Studia i Materiały Informatyki Stosowanej, Tom 11, Nr 2, 2019
str. 5-10

PROGRAMOWALNA PLATFORMA MOBILNA DO NAUKI PROGRAMOWANIA

Daniel Chudziński*[1], Piotr Kotlarz[2], Marcin Kempiński[2]

[1] (Student)Uniwersytet Kazimierza Wielkiego, Instytut Mechaniki i Informatyki Stosowanej, Chodkiewicza 30, 85-064 Bydgoszcz
[2] Uniwersytet Kazimierza Wielkiego, Instytut Mechaniki i Informatyki Stosowanej, Chodkiewicza 30, 85-064 Bydgoszcz
e-mail: piotrk@ukw.edu.pl

Streszczenie: *W artykule przedstawiono możliwości wykorzystania niskobudżetowych technologii do budowy programowalnych kołowych platform mobilnych, między innymi do zastosowań edukacyjnych. Prototyp opracowanej platformy porusza się z wykorzystaniem dwóch silników DC, komputer pokładowy to moduł Arduino Uno.*

Słowa kluczowe: *Arduino, programowanie, uczenie się*

Programmable mobile platform for learning programming

Abstract: *Athe article shows possibilities of appication of low-cost technologies to construction programmable wheeled mobile platforms for multipurpose use, including learning. The prototype moves thanks to use of two DC motors. Onboard copmputer is based on Arduino Uno.*

Key words: Arduino, programming, learning

1. Wprowadzenie

W ramach tej pracy przeprowadzamy analizę możliwości zastosowania niskobudżetowych układu elektroniki do budowy autonomicznych, programowalnych platform mobilnych. Opracowany został projekt oraz prototyp platformy mobilnej, która jest kontrolowana poprzez autorskie oprogramowanie dla systemu Android. Opracowano implementację protokołów wymiany danych w zakresie sterowania platformą w czasie rzeczywistym, oraz dla potrzeb akwizycji danych. W zakresie rozwiązań sprzętowych opracowano układ sterowania, zasilania oraz zestaw czujników dostosowanych do programowania ruchu platformy oraz do jej autonomicznego poruszania się. Założeniem było opracowanie prototypu platformy zarówno w wymiarze sprzętowym i programistycznym możliwej do zastosowania w edukacji na poziomie szkoły średniej oraz studiów kierunków inżynierskich. Tematyka zastosowań programowalnych platform mobilnych jest obecnie bardzo popularna w światowych pracach B+R i badaniach naukowych [1][2].

2. Zastosowane technologie i sytuacja licencyjna

W zakresie modułu sprzętowego wybrano Platformę Arduino [3] ze względu na dużą dostępność na rynku zarówno mikrokontrolerów jak i akcesoriów. Do programowania urządzenia użyto programu dedykowanego przez Arduino i dostępnego na stronie *www.arduino.cc*. Zarówno platforma Arduino [4] jak i kompilator to popularne rozwiązania, które pozwalają zautomatyzować pewne czynności z zakresu prostej robotyki i automatyki bez znajomości zagadnień sterowania i mechaniki, oraz przy minimalnej wiedzy z zakresu elektroniki.

Aplikacja do kontroli platformy mobilnej została napisana dla systemu Android, który cechuje się dużą popularnością w rozwiązaniach mobilnych. Do napisania aplikacji użyto programu AndroidStudio [5], który jest dedykowanym środowiskiem deweloperskim dostarczanym przez firmę Google.

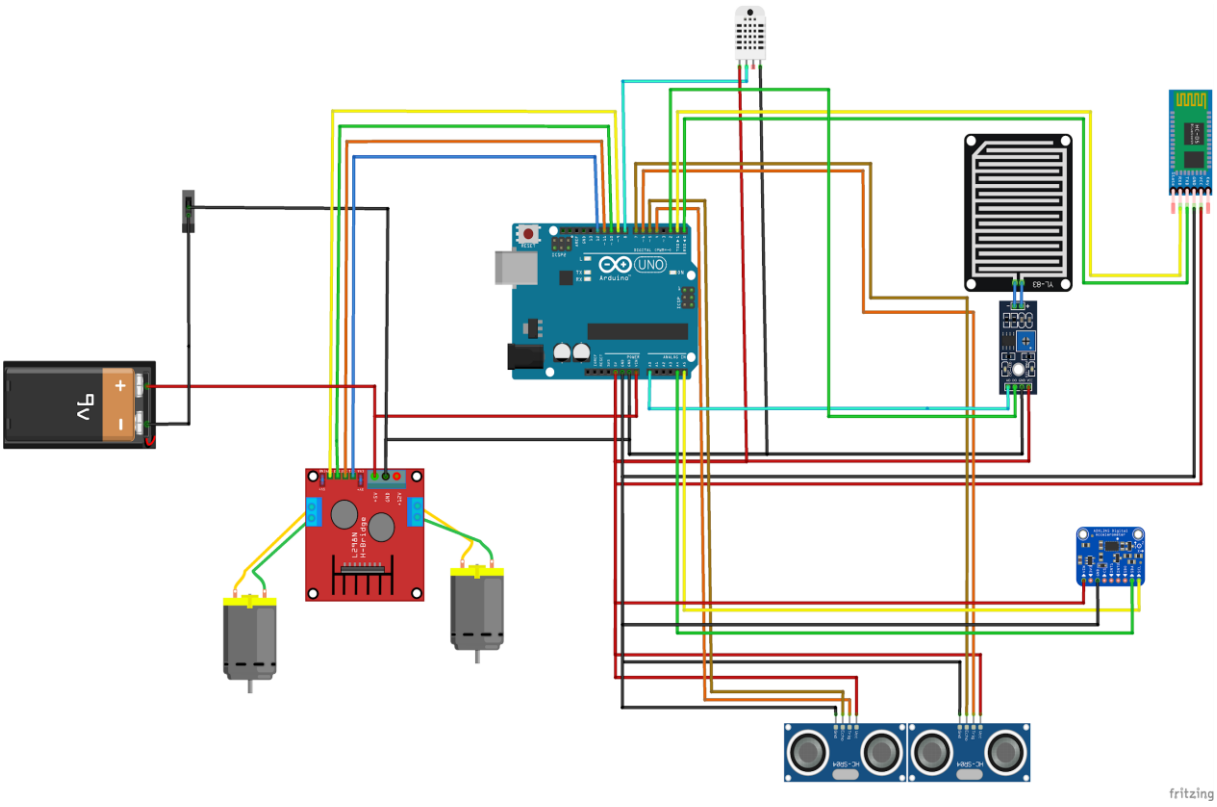
Jako silnik bazy danych zastosowano SQLite jest to system zarządzania bazą danych, który obsługuje język SQL dostępny w ramach licencji public domain. SQLite posiada API między innymi dla języka programowania Java, który jest wykorzystywany w tym projekcie. Baza znajduje się w pliku lokalnie zapisanym na urządzeniu.

Kwestie licencyjne, które występują w projekcie przedstawiają się następująco: SQLite działa na podstawie licencji public domain, natomiast Android Studio działa na licencji Freeware. Jeżeli chodzi o platformę Arduino to działa ono na licencji GNU Lesser General Public License (LGPL) tak samo jak sam język programowania Java. W programie Arduino

wykorzystywane są biblioteki ADXL345.h (odpowiada za obsługę akcelerometru) oraz DHT.h (odpowiada za odczyt temperatury i wilgotności z czujnika DHT22), które są udostępniane użytkownikom na podstawie licencji MIT. Natomiast w aplikacji Android zastosowana została biblioteka Gauge, która odpowiada za wizualizację pomiarów i jest dostępna w licencji Apache.

3. Moduł sprzętowy projektu

Sercem moduły sprzętowego jest układ komputera jednopłytkowego Arduino Uno, poniżej przedstawiono i opisano kluczowe kwestie realizacji całego moduły sprzętowego.

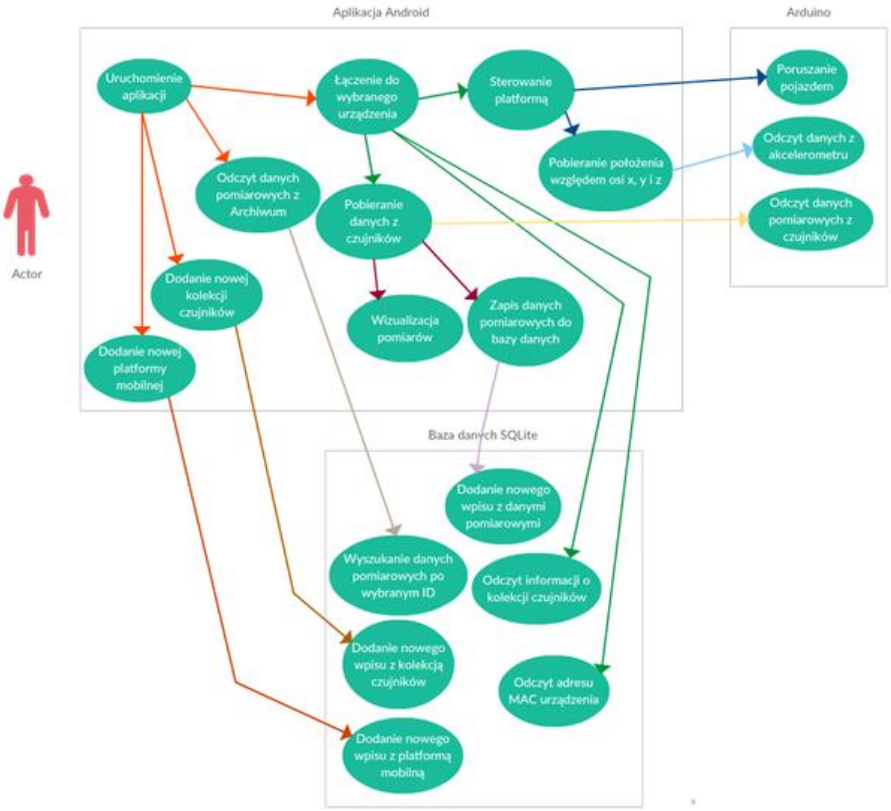


Rysunek 1. Schemat połączeniowy układu sterowania i akwizycji danych

Sterownik silników L298N jest podłączony bezpośrednio do układu zasilania natomiast piny wejściowe sterownika podłączone są do Arduino Uno, a dokładniej do pinów cyfrowych o numerach 9, 10, 11 oraz 12. Do sterownika są podłączone dwa silniki DC, które będą poruszać się do przodu lub do tyłu w zależności od tego, na który port wejściowy sterownika ustawimy stan wysoki. Samo Arduino jest także zasilane zestawem baterii 4xAAA i regulowane włącznikiem. Czujnik temperatury i wilgotności DHT22 jest podłączony do portu cyfrowego o numerze 8 znajdującego się w mikrokontrolerze, a zasilany jest napięciem 5V. Czujnik deszczu, śniegu i wody FC-37 podłączony jest zarówno do portu analogowego A0 oraz do portu cyfrowego o numerze 2, a zasilany jest także napięciem 5V udostępnianego z Arduino. Pierwszy czujnik odległości HC-SR04 jest podłączony do portów cyfrowych o numerach 4 (Trig) oraz 5 (Echo). Drugi czujnik jest podłączony pod portów o numerach 6 (Echo) oraz 7 (Trig). Oba są zasilane takim samym napięciem 5V. Moduł bluetooth jest podłączony portem TX do portu cyfrowego o numerze 0 (RX), natomiast portem RX do portu o numerze 1 (TX), a zasilany jest napięciem 5V. Akcelerometr GY-291 jest podłączony do zasilania 5V. Pin SCL jest podłączony do pinu analogowego o numerze 5, natomiast pin SDA jest podłączony do pinu analogowego o numerze 4.

4. Moduł programistyczny

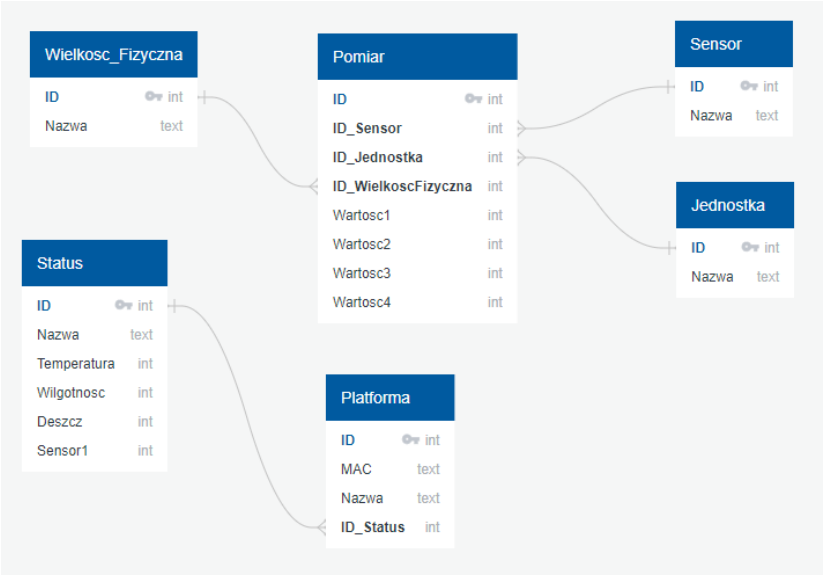
Poniżej przedstawiono diagram przypadków użycia (DPU), który prezentuje główne funkcjonalności opracowanego rozwiązania.



Rysunek 2. Diagram przypadków użycia

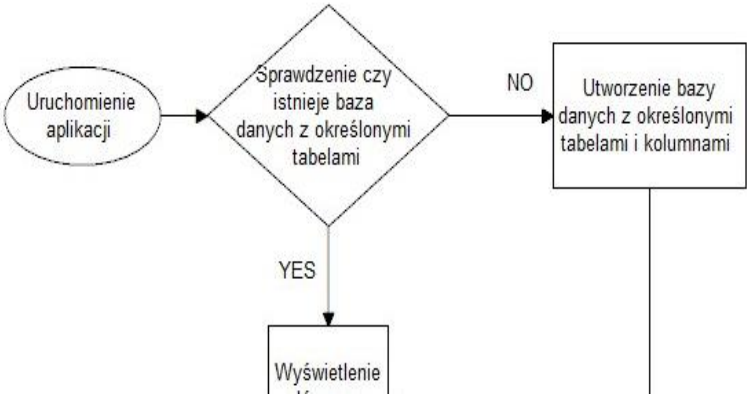
Powyższy diagram przedstawia przypadki użycia aplikacja. Składa się ona z trzech modułów: Aplikacja Android, Arduino oraz Baza danych SQLite. Każdy z tych systemów posiada funkcjonalności przedstawione w zielonych elipsach. Strzałki wskazują powiązania jakie istnieją między tymi funkcjami. Dla przykładu po uruchomieniu programu przez użytkownika oraz dodaniu nowej kolekcji czujników, funkcja ta odwołuje się do funkcji Dodawanie nowego wpisu z kolekcją czujników w systemie Baza danych SQLite.

Na potrzeby praktycznej realizacji projektu oraz przeprowadzenia testów rozwiązania, zaprojektowano i implementowano bazę danych, której model przestawiony jest poniżej.

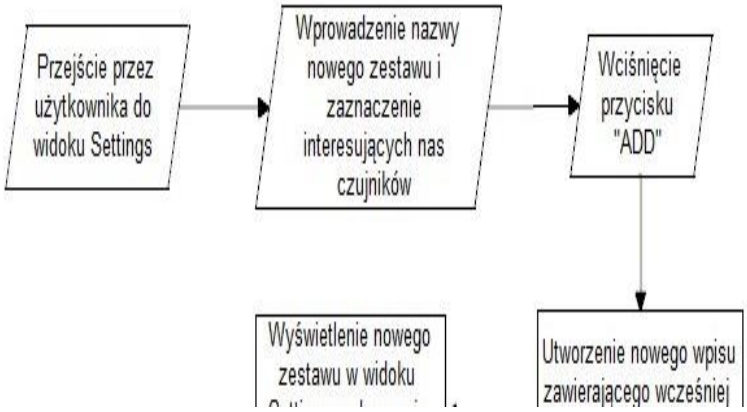


Rysunek 3. Model relacyjny bazy danych.

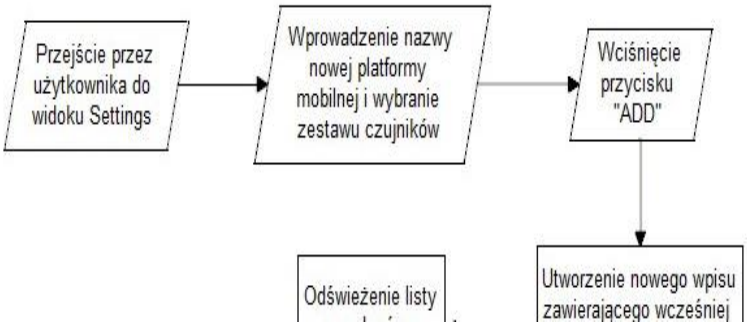
Baza danych programu składa się z sześciu tabel. Tabele Wielkosc_Fizyczna, Sensor oraz Jednostka składają się z dwóch kolumn ID oraz Nazwa, gdzie pierwsza z nich jest kluczem głównym i są w relacjach jeden do wielu z tabelą Pomiar. Tabela Pomiar zawiera klucz główny ID, klucze obce ID_Sensor, ID_Jednostka, ID_WielkoscFizyczna oraz kolumny Wartosc1, Wartosc2, Wartosc3 i Wartosc4 przechowujące wartości odczytów z czujnika. Dwie ostatnie tabele to Status, która zawiera informacje o kolekcjach czujników oraz Platforma, która zawiera informacje o urządzeniach. Kolumna ID w tabeli Status jest kluczem głównym połączonym z tabelą Platforma w relacji jeden do wielu z kluczem obcym ID_Status.
Na poniższych schematach przedstawiono opracowano algorytmy, które służą do obsługi bazy danych oraz związanych z komunikacją z bazą funkcjonalności.



Rysunek 4. Schemat blokowych tworzenia baz danych



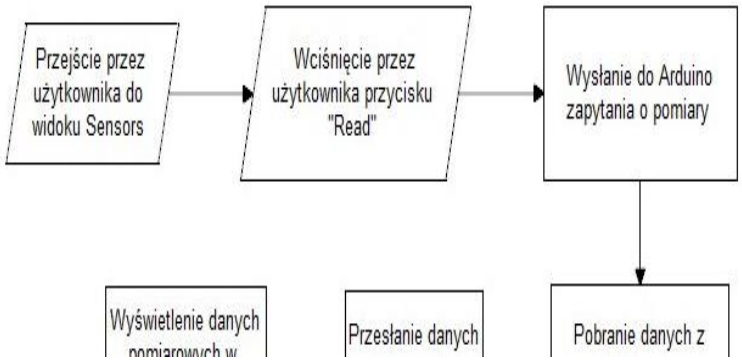
Rysunek 5. Schemat blokowy dodawania kolekcji



Rysunek 6. Schemat blokowy dodawania nowej platformy

Oprogramowanie platformy wyposażone zostało również niezbędna funkcjonalności związane z akwizycją danych pomiarowych oraz wizualizacją tych danych. Z akwizycją danych wiąże się również problem komunikacji, które sposób rozwiązanie przedstawia poniższy kod.

```
blSocket.getOutputStream().write("a".getBytes());
try {
    Thread.sleep( millis: 1000);
}
catch (InterruptedException e) { }
inputStream=blSocket.getInputStream();
int byteCount = inputStream.available();
if(byteCount > 0) {
    byte[] rawBytes = new byte[byteCount];
    inputStream.read(rawBytes);
    final String string = new String(rawBytes,  charsetName: "UTF-8");
    temperature = string;
    tbTemperature.setText(temperature);
```



Rysunek 7. Schemat blokowy odczytu pomiarów

Kod ten przedstawia jakie działania są podejmowane przez aplikację po wciśnięciu przycisku “Read”, który ma za zadanie pobranie odczytów z urządzenia Arduino. Wysyłane są poprzez Bluetooth wartości “a”, “b”, “c” oraz “d”, które odpowiadają za wysłanie przez Arduino do aplikacji odpowiedniego odczytu, to znaczy dla wartości “a” jest to odczyt z czujnika temperatury, dla “b” jest to odczyt z czujnika wilgotności, dla “c” jest to czujnik deszczu, natomiast dla “d” jest to sensor1, który jest miejscem na dodanie przez użytkownika dodatkowego wybranego czujnika. Dane są wysyłane w formacie “byte” do Arduino, a urządzenie przesyła dane do aplikacji także w formacie “byte”, co jest następnie konwertowane do formatu string z kodowaniem “UTF-8”. Na samym końcu otrzymana wartość jest przypisywana do textboxa odpowiadającego danemu czujnikowi.w ramach zestawu.

Literatura

1. Jaskot A., Posiadała B. Analysis of motion of the three wheeled mobile platform. MATEC Web of Conferences 157, 01008, 2018.
2. Tao Song, Bang-Guo Wei A calibration method of dual two-dimensional laser range finders for mobile manipulator., Intelligent Manufacturing and Robotics – Research Article 2019.
3. Rodriguez K., Crespo J, Barber R. An Android Interface for an Arduino Based Robot for Teaching in Robotics, The 6th International Conference of Education, Research and Innovation, Sevilla 2013
4. Simon Monk, Arduino : 36 projektów dla pasjonatów elektroniki, Wydawnictwo Helion, 2015.
5. Android Studio: tworzenie aplikacji mobilnych, Marcin Płonkowski, Helion, 2018.