KOCAELI UNIVERSITY ENGINEERING FACULTY

ELECTRONICS AND COMMUNICATION ENGINEERING

C# PROGRAMMING PROJECT

SMART ALCOHOLMETER APPLICATION

Yusuf Özcan KARAMAN

160207029

Consultant: Dr. Oğuzhan KARAHAN

KOCAELI 2021

PREFACE

In this study, it is aimed to display the data received from the external environment on the C # interface screen and hardware control with the interface according to the inputs entered from the interface.

First of all, our thesis advisor, we thank Dr. Oğuzhan KARAHAN, the selection stages of the thesis topic and his support make a great contribution to the main structure of the study.

CONTENTS

ABSTRACT

The goal of this research is to show data (card-identification information, fingerprint information, alcohol value, heart pulse value) obtained from external environments on the C # interface screen and control various devices with the interface based on the inputs. The hardware's embedded software is developed in C, with the IAR compiler, while the interface is designed in C # with Visual Studio. Hardware includes a fingerprint reader, heart rate sensor, alcohol measuring sensor, and RFID card reader sensor.

KEYWORDS

mg :	miligram
L :	Litres
V:	Volt
k:	Kilo (103)
Ω:	Ohm (unit of resistance)
° C:	Degrees centigrade,
mA :	Mili ampere
RFID:	Radio Frequency Identification
NFC:	Near Field Communication
MHz:	Mega Hertz
kbit / s :	Kilobite per second
KHz :	Kilo Hertz
DC :	Direct Current
LED :	Light Emitting Diode
Tx:	Transciever
Rx :	Receiver
SS:	Slave Select
CS:	Chip Select
IDE:	Integrated Development Environment
ID:	Identification
API :	Application Programming Interface

USART:	Universal Synchronous/Asynchronous Receiver/Transmitter
UART:	Universal Asynchronous Receiver/Transmitter
DTR :	Data Transfer Rate
OS :	Operating System
uA :	Micro ampere
SPI:	Serial Peripheral Interface
Mm :	Milimeter
Cm:	Cantimeter
DSP:	Digital signal processor
Bps :	Bite per second
MEMS :	Micro Electro Mechanical Systems
DAC :	Digital Analog Convertor
Usb :	Universal Serial Bus
FPU :	Floating Point Unit
Mb :	Megabyte
RAM :	Random Access Memory

TABLES OF FIGURES

1. INTRODUCTION

The Smart Alcoholmeter Application's construction stages will be discussed as part of the project's scope. Section 1: Introduction Created to provide context for the following parts.

1.1 Introduction and Purpose

A breathalyzer application will be built with STM32 and C# as part of the project's scope. The information gathered by the application will be saved in a database. The materials used in the project, as well as the basic project methods, are discussed in the following sections.

2. DESCRIPTION OF THE SENSORS AND MODULES USED

In this section, the modules and circuit elements used for the application will be introduced.

2.1 MQ-3 Alcoholmeter Sensor

The MQ-3 alcohol sensor detects the presence of alcohol gas at concentrations between 0.04mg/L and 4mg/L, making it useful for manufacturing alcohol meters.

This sensor, like other MQ sensors, produces analog voltage based on gas density. It's a sensor that detects the gas range that alcohol meters utilize.

A carrier board is required to use the sensor with microcontrollers such as Arduino. By applying 5V to the H pins, you can assure that the sensor is sufficiently heated to perform precisely and accurately. When you apply 5V to pin A or B, the sounder produces analog voltage on the other pins. You can adjust the sensor's sensitivity by changing the resistance value you link between the ground and the sensor's output pin.

This resistance value may vary according to the application you will make, and you can access the calculations related to it from the datasheet. As an example, you can use a 200 kΩ resistor. Alcoholmeter sensor module is shown in Figure 1.

Şekil 2.1 Alcoholmeter Sensor Module

Features:

- Working Temperature: -10 ° -50 ° C

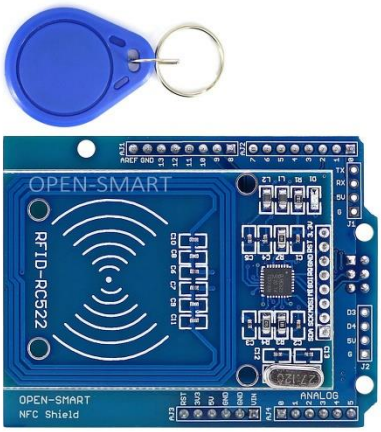
- Working Voltage: 5V
- Current Drawn: 150 mA

2.2 RFID Card Reader

The RC522 RFID card is a low-power, small-size card that can read and write tags that operate at the 13.56 MHz NFC frequency.

It works well with a variety of microcontroller platforms, particularly Arduino. It communicates at a rate of 424 kbits per second. On RFID, it supports a variety of encryption types. It accepts mifare1 S50, mifare1 S70, mifare ultralight, mifare pro, and mifare desfire cards.

It does not support RFID cards with a frequency of 125 KHz. It only works with cards that have a frequency of 13.56 MHz. NFC modules can be utilized with NFC cards because they operate at this frequency. Figure 2 depicts the RFID Card Reader Module.



Şekil 2.2 RC522 RFID Card Reader

Features:

- Operating Frequency: 13,56 MHz

- Working Current: 13-26mA
- Sleep Current: 80uA
- Communication Protocol: SPI
- Supported Cards: mifare1 S50, mifare1 S70 mifare ultralight, mifare pro and mifare desfire
- Card Dimensions: 40x60mm

2.3 Heart Pulse Sensor

It's a pulse sensor that you may use in your Arduino and microcontroller projects. It's easy to use: just attach it to the tip of your finger or your ear and give 3V or 5V to the sensor. You can take a reliable heart rate measurement thanks to the noise reducing amplifier circuit. Figure 3 depicts the pulse sensor.

Şekil 2.3 Heart Pulse Sensor

Specifications:

- Has ~ 60cm male jumper cable
- Can be used to measure heart rate from ear or fingertip
- Diameter: 18.75mm
- Thickness: 3.17mm

2.4 ARD-MDL-1136 Fingerprint Reader Sensor

Grove Fingerprint reader is a very useful product that is easy to use, can be adapted to different platforms. The powerful DSP AS601 chip on it performs operations such as image processing, calculations, fingerprint detection. Up to 162 fingerprints can be stored on the internal flash memory.

A red light is illuminated during fingerprint reading. In this way, you can understand the fingerprint reading process. Fingerprint Reader Sensor is shown in Figure 4.

Şekil 2.4 Fingerprint Sensor Module

Specifications:

- Working Voltage: 3.6 ~ 6.0 V
- Maximum Operating Current: 120 mA
- Fingerprint Reading Time: 1.0 S
- Matching Mode:

- Comparison Mode: 1: 1
- Search Mode: 1: N

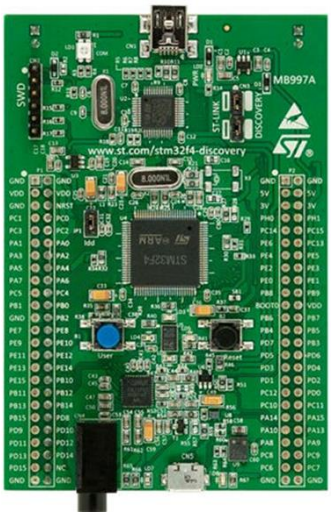
- Memory: 162 Fingerprints
- False Acceptance Rate: 0.001% (Level 3 safe)
- False Reject Rate ： 0% (Level 3 safe)

- Communication Speed : 9600, 19200, 28800, 38400, 57600bps (default is 57600)
- Communication Interface : TTL Serial
- Working Temperature : -20 ~ +50 ℃

2.5 STM32407 Discovery Board

With the high power ARM®Cortex®-M4 32bit processor on the STM32F4 DISCOVERY development board, you can make your projects easier. This card contains everything necessary for beginners or experienced users to get started quickly.

On this development board, ST-LINK / V2 or ST-LINK / V2-A debug tool, two digital ST MEMS 3 axis accelerometers, a digital microphone, an audio DAC with a built-in class D speaker driver, leds and push buttons and 1 usb OTG connector. STM32F407 Discovery Board is shown in Figure 5.



Şekil 1.5 STM32407 Discovery Board

STM32 Discovery Features:

STM32F407VGT6 32bit FPU core ARM Cortex® -M4 microprocessor, 1Mb Flash Memory, 192 kb RAM in LQFP100 package

Built-in ST-LINK / V2 debug tool

ARM® mbed ™ -enabled (http://mbed.org)

USB ST-LINK and 3 different interfaces with renumbering capability:

1- Virtual com port

2- Storage system

3- Debug port

Card Power: Usb bus or 5V DC

Power for External Applications: 3V or 5V DC

LIS302DL or LIS3DSH ST MEMS 3 axis accelerometer

MP45DT02 ST MEMS omnidirectional digital microphone sound sensor

CS43L22 an audio DAC with a built-in class D speaker driver

Eight LEDs:

1- LD1 (red / green) for USB connection

2- LD2 (red) when 3.3V powered

3- 4 user leds; LD3 (orange), LD4 (green), LD5 (red) and LD6 (blue)

4- 2 USB OTG LEDs LD7 (green) VBUS and LD8 (red) over current

Two push buttons (User and reset buttons)

USB OTG FS with micro-AB connectors

LQFP100 I / Os extension headers for prototyping and easy-fast connection

Comprehensive and free software including various examples, STSW-STM32068 or STM32CubeF4 packages for use of built-in libraries

System requirements; Windows® OS (XP, 7, 8,10) and mini USB

Development Tools;

1- IAR® EWARM (IAR Embedded Workbench®)

2- Keil® MDK-ARM ™

3- GCC-based IDEs (free AC6: SW4STM32, Atollic® TrueSTUDIO®, ...)

4- ARM® mbed ™ online

2.6 USB Ttl

The Prolific PL2303 USB-TTL serial converter board is a USB-to-serial converter board that allows you to use the USB unit in the TTL interface.

You can connect any sensor unit or test card with a TTL interface to your computer via the USB port via this cable and communicate between the two units. Since the logic voltage level and supply voltage are 5V, it can be used easily with many systems. One side of the cable has a standard A type USB connector and the other end is a female pin connector. USB TLL is shown in Figure 6.

Şekil 2.6 USB TTL

Features:

- USB-TTL converter board
- PL2303 Integration
- Pins:

  -+ 5V -GND -Tx -Rx

3. SERIAL COMMUNICATION PROTOCOL USED

The Serial Peripheral Interface (SPI) is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems. The interface was developed by Motorola in the mid-1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.

SPI devices communicate in full duplex mode using a master-slave architecture with a single master. The master device originates the frame for reading and writing. Multiple slave-devices are supported through selection with individual slave select (SS), sometimes called chip select (CS), lines.

Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. The SPI may be accurately described as a synchronous serial interface,[1] but it is different from the Synchronous Serial Interface (SSI) protocol, which is also a four-wire synchronous serial communication protocol. The SSI protocol employs differential signaling and provides only a single simplex communication channel. SPI is one master and multi slave communication. Basic SPI bus example is shown in Figure 7. In this project, the communication between the RFID card reader module and the microcontroller was provided by the SPI Protocol.

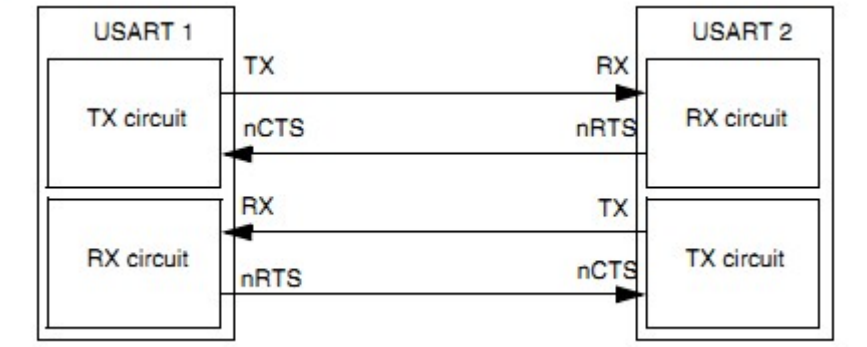
Şekil 3.1 Basic SPI Bus Example

3.1 Universal Synchronous Asynchoronous Receiver Transmitter

USART (Universal Synchronous/Asynchronous Receiver/Transmitter) is a microchip that facilitates communication through a computer's serial port using the RS-232C protocol.

Like a UART (Universal Asynchronous Receiver/Transmitter), a USART provides the computer with the interface necessary for communication with modems and other serial devices. However, unlike a UART, a USART offers the option of synchronous mode. In program-to-program communication, the synchronous mode requires that each end of an exchange respond in turn without initiating a new communication. Asynchronous operation means that a process operates independently of other processes.

Practical differences between synchronous mode (which is possible only with a USART) and asynchronous mode (which is possible with either a UART or a USART) can be outlined as follows: USART Block Diagram is shown in Figure 8.

- 

Şekil 3.2 USART Block Diagram

- Synchronous mode requires both data and a clock. Asynchronous mode requires only data.
- In synchronous mode, the data is transmitted at a fixed rate. In asynchronous mode, the data does not have to be transmitted at a fixed rate.

- Synchronous data is normally transmitted in the form of blocks, while asynchronous data is normally transmitted one byte at a time.
- Synchronous mode allows for a higher DTR (data transfer rate) than asynchronous mode does, if all other factors are held constant.

In this Project the communication between the footprint reader sensor and microcontroller, and the communication between the microcontroller and computer was provided by the USART Protocol (Baudrate:9600).

4. SOFTWARE TOOL

In this section, I will explain to software tool.

3.2 IAR Workbench IDE For Arm 9.20.2

The toolchain IAR Embedded Workbench gives you a complete IDE with everything you need in one single view - ensuring quality, reliability and efficiency in your embedded application. IAR Embedded Workbench is by many considered the best compiler and debugger toolchain in the industry. The necessary codes to program the microcontroller were written in the free trial of IAR Workbench IDE For Arm 8.50.5.

3.3 Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs, as well as websites, web apps, web services and mobile apps. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

5. PRESENTATION OF TOOLBOXES USED

Button: It is the tool used to get any job done.

CheckBox: You can choose this tool when you offer more than one option to the user. For example: If you add a checkbox for Football, Basketball, Volleyball, Swimming, Trekking when you are asked to choose the sports branches you like, you will give the user the chance to select all or not select any of them.

ComboBox: Let's say you want to know the weather forecast of the city you live in that day. All you have to do is to enter the relevant site, select the name of the city and click the show button. For this scenario, you add ComboBox and change its title to "Select City" and when the user clicks it, all cities are listed one after the other.

Label: You can raise awareness of users using this tool. For example, if you add a Label to the left of the TextBox where the user name needs to be entered and edit the "Text" property as "Username:" the user understands what to write to the TextBox. Moreover, while your application is running, you can display the information you want in the Text feature of the Label.

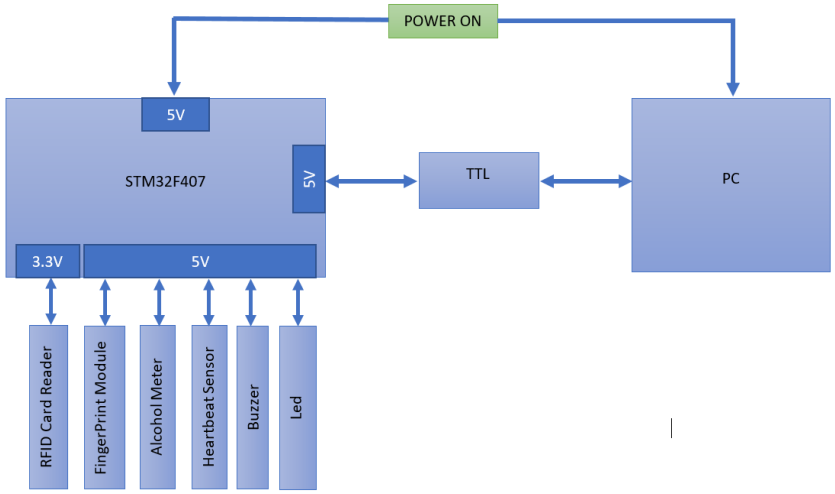
TextBox : Textbox is the control that allows us to get data from the user or enter a value into the system.

6. BLOCK DIAGRAM

The flow charts of the project will be included in the section.

6.1 Electrical Block Diagram

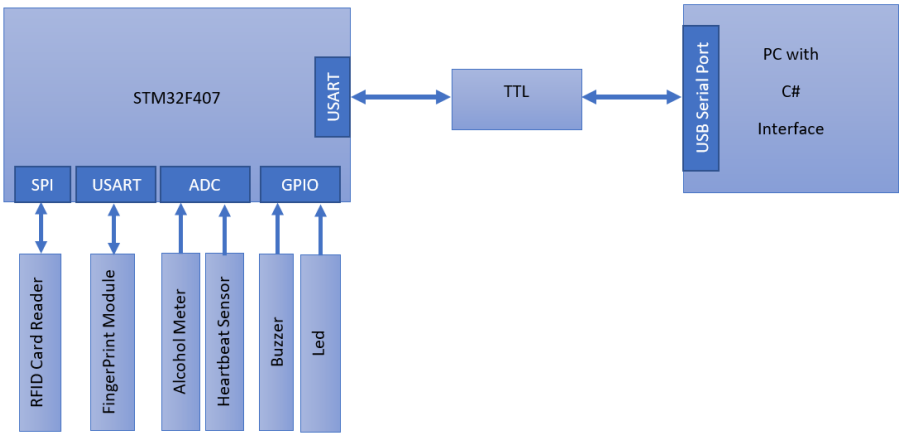
Electrical block diagram shown in the Figure 6.1.



Şekil 6.1 Electrical Block Diagram

6.2 Communicaton Diagram

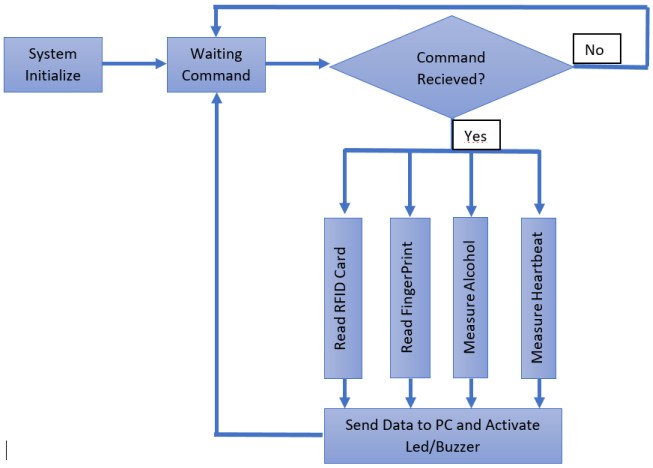
Communication diagram shown in the Figure 6.2



Şekil 6.2 Communication Diagram

6.3 STM32F407VG Software Block Diagram

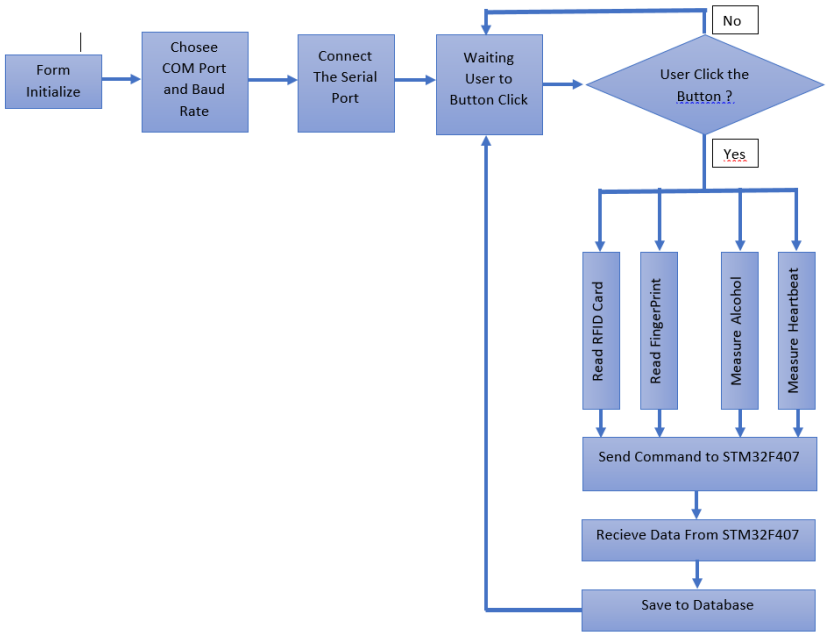
STM32F407VG software block diagram shown in Figure 6.3



Şekil 6.3 STM32F407VG Software Block Diagram

6.4 C# Software Diagram

C# software diagram shown Figure 6.5.



Şekil 6.5 C# Software Diagram

7. CODE EXPLANATION

In this section, the codes used in the application will be explained.

Form1 Code Explanation

First of all, we transferred the data that we received from our microcontroller to our C # project via a serial port connection using TTL. To do this, we selected SerialPort from the Toolbox section of Visual Studio and added it to our project.

Before making a serial port connection, we set the properties of the port we selected, such as port name, connection speed, etc.

```
try
{
    serialPort1.PortName = comboBox1.SelectedItem.ToString();
    serialPort1.BaudRate = num;
    serialPort1.ReadTimeout = 200000;
    serialPort1.WriteTimeout = 200000;


    serialPort1.Open();
```

Figure 7.1 C# SerialPort Configuration

Figure 7.1 C# SerialPort Configuration

Then we checked if the serial port connection was opened. If the connection is successful, a warning message such as “serial port connection has been established” should be written, otherwise an “error” warning will appear.

```
    serialPort1.Open();

    if (!serialPort1.IsOpen)
        serialPort1.Open();
    Thread.Sleep(200);
    serialPort1.BaseStream.Flush();


}
catch (Exception err)
{
    MessageBox.Show(err.Message, "Hata", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
if (serialPort1.IsOpen)
{
    MessageBox.Show("Seri porta bağlantı Sağlandı", "Bildirim", MessageBoxButtons.OKCancel);

}
```

Figure 7.2 C# Serial Port Succes and Error Configuration

After the serial port connection is made, the "serialPort1_DataReceived" function works at each data entry, we assign the incoming data to the "stm32_received_data" variable. Then we call the "F1_show_Form_data" function.

We used the Office Access Database (oledb) Program as a database to save the data. There are 7 columns in our database. These are: Identity ID, RFID card information, person name, Alcohol value, Pulse value and Fingerprint.

Thus, we have successfully transferred the data from our microcontroller to the C # interface.

Form2 Code Explanation

When Form2 is created, we create an integer value called index and set it equal to 0. Depending on this index number, only 1 of the 4 buttons becomes active and we get data from the hardware with that button (for example, for first button index value is 0, we read Card ID) . In addition, in Form2, the name we wrote in Form1 is saved in oledb, and we do every reading process by updating this new data that we save. Form1 and Form2 are shown in Figure 13 and Figure 14.

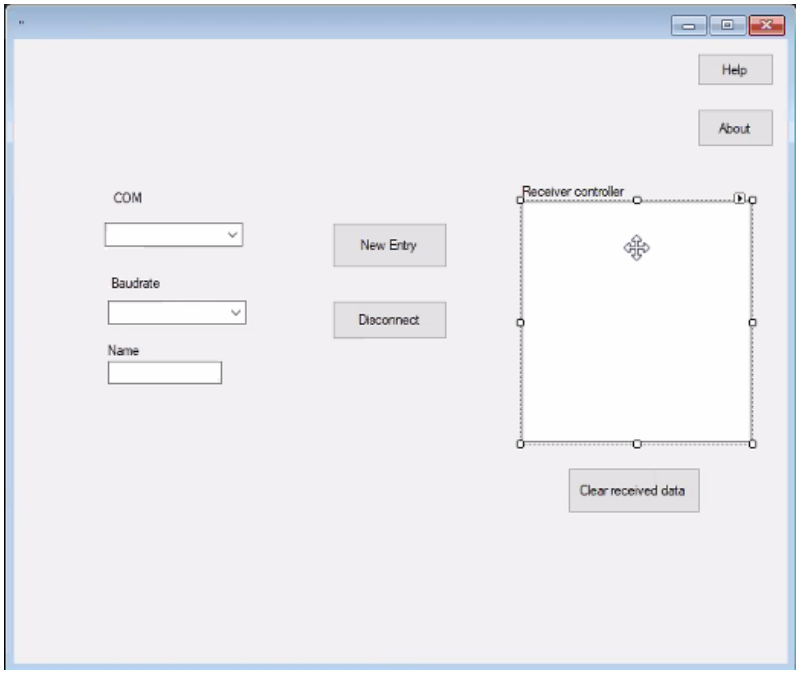


Figure 7.3 Form1

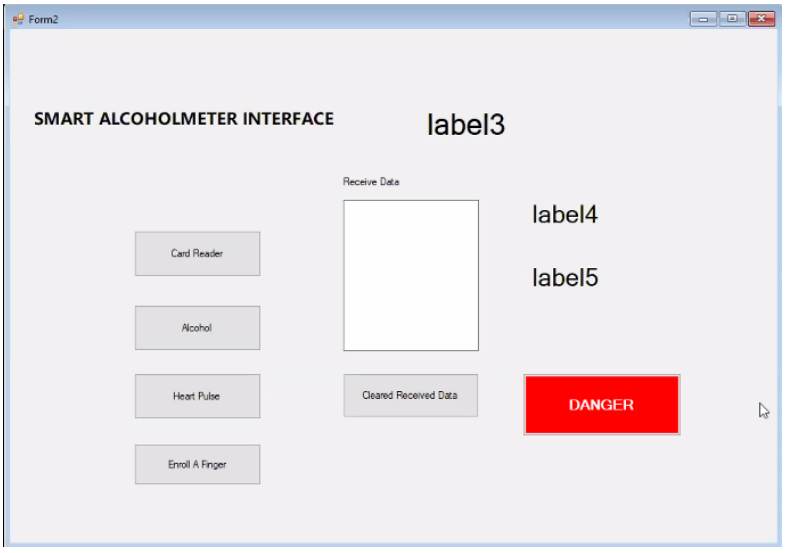


Figure 7.4 Form2

3.4 Interactives

The "new entry" button was clicked on Form1 and Form2 was loaded. The card information of the persons is taken from an external hardware with the RFID module. To receive this data, 1 is sent from the serial port and the STM32 sends the data read from the RFID to the interface via the serial port. The user is asked whether this data is accepted or not. If the user thinks that the read data is not correct and does not accept it, the data is read again, this process occurs again. If the user thinks that the data is correct, the user is asked whether the data will be recorded in the database or not. If the user accepts the data to be saved in the database, the screen writes green OKAY, the buzzer sounds 1 time and updates the database with reference to its name, if the user does not want the data to be saved in the database, the data will not be saved and in both conditions, it will increase the index by 1 and switch to the other data reading operation.

The alcohol button is clicked on the interface, the value 2 goes to STM32 from the interface. Alcohol values of the persons are measured with alcohol meter. The measured value is sent from STM32 to the interface via the serial port. This received value is shown on the screen. If the person using the interface thinks that the alcohol level is high and presses the high button, the red LED lights up, the buzzer sounds a long time and the interface says "high Alcohol value". If the person using the interface thinks that the alcohol level is low and presses the low button, the green LED will turn on, the buzzer will sound once and the interface will write "alcohol free".

The heart rate measurement button is clicked on the interface, the 3 value is sent to STM32 via serial port. Taking 3 values, STM32 performs the pulse measurement and transfers the measured value to the interface with the serial port. The measured value is displayed on the interface. If the person using the interface thinks that the pulse level is low and presses the low button, the green LED turns on, the buzzer sounds once and the interface shows "no risk". If the user thinks that the pulse level is high, he / she clicks the high button and a warning message appears. If the person is thought to be both intoxicated and has a high pulse rate, click the "danger" button and a "risk" warning appears on the interface, the buzzer beeps intermittently for a long time.

The "Enroll fingerprint" button is clicked from the interface and the value 1 is sent to STM32 from the interface, STM32 takes the fingerprint and transfers the data back from the serial port to the interface. If the user thinks that the information is correct, he clicks the OK button and the screen says green OKAY, the buzzer chirps once. If the user thinks that the information is wrong, he clicks the button again and re-reading processes occur. After the information is received, the user is asked whether to register in the database or not. If yes, the data database is updated, and the green LED is lit. If it is said no, the data are not saved.

Embedded C Codes Explanations

First of all, it started from GPIO settings. In our project, 3 communication units, 2 USART and 1 SPI, are used. In addition, 1 pin for the user button, 5 output pins for general purpose use, and 2 pins for the ADC have been set.

General settings for the 2 USART units are made in the USART_Config function. The baudrate of the fingerprint sensor is set to 57600. This setting is adjusted to determine the amount of fingerprints to be stored. USB TTL is used to provide the old communication between the C # interface and the device. TTL baudrate was chosen as 9600.

SPI communication has been selected for RFID. The commands specified in the RFID's datasheet are sent to the device and the card reading functions of the device have been adjusted.

The ADCs to be used have been selected as 12bit, and their measurements with the DMA unit are set to be sent to the C # interface via the serial port. For interrupt setup, the function named NVIC_Config has been used to configure two different interrupts according to their priority settings.

Thanks to a boolean value defined in the interrupt, the transition to the necessary functions was made by checking whether data is received from the C # interface.

1 struct has been defined for the fingerprint sensor. Thanks to Struct, it is provided to add address, password and reading functions to be used in fingerprint. Afterwards, the start

commands were sent in accordance with the datasheet with the USART interface and reading functions were created. While the fingerprint is being read, the sensor is set to send finger data to the microprocessor. The reading function has been created by transmitting the necessary commands with the RFID card reader sensor SPI interface. Alcohol and heart rate values are set to be sent to the C # interface by DMA and ADC sensors.

8.

REFERENCES

http://www.prolific.com.tw/UserFiles/files/ds_pl2303HXD_v1_4_4.pdf

https://www.st.com/resource/en/datasheet/dm00037051.pdf

https://ozgurayik.com/2020/12/12/stm32-programlama-stm32f4-clock-configration/

http://www.lojikprob.com/embedded/stm32/stm32-ilk-program-led-yakma-temel-giris-ve-cikis-ve-truestudio-kullanimi/

http://www.lojikprob.com/embedded/stm32/stm32-ilk-program-led-yakma-temel-giris-ve-cikis-ve-truestudio-kullanimi/