Communication Protocols Lab

Table of Contents

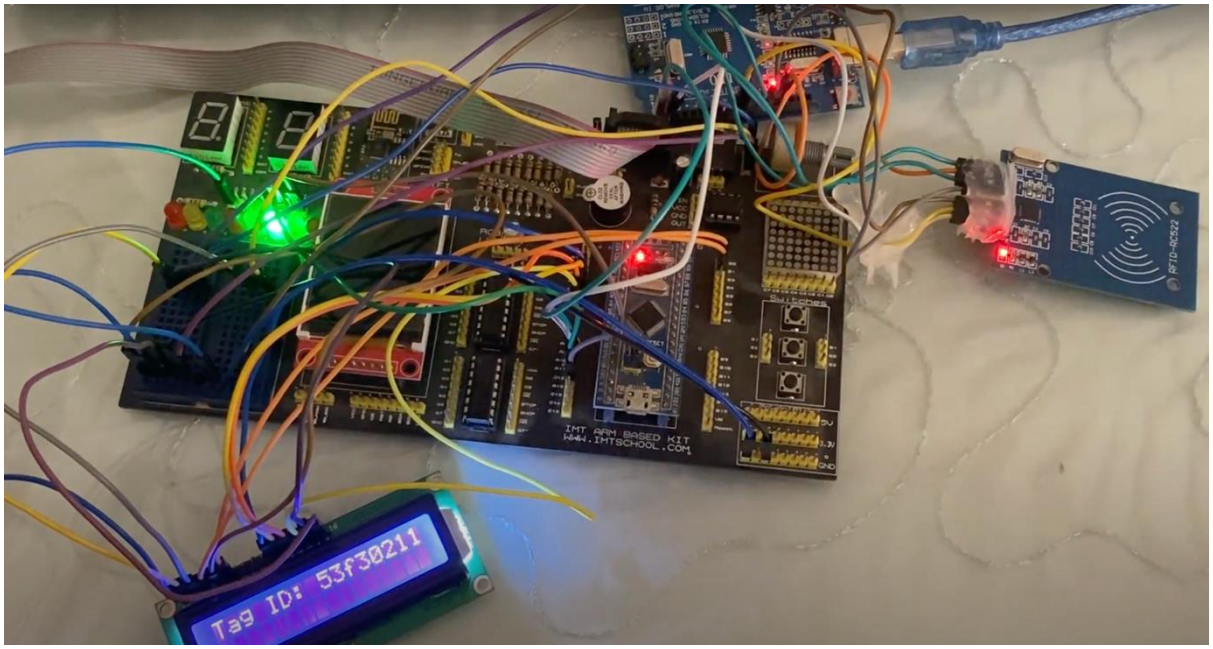
Team names

1- Ahmed Atef Saleh Hassan
2- Alaa Emad Abdelhamed Wahba
3- Gehad Alaa Eldeen Ibrahim
4- Hazem Abdelnasser Mohamed Elsayed
5- Mohamed Ahmed Abdelhalem Abdelhamed
6- Moaaz Ahmed Mohamed Abdelaziz
7- Hager Tarek Abdelmoniem Megahed

Our Project

The main objective of this project is to establish communication between an STM32 microcontroller and an Arduino using the USART (Universal Synchronous Asynchronous Receiver Transmitter) protocol. This communication is utilized to receive data from RFID cards and process that information. The project further involves the use of an LCD for displaying relevant data and two LEDs, a green one and a red one, to provide visual indications. These LEDs are typically used to signal whether the received RFID data matches an authorized ID, thereby serving as a basic access control or authentication system.

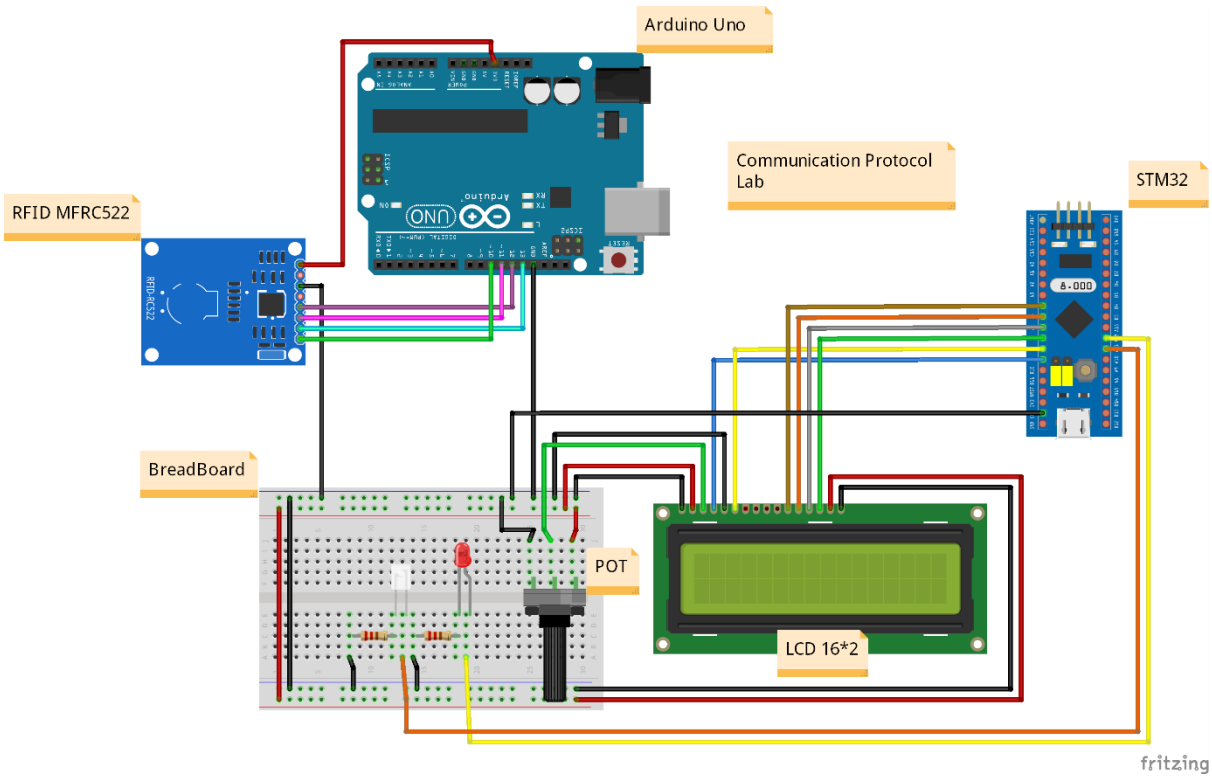
Hardware Description



Hardware components:

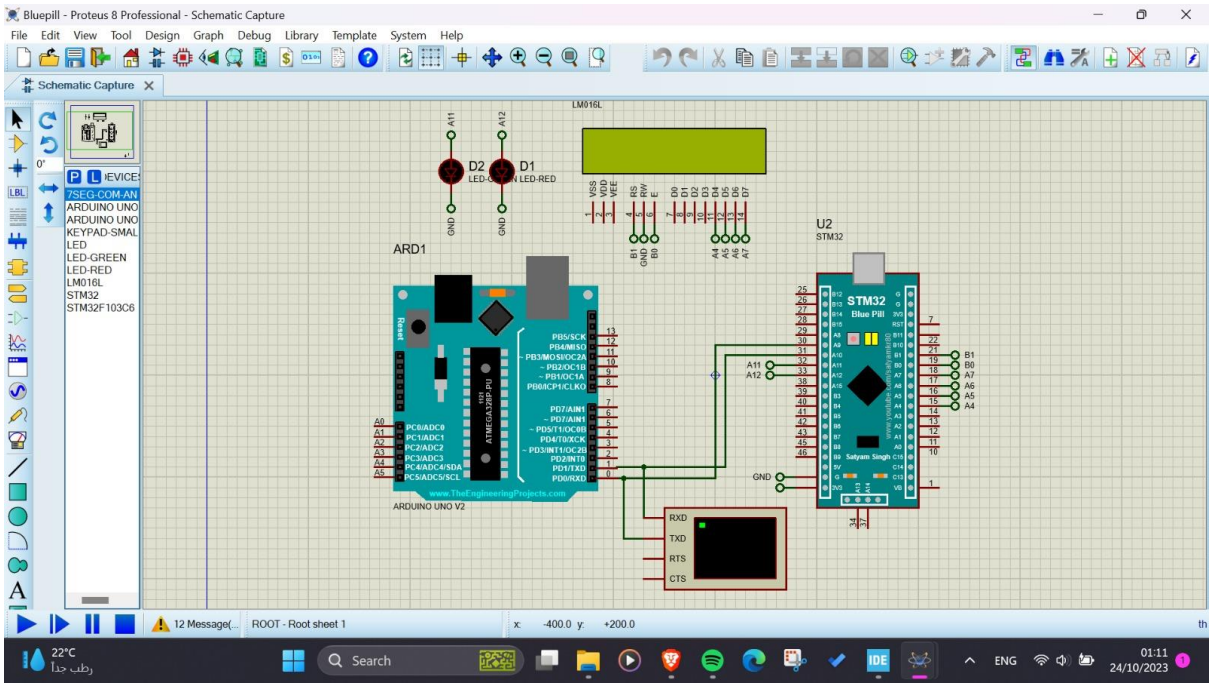
1- The STM32: microcontroller plays a critical role in initializing various system settings. Specifically, it handles the configuration of clock settings and necessary peripherals. Clock initialization is essential to ensure that the microcontroller operates at the desired frequency and timing, which is crucial for communication and other operations.
2- USART: is set up and configured for serial communication with Arduino. Serial communication is a common method for transmitting data between microcontrollers and other devices.
3- LCD: is initialized to serve as a medium for displaying messages and information
4- Arduino (MFRC522): The Arduino reads RFID card data, converts it into a hexadecimal string, and sends the card's unique identifier (UID) to the STM32 microcontroller via UART

Fritzing:



Software Description

Proteus:



Arduino Code:

- Initialize serial communication with baud rate 9600.
- Initialize SPI (Serial Peripheral Interface) bus a common communication protocol used to transfer data.
- Initializes communication with the MFRC522 RFID module.

```
void setup() {
  Serial.begin(9600);
  SPI.begin(); // Init SPI bus
  rfid.PCD_Init(); // Init MFRC522
}
```

- These conditional statements are used to manage the RFID card reading process. They first check if a new card is present, and if so, they proceed to read the card's ID. If the card's ID is successfully read, further processing can take place. If no card is present or if there's an issue with reading the card, the code immediately returns and doesn't proceed, ensuring that only valid card readings are acted upon.

```
void loop() {

  // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
  if ( ! rfid.PICC_IsNewCardPresent())
    return;

  // Verify if the NUID has been readed
  if ( ! rfid.PICC_ReadCardSerial())
    return;
```

- This code processes each byte of the UID, ensures that each byte is represented as a two-character hexadecimal string (e.g."0F" instead of "F" for the value 15), and concatenates these hexadecimal strings together. As a result, ID_str will contain the full hexadecimal representation of the RFID card's UID.

```
 for (int i = 0; i < rfid.uid.size ; i++) {
   if (rfid.uid.uidByte[i] < 0x10) {
     ID_str += "0";  // Add leading zero if the byte is less than 0x10
   }
   ID_str += String(rfid.uid.uidByte[i], HEX);
 }
```

- This code reads and prints the RFID card ID to the serial monitor char by char, then clear ID_str to make it ready for the next card reading.

```
for (int i = 0; i <rfid.uid.size*2 ; i++) { // *2 since byte 0x01 is sent as '0' '1'
  Serial.print(ID_str[i]);
}
ID_str = "";
// Halt PICC
rfid.PICC_HaltA();

// Stop encryption on PCD
rfid.PCD_StopCrypto1();
```

STM32F103C6 code:

- This code snippet is used to receive 8 bytes of data through UART communication char by char, clear the display, and then display a label "Tag ID" along with the received data on an LCD screen.

```
for (int i = 0; i<8; i++) {
        // Uart recieve with polling enabled
        USART_Recieve(USART1, &Buffer[i], Enable);
}
lcd_Clear_Screen(&LCD_pinConfig);
lcd_send_String("Tag ID: ", &LCD_pinConfig);
```

- This loop is designed to display a sequence of characters on the LCD, typically one character at a time, as it increments through the Buffer array.

```
for (int i = 0; i<8; i++) {
        lcd_Send_Char(Buffer[i], &LCD_pinConfig);
}
```

- This if condition compares the content of the Buffer with an authorized ID stored in authorized_ID. If they match, it takes one action: turning on the green LED, and if they don't match, it takes another action: turning on the red LED.

```
// Compare the recieved data with the stored one.
if (!strcmp(Buffer,authorized_ID)) { // if strings are equal it returns 0
        // correct
        GPIO_WritePin(LED_GPIO, Red_LED, PIN_LOW);
        GPIO_WritePin(LED_GPIO, Green_LED, PIN_HIGH);
} else {
        // incorrect
        GPIO_WritePin(LED_GPIO, Green_LED, PIN_LOW);
        GPIO_WritePin(LED_GPIO, Red_LED, PIN_HIGH);
}
```