This folder will contain all the projects ever created in Arduino IDE as well as all the installed libraries. The folder may be created anywhere on your computer and may be called any name you give to it.   
   Now, in your Arduino IDE, go to **Arduino > Settings** and at the top of the opened Settings window add the created folder path.
2. Make sure all the required libraries from the „LIBRARIES AND THEIR USE“ list are installed. To do so, in your Arduino IDE, go to **Arduino > Add library > Manage libraries**. In the opened window of the libraries manager you may find all the installed libraries as well as all the available libraries on the Internet.

It is recommended to install the libraries versions stated in the list even though they might be outdated.

1. Set the compilation target. The compiler has to be told what exact model of an ESP32 board to compile for. In case of this project it is „**XIAO\_ESP32C3**“. To do so, in your Arduino IDE, go to **Tools > Boards > ESP32 Arduino > XIAO\_ESP32C3**.

## Opening the project

1. Open the folder for Arduino IDE projects (the one created in step 2 above) in your terminal and use the following command to get yourself a copy of the project:

**git clone https://github.com/RomanAlexandroff/42-Prague-Smart-Sign.git**

1. Open your Arduino IDE and go to **File > Projects > 42-Prague-Smart-Sign > src**. Your project will open.
2. The project comes without any security-sensitive credentials. They may be found printed on the back of the Sign. Rename the „**credentials-example.h**“ file included in the project into „**credentials.h**“ and fill-in the credentials from the Sign.

**DO NOT COMPROMISE THE CONFIDENTIALITY OF THE CREDENTIALS !!!**

## Uploading the changes

1. Connect the Sign to your computer if you have not done so by this time.
2. Activate the software update mode. On the back of the Sign locate button B and button R. Push and hold button B. While holding button B, press button R once. Release button B. Software update mode is now active.
3. In Arduino IDE, go to **Tools** and set the following settings as follows:

* Upload speed: 115200
* CPU Frequency: 160 Mhz
* Flash Frequency: 80 Mhz
* Flash Mode: "QIO"  
   the fastest mode for the flash memory
* Partition Scheme: "Minimal SPIFFS"  
   do not use partition schemes marked "No OTA"
* Core Debug Level: "Verbose"  
   the most detailed debugging output into the Serial monitor
* Erase All Flash Before Sketch Upload: "Disabled"
* Port: choose the development board port.

1. In Arduino IDE, click the Upload button to start uploading.

[PROGRAM FILES DESCRIPTION.]

|  |  |
| --- | --- |
| src.ino | Main file. |
| 42-Prague-Smart-Sign.h | Main header. It contains the Debug macro, some libraries and the functions declarations. The Debug macro allows to turn the debugging serial output off and on again throughout the whole project, but it does not affect the Core Debug output. Including ota.h at the bottom of the file is not coinsidential – it has to be kept below the OTA functions declarations for the OTA update to work. |
| battery\_management.cpp | Initialize inner ADC module, measure battery voltage level, assign the results to a battery state, act upon the battery state. |
| bitmap\_library.h | Contains all the images to be displayed on the device screen in bit-map form. The complete images list can be found in the file. |
| buttons\_handling.cpp | Buttons initialisation and interrupt service routines. |
| cluster\_number\_mode.cpp | Everything that the Sign does outside of the exam time. |
| constants.h | Software version number, general constants, buttons and display SPI port configurations, display driver coniguration, images and errors enumerators. |
| credentials.h | Contains project confidential information, such as Intra API authorisation data, Telegram bot token, Wi-Fi access point SSID and password. If instead of credentials.h there is only credentials-example.h, in the **[GETTING READY TO MAINTAIN AND DEVELOP THE PROJECT.]** chapter, please, navigate to „Opening the project“, point 3. |
| display\_handling.cpp |  |
| exam\_mode.cpp |  |
| file\_system.cpp |  |
| globals.h |  |
| globals.cpp |  |
| intra\_interaction.cpp |  |
| ota.h |  |
| other.cpp |  |
| power\_down\_recovery.cpp |  |
| telegram\_bot.cpp |  |
| telegram\_compose\_message.cpp |  |
| time\_utilities.cpp |  |

[FUNCTIONS DESCRIPTION.]

|  |  |
| --- | --- |
|  |  |
| ERROR\_t ft\_fetch\_exams(void); |  |
|  |  |
|  |  |
| void ft\_battery\_check(void); |  |
| void ft\_battery\_init(void); |  |
|  |  |
|  |  |
| void ft\_buttons\_init(void); |  |
| void IRAM\_ATTR isr\_diagnostics(void); |  |
| void IRAM\_ATTR isr\_ota(void); |  |
| void IRAM\_ATTR isr\_warning(void); |  |
|  |  |
|  |  |
| void ft\_cluster\_number\_mode(unsigned int\* p\_sleep\_length); |  |
|  |  |
|  |  |
| void ft\_draw\_colour\_bitmap(const unsigned char\* black\_image, const unsigned char\* red\_image); |  |
| void IRAM\_ATTR ft\_display\_cluster\_number(IMAGE\_t mode); |  |
| void ft\_clear\_display(void); |  |
| void IRAM\_ATTR ft\_display\_init(void); |  |
|  |  |
| /\* exam\_mode.cpp \*/ |  |
| void ft\_exam\_mode(void); |  |
|  |  |
| /\* file\_system.cpp \*/ |  |
| ERROR\_t ft\_secret\_verification(String text); |  |
| credentials.h |  |
| display\_handling.cpp |  |
| exam\_mode.cpp |  |
| file\_system.cpp |  |
| globals.h |  |
| globals.cpp |  |
| intra\_interaction.cpp |  |
| ota.h |  |
| other.cpp |  |
| power\_down\_recovery.cpp |  |
| telegram\_bot.cpp |  |
| telegram\_compose\_message.cpp |  |
| time\_utilities.cpp |  |

[How to Get Exams Info from Intra.]

The Smart Sign does it in the following 6 steps:

1. connects to Wi-Fi,
2. connects to the 42 Intra server,
3. asks the server for a temporary access token using the UID and the Secret,
4. retreives the temporary access token from the server response,
5. asks the server for exam information for a particular campus, a particular cluster, on a particalar date,
6. retreives the exam information from the server response.

For testing purposes, this process can be recreated on a computer, in Terminal using Curl:

1. enter these variables into the Terminal

**CLIENT\_ID=**put\_your\_42\_API\_app\_UID\_number\_here

**SECRET\_ID=**put\_your\_42\_API\_app\_Secret\_token\_here

2. ask the 42 Intra server for a temporary access token

**curl -X POST --data "grant\_type=client\_credentials&client\_id=${CLIENT\_ID}&client\_secret=${SECRET\_ID}" https://api.intra.42.fr/oauth/token**

3. copy the access token from the server response and enter it as a variable into the Terminal

**TKN=**put\_received\_access\_token\_here

4. ask the server to send you the information about exams in the cluster C3 and put it into a .json file. 56 is the ID of the 42 Prague campus. Curl does not like square brackets [ ] in its calls, so they need to be escaped with a backslash \.

**curl -H "Authorization: Bearer $TKN" "https://api.intra.42.fr/v2/campus/56/exams&filter\[location\]=C3" > c3\_exams.json**

If you want to filter the results down to the exact date, as the Smart Sign does, use the following call instead.

**curl -H "Authorization: Bearer $TKN" "https://api.intra.42.fr/v2/campus/56/exams?filter\[location\]=C3&range\[begin\_at\]=2024-07-12T05:00:00.000Z,2024-07-12T22:00:00.000Z" > c3\_exams1.json**

5. this command opens the .json file in the Terminal

**python -m json.tool < prague\_exams.json | grep "begin\_at" | tr -d " ," | awk -F '"begin\_at":' '{print("["++count"]:", $2)}'**

[example of the 42 server access token response as the smart sign sees it.]

HTTP/2 200

date: Thu, 11 Jul 2024 13:19:37 GMT

content-type: application/json; charset=utf-8

cache-control: no-store

etag: W/"77a2df7a4e20f5f76e6364d36bc76e8a"

pragma: no-cache

set-cookie: \_mkra\_stck=15e20a8020c702e70007eb1e185a06fb%3A1720703982.2018037; path=/; max-age=10; expires=Thu, 11 Jul 2024 13:19:47 -0000; HttpOnly

status: 200 OK

vary: Origin,Accept-Encoding

x-rack-cors: preflight-hit; no-origin

x-request-id: 3d153728-82b5-48a0-84e7-7c1f1efe598a

x-runtime: 0.076367

cf-cache-status: DYNAMIC

report-to: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v4?s=5%2Bb21KLqrzLETXPtKW2gerMAMrEPjiLAWT6eRUKeyuOVy3b5pvEr6Tc7D%2BMB%2BB4gqUHrTyXWaYy01CmZjQqUGReP7COyDKfBhKpl75Kwd%2FWrMWCVZD%2FkWhvM1iHF0V43hw%3D%3D"}],"group":"cf-nel","max\_age":604800}

nel: {"success\_fraction":0,"report\_to":"cf-nel","max\_age":604800}

server: cloudflare

cf-ray: 8a191610cec6bc03-FRA

{"access\_token":"03e4cb9b861dad6c49f2267cf97bd18a942507efa7840dc971008d264596cf89","token\_type":"bearer","expires\_in":6564,"scope":"public","created\_at":1720703340,"secret\_valid\_until":1722585613}

[example of the 42 server EXAM INFORMATION response as the smart sign sees it.]

[HOW TO DRAW ON THE DISPLAY.]

[SERVICE MESSAGES MEANING.]

[LIBRARIES AND THEIR USE.]

The project was built in Arduino IDE 1.8.19.

It uses board 'esp32' version 2.0.16

The libraries in bold are explicitly included in the project.

|  |  |  |
| --- | --- | --- |
| **Arduino.h** |  | String variables manipulations |
| **LittleFS** | 2.0.0 | stores data even without electricity (Telegram chat number, Secret, OTA flag value) |
| FS | 2.0.0 | dependency for the LittleFS library |
| **ArduinoOTA** | 2.0.0 | for the Over The Air update functionality |
| **WiFiUdp** | 2.0.0 | dependency for the ArduinoOTA library |
| **ESPmDNS** | 2.0.0 | dependency for the ArduinoOTA library |
| Update | 2.0.0 | dependency for the ArduinoOTA library |
| **time.h** |  | gets NTP Server date and time; deciphers UNIX timestamp for the SECRET expiration date |
| **stdio.h** |  | provides printf() function for the DEBUG macro |
| **stdint.h** |  | provides fixed-width integer types |
| **esp\_system.h** |  | allows to use ESP-IDF native functions |
| **esp\_sleep.h** |  | allows to use the Deep Sleep power-saving functionality |
| **driver/adc.h** |  | for battery charge measurements |
| **Wire** | 2.0.0 | for SPI reconfiguration in the ft\_display\_init function |
| SPI | 2.0.0 | dependency for the Wire library |
| **GxEPD2\_3C** | 1.5.2 | 3-coloured version of the GxEPD2 library for e-paper displays |
| **GxEPD2\_BW** | 1.5.2 | dependency for the GxEPD2\_3C library |
| Adafruit\_GFX\_Library | 1.11.8 | dependency for the GxEPD2\_3C library |
| Adafruit\_BusIO | 1.14.4 | dependency for the GxEPD2\_3C library |
| **Fonts/FreeSansBold24pt7b.h** |  | the fonts come from the Adafruit GFX library which gets called by the GxEPD2 library |
| **WiFi** | 2.0.0 | for Wi-Fi functionality |
| **WiFiClientSecure** | 2.0.0 | for secure HTTPS requests |
| **UniversalTelegramBot** | 1.3.0 | Telegram bot; for wireless SECRET update and low battery notifications |
| ArduinoJson | 6.21.3 | dependency for the UniversalTelegramBot library |

[BUGS AND SUGGESTIONS HOW TO FIX THEM.]

***Display does not work / does not draw an image.***

* you may have miscalculated the image coordinates and the image gets drawn outside of the display field of coordinates. Remember that setRotation also rotates the display field of coordinates. Remember that coordinates always point to the top left corner of an image, but for a text it is the bottom left pixel of the first character in the first line.
* you may have misaligned the image with the partial update window if you are using one. Remember that even when you draw an image in a partial update window, you use the display field of coordinates to place it; a partial update window does not have its own field of coordinates.
* the display driver memory may be full. Run your program and open the Serial monitor. When an image gets drawn on the display, in the Serial monitor it says “Updating xxxxxxxxxx” where xxxxxxxxxxxx is a very long number that can be different each program run. Among all the outputted messages find the “Updating” message that corresponds to your image being drawn. Now look higher and find the “Power on” message. Keep looking higher and find the “Power off” message, it should not be separated from your “Updating” message by other “Updating” messages. If you cannot find it, it means that this is a bug. To fix it, in your code use the function to force the display to power off before drawing the image. The display will power back on automatically.

***DEBUG\_PRINTF causes a compilation error.***

* the DEBUG\_PRINTF macro is designed in such a way, that every time it is called, it shall be given exactly 2 arguments. If you need to output only one argument, simply send an empty string as the second argument. An empty string looks like quotation marks with nothing between them (“”). Look for examples in the program code.

***DEBUG\_PRINTF does not output a message.***

* the DEBUG\_PRINTF macro cannot output String type variables natively. To do that, you need to explicitly cast the String variable into the C-style string with c\_str() command. Look for examples in the program code.

***Serial monitor is empty / outputs gibberish.***

* check the DEBUG macro in the 42-Prague-Smart-Sign.h header file. The DEBUG definition should not be commented out for the Serial output to work. Additionally, you can set the Core Debug Level to "Verbose" in the Arduino IDE Tools to get detailed information about the firmware processes.
* make sure that the baud rate in the Serial monitor is set to the same baud rate as in the ft\_serial\_init() function of the program.
* you may encounter such behaviour right after the software update. It is normal. Try closing and opening again the Serial monitor window. If that does not help, push the Reset ("R") button on the module.

***Serial monitor skips some messages / does not show some messages.***

* it is a common situation at the beginning of the program. Serial communication between the computer and the microcontroller needs time to stabilise and synchronise itself. ESP32-C3 USB Serial is especially prone to this issue. To overcome it, increase the delay inside of ft\_serial\_init() or add a few empty messages to be outputted after the Serial.begin() command. You may well try to implement both of the suggested solutions at the same time.

***Wi-Fi does not connect / reconnect without apparent reason.***

* thoroughly check your network SSID and password spelling. Surprisingly, it is a very widely spread cause. A single character written small instead of capital may easily prevent you from connecting.
* make sure not to use ft\_delay() in any of your functions responsible for connecting or reconnecting to Wi-Fi. The ft\_delay() function not only delays the program execution but also puts the microcontroller's inner Wi-Fi module to sleep. Using ft\_delay() in functions responsible for retrieving information from the Internet may result in unexpected behaviour. If you are not sure that using ft\_delay() is safe in your particular function, use delay() instead.

***OTA does not work. Cannot see the device in the ports list.***

* make sure that the Sign and your computer are connected to the same Wi-Fi network.
* try closing and reopening Arduino IDE.
* the school firewall may be blocking OTA connection. Ask your campus system administrator if it could be overcome.

***Adding multiple Strings together with the “+” command causes compilation error.***

* strangely, sometimes the compiler may not like it in one part of the code and be completely fine with it in another. The solution is to explicitly cast the variable after the first “+” command into String with the String(your\_variable\_or\_text) command. Understandably, it is strange to cast a String variable into String, but it works.

***WARNING: Skipping SSL Verification. INSECURE!***

* not a bug.
* this message appears when connecting to the Intra server and is caused by the following line in the intra\_interaction.cpp file: „client1.setInsecure();“.
* one one hand, it can be solved by getting and setting up a certificate for this connection. On the other hand, it does not affect the program run at all and can be ignored.

***setSocketOption(): fail on 0, errno: 9, "Bad file number"***

* a minor issue and does not necessarily indicate a problem with the program.
* this message may appear when the Smart Sign fails the first attempt to get a server response from the Intra server and goes for the second or third attempt.
* this error can occur when you try to set a socket option on a socket that has already been closed or is in the process of being closed. This can happen during the transition between closing the previous connection and opening a new one. As long as the SSL/TLS communication with the Intra server is functioning correctly after the reconnection, this error can generally be ignored.

***spiAttachMISO(): SPI Does not have default pins on ESP32C3!***

* not a bug.
* This message appears when the microcontroller assigns pins for the display SPI port. In this project we do not use the MISO pin (thus the „-1“ value defined for the SPI\_MISO\_PIN in the constants.h file).

***401 Unauthorized. Error! Server response came without the Access Token.***

* often happens when something is wrong with the Secret token authentication, commonly with the Secret token itself. Most likely, an extra character was added to your Secret token somewhere along the way. The character may even not to be visible in the Serial monitor. It may happen when you write to or read from the filesystem files. Try using trim() on the variable (e.g. your\_string\_variable.trim();), it will remove spaces and/or new line signs at the beginning and at the end of the string.
* rarely may happen due to the Intra server maintenance. There is no solution to it but to wait.

***Compilation error: “Section .dram0.bs 'Will Not Fit In Region Dram0\_0\_seg' Region.`Dram0\_0\_seg 'Overflowed by 9648 Bytes. Collect2: Error: LD Returned 1 Exit Status”***

* it means that the program takes more RAM space than it is available. DRAM stands for Data Random Access Memory and is used for data.
* This error may be caused for example by excessive use of global variables, large arrays, big buffers, etc.
* The most likely reason for this error in this project is the display buffer being too big. The ESP32 and the ESP32-S2 are especially prone to this issue. To overcome this problem the display buffer size should be reduced. It can be done in the display instantiation. Full display buffer instantiation: GxEPD2\_3C<GxEPD2\_750c\_Z08, GxEPD2\_750c\_Z08::HEIGHT> display(GxEPD2\_750c\_Z08(SPI\_SS\_PIN, DC\_PIN, RST\_PIN, BUSY\_PIN)); Reduced display buffer instantiation: GxEPD2\_3C<GxEPD2\_750c\_Z08, GxEPD2\_750c\_Z08::HEIGHT/2> display(GxEPD2\_750c\_Z08(SPI\_SS\_PIN, DC\_PIN, RST\_PIN, BUSY\_PIN));
* Less likely reason for this error is an excessive use of the global scope for data. To solve it reduce the number of global variables, use the file system to store the data instead of arrays.
* Software update fails while trying to connect to the microcontroller board
* Go into the "Tools" menu and change the Upload Speed to 115200. Sometimes the IDE automatically sets the Upload Speed to the highest value and your board may happen not to support it.

[External Information Sources.]

Документация к данным NTP сервера: <https://cplusplus.com/reference/ctime/tm/>

Страница дисплея **GDEY075Z08** <https://www.laskakit.cz/good-display-gdey075z08-7-5--800x480-epaper-cerveny-displej/>

Даташит дисплея **GDEY075Z08** <https://www.laskakit.cz/user/related_files/gdey075z08.pdf>  Пример кода для дисплея от LaskaKit <https://github.com/LaskaKit/Testcode_examples/blob/main/Displays/E-Paper/7-50/GDEY075Z08_GxEPD2/GDEY075Z08_GxEPD2.ino>

Драйвер дисплея называется UC8179, вот даташит <https://www.laskakit.cz/user/related_files/uc8179.pdf>

Библиотека для использования дисплея называется **GxEPD2**, вот страница <https://github.com/ZinggJM/GxEPD2>

Форум с вопросами по фичам и багам библиотеки **GxEPD2** с дисплеем **GDEY075Z08** <https://forum.arduino.cc/t/good-display-epaper-for-arduino/419657>

Страница обсуждения ошибки размера RAM памяти у ESP32 <https://github.com/espressif/arduino-esp32/issues/1163>

Инструкция к **XIAO ESP32C3** <https://wiki.seeedstudio.com/XIAO_ESP32C3_Getting_Started/>

Видео туториал по устранению ошибок с ArduinoOTA <https://www.youtube.com/watch?v=z_btZfxrS48>

Самая детальная инструкция по **использованию кнопок** с ESP32, которая только существует <https://esp32io.com/tutorials/esp32-button>