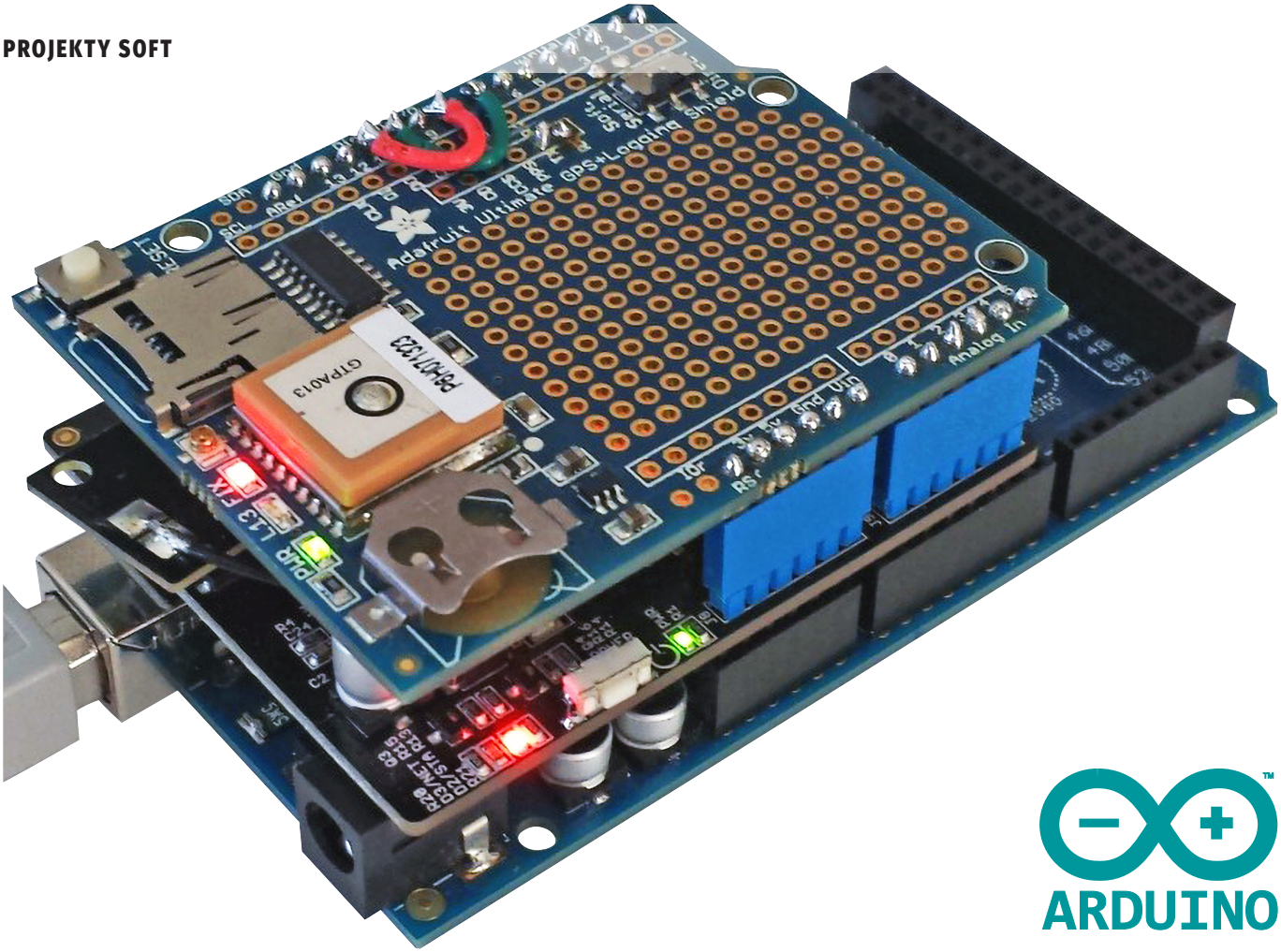


Lojack – tracker GPS do samochodu na bazie Arduino

Kradzieże samochodów to prawdziwy problem! Zwiększ swoje szanse na odzyskanie samochodu, instalując w nim prosty tracker oparty na Arduino. Urządzenie pozwala śledzić w czasie rzeczywistym położenie samochodu.

Prezentowany projekt jest ciekawym i dosyć prostym urządzeniem, które idealnie nadaje się do budowy nawet przez początkującego programistę Arduino. Jednocześnie jest to realnie przydatne urządzenie, które może być zamontowane w każdym samochodzie, szczególnie jeśli często parkujemy na niestrzeżonych parkingach i obawiamy się o ochronę naszego mienia.

Przed rozpoczęciem realizacji tego projektu należy opanować kilka podstawowych umiejętności, które będą potrzebne nam podczas realizacji projektu:

- Powinieneś być zaznajomiony z obsługą środowiska programistycznego Arduino IDE,
- Przydatna będzie wiedza na temat elektryki samochodowej 12 V,
- Wymagana będzie elementarna wiedza na temat wykorzystania PHP i MySQL,
- Wskazany jest także dostęp do serwera webowego, gdzie uruchomić można prostą aplikację w PHP.

Projekt składa się z dwóch głównych elementów:

- Lokalizatora, który jest ukryty w samochodzie i przesyła informacje GPS za pośrednictwem sieci GPRS (komórkowej) do Internetu,
- Aplikacji internetowej, która odbiera informacje GPS z urządzenia śledzącego i umożliwia ich prezentację, aby zlokalizować położenie pojazdu.

Tracker jest zbudowany na bazie modułu Arduino Mega z dołączonym shieldem zapewniającym możliwość odbierania i dekodowania sygnałów systemu nawigacji satelitarnej (GPS, GLONASS, Galileo i innych – w zależności od lokalizacji i możliwości wybranego przez nas modułu) oraz shielda umożliwiającego przesyłanie danych poprzez sieć komórkową (GPRS).

Aplikacja internetowa została napisana w języku PHP. Korzysta z bazy danych MySQL jako magazynu danych dotyczących położenia monitorowanego pojazdu. Zaprezentowana w artykule implementacja to jedynie minimalistyczny program składający się z REST-owego API, które odbiera informacje z trackera i wyświetla lokalizację na mapie. Istnieje możliwość rozbudowania aplikacji o dodatkowe funkcje, jeśli takowe są potrzebne.

Potrzebne elementy

Do budowy urządzenia śledzącego potrzebne będą nam:

- moduł Arduino Mega,
- moduł transceivera GPRS, najlepiej w postaci shielda kompatybilnego ze standardem Arduino,
- moduł odbiornika GPS, również w postaci standardowego shielda.

Arduino Mega jest mózgiem całego urządzenia. Znajdujący się w tym module mikrokontroler (ATmega2560) zajmuje się odczytywaniem komunikatów zbieranych przez odbiornik GPS, formatowaniem ich i wysyłaniem przez sieć bezprzewodową do aplikacji internetowej. Konieczne jest załadowanie odpowiedniego oprogramowania układowego (firmware) do mikrokontrolera, gdzie wpisany będzie szereg informacji dla konkretnej aplikacji, takich jak adres serwera WWW, dokąd wysyłane są dane GPS.

Autor zaznacza, że początkowo próbował zastosować zamiast tego modułu zwykły, podstawowy moduł Arduino; okazało się jednak, że ograniczenia m.in. w ilości pamięci tego układu uniemożliwiły uruchomienie tej konfiguracji. Warto zainwestować w droższy moduł o wyższych parametrach, aby uprościć tworzenie i implementację aplikacji, a także zostawić możliwość rozbudowy aplikacji czy systemu w przyszłości.

Drugi z potrzebnych modułów – shield GPRS – potrzebny jest do dwukierunkowej transmisji danych pomiędzy trackerem a aplikacją internetową. Autor konstrukcji wybrał shield GPRS firmy Seeed Studio z uwagi na proste, standardowe podłączenie do modułu Arduino i sprawdzone działanie w ekosystemie Arduino. Shield ten nie wymaga żadnych modyfikacji przed zastosowaniem w naszym urządzeniu.

Oprócz samego modułu GPRS musimy zaopatrzyć się także w kartę SIM. Należy ją zainstalować w module GPRS, aby mógł korzystać z sieci komórkowej. W module najlepiej sprawdzi się karta SIM wyłącznie do transferu danych. Tego rodzaju karty oferuje większość dostawców obecnych na naszym rynku – zazwyczaj są one stosowane w systemach zdalnych sensorów itp.

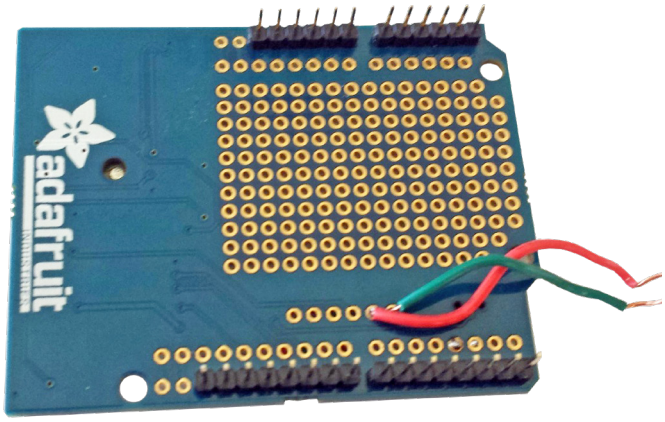
Ostatnim modułem, jaki jest potrzebny w tym systemie, jest oczywiście odbiornik GPS. Autor wybrał moduł GPS firmy Adafruit, który również jest w pełni kompatybilny ze standardem Arduino, aczkolwiek w tej aplikacji wymaga kilka drobnych zmian. Moduł GPS montowany jest bezpośrednio na module GPRS, co jest istotne, gdyż moduł GPS także wykorzystuje łączność komórkową, do uzyskiwania codziennych poprawek itp., które mają na celu zwiększenie precyzji pomiaru położenia poprzez GPS.

Dodatkowo konieczne jest zasilenie urządzenia. W samochodzie można to zrobić na dwa sposoby:

- Używając kabla USB A-B i adaptera wtyczki zapalniczki samochodowej do USB (autor zastosował ładowarkę USB z prądem do 2,1 A, żeby zapewnić wystarczającą moc do zasilania całego systemu). Jest to najprostsze rozwiązanie, jednak wymaga dostępu do gniazda zapalniczki i instalacji układu gdzieś w jego pobliżu.
- Używając stabilizatora 5 V, zasilanego bezpośrednio z instalacji 12 V samochodu. Nie zapomnijmy dołączyć odpowiednich elementów do stabilizatora (kondensatorów na wejściu i wyjściu układu oraz zabezpieczenia przeciwprzepięciowego na wejściu zasilacza). Takie rozwiązanie jest o wiele bardziej elastyczne – pozwala na instalację systemu niemalże w dowolnym miejscu samochodu.

Oprócz zwykłego zasilania warto rozważyć jeszcze dodanie awaryjnej baterii lub akumulatorka, które pozwoliłyby układowi na działanie przy braku zewnętrznego zasilania – na przykład w momencie, gdy złodziej odłączył akumulator od samochodu, by rozbroić konwencjonalny alarm czy inne zabezpieczenia.

Finalnym, istotnym elementem, jest obudowa układu. Powinna ona zabezpieczać go przed dostępem kurzu i wilgoci, a jednocześnie pozwolić na wyprowadzenie wszystkich potrzebnych kabli – do anten i zasilania. Autor konstrukcji obudowę dla swojego układu wykonał… z klocków Lego, co jest zadziwiająco dobrym rozwiązaniem dla amatorskiej konstrukcji.



Fotografia 1. Krótki kabelek przylutowany do pól TX i RX modułu GPS

Modyfikacje modułu GPS

W module GPS wprowadzić trzeba kilka drobnych zmian, zanim będzie można używać go wraz z Arduino w tym systemie. Niestety oba moduły – GSM i GPRS – używają do komunikacji z mikrokontrolerem pinów 7 i 8, więc konflikt ten trzeba rozwiązać sprzętowo. W tym celu wprowadzamy następujące drobne zmiany w module odbiornika systemu nawigacji satelitarnej.

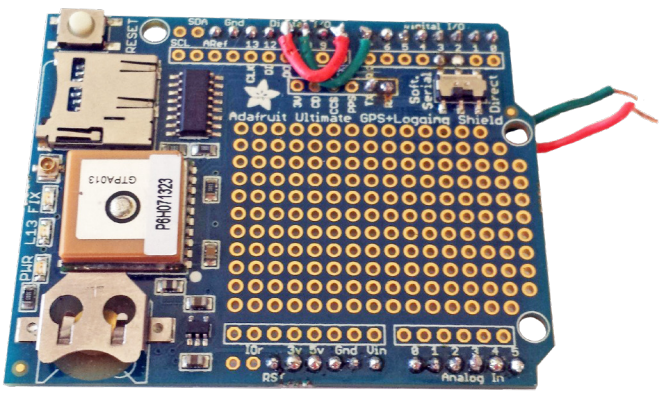
1. Przetnij ścieżkę pomiędzy pinami TX i RX modułu GPS a pinami 7 i 8 wyprowadzenia z płytki. Można to zrobić za pomocą skalpela bądź ostrego noża.
2. Na spodzie płytki przylutuj parę krótkich (około 7 cm) kabli do pinów TX i RX odbiornika GPS (**fotografia 1**).
3. Przeniesione zostaną też linie portu szeregowego modułu GPRS, który znajduje się poniżej modułu GPS. Aby przenieść te sygnały pomiędzy pinami 7 i 10 oraz 8 i 11, lutujemy zworki (**fotografia 2**).
4. Na koniec ustaw przełącznik w module na Soft Serial (port szeregowy realizowany programowo).

Po zmontowaniu wszystkich modułów ze sobą, tj. po umieszczeniu shieldu GPS na GPRS, należy przylutować przewód od pinu TX modułu GPS do pinu RX na płytce z mikrokontrolerem, a przewód z pinu RX do pinu TX. W ten sposób sprzęt jest gotowy do działania. Po podłączeniu Arduino kablem USB do komputera, na modułach powinny zaświecić się kontrolki, jak na fotografii tytułowej. Urządzenie jest gotowe na załadowanie oprogramowania układowego.

Firmware Arduino

Całość oprogramowania projektu, w najaktualniejszej wersji, znaleźć można na repozytorium autora na portalu GitHub (link: ***https://bit.ly/2K21APd***).

Oprogramowanie dla modułu Arduino znajduje się w folderze Arduino i składa się z pojedynczego szkicu *lojack.ino*, którego fragment pokazany jest na **listingu 1**. W kodzie tego programu należy wprowadzić pewne modyfikacje, ustawiając wartość zmiennej *remote_server* na adres serwera,



Fotografia 2. Zworka przenosząca lokalizację sygnałów TX i RX modułu GPRS

który będzie hostował aplikację webową, zbierającą dane z trackera. Jeśli chcesz, aby system był w stanie śledzić wiele urządzeń, upewnij się, że zmienna *deviceId* jest unikalna dla każdego urządzenia.

Po uzupełnieniu wszystkich potrzebnych danych w szkicu Arduino można program skompilować i wgrać do mikrokontrolera. Przed kompilacją należy upewnić się, że w IDE ustawiona jest właśnie ta płytka jako docelowa. Po wgraniu odpowiednio skompilowanego szkicu do modułu wydarzyć powinny się dwie rzeczy:

1. Lampka FIX na module GPS powinna, po chwili (od kilku sekund do kilku minut, zależnie od lokalizacji), przestać migać i zapalić się na stałe. Wskazuje to, że odbiornik GPS połączył się poprawnie z satelitami. Oczywiście, najlepiej testować to na zewnątrz lub w pobliżu okna.
2. Czerwona dioda LED na shieldzie GPRS powinna zaświecić się, a dioda zielona migać z częstotliwością około raz na sekundę. Po połączeniu modułu GPRS z siecią komórkową zielona dioda LED będzie migać rzadziej, co około trzy sekundy.

W tym momencie moduł śledzenia jest już uruchomiony i określa swoją lokalizację, a następnie próbuje przesłać ją na zdalny serwer. Na razie, żaden serwer jeszcze nie istnieje, dlatego też konieczne jest jego stworzenie, co zostało opisane poniżej.

Aplikacja webowa

Aby ukończyć projekt, potrzebny jest dostęp do serwera połączonego z Internetem, na który będzie można wgrać skrypt napisany w PHP i uruchomić bazę MySQL. Istnieje wiele różnych sposobów rozwiązania tego problemu. Autor skorzystał z maszyny wirtualnej, dostępnej u jednego z usługodawców w chmurze. Do uruchomienia tej aplikacji potrzebna jest zazwyczaj najsłabsza z dostępnych konfiguracji. Na maszynie wirtualnej trzeba zainstalować Apache (lub inny wybrany serwer www), PHP i MySQL (lub inną bazę danych). Skrypt autora zakłada użycie Apache i MySQL jeśli chcemy korzystać z innych narzędzi, konieczna może być modyfikacja w celu dopasowania skryptu.

Oprogramowanie na serwerze składa się z dwóch skryptów – *reportLocation.php* oraz *where.php*. Pierwszy z nich odbiera dane lokalizacyjne z trackera w samochodzie i zapisuje dane w bazie. Drugi skrypt pobiera dane z bazy i prezentuje pozycję auta na mapie.

Przed uruchomieniem któregokolwiek ze skryptów należy utworzyć na serwerze bazę danych o nazwie „lojack” z użytkownikiem z dostępem do odczytu i zapisu. Utwórz w bazie tabelę o nazwie *gpsUpdates* o następującej strukturze:

- id : int(11) : not null : primary key: default null : auto_increment
- latitude: (decimal(6,4): nullable: default null
- longitude: (decimal(7,4): nullable: default null
- deviceId: int(11) : nullable : default null
- timestamp: timestamp: not nullable: default: current_timestamp:on update CURRENT_TIMESTAMP

Aby utworzyć taką bazę, wykorzystaj następujące polecenia SQL:

```
CREATE DATABASE lojack;
USE lojack;
CREATE TABLE lojack(
id INT( 11 ) NOT NULL AUTO_INCREMENT,
latitude decimal( 6,4 ) NULL DEFAULT NULL,
longitude decimal( 7,4 ) NULL DEFAULT NULL,
deviceId INT ( 11 ) NULL DEFAULT NULL,
timestamp TIMESTAMP NOT NULL DEFAULT CURRENT_
TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY ( id ) , UNIQUE (id)
);
```

W głównym katalogu serwera umieszczamy skrypty. Pierwszy z nich, pobierający dane z trackera, pokazano na **listingu 2**. Skrypt otrzymuje od urządzenia trzy parametry: *latitude*, *longitude* oraz *deviceId*. Wartości te zapisywane są następnie do bazy danych lojack. W skrypcie, przed jego pierwszym uruchomieniem, zmienić trzeba nazwę użytkownika i hasło do bazy danych. W momencie, gdy i tracker i skrypt ładujący dane do bazy działają, powinno być możliwe uruchomienie skryptu do prezentacji danych. Zawartość skryptu *where.php* zaprezentowana została na **listingu 3**. Po jego uruchomieniu

Listing 1. Fragmenty kodu programu uruchamianego na module Arduino w trackerze

```
SIGNAL(TIMER0_COMPA_vect) {char c = GPS.read();}

void setup(){
    Serial.begin(115200);
    setupGPS();
    setupGprs();
}

void setupGPS(){
    GPS.begin(9600);
    GPS.sendCommand(PMTK_SET_NMEA_OUTPUT_RMCGGA);
    GPS.sendCommand(PMTK_SET_NMEA_UPDATE_1HZ);
    GPS.sendCommand(PGCMD_ANTENNA);
    setupInterrupt();
    delay(1000);
    gpsSerial.println(PMTK_Q_RELEASE);
}

void setupInterrupt(){
    OCR0A = 0xAF;
    TIMSK0 |= _BV(OCIE0A);
}

void setupGprs(){
    pinMode(9, OUTPUT);
    digitalWrite(9,LOW);
    delay(100);
    digitalWrite(9,HIGH);
    delay(500);
    digitalWrite(9,LOW);
    delay(100);
    gprs.init();
}

void joinNetwork(){
    while(false == gprs.join()) {
        Serial.println(„gprs join network error”);
        setupGprs();
        delay(2000);
    }
    Serial.println(„joined network!”);

    // Poprawne uruchomienie klienta DHCP
    Serial.print(„IP Address is „);
    Serial.println(gprs.getIPAddress());

    if(false == gprs.connect(TCP,remote_server, 80)) {
        Serial.println(„connect error”);
    }else{
        Serial.println(„connect success”);
    }
}

void loop(){
    if (GPS.newNMEAreceived()) {
    if (!GPS.parse(GPS.lastNMEA()))
        return;
    }
    if (timer > millis())    timer = millis();
    if (millis() - timer > 2000) {
        timer = millis();
        sendLocation();
        Serial.print(GPS.latitudeDegrees, 4);Serial.print(„, „);
        Serial.println(GPS.longitudeDegrees, 4);
    }
}

void updateRequestString(){
    char charLatitude[10];
    char charLongitude[10];
    dtostrf(GPS.latitudeDegrees, 7, 4, charLatitude);
    dtostrf(GPS.longitudeDegrees, 9, 4, charLongitude);

    for (int i=0;i<7;i++) {http_cmd[33+i] = charLatitude[i];}
    for (int i=0;i<9;i++) {http_cmd[51+i] = charLongitude[i];}
}

void sendLocation(){
    joinNetwork();
    updateRequestString();

    Serial.print(„Sending: „);
    Serial.println(http_cmd);
    char buffer[512];
    gprs.send(http_cmd, sizeof(http_cmd)-1);
    while (true) {
        int ret = gprs.recv(buffer, sizeof(buffer)-1);
        if (ret <= 0){
            Serial.println(„fetch over...”);
            break;
        }
        buffer[ret] = ‘\0’;
        Serial.print(„Recv: „);
        Serial.print(ret);
        Serial.print(„ bytes: „);
        Serial.println(buffer);
    }
    gprs.close();
    gprs.disconnect();
}
```

Listing 2. Kod aplikacji webowej do zbierania danych o pozycji i ładowania ich do bazy danych

```
<?php
$servername = „localhost”;  // Adres serwera z bazą MySQL
$username = „DBUSERNAME”;   // Użytkownik bazy MySQL
$password = „DBPASSWORD”;   // Hasło do bazy MySQL
$database = „lojack”;       // Nazwa bazy MySQL

$latitude = htmlspecialchars($_GET[„latitude”]);
$longitude = htmlspecialchars($_GET[„longitude”]);
$deviceId = htmlspecialchars($_GET[„deviceId”]);

echo „latitude: $latitude <br>”;
echo „longitude: $longitude <br>”;
echo „deviceId: $deviceId <br>”;

try {
    $dbConnection = new PDO(„mysql:host=$servername;dbname=$database”,          $username, $password);
    // Ustaw tryb błędów PDO jako wyjątki
    $dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $sql = „INSERT INTO gpsUpdates (latitude,longitude,deviceId,timestamp)      values ($latitude,$longitude,$deviceId,now())”;
    $result = $dbConnection->query($sql);

    $dbConnection = null;
}
catch(PDOException $e){
    echo „Connection failed: „ . $e->getMessage();
}
?>
```

(poprzez przeglądarkę internetową) naszym oczom powinna ukazać się mapa, podobna do pokazanej na **rysunku 1** wraz z zaznaczoną lokalizacją pojazdu.

Podsumowanie

Teraz aplikacja jest w stanie prezentować, w czasie rzeczywistym, dane na temat pozycji pojazdu. Możemy z nich skorzystać, np. gdy pojazd zostanie skradziony. Z łatwością zlokalizujemy nasz skradziony samochód, co pomoże nam w jego odzyskaniu.

Listing 3. Kod aplikacji webowej do prezentacji danych o pozycji na mapie

```
<html>
<head>
    <script type=”text/javascript”
    src=”http://maps.google.com/maps/api/js?sensor=false”></script>
    <meta http-equiv=”refresh” content=”30”>
</head>
<body>

<h1>Where’s my car?</h1> <!-- Tytuł na stronie -->
<?php
$servername = „localhost”;  // Adres serwera z bazą MySQL
$username = „DBUSERNAME”;   // Użytkownik bazy MySQL
$password = „DBPASSWORD”;   // Hasło do bazy MySQL
$database = „lojack”;       // Nazwa bazy MySQL

try {
    $dbConnection = new PDO(„mysql:host=$servername;dbname=$database”, $        username, $password);
    // Ustaw tryb błędów PDO jako wyjątki
    $dbConnection->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $sql = „select * from gpsUpdates ORDER BY id DESC LIMIT 1”;
    $result = $dbConnection->query($sql);
    if ($result->rowcount() > 0) {
        $row=$result->fetch();
        echo „latitude: „ . $row[„latitude”]. „ longitude: „ .
        $row[„longitude”]. „<br><p>”;
        }
    $dbConnection = null;
}
catch(PDOException $e){
    echo „Connection failed: „ . $e->getMessage();
}
?>
    <div id=”map” style=”width: 800px; height: 600px”></div>
    <script>
        var myOptions = {
            zoom: 17,
            center: new google.maps.LatLng(<?php echo $row[„latitude”].”,”
            $row[„longitude”];?>),
            mapTypeId: google.maps.MapTypeId.ROADMAP
        };
        var image={
            // ikonka samochodu na mapie
            url:'http://FQDN.example.com/CarIcon.png',
            size: new google.maps.Size(40,40),
            origin: new google.maps.Point(0,0),
            anchor: new google.maps.Point(20,20)
        };
        var map = new google.maps.Map(document.getElementById(„map”),
        myOptions);
        var marker = new google.maps.Marker({
            position: new google.maps.LatLng(<?php echo $row[„latitude”].”,”
            $row[„longitude”];?>),
            map:map,
            icon:image,
            title: ‘My Car’
        });
    </script>
</body>
</html>
```

Uwaga! Nigdy nie próbuj sam odzyskać pojazdu. Zawsze współpracuj z organami ścigania. Zmaganie się z przestępcami nigdy nie jest dobrym pomysłem dla amatorów – pozostaw to profesjonalistom!

Pamiętaj – opisana konstrukcja przeznaczona jest do śledzenia własnego samochodu. Stosowanie jej, np. by kogoś śledzić, jest nie tylko nieetyczne, ale także w dużej mierze nielegalne.

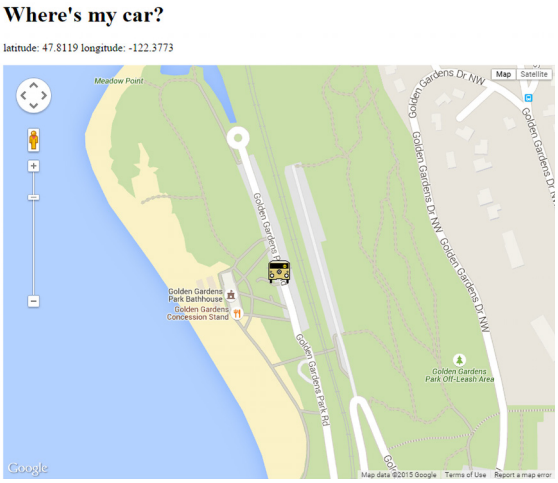
Z uwagi na brak zabezpieczeń interfejsu webowego aplikacji utrzymuj adres URL strony z informacją o położeniu auta w tajemnicy. Autor rozważa dodanie w przyszłości obsługi logowania się i uwierzytelniania na stronie, jednakże obecnie każdy, kto zna adres URL strony, może wiedzieć, gdzie zlokalizowany jest pojazd. Inne plany autora, co do konstrukcji, to między innymi:

- instalacja aparatu cyfrowego, aby system mógł fotografować i przesyłać do Internetu zdjęcia swojego otoczenia lub zdjęcia pasażerów,
- dodanie ręcznego wyłącznika awaryjnego systemu zapłonowego lub odcięcia dopływu paliwa, co pozwoli na zdalne wyłączenie auta,
- opracowanie aplikacji mobilnej,
- miniaturyzacja systemu, by tracker mógł zmieścić się w dowolnym zakamarku w samochodzie.

Nikodem Czechowski, EP

Źródła

1. *https://bit.ly/2RDbk6M*, 2. *https://bit.ly/2K21APd*



Rysunek 1. Widok uruchomionej aplikacji webowej do lokalizacji pojazdu