
Špecifikácia softwarového diela

***Arduino simulátor s
grafickým používateľským
rozhraním***

Cieľom diela je poskytnúť programátorom mikropočítačov Arduino bezpečné prostredie, v ktorom môžu testovať, analyzovať a debugovať svoj kód s intuitívnou vizuálnou odozvou.

0.0.1

Adam Balko

16.2.2026

Obsah

1. Základné informácie 1

1.1 Popis a zameranie softwarového diela.....	1
1.2 Použité technológie.....	1
1.3 Odkazy (Referencie).....	1

2. Stručný popis softwarového díla 2

2.1 Základné časti diela a ciele riešenia.....	2
2.2 Hlavné funkcie.....	2
2.3 Motivačný príklad použitia.....	2
2.4 Prostredie aplikácie.....	3
2.5 Obmedzenia diela.....	3

3. Vonkajšie rozhranie 3

3.1 Užívateľské rozhranie, vstupy a výstupy.....	3
3.2 Komunikačné rozhranie.....	3

4. Detailný popis funkcionality 4

4.1 Spustenie a debugovanie Arduino kódu v simulátore.....	4
4.2 Ovládanie digitálneho elektronického obvodu v Gui pomocou simulovaného Arduina.....	4
4.3 Ovpplyňovanie vstupov Arduina pomocou interaktívnych komponent na plátne Gui.....	4
4.4 Definovanie vlastných elektronických komponent.....	4
4.5 Definovanie scény.....	5
4.6 Simulácia digitálneho obvodu.....	5
4.7 Logovanie obvodu a TCP pripojenia.....	5

5. Obrazovky 6

5.1 Hlavná obrazovka.....	6
---------------------------	---

6. Požiadavky k používaniu 6

6.1 Softwarové požiadavky.....	6
6.2 Požiadavky na zabezpečenie dát.....	7

7. Negatívne vymedzenia 7

8. Poznámky 8

Tabulka revizí

Jméno	Datum	Důvod změny	Verze
<Autor>	<Datum>	<Stručný popis změny>	<Verze>

1. Základné informácie

1.1 Popis a zameranie softwarového diela

Arduino je veľmi populárnym mikropočítačom medzi laikmi aj profesionálmi. Avšak testovať kód Arduina na samotnom zariadení môže byť dosť náročné; debugovacie možnosti sú limitované a časovo kritický kód je nemožné pozorovať voľným okom. Programátor má vtedy možnosť stiahnuť si emulátor tretej strany, aplikáciu, ktorá spustí program používateľa do strojového kódu AVR, procesora zabudovaného do Arduina. Niekedy je emulátor súčasťou väčšieho simulátora elektronických obvodov, inokedy ide o samostatný program.

Na rozdiel od emulátora AVR chceme vytvoriť simulátor samotného Arduina. Program používateľa nebude preložený do strojového kódu AVR, ale do kódu hostiteľskej platformy. Existujúcu Arduino knižnicu nahradíme vlastnou implementáciou, do ktorej môžeme vnoriť diagnostiku simulovaného zariadenia. Ďalším krokom je vytvoriť GUI aplikáciu, ktorá sa na simulátor naviaže. V tejto aplikácii si používateľ zloží jednoduchý digitálny obvod, na ktorom môže zmeny v Arduinu pozorovať, zatiaľ čo svoj program debuguje vo svojom obľúbenom vývojovom prostredí.

Ambíciou je pripraviť základ grafickej aplikácie, v ktorej môže užívateľ vytvárať komplikovanejšie obvody, analyzovať výstup zo simulátora Arduina, a ovládať simuláciu zastavením, spomalením apod. Cieľom je uľahčiť prácu najmä začiatčikom, ktorí sa na Arduinu učia programovať na veľmi nízkej úrovni.

1.2 Použité technológie

- .NET 9 + C# 13 (Frontend)
- C++20 (Backend)
- Arduino-CLI (balík na kompiláciu Arduino kódu z príkazového riadku)
- Avalonia UI (.NET multiplatform framework pre vývoj GUI)
- Boost.ASIO (C++ knižnica pre sieťové programovanie)
- Ďalšie NuGet balíky
 - PanAndZoom (Avalonia knižnica – intuitívne približovanie a posun plátna)
 - CommandLineParser (spracovanie argumentov príkazového riadku)
 - Svg.Skia (manipulácia s Svg obrázkami v C#)
 - YamlDotNet (deserializácia Yaml súborov do C# objektov)

1.3 Odkazy (Referencie)

- Programátorská dokumentácia Arduino
<https://docs.arduino.cc/programming/>
- Dokumentácia Avalonia UI
<https://docs.avaloniaui.net/docs/welcome>

- Dokumentácia Boost.ASIO
https://www.boost.org/doc/libs/latest/doc/html/boost_asio.html

2. Stručný popis softwarového díla

2.1 Základné časti diela a ciele riešenia

Dielo je rozdelené na 2 procesy: Core a Gui. Tie spolu komunikujú cez TCP kanál na spoločnom porte.

Core je backendom projektu, je písaný v C++ a kompiluje sa spoločne s používateľským kódom. Tento kód je spúšťaný v simulátore kontrolovanej slučky. Core obsahuje náhradnú implementáciu Arduino knižnice a vnútorný stav simulovaného Arduina. Taktiež zakladá TCP server, na ktorý sa môžu pripojiť iné procesy, ktoré budú stav Arduina čítať alebo doň zapisovať.

Gui je príkladom takéhoto procesu. Jeho hlavnou časťou je komponentový systém. V tomto systéme si užívateľ zadefinuje elektronické komponenty, ktoré bude Arduinom ovládať, a zapojí ich do obvodu. Jednotlivé komponenty budú reagovať na zmeny v obvode, vyvolané buď samotným Arduinom alebo inými komponentami, napr. tlačidlá s ktorými môže používateľ interagovať. Tieto zmeny sa potom premietnu na plátne v Avalonia UI. Základne komponenty ako LED diódy alebo tlačidlá budú v Gui už zahrnuté.

Súčasťou diela je taktiež pomocný preprocess skript, ktorý zakomponuje .ino súbory s Arduino kódom do Coru.

2.2 Hlavné funkcie

- Spustenie a debugovanie Arduino kódu v simulátore
- Ovládanie digitálneho elektronického obvodu v Gui pomocou simulovaného Arduina
- Ovplyvňovanie vstupov Arduina pomocou interaktívnych komponent na plátne Gui
- Vytváranie scény digitálneho obvodu a komponent interagujúcich s Arduinom

2.3 Motivačný príklad použitia

Študent MFF UK napíše riešenie domácej úlohy na predmet Počítačové systémy. Než svoj kód nahrá na svoje fyzické Arduino, spustí si ho v simulátore, a testuje ho v Gui. Zistí, že tlačidlá nefungujú tak ako by od svojho kódu očakával. Vo svojom IDE nastaví breakpoint v riadku kde predpokladá, že spravil chybu. Ďalej krokuje svoj kód v debugovacom režime, zatiaľ čo sleduje lokálne premenné svojho kódu, ako aj výstupy Arduina v Gui. Presunie sa pár krokov simulácie späť, a prepíše hodnotu na jednom pine, a skúša, či sa správanie tlačidla zmení. Týmto spôsobom nájde vo svojom programe chybu rýchlejšie, než keby to skúšal priamo na fyzickom Arduine.

2.4 Prostredie aplikácie

Projekt je testovaný a ladený primárne na Linuxe, s použitím IDE Code OSS. Avšak implementácia ani použité knižnice nie sú závislé na konkrétnom operačnom systéme ani vývojovom prostredí.

2.5 Obmedzenia diela

Simulátor je modelovaný podľa Arduino UNO. Modely s rovnakým pinovým a knižničným rozhraním môžu taktiež fungovať, avšak na implementačné rozdiely oproti Arduino UNO sa neberie ohľad.

S momentálnou implementáciou nie je možné definovať komponenty Gui reagujúce na impulzy rýchlejšie ako niekoľko milisekúnd. Túto limitáciu sa pokúsime odstrániť v neskorších verziách aplikácie. Analógové a sériové prvky Arduina taktiež nie sú momentálne podporované.

3. Vonkajšie rozhranie

3.1 Užívateľské rozhranie, vstupy a výstupy

Hlavným vstupom je priečink s .ino súbormi obsahujúcimi Arduino kód. Ten je nutné špecifikovať ako argument preprocess skriptu. Vedľajšími vstupmi sú priečink definícií jednotlivých komponent a definícia scény (teda simulovaného obvodu). Definícia scény je vo formáte .yaml a popisuje, ktoré komponenty sa v scéne nachádzajú, na akých pozíciách a ako sú ich piny pospájané. Definícia komponenty je zložená z .cs súboru s logikou, .yaml súboru so statickou konfiguráciou a niekoľkých .svg súborov, ktoré budú reprezentovať stav komponenty na plátne Gui.

Ďalej používateľ interaguje so simulátorom a obvodom pomocou grafického používateľského prostredia. Tu môže pozorovať zmeny vyvolané stlačením tlačidla, alebo iných komponent, ktoré implementujú handler udalostí myši.

Za výstupy sa dajú považovať tiež .log súbory s diagnostickými záznamami o behu programu.

3.2 Komunikačné rozhranie

Dielo sa skladá z 2 procesov, čiže je nutné zabezpečiť ich synchronizáciu pomocou medziprocesovej komunikácie. Zvolili sme postup s TCP komunikačným kanálom. Core založí TCP server pomocou knižnice Boost.ASIO a Gui sa k nemu pripojí ako TCP klient. Klient sa periodicky pýta servera na momentálny stav Arduino simulátora zaslaním správy s príznakom pre čítanie. Ten odošle späť správu so stavom jednotlivých pinov. Keď zmena obvodu v Gui ovplyvní signál vstupných pinov Arduina, klient odošle správu s príznakom pre zápis. Vtedy sa stav Arduina v Gui prepíše do prechodného stavu Arduina v simulátore. Po prepísaní všetkých pinov je tento prechodný stav atomicky zamenený za skutočný stav simulátora, na ktorom prebieha výpočet.

4. Detailný popis funkcionality

4.1 Spustenie a debugovanie Arduino kódu v simulátore

Používateľ uvedie cestu k svojmu Arduino kódu ako argument skriptu preprocess.sh. Ten vytvorí nový súbor sketch.cpp, ktorý bude obsahovať referenciu na Arduino.h knižnicu, a odkazy na pôvodný súbor pre účely debugovania. Tento súbor bude uložený v Core, v projekte executable spolu s main.cpp.

Používateľ následne skompiluje a spustí Core. V tomto momente je už kód simulovaný, raz sa spustí funkcia setup() v slučke sa spúšťa funkcia loop(). Kód je už možné debugovať bežnými C++ debugermi pridaním breakpointov do nepredspracovaného .ino súboru.

4.2 Ovládanie digitálneho elektronického obvodu v Gui pomocou simulovaného Arduina

Spustením Gui sa simulované Arduino napojí na digitálny obvod, ktorý používateľ zdefiniuje vo svojej scéne. Na plátne Gui tak môže pozorovať efekty Arduina na komponentoch ako sú napríklad LED diódy, ktoré zmenia svoj SVG sprite na žiarivejší, keď je na ich katódu privedená digitálna 1.

4.3 Ovpływňovanie vstupov Arduina pomocou interaktívnych komponent na plátne Gui

V Gui môže používateľ interagovať so simulovaným Arduinom pomocou interaktívnych komponent, ako je napríklad tlačidlo, ktoré pri stlačení myši na jeho sprite, prepošle digitálnu 1 z jedného pinu na druhý. Takto môže používateľ meniť signál na vstupných pinoch Arduina a ovplyvniť tak jeho beh programu.

4.4 Definovanie vlastných elektronických komponent

Používateľ môže vytvárať vlastné komponenty, s vlastnou logikou, ktoré môže potom dosadiť do scény. Každá komponenta vyžaduje niekoľko súborov umiestnených do jedného priečinka. Tento priečinok potom vloží do priečinka so všetkými ostatnými komponentami. Priečinok so všetkými komponentmi je poskytnutý Gui ako argument príkazového riadka.

Vyžadované súbory sú:

- Konfigurácia - .yaml súbor so základnými konštantami identifikujúcimi komponentu a jej časti. Obsahuje napríklad názov komponenty, jej popis, názvy jej pinov apod.
- Sprity – 1 alebo viac .svg obrázkov, ktoré budú vykreslené na plátno obvodu. Pre komponenty s viacerými vnútornými stavmi je vhodné pridať viac obrázkov reprezentujúcich rôzne stavy, napr. zapnutá a vypnutá LED.
- Logika – C# súbor s triedou dediacou z Component triedy. Component trieda poskytuje rozhranie pre ovládanie pinov a sprítov pomocou ich názvov. Ďalej poskytuje virtuálne metódy na spracovanie bežných eventov, ako napríklad zmena signálu na pine, kliknutie na sprite.

Súčasťou projektu sú už preddefinované komponenty zdroja napätia, LED diódy, tlačidlá a Arduina. Ďalšími plánovanými komponentami sú logické hradlá, posuvné registre, 7-segmentový displej apod.

4.5 Definovanie scény

Digitálny obvod je súčasťou scény definovanej ako .yaml súbor. Tento súbor je poskytnutý Gui ako argument na príkazovom riadku. Tento súbor tvorí niekoľko sekcií.

- Komponenty – Jednotlivé inštancie zadaných komponentov. Musia mať unikátny názov a odkazovať sa na existujúci typ komponenty. Ďalej môžu špecifikovať umiestnenie na plátne alebo iné unikátne vlastnosti.
- Uzly – Definuje, ktoré piny sú pripojené k jednému elektrickému uzlu. Inštancia komponentu, ku ktorej pin patrí musí byť špecifikovaná menom zo sekcie komponent.

4.6 Simulácia digitálneho obvodu

Aj bez pripojenia na simulátor Arduina, dokáže Gui simulovať základné digitálne elektronické obvody. Zmena na pine jednej komponenty sa cez uzly propaguje na piny iných komponent. Pin môže byť vzбудený. Pokiaľ sa na uzle nachádza aspoň jeden vzbudený pin, digitálna hodnota uzlu je nastavená na 1. Táto zmena je oznámená všetkým ostatným pinom uzlu. Na komponente pinu sa vyvolá virtuálna funkcia Component triedy. Pin môže byť nastavený na vstupný (nemôže budiť), výstupný (nereaguje na zmeny uzlu) alebo obecný mód.

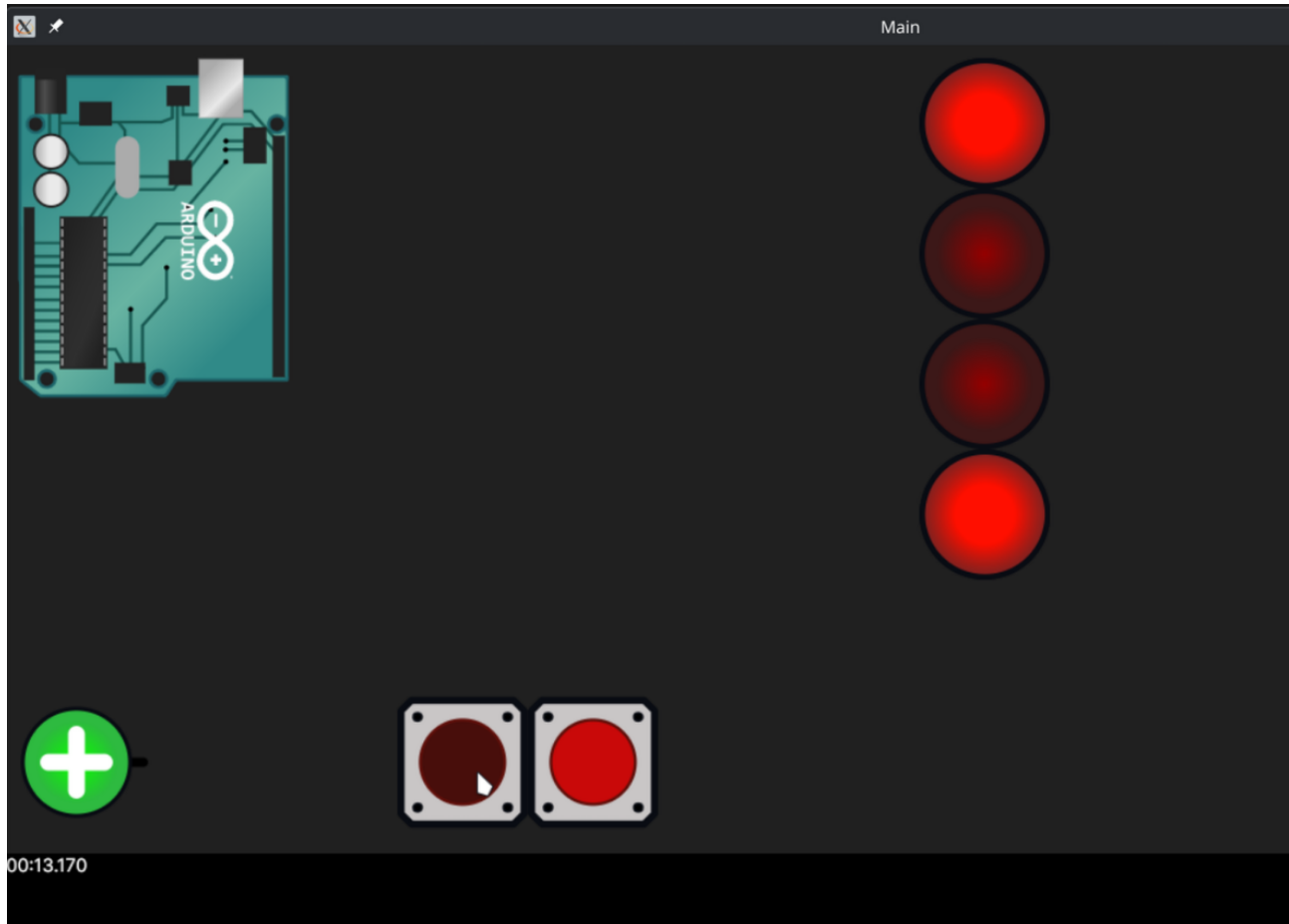
4.7 Logovanie obvodu a TCP pripojenia

Core aj Gui zaznamenávajú dôležité udalosti do .log súborov. V prípade Gui môžu byť špecifikované ako argumenty príkazového riadku. Logy Gui sú označené časovou značkou a závažnosťou udalosti (Debug, Info, Warning, Error). S východzím možnosťami sú v pracovnom priečinku založené tieto súbory:

- core_tcp.log – udalosti TCP servera, prijaté a odoslané správy
- frontend_tcp.log – udalosti TCP klienta, prijaté a odoslané správy
- frontend.log – udalosti komponentového systému, jeho inicializácia, definované komponenty a uzly, zmeny na pinoch a uzloch+

5. Obrazovky

5.1 Hlavná obrazovka



Hlavná obrazovka Gui s plátnom digitálneho obvodu v hornej časti a globálneho časovača v časti spodnej. Na plátne je príklad scény so 4 LED diódmi napojenými na výstupy Arduina a 2 tlačidlami pripojenými na vstupy Arduina.

6. Požiadavky k používaniu

6.1 Softwarové požiadavky

- Minimálne systémové požiadavky Avalonia UI
<https://docs.avaloniaui.net/docs/overview/supported-platforms>

- Knižnica Boost.ASIO
<https://www.boost.org/releases/latest/>
- .NET 9 Runtime
<https://dotnet.microsoft.com/en-us/download/dotnet/9.0>
- Balík arduino-cli
<https://docs.arduino.cc/arduino-cli/installation/>

6.2 Požiadavky na zabezpečenie dát

Používanie diela je tak bezpečné ako vstupný Arduino kód, alebo C# implementácie jednotlivých používateľom definovaných komponent Gui. Používanie diela nevyžaduje prístup k citlivým informáciám.

7. Negatívne vymedzenia

Toto dielo nie je:

- Emulátorom – Nikde v priebehu používania simulátora nedochádza k prekladu do strojového kódu AVR procesora, ktorý fyzické Arduino používajú. Kód závislý na architektúre konkrétneho AVR nebude fungovať.
- Simulátorom obecného elektronického obvodu – Fyzikálne vlastnosti obvodu, ako napríklad prúdy a odpory sú mimo rozsah tohoto diela. Aby sme udržali komplexitu diela na rozumnej úrovni, limitujeme sa iba na digitálne obvody s jednoduchými prototypmi komponent.

Dodatok A: Vymedzenie pojmov

- *hostiteľská platforma* - operačný systém a architektúra procesora (napr. x86_64 Windows/Linux), na ktorom beží simulátor, nie cieľový mikrokontrolér.
- *TCP* – “Transmission Control Protocol”, protokol určený na spoľahlivý prenos dát po sieti. Pre naše účely ho používame pre prenos dát medzi dvoma procesmi.
- *AVR* - architektúra mikroprocesorov od spoločnosti Atmel, ktoré sa nachádzajú na fyzických doskách Arduino (napr. Atmega328P).
- *NuGet* – správca balíkov pre .NET.
- *serializácia* – prevod dátovej štruktúry v pamäti do konkrétneho textového formátu. Opačnou operáciou je *deserializácia*.
- *SVG* – “Scalable Vector Graphics”, dátový formát vektorových obrázkov, založený na XML.
- *YAML* – “YAML ain’t markup language”, serializačný jazyk, zameraný na ľahké čítanie a úpravu človekom.

- *.ino* – prípona súborov s používateľským kódom Arduina. Samotný kód je písaný v jazyku takmer totožnom s C++.
- *IDE* – “Integrated Development Environment”, aplikácia určená na pohodlné písanie a ladenie kódu (napr. VS Code, IntelliJ IDEA).
- *breakpoint* – Nástroj na pozastavenie chodu programu na konkrétnom riadku kódu.
- *pin* – fyzické vodivé zakončenie elektronickej súčiastky, pomocou ktorej sa pripája do elektrického obvodu.
- *vzbudený pin* – pin, ktorý aktívne vynucuje stav digitálnej jednotky na pripojenom vodiči.
- *sprite* – 2D grafický objekt.

8. Poznámky

Táto špecifikácia je viac než inšpirovaná týmito šablónami:

- Software Requirements Specification by Karl E. Wiegers
- SAFE™ Development System Requirements