

Sprawozdanie

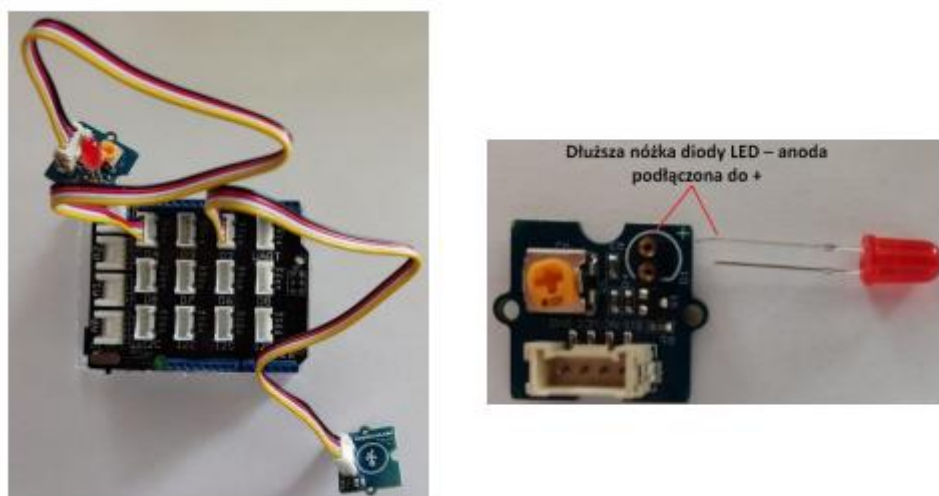
Internet rzeczy: RFID, CZUJNIK DOTYKU, KOMUNIKACJA UART Z OPROGRAMOWANIEM ZEWNĘTRZNYM

1. Cel ćwiczenia - Celem ćwiczenia było zapoznanie z zasadą działania RFID, czujnika dotyku oraz komunikacją UART.

2. Przebieg ćwiczenia.

2.1. Obsługa czujnika/włacza dotykowego.

Postępując zgodnie z instrukcją podłączyliśmy sensor dotyku oraz płytę Seede Studio LED Socket Kit służącą do podłączenia diody LED. Wszystko wyglądało tak jak na podanym schemacie.



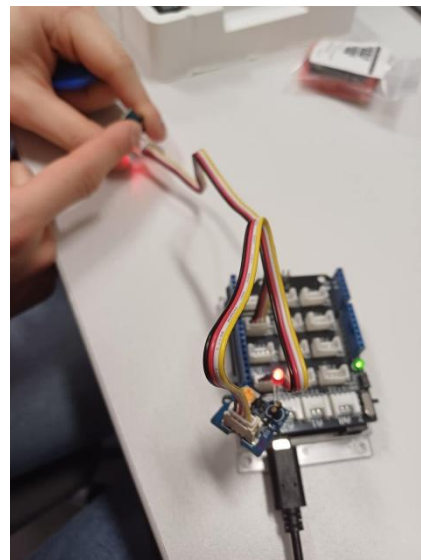
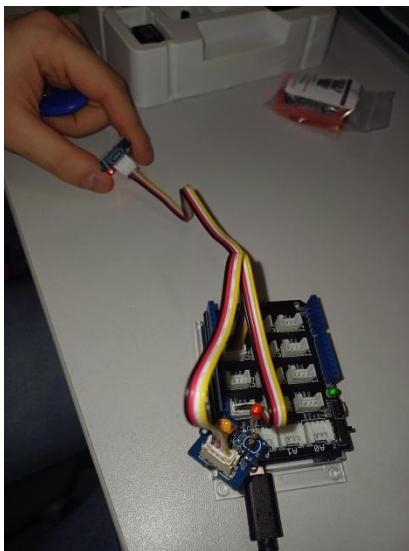
Rys. 1. Podłączanie diody LED

Następnie wykorzystaliśmy podany kod programu otrzymując przy tym w pełni działający układ.

```
const int TouchPin=2;
const int ledPin=4;

void setup() {
  pinMode(TouchPin, INPUT);
  pinMode(ledPin,OUTPUT);
}

void loop() {
  int sensorValue = digitalRead(TouchPin);
  if(sensorValue==1) {
    digitalWrite(ledPin,HIGH);
  }
  else {
    digitalWrite(ledPin,LOW);
  }
}
```



2.2. Komunikacja i wymiana danych z komputerem z wykorzystaniem języka Python.

Następnie wykorzystując kolejny fragment kodu podany w instrukcji, stworzyliśmy aplikację do dwukierunkowej wymiany danych między Arduino i programem napisanym w języku Python z wykorzystaniem magistrali UART.

```

int x;

void setup() {
  Serial.begin(9600);
  Serial.setTimeout(1);
}

void loop() {
  while (!Serial.available());
  x = Serial.readString().toInt();
  Serial.print(x + 1);
}

```

W celu napisania programu w języku Python konieczna była jego instalacja.

```

Wiersz poleceń
Microsoft Windows [Version 10.0.22631.4602]
(c) Microsoft Corporation. Wszelkie prawa zastrzeżone.

C:\Users\student>pip install pyserial
Defaulting to user installation because normal site-packages is not writeable
Collecting pyserial
  Downloading pyserial-3.5-py2.py3-none-any.whl.metadata (1.6 kB)
  Downloading pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
  WARNING: The scripts pyserial-miniterm.exe and pyserial-ports.exe are installed in 'C:\Users\student\AppData\Roaming\Python\Python312\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pyserial-3.5

C:\Users\student>

```

Następnie przepisaliliśmy kod w języku Python, dokonaliśmy żądanych modyfikacji i otrzymaliśmy działającą aplikację.

```

jajca.py - C:/Users/student/Desktop/jajca.py (3.12.0)
File Edit Format Run Options Window Help

import serial
import time

arduino = serial.Serial(port='COM6', baudrate=9600, timeout=.1)

def write_read(x):
    arduino.write(bytes(x, 'utf-8'))
    time.sleep(0.05)
    data = arduino.readline()
    return data

while True:
    num = input("Enter a number: ")
    value = write_read(num)
    print(value)

```

```

IDLE Shell 3.12.0*
File Edit Shell Debug Options Window Help

Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [MSC v.1935 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>>
===== RESTART: C:/Users/student/Desktop/jajca.py =====
Traceback (most recent call last):
  File "C:/Users/student/Desktop/jajca.py", line 1, in <module>
    import serial
ModuleNotFoundError: No module named 'serial'

>>>
===== RESTART: C:/Users/student/Desktop/jajca.py =====
Traceback (most recent call last):
  File "C:/Users/student/Desktop/jajca.py", line 1, in <module>
    import pyserial
ModuleNotFoundError: No module named 'pyserial'

>>>
===== RESTART: C:/Users/student/Desktop/jajca.py =====
Traceback (most recent call last):
  File "C:/Users/student/Desktop/jajca.py", line 1, in <module>
    import pyserial
ModuleNotFoundError: No module named 'pyserial'

>>>
===== RESTART: C:/Users/student/Desktop/jajca.py =====
Enter a number: 13
b'24'
Enter a number: 12
b'23'
Enter a number: 10
b'21'
Enter a number: 2
b'3'
Enter a number:

```

2.3. Odczyt i dekodowanie tagów RFID.

Korzystając z nakładki Grove, połączyliśmy ją z Arduino, a następnie podłączyliśmy moduł czytnika RFID 125kHz. Następnie przepisaliśmy kod pokazany w instrukcji.

```
#include <SoftwareSerial.h>

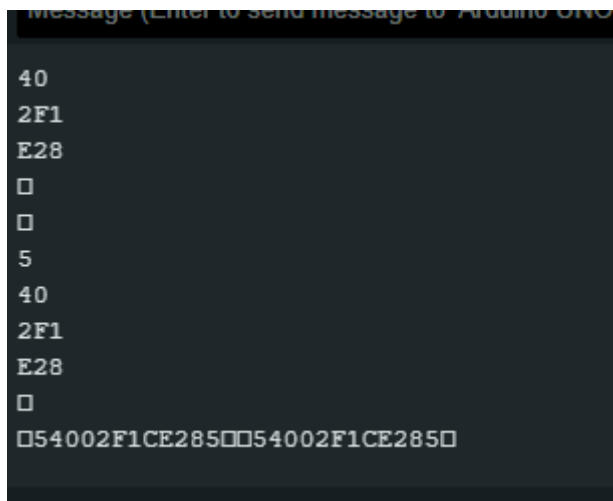
SoftwareSerial SoftSerial(2, 3); //RFID MUST BE CONNECTED To D2
PORT!!!
Unsigned char buffer[64];
int count = 0;           // counter for buffer array

void setup()
{
  SoftSerial.begin(9600);
  Serial.begin(9600);
}

void loop()
{
  if (SoftSerial.available())
  {
    while(SoftSerial.available()) // reading data into char array
    {
      buffer[count++] = SoftSerial.read(); // writing data into array
      if(count == 64) break;
    }
    Serial.write(buffer, count);
    clearBufferArray();
    count = 0;
  }
}
```

```
void clearBufferArray()
{
  for (int i=0; i<count; i++)
  {
    buffer[i]=NULL;
  }
}
```

W rezultacie w Serial Monitorze był widoczny dany rezultat.



```
message (Enter to send message to Arduino Uno)

40
2F1
E28
□
□
5
40
2F1
E28
□
54002F1CE28554002F1CE285
```

Rezultat ten był widoczny oczywiście po przyłożeniu tagu RFID do czytnika. Po konwersji na system dziesiętny, otrzymaliśmy numer 3087586 (z HEX 002F1CE2) co zgadzało się z numerem naszego tagu.



2.4. Komunikacja Python z czytnikiem tagów RFID.

W następnym punkcie wykorzystaliśmy język Python w celu odczytania numeru tagu RFID przesłanego z Arduino.

```
import serial
import time

arduino = serial.Serial(port='COM3', baudrate=9600, timeout=.1)

while True:
    time.sleep(0.5)
    data = arduino.readline()
    print(data)
```

W rezultacie otrzymaliśmy:

```
File Edit Shell Debug Options Window Help
Python 3.12.0 (tags/v3.12.0:0fb18b0, Oct 2 2023, 13:03:39) [M
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more in
>>>
= RESTART: C:/Users/student/ddd.py
b''
b''
b''
b''
b''
b''
b''
b'\x0254002F1CE285\x03'
b''
b''
b''
b''
b'\x0254002F1CE285\x03'
b''
b''
b''
b''
b''
b''
b''
|
```

2.5. Napisać program do sterowania dostępem do pomieszczenia według założeń poniżej.

W tym punkcie wykorzystaliśmy wiedzę z wcześniej wykonanych zadań i stworzyliśmy program, który działał tylko dla jednego podanego tagu RFID.

Poniżej podany kod oraz rezultat po jego odpaleniu.

```

1 #include <SoftwareSerial.h>
2
3 // Konfiguracja pinów
4 const int RFID_RX = 2; // Wejście danych od czytnika RFID do Arduino
5 const int RFID_TX = 3; // Wyjście danych z Arduino do czytnika (nie zawsze wykorzystywane)
6 const int TouchPin = 5;
7 const int ledPin = 4;
8
9 // Oczekiwana długość ramki z RFID (0x02 + 10 znaków ID + 2 znaki checksum + 0x03 = 1 + 10 + 2 + 1 = 14 bajtów)
10 const int FRAME_LENGTH = 14;
11
12 // Znany, dozwolony ID (10 znaków ASCII, np. "1234567890" - należy podmienić na faktyczny ID karty)
13 const char validID[] = "54002F1CE2";
14
15 SoftwareSerial SoftSerial(RFID_RX, RFID_TX);
16
17 void setup() {
18     pinMode(TouchPin, INPUT);
19     pinMode(ledPin, OUTPUT);
20
21     Serial.begin(9600);
22     SoftSerial.begin(9600);
23
24     Serial.println("System start - waiting for RFID and touch...");
25 }
26
27 void loop() {
28     // Sprawdzamy czy jest dostępny pełny odczyt RFID
29     char rfidFrame[FRAME_LENGTH];
30
31     if (readRFIDFrame(rfidFrame)) {
32         // Odczytaliśmy pełną ramkę z RFID
33         // Format ramki: [STX=0x02] + [10 znaków ID] + [2 znaki Checksum] + [ETX=0x03]
34         // Indeksy: 0x02 na rfidFrame[0], ID na [1..10], Checksum [11..12], ETX na [13].
35
36         bool isChecksumOK = verifyChecksum(rfidFrame);
37         bool isIDOK = checkID(rfidFrame);
38
39         int touchValue = digitalRead(TouchPin);

```

```

41 // Informacja do PC o próbie dostępu
42 Serial.print("Access attempt -> ID: ");
43 for (int i = 1; i <= 10; i++) {
44     Serial.print(rfidFrame[i]);
45 }
46 Serial.print(", ChecksumOK: ");
47 Serial.print(isChecksumOK ? "YES" : "NO");
48 Serial.print(", Touch: ");
49 Serial.print(touchValue == HIGH ? "YES" : "NO");
50 Serial.print(" -> ");
51 Serial.print("RFID Frame: ");
52 for (int i = 0; i < FRAME_LENGTH; i++) {
53     Serial.print(rfidFrame[i], HEX); // Wyświetlaj ramkę w HEX
54     Serial.print(" ");
55 }
56 Serial.println();
57 if (isChecksumOK && isIDOK && (touchValue == HIGH)) {
58     // Dostęp przyznany
59     Serial.println("ACCESS GRANTED");
60     grantAccess();
61 } else {
62     // Odmowa dostępu
63     Serial.println("ACCESS DENIED");
64     denyAccess();
65 }
66 }
67 }
68
69 // Funkcja odczytująca pełną ramkę z RFID
70 bool readRFIDFrame(char *frame) {
71     // Czekamy na początek ramki (0x02)
72     if (SoftSerial.available()) {
73         // Spróbujmy odczytać tyle bajtów, aby dostać pełną ramkę
74         // Aby uniknąć zakleszczeń, możemy oczekiwać na dane z limitowanym czasem
75         unsigned long startTime = millis();
76         int count = 0;
77         while ((millis() - startTime) < 500) {
78             if (SoftSerial.available()) {
79                 char c = SoftSerial.read();
80

```

```

81 // Pierwszy znak musi być 0x02, jeśli nie trafimy na niego, czytamy dalej
82 if (count == 0 && c != 0x02) {
83     // Nieprawidłowy start, kontynuujemy szukanie
84     continue;
85 }
86
87 frame[count++] = c;
88 if (count == FRAME_LENGTH) {
89     // Sprawdzamy czy ostatni znak to 0x03
90     if (frame[FRAME_LENGTH - 1] == 0x03) {
91         return true; // Otrzymano poprawną długościowo ramkę
92     } else {
93         return false; // Ostatni znak nie pasuje
94     }
95 }
96 }
97 }
98 }
99 return false; // Brak pełnej ramki
100 }
101
102 // Funkcja weryfikująca sumę kontrolną
103 // Założenie: suma kontrolna to XOR bajtów ID (zdekodowanych z ASCII HEX), a wynik porównujemy z 2 znakami checksum
104 bool verifyChecksum(char *frame) {
105     // ID to znaki w frame[1..10]
106     // Checksum to znaki w frame[11..12]
107     // Każda para znaków ID reprezentuje jeden bajt hex?
108     // W tym przykładzie zakładamy, że ID składa się z cyfr, a checksum to 2 znaki hex do porównania z XOR ID.
109
110     // Najprostszy przypadek: interpretujemy ID jako 10 ASCII znaków (0-9), a checksum to XOR tych znaków.
111     // Jeśli posiadany czytnik używa innego formatu, należy to dostosować.
112
113     byte xorVal = 0;

```

```

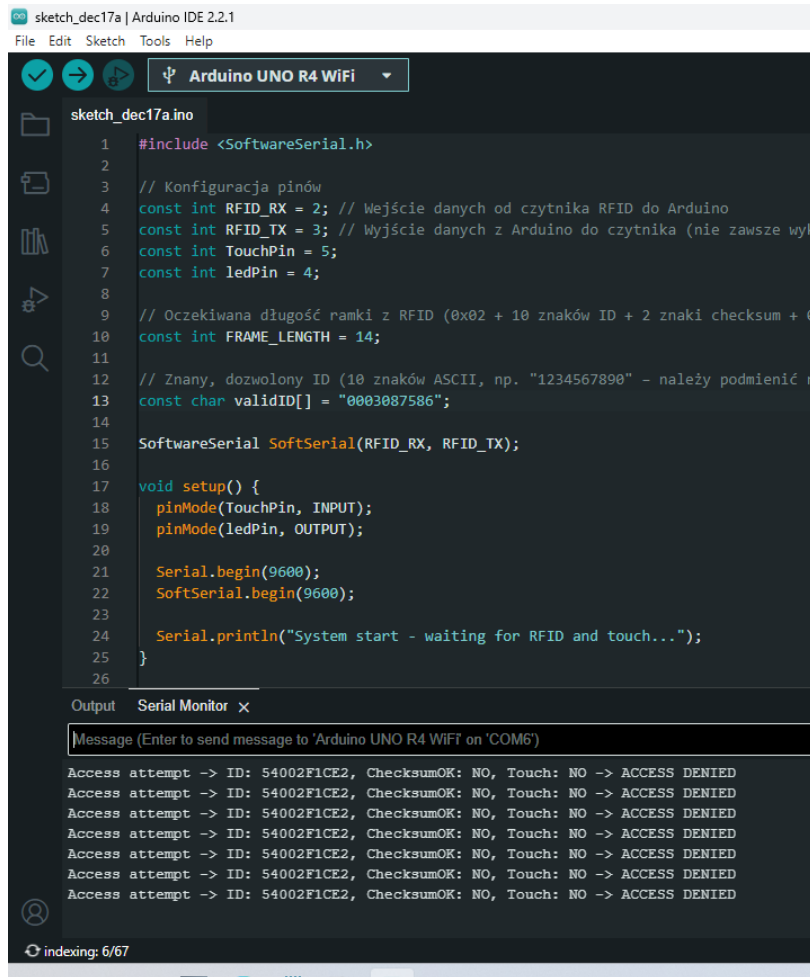
114     for (int i = 1; i <= 10; i++) {
115         | xorVal ^= frame[i]; // XOR bez konwersji, dla prostoty, zakładając że ID to znaki ASCII 0-9
116     }
117
118     // Teraz musimy porównać xorVal z sumą kontrolną zapisaną w dwóch znakach ASCII hex w frame[11] i frame[12].
119     byte readChecksum = hexToByte(frame[11], frame[12]);
120     |
121     return (xorVal == readChecksum);
122 }
123
124 // Konwersja dwóch znaków hex na bajt
125 byte hexToByte(char highNibble, char lowNibble) {
126     return (charToHex(highNibble) << 4) | charToHex(lowNibble);
127 }
128
129 byte charToHex(char c) {
130     if (c >= '0' && c <= '9') return c - '0';
131     if (c >= 'A' && c <= 'F') return c - 'A' + 10;
132     if (c >= 'a' && c <= 'f') return c - 'a' + 10;
133     return 0; // W razie błędu
134 }
135
136 // Sprawdzenie czy odczytane ID to ID uprawnione do dostępu
137 bool checkID(char *frame) {
138     char readID[11];
139     for (int i = 0; i < 10; i++) {
140         | readID[i] = frame[1 + i];
141     }
142     readID[10] = '\0';
143
144     Serial.print("Read ID: ");
145     Serial.println(readID);
146
147     return (strcmp(readID, validID) == 0);
148 }

```

```

149 // Funkcja sygnalizująca przyznanie dostępu
150 void grantAccess() {
151     digitalWrite(ledPin, HIGH);
152     delay(1000);
153     digitalWrite(ledPin, LOW);
154 }
155
156 // Funkcja sygnalizująca odmowę dostępu
157 void denyAccess() {
158     for (int i = 0; i < 3; i++) {
159         digitalWrite(ledPin, HIGH);
160         delay(200);
161         digitalWrite(ledPin, LOW);
162         delay(200);
163     }
164 }

```

The screenshot displays the Arduino IDE 2.2.1 environment. The top menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar shows icons for checking, uploading, and downloading, along with a dropdown menu for the board, currently set to 'Arduino UNO R4 WiFi'. The main editor window shows the sketch 'sketch_dec17a.ino' with the following code:

```
1 #include <SoftwareSerial.h>
2
3 // Konfiguracja pinów
4 const int RFID_RX = 2; // Wejście danych od czytnika RFID do Arduino
5 const int RFID_TX = 3; // Wyjście danych z Arduino do czytnika (nie zawsze wyk
6 const int TouchPin = 5;
7 const int ledPin = 4;
8
9 // Oczekiwana długość ramki z RFID (0x02 + 10 znaków ID + 2 znaki checksum + 0
10 const int FRAME_LENGTH = 14;
11
12 // Znany, dozwolony ID (10 znaków ASCII, np. "1234567890" - należy podmienić n
13 const char validID[] = "0003087586";
14
15 SoftwareSerial SoftSerial(RFID_RX, RFID_TX);
16
17 void setup() {
18     pinMode(TouchPin, INPUT);
19     pinMode(ledPin, OUTPUT);
20
21     Serial.begin(9600);
22     SoftSerial.begin(9600);
23
24     Serial.println("System start - waiting for RFID and touch...");
25 }
26
```

Below the code editor is the 'Serial Monitor' window, which is open. It shows a message input field and a list of messages. The messages are all 'Access attempt -> ID: 54002F1CE2, ChecksumOK: NO, Touch: NO -> ACCESS DENIED'. The status bar at the bottom indicates 'indexing: 6/67'.

3. Wnioski.

- Dwukierunkowa wymiana danych między Arduino a Pythonem przez port UART jest przydatna w kontekście zaawansowanych aplikacji IoT. Oprogramowanie Python z biblioteką pyserial ułatwia komunikację i analizę przesyłanych danych.
- Czytnik RFID wymaga precyzyjnego przetwarzania danych, w tym weryfikacji sumy kontrolnej oraz dekodowania numeru tagu. Wprowadzone procedury zapewniają niezawodność odczytu i pozwalają na kontrolę dostępu.
- Projekt sterowania dostępem do pomieszczenia, oparty na RFID oraz czujniku dotykowym, demonstruje praktyczne zastosowanie IoT w bezpieczeństwie. Wymaga on integracji kilku elementów systemu, w tym komunikacji z komputerem i sygnalizacji LED.

