Politechnika Wrocławska
Wydział Informatyki i Zarządzania

# Raport

## System ewidencji czasu pracy wykorzystujący technologie Internetu Rzeczy

| | |
|---|---|
| Przedmiot: | Podstawy Internetu Rzeczy Laboratorium |
| Imię i nazwisko autora: | Kamil Graczyk |
| Nr indeksu: | 246994 |
| Semestr studiów: | 4 |
| Data ukończenia pracy: | 15 maja 2020 r. |
| Prowadzący laboratorium: | mgr inż. Kamil Nowak |

HR EXCELLENCE IN RESEARCH
Politechnika Wrocławska
Wybrzeże Wyspiańskiego 27
50-370 Wrocław
wiz.pwr.edu.pl

# 2. Spis treści

# 3. Wymagania projektowe

## Podstawowe wymagania funkcjonalne

**Wymaganie:**
Zapis akcji do bazy tekstowej.
**Opis:**
Aplikacja centralna, zwana dalej serwerem , która gromadzi i przetwarza dane nadsyłane od klientów.

**Wymaganie:**
Serwer powinien obsługiwać więcej niż jednego klienta.
**Opis:**
Serwer powinien obsługiwać dowolną liczbę czytników, które zostaną do niego podłączone za pośrednictwem sieci. W skład systemu może wejść więcej niż jedna instancja klienta. Wszystkie instancje klienta komunikują się z jednym serwerem.

**Wymaganie:**
Dodanie nowego użytkownika
**Opis:**
System musi posiadać możliwość dodania nowego użytkownika (pracownika) do bazy za pośrednictwem serwera podając *Imię* oraz *Nazwisko*.

**Wymaganie:**
Przypisanie/usunięcie karty RFID użytkownikowi.
**Opis:**
System musi posiadać możliwość przypisania nowej lub usunięcia istniejącej karty RFID użytkownikowi za pośrednictwem serwera.

**Wymaganie:**
System, jako całość, rejestruje termin użycia karty RFID przez użytkownika wraz z użytym terminalem RFID (klienta) oraz id karty, których użył pracownik.
**Opis:**
System musi rejestrować termin użycia karty, id terminala RFID oraz kartę, których użył pracownik podczas przybycia do miejsca pracy i zeskanowaniu karty RFID przy dowolnym czytniku.

**Wymaganie:**
System rejestruje termin opuszczenia miejsca pracy przez pracownika i terminal RFID, którego użył pracownik.
**Opis:**
System musi rejestrować termin użycia karty, id terminala RFID oraz kartę, którą użył pracownik podczas opuszczenia miejsca pracy i zeskanowaniu karty RFID przy dowolnym czytniku.

**Wymaganie:**
W przypadku użycia nieznanej systemowi karty, system rejestruje jej identyfikator, termin jej użycia i terminal.
**Opis**:
System musi w czasie rzeczywistym rozpoznawać czy zeskanowana karta RFID jest przypisana do użytkownika, w przeciwnym wypadku zeskanowanie rejestrowane jest bez danych pracownika z identyfikatorem, terminem jej użycia oraz id terminala RFID.

# Podstawowe wymagania niefunkcjonalne

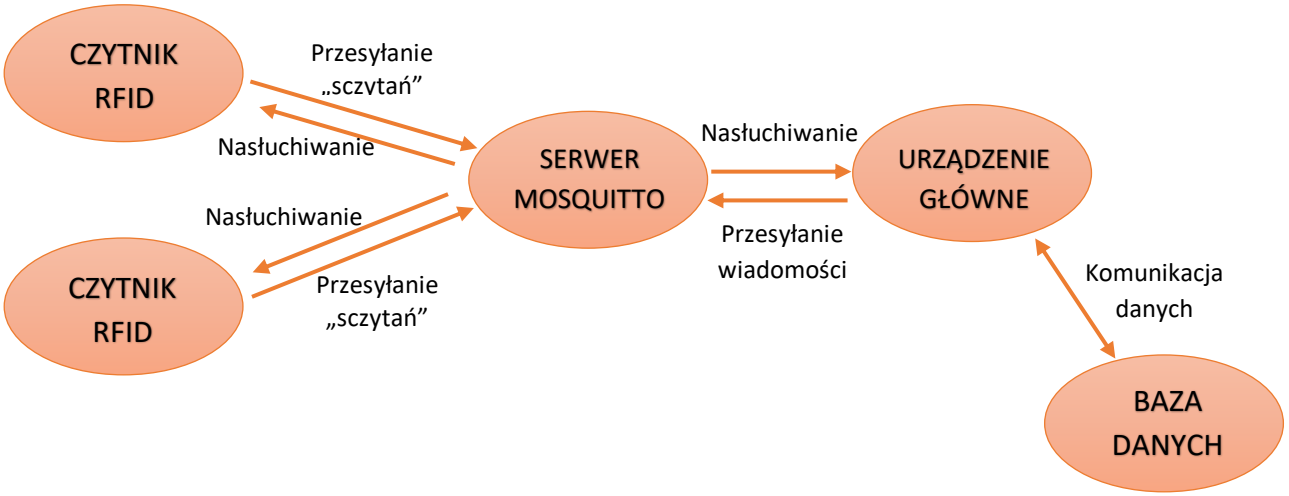
**Wymaganie:**
Obsługa Python v 3
**Opis:**
Aplikacja musi zostać zaimplementowana w języku Python v3 wraz z dodatkiem MQTT z uwagi na zestaw czytników RFID działających na platformie Raspberry Pi 4 obsługujących ten język.

**Wymaganie:**
Małe zapotrzebowanie pamięciowe oraz zasobowe
**Opis:**
Każda z części systemu, podzielona na klientów oraz serwer nie powinny zużywać dużej ilości zasobów i pamięci potrzebnej do pracy oraz nie generować niepotrzebnych plików ze względu na ograniczenia sprzętowe.

**Wymaganie:**
Raporty generowane w sposób łatwy do edycji w programie Office Excel
**Opis:**
Aplikacja serwera musi generować raport czasu pracy w sposób łatwy do ewentualnej edycji oraz obróbki w programie Excel.

# 4. Opis architektury systemu

System składa się z Urządzenia Głównego obsługującego wszystkie zdarzenia przychodzące z urządzeń udostępnionych publicznie (Czytników RFID) odpowiednio je rejestrując oraz zapisując dane w bazie. Odpowiada ono również za wszelkie polecenia wykonane przez użytkownika w interfejsie graficznym. Urządzenie te może jednocześnie obsługiwać Serwer Mosquitto, co nie jest obligatoryjne, może również znajdować się on na urządzeniu zewnętrznym. Czytniki RFID oraz Urządzenie Główne muszą być połączone z istniejącym, wcześniej odpowiednio oraz poprawnie skonfigurowanym, Serwerem Mosquitto za pomocą zdefiniowanego na nim adresie IP oraz porcie. Serwer przetwarza odpowiednio wszystkie komunikaty i przekazuje je adresatowi. Urządzenie Główne po otrzymaniu informacji przesłanej pierwotnie z Czytnika przetwarza je oraz przekazuje w poprawnej formie do zapisu w Bazie Danych. Możliwa jest również komunikacja zwrotna z Urządzenia Głównego, przez Serwer Mosquitto do Czytników RFID wysyłając im różne komunikaty oraz polecenia (jednak nie zostały one uwzględnione w pierwotnych wymaganiach funkcjonalnych).



# 5. Opis implementacji i zastosowanych rozwiązań

## Urządzenie Główne:

**Konfiguracja MQTT na Urządzeniu Głównym:**

```
10  MQTT_BROKER = "                "
11  MQTT_PORT = 8883
12  MQTT_TLS_CRT = 'ca.crt'
13  USERNAME = '       '
14  PASSWORD = '       '
```

W sekcji konfiguracji czytnika należy zawsze ustawić nazwę sieciową urządzenia, które ma służyć jako serwer, które uzyskamy za pomocą Wiersza Polecenia oraz komendy „hostname”. W następnej linijce należy ustawić port 8883. MQTT_TLS_CRT odpowiadający za ścieżkę wcześniej wygenerowanego certyfikatu SSL, który z kolei odpowiada za bezpieczeństwo połączeń. Username i Password zgodnie z nazwą to kolejno login oraz hasło wcześniej utworzonego użytkownika dodanego do pliku konfiguracyjnego Mosquitto.

**Metoda łącząca oraz rozłączająca połączenie Urządzenia Głównego z serwerem Mosquitto**

```
175  def polacz_z_broker():
176      client.tls_set(MQTT_TLS_CRT)
177      client.username_pw_set(username=USERNAME, password=PASSWORD)
178      client.connect(MQTT_BROKER, MQTT_PORT)
179      client.on_message = process_message
180      client.loop_start()
181      client.subscribe("worker/name")
182
183
184  def rozlacz_z_broker():
185      client.loop_stop()
186      client.disconnect()
```

Metoda ta, z wykorzystaniem danych z sekcji konfiguracji opisanej wyżej, łączy się z serwerem Mosquitto. W 179 linijce przekazujemy informację jakiej metody ma użyć Urządzenie Główne podczas otrzymania informacji zaadresowanej właśnie do niego od Serwera. Włączamy nieskończoną pętle aby program mógł stale nasłuchiwać wiadomości oraz ustawiamy nazwę nadawcy, od którego możemy spodziewać się wiadomości, które będzie trzeba przetworzyć.

# Czytnik RFID

**Konfiguracja MQTT na Czytniku RFID:**

```
7   ID_TERMINALA = "T0"
8   MQTT_BROKER = "                "
9   MQTT_PORT = 8883
10  MQTT_TLS_CRT = 'ca.crt'
11  USERNAME = '      '
12  PASSWORD = '       '
```

Konfiguracja czytnika w kodzie wygląda identycznie jak serwera za wyjątkiem pierwszej linijki. Ustawiamy w niej unikatową nazwę naszego terminala (czytnika RFID), która będzie przekazywana wraz z sczytaniem karty do bazy danych oraz raportów. Kolejne linijki to nazwa sieciowa urządzenia, na którym włączony jest czytnik, port, ścieżka do certyfikatu, login i hasło użytkownika czyli stworzony specjalny profil uwierzytelniający w sieci dla czytnika.

**Metoda łącząca Czytnik RFID z serwerem Mosquitto**

```
42  def polacz_z_broker():
43      client.tls_set(MQTT_TLS_CRT)
44      client.username_pw_set(username=USERNAME, password=PASSWORD)
45      client.connect(MQTT_BROKER, MQTT_PORT)
46      poinformuj_serwer("Client połączył się")
47      client.on_message = process_message
48      client.loop_start()
49      client.subscribe("server/name")
```

Metoda również analogicznie podobna do tej z serwera, korzysta ona z konfiguracji zdefiniowanej wyżej. Dodatkowo jednak korzystamy z komunikatu pozwalającego osobie korzystającej z Urządzenia Głównego zauważyć, że jeden z Clientów, czyli czytników połączył się z siecią. Kolejne linijki metody mają identyczne działanie jak wcześniej.

**Fragment kodu czytnika odpowiadający za symulowanie sczytań:**

```
18  def poinformuj_serwer(informacja):
19      client.publish("czytnik", informacja + "." + ID_TERMINALA, )
20
21
22  def stworz_glowne_okno():
23      window.geometry("400x320")
24      window.title("Czytnik kart RFID")
25
26      tkinter.Label(window, text="Wybierz symulacje zczytania karty:",
27                    font=("Sans-serif", 14, "italic")).pack(pady=20)
28      tkinter.Button(window, text="325643567864",
29                     command=lambda: poinformuj_serwer("325643567864")).pack(pady=3)
30      tkinter.Button(window, text="324685456443",
31                     command=lambda: poinformuj_serwer("324685456443")).pack(pady=3)
32      tkinter.Button(window, text="345678754653",
33                     command=lambda: poinformuj_serwer("345678754653")).pack(pady=3)
34      tkinter.Button(window, text="375689657564",
35                     command=lambda: poinformuj_serwer("375689657564")).pack(pady=3)
36      tkinter.Button(window, text="Losowe ID Karty",
37                     command=lambda: poinformuj_serwer(str(randint(300000000000, 399999999999)))).pack(pady=3)
38
39      tkinter.Button(window, text="Wyłącz czytnik", command=window.quit).pack(pady=20)
```

Metoda ta została stworzona specjalnie na potrzeby testowe programu. Korzystając z pięciu stworzonych przycisków w naszym GUI (interfejsie graficznym) możemy zasymulować sczytanie karty wcześniej przypisanej do jednego z pracowników lub wykorzystać możliwość wygenerowania dowolnego ID karty (ze względu na możliwość wygenerowania stu miliardów różnych konfiguracji numerów ID istnieje znikoma szansa, że wygenerowany testowo numer ID jest przypisany do karty któregokolwiek z pracowników).

# 6. Opis działania i prezentacja interfejsu

## Instalacja potrzebnych zasobów

W celu poprawnego uruchomienia programu należy wykonać w wyznaczonej kolejności polecenia umieszczone w instrukcji, która została zawarta w pierwszym [1] oraz drugim [2] punkcie literatury z wyłączeniem edycji kodu programu. W skrócie jest to instalacja Brokera MQTT Mosquitto, certyfikatu OpenSSL, generowanie certyfikatów do komunikacji TLS oraz konfiguracja Mosquito w celu zapewnienia szyfrowania danych MQTT przesyłanych przez sieć. W celu zastosowania dokonanych zmian będzie konieczny restart skryptów lub ponowne uruchomienie komputera. Z moich obserwacji wynika, że podczas wykonywania tych poleceń z instrukcji nie zawsze Mosquitto uruchamia się z zastosowaniem plików konfiguracyjnych. W tym celu należy stworzyć własny plik uruchamiający Mosquitto:

```
cd C:\Program Files\mosquitto\
mosquitto -c mosquitto.conf -v
```

Zapisując go z rozszerzeniem '.bat', dla jaśniejszego przekazu będę go nazywał plikiem uruchamiającym Mosquitto.

## Uruchomienie stworzonego projektu

W pierwszej kolejności należy uruchomić stworzony w kroku wyżej plik uruchamiający Mosquitto z rozszerzeniem '.bat'. Następnie w zachowanej kolejności uruchomić plik o nazwie Serwer na urządzeniu, które będzie służyło jako serwer. Najlepszym rozwiązaniem będzie pozostawienie przy nim pliku obsługującego bazę danych. Każdy z czytników należy zaopatrzyć w poprawnie skonfigurowany plik o nazwie Client.py oraz go uruchomić.

## Możliwe błędy podczas uruchamiania

**Błąd 1**

Podczas uruchamiania pliku Serwer.py otrzymujemy podobny błąd:


Błąd świadczy o braku możliwości połączenia z Serwerem Mosquitto.
Należy uruchomić stworzony plik uruchamiający Mosquitto i spróbować ponownie kompilować plik Serwer.py.

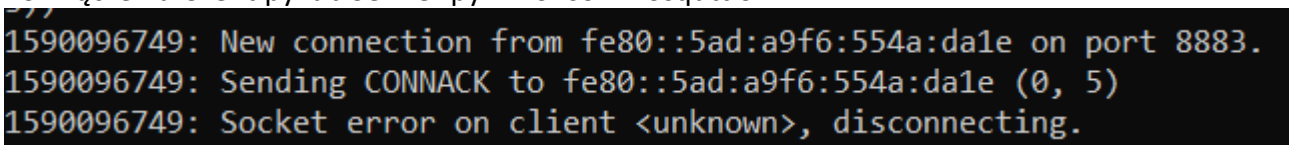
**Błąd 2**

Podczas włączania Client.py


Błąd świadczy o braku możliwości połączenia z Serwerem Mosquitto.
Należy uruchomić stworzony plik uruchamiający Mosquitto i spróbować ponownie skompilować plik Client.py.

**Błąd 3**

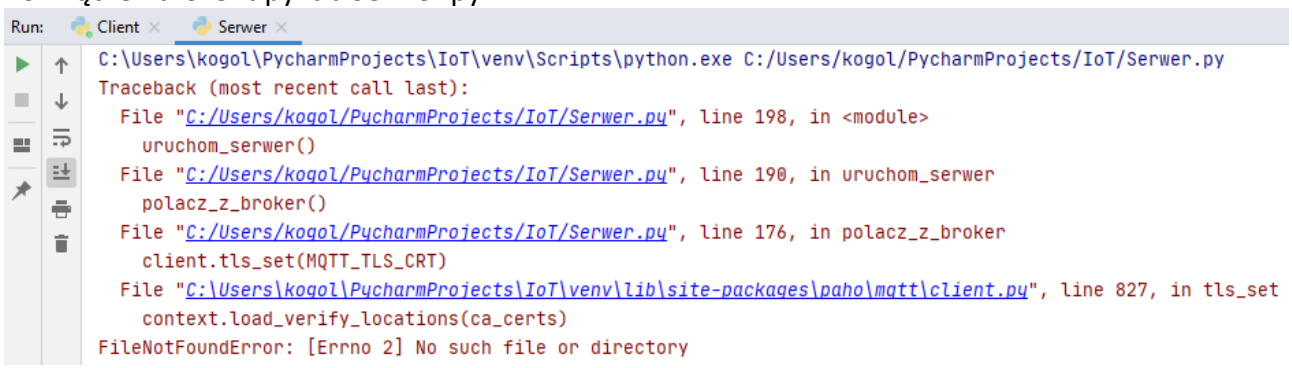
Po włączeniu Client.py lub Serwer.py w konsoli Mosquttio




Błąd świadczy o braku poprawności danych do logowania jednego lub wielu z użytkowników lub niepoprawnego certyfikatu ca.crt

**Błąd 4**

Po włączeniu Client.py lub Serwer.py




Niepoprawnie podana ścieżka do certyfikatu '.crt'

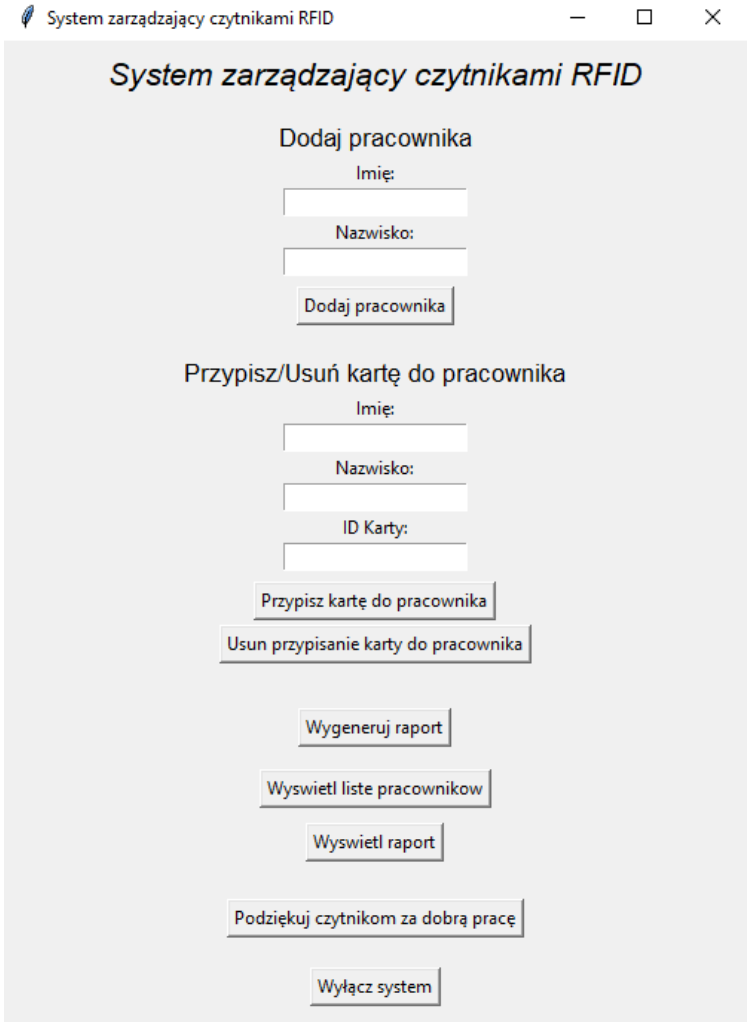
**Błąd 5**

Po włączeniu Client.py lub Serwer.py


Zbliżony treścią błąd występuje podczas próby uruchomienia programu z niewłaściwą nazwą brokera MQTT. Należy pobrać właściwą nazwę sieciową urządzenia przez wpisanie w Wierszu Poleceń komendy ‘hostname’.

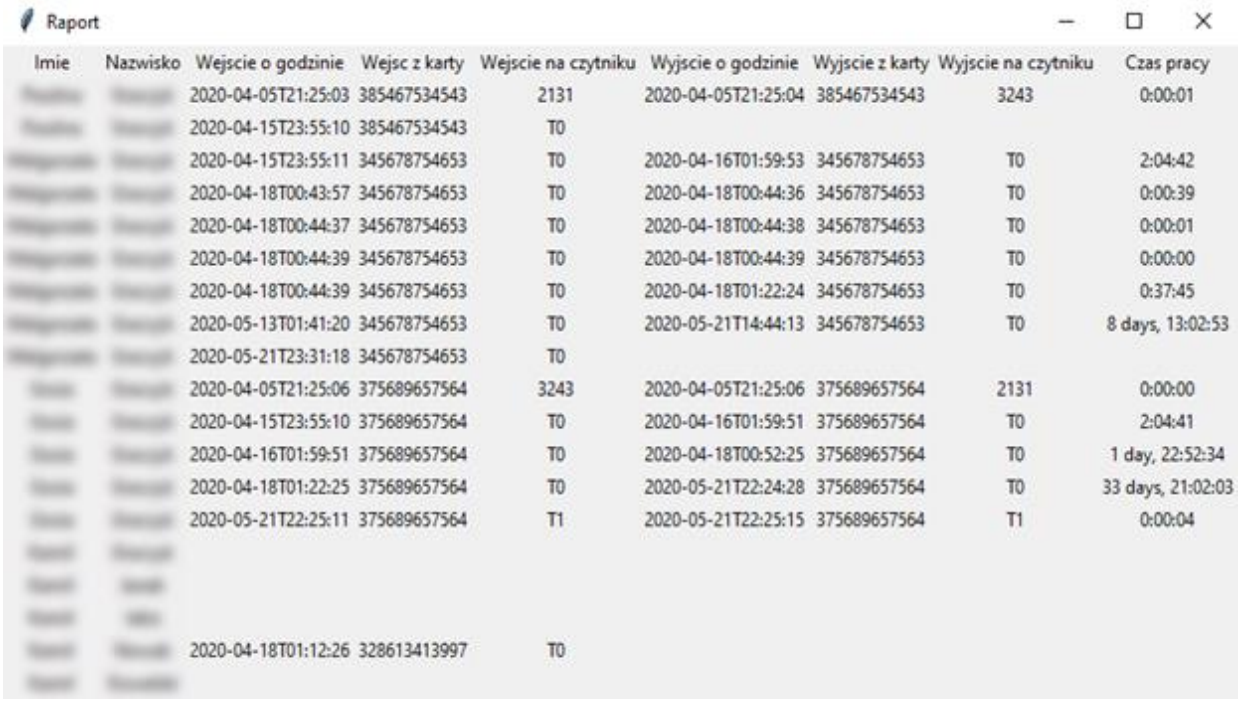
# Poprawnie uruchomiony program Urządzenia Głównego

Poprawnie uruchomione Urządzenie Główne wyglądać będzie następująco:



W wyświetlonym interfejsie użytkownik ma możliwość:

- dodania nowego pracownika podając jego imię i nazwisko oraz zatwierdzając przyciskiem „dodaj pracownika”
- przypisanie lub usunięcie karty pracownika poprzez wpisanie imienia, nazwiska oraz ID karty - potwierdzenie odbywa się przez wciśnięcie właściwego przycisku do zamierzonej akcji
- wygenerowania raportu w celu sprawdzenia jego w aplikacji poprzez wciśnięcie poniższego przycisku „Wyświetl raport”:




| Imie | Nazwisko | Wejscie o godzinie | Wejsc z karty | Wejscie na czytniku | Wyjscie o godzinie | Wyjscie z karty | Wyjscie na czytniku | Czas pracy |
|---|---|---|---|---|---|---|---|---|
| [redacted] | [redacted] | 2020-04-05T21:25:03 | 385467534543 | 2131 | 2020-04-05T21:25:04 | 385467534543 | 3243 | 0:00:01 |
| [redacted] | [redacted] | 2020-04-15T23:55:10 | 385467534543 | T0 | | | | |
| [redacted] | [redacted] | 2020-04-15T23:55:11 | 345678754653 | T0 | 2020-04-16T01:59:53 | 345678754653 | T0 | 2:04:42 |
| [redacted] | [redacted] | 2020-04-18T00:43:57 | 345678754653 | T0 | 2020-04-18T00:44:36 | 345678754653 | T0 | 0:00:39 |
| [redacted] | [redacted] | 2020-04-18T00:44:37 | 345678754653 | T0 | 2020-04-18T00:44:38 | 345678754653 | T0 | 0:00:01 |
| [redacted] | [redacted] | 2020-04-18T00:44:39 | 345678754653 | T0 | 2020-04-18T00:44:39 | 345678754653 | T0 | 0:00:00 |
| [redacted] | [redacted] | 2020-04-18T00:44:39 | 345678754653 | T0 | 2020-04-18T01:22:24 | 345678754653 | T0 | 0:37:45 |
| [redacted] | [redacted] | 2020-05-13T01:41:20 | 345678754653 | T0 | 2020-05-21T14:44:13 | 345678754653 | T0 | 8 days, 13:02:53 |
| [redacted] | [redacted] | 2020-05-21T23:31:18 | 345678754653 | T0 | | | | |
| [redacted] | [redacted] | 2020-04-05T21:25:06 | 375689657564 | 3243 | 2020-04-05T21:25:06 | 375689657564 | 2131 | 0:00:00 |
| [redacted] | [redacted] | 2020-04-15T23:55:10 | 375689657564 | T0 | 2020-04-16T01:59:51 | 375689657564 | T0 | 2:04:41 |
| [redacted] | [redacted] | 2020-04-16T01:59:51 | 375689657564 | T0 | 2020-04-18T00:52:25 | 375689657564 | T0 | 1 day, 22:52:34 |
| [redacted] | [redacted] | 2020-04-18T01:22:25 | 375689657564 | T0 | 2020-05-21T22:24:28 | 375689657564 | T0 | 33 days, 21:02:03 |
| [redacted] | [redacted] | 2020-05-21T22:25:11 | 375689657564 | T1 | 2020-05-21T22:25:15 | 375689657564 | T1 | 0:00:04 |
| [redacted] | [redacted] | | | | | | | |
| [redacted] | [redacted] | | | | | | | |
| [redacted] | [redacted] | | | | | | | |
| [redacted] | [redacted] | 2020-04-18T01:12:26 | 328613413997 | T0 | | | | |
| [redacted] | [redacted] | | | | | | | |

lub uruchomienie wygenerowanego pliku ‘.txt’ w programie Excel wybierając wstążkę ‘Dane’, a następnie ‘Z tekstu’. Wczytane dane pojawią się w postaci sformatowanej tabeli, a w niej:

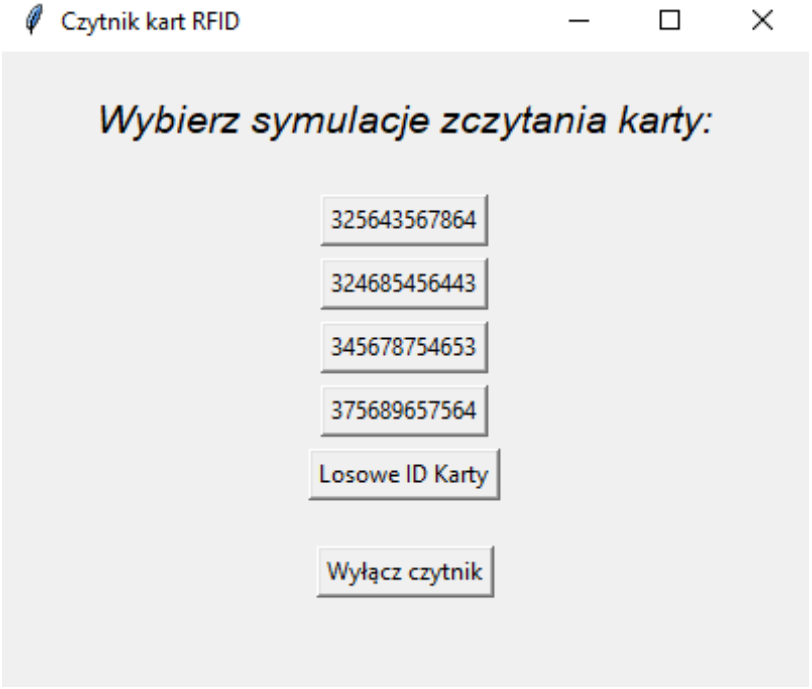
| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Imie | Nazwisko | Wejscie o godzinie | Wejsc z karty | Wejscie na czytniku | Wyjscie o godzinie | Wyjscie z karty | Wyjscie na czytniku | Czas pracy |
| 2 | [redacted] | [redacted] | 2020-04-05T21:25:03 | 385467534543 | 2131 | 2020-04-05T21:25:04 | 385467534543 | 3243 | 00:00:01 |
| 3 | [redacted] | [redacted] | 2020-04-15T23:55:10 | 385467534543 | T0 | | | | |
| 4 | [redacted] | [redacted] | 2020-04-15T23:55:11 | 345678754653 | T0 | 2020-04-16T01:59:53 | 345678754653 | T0 | 02:04:42 |
| 5 | [redacted] | [redacted] | 2020-04-18T00:43:57 | 345678754653 | T0 | 2020-04-18T00:44:36 | 345678754653 | T0 | 00:00:39 |
| 6 | [redacted] | [redacted] | 2020-04-18T00:44:37 | 345678754653 | T0 | 2020-04-18T00:44:38 | 345678754653 | T0 | 00:00:01 |
| 7 | [redacted] | [redacted] | 2020-04-18T00:44:39 | 345678754653 | T0 | 2020-04-18T00:44:39 | 345678754653 | T0 | 00:00:00 |
| 8 | [redacted] | [redacted] | 2020-04-18T00:44:39 | 345678754653 | T0 | 2020-04-18T01:22:24 | 345678754653 | T0 | 00:37:45 |
| 9 | [redacted] | [redacted] | 2020-05-13T01:41:20 | 345678754653 | T0 | 2020-05-21T14:44:13 | 345678754653 | T0 | 8 days, 13:02:53 |
| 10 | [redacted] | [redacted] | 2020-05-21T23:31:18 | 345678754653 | T0 | | | | |
| 11 | [redacted] | [redacted] | 2020-04-05T21:25:06 | 375689657564 | 3243 | 2020-04-05T21:25:06 | 375689657564 | 2131 | 00:00:00 |
| 12 | [redacted] | [redacted] | 2020-04-15T23:55:10 | 375689657564 | T0 | 2020-04-16T01:59:51 | 375689657564 | T0 | 02:04:41 |
| 13 | [redacted] | [redacted] | 2020-04-16T01:59:51 | 375689657564 | T0 | 2020-04-18T00:52:25 | 375689657564 | T0 | 1 day, 22:52:34 |
| 14 | [redacted] | [redacted] | 2020-04-18T01:22:25 | 375689657564 | T0 | 2020-05-21T22:24:28 | 375689657564 | T0 | 33 days, 21:02:03 |
| 15 | [redacted] | [redacted] | 2020-05-21T22:25:11 | 375689657564 | T1 | 2020-05-21T22:25:15 | 375689657564 | T1 | 00:00:04 |
| 16 | [redacted] | [redacted] | | | | | | | |
| 17 | [redacted] | [redacted] | | | | | | | |
| 18 | [redacted] | [redacted] | | | | | | | |
| 19 | [redacted] | [redacted] | 2020-04-18T01:12:26 | 328613413997 | T0 | | | | |
| 20 | [redacted] | [redacted] | | | | | | | |

- wyświetlenia graficznego listy wszystkich pracowników wraz z przypisanymi do nich kartami:


- wysłania wiadomości z podziękowaniami dla czytników RFID za dobrą prace (na czytniku wyświetlany jest komunikat „Urządzenie główne dziękuje za dobrą pracę”
- wyłączenia programu na Urządzeniu Głównym

## Poprawnie uruchomiony program Czytnika RFID

Poprawnie uruchomiony Czytnik RFID (gdy jest możliwość na urządzeniu spojrzenia na interfejs graficzny, w założeniu projektu miał on być na Raspberry Pi, lecz obecnie uruchamiamy go na komputerze dlatego możemy spojrzeć na GUI):




W tym prostym interfejsie znajdziemy cztery przyciski odpowiadające za sczytanie z góry zaprogramowanych numerów kart. Niektóre są już przypisane do pracowników, niektóre istnieją tylko jako karty, bez pracownika. Piąty przycisk służy do wygenerowania losowego ID karty z przedziału 300000000000 – 399999999999. Są to zakresy numerów rzeczywistych kart RFID wykorzystywanych w projekcie.

Ostatni przycisk służy do wyłączenia Czytnika RFID i zakończenia połączenia z Serwerem Mosquitto.

# 7. Podsumowanie

W projekcie zostały zaimplementowane wszystkie wstępne wymagania projektowe, zarówno funkcjonalne jak i te niefunkcjonalne. Projekt został wzbogacony o niewymaganą skromną szatę graficzną, aby ułatwić możliwość przedstawienia funkcji systemu oraz ułatwić korzystanie.

W projekcie zgodnie z pierwotnym założeniem zabrakło możliwości powiązania programu z rzeczywistym czytnikiem kart RFID, co zostało przeze mnie uzupełnione symulatorem sczytań takich kart. Implementacja takiego modułu według informacji zdobytych w Internecie wymagałaby zmiany jednej lub dwóch linijek kodu.

W trakcie pracy nad systemem największym problemem była instalacja modułu MQTT i jego implementacja w kodzie, ponieważ nie znalazłem w źródłach pisanych i elektronicznych, a także w instrukcjach informacji, że Mosquitto uruchamia się bez plików konfiguracyjnych. Musiałem sam dojść do wniosku, że należy uruchamiać go specjalną komendą w Wierszu Poleceń CMD lub utworzyć własny plik '.bat'. Poziom trudności projektu, a dokładnie jego część funkcjonalna był dosyć wysoki, jego utworzenie stanowiło dla mnie wyzwanie. Początkowo dużym wyzwaniem był nowy język programowania, którego nie poznaliśmy wcześniej w toku studiów, czyli język Python.

# 8. Literatura


1. https://eportal.pwr.edu.pl/pluginfile.php/297452/mod_resource/content/2/MQTT%20-%20security%20-%20PL%20v2.pdf
2. https://eportal.pwr.edu.pl/pluginfile.php/304348/mod_resource/content/1/MQTT%20-%20security%20-%20authentication%20%20authorization%20-%20PL.pdf
3. https://docs.python.org/3/library/tk.html
4. https://docs.python.org/3/
5. https://1380662.netacad.com/courses/999206 [2020l_PEinpython_pl]
6. https://mosquitto.org/download/
7. https://slproweb.com/download/Win64OpenSSL_Light-1_1_1g.exe

# 9. Aneks

- Kod zaimplementowanego projektu w formie elektronicznej spakowany do pliku ‘.rar’