Creating a Homemade Smartphone for Educational Purposes

Ivaniuk Oleksandr
Faculty of Applied Sciences
Ukrainian Catholic University
ivaniuk.pn@ucu.edu.ua

Savorona Kostyantin
Faculty of Applied Sciences
Ukrainian Catholic University
savorona.pn@ucu.edu.ua

Parnosov Nazar
Faculty of Applied Sciences
Ukrainian Catholic University
parnosov.pn@ucu.edu.ua

Humeniuk Denis
Faculty of Applied Sciences
Ukrainian Catholic University
humeniuk.pn@ucu.edu.ua

Haiduk Kostiantyn
Ukraine
ut8iaf@ukr.net

***Abstract*—This document describes our development of a homemade smartphone for educational purposes using the STM32 MCU. We describe the electronic components utilized, the code development process, and the current functionality. Our goal is to present our experience clearly and accessibile for all readers.**

I. Introduction

Communication is critical in times of war. Its organization requires specific skills. Therefore, one of the main needs of the state is to train future specialists who will be able to do this. Since one of the most effective teaching methods is practice, a training device that demonstrates the basic principles of mobile communications and the nuances of its organization may be useful. This project aims to develop a portable learning device that provides basic communication services and allows you to understand which protocols and hardware can be used for organizing the simplest mobile communication. The code for the phone can be found here https://github.com/kostyaCS/Smartphone.git.

II. Materials and Components

Before development, an important stage is the selection of hardware according to the requirements: a device to control the entire system; information should be visualized, but the project does not require a detailed graphical interface; a set of numbers and a few navigation buttons would be enough for input; there should be no noise during the conversation. So, to implement a minimal communication device, we needed a microcontroller, a GSM module, a keyboard, a screen, a microphone, and a speaker.

The following hardware was selected:

- STM32F411E-Discovery,
- GSM Module A9 with a microphone (Fig. 1),
- speaker (Fig. 2),
- 4x4 matrix Keypad (Fig. 3),
- Nokia5110 display, (Fig. 4).

III. User interface

First of all, user interaction with the device requires a graphical interface. Choosing the Nokia5110 LED screen makes it much easier to display information because it is easy to use (with the help of an existing library [3] [4]), does not require color processing (black and white), is energy-efficient, and is compact.

The best way to mathematically describe the operation of the mobile interface is to use the finite state machine principle. There are a finite number of states, from each state you can go to several others. It is logical to start with implementing the menu, for which it was decided to make four icons: call

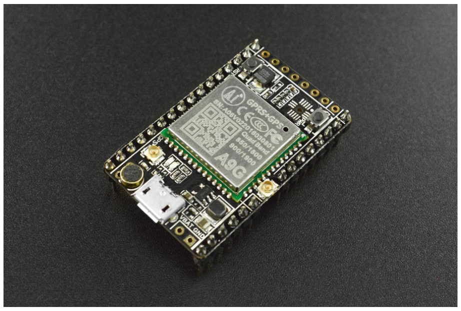

Figure 1: GSM Module A9.

Figure 2: Speaker.

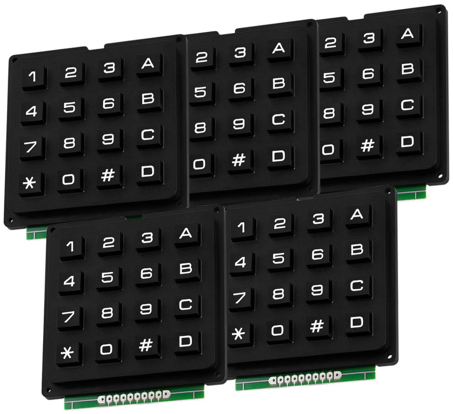

Figure 3: 4*4 Keypads.

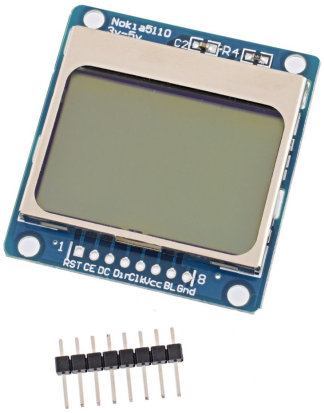

Figure 4: Nokia5110 Screen.

a number, send a message, read a message, and start a snake game. To display our graphic icons, we used online converters to hex representation of the figure and existing library. For such a simple interface, a matrix keyboard with numbers, four letters, and two symbols was more than enough. The transition between screens was usually done through the auxiliary keys of the keyboard, such as ’A’, ’*’, etc.

IV. GSM MODULE

GSM module is a device that connects to the mobile network and can receive and send packets of data via different protocols such as SMPP (Short message peer-to-peer protocol), GPRS (General Packet Radio Service), and voice call-related protocols [1] [2]. Using the STM32, you can send commands to the module and receive responses using the following:

```
GSHAL_UART_Transmit ( ) ;
HAL_UART_Receive ( )
```

functions. The module is connected to the STM via the RX and TX pins. The module accepts the so-called AT commands [6]. A function is written for each operation that performs all the necessary preparation and sending of the command.

For example, here are the most common AT commands:

- **AT**: Check if the modem is responding.
- **AT+CMGF=1**: Set SMS text mode.
- **AT+CMGS=”phone number”**: Send SMS message.
- **AT+CMGR=index**: Read SMS message.
- **AT+CMGL=”status”**: List SMS messages (status can be ”ALL”, ”REC UNREAD”, ”REC READ”, ”STO UNSENT”, ”STO SENT”).
- **AT+CPIN?**: Check if the SIM card is inserted and ask for a PIN if necessary.
- **AT+CPIN=”PIN”**: Enter PIN for SIM card.
- **AT+CREG?**: Check network registration status.

V. INTERRUPTS HANDLING

The GSM module supports data transfer via UART, facilitating the transmission of data. Data transfer is implemented through UART interfaces in the project. The function responsible for receiving data operates without prior knowledge of the transmitted message’s length, accepting it one byte at a time. The interrupt function is designed as a small Finite State Machine (FSM) because the GSM module transmits extraneous characters, including excessive \r \n sequences and the command originally sent. The FSM comprises two states: one that disregards garbage and another that accepts information. Upon receiving information, it promptly calls the information processing function before relinquishing control.

To process messages from the module, a function has been written that decides what to do with each of its responses, which is called at each interrupt. Thus, you can monitor the state of the module and create a plan of its actions; for example, accept a call when the module sends ”RING”.

VI. AUDIO CASE AND GAME

There are two GSM modules that can be used: A6 and A9. A9 is simpler to use since it has its own microphone and reduces noise. So all you have to do is connect a speaker to it, and you can get quite good sound quality.

After implementing all the necessary functions of the phone, we decided to make a case for it. First of all, we took measurements, which were used to create a 3D model in Blender. When everything was ready, the separate sides of the case were printed on a 3D printer one by one. In total, it took about 12 hours. When we had all the parts in our hands, we started assembling the phone: we inserted the screen, speaker, and keyboard into the right holes, fixed them, and carefully placed the other electronic components inside. As a result, our phone looks like in Fig. 5.

When thinking about old phones, the first thing that comes to mind about it is the snake game, which used to be a cult. Our phone actually belongs to the old category, so for complete similarity, our phone also has this game.

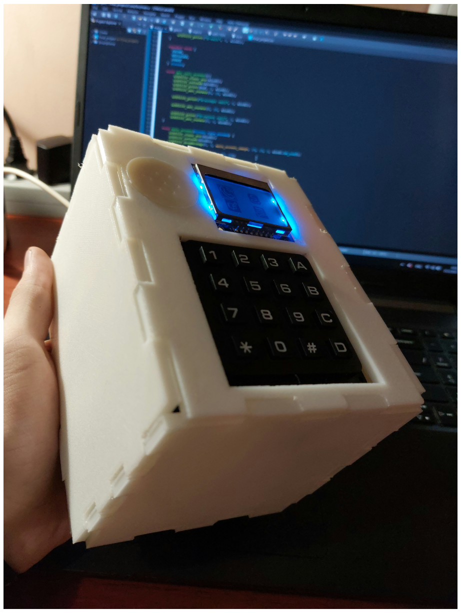

Figure 5: Final phone view-1.

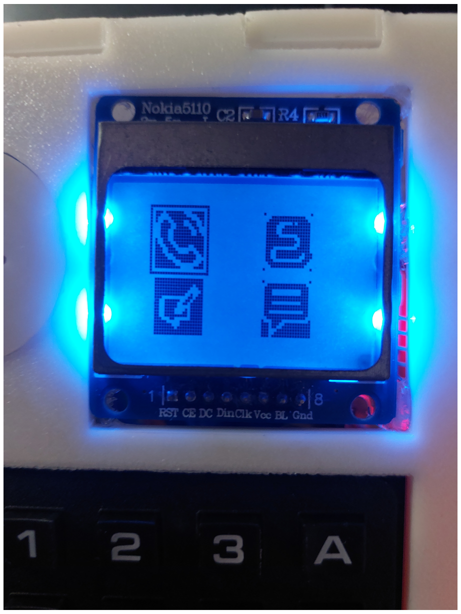

Figure 6: Final phone view-2.

VII. Functionalities and Applications

Our phone has the following functions:

- Make a call.
- Receive incoming calls.
- Send message using two modes: numeric/char.
- Receive incoming messages.
- Play a snake game.

Our phone functionality's finite state machine diagram is shown in Fig. 7. One can see that from all states, our phone can switch to an incoming voice call (via an interrupt).

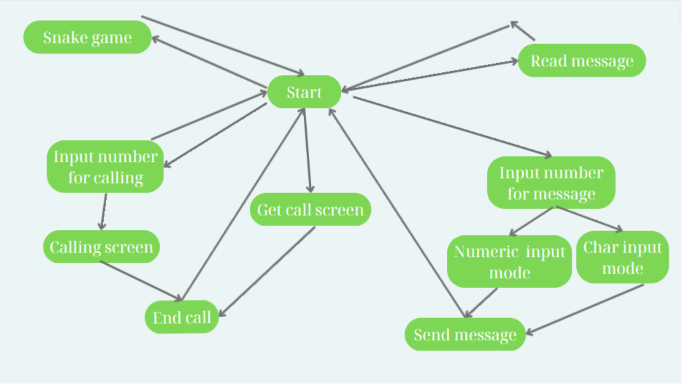

Figure 7: Phone functionality FSM diagram.

VIII. Data flows in our phone

In the center of our system is STM32. All data is going through it. The GSM module implements communication between STM32 and the mobile network, redirects all messages, and receives and executes commands.

Other phone parts:

- Input devices:
  - Microphone (integrated on GSM module, receives sound and transmits it directly on the module while voice calls).
  - Keyboard (sends signals to STM32, which processes it, changes state or gets data, and sends it to GSM module or screen [5]).
- Output devices:
  - Nokia screen (displays all data received from STM32);
  - Speaker (connected to GSM module, plays sound while voice calls).

IX. Problems

While doing our job, we faced some problems, which we will describe below.

We found that a major issue was with the cellular connection of our GSM module. Sometimes, we spent much time looking for bugs in our code, only to realize that messages and calls weren't going through because of connection issues.

Additionally, there were challenges in handling USSD requests to check the balance or retrieve the current time, so we were unable to incorporate these features into our phone, but it is an interesting way to improve the device.

X. Conclusion

Initially, the STM32 was chosen as the main microcontroller, and the GSM module A9 was used to organize communication with the network. Code was written to control it from the microcontroller. Since it is necessary to receive input signals, interrupt processing was organized, which is essential in this type of device. The developed interface using the Nokia 5110 screen and keyboard allows you to use the basic phone functions: sending/receiving calls with the ability to talk and sending/viewing messages, which is necessary for clarity of work. Due to the vulnerability of the communication device to external factors, the following requirements were set: integrity in use and portability. A model of the device was developed, and its case, which contains the cells and power supply, was printed using a 3D printer. Thus, the final result is a portable educational device that provides basic communication services and allows you to understand the basic principles of its operation.

References

[1] AI-Thinker. Gsm module a9 product specification. https://docs.ai-thinker.com/_media/b100ps01a3_a9_product_specification.pdf, 2017.

[2] DFROBOT. Gsm module a9. https://wiki.dfrobot.com/A9G_Module_SKU_TEL0134, 2017.

[3] Oleg Farenyuk. Library for nokia5110 for stm32. http://indrekis2.blogspot.com/2017/01/nokia-5110-pcd8544-stm32.html, 2017.

[4] Oleg Farenyuk. Nokia5110 screen with pcd8544 controller. http://indrekis2.blogspot.com/2017/01/nokia-5110-pcd8544.html, 2017.

[5] MKDas. Interfacing 4×4 matrix keypad with stm32. https://labprojectsbd.com/2023/03/19/interfacing-4x4-matrix-keypad-with-stm32/#Interface_a_4%C3%974_matrix_keypad_with_an_STM32_microcontroller(works), 2023.

[6] Telit. At commands reference guide. https://www.sparkfun.com/datasheets/Cellular%20Modules/AT_Commands_Reference_Guide_r0.pdf, 2006.