Dokumentacja projektu zamka magnetycznego.

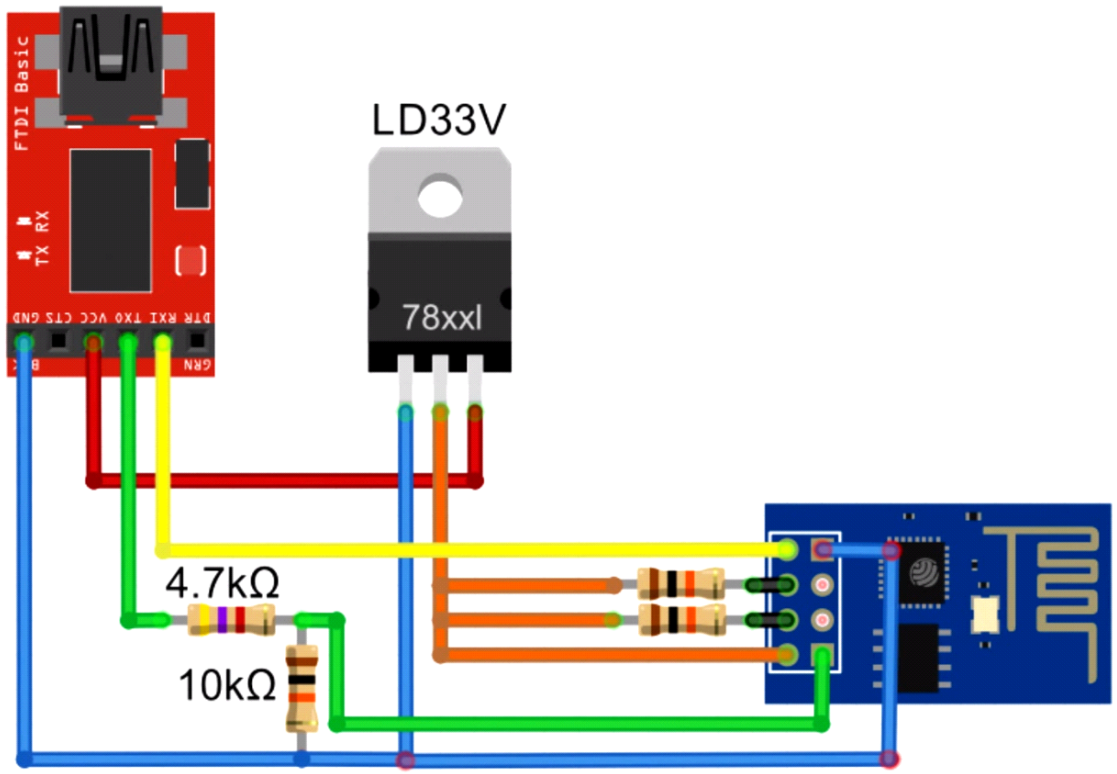
Informatyka Stosowana Grupa 8.

Cezary Matusiak
Paweł Gierczak
Jan Wojtkowski
Adam Lipiński

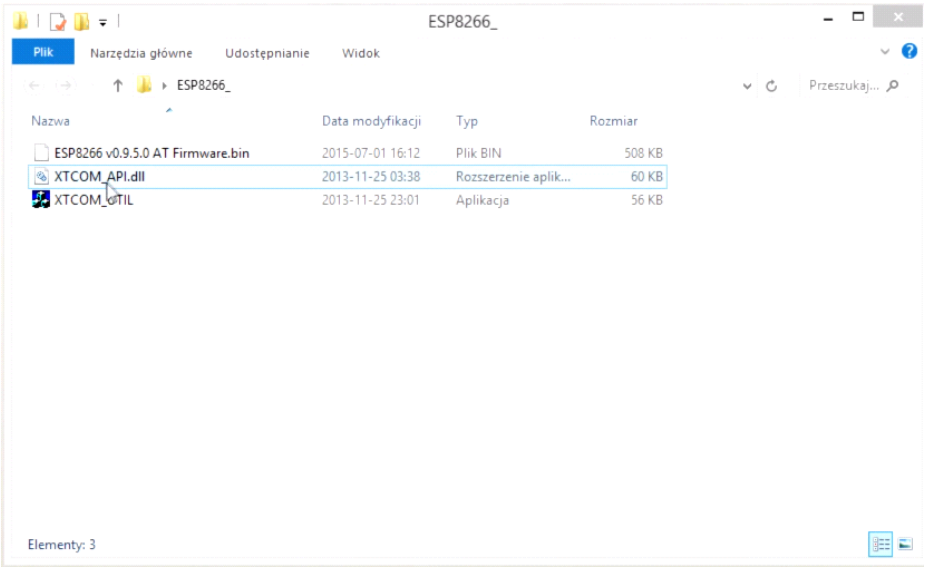
Głównym założeniem projektu jest wykonanie zamka magnetycznego wykorzystującego płytkę Arduino w oparciu o wymagane moduły. Dzięki podłączeniu modułów RFID MFRC522 oraz ESP8266, nasze Arduino będzie wstanie rozczytać kartę i uzyskać jej identyfikator. Następnie dzięki technologii REST, wyślemy zapytanie, które spowoduje komunikacje z wirtualną bazą daną. Otrzymana odpowiedź będzie wyjściową do uzyskania dostępu do drzwi/zamka.

Sprzęt

Na początku zaczęliśmy od podłączenia modułu wifi - “ESP8266” z UART’em RS2302.
Poniższy schemat przedstawia w jaki sposób wykonaliśmy to zadanie.
Do wykonania tego potrzebowaliśmy:
-UART RS 232,
-Stabilizator liniowy LD33V, 3.3V 0.8A,
-3 rezystory 10kOma,
-1 rezystor 4,7kOma,
-Moduł wifi ESP8266,
-Jumpery,
-Płytka projektowa,

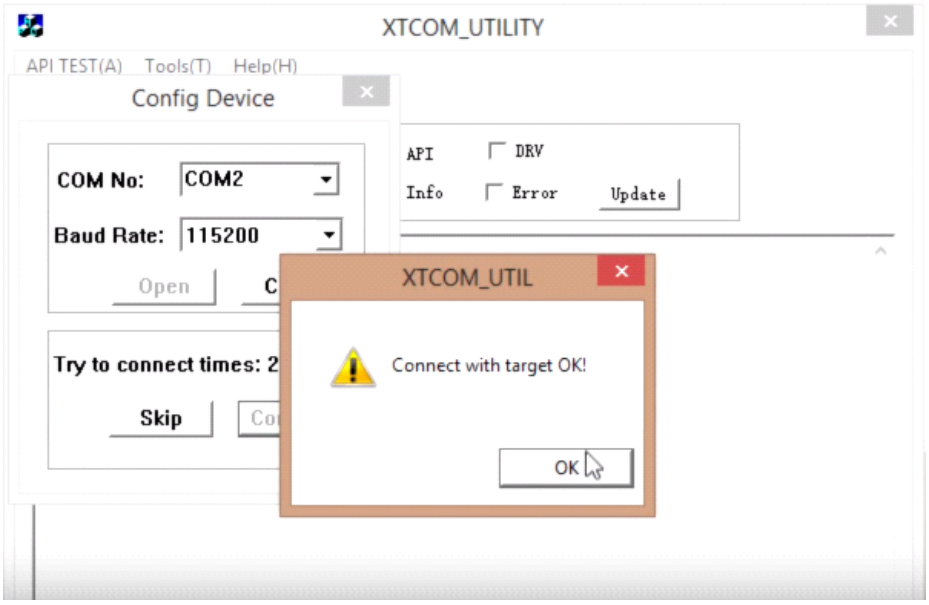


Następnie należało zaktualizować ESP8266,aby to wykonać trzeba było pobrać program ze strony internetowej, połączyć się z modułem poprzez UART, który był podłączony do komputera kablem USB.

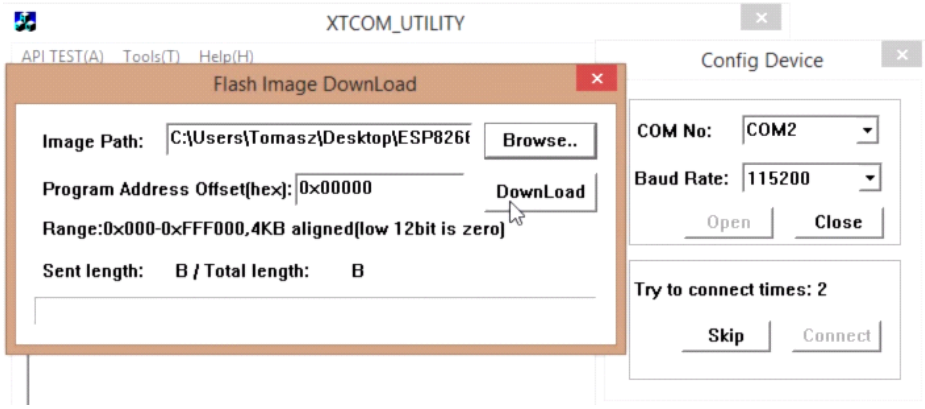


W zakładce tools możemy skonfigurować nasze urządzenie, mamy do wyboru porty COM od 1 do 6, dlatego musieliśmy ustawić urządzenie na jeden z tych portów. (screen)

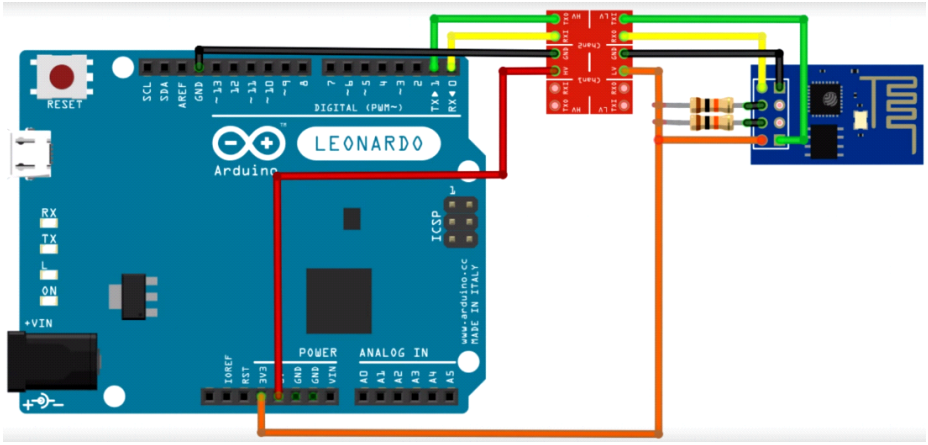
Na końcu należało połączyć się z modułem oraz wybrać oprogramowanie..



Wybieramy opcję API TEST(A) oraz czwartą opcję Flash Image DownLoad. Wybieramy oprogramowanie i klikamy opcję DownLoad, aktualizacja zakończona.



Kolejnym krokiem było podłączenie Arduino Uno z zaktualizowanym modułem WiFi ESP8266. Wykonaliśmy to w poniższy sposób. Potrzebne części:
-Arduino Uno,
-Jumpery,
-Dwa rezystory 10kOma,
-Moduł WiFi ESP8266,
-Konwerter stanów logicznych,

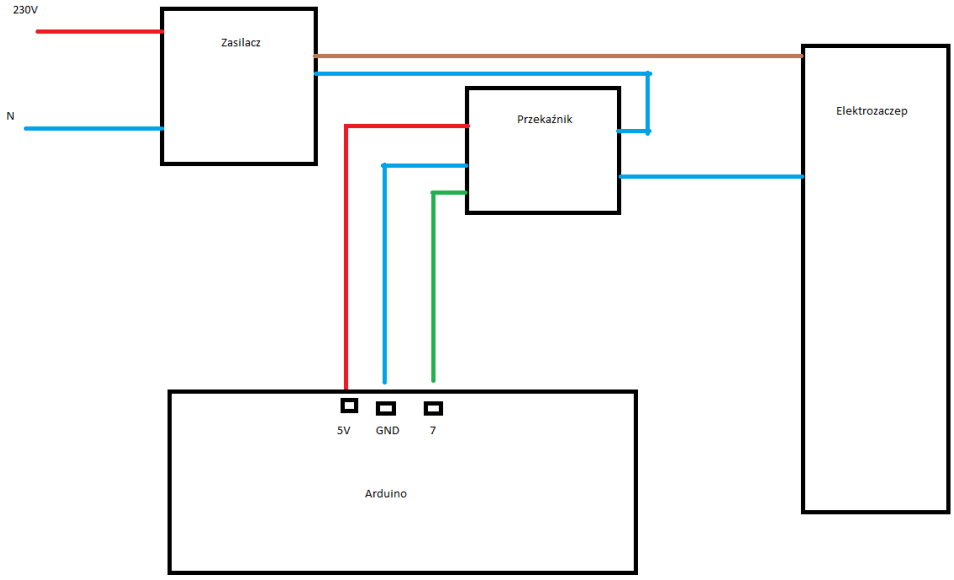


Biblioteka RFID oraz Rest jest już w podstawowej wersji Arduino, więc instalacja najnowszej wersji nie była problemem.
Natomiast bibliotekę do ESP pobraliśmy z internetu, dodaliśmy do Arduino, a następnie zainstalowaliśmy najnowszą wersję.

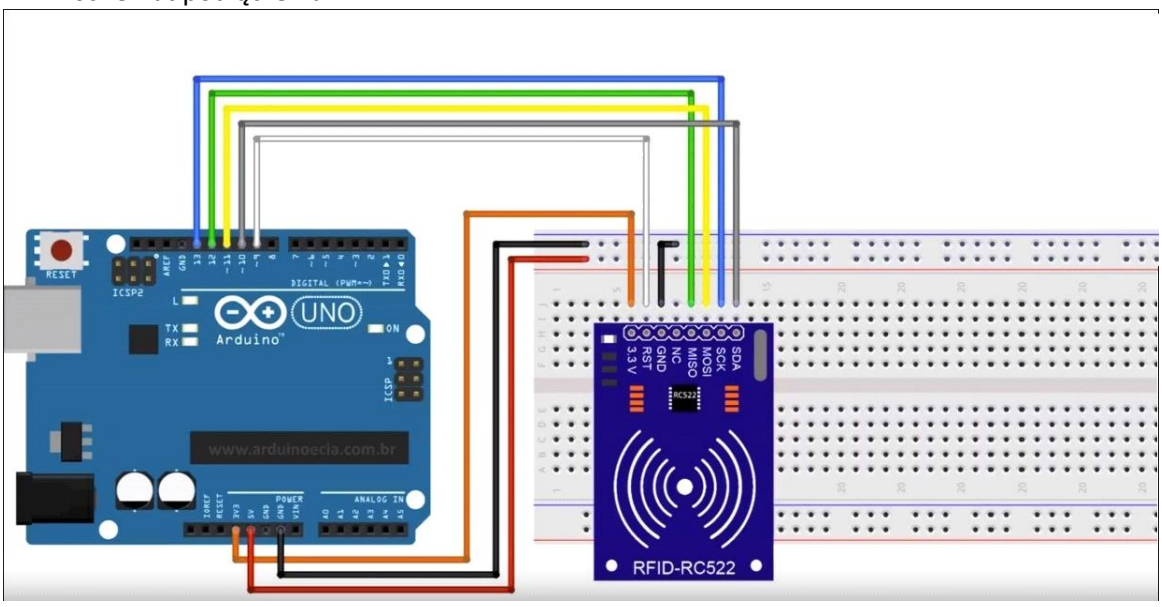
```
Typical pin layout used:
-----------------------------------------------------------------------------------------
            MFRC522      Arduino       Arduino    Arduino    Arduino          Arduino
            Reader/PCD   Uno/101       Mega       Nano v3    Leonardo/Micro   Pro Micro
Signal      Pin          Pin           Pin        Pin        Pin              Pin
-----------------------------------------------------------------------------------------
```

```
Typical pin layout used:
-----------------------------------
            MFRC522      Arduino
            Reader/PCD   Uno/101
Signal      Pin          Pin
-----------------------------------
RST/Reset   RST          9
SPI SS      SDA(SS)      10
SPI MOSI    MOSI         11 / ICSP-4
SPI MISO    MISO         12 / ICSP-1
SPI SCK     SCK          13 / ICSP-3
```

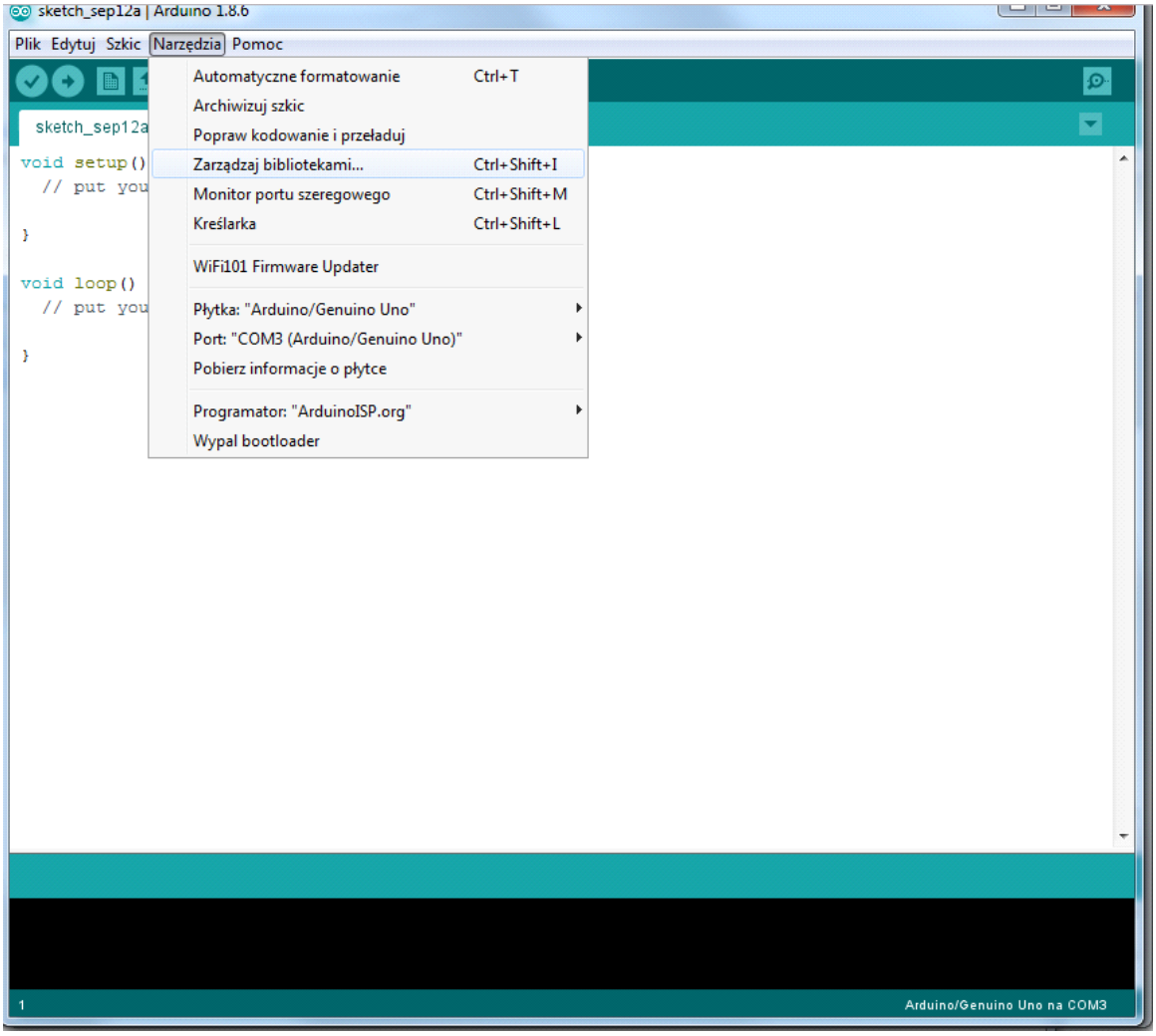
Elektrozaczep- schemat podłączenia.



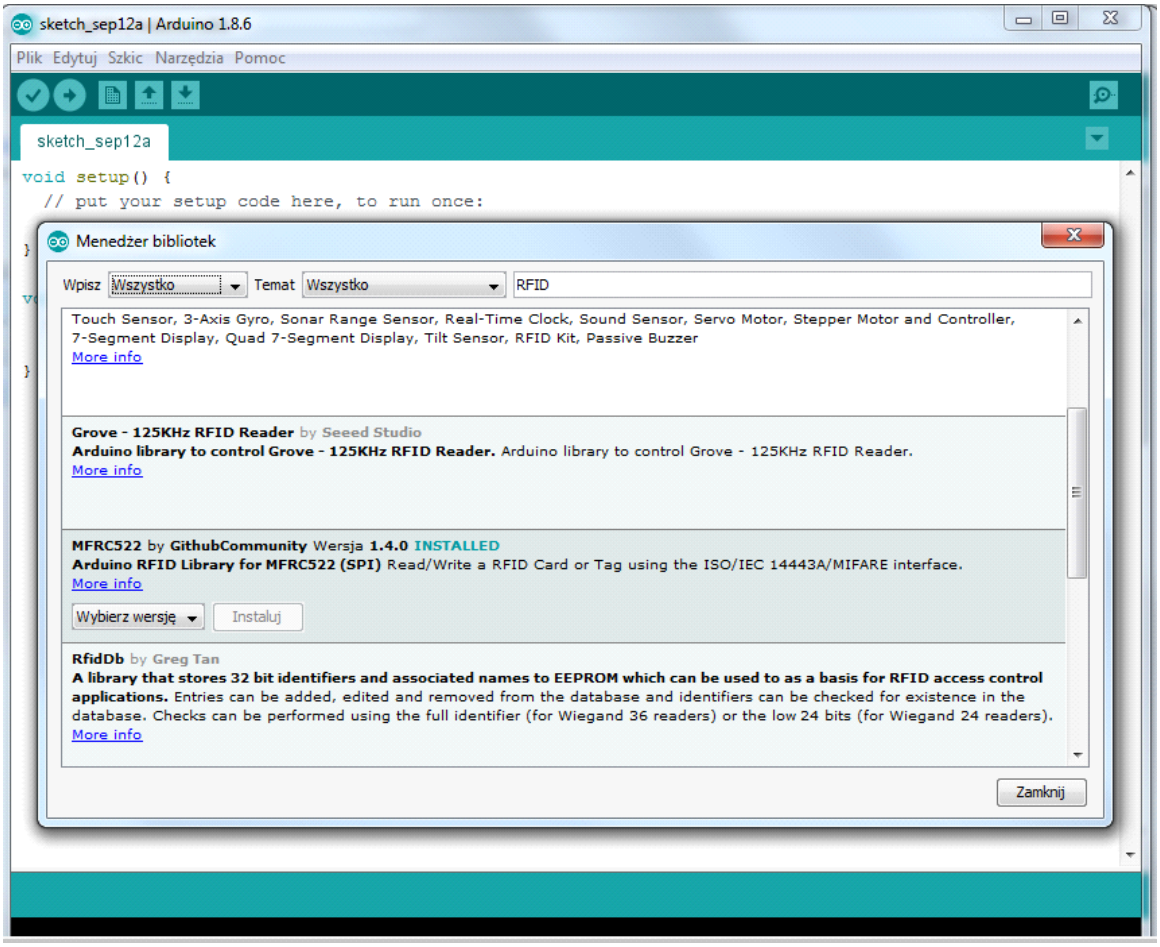
RFID -schemat podłączenia.



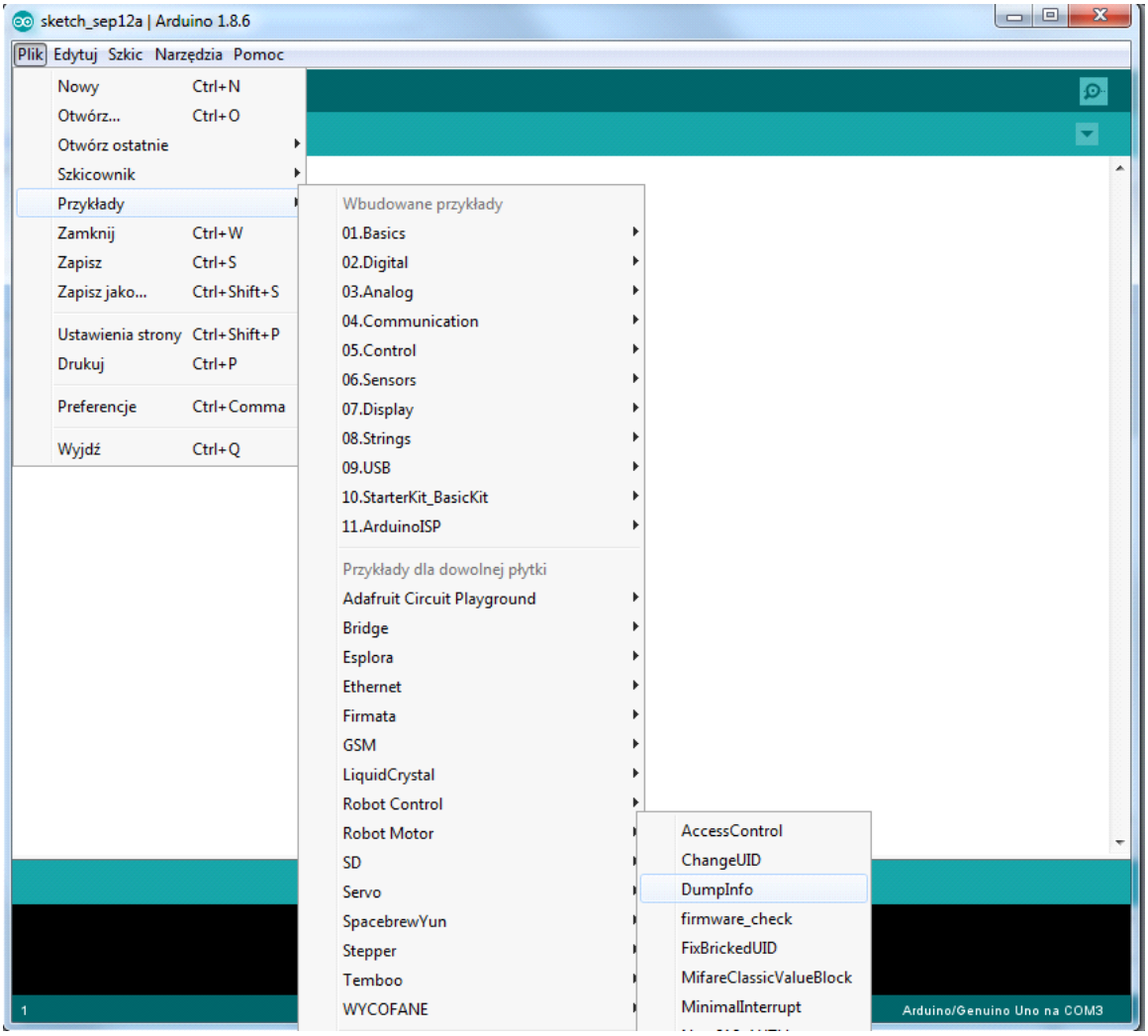
W pierwszej kolejności po zmontowaniu całego układu zgodnie ze schematem wyżej należało zainstalować biblioteki odpowiadające za odpowiednią pracę modułu RFID. Aby to zrobić należało udać się do zakładki narzędzia a następnie "zarządzaj bibliotekami".



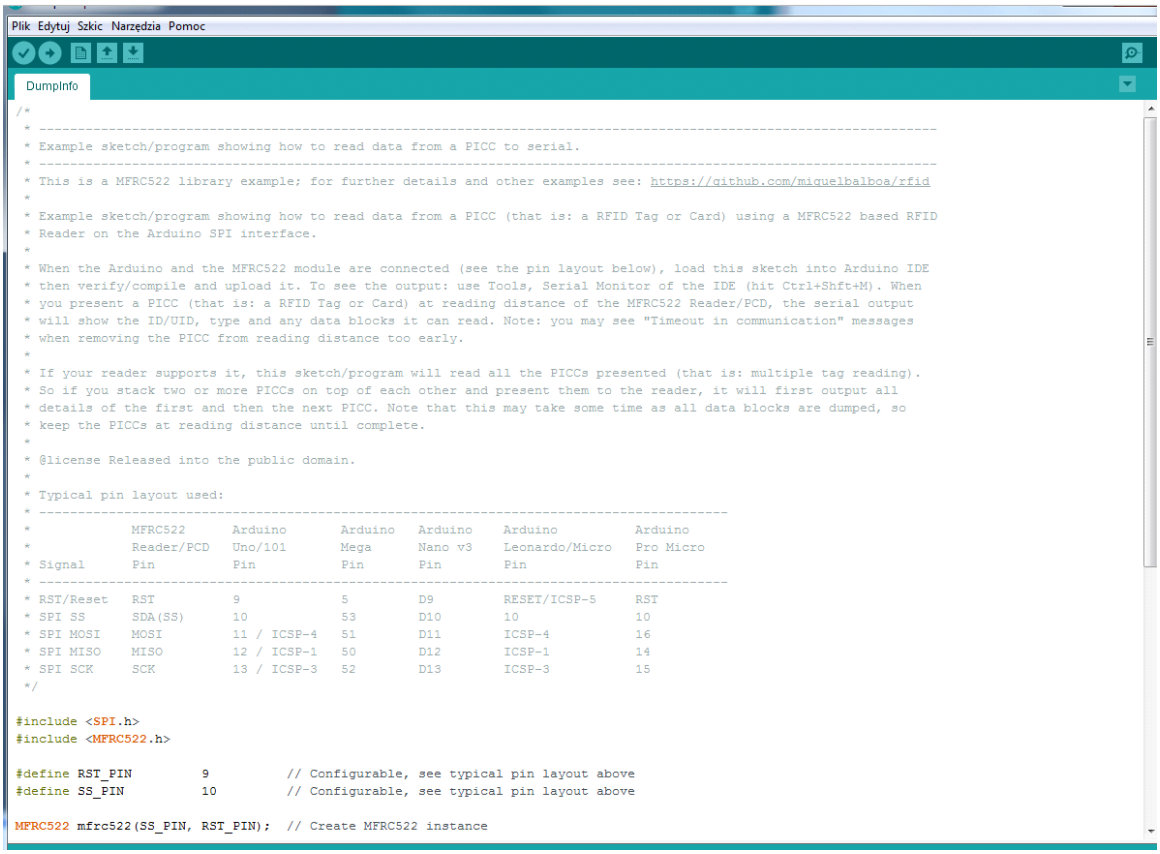
Po wpisaniu do wyszukiwarki odpowiedniego hasła powinny pojawić się biblioteki zawierające funkcjonalności oraz przykłady programów związane z naszym modułem. Należy pobrać bibliotekę o nazwie MFRC522 dzięki której będziemy mogli odczytać tag emitowany przez kartę co pozwoli na wprowadzenie do bazy danych wartości które będą odczytywane jako posiadające dostęp.



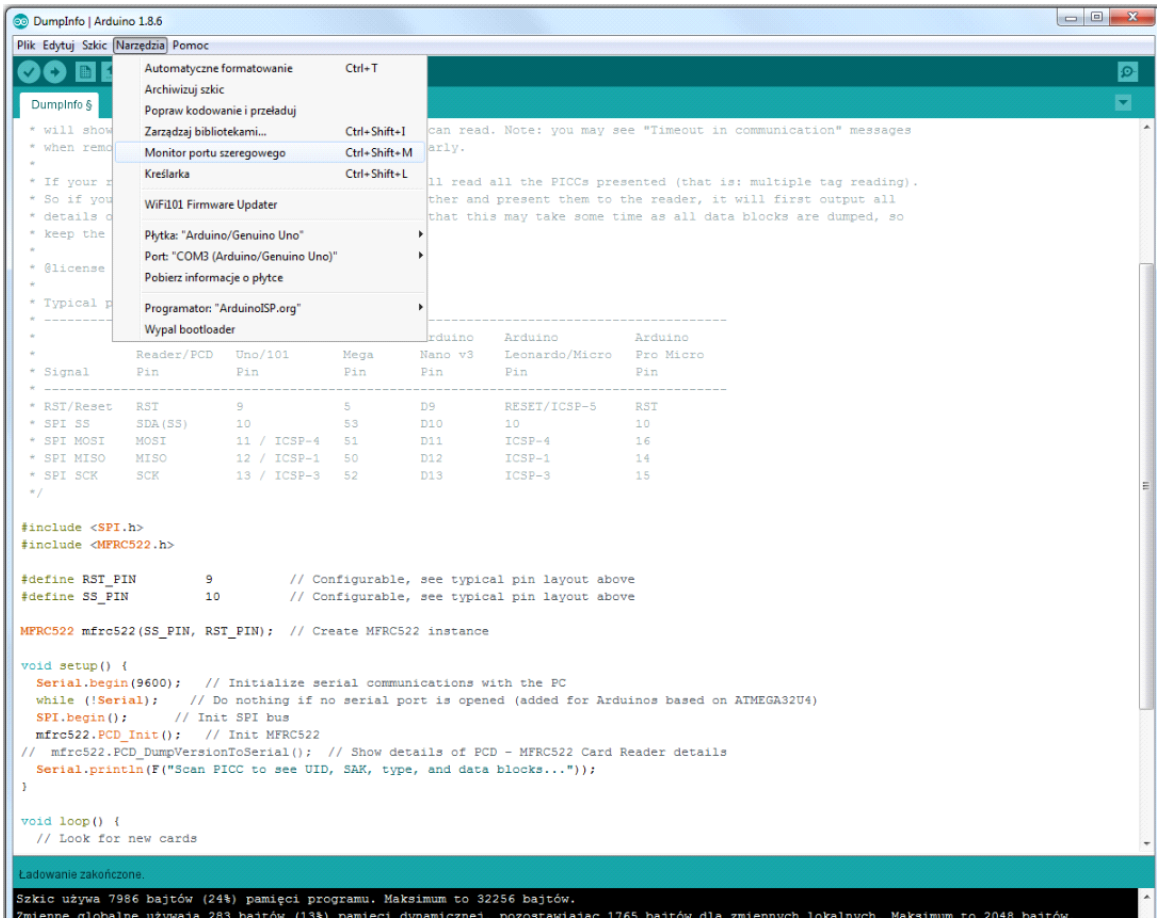
Powodzenie instalacji jest sygnalizowane napisem installed przy nazwie biblioteki oraz w zakładce pliki w panelu przykłady powinna pojawić się nowa biblioteka o tej właśnie nazwie. W tej bibliotece najbardziej interesuje nas przykład o nazwie DumpInfo gdzie odczytamy UID.



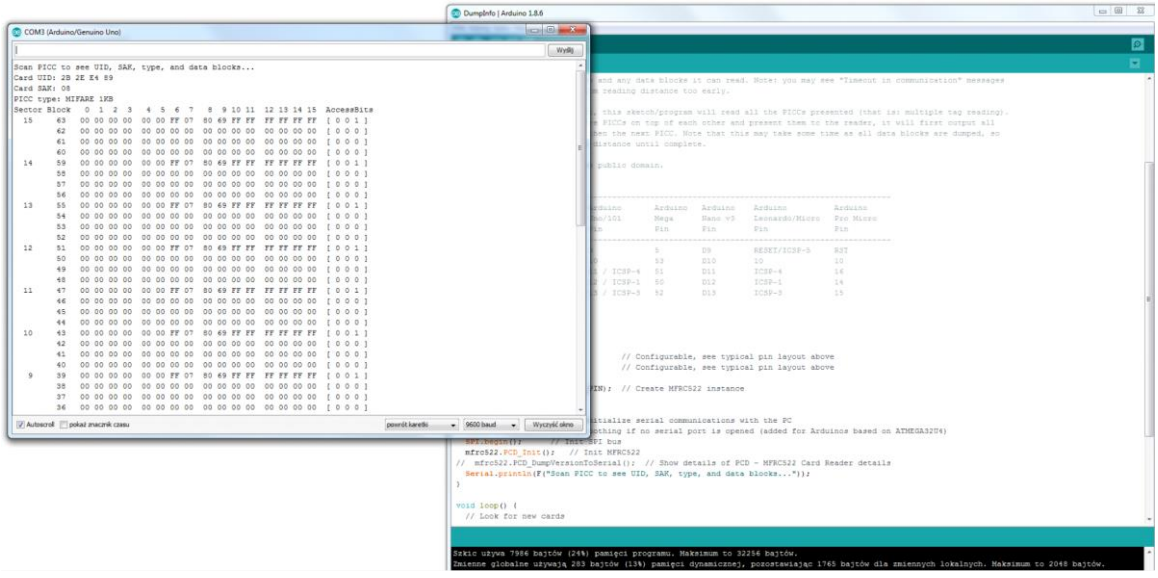
Po wybraniu tego przykładu otwiera się okno z programem który zaczyna się od informacji o module oraz o sposobie podłączenia pinów modułu do arduino co jest kluczowe do działania tej części.



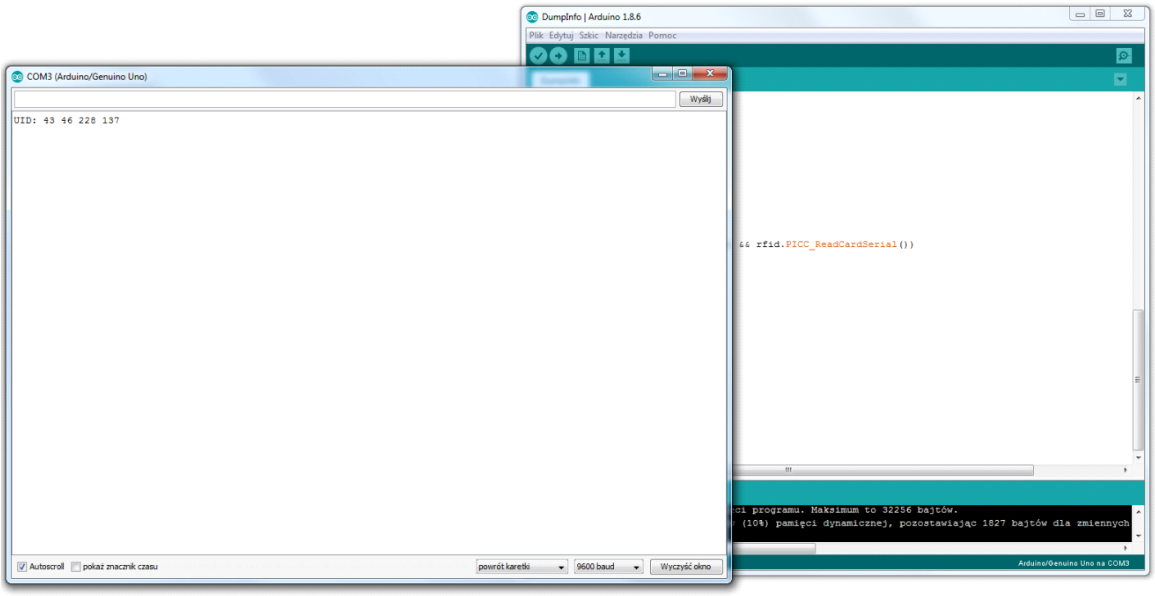
Wgranie programu do pamięci arduino należy uruchomić monitor portu szeregowego aby uzyskać komunikację z mikroprocesorem. Narzędzie to znajduje się w zakładce o tej nazwie.



Narzędzie to pozwoli nam na komunikację z mikroprocesorem i odczyt parametró przykładanych kart. Program DumpInfo po załadowaniu daje odczyt w sytsemie hexadecymalnym



Jednak nie jest to nasz zamierzony efekt więc po modyfikacji kodu otrzymaliśmy bardziej zadawalający nas efekt



Po podłączeniu niezbędnego sprzętu, zajęliśmy się pisaniem kodu naszego programu. Stworzyliśmy funkcję, która odczytuje tag z karty i poszczególne bajty zapisuje w zmiennych INT. Następnie wysyłamy w zapytaniu cztery zmienne, aby nasz BackEnd mógł przeszukać bazę danych w celu ustalenia odpowiedzi. Po otrzymaniu odpowiedzi w json, parsujemy ją na czytelny format. Na podstawie odpowiedzi wnioskujemy następny krok działania programu.

PROGRAM DLA ARDUINO

Po podłączeniu niezbędnego sprzętu, zajęliśmy się pisaniem kodu naszego programu. Stworzyliśmy funkcję, która odczytuje tag z karty i poszczególne bajty zapisuje w zmiennych INT. Następnie wysyłamy w zapytaniu cztery zmienne, aby nasz BackEnd mógł przeszukać bazę danych w celu ustalenia odpowiedzi. Po otrzymaniu odpowiedzi w json, parsujemy ją na czytelny format. Na podstawie odpowiedzi wnioskujemy następny krok działania programu.

Na wstępie dodaliśmy biblioteki stworzone dla czytnika kart oraz zdefiniowaliśmy odpowiednie PIN'y na naszej płytce.

```
#include <SPI.h>
#include <MFRC522.h>
//#include <RestClient.h>
#include <ArduinoJson.h>

#define SS_PIN 10
#define RST_PIN 9
#define RELAY_PIN 7


MFRC522 rfid(SS_PIN, RST_PIN);
//RestClient api = RestClient("192.168.100.100", 80);
//String sciezkaDoApi = "/";
```

Komentarze dotyczą biblioteki poświęconej wysyłaniu zapytań przez WiFi za pomocą technologii REST.(Kod łatwo przerobić na potrzebny sieci Ethernet)

Następnym krokiem było stworzenie funkcji 'setup', która wywoła się tylko po raz po włączeniu programu. Zawiera ona częstotliwość, oraz definiuje czytnik kart.

```
void setup() {
  Serial.begin(9600);
  SPI.begin();
  rfid.PCD_Init();
  rfid.PCD_SetAntennaGain(rfid.RxGain_max);
  //api.begin("SSID", "hasŁ,o");
  //api.dhcp();
  pinMode(RELAY_PIN, OUTPUT);
}
```

Ważną rzeczą jest zdefiniowanie tablicy, w której umieścimy nasze bajty tagu. Następnie tworzymy funkcję 'loop', która będzie się zapętlać. Odczytuje ona tag i zapisuje za pomocą pętli oddzielone od siebie bajty w elementach tablicy. Po tej operacji sprawdzany jest odczyt, czyli czy po przyłożeniu karty będzie ona odczytana.

```
int b[4];

void loop() {
  if (rfid.PICC_IsNewCardPresent())
  {
    if (!rfid.PICC_ReadCardSerial()) {Serial.println("BĹ‚Ä…d
odczytu ID"); return;}
    Serial.print("UID: ");
    for(int i = 0; i<4; i++)
    {
      b[i] = rfid.uid.uidByte[i];
      Serial.print(b[i]);
      Serial.print(" ");
    }
    Serial.println("");
    Serial.print("Sprawdzanie dostÄ™pu... ");
    if (checkAccessPermission())
    {
      Serial.println("OK!");
      Serial.println("");
      digitalWrite(RELAY_PIN, HIGH);
      delay(5000);
      digitalWrite(RELAY_PIN, LOW);
    }
    else
    {
      Serial.println("Blad odczytu!");
      Serial.println("");
    }
    rfid.PICC_HaltA();
  }

  //api.maintain();
}
```

Ostatnim krokiem jest zgodnie z typem danych przesłać zapytaniem http string'a z wartościami naszej tablicy do bazy danych. Następnie po otrzymaniu odpowiedzi udzielony zostanie nam dostęp, lub nie.

```
bool checkAccessPermission() {
  /*String response = "";
  String query = "{\"id0\": " + b[0] + ", \"id1\": " + b[1] +
", \"id2\": " + b[2] + ", \"id3\": " + b[3] + "}";

  com.setContentType = "application/json";
  int statusHTTP = com.post(sciezkaDoApi.c_str(),
query.c_str(), &response);

  if (statusHTTP != 200) return false;
  else
  {
    response.replace(" ","");
    JsonObject& root = jsonBuffer.parseObject(myInput);
    return root["Success"];
  }
```

BACK END

BackEnd został stworzony w celu łatwego rozpoznawania naszych tagów w bazie danych. Główna część programu tworzy bazę DataBase. Poniżej przedstawiamy ciało zapisu karty.

```
new UserData()
{
    id0 = 64,
    id1 = 76,
    id2 = 192,
    id3 = 73,
}
```

W pliku Users.cs zostają zdefiniowane zmienne id0,id1,id2,id3 jako INT.

```
10      public class UserData
11      {
12          /*public byte Byte1 { get; set; }
13          public byte Byte2 { get; set; }
14          public byte Byte3 { get; set; }
15          public byte Byte4 { get; set; }*/
16          public int id0 { get; set; }
17          public int id1 { get; set; }
18          public int id2 { get; set; }
19          public int id3 { get; set; }
20          /*public List<int> Rooms { get; set; }*/
21
22      }
```

W funkcji ICollection gromadzimy zestawy danych (bytes) dla każdej z kart. Na poniższym obrazku widnieje zapis kodu dla funkcji odpowiadającej za sprawdzanie wysłanych przez Arduino zmiennych z danymi zgromadzonymi w ICollection.

```
public static bool ValidateUser(int byte1, int byte2, int byte3, int byte4)
{
    foreach (var userData in Users)
    {
        if(userData.id0 == byte1)
            if(userData.id1 == byte2)
                if(userData.id2 == byte3)
                    if(userData.id3 == byte4)
                        //if (userData.Rooms.Contains(room))
                            return true;
    }

    return false;
}
```

Funkcja zwraca wartość ‘true’, jeśli wszystkie porównane zmienne są zgodne oraz ‘false’ jeśli, któraś z nich lub wiele nie zgadzają się z wysłanymi przez Arduino. Projekt BackEnd zawiera również pliki samo twórcze, które w następstwiee utworzenia projektu, wypełniały się wraz z dodawaniem bibliotek. Pierwotnie miała zostać użyta baza NoSQL LiteDatabase, jednak nie udało nam się jej poprawnie utworzyć(Pliki bazy nie zostały usunięte, więc istnieje możliwość udoskonalenia jej według własnych preferencji), przez co przenieśliśmy się na bazę Enum. Następnym etapem było utworzenie

GUI dla zapytań w localhost'cie, dzięki czemu możliwe było wykonanie testów wysyłanych zapytań. Wszystkie testy przeszły prawidłowo. Odpowiedzi do Arduino są w formacie json, przez co wymagane będzie jego parsowanie w głównym kodzie programu Arduino.

Przykładowe pytania oraz odpowiedzi na nie.

1.KORZYSTAM Z INSTANCJI SERWERA AUTORYZACJI NA HOSTINGU I KOMUNIKACJA POMIĘDZY PŁYTKĄ A SERWEREM NIE PRZEBIEGA PRAWIDŁOWO

Jest to błąd spowodowany hostingiem (shared server), który nie pozwala na uzyskanie osobnego adresu IP dla serwera autoryzacji. Płytka woli podanie adresu IP zamiast nazwy domenowej serwera. Możemy to rozwiązać poprzez wykorzystanie dedykowanego komputera/serwera, najlepiej w sieci lokalnej, z jasno określonym i dostępnym dla płytki adresem IP.

2.PŁYTKA NIE MOŻE POŁĄCZYĆ SIĘ Z SIECIĄ

Zazwyczaj jest to związane z problem z serwerem DHCP w sieci – upewnij się, że jest on aktywny i działa prawidłowo. Jest również możliwość ręcznego przypisania adresu IP dla płytki, ale wymaga ona dość skomplikowanych zmian w kodzie źródłowym szkicu Arduino.

3.MOJA KARTA NIE CHCE DZIAŁAĆ/NIE JEST ROZPOZNAWANA

Wykorzystany moduł NFC obsługuje tylko niektóre rodzaje kart zbliżeniowych – sprawdzaliśmy go dla tagów MIFARE Classic (ISO 14443-3A; standardowy tag komercyjny), MIFARE Ultralight (ISO 14443-3A) i MIFARE Plus(ISO 14443-4; ELS), ale dla ostatniej z wymienionych nie zawsze nie działa poprawnie. Elektroniczna Legitymacja Studencka korzysta z dwóch pętli, jedna z tagiem MIFARE Classic, a druga –MIFARE Plus. Oba mają jednakowy identyfikator taga, jednak dla sprawdzonych przez nas legitymacji program odczytywał tylko połowe tagu.

Wnioski

-Udało nam się zdobyć pewną wiedzę potrzebną do programowania Arduino w zakresie naszego projektu.
-Przyswoiliśmy w większym stopniu wymagane do napisania prostego programu w Arduino czynniki języka Ansi C.
-Mieliśmy możliwość zrozumienia pewnych pojęć z zakresu elektroniki, takich jak przewodniki, rezystory, płytki stykowe, przekaźniki czy działanie diody.
- Podnieśliśmy umiejętność wykorzystywania nabytej już wiedzy.
- Zdobyliśmy liczne doświadczenie w tworzeniu BackEnd'u, mimo wielu problemów.
- Ale co naszym zdaniem najważniejsze, podnieśliśmy umiejętność pracy w grupie, dzielenia obowiązków oraz wykorzystywania naszej wiedzy w pomocy innym członkom grupy, czego skutkiem było umocnienie cierpliwości.

https://github.com/pawcio197/ProjektCzytnik