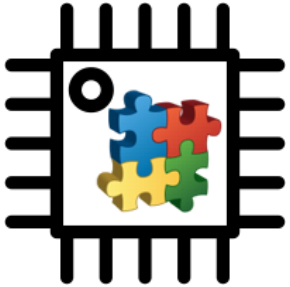


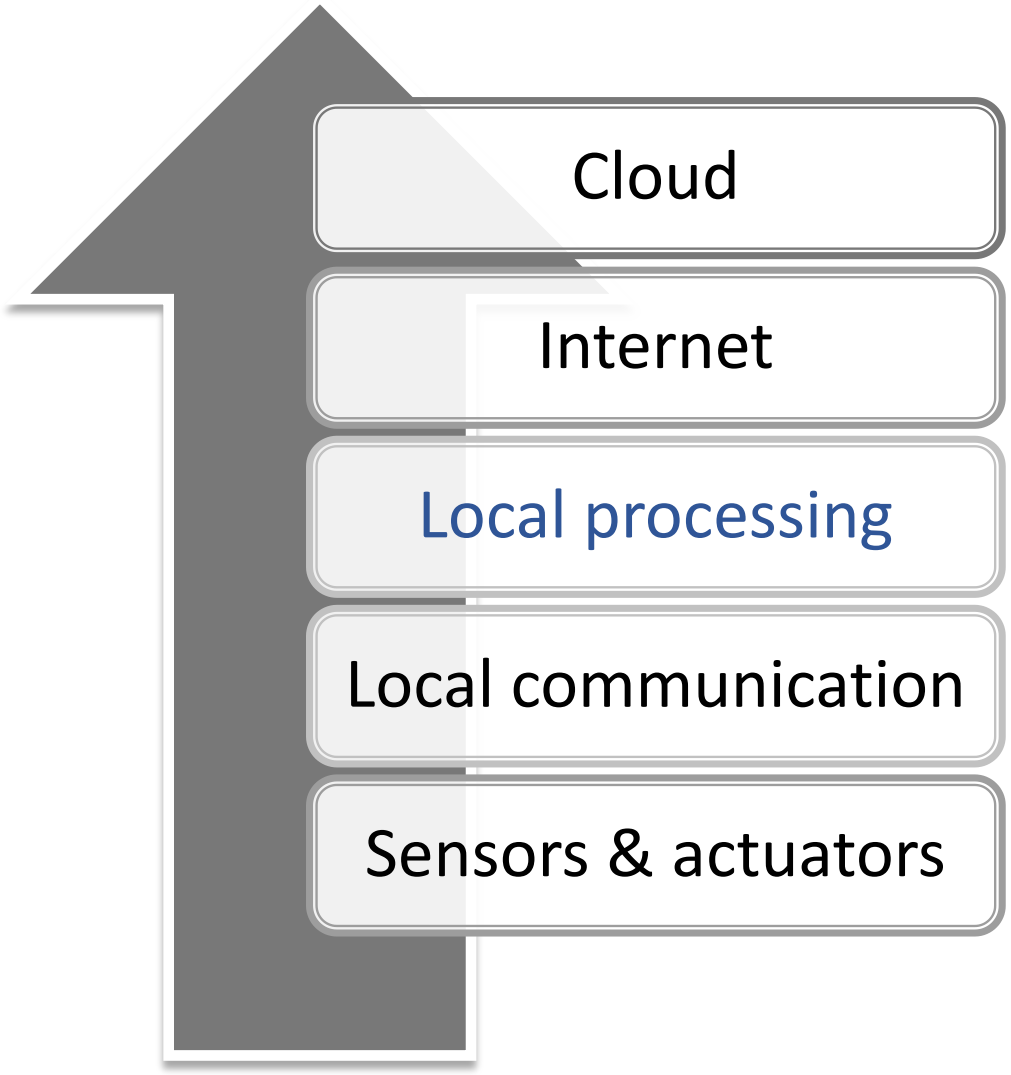
ACP

Arduino Component Programmer

Component-Oriented **Event**-Driven **Programming**

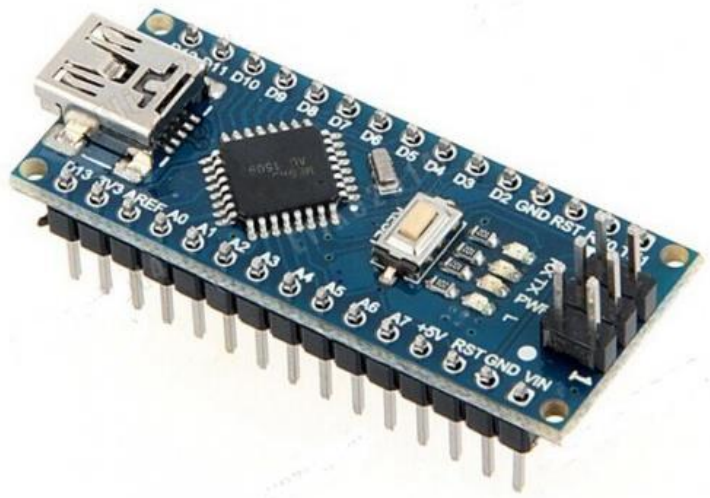
ŠVK 2018 – Patrik Pekarčík

Why smarter microcontrollers?



IoT - shift:

- **fog/edge computing**
- **local processing** = more complex local code (e.g. firmware in MCU)



OpenFog

arm

cisco

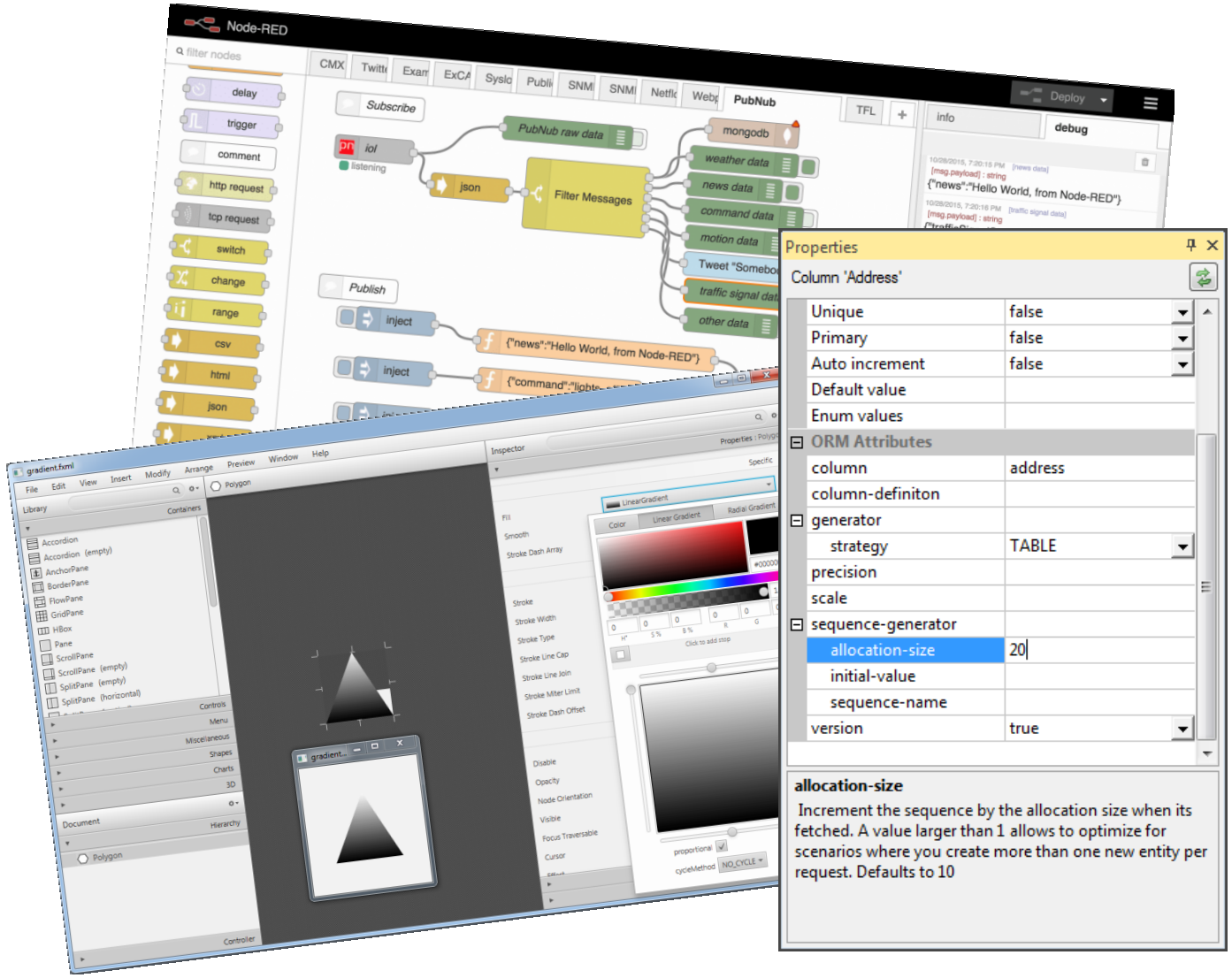
intel

DELL

Microsoft

PRINCETON UNIVERSITY

Complex code? No problem in 2018

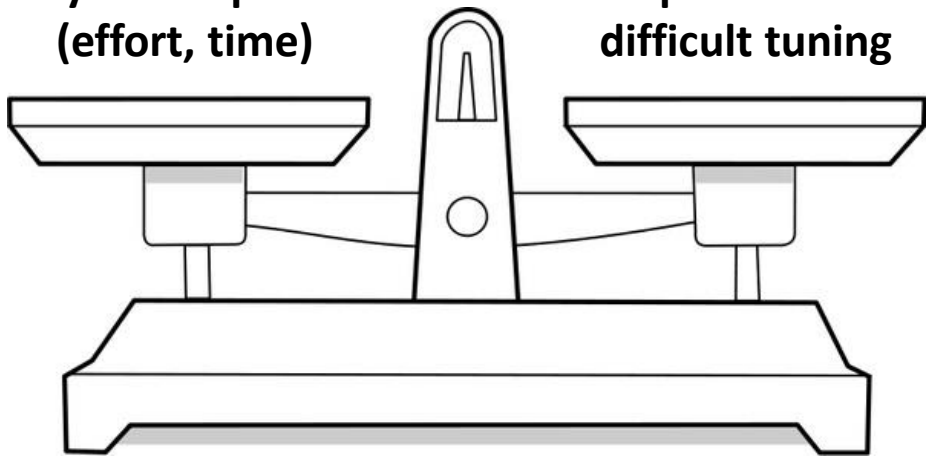


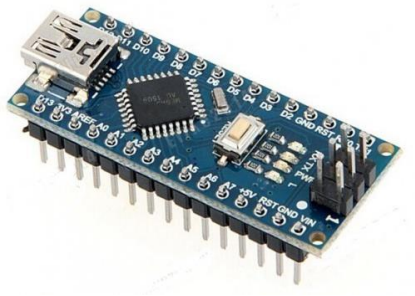
- **IDE support**
- **Visual editors** for everything:
  - configurations
  - window forms
  - scenes/activities
  - IoT flows
- Click, configure, generate (an application)

Trade-offs

Reusability, easy development (effort, time)

General and non-optimized code, difficult tuning





ARDUINO™

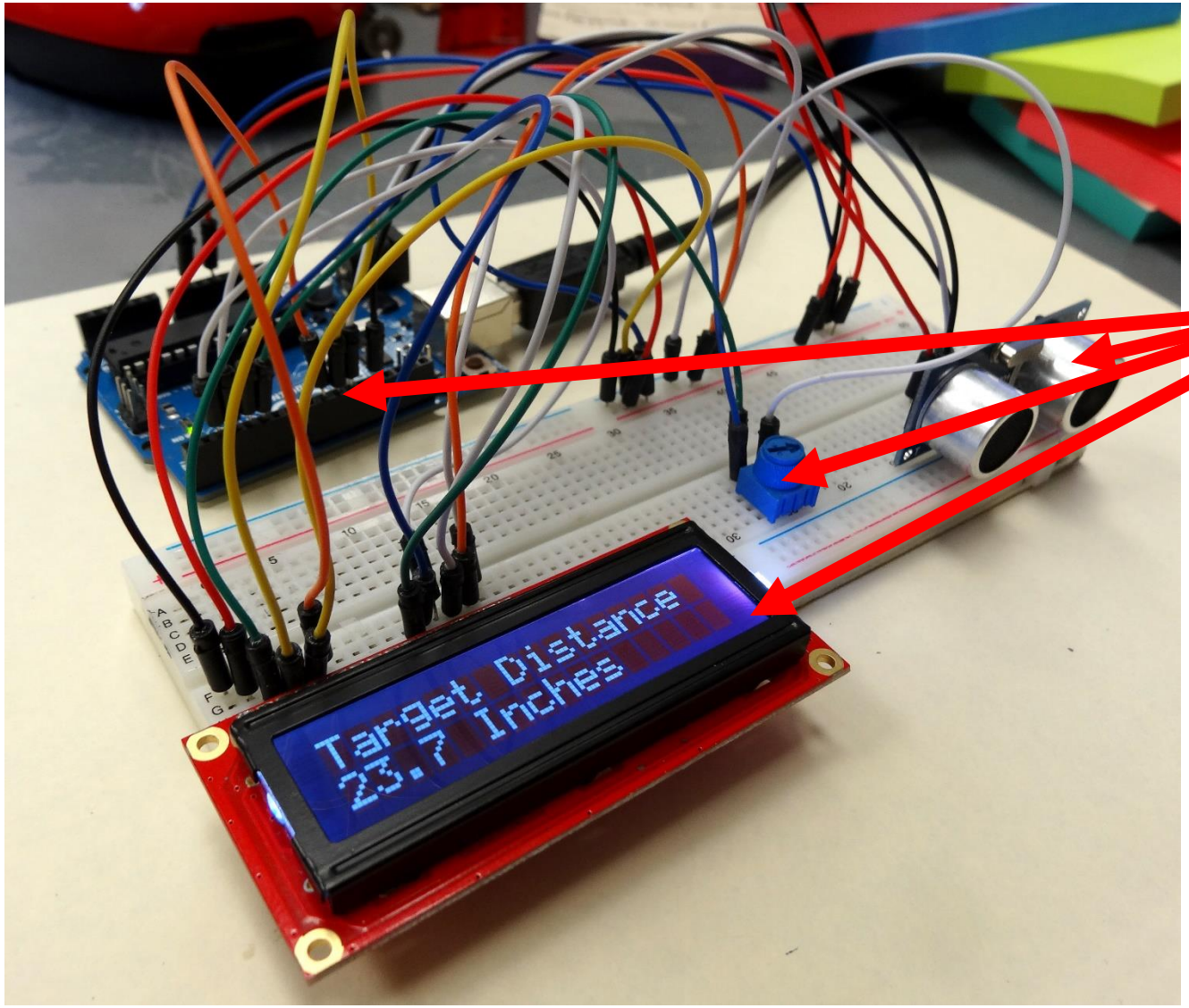
Flash memory	**32 KB** of which 2 KB is used by bootloader
SRAM	**2 KB**
Clock speed	**16 MHz**

Clear winner in the ”big world”

but

MCU winner: old-school C/C++ programming

No pins, no loops, no interrupts – only components



Device

=

Component (of a Type)

=

(Properties + Events + Methods)

ACP project

=

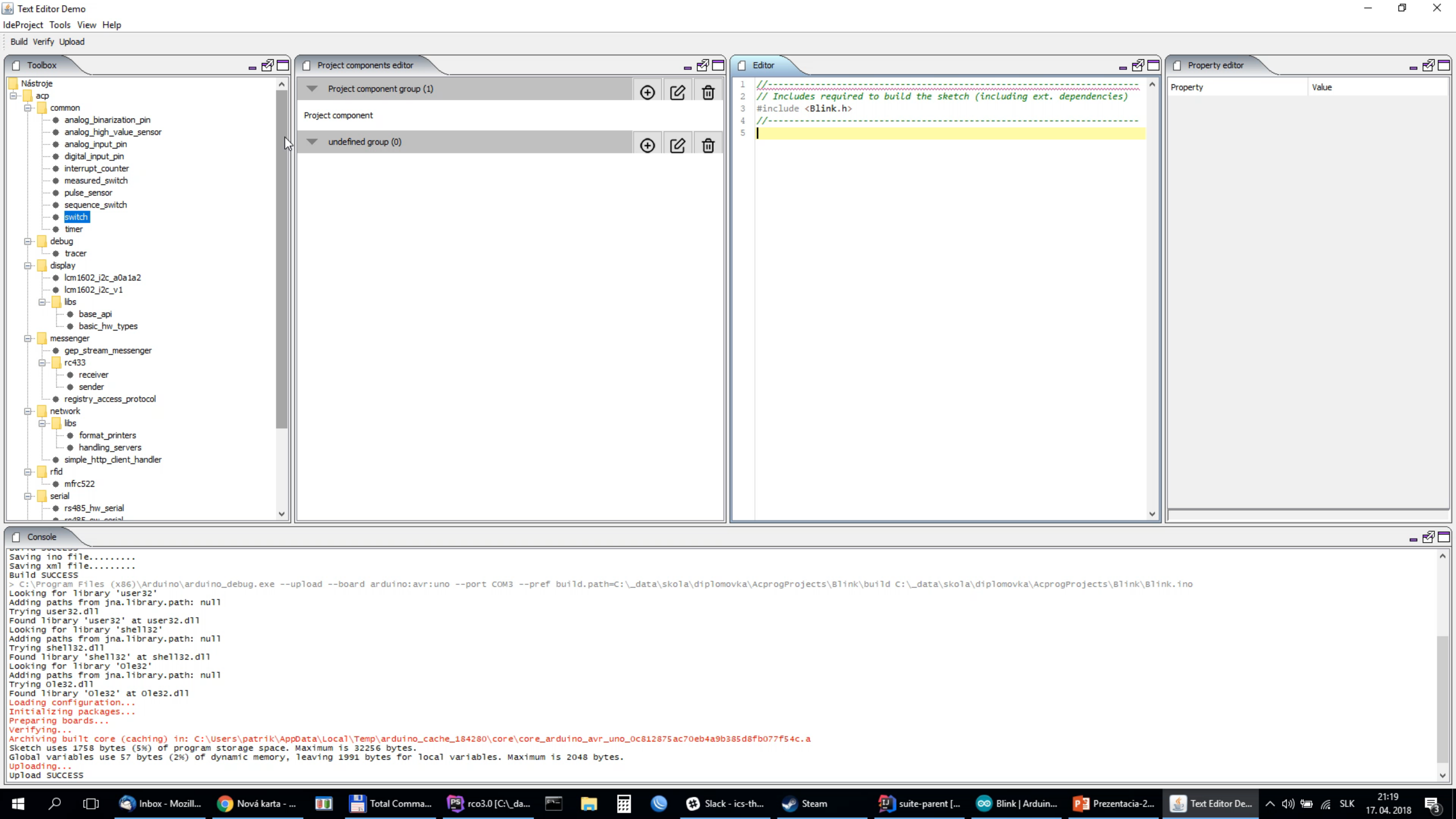
Configuration

+

C/C++ event handlers

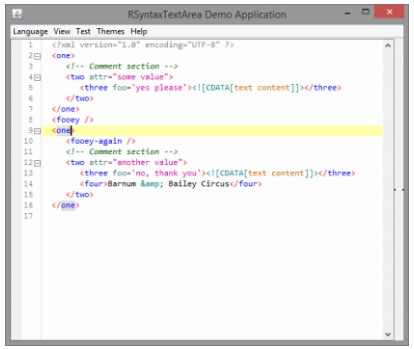
ACP suite

- **Code generator**
  - compiles ACP project and uses the Arduino IDE to code handlers/deploy
  - **lightweight** and **optimized** (generated) application core with simple API
    - verified in several IoT projects
- **IDE** - all in one
  - visual configuration (D&D), **intelligent code editor**, deployment
  - simple but powerful – for beginners and advanced IoT creators
- **Component library** – “Maven for ACP“
  - make your **own components**:
    only file structure + xml config + C/C++ code


Text Editor Demo
IdeProject Tools View Help
Build Verify Upload
Toolbox
Nástroje
acp
common
analog_binarization_pin
analog_high_value_sensor
analog_input_pin
digital_input_pin
interrupt_counter
measured_switch
pulse_sensor
sequence_switch
switch
timer
debug
tracer
display
lcm1602_i2c_a0a1a2
lcm1602_i2c_v1
libs
base_api
basic_hw_types
messenger
gep_stream_messenger
rc433
receiver
sender
registry_access_protocol
network
libs
format_printers
handling_servers
simple_http_client_handler
rfid
mfrc522
serial
rs485_hw_serial
Project components editor
Project component group (1)
Project component
undefined group (0)
Editor
1 //--
2 // Includes required to build the sketch (including ext. dependencies)
3 #include <Blink.h>
4 //--
5
Property editor
Property
Value
Console
Saving ino file.....
Saving xml file.....
Build SUCCESS
> C:\Program Files (x86)\Arduino\arduino_debug.exe --upload --board arduino:avr:uno --port COM3 --pref build.path=C:_data\skola\diplomovka\AcprogProjects\Blink\build C:_data\skola\diplomovka\AcprogProjects\Blink\Blink.ino
Looking for library 'user32'
Adding paths from jna.library.path: null
Trying user32.dll
Found library 'user32' at user32.dll
Looking for library 'shell32'
Adding paths from jna.library.path: null
Trying shell32.dll
Found library 'shell32' at shell32.dll
Looking for library 'Ole32'
Adding paths from jna.library.path: null
Trying Ole32.dll
Found library 'Ole32' at Ole32.dll
Loading configuration...
Initializing packages...
Preparing boards...
Verifying...
Archiving built core (caching) in: C:\Users\patrik\AppData\Local\Temp\arduino_cache_184280\core\core_arduino_avr_uno_0c812875ac70eb4a9b385d8fb077f54c.a
Sketch uses 1758 bytes (5%) of program storage space. Maximum is 32256 bytes.
Global variables use 57 bytes (2%) of dynamic memory, leaving 1991 bytes for local variables. Maximum is 2048 bytes.
Uploading...
Upload SUCCESS
Inbox - Mozill...
Nová karta - ...
Total Comma...
rco3.0 [C:_da...
Slack - ics-th...
Steam
suite-parent [...
Blink | Arduin...
Prezentacia-2...
Text Editor De...
SLK
21:19
17. 04. 2018

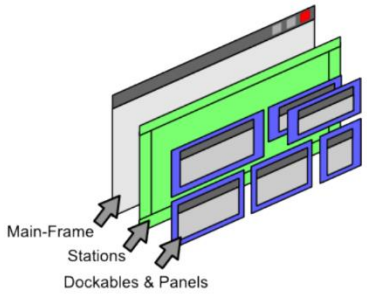
Technologies

+ Java Swing



RSyntaxTextArea

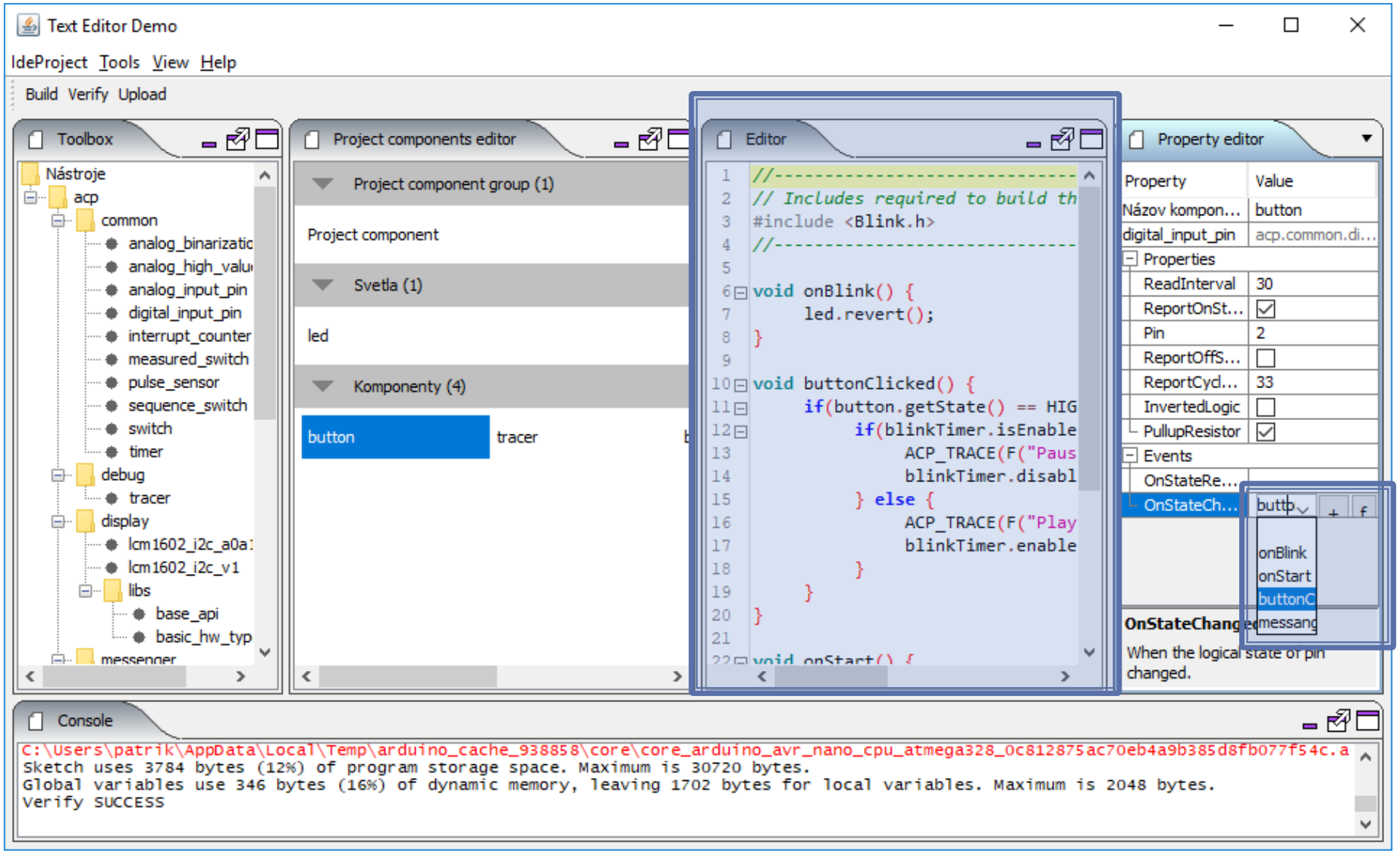
IDE & Compiler



DockingFrames

JFlex

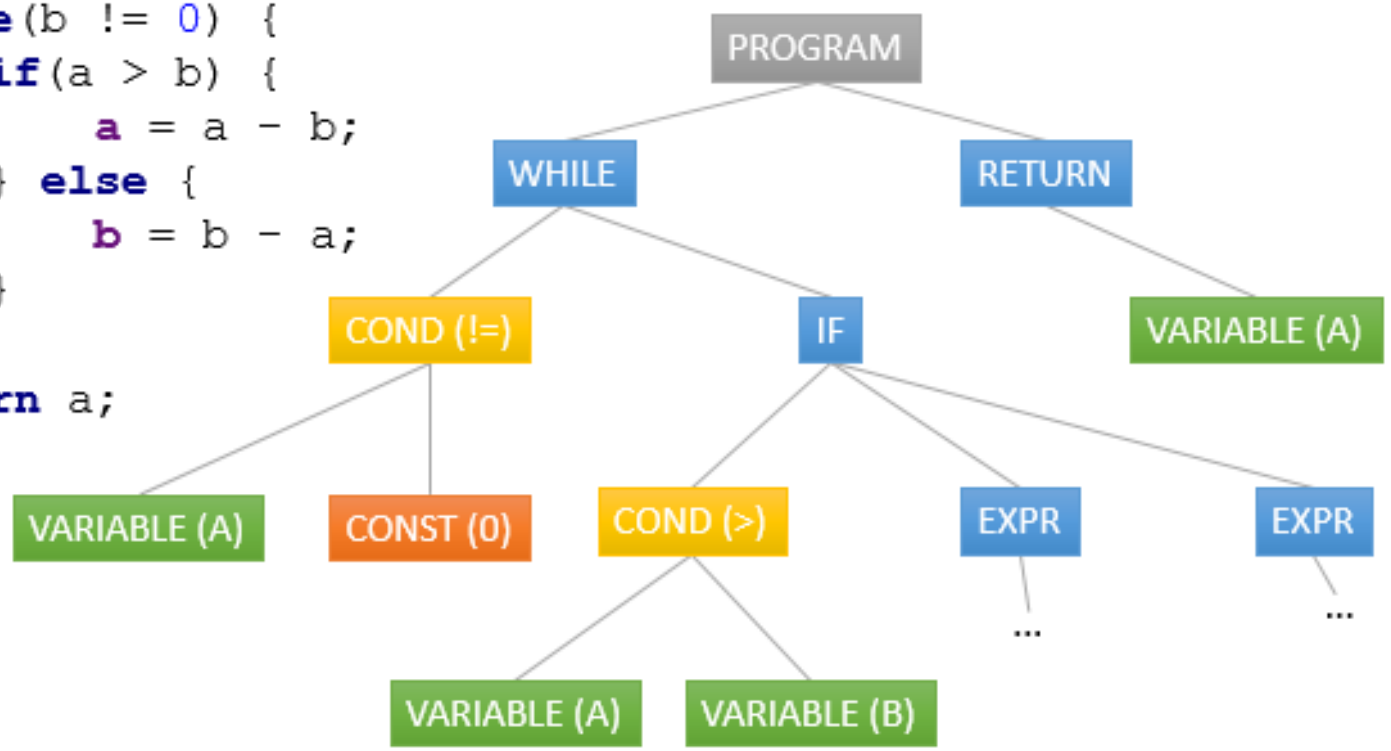
Abstract syntax tree



Abstract syntax tree

- C++ language grammar
- Lexical analysis
- Source code validation

```
while(b != 0) {
    if(a > b) {
        a = a - b;
    } else {
        b = b - a;
    }
}
return a;
```




```
terminal WHILE, LPAR, RPAR, SEMICOLON;

...

non-terminal condition, statement, iterationStatement;

...

iterationStatement
    ::= WHILE LPAR condition RPAR statement
    |   DO statement WHILE LPAR condition RPAR SEMICOLON
    ;

condition
    ::= postfixExpr
    ;

...
```

Not just a tree - **Metadata**

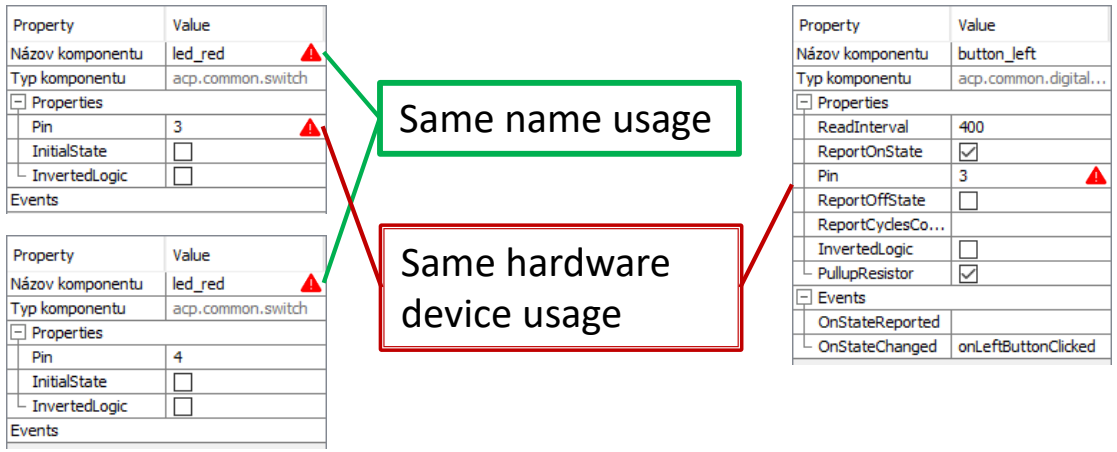
Functions

Variables

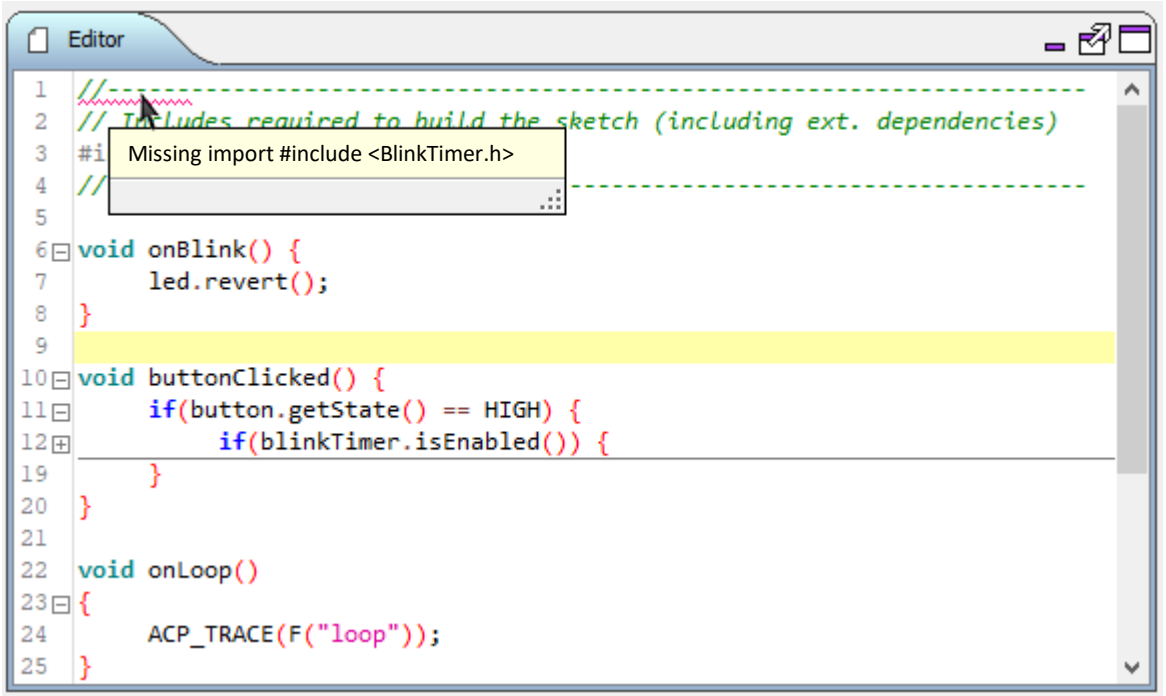
Includes

and much more...

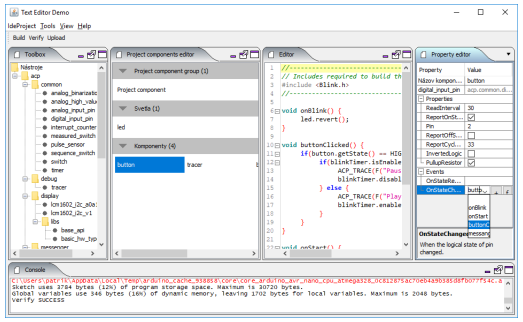
Benefits of AST

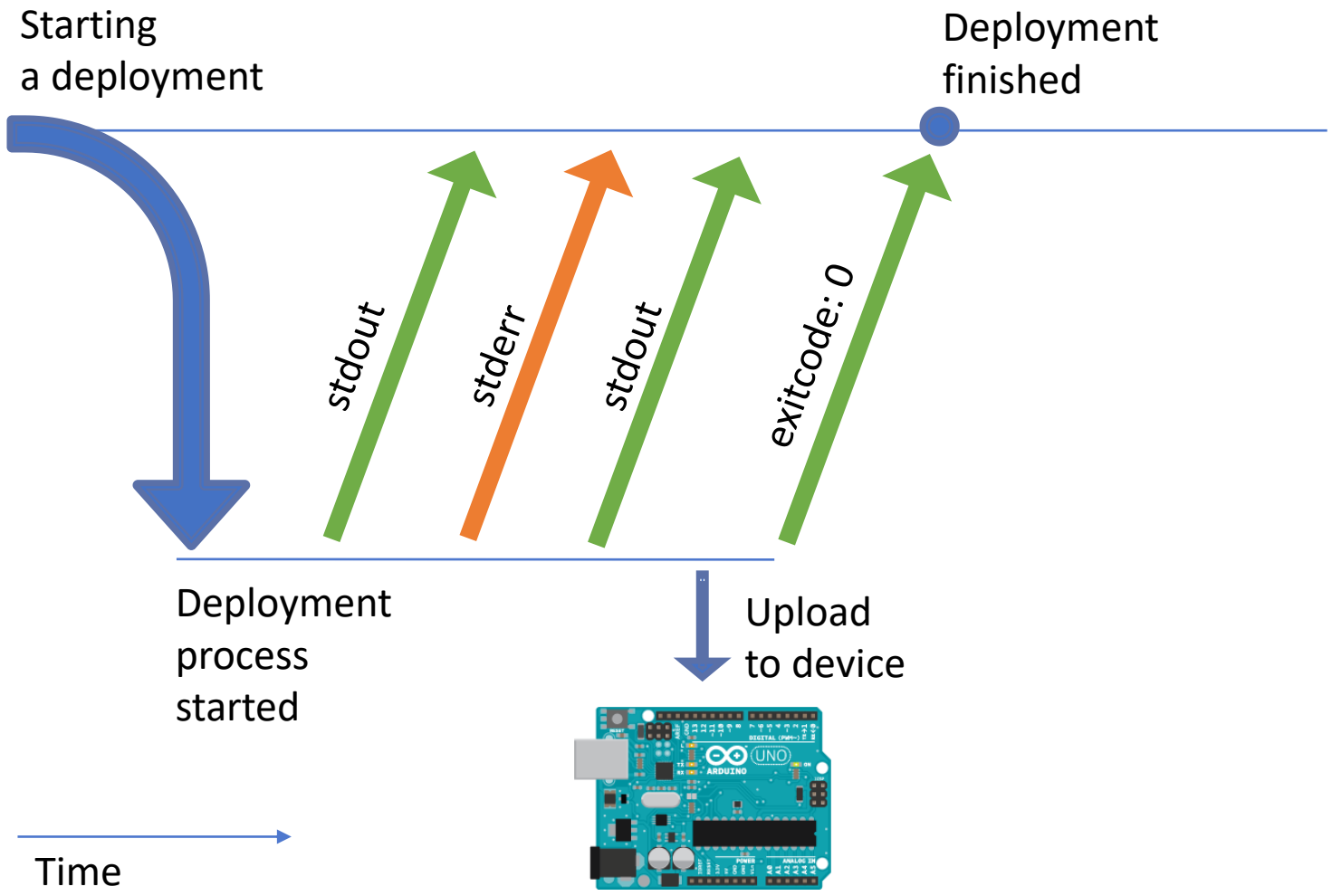


- Source code verification
- Event implementation parameters check
- Check properties pairing
- Missing imports
- Autocomplete
- Searching implementations



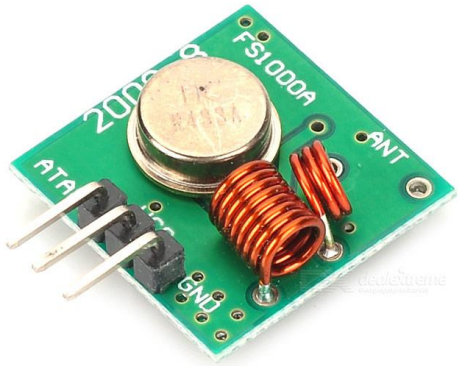
Deployment – inter-process communication

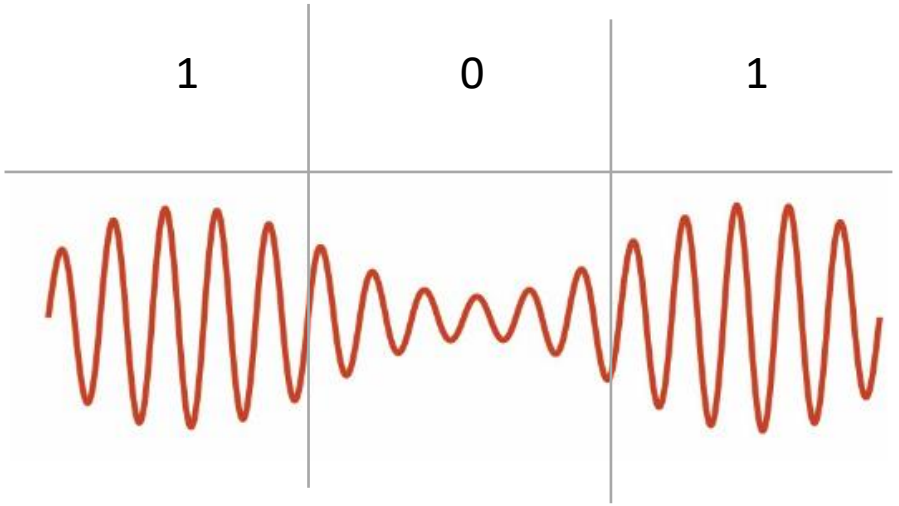




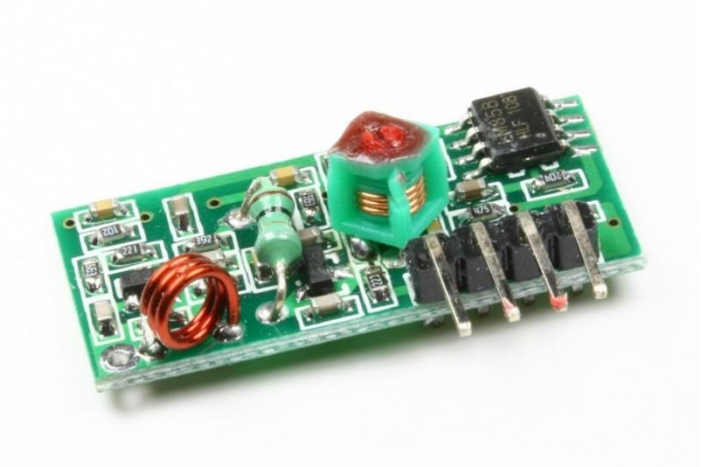
New component in the library - RC 433 MHz

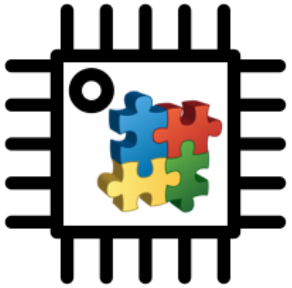
Transmitter





Receiver





ACP:

new way of Arduino prototyping

https://github.com/acptools

Thank you for your attention.

Similar solutions

Enhancing Arduino programming model

From **single execution cycle** to **preemptive multitasking**

ARTE

```
void loop() {

  <activity A>

  <activity B>

  <activity C>
}
```

```
void loop1(p1) {
  <activity A>
}
```

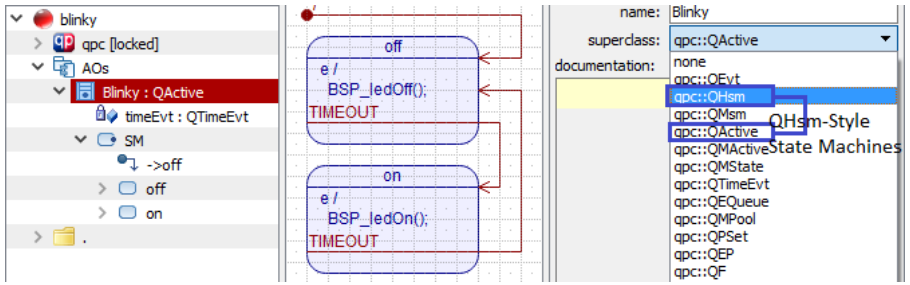
```
void loop2(p2) {
  <activity B>
}
```

```
void loop3(p3) {
  <activity C>
}
```

Arduino **EventManager**

```
void setup() {
  // Setup
  gEM.addListener( EventManager::kEventUser0, listener );
}

void loop() {
  // Handle any events that are in the queue
  gEM.processEvent();
}
```



QuantumLeaps
innovating embedded systems