Dokumentacja projektu aplikacji mobilnej

OBD II Reader

Autorzy

Konrad Bielecki

Dominik Gruew

Spis treści:

Autorzy projektu

Konrad Bielecki
Dominik Gruew

Cel projektu

Celem projektu jest stworzenie aplikacji mobilnej, umożliwiającej bezprzewodowe połączenie urządzenia typu smartphone z pojazdem wyposażonym w port komunikacyjny OBD II. Dodatkowo, aby nawiązać połączenie niezbędne jest wykorzystanie modułu bluetooth typu ELM 327, który wpinamy bezpośrednio do portu OBD II. Interfejs ELM 327 „czyta" dane sterownika pojazdu (ECU), np. obroty silnika, obciążenie silnika, temperatura płyny chodzącego itp. Dokładność i dostępność czytanych danych uzależniona jest tylko i wyłącznie od pojazdu z którym się komunikujemy. Dane mogą być niekompletne jeśli ECU pojazdu nie obsługuje wysłanego z aplikacji komunikatu (PID).
Pełną dokumentacje interfejsu ELM 327 możemy znaleźć na stronie klikając w link.

Wykorzystane technologie

Aplikacja powstała z wykorzystaniem środowiska programistycznego Android Studio.
Użyty język to Java.
Urządzenie zewnętrzne – interfejs komunikacyjny ELM 327.
Dołączona biblioteka SpeedView do obsługi liczników danych w czasie rzeczywistym.

Opis projektu

Zasada działania

Główną zasadą działania aplikacji jest wysyłanie i odbieranie informacji ze sterownika pojazdu (ECU). Aby tego dokonać należy zapoznać się z działaniem komunikacji poprzez wysyłanie wiadomości PID (Parameter ID).

Dla przykładu, jeśli chcemy sprawdzić aktualny stan temperatury płynu chłodzącego, sprawdzamy jaki PID odpowiada tej wartości.
Lista PID'ów dostępna jest na stronie OBD II PIDs

PID (hex)	PID (Dec)	Data bytes returned	Description	Min value	Max value	Units	Formula[a]
05	5	1	Engine coolant temperature	-40	215	°C	$A - 40$

Ten PID pochodzi z serwisu 01 (odczyt danych rzeczywistych), a więc w wysyłamy PID 010C.
Otrzymana odpowiedź jest wartością szesnastkową i ma postać np. 12 41 0C XX YY 1A
Pod uwagę bierzemy 0C, która oznacza, że odpowiedź rzeczywiście dotyczy temperatury płynu chłodzącego.

Wartość XX oznaczamy jako A.
Wartość YY oznaczamy jako B.

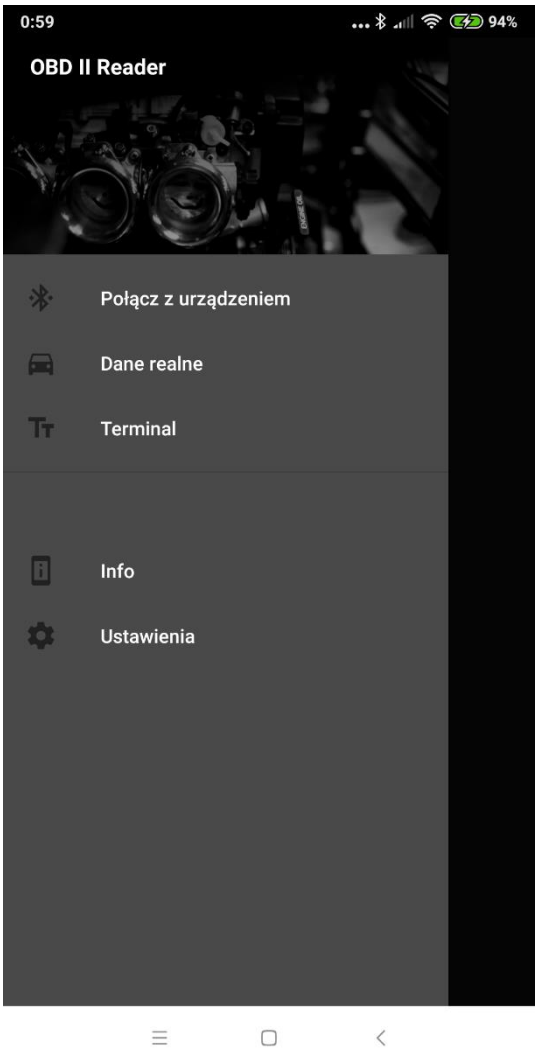
Korzystamy z formuły A – 40, która jest wartością temperatury w skali od -40 °C, do 215 °C.
Np. wartość A przyjmijmy jako 4F, przeliczamy na liczbę dziesiętną czyli 79, następnie korzystając z formuły musimy odjąć 40, zatem odczytana wartość to 39 °C.

Aplikacja wskazuje wartości sterownika (ECU) pojazdu, więc jest narzędziem diagnostycznym. Należy jednak liczyć się z faktem, że nie posiada pełnej obsługi pojazdu. Wybrane zostały wartości, które są najczęściej używane i kontrolowane podczas kontroli pojazdu.

Aplikacja posiada dwie główne funkcje: **terminal** i **dane realne**.

Nawigacja aplikacji

W aplikacji został użyty kontener NavigationView, dzięki któremu łatwo możemy przemieszczać się pomiędzy poszczególnymi funkcjami aplikacji.

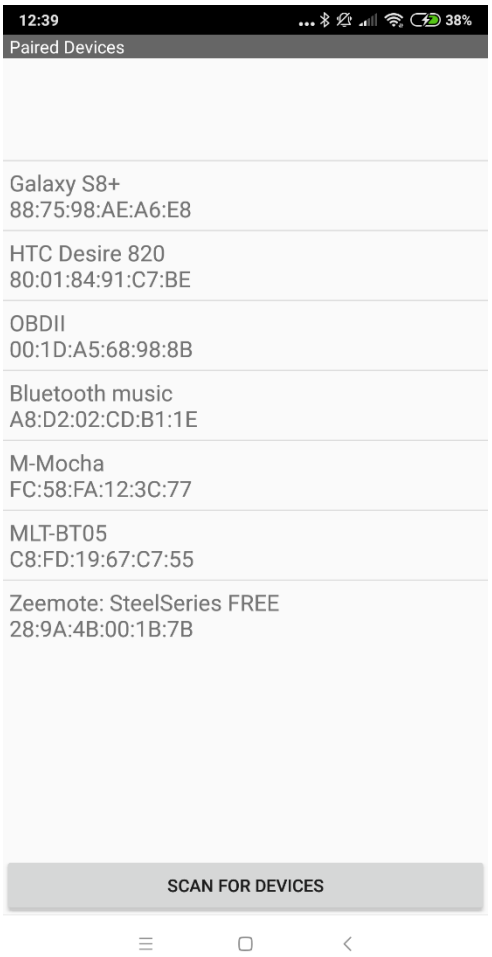


Połączenie aplikacji z urządzeniem ELM 327

Połączenie odbywa się poprzez wykorzystanie standardowych metod implementacji kodu do komunikacji pomiędzy urządzeniami wyposażonymi w moduł bluetooth.
Ważne jest ustawienie odpowiedniego UUID (Universal Unique Identifier), który jest 128 bitowym identyfikatorem obiektu, bądź urządzenia.

```
// Unique UUID for this application
//ELM327 UUID
private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
```

Aby połączyć się z urządzeniem ELM 327, w menu aplikacji klikamy „Połącz z urządzeniem”, następnie wyświetlona zostaje lista sparowanych urządzeń. Zazwyczaj interfejs ELM 327 znajdujemy pod nazwą „OBDII”.
Jeśli na liście nie znajdziemy interesującego nas urządzenia, używamy opcji „Wyszukiwanie urządzeń”.
Próba nawiązania połączenia następuje po kliknięciu na dane urządzenie.
Jeżeli otrzymamy komunikat o nieudanym połączeniu, należy sprawdzić czy interfejs ELM 327 został odpowiednio wpięty do złącza OBD II.
Jeśli interfejs był wcześniej używany, należy go odłączyć i podłączyć ponownie, aby zresetować dane sesji.

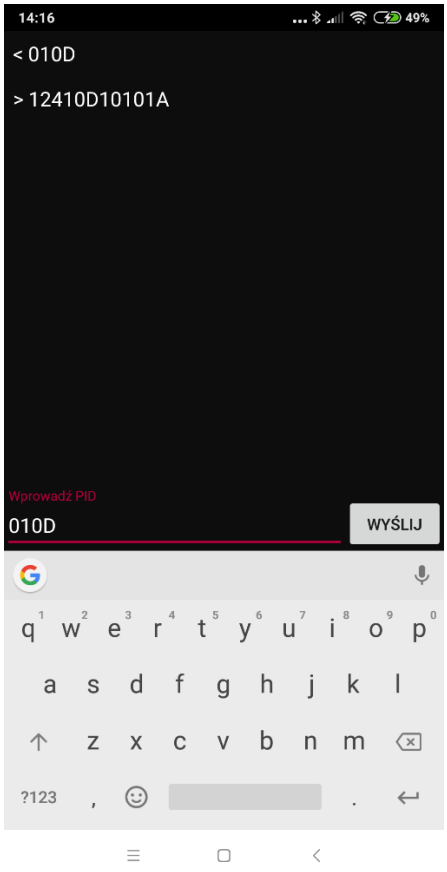


Terminal

Terminal umożliwia nam wysłanie dowolnej wiadomości PID, po czym jednokrotnie otrzymujemy nieprzetworzoną wiadomość zwrotną. Jest to przydatne narzędzie do prac warsztatowych, odpowiedzi mają duże znaczenie od warunków pracy pojazdu, warunków otoczenia itp.

Wiadomości które odczytuje terminal są filtrowane w taki sposób, aby otrzymać ciąg znaków który najbardziej nas interesuje. Wszystkie niekompletne wiadomości zostają pominięte.

Dodatkowo terminal daje nam możliwość zmiany ustawień interfejsu ELM 327 poprzez wprowadzenie odpowiednich komend AT. Standardowo aplikacja podczas uruchomienia połączenia przestawia wartości ATSP na 0, co oznacza, że interfejs ELM 327 wraca do ustawień fabrycznych.



Dane realne

Dane realne jest to zbiór wartości odczytywanych w czasie trwania sesji połączenia aplikacji z pojazdem. Wartości są odpowiednio przeliczane i przedstawiane w formie liczników bądź formie tekstowej. Jeśli w trakcie sesji poszczególne wartości nie zostaną uzupełnione, oznacza to, że pojazd nie obsługuje danej komendy PID.

Lista danych pobieranych z ECU	
RPM	Obroty silnika
Speed	Prędkość pojazdu
Coolant Temperature	Temperatura płynu chłodzącego
Intake Air Temperature	Temperatura powietrza w kolektorze dolotowym
MAF	Czujniki przepływu powietrza MAF
Throttle Position	Pozycja gazu
Control Module Voltage	Ładowanie akumulatora V
Engine Load	Obciążenie silnika
Short term fuel trim—Bank 1	Krótkotrwałe czasy wtrysków—Bank 1
Long term fuel trim—Bank 1	Długotrwałe czasy wtrysków—Bank 1
Short term fuel trim—Bank 2	Krótkotrwałe czasy wtrysków—Bank 1
Long term fuel trim—Bank 2	Długotrwałe czasy wtrysków—Bank 1
Fuel Pressure	Ciśnienie paliwa
Intake Manifold Absolute Pressure	Ciśnienie absolutne kolektora dolotowego
Timing Advance	Czas wyprzedzenia
Run time since engine start	Czas od uruchomienia silnika
Distance traveled with malfunction indicator lamp (MIL) on	Odległość przebyta z lampką kontrolną awarii
Fuel Rail Pressure	Cieśnieni na listwie wtryskowej
EGR	Stan EGR
Fuel Level	Poziom paliwa
Absolute Barometric Pressure	Całkowite ciśnienie barometryczne
RelativeThrottlePosition	Relatywna pozycja pedału gazu
Engine Oil Temperature	Temperatura oleju
VIN	nr identyfikacyjny VIN

Wszystkie komendy zapisane są w klasie OBDCommands i zostają przeliczone. Przykładowo dla obrotów silnika tworzymy komendę RPMCommand, która przyjmuje wartość String „010C”. Jest to komenda która zostaje wysyłana w pętli do ECU.

Otrzymując odpowiedź musimy sprawdzić czy znajduje się w niej ciąg znaków odpowiadający wysłanemu zapytaniu. w naszym przypadku jest to „410C”, które zapisaliśmy jako RPMPidEncoder. Aby odpowiednio odczytać dane, należy je przekalkulować używając metody calculate oraz według wzoru z tabeli PID’ów, w ten sposób ustawiamy currentRPM.

Metoda calculate dobiera odpowiedni ciąg znaków z odebranej wiadomości i przelicza ich wartość z szesnastkowego na dziesiętny.

```
//Gets result from response and calculate Hex to Dec
public static int calculate (String r, String pidEncoder, int resultLastIndex){
    r = r.substring(r.indexOf(pidEncoder) + 4);
    r = r.substring(0,resultLastIndex);
    return Integer.parseInt(r,  radix: 16);
}
```

```
public static String RPMCommand = "010C";
public static String RPMPidEncoder = "410C";
public static String currentRPM = "";
public static void setRPM (String result){
    currentRPM = String.valueOf(calculate(result,RPMPidEncoder, resultLastIndex: 4) / 4);
}
public static String getRPM () { return currentRPM; }
```

W momencie uruchomienia Real Data, następuje uruchomienie działania pętli w której wysyłane zostają PID’y. Wysyłane zostają jeden po drugim z opóźnieniem 200ms. Opóźnienie jest niezbędne z powodów technicznych. Interfejs OBD II obsługuje jedną komendę w danym czasie.

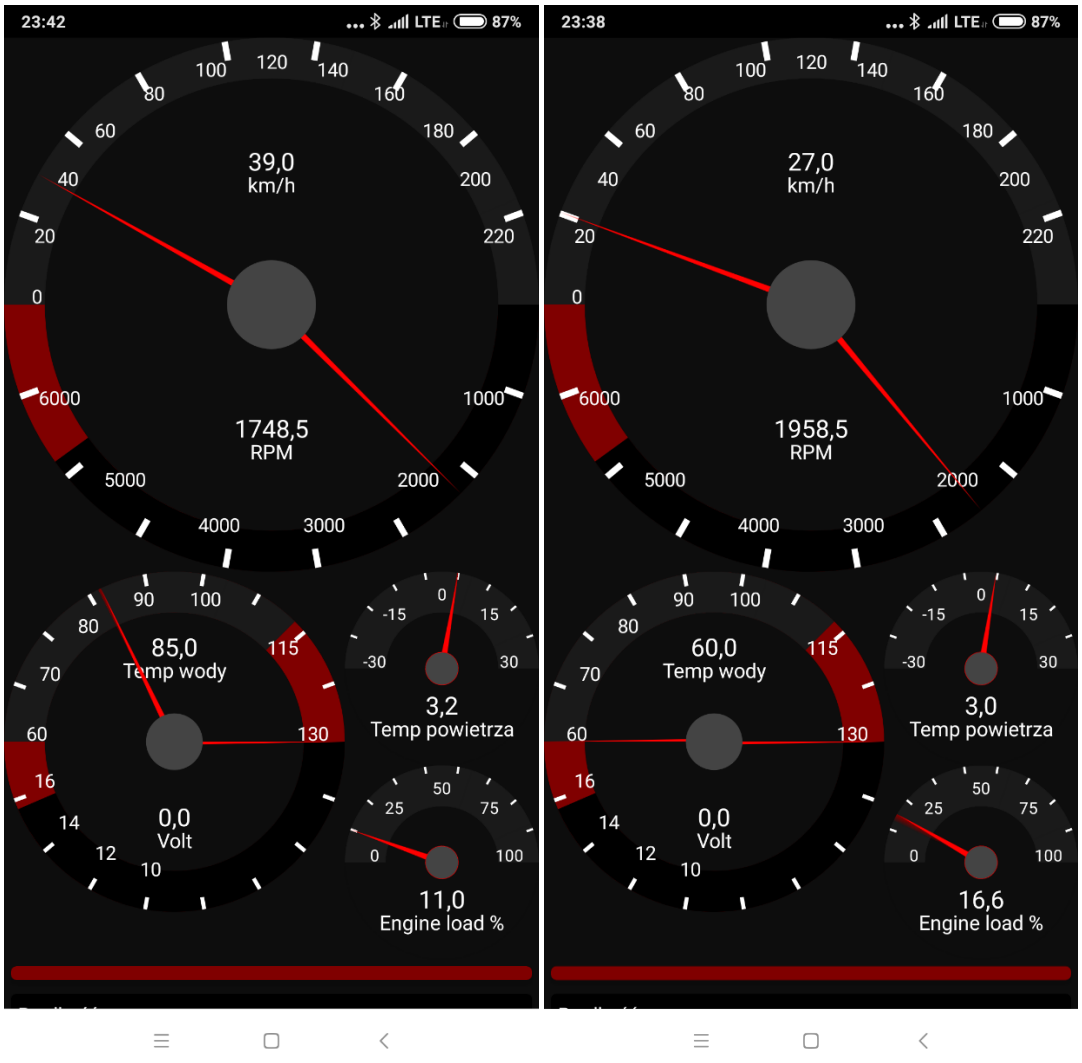
```
try {
    if (Thread.interrupted()) { throw new InterruptedException();}
    while (!Thread.currentThread().isInterrupted()) {
        BluetoothChat.sendMessage(OBDCommands.RPMCommand);
        delay();
        BluetoothChat.sendMessage(OBDCommands.CoolantTempCommand);
        delay();
        BluetoothChat.sendMessage(OBDCommands.SpeedCommand);
        delay();
        BluetoothChat.sendMessage(OBDCommands.MAFCommand);
        delay();
        BluetoothChat.sendMessage(OBDCommands.ThrottlePositionCommand);
        delay();
```

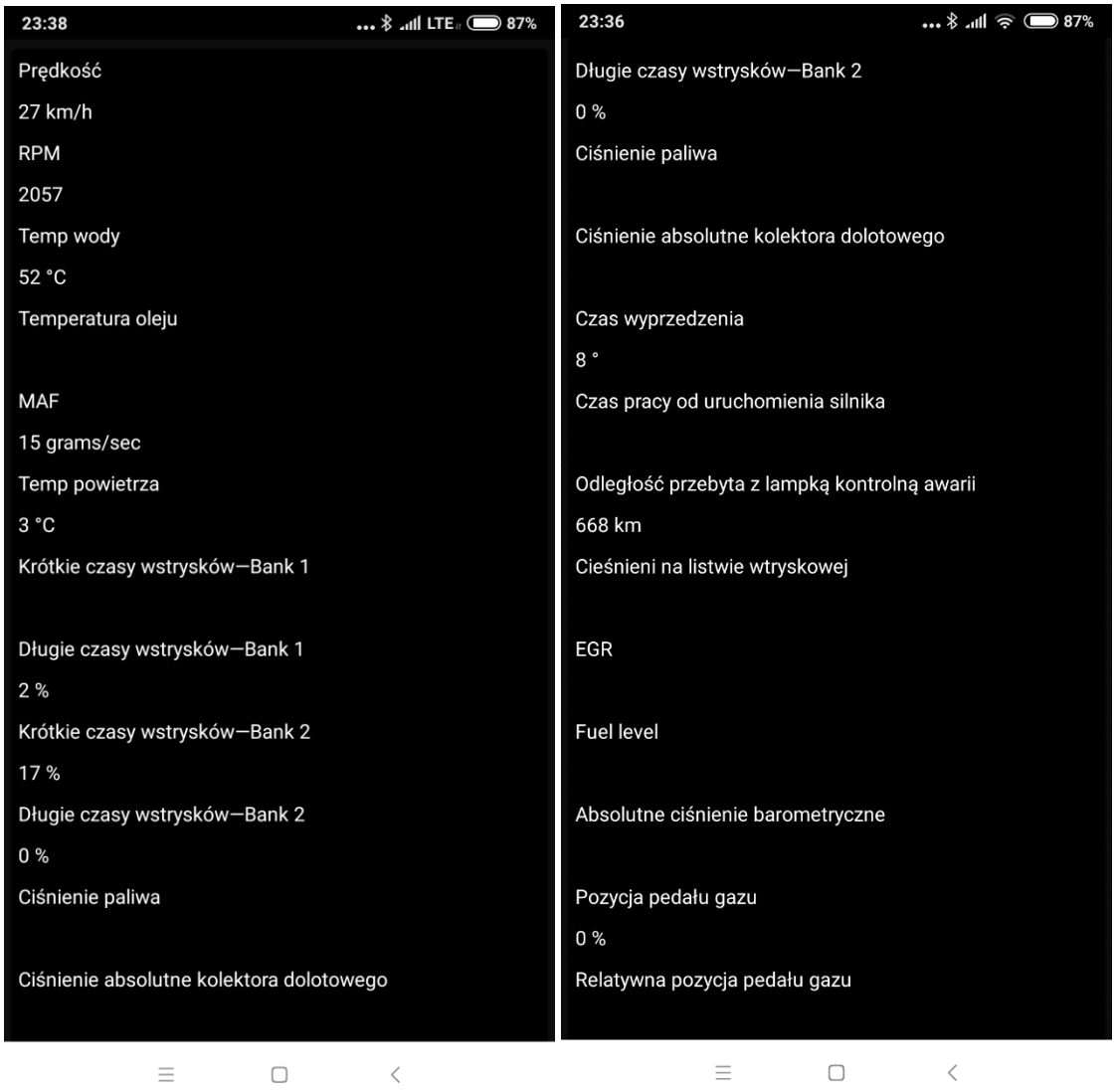
Użycie komend następuje w klasie głównej w momencie przechwycenia wiadomości zwrotnej. PidEncoder rozpoznaje użytą komendę i przeliczone dane przekazuje do licznika, bądź listy danych.

```
    //RPM - ok
if (readMessage.contains(OBDCommands.RPMPidEncoder)) {
    OBDCommands.setRPM(readMessage);
    RealData.rpmView.speedTo(Integer.parseInt(OBDCommands.getRPM()), moveDuration: 500);
    RealData.rpmTextView.setText(OBDCommands.getRPM());

    //Coolant Temp - ok
} else if (readMessage.contains(OBDCommands.CoolantTempPidEncoder)) {
    OBDCommands.setCoolantTemp(readMessage);
    RealData.coolantTempView.speedTo(Integer.parseInt(OBDCommands.getCoolantTemp()));
    RealData.coolantTextView.setText(OBDCommands.getCoolantTemp() + " °C");
```

Poniżej przykład wyników użycia aplikacji:

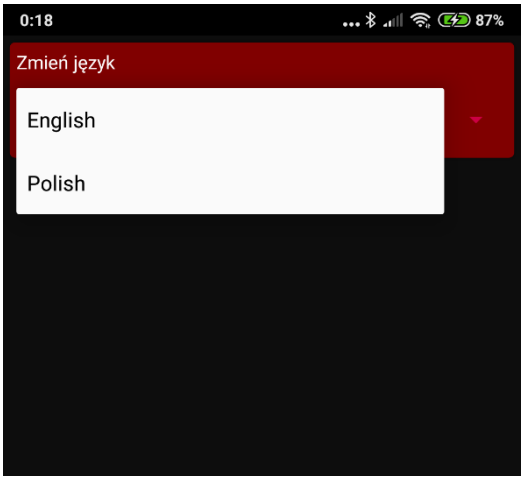




Jak widać samochód na którym przeprowadzono test nie obsługuje wszystkich komend, co skutkuje pustymi miejscami w liście, bądź licznikami z wartościami ustawionymi na minimum.

Ustawienia

W opcjach możemy znaleźć jedynie zmianę języka aplikacji.
Do wyboru mamy język Polski lub Angielski.




```
public static void SetLanguage(String lang) {
    if(GetLanguage() == lang) { return; }
    Locale myLocale = new Locale(lang);
    Resources res = _Instance.getResources();
    DisplayMetrics dm = res.getDisplayMetrics();
    Configuration conf = res.getConfiguration();
    conf.locale = myLocale;
    res.updateConfiguration(conf, dm);
    Intent refresh = new Intent(_Instance.getBaseContext(), BluetoothChat.class);
    _Instance.startActivity(refresh);
    _Instance.finish();
}
public static String GetLanguage()
{
    Locale x = _Instance.getResources().getConfiguration().locale;
    return x.getLanguage();
}
```

Biblioteka SpeedView

Dzięki wykorzystaniu biblioteki SpeedView do tworzenia liczników w aplikacjach mobilnych, mamy możliwość udoskonalenia wglądu w dane czytane przez naszą aplikacje. Poniższy zrzut ekranu obrazuje efekt użycia biblioteki, dokonywane zmiany są możliwe prawie na każdej płaszczyźnie. Dla porównania po prawej standardowy licznik oferowany przez bibliotekę.