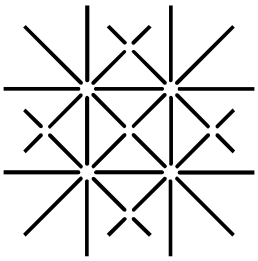

University of Basel
Faculty of Science

Computer Architecture - Project Report:
SMS Machine

Gioia Almer

Mike Müller

Max Reinert

Yanick Spichty

January 21, 2025

Contents

1 Introduction **4**

2 Methodology, Challenges, and Results **5**
2.1 Approach . . . 5
2.2 Complexity . . . 5
2.3 Challenges and Solutions . . . 5
2.3.1 Hardware Conflicts . . . 5
2.3.2 SIM Card Issues . . . 6
2.3.3 SIM Module Connection Problems . . . 6
2.3.4 Short Circuit in the DFPlayer Mini . . . 6
2.4 Final Components Used . . . 6
2.4.1 Arduino Mega 2560 Rev3 . . . 6
2.4.2 SIM Module . . . 6
2.4.3 Keyboard . . . 7
2.4.4 Whadda WPSH412 Touchscreen . . . 7
2.4.5 Piezo Buzzer . . . 7
2.4.6 Other Tools Used . . . 7
2.5 Schematics . . . 9
2.6 Result . . . 10

3 Division of Labor, Key Takeaways and Conclusion **11**

Declaration

We, Gioia Almer, Mike Müller, Max Reinert, and Yanick Spichty, students of Faculty of Science, University of Basel, hereby confirm that this report is entirely our own work. All figures, tables, equations, code snippets, artworks, and illustrations included in this report are original unless explicitly acknowledged, quoted, or properly referenced.
We understand that failure to acknowledge the contributions of others constitutes plagiarism, which is a serious form of academic misconduct and will be subject to appropriate penalties.
Furthermore, we consent to this report being shared with future students as an example of academic work. We also agree to make this report available to members of the University of Basel and the wider public who have an interest in teaching, learning, and research.

Gioia Almer, Mike Müller,
Max Reinert, Yanick Spichty
January 21, 2025

Abstract

This report details the development and successful implementation of an Arduino-based SMS communication system, created for the "Computer Architecture" course at the University of Basel during the Fall Semester 2024. The goal was to design a functional device similar to modern messaging apps, enabling users to manage multiple SMS chats with stored contacts, view chat history, and send or receive new messages seamlessly. The system provides an intuitive user interface on a 2.8-inch touchscreen display, allowing users to navigate between conversations and interact with their contacts efficiently. The touchscreen also features an integrated microSD slot for saving chat history, ensuring that previous messages are retained.
The project utilizes an Arduino Mega for its processing power and extensive pin support, a SIM7600G-H module for SMS network connectivity, and a Piezo Buzzer for audio feedback. A PS/2 keyboard was added to allow fast and efficient message input. The resulting system successfully fulfills all the defined objectives. It allows users to store contacts, navigate between active chats, view received messages in real time, and send new SMS messages reliably and without issues.
During development, the team faced hardware conflicts between the SIM module and other components, interrupt management issues, and debugging challenges. These were resolved through pin reassignments and iterative software troubleshooting. The final device integrates all components into a fully functional communication system, providing users with a practical and reliable solution for managing SMS communication.

1 Introduction

The objective of this project was to gain hands-on experience with hardware and explore the complexity of embedded systems by designing and implementing a functional SMS communication device. The project aimed to apply concepts introduced during lectures and demonstrate the capabilities of a team of four.
The system was inspired by modern messaging applications and allows users to store contacts, manage multiple SMS chats, send and receive messages, and save chat histories. It combines various hardware modules into a cohesive system, with a touchscreen providing an intuitive graphical user interface (GUI) for chat navigation. By focusing on practical applications of embedded computing techniques, the project highlights the integration of hardware and software to create a complete, reliable communication system.
To achieve these goals, the following key hardware components were used:

- **Arduino Mega 2560 Rev3** microcontroller by *Arduino* to coordinate the interaction between the different modules.
- **SIM-7600G-H** by *DFRobot* to connect to a mobile network provided by a SIM card.
- **WPSH412** 2.8-inch touchscreen (320x240 pixels) by *Whadda* to display a GUI of the chats.
- **ID0019A** keyboard by *LogiLink* to input messages.
- **Piezo Buzzer** speaker to implement a notification sound.
- **Breadboard and Jumper Wires** for secure and organized connections.

2 Methodology, Challenges, and Results

2.1 Approach

The project began with a research phase on Arduino-based projects, during which the concept of creating an SMS communication system was developed. This was followed by an initial evaluation and testing of hardware components. While the Arduino Uno was considered initially, its limitations in memory and available pins led to the selection of the Arduino Mega R3, which provided the necessary resources to integrate components such as the SIM7600G-H module, a PS/2 keyboard, a WPSH412 touchscreen, and a buzzer.
The main part of the project focused on implementation, where the system was divided into smaller subprojects. Each component—touchscreen, keyboard, SIM module, and speaker—has been developed and tested independently with several Arduino Uno boards for each of these subprojects before being integrated into the Arduino Mega. Using the Arduino IDE, C++ code was written to program the Arduino and manage component interactions. Iterative testing ensured seamless operation after integration.
To streamline development, the team started by programming individual branches locally and used a GitHub repository for synchronized version control. Initially, the project relied on multiple .ino files. As complexity increased, a modular structure was adopted, with functionalities encapsulated in separate source and header files. This modular design allowed for better management of complex structures and clear coupling between components.

2.2 Complexity

The complexity of the project arose primarily from the integration of multiple hardware components and the challenges of ensuring their seamless functionality. The Arduino Mega's extended memory and additional pins were critical for managing connections between the touchscreen, keyboard, SIM module, and speaker. Careful planning was required to ensure proper power distribution and pin allocation while avoiding conflicts. Another major challenge was the reliance on interrupt handlers, particularly for the keyboard and SIM module, which initially caused instability - caused by the "SoftwareSerial.h" library. These issues were addressed by reorganizing interrupts and optimizing their usage to ensure smooth operation. The PS/2 keyboard added further complexity, as the team manually implemented the protocol to decode raw clock and data signals. This approach allowed for the creation of a custom German keymap, ensuring compatibility with the Arduino Mega.
Configuring the SIM7600G-H module required precise AT commands for 4G SMS communication and extensive research to ensure compatibility with Swiss frequency bands due to limited documentation. Additionally, the touchscreen's microSD slot introduced challenges related to reliable read-and-write operations, which were essential for storing and retrieving chat histories efficiently. Together, these elements highlighted the intricate nature of integrating hardware and software into a functional and user-friendly system.

2.3 Challenges and Solutions

2.3.1 Hardware Conflicts

The team encountered pin conflicts when connecting the touchscreen and SIM module to the Arduino Mega. These conflicts were resolved by reassigning pins and carefully planning the connections to avoid overlap. This ensured all components could operate simultaneously without interference.

2.3.2 SIM Card Issues

The first SIM card we used showed inconsistency in message retrieval and we were able to diagnose memory corruption. We replaced this SIM card with a new one to remedy the issue.

2.3.3 SIM Module Connection Problems

At one point, the SIM module failed to connect to the network. Initially, a temporary workaround involved connecting the module to a computer to reactivate mobile data. The final solution was de- and reactivating network functionality of the module with the following commands:

- `AT+CFUN=0`: Sets the device to minimum functionality.
- `AT+CFUN=1`: Re-enables full functionality, online mode.

Furthermore, we had to ensure better power supply using a different output on the Arduino Mega. This approach successfully restored the module's connectivity.

2.3.4 Short Circuit in the DFPlayer Mini

The DFPlayer Mini overheated during testing due to a suspected short circuit. After diagnosing the issue with a multimeter, the team replaced the DFPlayer Mini with a Piezo Buzzer, which provided a simpler and more reliable solution for audio feedback.

2.4 Final Components Used

2.4.1 Arduino Mega 2560 Rev3

The Arduino Mega serves as the central micro controller, managing all modules and coordinating their interactions. The team selected the Arduino Mega over the Arduino Uno due to its greater memory capacity and additional pins, which were critical for integrating the various components into a single system.

Figure 2.1: Arduino Mega R3

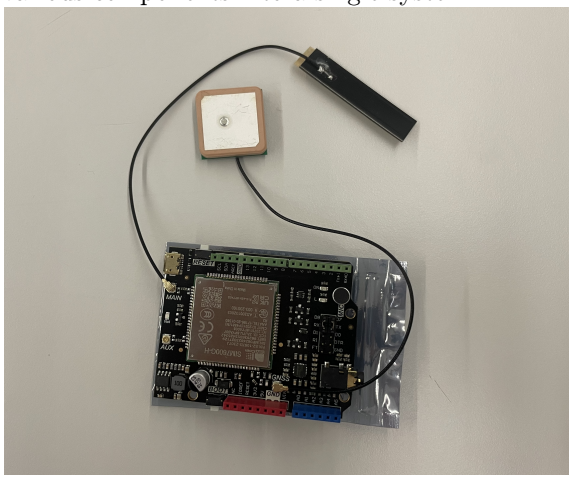

Figure 2.2: SIM7600G-H

2.4.2 SIM Module

The SIM7600G-H CAT4 4G (LTE) shield enables the device to send and receive SMS messages over a mobile network. AT protocol is used to configure the module and establish a network connection. The team selected this module because it supported 4G connectivity, a necessity given the decommissioning of 2G networks and

the reduced availability of 3G networks in Switzerland. The G-H model was chosen for its compatibility with the frequency bands of Swiss providers, although configuring the module required extensive research due to limited documentation. The SIM module facilitates SMS communication through the AT protocol. Upon receiving a new message, an interrupt is triggered to notify the system, allowing it to update the interface dynamically. Messages can be sent and received in real time, providing reliable communication functionality.

2.4.3 Keyboard

The PS/2 keyboard is employed for text input and additional key bindings. It communicates using the PS/2 protocol, an interrupt-driven system natively supported by the Arduino. Instead of relying on external libraries, the team implemented the PS/2 protocol manually to process clock and data signals, facilitating the creation of a custom German keymap. This tailored approach enhances flexibility and ensures seamless compatibility with the Arduino Mega, eliminating the need for additional modules. Each key press is detected via interrupts, setting a flag that enables the touchscreen to process the input efficiently.

Figure 2.3: PS/2 Keyboard

Figure 2.4: Whadda WPSH412 Touchscreen

2.4.4 Whadda WPSH412 Touchscreen

The Whadda WPSH412 touchscreen serves as the primary interface for displaying chats and messages, enabling seamless user interaction. Its integrated microSD slot allows storing and retrieving chat history, ensuring reliable management of multiple conversations.
The graphical user interface was developed using the *Adafruit_GFX* and *MCUFRIEND_kbv* libraries to design buttons and layout elements, organized into the Menu class. The *Touchscreen* library is used to detect touch inputs and identify clickable elements, providing smooth navigation, dynamic updates, and responsive interaction.

2.4.5 Piezo Buzzer

The Piezo Buzzer provides audio feedback for incoming messages. It replaced the DFPlayer Mini, which was damaged during testing, as a more reliable alternative. Whenever a new message is received, the Piezo Buzzer emits a short sound, alerting the user.

2.4.6 Other Tools Used

In addition to the main hardware components, several additional tools are used to assemble and ensure the proper functioning of the system. These tools include:

- A soldering station, along with solder and shrink tubing for creating durable and reliable electrical connections.

- A digital multimeter to measure voltages, test circuits, and diagnose issues during debugging.
- A breadboard, which was used to organize and manage the electrical connections between components in the final system.
- Various cables and resistors, as well as essential tools, as a side cutter and cable stripper, to prepare and connect the wiring.

2.5 Schematics

The schematics below illustrate the wiring and connections of all the components used in the system. This diagram provides a clear reference for reproducing the setup and understanding the hardware interactions. The schematics detail each component's placement and integration, ensuring transparency and ease of replication.

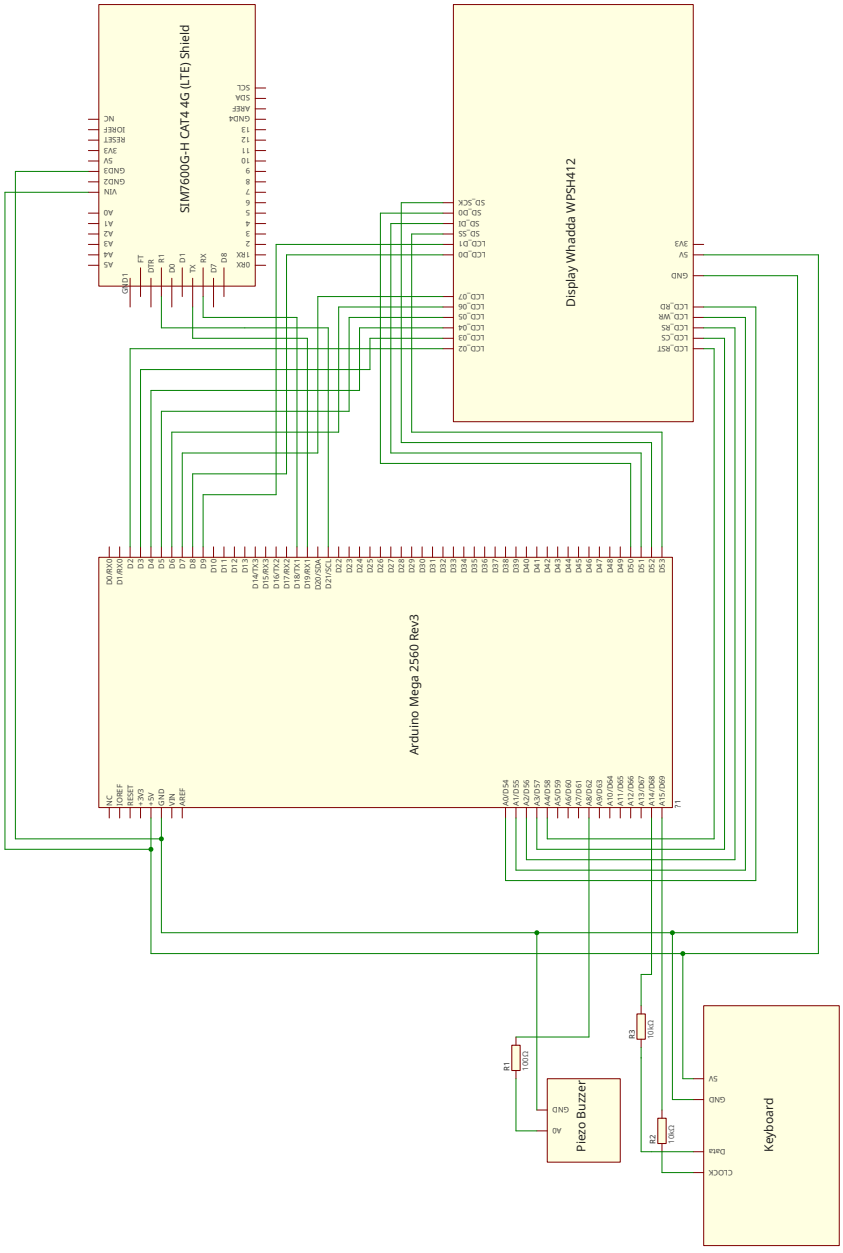

Figure 2.5: System Schematic of the SMS Machine

2.6 Result

Despite facing several challenges and the destruction of multiple components, we successfully achieved our primary goal: building a fully functional SMS machine. This device replicates the functionality of modern messaging applications, allowing users to store contacts, manage multiple chat conversations, and send and receive SMS messages in real time. The chats and messages are displayed on the touchscreen, loading and storing is handled via the displays integrated microSD slot. Additionally, we implemented audio notifications for new messages using the Piezo Buzzer, which can be switched on or off according to user preference. Text input is managed through a PS/2 keyboard with a custom German keymap, ensuring a smooth and user-friendly interaction.
The modularity of our system has been a key strength, allowing for potential future expansions. While we prioritized completing the core features, additional functionalities could be implemented later without significant changes to the existing system.
We even had time to refine certain aspects of the device, such as the implementation of sound alerts, which enhance the user experience. Overall, we finished the project on time and delivered a working system that fulfills the intended goals.

3 Division of Labor, Key Takeaways and Conclusion

The project was a collaborative effort where all team members initially contributed to brainstorming and assessing feasibility. As work progressed, responsibilities were distributed to ensure efficient task management and parallelization. Mike focused on the touchscreen and soldering tasks, ensuring stable hardware connections. Gioia and Yanick collaborated on configuring the SIM7600 module, addressing challenges with AT commands and network compatibility. Max worked on implementing the PS/2 keyboard functionality, including developing a custom German keymap. During the later stages, Mike transitioned to writing the documentation while fulfilling his military service, working on the report in his free time. Meanwhile, Gioia, Yanick, and Max finalized hardware and software integration through iterative testing to ensure seamless operation. In this stage of the project the focus was rather on fulfilling the goals set than division of labor. Hence everyone helped if someone had a problem.

The project provided valuable insights into hardware and software design. The team encountered several challenges, as mentioned earlier, which underscored on the one hand an extensive research and the importance of careful planning and testing hardware setups. Additionally, the bottom-up approach, while effective for developing individual components, created challenges during system integration. In hindsight, adopting a top-down approach with a clear software structure would have streamlined the process.

Ultimately, the project highlighted the importance of detailed planning, collaboration, and iterative testing. Despite challenges, the team successfully delivered a fully functional SMS machine and gained skills that will benefit future endeavors.

References

You can find our official GitHub repository for the source-code at the following link:

https://github.com/yabsp/SMS_Arduino

Guides and Others

- We bought our electronic components off Mouser which you can find via the following link:
  https://www.mouser.ch/
- Guide on how to set up a PS/2 keyboard with a simple LCD:
  https://www.hackster.io/michalin70/
- Guide on how to set up a SIM7600X-X module:
  https://wiki.dfrobot.com/
- Piezo buzzer code for "Never Gonna Give you Up" by Rick Astley:
  https://github.com/robsoncouto/arduino-songs/blob/master/nevergonnagiveyouup/

Use of Libraries

- **Touchscreen** by *Adafruit* as a way to implement the touchscreen.
- **Adafruit_GFX** by *Adafruit* as a uniform library for the GUI buttons.
- **MCUFRIEND_kbv** by *prenticedavid* as an expansion to the GFX library.
- **Arduino** by *Arduino* for external access to the pinouts.
- **stdio, stdlib, and string** by *Arduino* to work with strings.
- **SPI.h** by *Arduino* for communication with the SD card.
- **SdFat.h** by *Bill Greiman* for advanced handling of the SD card.

Use of AI

- **ChatGPT 4o** was used for improving the wording of this report and providing debugging assistance. We confirm that all content is accurate and originates from us.
- **DeepL** was used to assist with translating German to English in this report.