Energy-efficient Timetable Display for Meeting Rooms using e-Paper Technology and Low-powered Microcontrollers

Cintia Szabó
Babeş-Bolyai University
Cluj-Napoca, Romania
cintia.szabo@stud.ubbcluj.ro

Krisztián-Tamás Antal
Codespring
Cluj-Napoca, Romania
antal.krisztian@codespring.ro

Levente-Zoltán Bartus
Codespring
Cluj-Napoca, Romania
bartus.levente@codespring.ro

Károly Simon
Babeş-Bolyai University
Cluj-Napoca, Romania
ksimon@cs.ubbcluj.ro

***Abstract*—Company meetings have an important role in the decision-making process and the monitoring of daily work activities, having a powerful impact on the future of the organizations. Meeting rooms provide a physical space for these events, however their number is limited, which makes both their booking difficult, as well as keeping track of their schedules more complicated.**

The purpose of this research is to develop a solution for providing easy to access information regarding the status and daily schedule of a particular meeting room. This would help avoid possible overlaps and other inconveniences that can occur if reservations are not tracked accurately. The main goal was to build a device composed of low-cost hardware components that are also energy-efficient, while providing a configurable and easily programmable interface.

Multiple prototypes with different hardware and software components were designed, implemented and compared, leading to a microcontroller powered device, as the final result. It is an easily configurable device that can connect to an internal wireless network, it has an energy-efficient e-Paper display and a rotary encoder for managing user interactions.

In order to simplify the management of multiple rooms and devices, a central server has been created, providing integration with calendar services. The server is also designed to run within the company's internal network, guaranteeing security and possibility for customization.

I. Introduction

The future of a company or an institution is shaped by the decisions that are made during meetings and negotiations, therefore their impact is quite considerable on both short and long term. These events are mostly held in meeting rooms that are limited in number and capacity, meaning that scheduling them for in-person meetings is a demanding task, and must take into account their quantity, capacity and timetable. Staying up-to-date with a specific meeting room's agenda and status can also be a challenging task.

As a potential solution a display could be placed next to each meeting room, providing information regarding the current occupation status of the given room. Furthermore, a timeline could be displayed which is divided into hours of a workday, where the reserved intervals are marked. By providing a clear view of the agenda, these displays could help to avoid common inconveniences such as disturbing an ongoing event.

The aim of this research was to compare hardware and software configurations for developing a device that is cost and energy efficient, and also easily programmable and configurable. Some of these aspects might be conflicting, for instance inexpensive components, such as microcontrollers are more difficult to program due to their limited memory capacity. Therefore, during the development process the goal was to test different components in order to find an optimal balance between the listed objectives.

The device consists of four parts: an operating unit, a display, an adapter [1] for connecting these components and a rotary encoder [2] for managing user interactions. The original idea was to use a Raspberry Pi Zero for the implementation, but due to the global stock shortages, the development started with a Raspberry Pi 4B. That was later replaced with a Raspberry Pi Pico W, which offers reduced power consumption for the price of smaller memory capacity. As for the screen, the e-Paper display [3] is a device designed to save power, since it only reflects the ambient light without having emission of its own, and also retains the projected image even in the absence of a power source [4]. The rotary encoder serves multiple purposes at once. On one hand, it works as a switch between the different inner states of the display. On the other hand, it allows navigation on the display. For instance, in calendar mode the user is able to navigate between days using the encoder. The device communicates with a central server, which provides an abstraction above different calendar services, ensuring a unified interface.

Similar devices already exist, each having their own benefits and drawbacks compared to the currently presented device.

As presented in the comparison table in Fig. I, this solution has a clear advantage when it comes to price, despite the fact that it varies depending on the operating unit. In some cases (Joan 6 [5], Tapirx [7]), a possible security issue is present due to the fact that the server is hosted by an external service provider. This problem is avoided by the currently discussed solution, since each client runs its own server in a separate environment within the internal network of the institution, ensuring security and maintaining a high level of customizability. Another benefit is having direct user-display

	Price	Server	Interactive display
Joan 6 [5]	€349 + subscription	external	yes
SyncSign [6]	€349	internal	no
Tapirx [7]	€259	external	no
Peresented device	€90-150	internal	yes

Fig. 1. Comparison of similar devices

interaction, in the form of using the button, and in Joan 6's case by touching the screen [5]. The rest of the presented devices require the installation of applications in order to establish a connection.

It is also important to mention that despite the fact that the use case discussed in this article is related to company meetings and meeting rooms, the display might be used for other purposes too, such as providing information points in museums or public libraries, conferences or social events held in multiple rooms, etc.

II. Hardware components

As discussed in the previous section, the device consists of several hardware components illustrated in Fig. 2. These components were selected taking into account their availability, maintainability, compatibility, usability and power consumption.

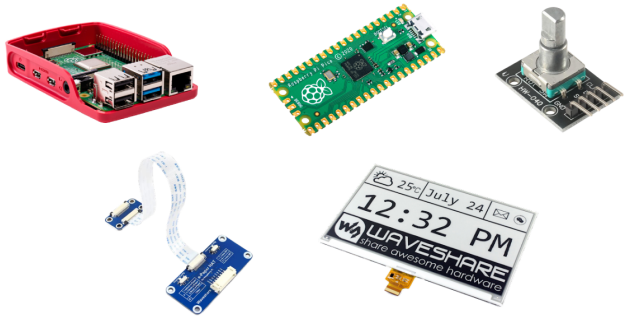


Fig. 2. Hardware components: Raspberry Pi 4B, Raspberry Pi Pico W, KY-040 Rotary Encoder, Waveshare Adapter, Waveshare 7.5 inch e-Paper display

A. E-Paper display

The e-Paper display is a reflective display, which increases its readability and enhances its power efficiency. Therefore it is widely used in public spaces to display information, such as prices and timetables.

The currently used model is Waveshare's 7.5 inch e-Paper display [8] with a resolution of 800x480 pixels. The screen uses e-Ink technology, meaning that it is made up of microcapsules which contain black and white particles [**?**]. Their visibility depends on the positivity and negativity of the capsules' charges. The image is displayed in black and white colours, which is sufficient when it comes to provide comprehensive information about the status of a room [**?**].

Waveshare also provides a native library [8] for all e-Paper models, which is available in Python, MicroPython and C. These libraries offer methods to manipulate the image displayed on the screen.

B. Rotary encoder

The KY-040 encoder [2] is an inexpensive button, supporting two operations – namely, rotation and pressing – through which interactions can be executed.

There are several pre-existing software modules for processing interactions with the encoder, and with a few modifications on their functionalities these modules were able to meet the project's requirements.

C. Raspberry Pi

The Raspberry Pi 4B is a single-board microcomputer. This model contains a quad-core 1.5GHz processor and 1-8GB of memory. It has its own operating system, a Unix-based Raspberry Pi OS developed specifically for Raspberry Pi. Contrarily, the Pico W is a microcontroller, having a more limited memory capacity, with only 264 KB of RAM and 2 MB of flash memory. Unlike a microcomputer, it has no operating system and can only run prewritten code [9].

There is a significant difference between the power consumption of these models. When idle, the 4B model consumes 540 mA, while the Pico consumes 38 mA. Despite the less power consumption, it is more cumbersome to program a Pico, due to the limited memory capacity.

D. Adapter

The adapter [1] enables a connection between the e-Paper and the operating unit. As for the connection with Raspberry Pi models, the adapter's pins are connected to the Raspberry Pi's pins via jumper wires.

III. Implementations for the display software

The purpose of the display is to be a power-efficient, easily accessible and configurable device. In light of these aspects the development process consisted of three different approaches, resulting in three prototypes, and the final version was selected comparing these implementations.

A. Web interface

This version uses Raspberry Pi 4B as the operating unit which is equipped with Raspberry Pi OS. The operating system allows the creation of a web interface running in a browser, which can be transferred onto the e-Paper display. The page and the display are refreshed at certain time intervals.

Upon startup the device enters *kiosk mode* – a state where only a single application is accessible, preventing the user from interacting with any other parts of the system. This functionality is resolved by a shell script that launches the web application, then opens it in a Chromium browser and displays it in full-screen mode. A *systemd service* is responsible for the

automatic start of the application. This service is managed by the service manager of the Linux-based system, and requires a configuration file to run the aforementioned script.

Because of its speed and flexibility, the web application is created using the Vite [10] build tool. The code is written in TypeScript and its components are put together using React libraries, for instance *react-big-calendar*. The results can be seen in Fig. 3.

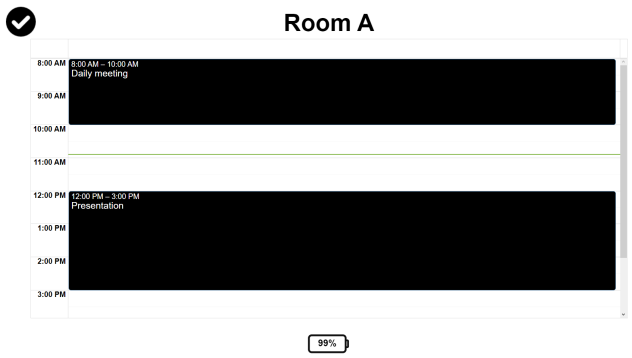


Fig. 3. Web interface

The redirection to the display is done using the *ePaper.js* [11] library, which offers a set of commands for drawing the content of the web page on the e-Paper display.

Developing and maintaining the look and feel of the display is a relatively simple task using these tools and technologies. As a downside, a browser has to be constantly running on the Raspberry Pi, which leads to a higher power consumption.

B. Native libraries

This prototype also uses Raspberry Pi 4B as operating unit. Python is used as programming language, because it provides a number of libraries for simplifying the development process.

Waveshare provides libraries for every e-Paper display written in C and Python – one for each model [8]. Their *epd7in5.py* library contains functions that implement display-manipulating operations. An Image object from the PIL library is the interface for drawing on the display. Thus, a calendar can be created by drawing different shapes, letters and icons on the Image. Although this process provides more flexibility from the perspective of UI design, it is more cumbersome to create a fully functional user interface using this approach.

The rotary encoder is also integrated in this prototype using the pyKY040 module. The display only shows an 8 hour period, but there is a scrolling functionality implemented. Rotating the encoder shifts the displayed interval, or switches to the preceding or the following day.

C. MicroPython with Pico

This prototype is powered by a Raspberry Pi Pico W, meaning that the microcomputer was replaced with a microcontroller. Due to the limited resources of the controller MicroPython is used as programming language, since it provides more control over the memory. MicroPython is a version of Python implemented and optimized specifically for microcontrollers, but it only contains a smaller set of Python libraries [12].

The implementation is based on the previously discussed prototype, the goal being to migrate its Python code to MicroPython and add other functionalities. However, the absence of some libraries makes this process complicated. For instance, there is no alternative to Python's *datetime* library, which requires additional development efforts. The handler of the display and the encoder is also replaced with a combination of different implementations to suit the project's requirements.

1) Event-driven workflow: The set of all possible operations is comprised of the different interactions supported by the encoder: short or long button presses and right or left rotation. The program continuously monitors when such an action is performed and handles the event. The same interaction may lead to the execution of different operations depending on the state of the terminal.

In order to enter the configuration mode, a long key press is required. This involves pushing the encoder for two seconds or more. When the state is activated, an *access point* is created, to which a connection can be established using a laptop or a mobile phone. The name and password of the access point are randomly generated. While in this state, the device hosts a web page (see Fig. 4), that can be accessed at a specified IP address. It provides an interface for configuring the network parameters and the server's URL.

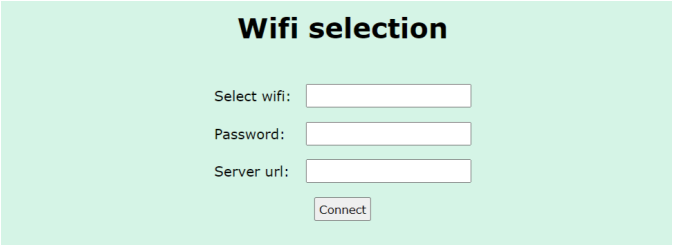
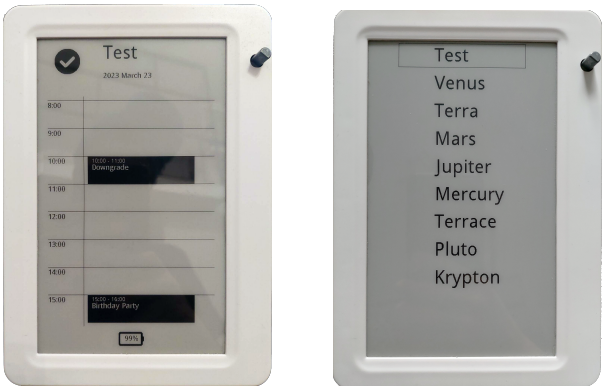


Fig. 4. Configuration interface

By submitting the parameters the system tries to connect to the WiFi network and checks the availability of the server. The result of these operations will be displayed on the screen, i.e. it prints whether the configuration was successful or not, and then exits the state.

After the successful configuration a short press switches to room selection mode (see Fig. 5b) listing the names of the existing rooms. A frame marks the currently selected room, and the selection can be changed by rotating the encoder. With a short press the current selection can be confirmed and the system enters the calendar mode.

The calendar mode is the main state (see Fig. 5a). It shows a timeline divided into hours with rectangles representing the busy periods. The name of the room is at the top of the screen and next to it an icon indicates the current occupancy status.



(a) Calendar mode (b) Room selection mode

Fig. 5. Different states of the terminal

Below the selected date, the corresponding timeline and the battery status is shown.

2) Working with the e-Paper display: The basic functions can set the color of the pixels on the display. Therefore by combining several existing implementations it is possible to define higher level operations, such as drawing lines, rectangles and *.pbm* images. The elements to be drawn are stored in memory and are only displayed once the *show()* method is invoked. Whenever the content is updated the screen flashes black to "clear" itself.

The buffer of the display is a byte array and stores the image to be drawn. For fast and efficient manipulation of the buffer the *memoryview* class is used, which avoids unnecessary data copying and provides direct access to the elements. Since it is a black-and-white display, one bit is sufficient to encode the color of one pixel (zero being white and one being black). Hence, every element of the byte array represents eight pixels.

3) Working with the rotary encoder: The current encoder handler is based on an asynchronous implementation [13] to which the short and long presses are added. Interrupt handlers are added to the three pins of the encoder to handle an event, making the delayed loop execute the appropriate callback functions by setting a *thread safe flag*.

4) Improvements: During the development process several improvements have been made to optimize the implementation, achieving a faster and more stable operation.

The first improvement addresses a performance issue related to the construction of the calendar's image in the buffer (initially lasting 13 seconds) and displaying that image (initially 24 seconds) on the screen. Instead of writing the pixel values one by one, sending them at once reduces the time needed to display the image to 5-6 seconds. Moreover, the rectangles that indicate the occupied periods are fully filled, and in this way all the corresponding bytes can be set at once, reducing the time needed to build the image in the buffer.

Another improvement relates to the memory errors that could occur due to the limited memory capacity of the device. In total there are 153 KB of free memory, of which importing the libraries takes 43 KB and instantiating the encoder and display managers cost an additional 48 KB, leaving only around 60 KB to work with. As a solution a garbage collector is called after every event in order to keep the remaining memory available.

In order to make the best use of the available memory, the modules are imported into the code as precompiled bytecode in the form of *.mpy* files, causing the main program to skip the compilation process when it runs. In this way the compiled bytecode is not stored in the RAM, it is loaded directly from the flash memory.

D. MicroPython with ESP32

Similarly to Pico, the ESP32 is a microcontroller that also integrates Bluetooth and WiFi. Despite the fact that it is three times the price of a Pico, it has a lower power consumption (20-25 mA), 4 MB flash memory, 520 KB RAM, and a processor performance that excels its rival's.

The currently used version has an adapter [14] built into the board making it possible to directly connect to the e-Paper display. At the start of the program it has 101 KB memory available, which is less than Pico's 153 KB. The remaining memory is taken up by the MicroPython language and its libraries.

ESP32 can also be programmed using MicroPython, thus the aim was to create a cross-platform adaptation of the previously described software prototype for both Pico and ESP32. However, ESP and Pico use different versions of MicroPython. After several development iterations, the cross-platform version is still unstable, for instance, the displayed image sometimes fades to a greyish tint, and other problems occur during runtime. Thus, the development and improvement of this solution is still in progress, and at the moment it cannot be considered a working prototype.

E. Comparison between implementations

The implemented prototypes were compared considering the project's main objectives: price-efficiency, energy-efficiency and programmability.

	Web interface	Native	MicroPython
Price-efficiency	●○○	●○○	●●●
Energy-efficiency	●○○	●●○	●●●
Easily programmable	●●●	●●○	●○○

Fig. 6. Comparison table of different implementations

According to the comparison table presented in Fig. 6, when it comes to price- and energy-efficiency, the third prototype seems to be the most suitable one. However, due to the limited resources of the microcontroller, this version required extra attention from the perspective of software implementation. Despite this drawback the aforementioned aspects represent significant advantages, and it is also easier to integrate a smaller microcontroller into the back of the display.

IV. Server

A central server is also part of the project, providing a connection between the calendar service providers and the operating units. As it can be seen on Fig. 7, the server acts as an intermediate layer, which is responsible for processing the incoming requests and sending responses to these requests. It is also responsible for the authentication required to access the calendar services. Responses from different calendar service providers are brought to a uniform structure.

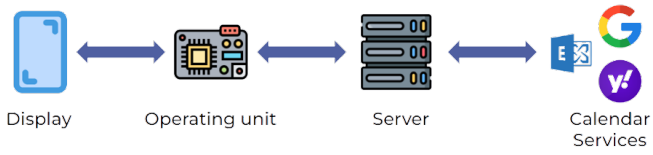


Fig. 7. System architecture

The server is supposed to be hosted within the internal network of a company, so it can be easily configured to suit the company's needs and does not cause security problems.

A. Endpoints

There are two HTTP endpoints, both comply with REST conventions:

- `api/v1/meeting-rooms`
- `api/v1/meeting-rooms/{id}/{date}`

The first endpoint returns a JSON list containing the names and IDs of all the meeting rooms. The second one requires a room ID and a date as path-parameters. The response contains a list of all the reservations made for the given meeting room on that specific day.

B. Structure

The server can be divided into three parts: models, services and controllers. The main entities of the project are rooms and corresponding reservations, represented by model classes. Calendars are represented in the service layer of the server. The layer contains the *ICalendarService* interface which defines a method for returning the reservations of a room. Each implementation corresponds to an existing calendar service provider, which is responsible for establishing a connection in order to access the reservations. Currently there are two implementations: *ExchangeCalendar* and *DummyCalendar*. The latter serves for testing purposes, containing predefined values. The first one is an implementation supporting the Microsofts's Exchange Calendar which is accessed by an internal account and communication is done through the Microsoft Exchange API.

The system is controlled by the *MeetingRoomController* unit, which manages the list of rooms. It acts as a REST API and responds to the incoming requests.

V. Conclusions and Further Development

The purpose of the display presented in this article is to provide information about the reservations of meeting rooms. It is equipped with a rotary encoder to interactively access its functions. It allows the selection of the displayed room and day, as well as the configuration of the internal behaviour. Data for the display is retrieved from a server that runs on the internal network of the company. In addition to the existing functionalities, further requirements can be formulated that would help increase productivity and iron out the imperfections of the project.

Currently only one type of calendar service provider is supported, other frequently used variations, such as Google and Yahoo Calendar should be integrated in the future.

Further development ideas concerning the hardware components offer improvements and other alternatives. One of them being the addition of a battery module, thus making the device portable and independent from the power circuit. Another development idea replaces the e-Paper display, since the currently used version does not support partial refresh of the screen, meaning that even minor changes lead to a full refresh. This process is time- and energy-consuming, which would not be the case with a more advanced display. As previously discussed in section III-D, the cross-platform solution developed for the ESP32 microcontroller powered prototype does not function properly, so it still requires further testing and development.

In addition to viewing the timetable, additional *CRUD* operations could be implemented, providing possibility for the management of the reservations directly from the display terminals.

References

[1] "E-paper driver hat manual." [Online]. Available: https://www.waveshare.com/wiki/E-Paper_Driver_HAT
[2] "Ky-040 rotary encoder user manual." [Online]. Available: https://www.epitran.it/ebayDrive/datasheet/25.pdf
[3] R. Gaddam, "E-paper technology documentation." [Online]. Available: https://www.slideshare.net/RajeshGaddam6/epaper-technology-documentation
[4] A. Joseph, "E-paper technology." [Online]. Available: https://www.ijert.org/research/e-paper-technology-IJERTCONV4IS06009.pdf
[5] "Joan6 official webpage." [Online]. Available: https://getjoan.com/shop/joan-6/
[6] "Syncsign official webpage." [Online]. Available: https://sync-sign.com/product/e-paper-display-7-5-inch/
[7] "Tapirx official webpage." [Online]. Available: https://www.tapirx.com/
[8] "7.5inch e-paper hat manual." [Online]. Available: https://www.waveshare.com/wiki/7.5inch_e-Paper_HAT_Manual#Working_With_Raspberry_Pi
[9] "Raspberry pi documentation." [Online]. Available: https://www.raspberrypi.com/documentation/
[10] "Vite documentation." [Online]. Available: https://vitejs.dev/
[11] "Epaper.js library description." [Online]. Available: https://www.npmjs.com/package/epaperjs
[12] "Micropython documentation." [Online]. Available: https://docs.micropython.org/en/latest/index.html
[13] "Micropython async manual." [Online]. Available: https://github.com/peterhinch/micropython-async/blob/master/v3/docs/TUTORIAL.md
[14] "E-paper esp32 driver board manual." [Online]. Available: https://www.waveshare.com/wiki/E-Paper_ESP32_Driver_Board