RFID TAG BOARD LOGGER

Autorzy: Mateusz Kapała i Jan Gąsienica-Józkowy
Elektronika, rok 3
Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Spis treści

1. Wstęp i funkcjonalność....2
2. Analiza problemu....3
3. Opis realizacji....4
3.1. Opis realizacji części RC522 ↔ STM32F401RE....5
A. UART2 jako podgląd testowy....5
B. Komunikacja z RC522 → inicjalizacja SPI....6
C. Komunikacja z RC522 → ciąg dalszy....6
D. Opóźnienia → SysTick....6
E. Timestamp→ RTC....7
3.2. Opis realizacji części STM32F401RE ↔ NodeMCU....7
A. UART1 oraz SoftwareSerial....7
B. Przesył danych pomiędzy płytkami....7
3.3. Opis realizacji części NodeMCU ↔ Firebase....7
3.4. Opis realizacji części NodeMCU ↔ LCD....8
3.5. Opis realizacji części Firebase ↔ Android App....8
4. Zastosowanie praktyczne....9
5. Opis protokołów, modułu, peryferium....9
A. SPI....9
B. UART....9
C. I2C....9
D. RC522....10
E. RTC....10
F. LCD HD44780....10
6. Konfiguracja……...…11

1. Wstęp i funkcjonalność

Celem projektu było utworzenie bazy logów wypożyczania wbudowanych zestawów rozwojowych (np. KL46Z, STM32 Nucleo lub inne) za pomocą modułu RC522, który pozwoliłby na odczytanie legitymacji studenckich oraz naklejonych tagów na wyżej wymienionych płytkach. Odczytane UID (numery identyfikacyjne) pozwalają na przesłanie przyporządkowanego zestawu rozwojowego do konkretnego studenta do bazy danych, co pozwala na kontrolowanie wypożyczanych urządzeń poprzez aplikację na systemy Android.

2. Analiza problemu

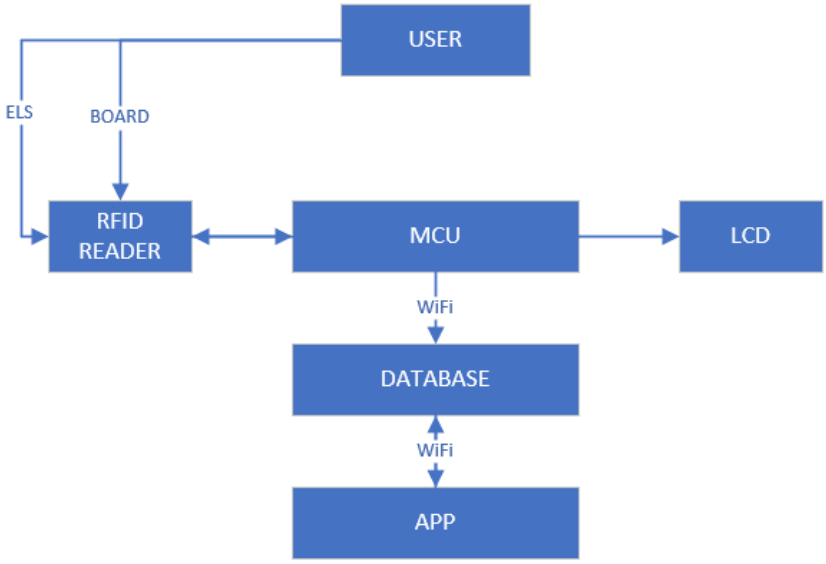
Na wstępie należy zapoznać się z pojęciem RFID.

RFID (z ang. Radio Frequency Identifiaction) jest to technika, wykorzystująca fale radiowe do przesyłania danych oraz zasilania elektronicznego układu stanowiącego etykietę obiektu przez czytnik, w celu identyfikacji obiektu.

Takim czytnikiem jest moduł RC522, umożliwia ona odczyt i zapis danych z urządzeń RFID pracujących na częstotliwości 13,56MHz. Komunikacja z modułem odbywa się poprzez SPI. W datasheet modułu opisane są poszczególne rejestry pozwalające na komunikację z nim.

Za pomocą czytnika należałoby więc odczytać numer identyfikacyjny ELS (elektroniczną legitymację studencką) a także umieszczoną na zestawie rozwojowym naklejkę z tagiem RFID. Po odczytaniu obu UID należy wysłać je do bazy danych do której to docelowo dostęp byłby z aplikacji na smartfony z systemem Android.

W aplikacji powinna być historia wypożyczonych płytek wraz z datą oraz UID płytki oraz ELS, a także możliwość edycji danych studenta.

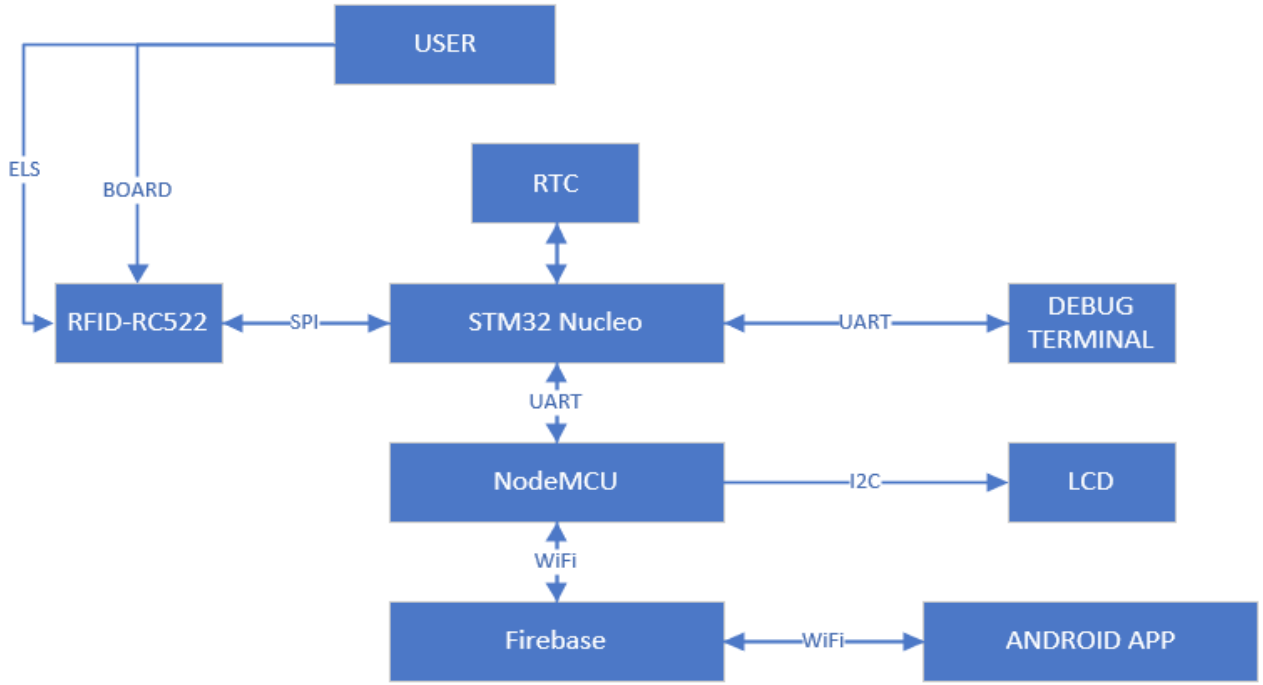


Schemat 1: Ideowa zasada działania

3. Opis realizacji

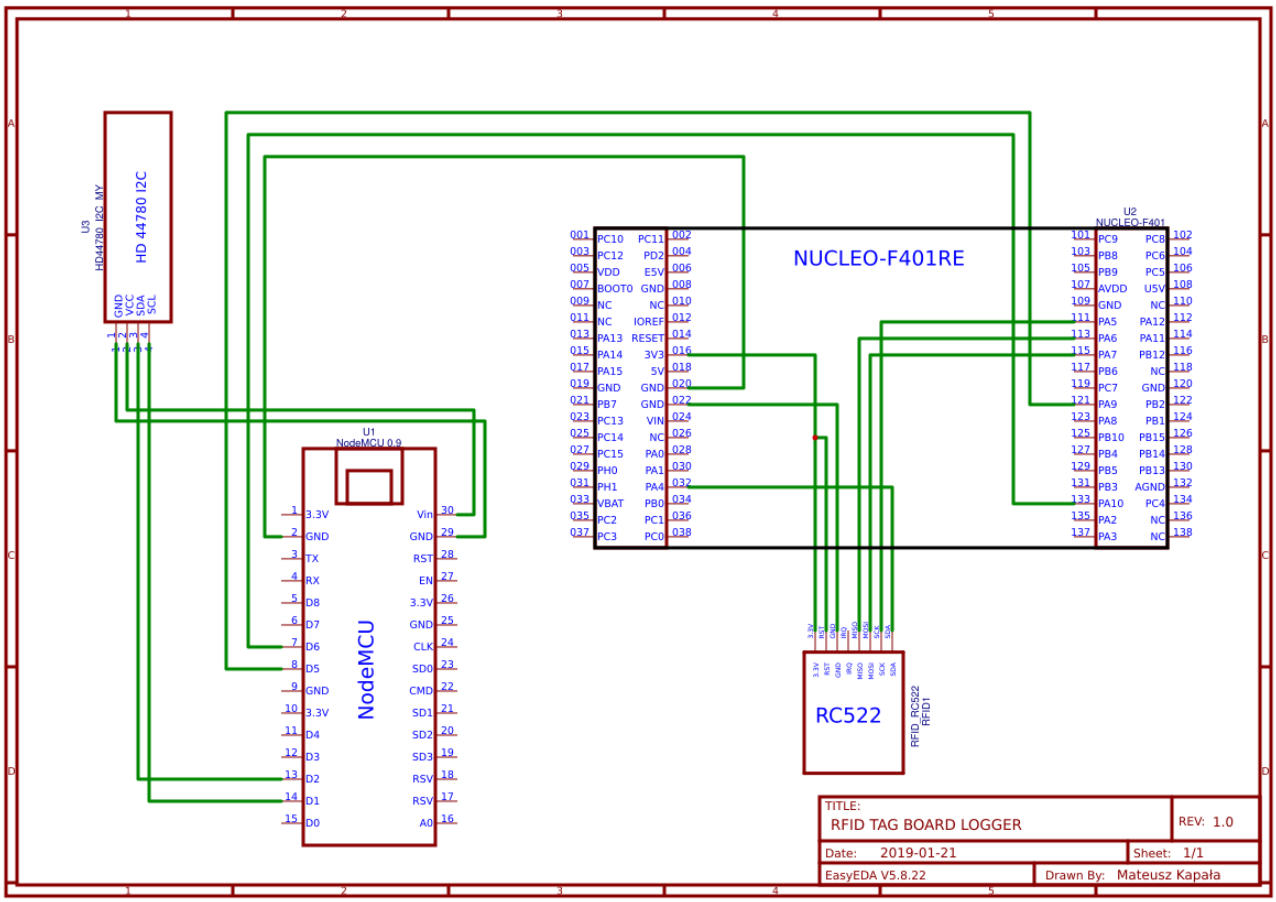
Jako kontroler obsługujący RC522 wykorzystano STM32 Nucleo w wersji STM32F401RE, natomiast przesył do bazy danych przez WiFi został zrealizowany na module NodeMcu 0.9 wyposażony w ESP8266. Dodatkowo dołączono ekran LCD na sterowniku HD44780 za pomocą którego poprzez I2C wyświetlane są instrukcje oraz komunikaty.

Bazą danych została Realtime Database FIREBASE od Google z darmowym abonamentem.



Schemat 2: Zasada działania RFID TAG BOARD LOGGER

Na schemacie numer 3 znajduje się schemat połączeń wszystkich modułów.



Schemat 3: Schemat połączeń

3.1. Opis realizacji części RC522 ↔ STM32F401RE

A. UART2 jako podgląd testowy

W celu debugowania poprawności działania układu RC522 najpierw napisano inicjalizacje UART w celu komunikacji Nucleo z komputerem. Jako że płytka wyposażona jest w konwerter z UART do USB wybrany został moduł UART2 którego piny PA2 i PA3 (TX i RX) są podłączone do wyżej wspomnianej przejściówki.

Następnie napisano funkcję wysyłającą dane po UART oraz funkcję wysyłającą stringa. W tym przypadku odbiór danych z komputera nie jest konieczny tak więc nie został on zaimplementowany.

- Funkcje dot. UART2:
  - **void UART2_Init(void);** → funkcja inicjalizująca
  - **void UART2_sendByte(char);** → funkcja wysyłająca jeden bajt danych
  - **void UART2_sendString(char*);** → funkcja wysyłająca stringa

B. Komunikacja z RC522 → inicjalizacja SPI

Do komunikacji z modułem wykorzystywany jest interfejs SPI. W tym interfejsie komunikacja odbywa się poprzez 3 linię (MOSI, MISO, SCLK) oraz linia aktywująca układ peryferyjny NSS.

W tym celu napisano funkcje inicjalizującą SPI a także funkce wysyłające oraz odbierające dane z tego interfejsu.

Wykorzystano moduł SPI1.

- Funkcje dot. SPI1:
  - **void SPI1_Init(void);** → funkcja inicjalizująca
  - **uint8_t SPI1_writeByte(uint8_t);** → funkcja wysyłająca jeden bajt danych
  - **void SPI1_writeReg(uint8_t, uint8_t);** → funkcja wysyłająca adres oraz dane po SPI
  - **uint8_t SPI1_readReg(uint8_t);** → funkcja czytająca rejestr znajdujący się pod danym adresem

C. Komunikacja z RC522 → ciąg dalszy

W celu skomunikowania się z modułem RC522, po inicjalizacji SPI czas na inicjalizacje samego układu, jednak najpierw trzeba było dopisać kolejne funkcje przesyłające/odczytujące dane z rejestrów ze względu na specyfikę ramki wymaganej przez moduł. Po uporaniu się z problemem w celu inicjalizacji wysyła się szereg danych do poszczególnych rejestrów układu RC522 takich jak ustawienie prescalera, wzmocnienie anteny czy też uruchomienie samej anteny.

By sprawdzić czy inicjalizacja się udała wykorzystano sprawdzenie wersji chipu zamieszczonego w module.

Następnie przyszedł czas na odczytanie UID karty. W tym celu napisano funkcję wysyłającą komendy to tagów RFID, a następnie wykorzystano ją do odczytu UID.

- Najważniejsze funkcje dot. RC522:
  - **void RC522_writeReg(uint8_t, uint8_t);** → funkcja wysyłająca zmodyfikowaną ramkę
  - **uint8_t RC522_readReg(uint8_t);** → funkcja odczytująca ramkę
  - **void RC522_Init(void);** → inicjalizacja RC522
  - **void RC522_firmwareVer(void);** → sprawdzenie wersji chipu
  - **uint8_t RC522_commandTag(...);** → komunikacja z tagiem
  - **uint8_t RC522_checkID(uint8_t*);** → sprawdzenie UID karty, sprawdzenie statusu czy się powiodło

D. Opóźnienia → SysTick

W celach testowych napisano również funkcję opózniające działające na SysTicku. Najpierw napisano funkcję inicjalizującą Systick ustawiającą przerwanie od SysTicka co 10us, w którym do przerwaniu zamieszczono funkcje dekrementującą licznik wymaganą do funkcji opóznienia. Zaprogramowano opóźnienia X*10us, 1ms oraz X*1ms.

E. Timestamp → RTC

W celu identyfikacji kiedy płytka została wypożyczona zaimplementowany został także RTC (zegar czasu rzeczywistego). Po inicjalizacji oraz ustawieniu odpowiedniej daty oraz godziny, należy pobrać godzinę oraz datę z rejestrów a następnie przesłać dalej. Dodatkowo, aby podtrzymać zasilanie RTC podczas gdy Nucleo będzie odłączone od prądu należałoby podłączyć baterię.

- Najważniejsze funkcje dot. UART1:
  - **void RTClock_Init(void);** → funkcja inicjalizująca
  - **void RTClock_getDate(char);** → funkcja pobierania daty z rejestru
  - **void RTClock_getTime(char);** → funkcja pobierania czasu z rejestru
  - **void RTClock_setDate(char);** →ustawianie daty
  - **void RTClock_setTime(char);** → ustawianie czasu
  - **void RTClock_fullToNodeMCU(void);** → przesyłanie timestampa po UART

3.2. Opis realizacji części STM32F401RE ↔ NodeMCU

A. UART1 oraz SoftwareSerial

W celu przesyłu i odbioru danych po stronie STM32 wykorzystano moduł UART1, którego zainicjalizowano oraz napisano funkcję do przesyłu analogicznie jak do UART2 opisanego poprzednio. Do odbioru danych natomiast wykorzystano przerwanie.

Po stronie NodeMCU wykorzystano gotową bibliotekę Arduino SoftwareSerial przystosowaną do modułów na bazie ESP8266. Powodem wyboru tego typu rozwiązania jest to, że ta płytka posiada jedynie dwa moduły UART, z czego jeden UART1 pozwala jedynie na przesył danych, natomiast na UART0 mogą pojawiać się „śmieci” z powodu połączenia go z interfejsem USB.

- Funkcje dot. UART1:
  - **void UART1_Init(void);** → funkcja inicjalizująca
  - **void UART1_sendByte(char);** → funkcja wysyłająca jeden bajt danych
  - **void UART1_sendString(char*);** → funkcja wysyłająca stringa
  - **void UART1_IRQHandler(void);** → przerwanie w którym odbierany jest stan automatu FSM

B. Przesył danych pomiędzy płytkami

W celu przesyłu danych wykorzystano automat. NodeMCU poprzez UART przesyła gotowość do odbioru danych, natomiast w tym czasie STM32 wchodzi w dany stan automatu i wykonuje odpowiednie instrukcje.

Najpierw pobierane jest UID legitymacji, a następnie UID zestawu rozwojowego oraz czas+data, po czym STM32 czeka na kolejne instrukcje.

3.3. Opis realizacji części NodeMCU ↔ Firebase

Do przesyłu logów z NodeMCU do Firebase wykorzystano gotowe biblioteki Arduino. Pierwsza ESP8266WiFi służy do ustalenia połączenia z siecią WiFi, natomiast biblioteki FirebaseArduino oraz ArduinoJson służa do przesyłu danych do Firebase’a.

W celu połączenia ESP8266 z WiFi należy zadeklarować SSID i hasło danej sieci, natomiast w celu połączenia z Firebase wymagany jest host bazy danych XXX.firebaseio.com oraz token autoryzacyjny.

Następnie zgodnie z punktem 3.2.B. pobierane są dane, z których tworzony jest obiekt JSON, który z kolei jest wysyłany do bazy danych. Jeśli jednakże wystąpi niepowodzenie w przesyle danych NodeMCU

zostaje zresetowany (zjawisko występuje głównie podczas dłuższej nieaktywności, kiedy brakuje przesyłu między NodeMCU a Firebase, a ponieważ biblioteka nie oferuje ponownego połączenia z nią pozostaje jedynie restart urządzenia).

3.4. Opis realizacji części NodeMCU ↔ LCD

Na koniec w celu poinstruowania użytkownika systemu dołączono ekran LCD na sterowniku HD44780 16x2 który został połączony przez interfejs I2C z NodeMCU. Ponownie, w celu inicjalizacji oraz sterowania wyświetlaczem została wykorzystana gotowa biblioteka Arduino LiquidCrystal_I2C.

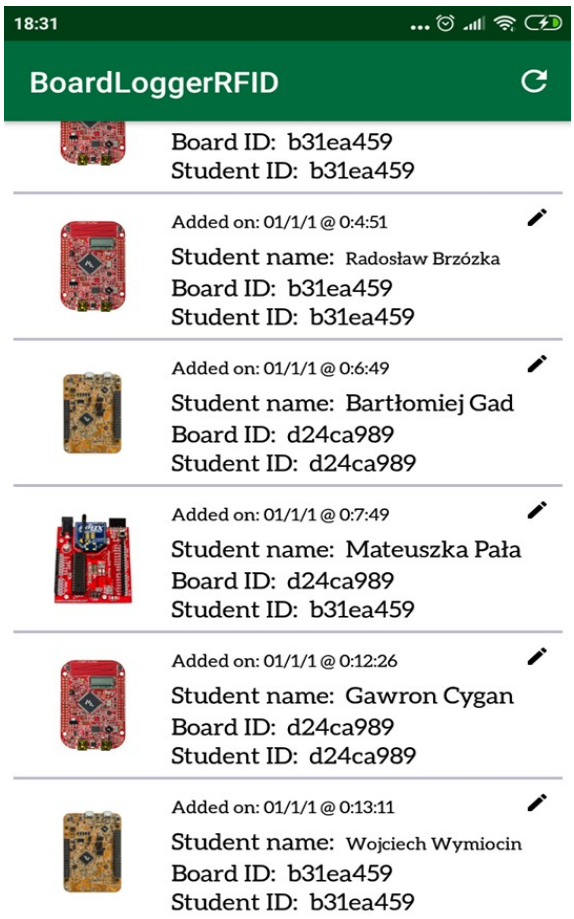
Na początku uruchamiania urządzenia pojawia się nazwa, następnie komunikat o łączeniu z WiFi oraz IP po udanym połączeniu, a następnie pokazywane są instrukcję dla użytkownika.

3.5. Opis realizacji części Firebase ↔ Android App

Aplikacja oparta na systemie Android składa się z dwóch głównych aktywności:

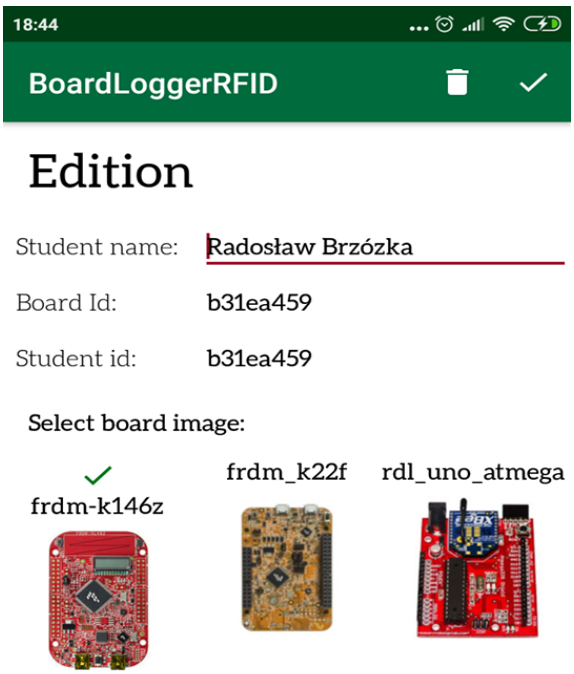
- CatalogActivity – To w nim znajduje się lista wszystkich dodanych logów, w której możemy je przeglądać i wybierać które z nich chcemy zedytować.
- EditorActivity – Aktywność służąca do edycji wybranego logu, dodania imienia i nazwiska studenta oraz wybrania zdjęcia wypożyczanej płytki.

CatalogActivity:



Ilustracja 2: CatalogActivity

EditorActivity:



Ilustracja 1: EditorActivity

4. Zastosowanie praktyczne

Projekt został wykonany w ramach przedmiotu Design laboratory oraz Systemy Mikroprocesorowe. Zakłada on zastosowanie w celu logowania wypożyczanych płytek przez studentów w celach ich monitorowania, czy to pod względem kradzieży czy też możliwych uszkodzeń płytek podczas laboratorium. Jednakże system ten mógłby z powodzeniem również zostać zaimplementowany w bibliotekach w celu wypożyczenia książek bądź w tzw. Unattended Lab, gdzie studenci mogliby wypożyczać nie tylko zestawy rozwojowe ale także inne urządzenia.

5. Opis protokołów, modułu, peryferium

A. SPI

SPI czyli Serial Peripherial Interface jest to szeregowy interfejs urządzeń peryferyjnych. Komunikacja poprzez ten interfejs odbywa się za pomocą 4 linii:

- NSS/SS/CS → Chip Select bądź Slave Select – służy do aktywowania wybranego urządzenia peryferyjnego
- SCLK → jest to sygnał zegarowy/taktujący
- MOSI → inaczej Master Out Slave Input – dane wychodzące dla układu peryferyjnego
- MISO → inaczej Master Input Slave Out – dane przychodzące z układu peryferyjnego.

SPI działa na zasadzie rejestru przesuwnego.

B. UART

UART czyli Universal Asynchronous Reciver-Transmitter jest to uniwersalny asynchroniczny nadajnik-odbiornik, służący do asynchronicznego nadawania oraz odbierania danych poprzez port szeregowy.

Transmisja opiera się na szeregowym wysyłaniu ciągu bitów, które następnie są składane w informację. Bajt danych nazywa się inaczej ramką. Ramka ta rozpoczyna się od bitu startu a kończy na bicie stopu.

Komunikacja poprzez UART wymaga 2 linii:

- TX → transmisja danych
- RX → odbieranie danych

C. I2C

I2C czyli Inter-Integrated Circuit, jest to kolejna szeregowa magistrala, służąca do przesyłu danych w dwie strony.

Interfejs wykorzystuje dwie dwukierunkowe linie, podciągnięte do zasilania przez rezystor (pull-up):

- SDA → linia danych
- SCL → linia zegara

Interfejs działa na bitach. Dane są wysyłane w kolejności od MSB do LSB. Po przesłaniu w jednym kierunku przesyłany jest bit potwierdzający odbiór bądź jego brak. Pierwszy przesyłany bajt to zawsze adres urządzenia.

D. RC522

RFID-RC522 jest to stosunkowo tani moduł RFID operujący na częstotliwości 13,56MHz. Operuje na napięciu 3,3V. Komunikacja z modułem odbywa się poprzez interfejs SPI. Sam moduł zgodnie z dokumentacją techniczną wymaga odpowiedniej ramki, co opisane jest w datasheet modułu MFRC522 w punkcie 8.1.2. W tym projekcie służy on do odczytania numeru identyfikacyjnego tagów RFID oraz elektronicznych legitymacji studenckich.

E. RTC

RTC czyli Real-time Clock, jest jednym z peryferium zawartych w użytym w projekcie procesorze STM32F401RE. Peryferium pozwala nam na odczyt i nastawa aktualnego czasu i daty, ustawienie 2 programowalnych alarmów a także okresowe wybudzenia. W projekcie został wykorzystany jedynie odczyt i nastawa aktualnego czasu w celach uzyskania tzw. timestampa, dzięki któremu można określić kiedy zostały pobrane dane.

F. LCD HD44780

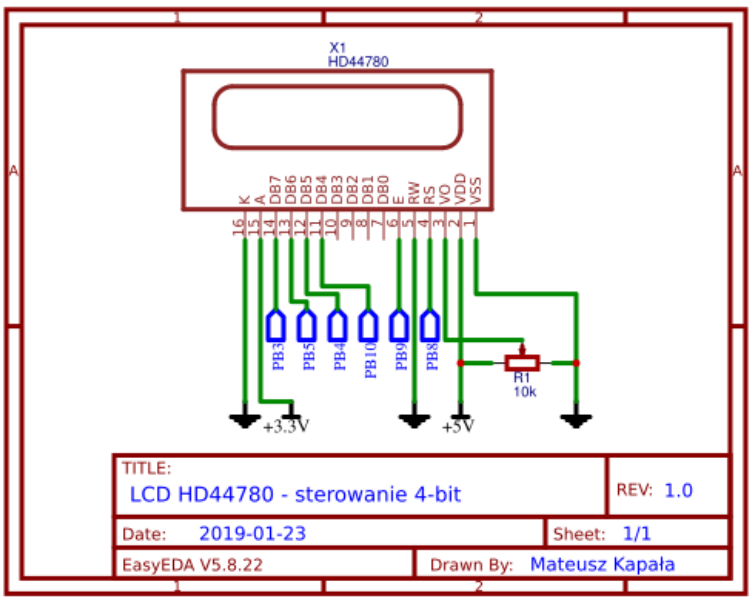
#1 Projekt Design Laboratory

W projekcie użyty został tani ekran LCD 16x2 znaków na sterowniku HD44780. Wyświetalacze te są wyświetlaczami alfanumerycznymi, czyli takimi które pozwalają na wyświetlenie danego znaku w kodzie ASCII, jednakże nie umożliwiają sterowania danymi pikselami.

Do tego wyświetlacza dodatkowo został podpięty konwerter I2C dla tego typu wyświetlaczy, dzięki czemu zmniejszyła się ilość wymaganych linii danych z minimum 6 do zaledwie 2 (nie licząc w obu przypadkach zasilania oraz masy).

#2 Projekt Systemy Mikroprocesorowe 2

Zamiast magistrali I2C do komunikacji z LCD wykorzystano sterowanie 4-bitowe bez odczytu flagi zajętości (pin wyświetlacza oznaczony jako R/W został podpięty do masy). Dlatego też wykorzystano - jak wspomniano w punkcie wyżej – aż 6 pinów płytki Nucleo. Brak odczytu flagi zajętości wymaga generowania dodatkowych programowych opóźnień między instrukcjami.



Schemat 4: Podłączenie LCD do Nucleo w trybie 4-bitowym bez flagi zajętości

6. Konfiguracja

Aby poprawnie skonfigurować system należy:

- Podłączyć wszystko zgodnie ze schematem 3. (strona 5)
- Zaprogramować STM32F401RE
- Ściągnąć biblioteki Arduino: Firebase-Arduino, ArduinoJson, LiquidCrystal_I2C oraz zainstalować biblioteki do płytki ESP8266.
- Skonfigurować program dla NodeMCU poprzez edycje w pliku NodeMCU_firebase.ino linijek:

```
#define FIREBASE_HOST "XXX.firebaseio.com"
#define FIREBASE_AUTH "XXX"
#define WIFI_SSID "XXX"
#define WIFI_PASSWORD "XXX"
```

  Gdzie w miejsca XXX należy wstawić odpowiednie dane:
  - W FIREBASE_HOST należy wpisać adres bazy danych (bez „/” na końcu oraz „https://”)
  - W FIREBASE_AUTH należy wpisać token dostępny na stronie Firebase w Ustawienia→Konta usługi→ Tajne klucze bazy danych → Obiekt tajny
  - W WIFI_SSID należy podać nazwę sieci WiFi
  - W WIFI_PASSWORD należy podać hasło do sieci
- Następnie po zaprogramowaniu należy zresetować ręcznie urządzenie (wymagane przez funkcję ESP.restart(); wystarczy odłączyć zasilanie i podłączyć z powrotem po kilku sekundach)
- Cieszyć się z działającego systemu RFID TAG BOARD LOGGER

Źródła:

- Instalacja płytki ESP8266:
  - http://arduino.esp8266.com/stable/package_esp8266com_index.json
- Biblioteka LiquidCrystal_I2C:
  - https://github.com/johnrickman/LiquidCrystal_I2C
- Biblioteka Firebase-Arduino:
  - https://github.com/Chriton/Firebase-Arduino/tree/master/Arduino%20Project#troubleshooting
- Biblioteka ArduinoJson:
  - https://github.com/bblanchon/ArduinoJson
- Źródła w celu zaprogramowania Nucleo-F401RE:
  - https://github.com/matikap2/RFIDtagboardlogger
- Paczka z STM32F4 do Keil uVision5:
  - Keil.STM32F4xx_DFP.X.X.X.pack → https://www.keil.com/dd2/pack/