|  |  |
| --- | --- |
| Univerzita Pavla Jozefa Šafárika v Košiciach  Prírodovedecká fakultaNázov fakultyNázov vysokej školy | |
| Komponentové a udalosťami riadené programovanie zariadení na platforme Arduino | |
|  | |
|  | |
| 2018 | Bc. Patrik Pekarčík |

|  |  |
| --- | --- |
| Univerzita Pavla Jozefa Šafárika v Košiciach  Prírodovedecká fakultaPrírodovedecká fakulta | |
| Komponentové a udalosťami riadené programovanie zariadení na platforme ArduinoKomponentové a udalosťami riadené programovanie zariadení na platforme Arduino | |
| **Diplomová práca**Diplomová práca | |
| Študijný program: | InformatikaInformatika |
| Pracovisko (katedra/ústav): | Ústav InformatikyÚstav Informatiky |
| Vedúci diplomovej práce: | RNDr. František Galčík, PhD.RNDr. František Galčík, PhD. |
| Konzultant diplomovej práce: (nepovinný) | Titul Meno Priezvisko, HodnosťTitul Meno Priezvisko, Hodnosť |
|  |  |
| KošiceKošice 20182018 | Bc.Bc. PatrikPatrik PekarčíkPekarčík |

**Zadanie záverečnej práce**

Zadanie záverečnej práce (ďalej len „zadanie“) je dokument, ktorým vysoká škola stanoví študentovi študijné povinnosti v súvislosti s vypracovaním záverečnej práce. Zadanie spravidla obsahuje: typ záverečnej práce, názov záverečnej práce, meno, priezvisko a tituly študenta, meno, priezvisko a tituly školiteľa, v prípade externého školiteľa meno, priezvisko a tituly konzultanta, školiace pracovisko, meno, priezvisko a tituly vedúceho pracoviska, anotáciu záverečnej práce, jazyk, v ktorom sa práca vypracuje, dátum schválenia zadania.

**Poďakovanie (nepovinné)**

Na tomto mieste môže byť vyjadrenie poďakovania napr. vedúcemu práce resp. konzultantom za pripomienky a odbornú pomoc pri vypracovaní práce. Nie je zvykom ďakovať za rutinnú kontrolu, menšiu spoluprácu alebo všeobecné rady. Vyjadrenie poďakovania v prípade využitia inej práce sa uskutočňuje formou citácie na konci hlavného textu práce a odkazy na citáciu sa musia uviesť aj na zodpovedajúcich miestach v texte.

|  |
| --- |
| Abstrakt v štátnom jazyku |
| Abstrakt obsahuje informáciu o cieľoch práce, jej stručnom obsahu a v závere abstraktu sa charakterizuje splnenie cieľa, výsledky a význam celej práce. Súčasťou abstraktu je  3 - 5 kľúčových slov. Abstrakt sa píše súvisle ako jeden odsek a jeho rozsah je spravidla 100 až 500 slov. |

|  |
| --- |
| Abstrakt v cudzom jazyku |
| Text abstraktu v svetovom jazyku je potrebný pre integráciu do medzinárodných informačných systémov (napr. The Network Digital Library of Theses and Dissertations). Ak nie je možné jazykovú verziu umiestniť na jednej strane so slovenským abstraktom, je potrebné umiestniť ju na samostatnú stranu (cudzojazyčný abstrakt nemožno deliť a uvádzať na dvoch stranách). |

Obsah

[Obsah 5](#_Toc508108721)

[Zoznam ilustrácií 7](#_Toc508108722)

[Zoznam tabuliek 8](#_Toc508108723)

[Zoznam skratiek, značiek a termínov 9](#_Toc508108724)

[Úvod 10](#_Toc508108725)

[1 Internet vecí 11](#_Toc508108726)

[1.1 Hardvér 12](#_Toc508108727)

[1.2 Softvér 12](#_Toc508108728)

[1.3 Mikrokontrolér v porovnaní so single-board počítačom 13](#_Toc508108729)

[2 Platforma Arduino 14](#_Toc508108730)

[2.1 Parametre mikrokontroléra 15](#_Toc508108731)

[2.2 Programovanie pre mikrokontrolér 16](#_Toc508108732)

[2.3 Existujúce riešenia pre platformu Arduino 17](#_Toc508108733)

[2.3.1 Arduino EventManager 17](#_Toc508108734)

[2.3.2 Quantum Leaps Modeling Tool 18](#_Toc508108735)

[2.3.3 ARTe (Arduino Real-Time extension) 18](#_Toc508108736)

[2.3.4 Cayenne 18](#_Toc508108737)

[3 Komponentové a udalosťami orientované programovanie 19](#_Toc508108738)

[3.1 Spracovanie udalosti 20](#_Toc508108739)

[4 Integrované vývojové prostredie 21](#_Toc508108740)

[4.1 Rapid application development 21](#_Toc508108741)

[5 Architektúra komponentového a udalosťami orientovaného riešenia ACProg 22](#_Toc508108742)

[5.1 XML Konfiguračný súbor 23](#_Toc508108743)

[5.2 Generátor knižnice pre Arduino 24](#_Toc508108744)

[5.3 Typy komponentov (moduly) 24](#_Toc508108745)

[6 IDE pre projekt ACProg 25](#_Toc508108746)

[6.1 Používateľské požiadavky 25](#_Toc508108747)

[6.1.1 Grafický návrh IDE 25](#_Toc508108748)

[6.2 Technologický návrh 25](#_Toc508108749)

[6.2.1 Framework Java Swing 25](#_Toc508108750)

[6.2.2 Docking framework 25](#_Toc508108751)

[6.3 Architektúra IDE 25](#_Toc508108752)

[6.4 Implementácia IDE 25](#_Toc508108753)

[6.4.1 Use case: Manažment komponentov 25](#_Toc508108754)

[6.4.2 Use case: Ponuka dostupných komponentov 26](#_Toc508108755)

[6.4.3 Use case: Kompilácia a spustenie projektu 26](#_Toc508108756)

[6.4.4 Use case: Syntaktická analýza kódu 26](#_Toc508108757)

[6.4.5 Use case: Volitelny UX pre programatora 26](#_Toc508108758)

[7 Vzorové komponenty pre ACProg 27](#_Toc508108759)

[7.1 Časovač 27](#_Toc508108760)

[7.2 Digitálny výstup 27](#_Toc508108761)

[7.3 Digitálny vstup 27](#_Toc508108762)

[7.4 Analógový vstup s threshold alert 27](#_Toc508108763)

[7.5 Komunikácia pomocou MQTT 27](#_Toc508108764)

[7.6 Rádiová komunikácia pomocou 433MHz 27](#_Toc508108765)

[Záver 28](#_Toc508108766)

[Zoznam použitej literatúry 29](#_Toc508108767)

[Prílohy 30](#_Toc508108768)

Zoznam ilustrácií

Obr. 1 Schéma reprezentujúca IoT 12

Obr. 2 Doska Arduino (vľavo) a rozširujúca doska shield (vpravo) 15

Obr. 3 Arduino IDE, v červenom kruhu je vyznačené tlačilo spustenia na doske. 16

Obr. 4 Rozdelenie zdrojového kódu Arduino príkladového programu. 18

Obr. 5 Znázornenie inštancie komponentu typu tlačidlo, s automatom pre prechody medzi jednotlivými stavmi. 20

Obr. 6 Práca plánovača úloh s registráciou a vyvolaním udalosti. 21

Obr. 7 Priebeh spracovania projektu, riešením ACProg. 23

Obr. 8 Priebeh spracovania projektu, riešením ACProg. 25

Zoznam tabuliek

Tab. 1 Porovnanie parametrov pre rôzne modely dosiek Arduino. 16

Zoznam skratiek, značiek a termínov

IDE **I**ntegrated **D**evelopment **E**nvironment – Integrované vývojové prostredie

XML E**x**tended **M**arkup **L**anguage

IoT **I**nternet **o**f **T**hings

Cloud Výkonné počítačové prostriedky umiestnené v sieti internet

CPU Central Processing Unit – Centrálna výpočtová jednotka

ROM Read Only Memory

RAM Random Access Memory

EEPROM Electrically Erasable Programmable Read-Only Memory

Hz Hertz, jednotka frekencie používaná na meranie rýchlosti CPU

I2C Zbernica používaná pre nízkoúrovňové zariadenia

USB Universal Serial Bus

SD Secure Digital card

Pin

RAD Rapid application evelopment

ACProg Názov v práci použivaného komponentového riešenia

.ino Prípona zdrojových súborov, programovacieho jazyka pre Arduino

Úvod

Hlavnú textovú časť záverečnej práce tvorí: úvod, jadro, záver, resumé (povinné iba v prípade, ak je práca vypracovaná v inom ako štátnom jazyku), zoznam použitej literatúry.

V úvode autor stručne a výstižne charakterizuje stav poznania alebo praxe v oblasti, ktorá je predmetom záverečnej práce a oboznamuje čitateľa s významom, cieľmi a zámermi práce. Autor v úvode zdôrazňuje, prečo je práca dôležitá a prečo sa rozhodol spracovať danú tému.

# Internet vecí

Pojem internetu vecí (IoT) pokrýva viac ako len koncept alebo technológiu. Je to nový prístup, ktorý ovplyvňuje vývoj aktuálnych technológií, aplikácií a vízií. Kvôli rozsiahlosti ešte nemáme presnú definíciu pre IoT ako také. Viaceré vedecké články, sa zhodujú v tom, že kľúčovým prvkom IoT je prepojenie fyzického a digitálneho sveta, pomocou senzorov a aktuátorov. Okrem toho je pre IoT zariadenia dôležité vedieť komunikovať s inými zariadeniami, počítačmi či ľuďmi pomocou počítačovej siete [1], pre následné vykonávanie akcií. Nasledujúca schéma reprezentuje IoT v komunikačnej postupnosti.

obrazok

Obr.  Schéma reprezentujúca IoT

Popis schémy začneme senzormi a budeme pokračovať v smere toku dát. Senzory sú harvérové zariadenia, ktoré zberajú informácie z fyzického sveta a menia ich na digitálny signál. Signál smeruje do mikrokontroléra, ten vykonáva lokálne spracovanie. Lokálne spracovanie spočíva v prevode signálu na digitálne dáta. Okrem tohto prevodu môže vykonávať akcie a ihneď ovplyvňovať fyzický svet. Po lokálnom spracovaní sa dáta uložia na lokálne úložisko pre ich budúce analýzy. Keďže hovoríme o internete vecí, tak je na miesto rozmýšľať o online zdieľaní dát s ostatnými zariadeniami pomocou počítačovej siete. IoT server je zariadenie na sieti, ktoré prijíma dáta zo všetkých senzorov a vykonáva cloud spracovanie. Pozícia cloud spracovania je veľmi podobná lokálnemu spracovaniu. Hlavným rozdielom je množstvo spracovávaných dát. Kým pre lokálne spracovanie máme nízke výkonnostné prostriedky pri cloud spracovaní sú k dispozícii gigabajty pamäte s výkonnými procesormi, ktoré nám umožňujú vykonávať náročné spracovania prijatých dát. Všetky spracované dáta sa následne ukladajú na cloud úložisku. Schéma predstavuje aký potenciál nám poskytuje IoT. V mnohých IoT projektoch sa však stáva, že nevyužijú všetky časti v schéme.

## Hardvér

Základným stavebným prvkom lokálneho spracovania v IoT je mikrokontrolér, ktorý riadi prácu zariadenia. Na mikrokontrolér sú pripojené senzory, aktuátory a obvody podporujúce komunikáciu s ďalšími zariadeniami. Mikrokontrolér je elektronický čip obsahujúci obvody pre zavedenie a beh softvéru. Menovite to sú RAM, ROM, CPU. Avšak parametre týchto obvodov sú relatívne nízke oproti bežne dostupným počítačom. Napríklad mikrokontrolér, ktorý používajú aj dosky Arduino, ATMEGA328 disponuje 32kB ROM a 2kB RAM so 16-bitovým 20MHz CPU v cene približne dvoch dolárov.

**Senzory** sú elektronické súčiastky pomocou, ktorých mikrokontrolér získava informácie o fyzickom svete v digitálnej informácii, napr. odmeranie elektrického odporu. Komunikácia senzora s mikrokontrolérom spočíva v čítaní elektrického napätia, ktoré mikrokontrolér spracuje do digitálnej podoby a prevedie do meraných jednotiek. Napríklad pri meraní elektrického odporu by sme prijaté elektrické napätie prepočítali na digitálnu verziu v ohmoch.

**Aktuátory** sú elektronické súčiastky ovplyvňujúce fyzický svet. Zaraďujeme tu motory, svetlá, reproduktory, a pod. Tieto súčiastky sú riadené informáciami prijatými od mikrokontrolérov.

## Softvér

Na riadenie mikrokontroléra je potrebný obslužný softvér napálený v jeho pamäti ROM. Tento softvér je zavedený pri privedení elektrického prúdu do mikrokontroléra. Pomocou neho mikrokontrolér číta hodnoty vstupných pinov (zo senzorov) a riadi výstupné piny (k autátorom). Najčastejším programovacím jazykom pre mikrokontroléry je upravená verzia jazyka C. Netreba zabudnúť, že mikrokontroléry poskytujú len rádovo 2kB operačnej pamäte pre beh vytvorené programy.

Okrem samotných programov pre mikrokontrolér je dôležitý aj podporný softvér. Podporný softvér slúži programátorom na uľahčenie vývoja. Základom je kompilátor, pomocou, ktorého vieme previesť program. Integrované vývojové prostredia tiež radíme medzi podporný softvér. Jeho hlavnou úlohou je odbremeniť programátorov od opakovaných, krokov a poskytnúť jednoduchú konfiguráciu bez väčšej potreby znalosti celého systému. Integrované vývojové prostredia taktiež veľmi prispievajú k rýchlejšiemu prototypovaniu nových nápadov.

## Mikrokontrolér v porovnaní so single-board počítačom

Hlavnou návrhovou črtou mikrokontroléra je komunikácia so senzormi, aktuátormi a pod. Pre takú komunikáciu je dôležité presné časovanie prenosu signálov a tú nám mikrokontroléry priamo poskytujú. Pri single-board počítačoch je hlavnou črtou vyšší výpočtový výkon za nízku cenu. Výkon na týchto počítačoch sa pohybuje okolo 1GB RAM s ARM 1,2Ghz procesorom. Tieto parametre nám už dovoľujú spustiť na procesore odľahčený operačný systém, avšak nevýhodou operačných systémov je, že prenechávame časovanie a plánovanie na nich a preto je takmer nemožná priama stabilná komunikácia so senzormi a aktuátormi. Single-board počítače často poskytujú rôzne mikrokontroléry implementujúce komunikačné protokoly používané v elektronike (napr. I2C). Pomocou nich vieme pomocou single-board počítačov komunikovať so senzormi. Avšak počet takých senzorov na trhu je veľmi nízky.

// TODO: napisat nieco malo o fog comutingu (cloud computing v malom)?? – možno to nie je potrebné

# Platforma Arduino

Projekt Arduino sa začal v roku 2003 ako program pre študentov v Interaction Design Institute Ivrea v Ivreu v Taliansku s cieľom poskytnúť novým a profesionálnym používateľom lacný a jednoduchý spôsob vytvárania zariadení, ktoré interagujú s prostredím pomocou senzorov a aktuátorov. Bežné príklady takýchto zariadení určené pre začiatočníkov zahŕňajú jednoduché roboty, termostaty a detektory pohybu.

Arduino je hardvér a softvér s otvoreným zdrojovým kódom, licencovaný licenciami GNPL Lesser General Public License (LGPL) a GNU General Public License (GPL). Licencie umožňujú výrobu dosiek Arduino a distribúciu softvéru každému. Dosky Arduino sú špecifické vyvedenými pinmi na ich okraji. Toto vyvedenie je kľúčové pre jednoduché prototypovanie embeded zariadení. Vďaka štandardizovanému tvaru a vyvedeniu pinov dosky Arduino je možné vytvárať rozširujúce dosky tzv. Arduino shield. Arduino shield poskytuje rovnaký tvar, rozloženie pinov a dodáva rôzne funkcionality, či už sú to senzory, aktuátory alebo kompletné riešenie pre embeded systém. Dosky Arduino majú vstavané sériové komunikačné rozhrania, vrátane univerzálnej sériovej zbernice (USB).

obrazok

Obr.  Doska Arduino (vľavo) a rozširujúca doska shield (vpravo)

Softvérová časť projektu Arduino spočíva v integrovanom vývojovom prostredí (IDE) a programovacom jazyku postavenom na c++. Integrované vývojové prostredie bolo postavené na projekte Language Processing. Keďže väčšina doskiek Arduino má vstavané rozhranie USB tak kompikácia s napálením projektu z IDE do zariadenia je na jedno kliknutie.

obrazok

Obr.  Arduino IDE, v červenom kruhu je vyznačené tlačilo spustenia na doske.

## Parametre mikrokontroléra

Základným prvkom dosky Arduino je mikrokontrolér. Pri originálnych doskách Arduino sa používajú mikrokontroléry Atmel ATmega. Na čipe mikrokontroléra sa nachádzajú obvody procesora (CPU), operačnej pamäte (RAM), programovej pamäte (ROM), trvalej pamäte (EEPROM). Pre komunikáciu s ostatnými zariadeniami sú z mikrokontroléra vyvedené vstupno výstupné piny. Výkonnostné hodnoty týchto parametrov sú pomerne nízke a preto je potrebné na ne brať ohľad pri návrhu softvéru. Tabuľka č. 1 zobrazuje konkrétne parametre niekoľkých modelov dosiek Arduino pre predstavu nízkych parametrov. Pre ilustráciu výkonnostného rozdielu medzi mikrokontrolérom a single-board počítačom, sme v poslednom riadku tabuľky vypísali parametre single-board počítača Raspberry PI. Pri single-board počítači si môžeme všimnúť absenciu analógových vstupno výstupných pinov.

Tab.  Porovnanie parametrov pre rôzne modely dosiek Arduino.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model Arduino** | **CPU Model** | **ROM (kB)** | **RAM (kB)** | **EEPROM (kB)** | **CPU frekvencia (MHz)** | **Analógové / digitálne piny** |
| UNO | ATmega323P | 32 | 2 | 1 | 16 | 6/14 |
| Mega 2560 | ATmega2560 | 256 | 8 | 4 | 16 | 16/54 |
| Nano | ATmega168 | 16 | 1 | 0,512 | 16 | 8/14 |
| 101 | Intel Curie | 196 | 24 | - | 32 | 6/14 |
| Gemma | ATtiny85 | 8 | 0,5 | 0,5 | 8 | 1/3 |
| Raspberry PI (singe-board) | ARMv8 64bit quad-core | SD slot | 512 MB | SD slot | 1200 | 0/40 |

## Programovanie pre mikrokontrolér

Programovacím jazykom pre mikrokontroléry je najčastejšie istá obdoba vyššieho programovacieho jazyka C++. Tvorcovia mikrokontrolérov vydávajú špecifikácie spolu s kompilátorom tohto programovacieho jazyka. Kompilátory prevádzajú tento jazyk na assembler pracujúci s konkrétnymi registrami a inštrukciami, ktoré sú hardvérovo implementované na použitom mikrokontroléri. Nasledujúci obrázok na jednoduchom programe ukazuje syntax a usporiadanie zdrojového kódu.

obrazok

Obr.  Rozdelenie zdrojového kódu Arduino príkladového programu.

V úvodnej sekcii je vypísaný zoznam knižníc a rozširujúcich súborov, ktoré program bude používať. V ďalšej sekcii sú definované premenné, konštanty ovplyvňujúce beh programu. Taktiež v tejto sekcii môžeme vytvoriť pomocné metódy pre prehľadnejší zdrojový kód. Nasleduje sekcia zavedenia, v ktorej nastavujeme zariadenie na mód v akom bude pracovať. Piny na mikrokontroléry sú vstupno-výstupné avšak pre konkrétny program ich s pravidla využívame iba jedným z módov (vstupný pin, výstupný pin). Pri zavedení môžeme nastaviť aj časovače zariadenia, Arduino UNO disponuje tromi časovačmi. Časovače v nastavenom intervale spustia zadanú sadu inštrukcií. Poslednou sekciou je sekcia behu programu (ďalej loop sekcia). Loop sekcia je cyklicky spúšťaná a predstavuje samotrné správanie mikrokontroléra. V tejto sekcii vykonáva mikrokontrolér všetky čítania senzorov a základné lokálne spracovanie.

## Existujúce riešenia pre platformu Arduino

Stále rastúca komunita sa venuje zariadeniam Arduino. S tým je spojený aj vývoj rôznych frameworkov pre túto platformu, ktorých cieľom je zjednodušiť vývoj programátorom. Riešenia aké sme našli vyhľadávaním na fórach Arduino komunity, rozdeľujeme do kategórií online a offline riešení.

**Online** sú riešenia, ktoré do zariadenia nainštalujú zavádzač. Úlohou zavádzača je preposielať všetky údaje zo senzorov na sieť, kde bude vykonané cloud spracovanie. Vo všetkých analyzovaných online riešeniach bola úplne vynechaná časť lokálneho spracovania a uloženie dát. **Offline** sú riešenia bežiace priamo na Arduino zariadení. Tieto riešenia využívajú najmä lokálne spracovanie s lokálnym uložením.

V nasledujúcich sekciách priblížime vybrané existujúce riešenia, ktoré sa svojou povanou najviac približujú nami navrhnutému riešeniu pre programovanie arduino zariadení.

### Arduino EventManager

Knižnica Arduino EventManager[12] patrí do skupiny offline riešení. Pri využití tejto knižnice sa periodicky vykonávaná funkcia loop() nevytvára. Namiesto nej v inicializačnej funkcii setup() zaregistrujeme obslužné funkcie (angl. callback) a programujeme spracovanie udalostí. Pri registrovaní funkcie určujeme na zmenu akého pinu má byť udalosť vyvolaná.

### Quantum Leaps Modeling Tool

Modelovací nástroj Quantum Leaps Modeling Tool zaradzujeme do skupiny offline riešení. Aplikácie v tomto nástroji modelujeme pomocou stavového automatu. Konečno stavový automat je matematický model výpočtu. Je to abstraktný stroj, ktorý môže byť v danom čase presne v jednom stave z konečného počtu stavov. Konečno stavový automat sa môže zmeniť z jedného stavu na druhý v reakcii na niektoré externé vstupy.

### ARTe (Arduino Real-Time extension)

Rozšírenie Arduino kompilátora ARTe zaradzujeme do skupiny offline riešení. Toto rozšírenie umožňuje vytvorenie viacerých loop sekcií, ktoré sa budú opakovať pseudoparalelne. Vývojári ARTe dosiahli tento pseudoparalelizmus prepísaním implementácie čakacích delay[[1]](#footnote-1) metód z aktívneho čakania na pasívne. Program si zapamätá kde bol delay vyvolaný a spustí ďalšiu časť programu. Ďalšou časťou programu rozumieme ďalšiu loop sekciu od začiatku, resp. od posledného volania delay funkcie ak presiahol požadovaný čas čakania.

### Cayenne

Aplikácia Cayenne[14] zaradzujeme do kategórie online riešení. Na dosku arduino je nahraná špeciálna aplikácia. Táto aplikácia automatizovane odosiela na servery Cayenne svoj aktuálny stav vstupných pinov. Doska taktiež čaká na inštrukcie zo serverov Cayenne pre nastavenie výstupných pinov. Vo webovom nástroji Cayenne vidíme reálny stav vstupných pinov, dokážeme nastavovať rôzne akcie pre zopnutie výstupných pinov. Príkladom akcie môže byť nasledovné: Ak stúpne teplota (teplotný senzor je na pine 5) nad 30C, tak zapni klimatizáciu (zapni 5V na pine 4).

# Komponentové a udalosťami orientované programovanie

S komponentovým a udalosťami orientovaným programovaním sa najmä stretávame pri používateľských aplikácií na počítačoch, príp. mobilných zariadeniach. Komponenty v týchto aplikáciách predstavujú jednotlivé grafické prvky zobrazené na obrazovke. Inštancie komponentov sú rozlišované svojím typom a stavom.

**Typom komponentu** je popísané jeho základné správanie a stavy aké môže komponent nadobúdať. Stav inštancie popisuje aktuálne hodnoty komponentu. Dobrým príkladom komponentu je tlačidlo. Tlačidlo má text, ktorý je na ňom zobrazený a môže nadobúdať nasledujúce stavy: nestlačené, stlačené, zablokované. Z takto popísaného typu komponentu, vytvoríme jeho inštanciu zadaním konkrétneho počiatočného stavu (nestlačené) a textu (stlač ma).

Stavy komponentov sa môžu v čase meniť a práve zmeny stavu komponentu vyvolávajú udalosti. **Udalosť** je metóda spustená po vykonanej zmene stavu komponentu. S pravidla typy komponentov majú definované mená dôležitých udalostí, ktoré môžu nastať. Pre vyvolanie udalosti, komponent sleduje prechody medzi svojimi stavmi. Nasledujúci obrázok znázorňuje formou automatu prechody medzi jednotlivými stavmi. Na šípkach je pomenovanie mena udalosti, ktorá nastáva pri prechode medzi vyznačenými stavmi. Hrubým kruhom je vyznačený aktuálny stav tlačidla.

obrazok

Obr.  Znázornenie inštancie komponentu typu tlačidlo, s automatom pre prechody medzi jednotlivými stavmi.

## Spracovanie udalosti

Aby program mohol spracovávať udalosti, potrebuje na to niekoľko samostatných modulov [2]. Prvým je samotný komponent, ktorý udalosť vyvolá pri zmene jeho stavu. Druhým je metóda alebo objekt, ktorý čaká na vznik udalosti a následne ju spracuje. Posledným a zároveň najdôležitejším modulom, ktorý prvé dva prepája, je plánovač udalostí. Plánovač udalostí je program, ktorý riadi beh programu, prijímaním a  sekvenčným vykonávaním prijatých udalostí. V nasledujúcom obrázku je nákres práce plánovača úloh v čase.

obrazok

Obr.  Práca plánovača úloh s registráciou a vyvolaním udalosti.

Na začiatku objekt SpracujUdalost vykoná registráciu na udalosť s názvom stlacenieTlacidla (komponentu Tlacidlo). Registráciu vykoná na plánovači úloh. Ten si do svojej internej pamäte túto registráciu zapíše. Následne keď používateľ stlačí komponent Tlacidlo, tak tento komponent zistí zmenu stavu a vytvorí udalosť. Táto udalosť je odoslaná na plánovač úloh, ktorý ju zaradí do radu spracovávania úloh. Keď príde na rad táto úloha, tak plánovač úloh skontroluje zoznam registrácií na udalosti a príslušnú udalosť vyvolá. V ukázanom prípade sa spustí spracovanie udalosti z objektu SpracujUdalost.

# Integrované vývojové prostredie

Integrované vývojové prostredia (IDE) vznikli aby zjednodušili programovanie aplikácií a informačných systémov. IDE sú navrhnuté tak, aby zahŕňali všetky úlohy spojené s vývojom softvéru v jednej aplikácií. Medzi tieto úlohy patria hlavne:

* editor zdrojového kódu,
* kompilátor,
* nástroje automatizácie.

Editor zdrojového kódu je grafický komponent navrhnutý tak aby zjednodušil písanie programov. Jeho hlavnou črtou je grafické vyznačovanie kľúčových slov v zdrojovom kóde. V lepších IDE je editor doplnený aj o automatické dokončovanie. Automatické dokončovanie podľa znalosti programovacieho jazyka, ale aj už vytvoreného zdrojového kódu, programátorovi ponúka existujúce metódy, premenné. Úloha integrovaného kompilátora je kľúčová pre IDE, pretože testovanie/spúšťanie programu je vďaka tomu na jedno kliknutie. Medzi nástroje automatizácie radíme rôzne ďalšie rozšírenia, ktoré sú spúšťané napr. pred kompiláciou programu. Medzi tieto nástroje môžeme zaradiť aj generátor zdrojového kódu, ktorý prevedie podporné súbory IDE do jazyka, ktorý bude kompilovaný.

## Rapid application development

Metodológia stojaca bok po boku k integrovaným vývojovým prostrediam je Rapid application development (RAD). RAD bol navrhnutý ako prístup adaptívneho softvérového vývoja. RAD odporúča aby sme kládli menší dôraz na plánovanie a väčší na adaptívne zmeny v softvéri. Taktiež sa odporúča zvýšené používanie prototypov, pretože tie skôr poskytnú spätnú väzbu o výslednom softvéri.

# Architektúra komponentového a udalosťami orientovaného riešenia ACProg

Navrhnuté komponentovo a udalosťami orientované riešenie pre Arduino zariadenia budeme nazývať ACProg. Toto riešenie, podľa kapitoly 2.3, zaradzujeme do kategórie offline riešení. Oproti bežnému programovaniu Arduino zariadení sa líši hlavne v tom, že programátor nepotrebuje programátorsky inicializovať zariadenie. Túto inicializáciu vykoná za neho pomocou priloženého konfiguračného XML súboru. Konfiguračný XML súbor obsahuje nastavenia modelu Arduina, definovanie projektu ACProg a zoznam komponentov s ich inicializačnými nastaveniami. Na nasledujúcom obrázku ilustrujeme schému použitia ACProg z pohľadu programátora.



Obr.  Priebeh spracovania projektu, riešením ACProg.

Súborová štruktúra ilustrovaná bodom 1 zobrazuje 2 hlavné projektové súbory. Projekt.xml je konfiguračný súbor projektu, budeme sa mu detailnejšie venovať v ďalšej podkapite, a Projekt.ino je zdrojový kód projektu. XML súbor je následne vstupom do generátora zdrojového kódu. Generátor (bod 2) pomocou konfiguračného súboru vytvorí zdrojový kód komponentov a implementuje základné metódy Arduino programovania setup() a loop(). Vygenerovaný kód je dostupný ako Arduino knižnica (bod 3), ktorá musí byť načítaná v zdrojovom kóde Projekt.ino, prvý riadok zrojového kódu bude #include “Project.h“. Vygenerovaná knižnica a súbor zdrojového kódu Projekt.ino je následne vstupom pre Arduino kompilátor (bod 4). Kompilátor vytvorí súbor inštrukcií pre miktokonrolér na zvolenom modeli Arduino dosky. Tieto inštrukcie môžeme napáliť na mikrokontrolér a vytvorený program spustiť.

## XML Konfiguračný súbor

V tejto podkapitole vysvetlíme ako správne vytvoriť konfiguračný súbor pre generátor projektu ACProg. Pod vysvetlením nájdeme nami vytvorenú ukážkovú XML konfiguráciu pre lepšiu vizuálnu predstavu. Hlavným koreňovým elementom je project. Povinným atribútom je platform, ktorým určíme model Arduino dosky pre ktorú budeme vytvárať projekt. Tento atribút slúži pre kontroly použitých HW prvkov v nasledujúcich prvkoch konfigurácie a tiež pre nastavenie kompilátora pri kompilácii. Nasleduje element program. Element obsahuje číselný atribút watchdog-level, na základe ktorého generátor nastaví hardvérový modul watchdogu na mikrokontroléri. Watchdoch je modul kontrolujúci či sa program nezasekol v mŕtvej slučke a pri detekcii tejto udalosti, zariadenie samostatne reštartuje. Element program obsahuje pole elementov events, uchovávajúce registrácie na udalosti. Programátor sa tak môže pripojiť na metódy setup() a loop() z bežného programovania Arduino zariadení.

Nasledujúcim elementom konfiguračného súboru je pole inicializácií komponentov components. Pole obsahuje elementy component, tie obsahujú 2 povinné elementy name a type. Element type určuje typ komponentu, aký budeme inicializovať. Element name určuje názov inštancie komponentu, pod ktorým k nemu budeme pristupovať v zdrojovom kóde. Element komponentu ďalej obsahuje pole vlastností properties, pomocou ktorých sa nastavujú počiatočné vlastnosti podľa zvoleného typu komponentu. Posledným elementom komponentu je pole udalostí events, v ktorom registrujeme udalosti, aké komponent dokáže vytvoriť, na naše obslužné funkcie.

<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<**project** platform="ArduinoUno">

<**program** watchdog-level="5">

<**events**>

<**event** name="OnLoop">onLoop</event>

<**event** name="OnStart">onStart</event>

</events>

</program>

<**components**>

<**component**>

<**name**>blinkTimer</name>

<**type**>acp.common.timer</type>

<**properties**>

<property name="Enabled">true</property>

<property name="Interval">1000</property>

</properties>

<**events**>

<**event** name="OnTick">mojeTajneKliknutie</event>

</events>

</component>

</components>

</project>

Obr.  Priebeh spracovania projektu, riešením ACProg.

## Generátor knižnice pre Arduino

Pri snahe zachovať syntax a princípy programovania Arduino zariadení, sme sa rozhodli nevytvoriť úplne nový programovací jazyk. Vybrali sme sa smerom vytvorenia Arduino knižnice podľa popisu komponentov. Toto vytvorenie sa nazýva generovanie zdrojového kódu. Generátor dostáva na vstupe zoznam komponentov, kde komponenty majú určený typ (modul). Generátor pridá do výslednej knižnice zdrojové kódy komponentu, ak tam ešte nie sú, a pri zavedení vytvorí kód vytvorenia inštancie. Inštancie komponentov budú globálne a používateľ v nich bude môcť programátorsky meniť konfigurácie za behu. Okrem komponentov generátor vytvorí aj jadro nášho riešenia. Jadro spočíva v inicializácii komponentov a následnom plánovaní vykonávania jednotlivých udalostí. Jadro poskytuje programátorovi aj aplikačné rozhranie na komunikáciu so stálou pamäťou EEPROM.

## Typy komponentov (moduly)

Predpisy typov komponentov sú uložené v repozitári generátora. Repozitár je priečinok uložený na disku. Názov typu komponentu generátor využíva na nájdenie definičného súboru komponentu v súborovej štruktúre repozitára. Názov je delený bodkami čo pre repozitár znamená delenie pod priečinkami.

Každý typ komponentu potrebuje definičný XML súbor pre generátor. Definičný súbor okrem rozšírenej poznámky o komponente obsahuje informácie o súboroch s triedou komponentu, so zoznamom parametrov konštruktora. Ďalej sú v tomto súbore definované názvy všetkých vlastností komponentov a udalostí, ktoré definovaný komponent dokáže vyvolať.

# IDE pre projekt ACProg

Po vzore úspešnosti projektu Arduino, je dôležité vytvoriť k novému riešeniu aj používateľsky prívetivé prostredie. Prostredie v ktorom programátor nie je nútený poznať celé pozadie projektu ACProg a bude mu stačiť koncepčný popis tohto riešenia. V nasledujúcich podkapitolách priblížime používateľské podmienky na nové integrované vývojové prostredie, predstavíme technológie navrhované pre vytvorenie prostredia a architektúru výsledného projektu. V ďalšej podkapitole sa budeme venovať implementačným krokom, ktoré viedli k naplneniu používateľských požiadaviek.

## Používateľské požiadavky

Pred začatím prác na akomkoľvek informačnom systéme (alebo sofvéri) je dobré definovať, čo od neho budeme ako používatelia očakávať. Používateľské požiadavky sme tvorili so znalosťou dnes dostupných integrovaných vývojových prostredí ako sú eclipse, netbeans ale aj samotné Arduino IDE. Kľúčovou úlohou týchto softvérov bolo zjednodušiť programátorom spracovanie ich napísaného zdrojového kódu po jeho spustenie. Preto jednou z používateľských požiadaviek je Automatizovaná kompilácia a spustenie projektu na doske Arduino (use case Kompilácia a spustenie projektu). Konkrétne táto požiadavka stojí za pomerne veľkým úspechom Arduino dosiek vo svete. Všetky ďalšie požiadavky budú priamo spojené s riešením ACProg uvedeným v 5. kapitole. Z XML konfiguračného súboru už poznáme štruktúru komponentov a preto ako ďalšiu požiadavku uvádzame používateľsky prívetivý editor pre túto štruktúru (use case Manažment komponentov). Editor v ktorom po výbere komponentu nám budú ponúknuté len tie vlastnosti a udalosti, aké typ komponentu môže nadobúdať podľa jeho definičného súboru. Keď hovoríme o typoch komponentov, tak potrebujeme používateľovi ukázať aj ponuku všetkých dostupných komponentov z ACProg repozitára (use case Ponuka dostupných komponentov). Neodmysliteľnou súčasťou IDE je editor zdrojového kódu projektu a možnosť usporiadania prostredia podľa vlastných predstáv (use case Voliteľný UX pre programátora). Dôležitou požiadavkou na editor je jeho nápomocnosť programátorovi. Editor by mal dokázať analyzovať vytvorenú konfiguráciu komponentov so zdrojovým kódom a interaktívne napomáhať programátorovi (use case Syntaktická analýza zdrojového kódu). Na začiatok pomoc predstavuje automatizované vytvorenie obslužných funkcií k udalostiam na správnom mieste v zdrojovom kóde, alebo ponúknuť existujúce funkcie v zdrojovom kóde. Okrem toho by mal editor vykonávať aj rozsiahlejšiu analýzu projektu, počnúc kontrolou použitých prostriedkov zvolenej dosky Arudino, cez kontrolu potenciálne duplicitného využívania vstupno-výstupných pinov, až po kontrolu chýbajúcich povinných konfigurácií projektu. Všetky tieto požiadavky by mali programátori napomôcť k rýchlejšiemu prototypovaniu nových zariadení, ale aj k vytvoreniu produkčných zariadení.

### Grafický návrh IDE

Po spísaní používateľských požiadaviek sme navrhli prototyp grafického rozhrania. Prototyp bol určený na otestovanie rozloženia obrazovky, tak aby bolo k dispozícií všetko potrebné práve tam kde by to programátor hľadal. Inšpiráciu sme čerpali z existujúcich používateľských rozhraní pre tvorbu počítačových windows forms aplikácií, konkrétne z programov Microsoft Visual Studio a NetBeans. Na tvorbu prototypu sme vyskúšali rôzne online nástroje (mock-up generátory), avšak ako najlepší nástroj na vytvorenie verného vzhľadu počítačovej formulárovej aplikácie sme použili Microsoft Visual Studio. Na nasledujúcom obrázku je vytvorený prototyp editora.



Obr.  Prototyp integrovaného vývojového prostredia pre ACProg.

Základným prvkom používateľského prostredia sú karty. Každá karta obsahuje grafický používateľský komponent. Komponenty v grafickom prostredí sú:

* Component editor – je komponent s ponukou dostupných ACProg komponentov.
* Designer – je komponent s prehľadom inštancií ACProg komponentov použitých v projekte.
* Code editor – je komponent určený pre editovanie zdrojového kódu.
* Properties – je komponent pre nastavovanie inštančných vlastnosti ACProg komponentu, zvoleného v grafickom komponente Designer.

V hornej lište aplikácie je navrhnuté menu pre nastavovanie typu dosky Arduino, správu projektu (uloženie, vytvorenie), a spustenie generovania s kompiláciou projektu, až po nahratie na dosku Arduino.

## Technologický návrh

### Framework Java Swing

### Docking framework

## Architektúra IDE

Rozdelenie namespacov. Nacrt hlavnych okien a prechody medzi nimi. Popis vlastnosti a spojenia Generatora s IDE.

## Implementácia IDE

### Use case: Manažment komponentov

Spomenut skupinovy zoznam komponentov s presuvanim medzi skupinami. Spomenut aj property editor

### Use case: Ponuka dostupných komponentov

Popis nacitania vsetkych dostupnych komponentov a implementacia vizualnej stranky.

### Use case: Kompilácia a spustenie projektu

Popis prepojenia na generátor, nasledne spustenie kompilacie cez arduino. Pre vystup spomenut konzolu v gui.

### Use case: Syntaktická analýza kódu

Popis vsetkeho co bude hotove

### Use case: Volitelny UX pre programatora

# Vzorové komponenty pre ACProg

## Časovač

## Digitálny výstup

## Digitálny vstup

## Analógový vstup s threshold alert

## Komunikácia pomocou MQTT

## Rádiová komunikácia pomocou 433MHz

Záver

V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom.

Zoznam použitej literatúry

|  |  |
| --- | --- |
| [1] | Giusto, D.; Iera, A.; Morabito, G.; Atzori, L.;, The Internet of Things, 20th Tyrrhenian Workshop on Digital Communications, Springer, New York, NY, 2010. |
| [2] | Tilmanis, P.; White, G., Further Programming in Java, Chapter 7: Event-Driven Programming, Melbourne: RMIT University, 2006. |

Prílohy

1. CD médium – diplomová práca v elektronickej podobe, prílohy v elektronickej podobe.
2. Používateľská príručka
3. Systémová príručka

Táto časť diplomovej práce je povinná a obsahuje zoznam všetkých príloh vrátané elektronických nosičov. Názvy príloh v zozname musia byt’ zhodné s názvami uvedenými na príslušných prílohách. Tlačené prílohy majú na prvej strane identifikačné údaje – informácie zhodné s titulnou stranou diplomovej práce doplnené o názov príslušnej prílohy (Systémová príručka, Používateľská príručka). Identifikačné údaje sú aj na priložených diskoch alebo disketách. Ak je médií viac, sú označené aj číselne v tvare I/N, kde I je poradové číslo a N je celkový počet daných médií.

Každá príloha začína na novej strane a je označená samostatným písmenom alebo číslom (Príloha A, Príloha B, ... alebo Príloha 1, Príloha 2, ...). Číslovanie strán príloh nadväzuje na číslovanie strán v hlavnom texte.

1. delay(int microseconds) funkcia implementuje, čakanie procesora zadaný počet mikrosekúnd. Tradičný kompilátor Arduino implementuje aktívne čakanie. [↑](#footnote-ref-1)