AUTOMATYKA/AUTOMATICS • 2019 • Vol. 23 • No. 1

https://doi.org/10.7494/automat.2019.23.1.7

Bartosz Kowalik*, Marcin Szpyrka*

Online environment for data acquisition from car sensors

Abstract: Modern cars are equipped with plenty of electronic devices called electronic control units (ECUs). They collect diagnostic data from a car's components to control their work, assess the quality of their work, and detect defects. Regardless of the development of onboard computers, only a small amount of information is passed on to the driver. This paper describes how to build a data-collecting environment for real-data acquisition from a car. The system (whose components include an Android--based smartphone and Torque PRO application) is able to send data via the Internet to a chosen server in real time. The collected data can be further analyzed both online and offline.

Keywords: *car sensors, data-acquisition system, Torque PRO application*

1. Introduction

Nowadays, the car industry equips cars with plenty of electronic control units (ECUs). An ECU is any embedded system that controls one or more of the electrical systems in a vehicle. Developed in 1983, a controller area network (CAN bus) [1] is used for communication between ECUs. On-board diagnostics (OBD [2]) is used in modern cars for diagnostic purposes. This has been a mandatory part of automotive equipment since 1996 in the USA, since 2001 in the EU, and since 2002 in Poland. The OBD device is connected to the CAN bus, enabling us to read a variety of automotive parameters. Modern cars equipped with onboard computers communicate only selected information to the driver – access to most of the available data is limited to specialized service stations.

This paper deals with the problem of data acquisition from a car's ECU and preparing it for further exploration with data-mining techniques. We propose a system that consists of three major components: data acquisition, automated log collection, and persistent storage with presentation tools. This system was built with the intention of keeping limited financial resources in mind; its components include an Android-based smartphone, the Torque PRO application, and the Akka HTTP framework. The presented approach is able to capture a live stream of data from a car and store it without unduly influencing the driver. As a result, it is possible to collect a real data set that can be analyzed with data-mining algorithms [3].

*AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, Computer Science and Biomedical Engineering, Al. Mickiewicza 30, 30-059 Krakow, Poland, e-mail: {bkowalik, mszpyrka}@agh.edu.pl

The considered data-acquisition system was tested using a 2009 Hyundai i30 vehicle with a 1.6-liter turbo diesel engine. Although such a car can produce a high volume of data every second, we were able to capture the data, send it to a server, and store it in a database system with no significant delays.

Preliminary information about the acquisition system was presented in two conference papers [4] and [5]. This article contains a more-detailed description of the data acquisition environment and is an extended version of [5] to some extent.

The paper is organized as follows. Section 2 deals with the architecture of the data--acquisition system. Selected data obtained from the Hyundai vehicle is presented in Section 3. A short summary is provided in the final section.

2. Data-acquisition system architecture

The system for data acquisition must work online while a car is working. Moreover, the system cannot require interaction with the driver. We decided to use the following hardware components:

– Xiaomi Redmi 4X – Android-based smartphone,
– ELM372 OBD2 – Bluetooth OBD2 interface.

The set is extended with an HTTP server that is crucial for data processing. The main architectural parts of the considered system are presented in Figure 1.

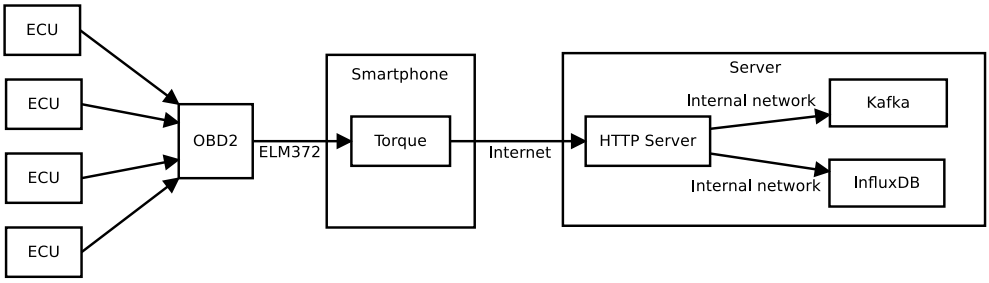

Fig. 1. Data-acquisition system architecture

Torque is an Android application that collects live data from a car – it uses an OBD2 Bluetooth device to which the application is connected. Torque is able to send data over the Internet to an external server. By default, it sends the data to a Torque developers' web server, which has an online lookup panel. In the application options, it is possible to override the target URL with a customized one. No documentation is provided for the communication protocol. In order to implement a custom server, the protocol was reverse engineered. This process required multiple steps. We started with an instance of the server that logged each incoming request to a file and collected sample logs.

Then, each URI (uniform resource identifier) parameter was analyzed to decode its meaning. Each parameter encodes a key-value pair. We used the rather cursory documentation for Torque plug-in developers to guess the meaning of some parameters. OBD2 PIDs are usually presented in the hexadecimal format, and Torque developers follow the same convention. Some parameters mentioned on the website [6] are similar to those that are sent to the server. This allowed us to determine that k means $0x$ in hexadecimal encoding. For instance, kc actually is $0x0c$ – based on the OBD2 PID list [7], this means the engine's revolutions-per-minute descriptor.

Moreover, it is possible to guess the meaning of some key-value pairs:

- eml – email address,
- v – version,
- session – milliseconds (since January 1, 1970, UTC identifies when the measurements are started),
- id – session identifier (after a broad analysis, it turned out to be string representation of a UUID with a stripped "-"),
- time – milliseconds (since January 1, 1970, UTC identifies a single measurement's precise measure time.

For testing purposes, we developed a script that parses log files and makes fake server calls. After the testing procedure, we found out that there are still unknown parameters not mentioned in the Torque documentation nor on the OBD2 PID list. Inspecting the Torque directory on the Android smartphone revealed a text file *torqueConf.dat*. A snippet of the file is shown in Listing 1.

Listing 1: Snippet of *torqueConf.dat* file

```
log_pid19 = 16732675
log_fullName9 = Acceleration Sensor(X axis)
log_pid18 = 16732674
log_fullName8 = Acceleration Sensor(Total)
```

Having analyzed the *torqueConf.dat* file's content, it was clear that very high integer numbers (such as 16732675 or 16732674) are in fact PIDs. This is because Torque stores custom PIDs in decimal form, which is different from what can be found in the official documentation. The next step was to convert all of the missing PIDs from hexadecimal numbers to decimal ones.

Having done this, the *torqueConf.dat* file was used as a lookup. This configuration file has a key-value structure. A decimal PID number is a value that means that part of a key has an internal ID of a measured value. For example, PID kff5203 is 0xff5203 (hexadecimal) and 16732675 (decimal). Value 16732675 is pointed out by key log_pid19. This indicates an internal ID (which is 19). Following the pattern, log_fullName19 indicates the description of a measurement. In this example, it means *liters per 100 kilometers* (long-term average).

Using this technique, the following PIDs were decoded:

- kff126b – remaining fuel calculated,
- kff126a – distance to empty,
- kff1273 – engine kW at wheels,
- kff1272 – average trip speed,
- kff1271 – fuel used on current trip,
- kff125d – fuel flow rate per hour,
- kff129a – android device battery level,
- kff1204 – trip distance,
- kff1267 – GPS bearings,
- kff1266 – trip time since start of journey,
- kff1269 – volumetric efficiency calculated,
- kff1268 – trip time while moving,
- kff5203 – liters per 100 kilometers,
- kff5202 – kilometers per liter,
- kff5201 – miles per gallon.

The presented reverse engineering of Torque was necessary to understand the data collected from OBD2. To cope with the high load generated by the live data uploaded from Torque, it was necessary to include an HTTP server into the system. We decided to use the Akka HTTP framework [8]. According to the documentation, it is advised to use high-throughput middleware.Torque uses a URI to send the recorded parameters. Listing 2 presents a Torque URI example.

Listing 2: Torque URI example

```
?eml=bartekviper@gmail.com&
v=8&
session=1521995741941&
id=80c9065bff55006e9f5f3d4f8d9456ae&
time=1521996644906&
kff1005=19.96196812&
kff1006=50.08527848&
kff1001=27.504&
kff1007=275.4&
kff129a=41.0&
k2d=99.21875&
k33=98.0&
kb=128.0&
k23=36700.0&
kff1267=232.387&
kff1268=577.541&
kff1273=6.591945&
kff1225=17.75426&
kff1238=14.1&
k42=14.349&
k4=0.0&
kc=2257.0&
k21=0.0&
k31=65535.0&
kff126b=40.63311&
kff126a=692.2412&
kff1204=4.1081877&
k10=38.25&
kff1001=27.504&
kd=32.0&
kff1237=1.4079971&
kff123a=14.0&
```

```
kff1266=809.0&
kff1239=6.068&
kff1269=92.0&
kff1005=19.96196812&
kff1006=50.08527848&
kff123b=275.4&
kff1203=2.646776&
kff5202=5.7647285&
kff5203=17.34651&
kff1201=7.4767685&
kff5201=16.284544&
kff1226=8.839944&
k2c=4.7058825&
k46=12.0&
kf=26.0&
k5=84.0&
kff1202=4.3511324&
kff1206=3.6139567&
kff1010=250.0&
kff1271=1.5658529&
kff125d=13.979028&
kff125a=232.9838&
kff1272=21.435392&
kff1208=27.669928
```

Due to some problems with handling the long URIs, we defined a custom Akka dispatcher. The server only accepts GET requests.

Apart from storing new data in the database, the HTTP server it also produces new records on the message queue. In order to achieve a high throughput, the Apache Kafka streaming platform is used [9]. It is composed of three main parts: *producer, consumer*, and *topic*. *Topic* is essentially a message queue. A new message is stored with an associated key. The key can be null, which means that there are no messages. It can be used to arrange messages logically and determine to which partition each message belongs to if a distributed mode is enabled. Each partition has its own message ordering based on keys. *Producer* is an element that publishes new messages to a topic. It sets the message content and associates a key with it. *Consumer* reads messages from the topic. Everyone is a member of the consumer group. A message is consumed once in a single consumer group. Each consumer has an offset that determines which messages were read and which are about to be read.

The HTTP server produces messages to a single topic called *raw_car_data*. Each measurement type (e.g., engine coolant temperature) is published with an individual key. Each published message is stored using the JavaScript Object Notation (JSON) format. A message example is presented in listing 3.

Listing 3: Message example (JSON format)

```
{ "timestamp": 1531427308000,
  "session": "2018-07-12T22:28:28+02:00",
  "id": "05d90cdf-b2f6-45d9-aafe-ffb96da96562",
  "name": "gps-bearing",
  "value": 51.0 }
```

The meaning of the record fields is as follows:

- *timestamp* – milliseconds since January 1, 1970,
- *session* – ISO 8601 format date and time,
- *id* – unique session identifier,
- *name* – parameter name,
- *value* – parameter value.

Finally, the collected data is stored in an InfluxDB time series database. InfluxDB is a NoSQL database. The data is stored in a structure called *measurement*. Each record has a key that is a timestamp of a measurement as well as fields that contain values and tags. Each measured parameter is stored in a different measurement. The session identifier is associated to every record as a tag, which allows such aggregations as averages.

3. Data set

The presented system was successfully used to collect real data from the Hyundai i30. The data set consisted of 44 hours of driving time. During each test drive, live data is obtained every second using OBD2. Then, a row is saved to a CSV file and uploaded to the server. A sample record is presented in listing 4. It contains the header of the file along with first line of data. In most cases, the name of the attribute describes its meaning quite well. *GPS Time* is the time based on GPS satellites, while *Device Time* is the smartphone-based time. $G(x)$, $G(y)$, and $G(z)$ denote acceleration in the x, y, and z axes, respectively. $G(calibrated)$ denotes the combined G value.

Missing information is represented as "-"; this may happen either when the OBD2 reader is plugged in but the engine is not running or when the given information is unavailable. Some values are primary and read directly from an ECU, but others are calculated by the Torque OBD. For example, both EGR values come from an ECU, but the GPS speed is calculated based on the GPS sensor in the smartphone.

Listing 4: Example data record

```
GPS Time,                                   Wed Dec 27 20:58:49 GMT+01:00
    2017
Device Time,                                27-gru-2017 20:58:48.388
Longitude,                                  20.4570165
Latitude,                                   50.80177848
GPS Speed (meters/second),                  8.24
Horizontal Dilution of Precision,           3.0
Altitude,                                   318.0
Bearing,                                    109.7
G(x),                                       0.86602783
G(y),                                       7.83947754
G(z),                                       4.78611755
G(calibrated),                              0.00805598
EGR Error(%),                               -
Barometric Pressure (from vehicle)(psi),    -
Intake Manifold Pressure(psi),              15.37400055
Fuel Rail Pressure(psi),                    -
Run Time since Engine Start(s),             -
```

```
Trip Time(while stationary)(s),            0
Trip Time(while moving)(s),                0
Trip Time(since journey start)(s),          0
GPS Bearing(°),                             109.69999695
Timing Advance(°),                          -
Liters Per 100 Kilometer(Instant)(l/100km), -
Horsepower (at the wheels)(hp),             -
Engine kW (at the wheels)(kW),              -
Torque(Nm),                                 -
Voltage (OBD adapter)(V),                   -
Voltage (control module)(V),                -
Engine Load(%),                             -
Engine RPM(rpm),                            -
Distance Travelled with MIL/CEL Lit(km),    -
Distance Travelled since Codes Cleared(km), -
Percentage of City Driving(%),              100
Percentage of Highway Driving(%),           0
Percentage at Idle(%),              0
Trip Distance(km),                          -
Trip Distance (stored in vehicle profile)(km), 491.32943726
Mass Air Flow Rate(g/s),                    17.04999924
Speed (OBD)(km/h),                          -
EGR Commanded(%),                           -
Ambient Air Temp(°C),                       -
Intake Air Temperature(°C),                 -
Engine Coolant Temperature(°C),             68
Turbo Boost & Vacuum Gauge(psi),            0.67400074
Trip Average KPL(kpl),                      -
Trip Average Liters/100 KM(l/100km)         -
```

To illustrate the quality of the collected data, let us focus on selected attributes. The engine coolant temperature is the only measure available in this car, which allows us to obtain the engine block temperature (more or less). Figure 2 presents the value of the attribute for a selected drive. Some cars have oil temperature gauges as well, but the test car is not equipped with such a sensor. The engine temperature influences fuel consumption (apart from many other factors, like driving style). Figure 3 represents the average fuel consumption during a test drive.

Other important information that can be read from OBD2 is the vehicle speed (an example plot is shown in Fig. 4). There are two values available – one is calculated based on the GPS position, and the other is received directly from an ECU. These two values may differ because OBD2 estimates the speed from the wheel speed (which is dependent on tire size). These values are often inconsistent.

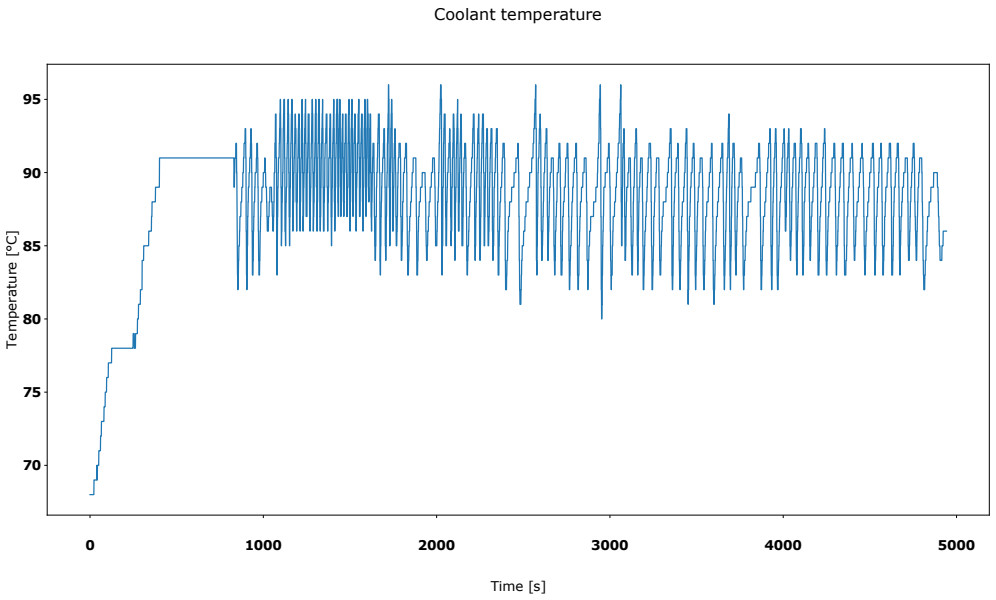

Fig. 2. Engine coolant temperature

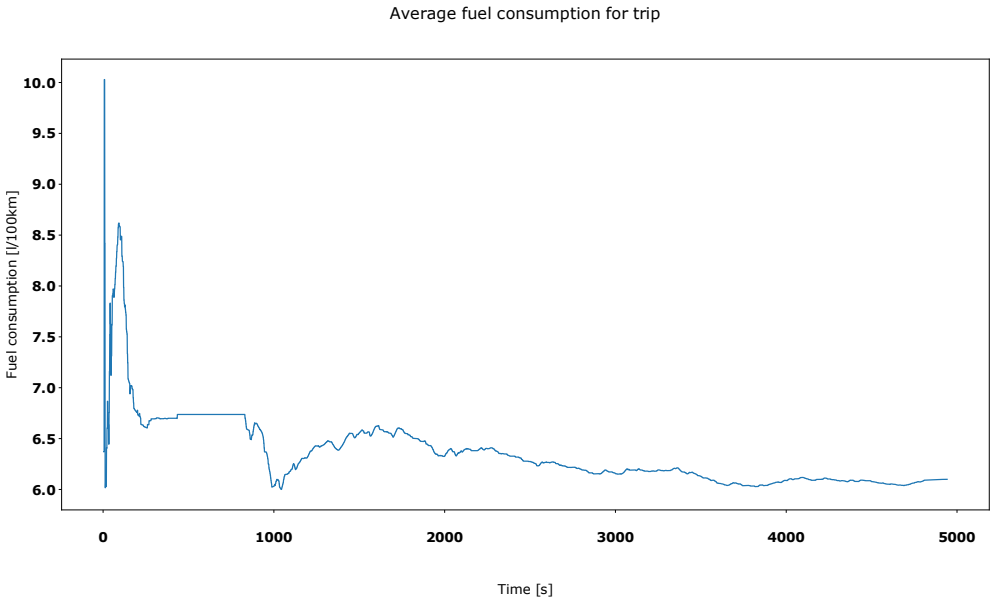

Fig. 3. Average fuel consumption

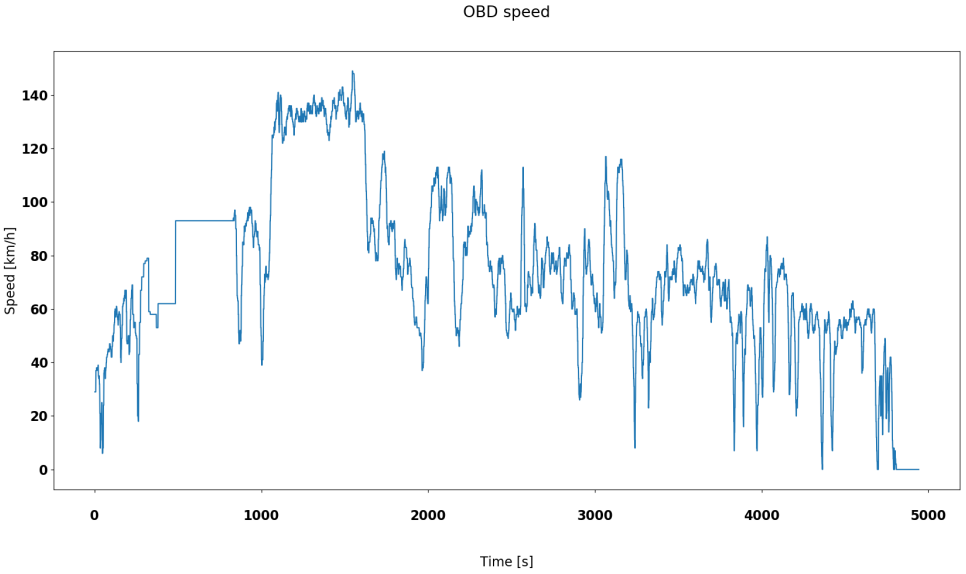

Fig. 4. Instant speed

4. Conclusion

The paper presents an online system for collecting sensor data directly from a car's ECUs. With the use of a piece of hardware, a smartphone, and an HTTP server, it is possible to gather holistic information on how your car is working. It should be underlined that the system components are quite inexpensive. The approach requires neither specialized software nor hardware. What is more, the developed web server allows us to handle multiple test drives at the same time, so scaling requires one smartphone and one OBD2 reader per car. The system can also be extended by on-the-fly big-data processing. This resulted in a scalable and flexible solution with an extremely low price.

The described data-acquisition system can be used to obtain a real data setfor further analysis with data-mining algorithms. We are working on a system that will allow us to detect defects in the work of selected car components based on the collected data. Moreover, we want to build a system the assesses driving style in terms of driving efficiency.

References

[1] Davis R.I., Burns A., Bril R.J., Lukkien J.J., *Controller area network (CAN) schedulability analysis: Refuted, revisited and revised*, Real-Time Systems, 2007, 35(3), pp. 239–272.

[2] Amarasinghe M., Kottegoda S., Arachchi A.L., Muramudalige S., Dilum Bandara H.M.N., Azeez A., *Cloud-based driver monitoring and vehicle diagnostic with OBD2 telematics*, in: *IEEE International Conference on Electro/Information Technology, EIT*, 2015, pp. 505–510.

[3] Tan P.-N., Steinbach M., Kumar V., *Introduction to Data Mining*, Pearson Education, Boston, MA, USA, 2006.
[4] Kowalik B., *Introduction to car failure detection system based on diagnostic interface*, in: *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, 2018, pp. 4–7.
[5] Kowalik B., Szpyrka M., *Architecture of on-line data acquisition system for car on-board diagnostics*, in: *MATEC Web of Conferences*, 2019, 252, 02003.
[6] Torque Wiki Plugin Documentation, https://torque-bhp.com/wiki/PluginDocumentation.
[7] OBD-II PIDs, https://en.wikipedia.org/wiki/OBD-II_PIDs.
[8] Akka HTTP documentation, https://doc.akka.io/docs/akka-http/current/index.html.
[9] Apache Kafka documentation, https://kafka.apache.org/documentation.

Środowisko on-line do akwizycji danych z sensorów samochodowych

Streszczenie: Współczesne samochody wyposażone są w dużą liczbę urządzeń elektronicznych określanych jako ECU (*Electronic Control Units*). Ich zadaniem jest pozyskiwanie danych diagnostycznych z podzespołów samochodu w celu kontroli ich pracy, ocena jakości pracy tych podzespołów i detekcja usterek. Mimo rozwoju komputerów pokładowych tylko niewielka ilość informacji pozyskiwanych z ECU jest przekazywana kierowcy. W artykule pokazano, jak można zbudować środowisko do pozyskiwania danych z samochodu. System, którego komponentami są m.in. smartfon z systemem Android i aplikacja Torque PRO, jest w stanie przesyłać dane do wybranego serwera przez Internet w czasie rzeczywistym. Zebrane dane można następnie analizować on-line lub off-line.

Słowa kluczowe: *sensory, motoryzacja, system akwizycji danych, aplikacja Torque PRO*

Guide for authors

Scope

Automatyka/Automatics publishes original papers in the field of automatic control engineering as well as in other related disciplines such as control theory and optimization, dynamical system analysis and synthesis, control engineering applications, electrical engineering, computer science, information technology, electronics, and biomedical engineering. Discussion and review papers presenting the current state of the art as well as new and emerging areas are also welcomed. The journal publishes letters to the editorial board, reviews, bibliographies, and biographies of distinguished researchers in pertinent fields, historical materials, etc.

Fees

Publication of scientific articles in *Automatyka/Automatics* is free of charge.

History

Automatyka/Automatics is a continuation of the non periodic series entitled *Zeszyty Naukowe AGH – Automatyka*, which included 67 volumes spanning 1966 through 1994. Since 1997, the journal has been published in a semi-annual cycle.

Submission method

Manuscripts should be submitted via our on-line Open Journal System (OJS), which is available at https://journals.agh.edu.pl/automat.

It is necessary for any corresponding author to be registered in the system in order to submit a manuscript. The registration form is available at https://journals.agh.edu.pl/automat/user/register. A prospective author should select the *Author*: *Able to submit items to the journal* check box at the bottom of the form during registration. Registered users can log in via the https://journals.agh.edu.pl/automat/login web page. At this stage, a list of a few AGH journals may be presented together with user's roles in each of them. The user should select the *Author* link under the *Automatics/Automatyka* heading. On the following page, one must select *CLICK HERE to go to Step One of the five step submission process* under the *Start a New Submission* heading to submit a manuscript. The corresponding author should provide all required metadata by filling in a series of successive forms and uploading the manuscript file.

Submission requirements

Initial stage (for review)

The corresponding author should upload a blinded manuscript as a single PDF file. The length of the document should not exceed 24 pages. The manuscript must represent a finished and complete work ready for review. Neither a copyright declaration nor any other supplementary files are required at this stage. To ensure a double-blind review, author name(s), affiliation(s), address(es), and thanks/acknowledgements must be removed from the PDF document. Metadata associated with the file should also be removed.

The submitting author is encouraged to provide a list of suggested reviewers who are recognized experts in a related discipline and preferably from countries and institutions other than those of the author(s). Their names, positions, affiliations, and contact information may be given in the *Comments for the Editor* section of the on-line submission electronic form. It is also possible to provide a list of non-preferred reviewers.

The submitting author should also enter the names and affiliations of all co-authors of the manuscript.

Resubmission for second review (if applicable)

In some cases, a *Resubmit for review* decision is announced to the corresponding author. In such a case, a major revision of the manuscript is required in accordance with the comments and suggestions provided by the reviewers. The author is then required to prepare a revised version of the document in a PDF format. Answers to the reviewers' remarks (a separate file for each reviewer) should also be provided. All files should be compressed and uploaded as a single ZIP archive via the on-line OJS system. The file should be added to the existing submission. Neither a copyright declaration nor any other supplementary files are required at this stage.

Final stage (for copy editing)

If the decision is *Revisions required*, the author(s) should carefully read all of the provided reviewers' or editor's remarks and improve the manuscript accordingly. If the decision is *Accept as is*, no modifications of the manuscript body are necessary. In both cases, the document should be supplemented with author name(s), affiliation(s), and e-mail address(es) as well as an abstract and a keyword list. *Acknowledgements* and *About authors* sections may be added after the *Conclusions* section to provide thanks, sources of funds, and author biographical notes, as the paper no longer needs to be blinded.

The corresponding author should then prepare a single ZIP archive file containing the following:

– camera-ready PDF file;
– document source file(s) in the LaTeX (recommended) or Microsoft Office Word (acceptable) format;
– images, charts, and figures in separate files (vector graphic formats preferred); in the case of raster graphics, a resolution of 300 dpi or better is required;

– a signed and scanned copyright declaration (a template form is available on the journal website at http://automatics.agh.edu.pl in *Guide for Authors* section).

Polish-speaking authors are requested to provide a paper title, abstract, and keyword list in both English and Polish. All of this data should be included in the paper body.

The camera-ready PDF document and its LaTeX or MS Word source file(s) should contain the following:

– author(s) name(s),
– author(s) affiliation(s) (university/institution, faculty/department, street address/city/state/country/postal code),
– author(s) e-mail address(es),
– paper title,
– abstract,
– keyword list,
– *Introduction* section,
– main part divided into sections and subsections (if applicable), with figures and tables included,
– *Conclusions* section,
– *Acknowledgments* section (optional: funding sources may be provided here),
– *About author(s)* section (optional: short biographical notes may be given here),
– *References* section,
– paper title in Polish (Polish-speaking authors only),
– abstract in Polish (Polish-speaking authors only),
– keywords in Polish (Polish-speaking authors only).

The ZIP file should be uploaded to the OJS system. Please add the file to the existing submission (do not start a new one).

Once the copy-editing process is finished, a galley proof will be sent to the corresponding author via e-mail for careful checking and approval. This concludes the publishing process as far as the authors are concerned.

Review process

Rules:

– For each manuscript, at least two independent reviewers will be assigned.
– Preferably, at least one of the reviewers should work in a country different from the countries of the authors' affiliations. The reviewers should also be associated with institutions other than those of the authors.
– The corresponding author may provide a list of suggested reviewers who are recognized experts in the related field.
– The papers are reviewed in a double blind process.

- The review process concludes with one of the following decisions: *Accept as is*, *Revision required*, *Resubmit for review*, or *Reject*.
- A reviewer provides her or his input by filling in the electronic form in the on-line OJS submission system available at https://journals.agh.edu.pl/automat.
- The reviewer should register in the system (declaring a reviewer role) and log in via the Internet website.

Further information

E-mail messages to the editor should be sent to automatics@agh.edu.pl

Manuscripts should be submitted via the online system available at https://journals.agh.edu.pl/automat

Articles from past issues are freely available from the archive at http://journals.bg.agh.edu.pl/AUTOMAT/index-en.php

Most up-to-date information can be found on the journal website http://automatics.agh.edu.pl