Programowanie Obiektowe

Projekt: Elektroniczny barman

Cel projektu: Opracowanie aplikacji kontrolującej elektronicznego barmana dla mobilnych systemów operacyjnych IOS oraz Android.

Założenia: Aplikacja ma wysyłać komendy do mikrokontrolera kontrolującego pracę elektronicznego barmana poprzez połączenie Bluetooth.

Mikrokontroler : DFBLUNO lub DFROMEO

**Android**

Aplikacja ma za zadanie wysyłać stringi (0x01,0x02...itp.) po naciśnięciu przycisku z danym drinkiem do mikrokontrolera Bluno poprzez bluetooth. W tym celu skorzystałem z biblioteki BluetoothLeService w celu zainicjalizowania połączenia.

Klasa DrinksMenu jest klasą która dziedziczy po klasie Fragment. Ma to na celu wprowadzenie obsługi fragmentów programu przez Activity Menu.

**1. public void** serialSend(String theString)

Ma za zadanie obsłużyć wysyłanie stringów do charakterystyk.

*mSCharacteristic*.setValue(theString);

**mBluetoothLeService**.writeCharacteristic(*mSCharacteristic*)

**2. public void** onResumeProcess()

Kiedy bluetooth jest wyłączony, okienko dialogowe zapyta użytkownika czy wyraża zgodę na włączenie go.

**3. public void** onPauseProcess()

Przechodzi w stan skanowania.

Czyści listę urządzeń.

**4. public void** onStopProcess()

Gdy przekroczy czas na połączenie, wyłącza usługę szukania.

**5. public void** onActivityResultProcess

Kiedy użytkownik nie włączy bluetooth przechodzi do wyłączenia aplikacji.

**6. boolean** initiate()

Tworzy bluetooth managera.

**7. void** buttonScanOnClickProcess()

Na case-ach określa stany połączenia.

**8. void** scanLeDevice(**final boolean** enable)

Uruchamia skanowania urządzeń.

**9. public** View getView

Odpowiada za widok listy urządzeń.

**10. public** View onCreateView

Odpowiada za obsługę fragmentu oraz całej jej zawartości (ImageButton itp.)

11422933_1007162522628442_555543642_o.jpg







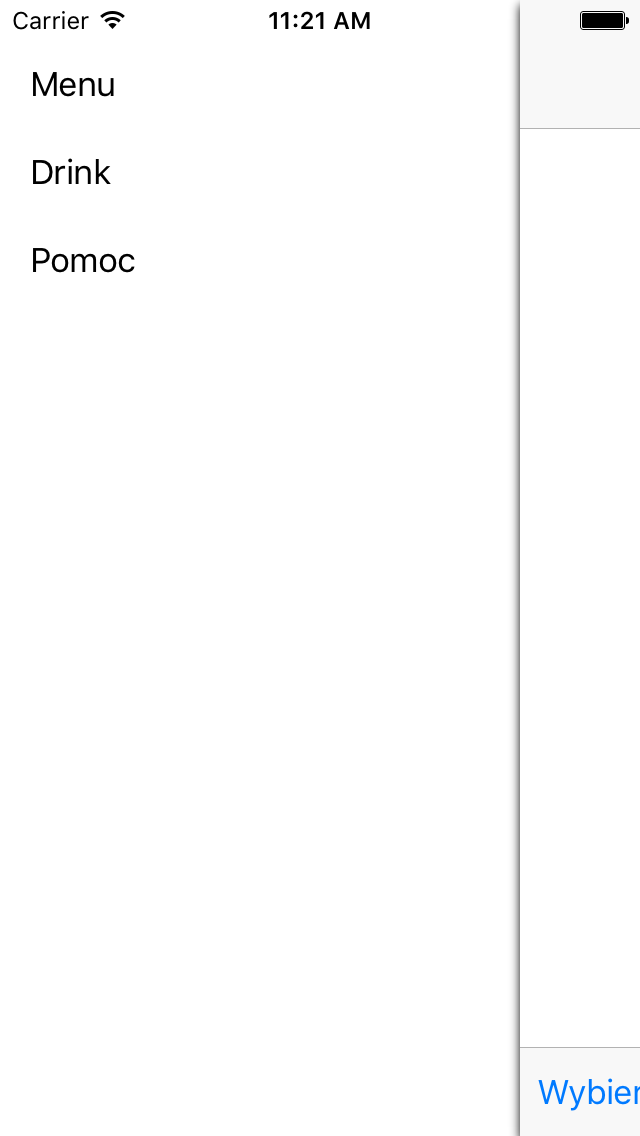


IOS

Aplikacja napisana w języku Swift 1.2. Ma za zadanie wysyłanie komend do mikrokontrolera poprzez Bluetooth.

Interfejs aplikacji zaprojektowany z użyciem Autolayout dzięki czemu dopasowuje się do każdego urządzenia z systemem IOS. Wysuwane menu zaimplementowane dzięki użyciu klasy SWRevealViewController napisanej w Objective-C.





Obsługa Bluetooth zaimplementowana poprzez systemową bibliotekę CoreBluetooth, z użyciem klas:

* CBCentralManagerDelegate,
* CBPeripheralDelegate.

oraz funkcji:

- func centralManager(central: CBCentralManager, didDiscoverPeripheral peripheral: CBPeripheral, advertisementData: [String : AnyObject], RSSI: NSNumber)

Opowiada za akcję po wykryciu urządzenia oraz połączenie się z nim.

- func centralManager(central: CBCentralManager, didConnectPeripheral peripheral: CBPeripheral)

Odpowiada za akcję po połączeniu z urządzeniem oraz wykrycie jego serwisów.

- func peripheral(peripheral: CBPeripheral, didDiscoverServices error: NSError?)

Odpowiada za akcję po wykryciu serwisów oraz wykrycie charakterystyk.

- func peripheral(peripheral: CBPeripheral, didDiscoverCharacteristicsForService service: CBService, error: NSError?)

Odpowiada za akcję po wykryciu charakterystyk określonego serwisu, uruchomienie notyfikacji oraz odczytanie charakterystyki.

- func sendText(data : String)

Odpowiada za odpowiednie formatowanie oraz wysyłanie danych do urządzenia.

- func centralManagerDidUpdateState(central: CBCentralManager)

Odpowiada za obserwowanie stanu Bluetooth i informowanie użytkowanika o zmianach.  
W przypadku aktywnego Bluetooth rozpoczyna skanowanie.