UNSTPB

Faculty of Electronics, Telecommunications and Information Technology

Project 3

Sending a text message from an Android phone, over Bluetooth, to an Arduino board and display the message on an LCD

Students: Meche Mircea-Ionuț, Moreanu Vlad-Ștefan, Opriș Teodora-Alexandra

Project Manager: Dumitrescu Anamaria

ETTI 2024

Contents

List of Figures/Tables

Acronyms

LCD – Liquid Crystal Display

LED – Light Emitting Diode

AT – Attention mode

1. Introduction

The aim of this project is to transmit a message from a smartphone to an Arduino UNO board via Bluetooth and displaying the message on an LCD.

Using MIT App Inventor (or something similar), an Android application will be created which will connect to the Arduino board through Bluetooth. The message will be received as a string and displayed on the LCD. If said string contains a keyword (e.g. ETTI) an LED will be lit for a few seconds.

2. Resources

As hardware, an Arduino UNO board, a Bluetooth Module and an LCD were used. A 10kΩ potentiometer and a 220Ω resistor were used as well.

For the software part, the code for the Arduino UNO board was written in C++, and the Android application was created using MIT App Inventor.

Two Arduino libraries were used when writing the code: SoftwareSerial[1] and LiquidCrystal[2]. The former allows serial communication on other digital pins of an Arduino board, using software to replicate the functionality. The latter allows an Arduino board to control LCDs (communication with alphanumerical liquid crystal displays).

3. Hardware Implementation

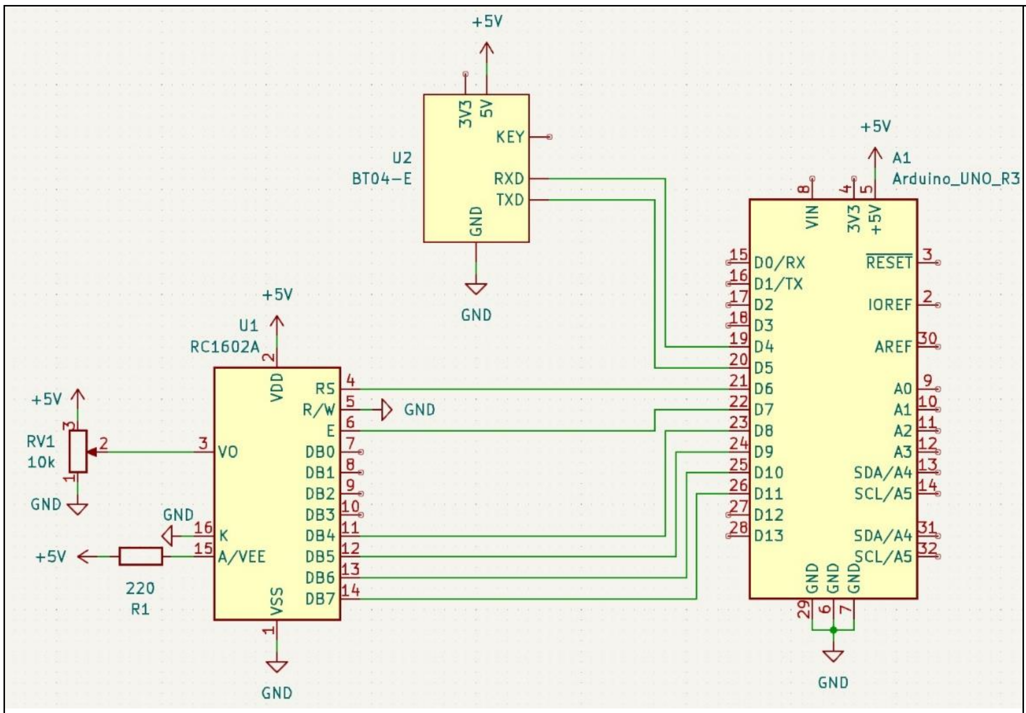


Fig. 3.1 Electric Scheme

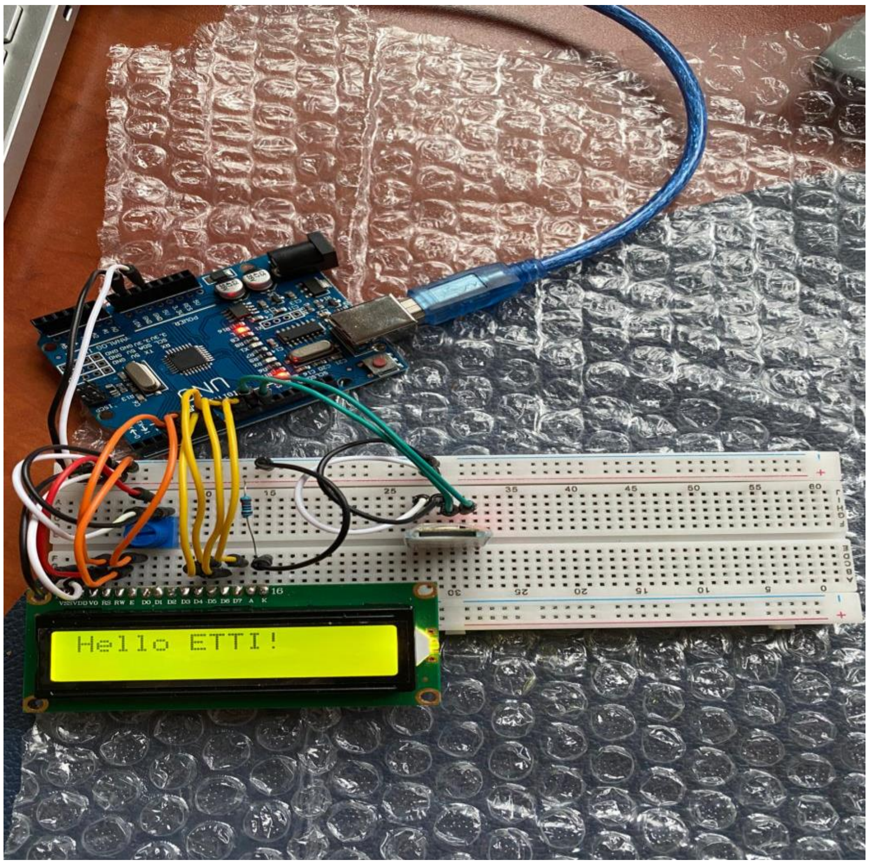


Fig. 3.2 Physical device

The 10kΩ potentiometer RV1 is used to set the contrast on the display. One terminal is connected to the LCD, another to the +5V power supply, and the last one to the ground.

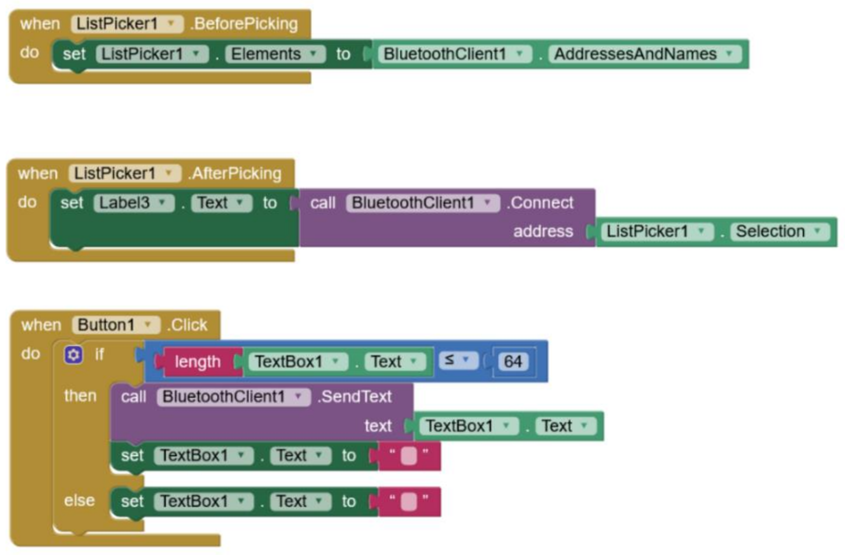
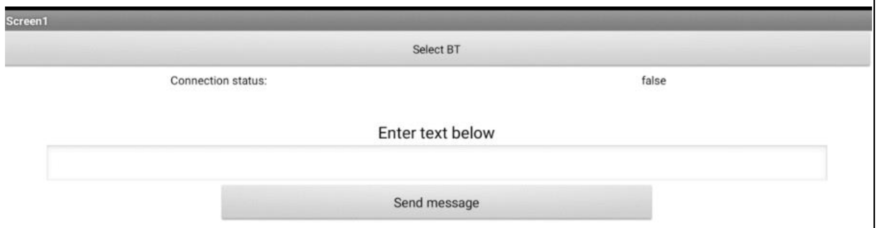
The cathode of the LCD is connected to the ground, and the anode to a 220Ω resistor (R1) which is also connected to the +5V power supply. The LCD module works based on a potential difference on its liquid crystal material, and by connecting its cathode to the ground, a reference for this potential difference is assured. The anode is connected to a resistor in order to limit the current passing through the LCD at a safe level, to ensure its proper functioning and to avoid any damages to the LCD. The other terminal of the resistor is connected to the +5V power supply in order to ensure the necessary voltage along the LCD in order for it to work. This configuration allows the LCD to display characters and symbols.

The D4 digital pin of the Arduino board is connected to the Key pin of the Bluetooth module. The Key pin is used to set the mode of the Bluetooth module (e.g. entering AT mode for configuration).

The D5 digital pin of the Arduino board is connected to the RXD pin of the Bluetooth module. This connection is used for receiving data from the Android phone. When the phone sends a text message over Bluetooth, the data is transmitted through the RXD pin to the Arduino.

The D6 digital pin of the Arduino board is connected to the TXD pin of the Bluetooth module. This connection is used for sending data from the Arduino to the Android phone (e.g. acknowledgements or other information).

4. Software Implementation (Android)

	Fig. 4.1 AppInventor Block Diagram
 Screen1 Select BT Connection status: false Enter text below Send message	Fig. 4.2 In-App View

```
when ListPicker1 .BeforePicking
do  set ListPicker1 . Elements to  BluetoothClient1 . AddressesAndNames

when ListPicker1 .AfterPicking
do  set Label3 . Text to  call BluetoothClient1 .Connect
                                                address  ListPicker1 . Selection

when Button1 .Click
do  if    length TextBox1 . Text  ≤  64
    then  call BluetoothClient1 .SendText
                                  text  TextBox1 . Text
          set TextBox1 . Text to " "
    else  set TextBox1 . Text to " "
```

To start, we connect to the developed application via the AppInventor Companion app. This allows us to directly access the cloud version of the application, regardless of the Android device we use, allowing us to test the portability.

The graphical interface consists of a table of elements: a list, a status indicator for the Bluetooth connection, the text box for user input and associtated button for sending said input.

The user first chooses the device to connect to from a given list (first parent block). If the connection is succesful, the status indicator will change (second parent block). Finally, the user can type a message having a maximum length of 64 characters. If the maximum length is exceeded, the message is dropped. After the message is sent, or if the message was too long and subsequently dropped, the text box will be emptied (third parent block).

5. Software Implementation (Arduino)

The necessary libraries for working with LCDs are included (*SoftwareSerial.h* and *LiquidCrystal.h*). Also, the pins are assigned for the Bluetooth communication (RX and TX), as well as for the LiquidCrystal display(RS, EN, D4, D5, D6 and D7).

The *LiquidCrystal* and *SoftwareSerial* objects are initialized for the LCD and Bluetooth communication.

The arrays *g_Disp* and *g_Buff* are declared and used for storing the display data and the messages received through Bluetooth.

g_NumStr and *g_Scroll* keep track of the number of non-empty lines and the current scrolling position.

The *setup* function initializes the serial communication with the Arduino board and Bluetooth module, it sets up the LCD display and initializes the global variables. It also sends an AT command to set the name of the Bluetooth module (*AT+NAMENBT04-E*), receives the response and displays it on the LCD.

The *loop* function waits for Bluetooth data every 5 seconds. If Bluetooth data is available, it checks for the presence of the substring „ETTI" in the received message and turns on the built-in LED accordingly. It prints the formatted Bluetooth message to the serial monitor, it converts the received message into a display format and prints it on the LCD. It also manages scrolling if more than two lines of text are present.

The *PrintLines* function clears the LCD and prints two lines of text.

The *ConvertBuffer* function takes a plain text buffer and converts it into a format suitable for the display by splitting it into four lines of 16 characters each.

6. Conclusion

The project was overall quite interesting. Although we have had experience working with Arduino Uno / Intel Galielo before duing the Project 2 subject, the result was a self-contained system. Working with a module that allowed for more modern connectivity to an otherwise simple and rudimentary device. One issue we faced with the Arduino development part was the memory management for the text strings done by the Arduino standard library. Developing an Android application was intriguing, though the needed focus on other university tasks, the relative lack of time and lack of experience with Android meant the app is very basic and it wasn't developed using more mainstream methods, such as utilising the Android IDE and Kotlin programming langauge. One issue encountered during the Android development stage was handling the Bluetooth methods, but we eventually decided to handle connectivity inside the app. Nevertheless, it was a fun endeavour and a good introduction to IoT development.

Potential improvements:

1) A faster, more reliable Bluetooth module could be used, or another microcontroller such as the ESP32, to grant the Arduino not only Bluetooth connectivity, but also internet access.
2) More peripherals, such as sensors, could make it into a good first-time smart appliance, capable of executing tasks received in text.

Potential uses:

1) Educational purpose (basics of Android and/or Arduino)
2) Basic IoT appliance (smart lock, weather station etc.)

7. Bibliography

[1] ***. SoftwareSerial - Arduino Reference.

URL:https://docs.arduino.cc/learn/builtin-libraries/software-serial (visited on 10/11/2023).

[2] ***. LiquidCrystal - Arduino Reference.

URL:https://www.arduino.cc/reference/en/libraries/liquidcrystal/ (visited on 07/11/2023).

[3] ***. AppInventor Community Forums.

URL:https://community.appinventor.mit.edu/t/bluetooth-hc-06-arduino-send-receive-send-text-file-multitouch-image/9518 (visited on 25/11/2023).

[4] ***. AppInventor Block Documentation.

URL:https://ai2.appinventor.mit.edu/reference/blocks/ (visited on 25/11/2023)

[5] ***. BT-04E Datasheet.

URL:https://www.smart-prototyping.com/image/data/2020/09/102049%20BT04-E%20BLE4.2%20+%20SPP3.0%20Module%20(FCC,%20CE,%20ROSH)/datasheet.pdf

(visited on 10/11/2023).

8. Annex 1: Arduino Code

```
#include <LiquidCrystal.h>
#include <SoftwareSerial.h>


// PINS
enum { RX = 10, TX};
enum { RS = 4, EN, D4, D5, D6, D7 };


// GLOBAL VARS


LiquidCrystal g_LCD(RS, EN, D4, D5, D6, D7); // Initialize LCD object
SoftwareSerial g_BT(RX, TX); // Initialize software UART object


char g_Disp[4][17]; // 4 lines of 16 characters +1 null terminator
String g_Buff; // Plain text buffer


uint8_t g_NumStr; // Number of non empty lines found in g_Disp
uint8_t g_Scroll; // Line of g_Disp from which the display will start
printing


void PrintLines(char msg0[17], char msg1[17]); // Print two 16 character
lines to the display
void ConvertBuffer(char dst[4][17], const char* buff); // Convert a
string to the display format


void setup() {
  Serial.begin(9600);
  g_BT.begin(9600);
  g_LCD.begin(16, 2);
  g_LCD.clear();
  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, LOW);

  g_NumStr = 0; g_Scroll = 0; g_Buff.reserve(64);

  // Configure name of bluetooth module
  g_BT.print("AT+NAMENBT04-E\r\n");
  delay(2000);
  // Receive and display Name and OK
  if(g_BT.available()) {
    g_Buff = g_BT.readStringUntil('\n');
    g_Buff.trim();
    //Serial.println(g_Buff);
    ConvertBuffer(g_Disp, g_Buff.c_str());
    PrintLines(g_Disp[0], g_Disp[1]);
  }
  else {
    ConvertBuffer(g_Disp, "No response from BT module");
    PrintLines(g_Disp[0], g_Disp[1]);
  }
}

void loop() {
  delay(5000);
  digitalWrite(LED_BUILTIN, LOW);

  if(g_BT.available()) {
    g_Buff = g_BT.readStringUntil('\n');
    if(g_Buff.length() > 63) g_Buff.remove(63);
    g_Buff.trim();

    int8_t found = g_Buff.indexOf("ETTI");
    if(found > -1) {
      Serial.println("-- Found ETTI");
      digitalWrite(LED_BUILTIN, HIGH);
    }

    Serial.println("-- Buffer: ");
    Serial.flush();
    Serial.println(g_Buff);

    ConvertBuffer(g_Disp, g_Buff.c_str());
    Serial.println("-- Formatted message:");
    Serial.flush();
    Serial.println(g_Disp[0]); Serial.println(g_Disp[1]);
    Serial.println(g_Disp[2]); Serial.println(g_Disp[3]);
    Serial.flush();

    g_NumStr = 0;
    g_Scroll = 0;
    if(strlen(g_Disp[0])!=0) g_NumStr++;
    if(strlen(g_Disp[1])!=0) g_NumStr++;
    if(strlen(g_Disp[2])!=0) g_NumStr++;
    if(strlen(g_Disp[3])!=0) g_NumStr++;

    Serial.println(g_NumStr);
    PrintLines(g_Disp[0], g_Disp[1]);
  }
  else if (g_NumStr > 2) {
    Serial.println("Scrolling");
    if(g_Scroll < 2) g_Scroll++;
    else g_Scroll = 0;
    PrintLines(g_Disp[g_Scroll], g_Disp[g_Scroll + 1]);
  }
  else { }
}

void PrintLines(char msg0[17], char msg1[17]) {
  g_LCD.clear();
  g_LCD.print(msg0);
  g_LCD.setCursor(0, 1);
  g_LCD.print(msg1);
  Serial.println("-- LCD display:");
  Serial.println(msg0);
  Serial.println(msg1);
  Serial.flush();
}

void ConvertBuffer(char dst[4][17], const char* buff) {
  //memset(dst[0], 0, 17);
  //memset(dst[1], 0, 17);
  static char inter[64];
  memset(inter, 0, 64);
  strncpy(inter, buff, 63);
  strncpy(dst[0], inter   , 16); strncpy(dst[1], inter+16, 16);
  strncpy(dst[2], inter+32, 16); strncpy(dst[3], inter+48, 16);
}
```