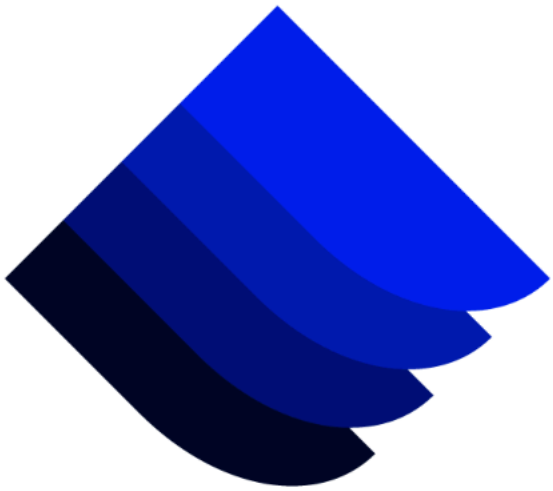


INDIGO

Low-code platform for integrating and controlling machines
and IT systems based on graphical process modeling.

Where are we headed and why?
To find methods that enable people to communicate with machines in an effective and comprehensible way is among the most ambitious pursuits of contemporary computer science. Programming languages are more and more like natural languages, which facilitates and significantly expedites the process of building reliable software.

The history of programming began with a machine language in which commands were input directly into the processor in the form of zeros and ones (binary code). However, this meant that separate instructions had to be written for each processor. Next, low-level programming languages emerged which supported operations controlled by symbols. This trend continued in subsequent languages: C and C++ that were increasingly departing from the machine code.

They were much more universal and handled a wide range of problems. On top of that, these languages, some of them still used and developed today, entailed significant time savings in the process of building applications and helped reduce the number of errors thanks to the automation of certain operations previously carried out by programmers. Primarily, however, they allowed the separation of software from the hardware layer, which resulted in software transferability, i.e. the option of using the same code on different hardware platforms or in different operating systems.

We have gone even further and have developed INDIGO platform: a technology that integrates machines and IT systems and controls them using process graphs. Indigo can operate on different platforms and, more importantly, connects machines, IT systems, PLCs or automated solutions in such a way that the individual nodes not only support free information interchange, but also control each other's operation.

What is more, programming in INDIGO is based on modelling individual processes on graphs in the Business Process Model Notation (BPMN 2.0). What is the advantage of this approach? While in traditional programming languages the smallest independent piece of code is in fact a list of simple instructions that have not evolved much since the Turing machine, in the case of our solution, it is a ready element of a process that performs a specific function, e.g. "Display window with message" or "Send command to device."

Consequently, process modelling in INDIGO comes down to designing its progress on a graph. Individual graph components can be server commands to write or read data or to perform a specific function; they can also be commands sent to devices to perform a previously defined action; finally, they can also initiate bidirectional communication with IT systems.

When building software in Indigo, we give priority to the source code functions that are triggered by some individual elements of the graph over the source code as such.

These graphs are our programming language, readable not only by programmers.

INDIGO - a platform for visual programming
The use of graphs for programming makes INDIGO an LCDP [low-code development platform]. The individual elements of the graph are like Lego blocks from which we build our applications agile and quickly.

The INDIGO platform meets all the low-cod assumptions: it significantly facilitates and speeds up the development of applications, operates on the basis of global supported standards, is open to integration with other systems and allows for agile application development and enables its easy development. Additionally, in INDIGO, graphs provide the possibility of modelling even the most complex processes and thus enable the construction of advanced IT systems fully adapted to the needs.

Meanwhile, there is one feature that distinguishes INDIGO from all LCDP platforms - this distinguishing feature is the area of application and the type of application that can be created with it. **INDIGO is the only LCDP platform for creating applications for the industry that allows you to create software that integrates hardware modules and IT systems.**

Why does it work so well? About the architecture and technology
The INDIGO architecture is modular and transparent. It has been designed as open to further development. It also supports horizontal scaling, i.e. it can enhance software productivity by adding additional servers (virtual/physical). In addition, the system runs built-in data aggregation mechanisms, in other words, it is capable of accepting a large volume of data (from many devices), at a very high frequency.

Going to a higher level of detail, the architecture of INDIGO can be described as based on distributed nodes responsible for specific tasks related to the control, integration as well as collection and processing of data originating from machines and users. Each of the nodes is independent and after disconnecting from the network, e.g. in the event of failure, it keeps collecting and processing information in order to send it out immediately after communication has been restored.

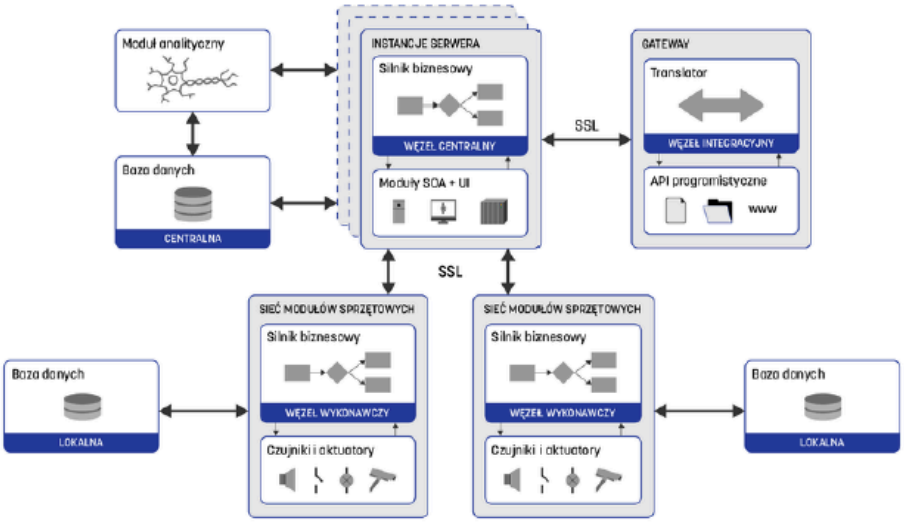
A noteworthy element of the system is the central node. It can operate in many instances and enables a high scalability of the solution.

The design of the individual system levels and the programming of processes with graphs enable real-time modification of the operation logic of any system component or its replacement with another one having the same functional scope.

On the other hand, isolation of the individual system levels and programming of processes with graphs facilitate real-time modification of the operation logic of any system component or its replacement with another one having the same functional scope.

The architecture so designed meets all the demands of Industry 4.0. In-production changeovers are now done in real time; short-series production is made easier, higher quality and optimisation is now achievable at every level of production, facility, and the entire organisation.

Fig. 1
Architecture



To enable users to take advantage of best-fitting hardware and software solutions from different vendors, while, at the same time, preventing vendor lock-in, INDIGO is entirely based on licence-free OpenSource technologies (MIT or equivalent). The platform works on GNU Linux operating systems and uses proven, active and mature solutions, such as PostgreSQL database server, ORQ SQLAlchemy, framework Vue.js, nginx HTTP server and Python 3.7 and JavaScript/TypeScript programming languages.

The security of data processed by Indigo is ensured by asymmetric encryption of communication by the SSL/TLS protocols.

Business engine
The heart of the INDIGO platform is a light process engine programmed using BPMN 2.0 graphs. The engine can operate on IoT devices by controlling their actions and communication with other nodes and the server.

The engine's design makes it more universal than a tool usable only in selected areas of operation. It can call any functions, control interfaces and carry out synchronous and scheduled tasks. The engine performs its tasks based on loaded graphs.

Since the INDIGO technology is based on scripting languages, the software can be run on any devices operating in the x86 or ARM architecture and on microcontrollers. INDIGO non-server nodes can also be run on Android devices.

All data can be saved (each node has its own database), which means the interrupted process can be continued after restart. Currently, the system runs the PostgreSQL database, but thanks to the use of the SQLAlchemy indirect layer, any other modern relational database can be connected.

The engine supports multiple processes both synchronously and asynchronously at the same time and enables their CRON scheduling. Messages can be queued when off-line and sent after restoring communication.

Graphs
The INDIGO platform is based on graphic programming. Graphs are used to programme each logical layer - from controlling hardware modules, through the description of the logic of operation of many modules accompanying the device, information workflow within a set of devices, to transferring information between various groups of devices and IT systems, also in remote locations.

Graphs also control the interface activities, i.e. showing and hiding windows, internal operation logic, and displayed content. At the system level, defined are only templates of the structure of the view and components that make up tables, text, or buttons.

Graphs define the response to interface and hardware messages as well as to other processes by defining action to be taken, its execution conditions, and messages to be sent. They also define errors and off-line operation.

Graph operations involve calling of functions from generic, low-level libraries, e.g. support for variables, communication through ports, etc. and high-level libraries, e.g. a downtime logging module.

We create graphs using our proprietary, dedicated IDE [Integrated Development Environment] tool.

Integration with external IT systems
INDIGO can be integrated with external systems in two ways.

The INDIGO platform connects to external systems that provide APIs of any type (SOAP, SQL, REST, etc.) by creating REST integration nodes. The mechanisms used in INDIGO to create these nodes allow for:

- validation of input and output data,
- displaying the interface for development calling of node integration points - useful for both developers working with INDIGO and external applications,
- running the node in test data mode without connecting to the target system.

Deployment
Indigo fully supports continuous deployment enabling, for example, remote uploading of views and resources as well as graph replacement. The system collects logs from all nodes in real time and in a centralised manner.

The platform is fully modular, i.e. only the modules required for operation are deployed.

Interface or how to communicate with the user
High flexibility and configurability of INDIGO is also seen in its interface. It has been designed in line with the latest trends in web application development.

It features desktops and widgets that make up management dashboards. The interface structure enables the presentation of various types of information in a variety of graphical and functional layouts that correspond to the needs of a specific implementation. Creating custom screens is simple and fast: they are designed at the interface level and without configuring files or performing additional programming (WYSIWYG). This means that the interface can be dovetailed with the needs of each user. It is also possible to create a uniform interface for a non-homogeneous machine park.

Graphs are able to control the interface
INDIGO fully supports autorisation mechanisms, such as role assignment and combination, internalisation, modification of the size and resolution of interface components. The interface look-and-feel can also be customised (also during operation), e.g. the colour scheme, icon/button appearance, etc.

How we do it

Fig. 2
A template describing the behaviour of the view.

```
<template-section id='body_section' type='vertical' scheme='fullscreen'>
  <template-section id='header_section' scheme='header'>
    <info-picture id='stabilis_logo' icon='images/stabilis-logo.png' position='start' left='--separation-big' right='--separation-big'/>
    <info-line id='console_info' position='start' left='--separation-big' right='--separation-big' interspace= '--separation-big' />
    <date-and-time id='clock' position='end' right='--separation-big' stick='end' />
  </template-section>
  <template-section id='main_section' type='vertical' >
    <template-section id='cuts_section' align='start' >
      <template-section id='table_section' type='vertical' size='stretch'>
        <info-table id='cuts_table' grow='expand'/>
        <info-line id='comments' size='--font-big'/>
      </template-section>
      <template-section id='input_section' type='vertical' align='end' grow='none' size='stretch'>
        <input-number id='input_1kat' />
        <input-number id='input_2kat' stick='end'/>
        <input-number id='input_waste' bottom='--separation-small'/>
      </template-section>
    </template-section>
    <template-section id='machine_section' size='stretch' grow='none' >
      <info-box id='machine_info' />
    </template-section>
    <template-section id='actions_section' scheme='commands'>
      <execute-command id='actions_navigate' />
      <execute-command id='actions_cuts' />
      <execute-command id='actions_machine' />
    </template-section>
  </template-section>
</template-section>
```

Fig. 3
A program section in the form of a graph describing the behaviour of the view.

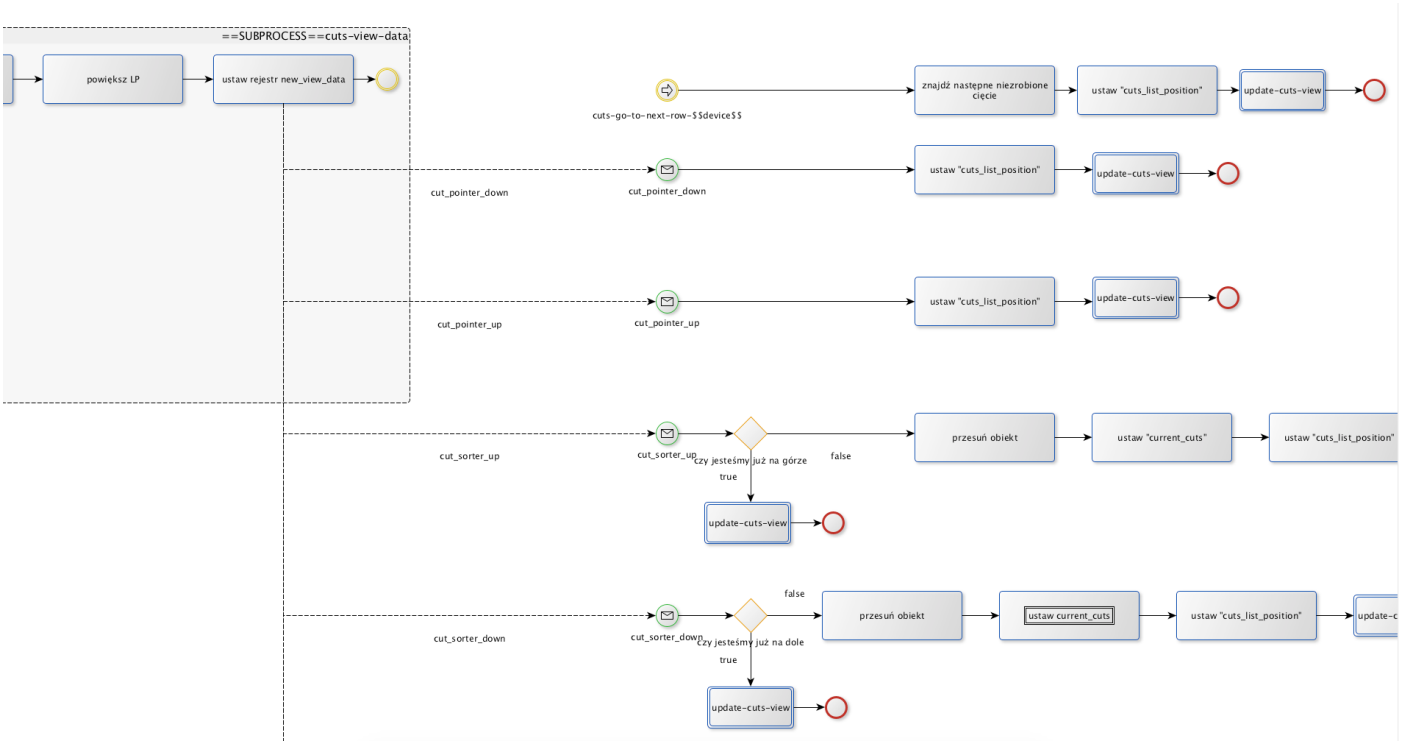
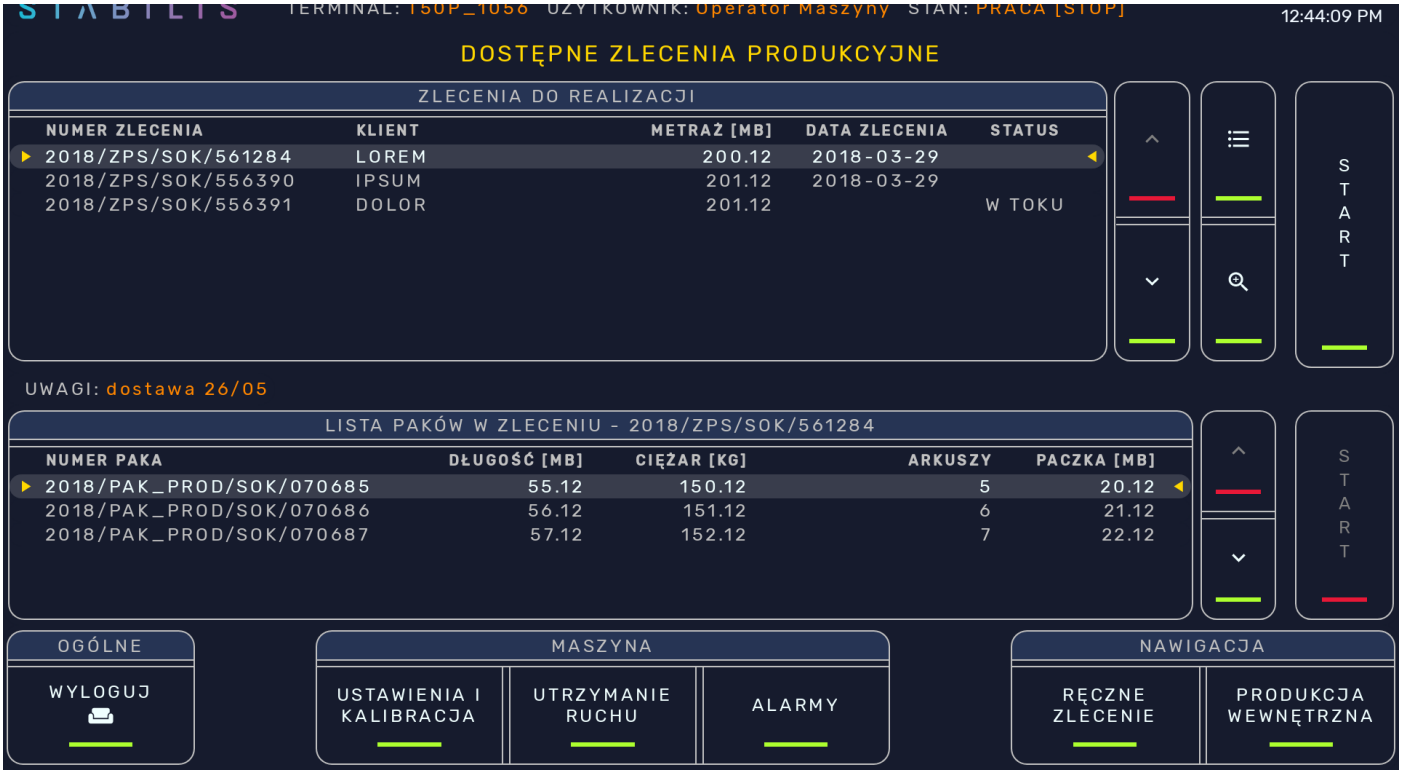


Fig. 4
Example of a system screen responsible for machine control and parameter display. The view and behaviour are fully described by means of a template and process graph.



End-to-end tests
Applications written in INDIGO are tested using test scenarios written in graphs. As part of the test scenario, it is possible to:

- simulating user actions on the interface,
- simulating signals from peripheral devices (barcode scanners, etc.),
- simulating signals from PLC,
- using test data in integration nodes,
- checking changes on the user interface,
- checking queries to integration nodes,
- checking messages to peripherals and PLC.

What makes us stand out?
Programming with INDIGO offers unique possibilities:

- enables the creation of stationary applications with the offline function,
- allows you to create complex applications because it models processes, not data flow,
- allows you to freely choose the version control system,
- has a built-in reporting module,
- enables remote monitoring of the workplace and the reconstruction of user activities,
- has a built-in module for integration with external systems,
- has integrated mechanisms for testing and local development data.

What is the conclusion?
The INDIGO platform provides many benefits, significant from the point of view of trends and needs related to the development of applications for the industry. The most important are:

- unique flexibility allowing the creation of dedicated solutions and implementation of very complex applications,
- significant acceleration of application development,
- ease and security of introducing changes to a running application that were not foreseen at the time of designing and implementing the software,
- stability,
- reduction of logic, regression and integration errors,
- significant reduction of the cost of changing requirements during the project,
- the possibility of evolutionary application development in accordance with agile software production methodologies,
- possibility of project implementation by personnel with lower competences (implementers instead of programmers),
- facilitating and accelerating the analysis process with the client.

Where can the INDIGO platform be used?
INDIGO flexibility affords the opportunity of using the framework in various areas. Our technology will prove effective wherever there is a need to collect, analyse and respond to large volumes of data from many devices. Regardless of whether this is about integrating machines in a factory, building automation systems or different hardware modules within one device (e.g. vending machine), we always address your individual needs.

Examples of application areas:

- data collection, analysis and control of industrial automation,
- software development for prototype production lines,
- creating advanced service vending machines and their networking (vending machines, car / bike sharing etc.),
- creating building management systems (integration of building systems, automation control)

STABILIS®MES
Global digital transformation is making significant inroads into the industrial sector. A growing demand for tools for production and business process optimisation has inspired us to create the first product based on our original Framework: software enabling quick development of custom systems integrating machinery and IT systems, i.e. a data-driven and modern management approach to running a production company. We call it STABILIS®.

Summary
What distinguishes INDIGO is its unique flexibility, which allows you to quickly create dedicated applications fully adapted to your needs and ensures that it is easy to introduce changes to the working application in accordance with the emerging needs.

Applications developed in INDIGO enable two-way communication with machines or IT systems programmed with graphs. This means that we can not only collect data, but also respond to it by sending feedback commands for machines or feedback for IT systems (e.g. ERP).

Because each process running in any system or network of machines is programmed with a graph, the logic of the system can

be modified almost in real time on our software: the only thing to do is to replace the relevant graphs without having to write hundreds of lines of code.

Graphs also make our software more reliable and better fit for purpose because the functional design is understandable for the client and modifiable even at the implementation stage when practice shows that certain processes should be designed differently than originally planned.

Through this approach, Indigo software stands out as exceptionally flexible, i.e. having the capacity to implement quick changes, being open to integration with new technologies and new functions that can be added by configuring our Framework without creating additional software. In a nutshell, we fully apply the idea of rapid application development (RAD).

The applications we create are primarily used to actively optimize business processes, hence the natural path to use our technology is the implementation of AI, machine learning and data science mechanisms.