

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**Strojnícka fakulta**

Evidenčné číslo: SjF-16754-104934

**Modulárny systém na ovládanie  
AutomationShield zariadení prostredníctvom  
webového rozhrania**

**Diplomová práca**

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE**

**Strojnícka fakulta**

Evidenčné číslo: SjF-16754-104934

**Modulárny systém na ovládanie  
AutomationShield zariadení prostredníctvom  
webového rozhrania**

**Diplomová práca**

Študijný program: automatizácia a informatizácia strojov a procesov

Študijný odbor: kybernetika

Školiace pracovisko: Ústav automatizácie, informatizácie a merania

Vedúci záverečnej práce: doc. Ing. Martin Gulan, PhD.

Konzultant: Ing. Mgr. Anna Mikulášová

**Bratislava 2025**

**Bc. Matúš Repka**



## ZADANIE DIPLOMOVEJ PRÁCE

Študent: **Bc. Matúš Repka**  
ID študenta: 104934  
Študijný program: automatizácia a informatizácia strojov a procesov  
Študijný odbor: kybernetika  
Vedúci práce: doc. Ing. Martin Gulan, PhD.  
Vedúci pracoviska: doc. Ing. Martin Juhás, PhD.  
Konzultant: Ing. Mgr. Anna Mikulášová

Názov práce: **Modulárny systém na ovládanie AutomationShield zariadení prostredníctvom webového rozhrania**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

1. Oboznámenie sa s iniciatívou AutomationShield a možnosťami jej rozšírenia v oblasti webového ovládania didaktických zariadení AutomationShield, vrátane analýzy spôsobov, ako znížiť vstupnú bariéru pri práci s týmito zariadeniami pomocou webového rozhrania a platformy Arduino R4 WiFi.
2. Štúdium a analýza vhodných riešení na zabezpečenie efektívnej komunikácie medzi Arduinom R4 WiFi a webovou aplikáciou.
3. Návrh a implementácia automatizovaného systému na ovládanie zariadení AutomationShield prostredníctvom webového rozhrania, ktorý umožní výber typu regulácie a zadávanie dát bez potreby opätovnej kompilácie a nahrávania programu.
4. Návrh a realizácia intuitívneho grafického používateľského rozhrania.
5. Rozšírenie webovej aplikácie o návody a používateľskú podporu, preklad do viacerých jazykov, implementáciu denného a nočného režimu, ako aj optimalizáciu pre rôzne rozlíšenia obrazovky.
6. Overenie funkčnosti finálneho systému pridaním ďalšieho modelu AutomationShield a vykonaním skupinového testovania v reálnych podmienkach.

Rozsah práce: 60–80 strán

Termín odovzdania diplomovej práce: 30. 05. 2025

Dátum schválenia zadania diplomovej práce: 29. 05. 2025

Zadanie diplomovej práce schválil: prof. Ing. Cyril Belavý, CSc. – garant študijného programu

## Čestné prehlásenie

Vyhlasujem, že predloženú záverečnú prácu som vypracoval(a) samostatne pod vedením vedúceho záverečnej práce, s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú citované v práci a uvedené v zozname použitej literatúry. Ako autor záverečnej práce ďalej prehlasujem, že som v súvislosti s jej vytvorením neporušil autorské práva tretích osôb<sup>1</sup>.

Bratislava, 30. mája 2025

.....  
Vlastnoručný podpis

---

<sup>1</sup>V súlade s metodickým usmernením STU č. 1/2024-O prehlasujem, že som pri písaní tejto práce využil(a) nástroje umelej inteligencie na podporu jazykovej korektúry, reformulácií, editovania, prekladov a štruktúrovania textu. Tento spôsob využitia spadá pod bod 2b) uvedeného usmernenia a týmto ho transparentne uvádzam.

Touto cestou by som sa úprimne chcel poďakovať pánovi doc. Ing. Martinovi Gulanovi, Phd., za to že ma zobral pod svoje krídla, za jeho odbornú pomoc pri vypracovaní tejto práce. Ďalej by som sa chcel poďakovať aj Ing. Mgr. Anna Mikulášová za jej nápomocné rady, pripomienky a konzultácie rozličných problémov, ako aj pomoc pri testovaní. V neposlednom rade by som chcel poďakovať aj Bc. Janovi Suchému a Bc. Michalovi Paučovi za pomoc pri testovaní a implementácii FurutaShieldu.

Bratislava, 30. mája 2025

Bc. Matúš Repka

**Názov práce:** Modulárny systém na ovládanie AutomationShield zariadení prostredníctvom webového rozhrania

**Kľúčové slová:** AutomationShield, webové rozhranie, Arduino, interaktívne vzdelávanie

**Abstrakt:** Práca sa zaoberá vývojom modulárneho systému na vzdialené ovládanie didaktických zariadení AutomationShield prostredníctvom webového rozhrania s využitím mikroprocesorovej dosky Arduino UNO R4 WiFi. Cieľom riešenia je znížiť vstupnú bariéru pri práci so zariadeniami AutomationShield a vytvoriť platformu, ktorá študentom sprístupní experimentovanie v odbore kybernetiky priamo cez internetový prehliadač. Súčasťou riešenia je analýza komunikačných mechanizmov medzi webovou aplikáciou a mikroprocesorovou platformou, návrh používateľsky prívetivého grafického rozhrania, ako aj implementácia funkcií umožňujúcich výber typu regulácie a zadávanie parametrov bez potreby opätovnej kompilácie programu. Systém je rozšírený o interaktívne návody, viacjazyčnú podporu, možnosť prepínania medzi denným a nočným režimom a responzívny dizajn prispôsobený rôznym typom zariadení. Funkčnosť riešenia bola overená zapojením viacerých zariadení AutomationShield a testovaním so skupinou používateľov v reálnych podmienkach. Navrhované riešenie má potenciál stať sa súčasťou výučby automatizácie na vysokých ako i stredných školách. V rámci ďalšej práce je plánované rozšírenie o ďalšie typy experimentálnych modulov a prepojenie s online vzdelávacími platformami.

**Title:** Modular System for Controlling AutomationShield Devices via a Web Interface

**Keywords:** AutomationShield, web interface, Arduino, interactive learning

**Abstract:** This thesis focuses on developing a modular system for remote controlling of didactic AutomationShield devices using a web interface for the Arduino UNO R4 WiFi microcontroller board. The goal of this thesis is to lower the entry barrier for working with AutomationShield devices and create a platform that enables students to experiment in the field of cybernetics directly through a web browser. The solution includes an analysis of communication mechanisms between the web application and the microcontroller platform, the design of a user-friendly graphical interface, and the implementation of features that allow users to select control modes and input parameters without needing to recompile the program. The system is enhanced with interactive tutorials, multilingual support, a toggle between light and dark modes, and a responsive design adapted to various device types. The functionality of the solution was validated by integrating multiple AutomationShield devices and testing with a group of users in real-world conditions. The proposed solution has the potential to become part of automation education at both university and high school levels. Future work includes expanding the system with additional experimental modules and integrating it with online educational platforms.

# Obsah

<b>Úvod</b>	<b>14</b>
1.1 Čo je to AutomationShield?	14
1.2 Motivácia, ciele a štruktúra práce	16
<b>2 AutomationShield Online</b>	<b>18</b>
2.1 Arduino UNO R4	18
2.2 Riešenie komunikácie medzi Arduino a serverom	19
2.2.1 Nevydarené riešenia a dôvody ich neúspechu	20
2.2.2 Finálne vyriešenie komunikácie	22
<b>3 Vývoj princípov a mechanizmov systému</b>	<b>25</b>
3.1 Registrácia Arduina na server	25
3.2 Ovládanie Shieldov pomocou serveru	27
3.2.1 Slučky experimentov v Arduine	28
3.3 Odosielanie dát experimentu na server	30
3.3.1 Vlastná aplikácia pre serial monitor	31
3.4 Detaily o webovom GUI	32
<b>4 Výsledný systém a jeho štruktúra</b>	<b>34</b>
4.1 Databáza a jej tabuľky	34
4.1.1 Tabuľka arduino_online	34
4.1.2 Tabuľky modelov	35
4.1.3 Tabuľky zaznamenaných dát	36
4.2 Štruktúra procesu celého systému	36
4.2.1 Štart a registrácia Arduina na server	36
4.2.2 Prihlásenie používateľa cez webstránku	38
4.2.3 Experimentovanie pomocou GUI	41
<b>5 Webstránka</b>	<b>46</b>
5.1 Kaskádové štýly	46
5.2 JavaScript kód	48
5.2.1 Cookies	48
5.3 Preklady	48
5.4 Iné	49
5.5 Podporné stránky	50
5.5.1 how-to.php	50
5.5.2 models.php	51
5.5.3 aboutus.php	51
<b>6 Podpora pre ďalšie AS zariadenia</b>	<b>54</b>
6.1 Pridané zariadenia	54
6.1.1 FloatShield	54
6.1.2 MagnetoShield	55
6.1.3 FurutaShield	57

---

6.1.4	Zhrnutie . . . . .	59
<b>7</b>	<b>Testovanie AutomationShield Online</b>	<b>62</b>
7.1	Problém s prekladmi . . . . .	62
7.2	Zlé zobrazenie Cookies . . . . .	63
7.3	Poznámky a požiadavky . . . . .	63
7.3.1	Pomenovanie kariet a Favicon.ico . . . . .	64
7.3.2	Drobné zmeny textov . . . . .	64
7.3.3	Zmeny v GUI formulároch . . . . .	64
7.4	Testovanie projektu na MacOS . . . . .	64
<b>8</b>	<b>Záver</b>	<b>66</b>
8.1	Motivácia na ďalšiu prácu . . . . .	66
	<b>Literatúra</b>	<b>68</b>
<b>A</b>	<b>Prílohy</b>	<b>71</b>
A.1	Arduino INO Flowchart . . . . .	71
A.2	Arduino Online PHP Flowchart . . . . .	74
A.3	Arduino Experiment Data PHP Flowchart . . . . .	76
A.4	Flowchart of submitting experiment data to the DB . . . . .	78



# Zoznam obrázkov

1.1	Didaktické zariadenia AS. . . . .	14
1.2	Bloková schéma PID regulátora. . . . .	16
2.1	Arduino UNO R4 WiFi s vysvieteným poľom LED diód do srdca . . . . .	19
2.2	Jednoduchý webový formulár na ovládanie LED diódy. . . . .	21
2.3	Overenie funkčnosti v DB. . . . .	23
3.1	Tabuľka aktívnych Arduín. . . . .	27
3.2	Prihlásenie do Arduina. . . . .	27
3.3	GUI pre aeroshield. . . . .	29
3.4	Používateľské rozhranie aplikácie SerialMonitor.exe. . . . .	32
3.5	Formulár pre zadávanie experimentov a vykresľované grafy v GUI. . . . .	33
4.1	Prehľad tabuľky arduino_online. . . . .	34
4.2	Prehľad modelovej tabuľky pre AeroShield. . . . .	35
4.3	Prehľad tabuľky nameraných dát pre AeroShield. . . . .	36
4.4	Detail grafu pre Arduino časť setup 1. . . . .	37
4.5	Detail grafu pre PHP časť setup 1. . . . .	37
4.6	Tabuľka dostupných Arduín. . . . .	38
4.7	Detail grafu pre Arduino časť setup 2. . . . .	39
4.8	Prihlasovací formulár. . . . .	39
4.9	Prihlasovací formulár čakajúci na odpoveď. . . . .	40
4.10	Detail grafu pre PHP skript časť setup 2. . . . .	40
4.11	Prihlasovací formulár po nezhode hesiel. . . . .	40
4.12	GUI oznámenie prerušenia spojenia s Arduinom. . . . .	42
4.13	Detail grafu pre Arduino časť experiment 1. . . . .	43
4.14	Detail grafu pre PHP časť experiment 1. . . . .	43
4.15	Detail grafu pre PHP časť experiment 2. . . . .	44
4.16	Detail grafu pre PHP časť experiment 3. . . . .	44
4.17	Detail grafu pre PHP časť experiment 4. . . . .	45
4.18	Detail grafu pre Arduino časť experiment 2. . . . .	45
4.19	Detail grafu pre Arduino časť ukončenie. . . . .	45
4.20	Detail grafu pre PHP časť experiment 5. . . . .	45
5.1	Príklad rozdielu denného a nočného štýlu. . . . .	47
5.2	Ohlásenie cookies používateľovi. . . . .	49
5.3	Znovu otvorenie cookies. . . . .	50
5.4	Príklad z prekladového súboru. . . . .	50
5.5	Úvodný náhľad na stránku how-to.php. . . . .	51
5.6	Náhľad na tretiu sekciu v how-to.php. . . . .	52
5.7	Náhľad na video sekciu v how-to.php . . . . .	52
5.8	Náhľad na stránku models.php. . . . .	53
5.9	Náhľad na stránku aboutus.php. . . . .	53
6.1	Didaktické zariadenie FloatShield od iniciatívy AutomationShield . . . . .	55

---

6.2	GUI pre FloatShield. . . . .	56
6.3	Didaktické zariadenie MagnetoShield od iniciatívy AutomationShield. . . . .	56
6.4	GUI pre MagnetoShield. . . . .	57
6.5	Didaktické zariadenie FurutaShield z open-source iniciatívy AutomationShield. . . . .	58
6.6	GUI pre FurutaShield . . . . .	58
6.7	Databázová tabuľka pre FurutaShield. . . . .	59
6.8	GUI pre FurutaShield s otvorenými nastaveniami. . . . .	60
6.9	Finálna stránka s dostupnými súbormi na stiahnutie pre jednotlivé modely. . . . .	61
7.1	Fotodokumentácia z hromadného testovania AutomationShiel Online. . . . .	62
7.2	Zlé zobrazenie Cookies. . . . .	63
7.3	Staré a nové karty. . . . .	64
7.4	Chybná hláška na vstupných poliach formulárov. . . . .	64
7.5	Oprava vstupných polí zadávania času. . . . .	65
7.6	Polia vstupných parametrov pre PID s prednastavenými hodnotami. . . . .	65

# Zoznam výpisov z kódu

2.1	Jednoduchý PHP výstup generovaný Arduinoom. . . . .	20
2.2	Arduino kód na odoslanie HTTP POST požiadavky. . . . .	22
2.3	PHP skript na spracovanie dát a uloženie do databázy. . . . .	22
2.4	PHP skript na prijatie požiadavky a zápis do databázy. . . . .	23
2.5	Arduino kód na odoslanie HTTP POST požiadavky a spracovanie odpovede. . . . .	24
3.1	Získanie adresy MAC. . . . .	25
3.2	Riadky určené na editáciu používateľom. . . . .	26
3.3	Získanie vstupných dát pre PID experiemnt. . . . .	28
3.4	Inicializácia experimentu. . . . .	28
3.5	Jedna iterácia experimentu. . . . .	29
3.6	Vynulovanie dát a ukončenie. . . . .	30
5.1	Príklad hlavičky HTML kódu. . . . .	47
7.1	Chyba načítania prekladov. . . . .	63

# Zoznam použitých skratiek

<b>AI</b>	umelá inteligencia (Artificial Intelligence).
<b>DB</b>	databáza (Database).
<b>AJAX</b>	Asynchrónny JavaScript a XML (Asynchronous JavaScript and XML).
<b>web</b>	celosvetová sieť www (World Wide Web).
<b>IoT</b>	Internet vecí (Internet of Things).
<b>R4</b>	Arduino UNO R4.
<b>WiFi</b>	bezdrôtová vernosť (Wireless Fidelity).
<b>DAC</b>	digitálno-analógový prevodník (Digital-to-Analog Converter).
<b>2D</b>	dvojdimenziálne (Two-Dimensional).
<b>3D</b>	trojdimenziálne (Three-Dimensional).
<b>MPC</b>	prediktívne riadenie (Model Predictive Control).
<b>BLE</b>	základná lineárna algebra (basic linear algebra).
<b>UART</b>	univerzálne asynchrónne prijímače a vysielачe (universal asynchronous receiver-transmitter).
<b>USB</b>	univerzálna sériová zbernica (universal serial bus).
<b>EMPC</b>	explicitné prediktívne riadenie (Explicit Model Predictive Control).
<b>LQI</b>	integrálne kvadratické riadenie (Linear Quadratic Integral).
<b>LQR</b>	lineárno-kvadratické riadenie (Linear Quadratic Regulator).
<b>LQ</b>	lineárne kvadratické (Linear Quadratic).
<b>PID</b>	proporcionálno-integračno-derivačné riadenie (Proportional-Integral-Derivative).
<b>Kp</b>	proporcionálna zložka (Proportional Gain).
<b>Ti</b>	integračný čas (Integral Time).
<b>Td</b>	derivačný čas (Derivative Time).
<b>AS</b>	AutomationShield.
<b>PCB</b>	doska plošných spojov (Printed Circuit Board).
<b>Shield</b>	rozširujúci modul, ako FurutaShield, AeroShield a ďalšie (Shield).
<b>HTTP</b>	Hypertext Transfer Protocol.
<b>PHP</b>	Hypertext Preprocessor.
<b>IP</b>	internetový protokol (Internet Protocol).
<b>COM</b>	sériový port (serial port, napr. COM port pre Arduino IDE).
<b>GUI</b>	grafické používateľské rozhranie (Graphical User Interface).

**HTML** HyperText Markup Language.

**CSS** Cascading Style Sheets.

**id** identifikátor (identifier).

Skratky neuvedené v tomto zozname, budú lokálne vysvetlené v texte.

# Úvod

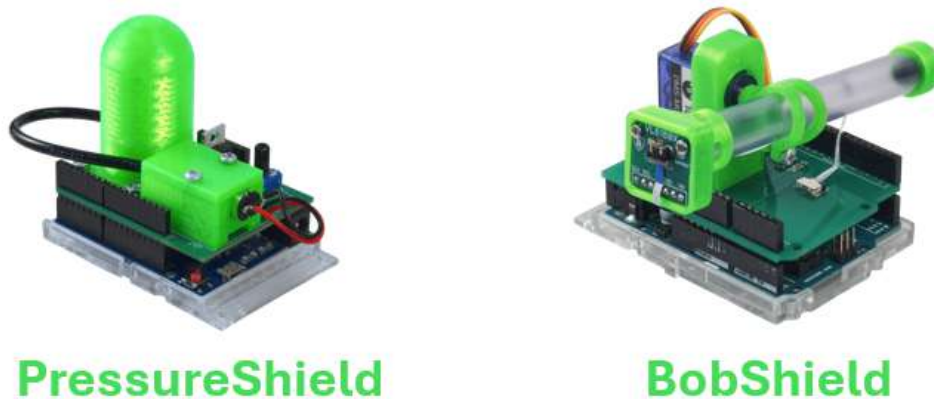
Túžba po nových poznatkoch je v hĺbke duše každého človeka a o to viac v duši študenta. Už od nepamäti historické osobnosti vedy pozorovali prírodné javy a snažili sa prísť na ich príčiny. Osobnosti ako Isaac Newton, Nikola Tesla či James Watt a nespočet ďalších dokázali previesť všetky javy okolitého sveta na matematiku. My máme vďaka nim výhodu v nespočetne veľkom objeme spísanej teórie, ktorú používame pre vývoj techniky a prakticky všetkého ostatného.

Za posledných 100 rokov sme sa posunuli vo vede viac ako za celé predchádzajúce obdobie dejín ľudstva a v rámci modernej vedy vznikla automatizácia, ktorá spája prácu mnohých veľikánov pred nami. Automatizácia je hnacou silou ľudstva, dovoľuje nám nielen rýchlejšiu výrobu, ale vďaka poddisciplíne zvanej *Teória automatického riadenia* [1] vieme pristáť s raketami späť na Zem, vytvoriť humanoidných robotov pohybujúcich sa na dvoch nohách ako človek, či zvýšiť bezpečnosť riadených strojov, ako napríklad automobilov.

## 1.1 Čo je to AutomationShield?

*Teória automatického riadenia* je nosným predmetom na Ústave *Automatizácie, Informatizácie a Merania Strojníckej fakulty STU*, v rámci ktorého spoločne s inými predmetmi vznikla iniciatíva *AutomationShield* (ďalej len AS). Tento projekt umožňuje študentom navrhovať a vyrábať prístroje určené na vzdelávanie a experimentovanie v automatickom riadení a mechanike. Študenti si na rôznych modeloch pracujúcich s odlišnými fyzikálnymi javmi môžu vyskúšať rôzne spätnoväzbové formy regulácie, ako sú PID, MPC, LQ, alebo identifikovať model na základe nameraných dát.

AS zariadenia, ktoré sa nazývajú Štít (anglicky Shield), sú rozširujúce moduly pre mikrokontroléry typu Arduino. Pozostávajú z dosky plošných spojov (PCB) a majú na sebe rôzne elektrotechnické komponenty ktoré v spojení s 3D výtlačkami vytvárajú zariadenie schopné spätnoväzbovej regulácie. Tieto Shiedy, ako môžete vidieť na Obr. 1.1, sú vyrobené tak, aby sa dali jednoducho pripojiť na už spomínané mikrokontrolery Arduino.



Obr. 1.1: Didaktické zariadenia AS.

Celý projekt je navrhnutý tak, aby boli modely dostupné pre každého a náklady neboli príliš vysoké, keďže cena takýchto komerčne dostupných zariadení sa pohybuje v rádoch stovkách až tisícach eur [2]. Medzi modely patria [3]:

- MagnetoShield – magnetická levitácia so spätnou väzbou.
- FloatShield – vznášajúca sa guľička so spätnou väzbou.
- BoBShield – guľička na hojdačke.
- LinkShield – experiment rotácie so spätnou väzbou.
- AeroShield – rýchlosť motora a uhol so spätnou väzbou.
- MotoShield – rýchlosť motora a poloha so spätnou väzbou.
- HeatShield – tepelný systém so spätnou väzbou.
- OptoShield – optický experiment so spätnou väzbou.
- PressureShield – kontrola tlaku.
- BicopShield – experiment s riadením bikotéry.
- FurutaShield – experiment s rotačným invertovaným (Furuta) kyvadlom.

Všetky *Shieldy* sú v kompaktnej veľkosti a kompatibilné s mikroprocesormi od firmy Atmel, ako sú Arduino UNO, ARDUINO MEGA, ARDUINO DUE a ďalšie. Táto kompatibilita umožňuje ich ďalšie rozšírenie, napríklad o novú verziu Arduina UNO R4 WiFi, ktorá umožňuje prepojenie modelov s internetom.

Najjednoduchší príklad spätnoväzbovej regulácie je Proporcionálno–integrálno–derivačný (PID) regulátor, ktorý vidíte na Obr.1.2. PID regulátor patrí k najpoužívanejším metódam spojitého riadenia v priemysle, pretože spája jednoduchú implementáciu s dostatočnou flexibilitou na riadenie širokého spektra procesov [4]. Jeho riadiaci zákon v časovej oblasti je

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}, \quad (1.1)$$

čo po Laplaceovej transformácii vedie k prenosovej funkcii

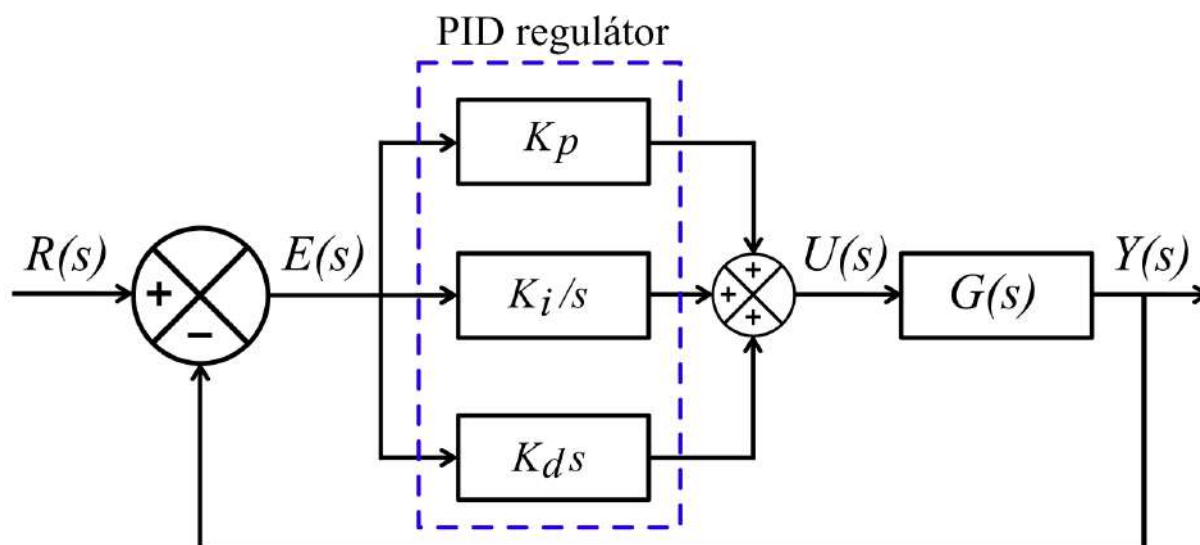
$$C(s) = K_p + \frac{K_i}{s} + K_d s. \quad (1.2)$$

V jednotlivých vetvách proporcionálna zložka ( $K_p$ ) úmerne reaguje na aktuálnu chybu, integrálna ( $K_i$ ) odstraňuje trvalú regulačnú odchýlku akumuláciou chyby a derivačná ( $K_d$ ) predvída budúci vývoj chyby podľa jej rýchlosti zmeny, čím skracuje čas ustálenia a potláča prekmit.

Najjednoduchší príklad spätnoväzbovej regulácie predstavuje bloková schéma PID regulátora zobrazená na Obr. 1.2.

Vysvetlenie premenných na Obr. 1.2

- $\mathbf{R}(s)$  – referenčný vstup (požadovaná hodnota riadenej veličiny),
- $\mathbf{Y}(s)$  – výstup systému (skutočná hodnota veličiny),
- $\mathbf{E}(s)$  – regulačná odchýlka,  $E(s) = R(s) - Y(s)$ ,
- $\mathbf{K}_p$  – proporcionálna (P) zložka,
- $\frac{\mathbf{K}_i}{s}$  – integrálna (I) zložka v Laplaceovej doméne,
- $\mathbf{K}_d s$  – derivačná (D) zložka v Laplaceovej doméne,
- $\mathbf{U}(s)$  – riadiaci signál generovaný PID regulátorom,
- $\mathbf{G}(s)$  – prenosová funkcia riadeného systému.



Obr. 1.2: Bloková schéma PID regulátora.

V moderných aplikáciách je ladenie parametrov  $K_p$ ,  $K_i$  a  $K_d$  často formulované ako optimalizačná úloha. AS poskytuje knižnice, pomocou ktorých používateľ vie naprogramovať spätnoväzbovú reguláciu na Arduino s danými Shieldami za pomoci C++, MATLABu a Simulinku [3]. Okrem klasického PID riadenia ponúka AS možnosť využívať aj pokročilejšie regulačné algoritmy, ako sú napríklad MPC (Model Predictive Control), LQ regulátor alebo Kalmanov filter. MPC je pokročilá technika riadenia, ktorá na základe matematického modelu systému predikuje budúce správanie a v reálnom čase optimalizuje akčný zásah s rešpektovaním obmedzení systému. LQ (Linear Quadratic) regulátor využíva optimalizáciu kvadratickej účelovej funkcie na určenie optimálneho riadenia v lineárnych systémoch. Kalmanov filter slúži ako stavový pozorovateľ, ktorý dokáže z dostupných meraní a modelu systému odhadovať vnútorné stavy aj v prípade výskytu šumu. Tieto metódy predstavujú dôležitý nástroj pre návrh moderných, robustných a efektívnych riadiacich štruktúr.

## 1.2 Motivácia, ciele a štruktúra práce

Moja motivácia pre túto prácu siaha až do detstva. Nikdy som nebol zameraný iba na jednu oblasť techniky. Už na základnej škole som reprezentoval školu vo fyzikálnych, matematických aj technických súťažiach. Automatizáciu a elektrotechniku som študoval už na strednej škole, no rozhodol som sa pokračovať vo vysokoškolskom štúdiu na Strojníckej fakulte. Nebol som však úplne spokojný s tým, že moje vysokoškolské štúdium prestalo zasahovať do iných technických disciplín, a tak ma zaujal Ústav automatizácie, informatizácie a merania (UAIM). Medzitým som sa stal aj webdeveloperom a hľadal som spôsob, ako prepojiť všetky tieto odbory. Keď sa naskytla možnosť spojiť mechatroniku s komunikáciami a webdevelopmentom, veľmi ma to nadchlo a na diplomovej práci som začal pracovať s veľkým elánom.

Mojou úlohou je vytvoriť interaktívne webové prostredie, ktoré zrýchli a zlepši prácu so Shieldami, ponúkne okamžitú odozvu a generovanie dát z experimentov, poskytne dobrú didaktickú pomocku pre vyučujúcich a vytvorí tzv. *nábežnú zónu* pre študentov menej skúsených v oblasti programovania alebo teórie automatizačného riadenia.

V rámci tejto diplomovej práce sa sústredím aj na vytvorenie návodov na inštaláciu a prácu s Arduino a AutomationShield zariadeniami, ako aj zaučenie študentov do problematiky rôznych riadiacich metód ako PID, LQ a MPC.

V rámci práce sa teda budem snažiť:

- oboznámiť sa s iniciatívou AutomationShield a preskúmať možnosti jej rozšírenia v oblasti webového ovládania didaktických zariadení, vrátane analýzy spôsobov, ako znížiť vstupnú bariéru pri



ich používaní pomocou webového rozhrania a platformy Arduino R4 WiFi,

- analyzovať a študovať vhodné riešenia na zabezpečenie efektívnej komunikácie medzi Arduino R4 WiFi a webovou aplikáciou,
- navrhnuť a implementovať modulárny automatizovaný systém na ovládanie zariadení AutomationShield cez webové rozhranie, ktorý umožní výber typu regulácie a zadávanie dát bez potreby opätovnej kompilácie a nahrávania programu,
- navrhnuť a realizovať intuitívne grafické používateľské rozhranie s jasnými inštrukciami pre používateľa,
- rozšíriť webovú aplikáciu o používateľské návody, podporu viacerých jazykov, denný a nočný režim a zabezpečiť optimalizáciu pre rôzne rozlíšenia obrazovky,
- overiť funkčnosť finálneho systému pridaním ďalšieho modelu AutomationShield a vykonaním skupinového testovania v reálnych podmienkach.

Štruktúra práce pozostáva zo siedmich kapitol. Výslednému projektu som priradil meno **AutomationShield Online**, ako zdôraznenie toho, že celý projekt je rozšírením iniciatívy AS. V prvej kapitole rozoberiem základy práce ako mikroprocesor, na ktorom bude celý projekt postavený, porovnanie s ostatnými mikroprocesormi a riešenie typu komunikácie medzi serverom a Arduino, ktorý použijem. V druhej kapitole rozoberiem vývoj celej štruktúry systému a riešenia, ktoré som použil. Tretia kapitola obsahuje popis výsledného systému, štruktúru databázy a chronologicky vysvetľuje dianie a postup od registrácie Arduina na server až po ovládanie Shieldov pomocou webového GUI. Vo štvrtej kapitole rozoberám štruktúru webstránky, pravidiel programovania, ktoré som si sám zadal a podrobnosti o webstránke spolu s kontentom, ktorý sa na nej nachádza. Piatá kapitola obsahuje proces implementácie ďalších AS zariadení do môjho systému. Posledná kapitola obsahuje skupinové testovanie a potvrdenie funkčnosti aj na iných zariadeniach.

## 2 AutomationShield Online

V tejto kapitole pokladám základy pre nasledujúci vývoj celého projektu. Výber správneho zariadenia (mikrokontrolera) a formy komunikácie sú kritické a odráža sa od nich následný vývoj. Pre vývoj a testovanie som si mohol vybrať z niekoľkých Shieldov, rozhodol som sa pre *AeroShield*, keďže som s ním počas štúdia najviac pracoval a dobre som ho poznal. Ako som spomínal v úvode, všetky Shiedly majú svoje vlastné PCB kompatibilné so širokým výberom zariadení, a tak som hardvérovú kompatibilitu nemusel riešiť. Jediný problém mohol vzniknúť pri funkcii vzorkovania od AutomationShieldu, keďže PCB Arduina UNO R4 (ďalej už len R4) má nový mikroprocesor. Na moje šťastie už bola táto softvérová kompatibilita vyriešená predchádzajúcimi vývojármi AutomationShieldu.

### 2.1 Arduino UNO R4

Projekt Arduino vznikol v roku 2005 a odvtedy ponúklo široké spektrum modelov, pričom sa stalo populárnou platformou nielen medzi hobbyistami, ale aj v akademických a priemyselných aplikáciách. Na internete možno nájsť tisíce publikácií, návodov a riešení na rôzne Arduino projekty – od jednoduchých mechanických hodín cez RC modely až po pokročilé priemyselné prototypy [5].

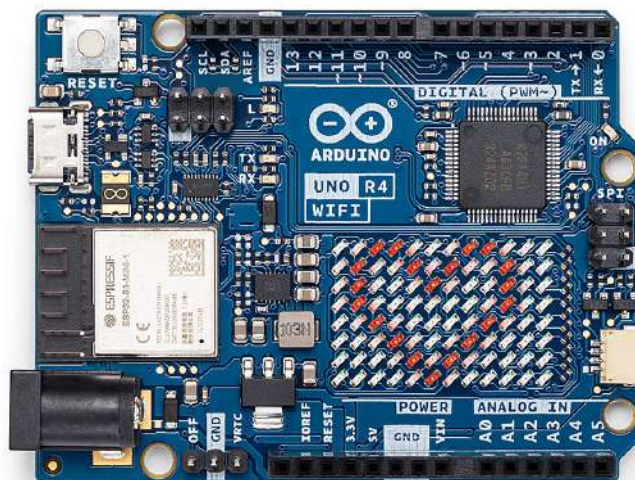
V roku 2023 bolo vydané *Arduino UNO R4* z Obr. 2.1, ktoré predstavuje významnú evolúciu oproti *Arduino UNO R3*, jednému z najpopulárnejších modelov v histórii platformy. Hlavnou zmenou je použitie výkonnejšieho mikrokontrolera *Renesas RA4M1 ARM Cortex-M4*, ktorý nahrádza starší *ATmega328P* použitý v R3. Tento nový čip prináša vyšší takt (48 MHz oproti 16 MHz v R3), viac pamäte (32 KB RAM oproti 2 KB v R3) a širší rozsah napájacieho napätia (3.3V aj 5V, zatiaľ čo R3 podporovalo len 5V) [6].

Okrem toho má model UNO R4 WiFi zabudovaný druhý mikrokontrolér – *Espressif ESP32-S3-MINI*, ktorý zabezpečuje komunikáciu cez USB-C port a zároveň umožňuje pripojenie na internet cez WiFi alebo Bluetooth. To je zásadný rozdiel oproti R3, ktoré nemalo žiadne natívne možnosti bezdrôtovej komunikácie a vyžadovalo externé WiFi alebo Ethernet Shieldy. Hlavné porovnanie môžete vidieť v Tab. 2.1.

Ďalšou novinkou v R4 je LED pole ( $8 \times 5$  matica), ktoré je určené predovšetkým na edukačné účely a umožňuje jednoduché vizualizácie dát priamo na doske. Vývojári Arduina k R4 poskytli aj oficiálnu knižnicu s niekoľkými príkladmi použitia, avšak nejde o rozsiahle návody – cieľom tohto modelu je poskytnúť flexibilnú platformu pre komunitu, ktorá môže objaviť jeho plný potenciál.

Keďže je tento model pomerne nový a aj keď podľa firmy Atmel využívajú Arduino platformu desiatky miliónov ľudí po celom svete, na internete zatiaľ nie je veľa komplexných projektov založených na R4 – obzvlášť tých, ktoré by využívali jeho internetové možnosti [8]. Používatelia, ktorí potrebovali sieťovú konektivitu pred vydaním R4, si väčšinou vystačili s alternatívami, ako sú ESP8266, ESP32 alebo Ethernet/WiFi Shieldy pre R3. Tento projekt je preto do istej miery jedinečný a neexistuje mnoho priamych zdrojov inšpirácie. Navyše, väčšina Arduino komunity sú hardvéroví nadšenci, nie webdeveloperi, čo môže byť ďalším dôvodom menšieho počtu projektov súvisiacich s internetovou konektivitou R4.

V súhrne, aj keď sa UNO R4 na prvý pohľad môže zdať ako len mierne vylepšená verzia UNO R3, v skutočnosti ide o zásadnú zmenu v architektúre, výkone a možnostiach. Kvôli týmto rozdielom, na základe autorov prehľadu literatúry, neexistujú hotové riešenia pre všetky použitia a bude potrebné vyvinúť vlastné metódy komunikácie medzi serverom a Arduinom. Práve táto komunikácia bude kľúčovým aspektom mojej práce a ovplyvní aj celkový smer vývoja a realizovateľné funkcie.



Obr. 2.1: Arduino UNO R4 WiFi s vysvieteným poľom LED diód do srdca [7].

Tabuľka 2.1: Porovnanie vlastností Arduino UNO R3 a Arduino UNO R4 [6]

Vlastnosť	Arduino UNO R3	Arduino UNO R4
Mikrokontrolér	ATmega328P (8-bit, 16 MHz)	Renesas RA4M1 (32-bit, 48 MHz)
RAM	2 KB	32 KB
Flash pamäť	32 KB	256 KB
Napájanie	Iba 5V	5V aj 3,3V
USB port	Micro-USB	USB-C
WiFi / Bluetooth	Nie, vyžaduje Shield	Áno, v modeli UNO R4 WiFi
Druhý mikrokontrolér	Nie	ESP32-S3-MINI
LED pole	Nie	Áno, 8x5 matica
DAC	Nie	Áno (12-bitový DAC)

## 2.2 Riešenie komunikácie medzi Arduino a serverom

Prvý a najpodstatnejší problém, ktorý som musel prekonať, bolo vyriešiť komunikáciu medzi Arduino a serverom. Arduino UNO R4 (ďalej iba R4) ponúka niekoľko riešení ako napríklad:

- Hostovanie vlastného servera – Arduino môže bežať ako webový server a spracovávať HTTP požiadavky [9].
- Počúvanie prichádzajúcich správ (server mode) – Arduino môže byť v režime servera a prijímať dáta od klientov.
- Klientská komunikácia cez HTTP/S – Arduino sa môže pripájať k externému serveru a odosielať/prijímať dáta cez HTTP GET/POST.
- MQTT (Message Queuing Telemetry Transport) – Protokol vhodný pre IoT aplikácie, umožňuje efektívnu výmenu správ medzi zariadeniami a serverom [10].
- WebSockets – Obojsmerná komunikácia v reálnom čase, vhodná pre aplikácie, ktoré vyžadujú rýchlu odozvu [11].
- Bluetooth (R4 Minima nemá, ale R4 WiFi podporuje BLE) – Použiteľné na komunikáciu so smartfónmi alebo inými zariadeniami [12].

- Serial (UART) – Tradičná sériová komunikácia cez USB alebo hardvérové sériové piny.

Pre moju prácu bolo podstatné nájsť takú formu komunikácie, akou by som mohol sekvenčne alebo stavovo riadiť Arduino pre spúšťanie experimentov, ale aj vyriešiť to, ako sa užívateľ pripojí na svoje Arduino. Pre toto som potreboval hlavne možnosť odosielať dáta v podobe premenných s hodnotami z Arduina na server, ale aj opačne. Pre nájdenie správneho riešenia som išiel systematicky podľa toho, čo som našiel v článkoch o novom R4.

### 2.2.1 Nevydarené riešenia a dôvody ich neúspechu

Nanešťastie, správne riešenie nebolo ľahké nájsť a predstavovalo to veľkú výzvu. V rámci toho, že prácu som chcel uvoľniť od potenciálnych nekompatibilit v budúcnosti a zároveň uľahčiť prácu pre užívateľa, som sa rozhodol nepoužívať knižnice od tretích strán, ale len tie, ktoré poskytuje samotné Arduino.

#### Server hostovaný na Arduine

Ako prvé som začal testovať hostovanie vlastného servera na Arduine. Snažil som sa vytvoriť server, ktorý by počúval prichádzajúce POST a GET požiadavky, ktoré by určili, čo má Shield robiť a slúžili by na prenos dát medzi R4 a mojím serverom [13]. Táto komunikácia sa zdala sľubná, vytvoril som server, ktorý mal kód fungujúci na princípe z výpisu z kódu 2.1.

Výpis z kódu 2.1: Jednoduchý PHP výstup generovaný Arduinom.

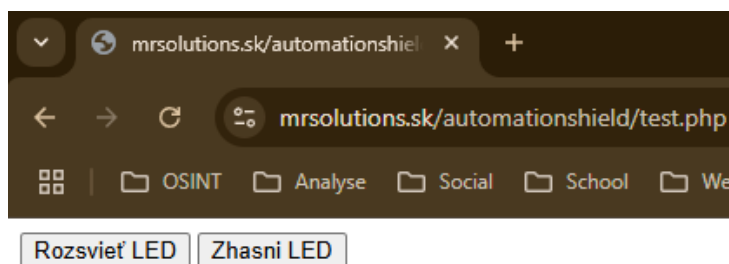
```
<?php
if (isset($_GET['x'])) {
    $x = $_GET['x'];

    if ($x == "1") {
        echo "LED_ON";
        // Tu by sa na Arduine zapla LED
    } elseif ($x == "0") {
        echo "LED_OFF";
        // Tu by sa na Arduine vypla LED
    } else {
        echo "Neplatna_hodnota!";
    }
} else {
    echo "Chyba_parameter_x!";
}
?>
```

Pre rozsvietenie LED diódy bolo potrebné do internetového prehliadača zadať IP adresu spolu s GET žiadosťou, napríklad: `192.168.1.100?x=1`. Po odoslaní tejto požiadavky sa LED dióda rozsvietila alebo zhasla podľa hodnoty parametra *x*. Moje experimentovanie s týmto typom komunikácie prinieslo pozitívne výsledky – dokázal som zapisovať hodnoty do premenných aj spúšťať funkcie v Arduine.

Problém však nastal pri návrhu používateľského rozhrania na serveri. V predchádzajúcich experimentoch som GET požiadavky zadával priamo do prehliadača, ale teraz som potreboval tento proces integrovať do webového formulára. Ako prvý test som vytvoril jednoduché rozhranie s dvoma tlačidlami, ako je vidieť na Obr. 2.2.

Problém bol v tom, že tento formulár bol hostovaný na mojom serveri, zatiaľ čo kód uvedený vo výpise z kódu 2.1 bol nasadený na zariadení R4. To znamenalo, že klient musel komunikovať priamo so serverom R4. Jednoduché presmerovanie požiadaviek, kde by tlačidlá obsahovali odkazy typu `192.168.1.100?x=1` a `192.168.1.100?x=0`, nefungovalo, pretože presmerovalo klienta z mojej stránky na R4 bez možnosti návratu na stránku formulára. Možným riešením by bolo, aby R4 po spracovaní požiadavky presmerovala používateľa späť, ale takéto riešenie nebolo vhodné na produkčné nasadenie, kde by viacerí používatelia



Obr. 2.2: Jednoduchý webový formulár na ovládanie LED diódy.

ovládali rôzne Shiedly. Navyše by toto riešenie porušovalo môj cieľ zachovať modularitu a jednoduchosť systému, nehovoriac o možných problémoch so spoľahlivosťou a rýchlosťou siete.

Ako alternatívu som skúšal odosielať dáta z formulára pomocou AJAX-u, teda posielať GET požiadavky v štýle `192.168.1.100?x=a` asynchrónne, bez nutnosti opätovného načítania stránky. AJAX umožňuje odosielanie a prijímanie dát na pozadí, čím sa zvyšuje interaktivita a dynamika webových aplikácií. V našom prípade by AJAX umožnil komunikovať s Arduino priamo z webového rozhrania, pričom tlačidlá by len odosieli požiadavky na R4 bez nutnosti opustiť stránku.

Toto riešenie však nefungovalo – LED dióda sa nerozsvietila ani nezhasla. Hľadanie príčiny bolo zdĺhavé, no dospel som k záveru, že R4 vyžaduje, aby požiadavky prichádzali z fyzického klienta (napríklad z prehliadača), nie z asynchrónneho skriptu na strane klienta.

Ďalšou možnosťou bolo nehostovať GUI na externom serveri, ale priamo na Arduino. Toto riešenie som však považoval za neefektívne, pretože by sme stratili podporu pre dynamické prvky, ako sú interaktívne grafy, animácie a moderný dizajn. Z tohto dôvodu som sa preto rozhodol ukončiť experimentovanie s vlastným serverom a hľadať alternatívne riešenia.

## WiFi Chat Server

Ako ďalší prístup som sa rozhodol vyskúšať WiFi Chat Server, ktorý by umožňoval obojsmernú komunikáciu medzi Arduino a klientským zariadením prostredníctvom webového prehliadača [14]. Myšlienkou bolo využiť jednoduchý server bežiaci na Arduino, ku ktorému by sa bolo možné pripojiť cez prehliadač a odosielať textové správy, ktoré by zariadenie spracovalo.

Prvým krokom bolo spustenie servera na Arduino a pokus o nadviazanie spojenia cez prehliadač. Napriek správne nastaveniu IP adresy a portu som sa však nedokázal pripojiť. Spojenie bolo neustále blokové a prehliadač nedokázal nadviazať kontakt so serverom. Vzhľadom na vyššie spomínané problémy s tým, že na internete o Arduino R4 informácií zatiaľ moc nenachádza, som nedokázal nájsť riešenie môjho problému.

Možné príčiny zlyhania tohto riešenia sú:

- **Chýbajúca podpora WebSockets v prehliadači** – Ak server využíval WebSockets a prehliadač nebol správne nakonfigurovaný na ich podporu, spojenie sa nemuselo podariť.
- **CORS (Cross-Origin Resource Sharing)** – Požiadavky odoslané z iného zdroja (napríklad z webovej stránky na mojom serveri) mohli byť prehliadačom automaticky zablokovévané kvôli bezpečnostným obmedzeniam.
- **IPv4 vs. IPv6 adresovanie** – Niektoré zariadenia môžu preferovať IPv6 adresy, pričom Arduino mohlo podporovať len IPv4, čo mohlo spôsobovať konflikty pri nadviazovaní spojenia.
- **Problém s autentifikáciou** – Niektoré servery vyžadujú špecifický handshake alebo iný spôsob autentifikácie, ktorú Arduino neimplementovalo, čím mohla byť komunikácia zamietnutá.
- **Port Blocking** – Firewall alebo router mohol blokovévať komunikáciu na danom porte, čo znemožňovalo spojenie.

- **Nesprávny sieťový mód** – Ak bolo Arduino nastavené ako Access Point (AP) a prehliadač sa pokúšal pripojiť cez inú sieť, spojenie nemohlo byť úspešné.

Po viacerých pokusoch som nedokázal spoľahlivo vyriešiť tento problém tak, aby bolo možné stabilne komunikovať s WiFi Chat Serverom cez prehliadač. Hoci by bolo teoreticky možné obísť niektoré z uvedených prekážok, uvedomil som si, že takéto riešenie by mohlo byť príliš komplikované pre bežného používateľa.

Aj keby sa mi podarilo odstrániť tieto problémy, stále by existovalo riziko, že by sa s nimi stretli aj budúci používatelia môjho riešenia, čím by sa znížila jeho spoľahlivosť a použiteľnosť. Z tohto dôvodu som sa rozhodol upustiť od experimentovania s WiFi Chat Serverom a hľadať alternatívne riešenia, ktoré by poskytovali jednoduchšiu a stabilnejšiu komunikáciu medzi Arduino a serverom.

## 2.2.2 Finálne vyriešenie komunikácie

Po niekoľkých slepých uličkách som sa rozhodol rozdeliť komunikáciu a vyriešiť najprv, ako odosielať dáta z Arduina na web, a to, ako by som odosielať dáta späť do Arduina, by som vyriešil až potom. Pri mojom hľadaní som objavil, že niektorí autori riešení používali inú knižnicu od Arduina, ktorá nie je predvolene nainštalovaná v Arduino IDE. Ide však o oficiálnu knižnicu od Arduina a jej inštalácia je natoľko jednoduchá, že som sa rozhodol skúsiť ju, nakoľko by nemala robiť žiadne problémy pre užívateľov a zároveň nemám strach, že by Arduino zmenilo jej funkčnosť v budúcich verziách tejto knižnice. Jedná sa o knižnicu *ArduinoHttpClient*, okrem iných vecí táto knižnica pridáva možnosť odosielať POST a GET požiadavky priamo z Arduina [15].

### Testovanie odosielania dát na web

Pri testovaní knižnice *ArduinoHttpClient* som ako prvé musel zistiť, či je možné spoľahlivo odosielať dáta na webový server. Bežne POST a GET požiadavky vo webových prehliadačoch fungujú na princípe fyzického presmerovania – ak klient odošle dáta prostredníctvom formulára, po ich odoslaní je presmerovaný na stránku, ktorá ich spracuje. Na tejto stránke môže byť zobrazovací skript, ktorý odoslané údaje vypíše.

V mojom prípade som však na odosielanie dát nepoužíval prehliadač, ale Arduino, ktoré nemá natívnu možnosť zobrazovania prijatej odpovede v grafickom rozhraní. Preto som potreboval iný spôsob, ako overiť, či dáta boli úspešne odoslané.

Riešením bolo jednoduché nastavenie testovacieho PHP skriptu na mojom serveri, ktorý prijaté dáta uloží do databázy. Arduino následne odosielať testovacie dáta pomocou HTTP POST požiadavky. Po odoslaní som mohol údaje skontrolovať priamo v databáze a overiť tak správnosť komunikácie. Testovací kód použitý v Arduino na odosielanie údajov je uvedený vo výpise z kódu 2.5 [16].

Výpis z kódu 2.2: Arduino kód na odoslanie HTTP POST požiadavky.

```
String postData = "data=HelloWorld";
client.beginRequest();
client.post("test.sk/test.php");
client.setHeader("Content-Type", "application/x-www-form-urlencoded");
client.setHeader("Content-Length", postData.length());
client.beginBody();
client.print(postData);
client.endRequest();
String response = client.responseBody();
Serial.print("Server response: ");
Serial.println(response);
```

Na strane servera som vytvoril PHP skript (výpis z kódu 2.4), ktorý spracováva prichádzajúcu POST požiadavku a ukladá prijaté údaje do databázy.

Výpis z kódu 2.3: PHP skript na spracovanie dát a uloženie do databázy.

```
<?php
```

```

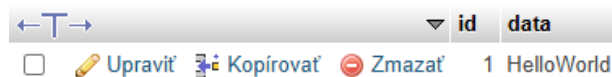
$db = new mysqli("localhost", "user", "password", "database");

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $stmt = $db->prepare("INSERT INTO test (data) VALUES (?)");
    $stmt->bind_param("s", $_POST["data"]);
    $stmt->execute();
    echo "Data saved!";
    $stmt->close();
}

$db->close();
?>

```

Po úspešnom odoslaní dát bolo možné ich skontrolovať v databáze, čo potvrdilo, že komunikácia medzi Arduino a webovým serverom funguje správne. Tento výsledok je znázornený na Obr. 2.3.



Obr. 2.3: Overenie funkčnosti v DB.

## Odosielania dát na Arduino

V kóde z výpisu z kódu 2.5 si môžete všimnúť, že v poslednom riadku je napísané *Serial.println(response)*. Hoci bol tento experiment zameraný na odosielanie dát z Arduina na web, testoval som zároveň aj možnosť prijatia odpovede zo servera.

HTTP odpoveď sa v prehliadačoch používa hlavne pre chybové kódy, ako je známa *404 - page not found*, avšak v odpovedi môže byť aj textový reťazec. To som využil pri testovaní. V PHP kóde z výpisu z kódu 2.4 sa nachádza riadok *echo "Data saved"*. Po odoslaní dát z Arduina som na serial monitore skutočne uvidel správu *"Data saved"*, čím som potvrdil, že Arduino dokáže odpoveď zo servera nielen prijať, ale aj spracovať.

Výpis z kódu 2.4: PHP skript na prijatie požiadavky a zápis do databázy.

```

<?php
$db = new mysqli("localhost", "user", "password", "my_database");

if ($db->connect_error) {
    die("Connection failed: " . $db->connect_error);
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['case'])) {
    $stmt = $db->prepare("INSERT INTO test (data) VALUES (?)");
    $stmt->bind_param("s", $_POST['case']);

    if ($stmt->execute()) {
        echo "ON";
    } else {
        echo "ERROR";
    }

    $stmt->close();
}

```

```
$db->close();  
?>
```

Po tejto pozitívnej odozve som sa zameral na testovanie možnosti spustiť na Arduino funkciu na základe prijatej odpovede. Zmenil som teda odpoveď v PHP skripte z *echo "Data saved"*; na *echo "ON"*; a doplnil som do Arduina podmienku, ktorá rozsvieti LED diódu v prípade prijatia tejto odpovede, ako je vidieť vo výpise z kódu 2.5.

Výpis z kódu 2.5: Arduino kód na odoslanie HTTP POST požiadavky a spracovanie odpovede.

```
String postData = "case=HelloWorld";  
client.beginRequest();  
client.post("test.sk/test.php");  
client.setHeader("Content-Type", "application/x-www-form-urlencoded");  
client.setHeader("Content-Length", postData.length());  
client.beginBody();  
client.print(postData);  
client.endRequest();  
String response = client.responseBody();  
Serial.print("Server response:");  
Serial.println(response);  
  
if(response == "ON"){  
    digitalWrite(LED_BUILTIN, HIGH);  
}else{  
    Serial.print("Nieco je zle");  
}
```

Prvý test ukázal, že riešenie funguje. Po niekoľkých neúspešných pokusoch s inými metódami som konečne našiel efektívny spôsob, ako dosiahnuť spoľahlivú komunikáciu medzi Arduino a webovým serverom.

Rozdiel medzi týmto riešením a predchádzajúcimi pokusmi spočíva v tom, že pri predchádzajúcich testoch som sa snažil odosielať POST a GET požiadavky z webového servera na Arduino. Toto riešenie však funguje opačne – komunikáciu iniciuje vždy Arduino a server mu len odpovedá. Tento prístup je stabilnejší a jednoduchší na implementáciu, keďže eliminuje problémy s konfiguráciou sieťových povolení a firewallov, ktoré by mohli blokovat priamu komunikáciu zo servera na zariadenie.



## 3 Vývoj princípov a mechanizmov systému

Po úspešnom testovaní a nájdení tej správnej formy komunikácie som sa zameral na navrhnutie správnej štruktúry vyhotovenia celého projektu. V tejto časti som musel nie len prísť na to, ako vyriešiť nové problémy, ale aj navrhnuť scenáre týchto problémov, nakoľko ešte nebolo jasné, ako bude celá práca vyzeráť. Ako prvé som odložil samotné ovládanie Shieldu stranou a zameral som sa na to, ako sa študent či iný používateľ vôbec pripojí na svoje Arduino s tým, že moja práca je zameraná na to, aby mohlo viacero používateľov ovládať svoje Shiedly v rovnakom čase. K tomu je potrebné, aby si každý používateľ mohol nejakým spôsobom nájsť svoje Arduino a prihlásiť sa do neho. Ďalej je potrebné uistiť sa, že sa cudzia osoba nemôže pripojiť na Arduino, ktoré nie je ich.

### 3.1 Registrácia Arduina na server

Na vyriešenie problému prihlasovania som potreboval nejaký ukazovateľ, ktorý by bol jedinečný pre každé jedno Arduino. Našťastie je takýto ukazovateľ práve pre môj účel už vyvinutý firmou Xerox Network Systems vyše 50 rokov späť [17]. MAC adresa sa používa práve pri sieťových komunikáciách ako pevná adresa zariadení, keďže adresa ako IP sa môže meniť zmenou pripojenia alebo geografiou, MAC adresa zariadení ostáva pevná. Toto vyhovuje pre moju prácu. Rýchly prieskum ukázal, že keďže Arduino UNO R4 má v sebe mikroprocesor *ESP32-S3-MINI-1-N8*, tak je vybavené MAC adresou, ktorá sa dá veľmi ľahko získať, ako vidno vo výpise 3.1 z môjho kódu [18].

Výpis z kódu 3.1: Získanie adresy MAC.

```
String getMacAddress() {
    byte mac[6];
    WiFi.macAddress(mac);
    String macAddress = "";
    for (int i = 0; i < 6; i++) {
        if (i > 0) {
            macAddress += ":";
        }
        macAddress += String(mac[i], HEX);
    }
    macAddress.toUpperCase();
    return macAddress;
}
```

Ukazovateľ v podobe MAC adresy, mi sám o sebe stačil na samostatnú registráciu jednotlivých Arduín. Stále som však potreboval nejaké jednoduché riešenie aj pre používateľa, nakoľko hľadať svoje Arduino v tabuľke plnej MAC adries alebo manuálne zadávanie tejto adresy pre pripojenie, nepovažujem za *používateľsky priateľské* riešenie. Z toho dôvodu som pridal premennú *ArduinoName*, ktorá je jednou zo štyroch premenných, ktoré ako jediné musí používateľ editovať v mojom finálnom riešení. Na začiatku

kódu pre každý jeden Shield sa nachádza kód z výpisu z kódu 3.2, v ktorom sa nachádza meno a heslo od WiFi. Tieto premenné sú sebevysvetľujúce, keďže R4 sa vie pripojiť na WiFi, potrebuje najprv jej meno a heslo. To musí zabezpečiť používateľ. Môže kľudne použiť aj *mobilný hot-spot*.

Výpis z kódu 3.2: Riadky určené na editáciu používateľom.

```
//#####
//##### Login and Wifi credentials #####
//##### User should edit this #####

const char* ssid = "T-263844";           // WiFi network name
const char* password = "ua3fhuaedtha";   // WiFi network password
const char* arduinoName = "Test_arduino"; // Name displayed on the web
const char* shield_password = "pass";     // Password for web login

//#####
//##### Don't edit under this line #####
//#####
```

V celej práci som kládol dôraz na dobré označenie a okomentovanie každého riadku, funkcie či premennej, ako môžete vidieť vo výpise z kódu 3.2. Poslednou premennou z výpisu z kódu 3.2 je premenná *shield\_password*. Jej predvolená hodnota je *pass* a však užívateľ by mal túto hodnotu zmeniť na vlastnú, nakoľko práve vďaka tomuto heslu som zaručil, že sa cudzia osoba neprihlási na Arduino, ktoré jej nepatrí. V rámci toho, že sa nejedná o žiadne osobné dáta a aj po mojej stáži vo *vládnej jednotke CSIRT.SK* (*Computer Security Incident Response Team Slovakia*), som usúdil, že zabezpečenie pri výmene mena a hesla medzi Arduinom a webom nie je potrebné zabezpečovať viac, ako formou POST požiadavky. V prípade potreby by však bolo možné použiť bežné kryptografické techniky, ako napríklad *bcrypt* [19], *Argon2* [20] či *SHA-256* na ukladanie hesiel, alebo šifrovanie komunikácie pomocou *AES* či *ECC* [21].

### Ako to funguje?

V jednoduchosti, Arduino po zapnutí a pripojení sa na WiFi odosiela svoje dáta a to **model, MAC, meno a heslo**, webstránke, ktorá tieto dáta spracuje a vloží do databázy. Po úspešnej registrácii sa Arduino každých pár sekúnd dotazuje, aké je užívateľom zadané heslo. Na Obr. 3.1, môžete vidieť výslednú tabuľku, pomocou ktorej si užívateľ nájde svoje Arduino.

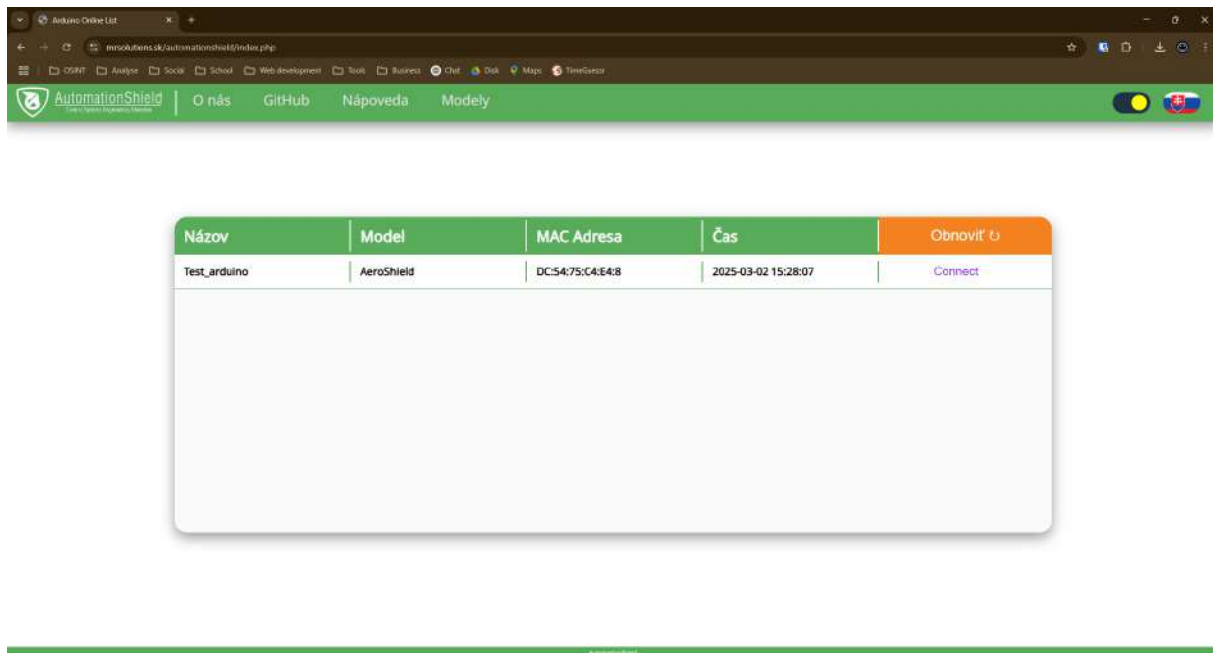
Po zvolení svojho Arduina, je užívateľ presmerovaný na stránku prihlásenia, ktorá je vidieť na Obr. 3.2. Po zaslaní správneho hesla je užívateľ presmerovaný na stránku jednotlivého modelu, napríklad **AeroShield.php**, kde, ako si neskôr vysvetlíme, môže ovládať Shield pomocou GUI, ktoré som vytvoril.

Aj keď celý proces od štartu Arduina až po experimentovanie v nekonečnej slučke bude vysvetlený v kapitole 3, je dôležité dodať niekoľko vecí.

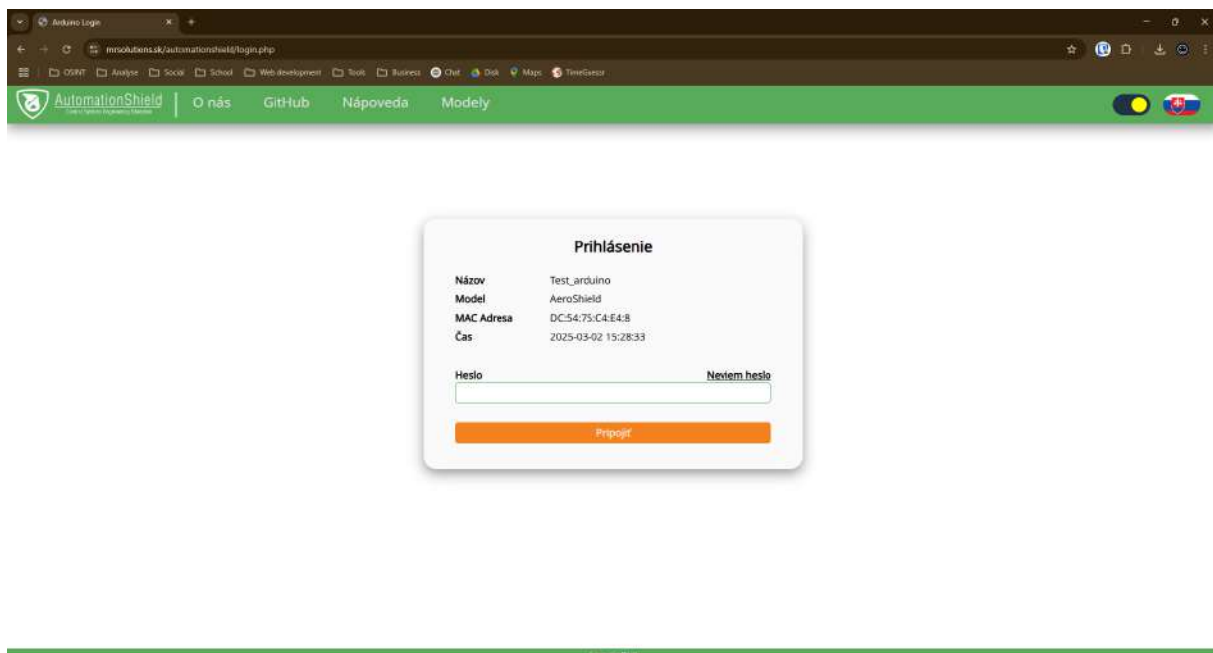
Arduino sa zobrazuje v tabuľke z Obr. 3.1 15 sekúnd. Tento 15 sekundový odpočet je však reštartovaný vždy, keď sa Arduino dotazuje webstránky na heslo. Tým pádom, pokiaľ má Arduino stabilné pripojenie na internet a prívod elektrickej energie, tak má používateľ neobmedzený čas na prihlásenie. V momente prihlásenia sa v databáze zmení booleovská premenná *connected*, ktorá po zmene zaručí, že Arduino okamžite zmizne z tabuľky dostupných Arduín.

Druhý dodatok je pripomenutie komunikácie, nakoľko zvolená forma je, že všetka komunikácia, či už registrácia Arduina alebo kontrolovanie zadaného hesla, je inicializovaná zo strany Arduina. Webstránka sama o sebe neodosiela Arduinu žiadne dáta, ale len odpovedá na žiadosti Arduina. Na tieto žiadosti odpovedá formou funkcie *switch case*, kde Arduino okrem dát ako meno, heslo či MAC, odosiela vždy aj premennú *case*, ktorá hovorí stránke, aká je práve otázka. Príklad premennej *case*:

- **case:setup** = registrácia Arduina do databázy.
- **case:password\_check** = dotazovanie na heslo.
- **case:updateStatus** = oznámenie že heslo bolo správne.



Obr. 3.1: Tabuľka aktívnych Arduín.



Obr. 3.2: Prihlásenie do Arduina.

## 3.2 Ovládanie Shieldov pomocou serveru

Po úspešnom prihlásení, je užívateľ prepojený na stránku modelu. V tejto stránke som mal za úlohu vytvoriť už samotné GUI pre ovládanie Shieldu. Vzhľadom na efektívny systém registrácie a prihlásenia som len jednoducho pokračoval. Po prvýkrát za celú prácu tu už nie sú potrebné hlbšie výskumy, teda aspoň čo sa týka ovládania Arduina a komunikácie. Čo bolo však potrebné otestovať, je metóda získavania dát z odpovedí webstránky. Doteraz sme používali veľmi jednoduché odpovede, teraz však

prichádza na rad zasielanie dát pre experiment, keďže napríklad taký **PID** experiment pre AeroShield potrebuje parametre **Kp**, **Ti**, **Td**, **referenčný vektor a časové údaje** [22]. Na tieto parametre by som sa mohol pýtať po jednom, no v rámci jednoduchosti PHP skriptu a lepšej modulárnosti na pridávanie iných experimentov a iných modelov, som vyskúšal vyhotoviť toto dotazovanie komplexnejšie. Vo výpise z kódu 3.3 je vidieť, ako jednoducho dostanem parametre experimentu z odozvy stránky, ktorá môže vyzerať napríklad takto: **Response: 'Experiment=PID&Kp=1.55&Ti=0.55&Td=0.25&r1=23.00&r2=30.00&r3=50.00&r4=15.00&r5=40.00&r6=15.00&period=5'**. Jednotlivé dáta viem z takéhoto stringu selektovať a zapísať do preddefinovaných premenných, ktoré viem následne aplikovať pre vykonanie experimentu.

Výpis z kódu 3.3: Získanie vstupných dát pre PID experiemnt.

```
Kp = response.substring(response.indexOf("Kp=") + 3,
response.indexOf("&Ti=")).toFloat();
Ti = response.substring(response.indexOf("Ti=") + 3,
response.indexOf("&Td=")).toFloat();
Td = response.substring(response.indexOf("Td=") + 3,
response.indexOf("&r1=")).toFloat();
R[0] = response.substring(response.indexOf("r1=") + 3,
response.indexOf("&r2=")).toFloat();
R[1] = response.substring(response.indexOf("r2=") + 3,
response.indexOf("&r3=")).toFloat();
R[2] = response.substring(response.indexOf("r3=") + 3,
response.indexOf("&r4=")).toFloat();
R[3] = response.substring(response.indexOf("r4=") + 3,
response.indexOf("&r5=")).toFloat();
R[4] = response.substring(response.indexOf("r5=") + 3,
response.indexOf("&r6=")).toFloat();
R[5] = response.substring(response.indexOf("r6=") + 3,
response.indexOf("&period=")).toFloat();
T = response.substring(response.indexOf("period=") + 7).toInt();
```

Po krátkom teste odosielania viacerých informácií cez odozvu stránky, som vytvoril nekonečnú štruktúru, kde sa Arduino do nekonečna každé dve sekundy pýta, či užívateľ zadal cez GUI z Obr. 3.3 nejaký experiment. Pokiaľ bol experiment zadáný, Arduino sa spýta, aké sú dáta experimentu. Z odpovede najprv vyberie premennú experimentu a určí, či sa jedná o PID, MPC, LQ alebo iné riadenie. Následne podľa experimentu vyberie premenné z odpovede, ktoré potrebuje. Potom inicializuje experiment, po jeho ukončení odosiela dáta o tom, že bol experiment ukončený a hneď sa pýta, či nebol zadán nový experiment. Je teda v takejto nekonečnej slučke, ktorá je opísaná v nasledujúcej kapitole.

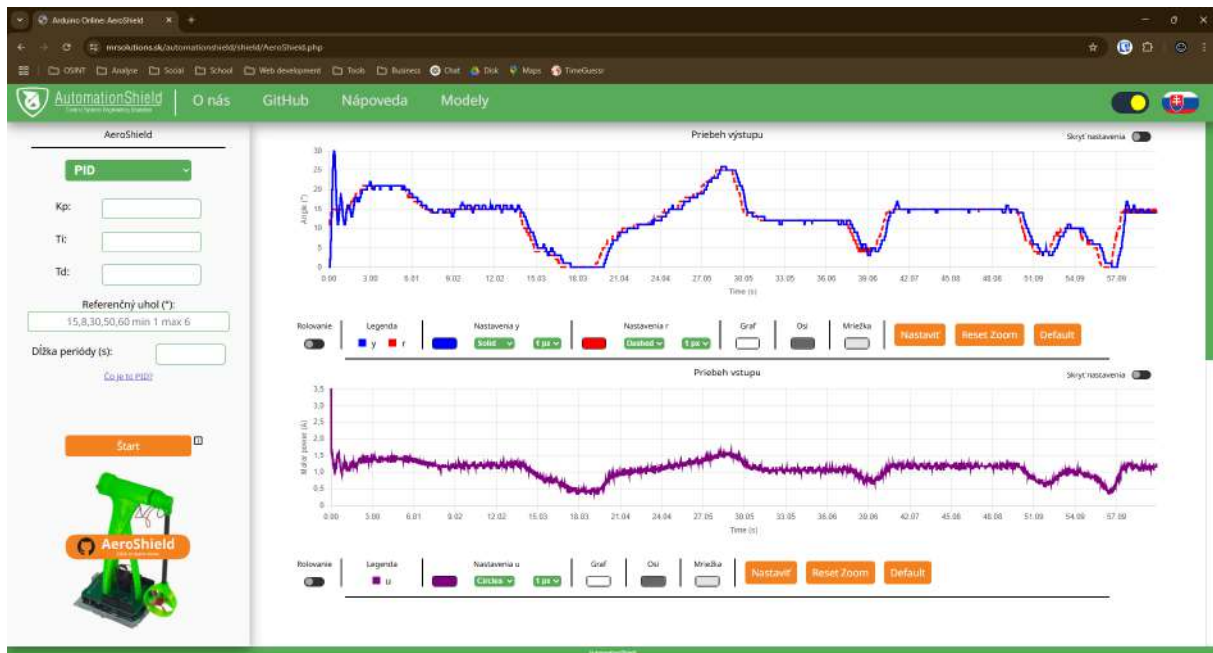
### 3.2.1 Slučky experimentov v Arduine

Zatiaľ čo v originálnych programoch sa všetky hodnoty pre funkcie daných experimentov inicializovali ešte pred alebo vo funkcii **void setup()**; a experimenty prebiehali priamo vo **void setup()**; s využitím vlastnej funkcie **void step()**, ja som musel vymyslieť iný systém. V mojom riešení sa po získaní dát inicializuje experiment a vstúpi do časovo obmedzenej slučky, ktorá sa v originálnych experimentoch nenachádzala. Dobu trvania tejto slučky pre každý experiment volí užívateľ v mojom GUI. Vo výpise z kódu 3.4 je vidieť príklad toho, ako sa experiment spúšťa, nadväzujúc na výpis z kódu 3.3.

Výpis z kódu 3.4: Inicializácia experimentu.

```
while(R[Ri]>0){
    Ri++;
}
T = response.substring(response.indexOf("period=") + 7).toInt();
timer = T * Ri;
T = T * 200;
```

### 3. Vývoj princípov a mechanizmov systému



Obr. 3.3: GUI pre aeroshield.

```
AeroShield.calibrate();
PIDAbs.setKp(Kp);
PIDAbs.setTi(Ti);
PIDAbs.setTd(Td);
unsigned long startTime = millis();
safetyStop = false;

//### PID control loop ###
while ((millis() - startTime < (unsigned long)timer * 1000)
&& !safetyStop) {
    safetyAngle();
    if (i > Ri) {
        analogWrite(5, 0);
        safetyStop = true;
    }
    if (nextStep) {
        stepPID();
        nextStep = false;
    }
}
```

V príklade 3.4 je vidieť dve funkcie a to **safetyAngle()** a **stepPID()**. Funkcia **safetyAngle** je kontrolná funkcia, ktorá ukončí experiment predčasne, ak je uhol naklonenia priveľký pomocou premennej **safetyStop**. Funkciu **stepPID** môžete vidieť vo výpise z kódu 3.5. Ide o aplikáciu funkcie **step()**, ktorá sa používa v originálnych funkciách pre Shildy, upravenú, aby vyhovovala mojim potrebám.

Výpis z kódu 3.5: Jedna iterácia experimentu.

```
void stepPID() {
    if (k % (T * i) == 0) {
        r = R[i];
```

```

    i++;
}
k++;
y = AeroShield.sensorRead();
u = PIDAbs.compute(r - y, 0, 100, 0, 100);
AeroShield.actuatorWrite(u);
}

```

S takouto štruktúrou sa mi podarilo bez problémov vytvoriť experimentové slučky pre niekoľko rôznych druhov experimentov, ktoré sú:

- PID,
- LQ,
- LQ manuálne (referencia je ovládaná potenciometrom),
- MPC,
- MPC manuálne (referencia je nastavená potenciometrom),
- otvorená slučka (výkon motora je ovládaný potenciometrom bez regulácie),
- Identifikácia (výkon motora je zadaný užívateľom v GUI).

Pravá efektívnosť sa potvrdí až pri implementovaní tohto systému spolu s registráciou a ostatnými vecami na ďalší Shield, nakoľko pri vývoji bol použitý len už spomínaný AeroShield. Pri vývoji však bol vždy kladený dôraz na modularitu, čo sa však podarilo, nakoľko počas vývoja som pridával a upravoval mnoho vecí, no úpravy aj pridania boli v skutku jednoduché.

Po vykonaní experimentu sú vždy použité premenné vynulované, ako vidno vo výpise z kódu 3.6 a následne je odoslaná na web informácia o ukončení experimentu.

Výpis z kódu 3.6: Vynulovanie dát a ukončenie.

```

for (int clear = 0; clear < Ri ; clear++) {
    R[clear] = 0;
}
i=0;
k=0;
Ri=0;
timer=0;
T=0;
experiment = 0;
u = 0;
y_safety = 0;
AeroShield.actuatorWrite(0);

```

### 3.3 Odosielanie dát experimentu na server

Odosielanie dát experimentu späť na web, kde môžu byť dáta zobrazené, je jediný problém, pri ktorom mi funkčnosť Arduina a Shieldov ako takých nedovolili vytvoriť efektívne riešenie, ktoré by užívateľa nestálo žiadnu námahu. Pri AeroShielde sú dáta experimentu väčšinou uhol naklonenia, výkon motora a referenčný uhol. Tieto dáta sa merajú v každej iterácii spomínaných funkcií typu **step**. Ako prvé som sa pokúsil odosielať dáta v podobe POST požiadavky, rovnako ako pri každej inej komunikácii medzi Arduino a webstránkou. Tu nastal vážny problém s tým, že AutomationShield pracuje s veľmi vysokou frekvenciou vzorkovania a teda aj výpočtu, rádovo 100 iterácií za sekundu. Arduino UNO R4, aj napriek tomu, že má dva mikroprocesory, nevie odosielať dáta a zároveň vykonávať výpočty a merať parametre.

Pri každom pokuse o odoslanie dát na webstránku počas experimentu sa cyklus experimentu prerušil, nové iterácie neprebíhali a na motorčeku bola zapísaná posledná hodnota pred POST požiadavkou. Iterácie sa ďalej spustili až po dostaní odpovede, čo bolo aj 2 sekundy, neprípustná hodnota pre AutomationShield experimenty. Pokúsil som sa zmenšiť dobu prerušenia tým, že som ukončil POST požiadavku hneď po jej zaslaní, bez čakania na odpoveď, no prerušenie bolo stále prítomné a narušovalo to regulačné algoritmy.

Ako riešenie som sa pokúsil dáta ukladať v preddefinovaných maticiach, ktoré by som na stránku zaslal až po vykonaní experimentu. Toto riešenie sa však ukázalo neplatným, pretože merané hodnoty boli reprezentované vo FLOAToch, a teda pri tomto množstve iterácií sa zaplnila pamäť Arduina v priebehu dvoch až troch sekúnd a Arduino nabehlo na chybu nedostatku pamäte. Pokúšal som sa nájsť iné riešenie, ako použiť napríklad EEPROM pamäť, tá má však ešte menšiu kapacitu ako pamäť SRAM.

Ďalším možným riešením bolo preprogramovanie mikroprocesora **ESP32-S3-MINI-1-N8**, ktorý sa v systéme využíva na komunikáciu. ESP32-S3-MINI-1-N8 je v rámci Arduino UNO R4 fyzicky prepojený so zvyškom dosky, pričom jeho komunikácia a správa prebieha cez špecifické hardvérové spojenia [23]. Na to, aby bolo možné tento mikroprocesor preprogramovať, je potrebné manuálne spojiť pájkou jednu cestu na PCB R4. Túto úpravu je ťažké vrátiť späť a môže viesť k nepredvídaným problémom, vrátane straty záruky alebo trvalého poškodenia dosky pri nesprávnom zásahu [24].

Okrem komunikačných funkcií zabezpečuje ESP aj ďalšie dôležité procesy, medzi ktoré patrí správa USB komunikácie. To znamená, že akékoľvek zásahy do jeho firmvéru by mohli viesť k nestabilite celého systému alebo k strate niektorých dôležitých funkcií dosky. Pri návrhu riešenia by tak bolo nevyhnutné zachovať existujúce správanie ESP, čo výrazne komplikuje vývoj nového firmvéru [24].

Preprogramovanie ESP vyžaduje detailnú znalosť jeho hardvérových a softvérových špecifikácií, ako aj dôkladné testovanie výsledného firmvéru. Vzhľadom na to, že ESP32-S3-MINI-1-N8 používa špecifické spôsoby správy periférií a komunikácie, bolo by potrebné vykonať komplexnú analýzu existujúceho kódu, následne ho upraviť a vykonať rozsiahle testovanie [24]. Tento proces je časovo veľmi náročný a vyžaduje pokročilé znalosti *vstavaných (embedded) systémov*.

Hlavným dôvodom, prečo som toto riešenie zavrhol, je vysoká náročnosť pre užívateľa a potenciálne znehodnotenie Arduina len na jeden účel. Aj keď by toto riešenie mohlo byť reálne a dala by sa do firmvéru zakomponovať funkcia, ktorá by odosielať dáta automaticky, riziká a marginálne zhoršenie modularity a jednoduchosti použitia jednoznačne nedovoľujú toto riešenie zrealizovať.

### 3.3.1 Vlastná aplikácia pre serial monitor

V rámci doterajšieho výskumu som dospel k záveru, že vytvorenie vlastného seriálneho monitora, ktorý by sledoval a spracovával dáta zozbierané Arduinom, by mohlo umožniť efektívne odosielanie týchto dát na webovú platformu bez negatívneho vplyvu na priebeh experimentov. Pri debugovaní som už aj tak vypisoval rôzne parametre, ako sú napríklad model, experiment, MAC adresa a samotné namerané dáta experimentov vo formáte:

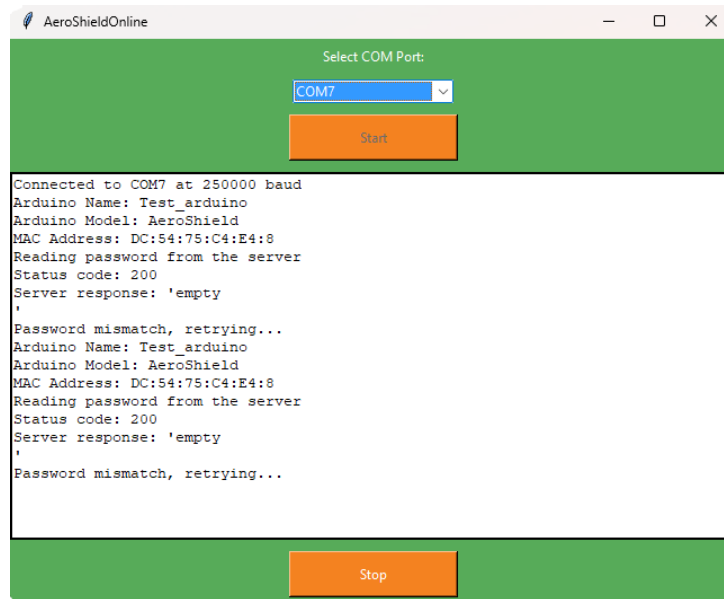
r: 20.0 y= 19.59 u= 6.5

Kde:

- **r** – požadovaný uhol kyvadla v stupňoch,
- **y** – nameraný uhol kyvadla v stupňoch,
- **u** – akčný zásah na motor v miliampéroch.

Na tento účel som navrhol a implementoval Python aplikáciu, ktorá bola následne skompilovaná do spustiteľného súboru `SerialMonitor.exe` [25]. Grafický výstup z tejto aplikácie je znázornený na Obr. 3.4.

Navrhnutý serial monitor funguje na princípe vyhľadávania MAC adresy, modelu zariadenia a experimentu v prenášaných dátach. Po spustení Arduina je jedinou povinnosťou používateľa vybrať správny komunikačný kanál (COM port), čo zabezpečí automatickú detekciu potrebných identifikačných údajov. Tieto údaje sa následne použijú ako parametre pre PHP skript, ktorý zaistí uloženie experimentálnych dát do správnej tabuľky v databáze.



Obr. 3.4: Používateľské rozhranie aplikácie SerialMonitor.exe.

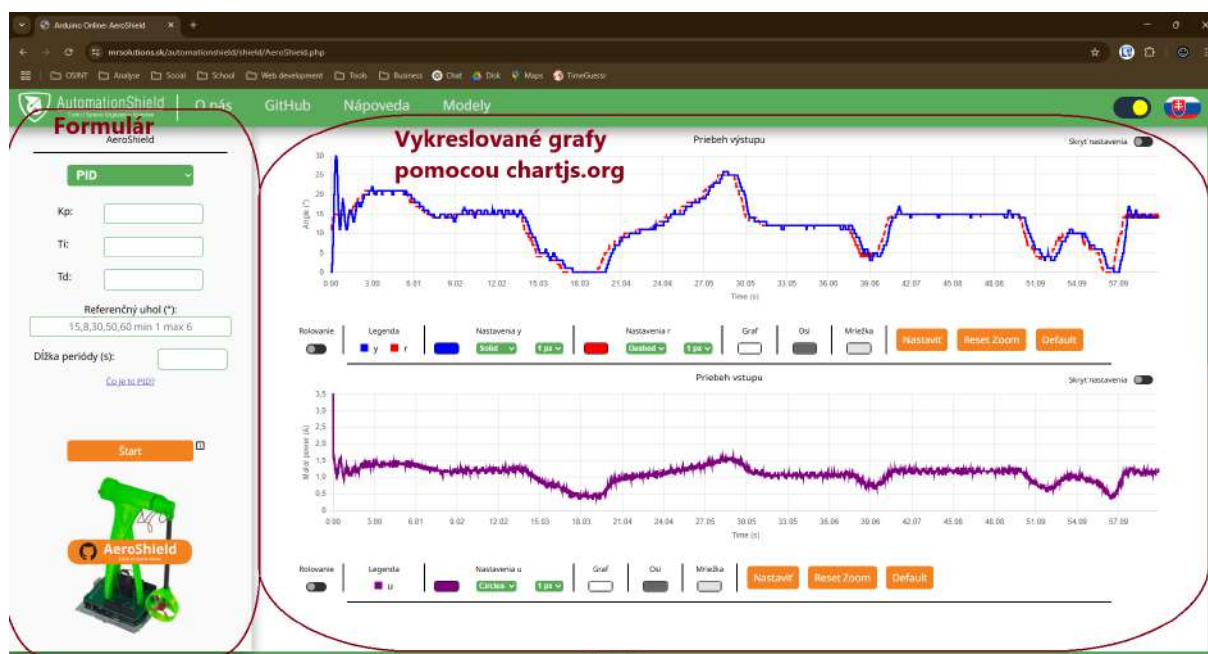
Zber experimentálnych dát prebieha paralelne so seriálnym monitorovaním. Skript spracováva premenné **r**, **y** a **u**, ktoré sa ukladajú do matíc a odosielaajú na web po každých 50 iteráciách alebo po ukončení experimentu. Implementované riešenie sa osvedčilo ako spoľahlivé a používateľsky jednoduché. Napriek tomu, že tento prístup mierne vybočuje od pôvodných cieľov, jeho praktické výhody prevyšujú nevýhody spojené s nutnosťou stiahnutia a používania aplikácie SerialMonitor.exe.

### 3.4 Detaily o webovom GUI

Pri webovom GUI som nemusel robiť nejaké výskumy, nakoľko väčšinu vecí som vyriešil vďaka klasickému PHP, jQuery a AJAXu. Formulár, ktorý vidíte na Obr. 3.5, posiela dáta pomocou AJAXu do tabuľky AeroShield, odkiaľ ju číta Arduino. Každý Shield má svoje vlastné GUI, s rovnakými prvkami, upravenými pre individuálne požiadavky modelov. Zaujímavou časťou je však vykreslenie dát. Pomocou jQuery a PHP skriptu sú dáta z experimentu získavané z tabuľky, ktorej názov je MAC adresa Arduina, na ktorú dáta posiela vyššie spomínaná aplikácia serial monitoru. Na vykresľovanie som použil open-source knižnicu **chartjs.org**, ktorá je spolu navrhnutá pre vykresľovanie veľkých datasetov vo webových rozhraniach pomocou rôznych developerských jazykov, v mojom prípade JavaScriptom a PHP skriptom [26].

Predvolené nastavenie vykreslenia grafov je veľmi neestetické a neupravené pre veľké datasety a živé pridávanie dát. Dostať grafy do konečnej verzie vyžadovalo nielen mnoho grafických a funkčných úprav, ale aj veľa optimalizácie vykresľovania, nakoľko sa v týchto grafoch nachádzajú radovo tisíce iterácií z experimentov a webový prehliadač nie je vhodne optimalizovaný pre takéto procesy a priebežné prepočty. Musel som preto nájsť vhodný kompromis medzi plynulosťou, estetickosťou a prehľadnosťou. GUI prešlo počas vývoja mnohými zmenami, je plné tooltipov, možnými úpravami zobrazovaných grafov a podobne. Podrobné fungovanie grafu je opísané v nasledujúcej kapitole.





Obr. 3.5: Formulár pre zadávanie experimentov a vykresľované grafy v GUI.

## 4 Výsledný systém a jeho štruktúra

Všetky funkcie, protokoly a štruktúry, ktoré som použil, spolu s dôvodmi ich použitia, sme si zhrnuli v predchádzajúcej kapitole. Táto kapitola poskytuje podrobný prehľad kompletnej práce a opis celého systému krok po kroku. Predtým si však musíme prejsť všetky databázové tabuľky, ich stĺpce, vysvetliť ich význam a kde sa používajú.

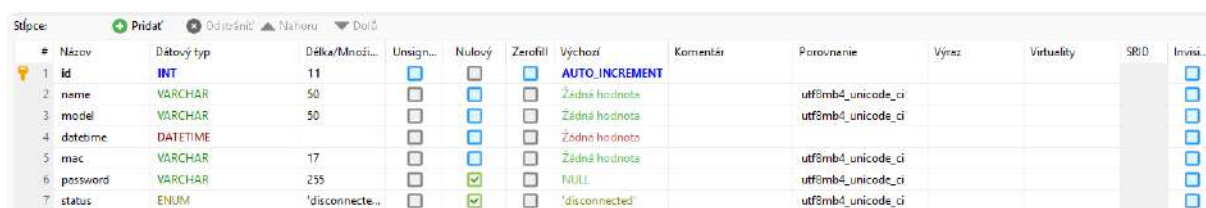
### 4.1 Databáza a jej tabuľky

V rámci celej práce sa používa niekoľko databázových tabuliek, ktoré sú rozdelené do troch hlavných kategórií:

- Štádiovo-komunikačná tabuľka **arduino\_online**.
- **Tabuľky modelov**, pričom každá nesie názov príslušného modelu Shieldu, napríklad **AeroShield** alebo **FurutaShield**.
- **Tabuľky zaznamenaných dát**, ktoré sa generujú automaticky a sú pomenované podľa MAC adresy konkrétneho Arduina.

#### 4.1.1 Tabuľka arduino\_online

Hlavná tabuľka **arduino\_online**, ktorej štruktúru možno vidieť na Obr. 4.1, slúži na identifikáciu a sledovanie pripojených zariadení.



The screenshot shows a database table structure for 'arduino\_online'. The table has 12 columns: #, Názov, Datový typ, Dĺžka/Množ., Unsigned, Nullový, Zerofill, Výchozí, Komentár, Porovnanie, Výrez, Virtuálny, SRID, and Invisi... The rows are as follows:

#	Názov	Datový typ	Dĺžka/Množ.	Unsigned	Nullový	Zerofill	Výchozí	Komentár	Porovnanie	Výrez	Virtuálny	SRID	Invisi...
1	id	INT	11	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO INCREMENT						<input type="checkbox"/>
2	name	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Žiadna hodnota		utf8mb4_unicode_ci				<input type="checkbox"/>
3	model	VARCHAR	50	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Žiadna hodnota		utf8mb4_unicode_ci				<input type="checkbox"/>
4	datetime	DATETIME		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Žiadna hodnota						<input type="checkbox"/>
5	mac	VARCHAR	17	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Žiadna hodnota		utf8mb4_unicode_ci				<input type="checkbox"/>
6	password	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		utf8mb4_unicode_ci				<input type="checkbox"/>
7	status	ENUM	'disconnect...	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	'disconnected'		utf8mb4_unicode_ci				<input type="checkbox"/>

Obr. 4.1: Prehľad tabuľky arduino\_online.

Táto tabuľka obsahuje nasledujúce stĺpce:

- **id** – automaticky pridelené číslo označujúce riadok databázy, ktoré sa však používa ako primárny spájací kľúč s ostatnými tabuľkami a PHP formuláry.
- **name** – do tejto premennej je vložené používateľom zvolené meno Arduina, použité len pre jednoduchšiu identifikáciu v tabuľke aktívnych Arduín.
- **model** – AeroShield / FurutaShield / všetky ostatné - identifikátor toho, aký Shield sa na danom Arduine nachádza, zároveň táto premenná slúži pre nasmerovanie na správnu GUI stránku po úspešnom prihlásení.

- **datetime** – periodicky upravovaná premenná Arduina poukazujúca na to, že Arduino je naozaj online, pokiaľ sa daný čas neupraví, Arduino zmizne z tabuľky aktívnych Arduín, vyobrazenej na hlavnej stránke alebo sa preruší spojenie s GUI, podľa toho, v akej fázy sa Arduino a používateľ nachádzajú.
- **mac** – MAC adresa Arduina, ide o premennú, ktorá je so 100% určitou unikátna a teda je použitá ako hlavný vyhľadávací kľúč v databáze spolu s premennou **id**, vďaka jej unikátnosti ju zároveň používam aj ako názov tabuľky, ktorú vysvetľujem nižšie.
- **password** – táto premenná slúži na zapisovanie hesla z prihlasovacieho formuláru, odkiaľ sa na nu dotazuje Arduino.
- **status** – premenná poukazujúca na to, či sa užívateľ do Arduina prihlásil, alebo nie.

Táto tabuľka je pevná a teda existuje neustále, dáta sa do nej len vkladajú alebo upravujú.

### 4.1.2 Tabuľky modelov

Pre každý model je vytvorená samostatná tabuľka, v ktorej sa nachádzajú vstupné dáta pre experiment ako napríklad referenčná krivka, doby trvania experimentu a podobne. Príklad môžete vidieť na Obr. 4.2 kde je zobrazená štruktúra modelovej tabuľky pre AeroShield .

#	Názov	Datový typ	Dĺžka/Množ.	Unsigned	Nullový	Zerofill	Výchozí	Komentár	Porovnanie	Výraz	Virtuality	SRID	Invis...
1	id	INT	11	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT						<input type="checkbox"/>
2	arduino_online...	INT	11	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Základná hodnota						<input type="checkbox"/>
3	Experiment	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	IDLE		utf8mb4_unicode_ci				<input type="checkbox"/>
4	START	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	STOP		utf8mb4_unicode_ci				<input type="checkbox"/>
5	TIMER	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	0						<input type="checkbox"/>
6	Kp	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
7	Ti	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
8	Td	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
9	r1	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
10	r2	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
11	r3	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
12	r4	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
13	r5	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
14	r6	DECIMAL	10,2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
15	T	INT	11	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>

Obr. 4.2: Prehľad modelovej tabuľky pre AeroShield.

Na rozdiel od tabuľky arduino\_online, nemajú však všetky premenné udelenú len jednu funkciu, pretože AeroShield, tak ako aj ostatné Shildy, vedia previesť niekoľko rôznych druhov experimentov, na ktoré potrebujú rozdielne dáta. Potrebne dáta sú často v tvare vektorov a je zbytočné vytvárať novú premennú v tabuľke pre každý jeden prípad, keď ide o ten istý typ dát. Popis aktuálnych funkcií stĺpcov tabuľky:

- **id** – identifikačné označenie riadku v tabuľke.
- **arduino\_online\_id** – je id daného Arduina pre danú MAC adresu v tabuľke arduino\_online, slúži ako jednoznačné prepojenie medzi tabuľkou modelu a tabuľkou arduino\_online.
- **experiment** – v tejto tejto premennej je zapísaný typ experimentu, napríklad PID.
- **START** – oznamuje zadanie nového experimentu vo webovom rozhraní používateľom.
- **Timer** a **T** – obe kolónky slúžia na zapisovanie času, v niektorých experimentoch sú potrebné až dva časové údaje, primárne je však používaná kolónka Timer.
- **Kp, Ti, Td** – slúžia výhradne len pre zapísanie hodnôt PID regulátora.
- **r1 až r6** – podľa experimentu slúžia buď na zapísanie hodnôt referenčnej krivky, alebo hodnôt krivky výkonu motorčeka.

Teoreticky by táto tabuľka nemusela existovať a všetky premenné by sa mohli nachádzať v tabuľke aero\_shield. Rozdelenie týchto tabuliek však prináša lepší prehľad pri vývoji, poskytuje lepšiu modularitu a zjednodušuje pridávanie ďalších modelov, nakoľko sa už nemusia upravovať existujúce tabuľky a nepríde tak k zbytočným chybám v programoch už pridaných Shieldov.

### 4.1.3 Tabuľky zaznamenaných dát

Tieto tabuľky majú v sebe premenné pre všetky namerané dáta z experimentov jednotlivých modelov. Pre lepšie pochopenie môžete vidieť štruktúru tejto tabuľky pre AeroShield na Obr. 4.3. Na rozdiel od tabuľky **arduino\_online**, ktorá je spoločná pre všetky Arduina a každé zariadenie v nej má predelený vlastný riadok, alebo od tabuľky modelov, kde má každý model svoju vlastnú spoločnú tabuľku. Tabuľka nameraných dát patrí celá len konkrétnemu Arduinu. Pre túto tabuľku platí, že každá iterácia experimentu je zapísaná do nového riadku a nachádzajú sa v nej rádovo tisíce riadkov. Tabuľka je na začiatku každého nového experimentu premazaná a nachádza sa v nej len posledný experiment.

#	Nazov	Datový typ	Dĺžka/Množst...	Unsigned	Null	Zero-filled	Default	Komentár	Porovnanie	Výraz	Virtuality	SRID	Invisi...
1	r	FLOAT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
2	y	FLOAT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>
3	u	FLOAT		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL						<input type="checkbox"/>

Obr. 4.3: Prehľad tabuľky nameraných dát pre AeroShield.

Kolónky v tejto tabuľke sú pevne dané:

- **r** – referencia v stupňoch,
- **y** – nameraný uhol naklonenia kyvadla v stupňoch,
- **u** – prúd privádzaný na motor v Ampéroch.

Táto tabuľka slúži len pre vykreslenie aktuálneho experimentu pomocou chartjs.

## 4.2 Štruktúra procesu celého systému

V nasledujúcich podkapitolách vysvetlím jednotlivé procesy, ktoré sa dejú na pozadí všetkých krokov tejto práce. Pre lepšie pochopenie budú v tejto kapitole uvedené detaily z vývojových diagramov, ktoré sú uvedené v jednotlivých prílohách:

- Arduino IDE A.1 – Arduino IDE kód.
- arduino\_online.php A.2 – hlavný komunikačný PHP skript
- arduino\_online\_experiment\_data.php A.3 – PHP komunikačný skript pre Arduino na získanie parametrov experimentov.
- AeroShield\_update.php A.4 – PHP skript na vloženie parametrov experimentov do DB.

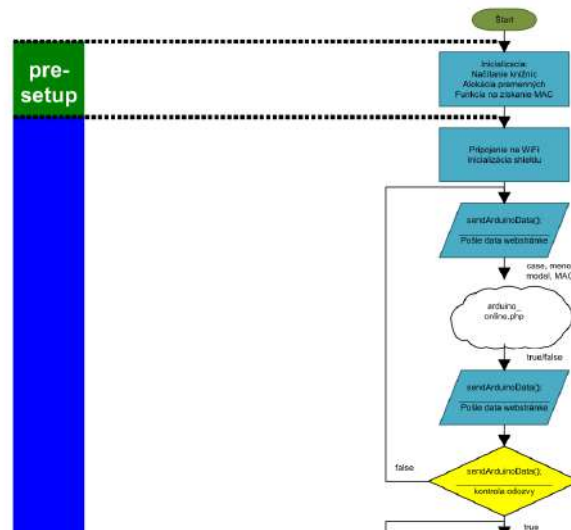
### 4.2.1 Štart a registrácia Arduina na server

Po tom, čo užívateľ upraví v kóde štyri premenné:

- **ssid**,
- **password**,
- **arduinoName**,
- **shield\_password**,

nahrá program na Arduino a spustí ho. Hneď po spustení otvorí aplikáciu **SerialMonitor.exe** alebo **SerialMonitor.app** podľa jeho operačného systému, nastaví správny komunikačný kanál COM a stlačí štart. Aplikáciu nechá bežať na pozadí, počas celého procesu s ňou už nič ďalšie robiť nebude.

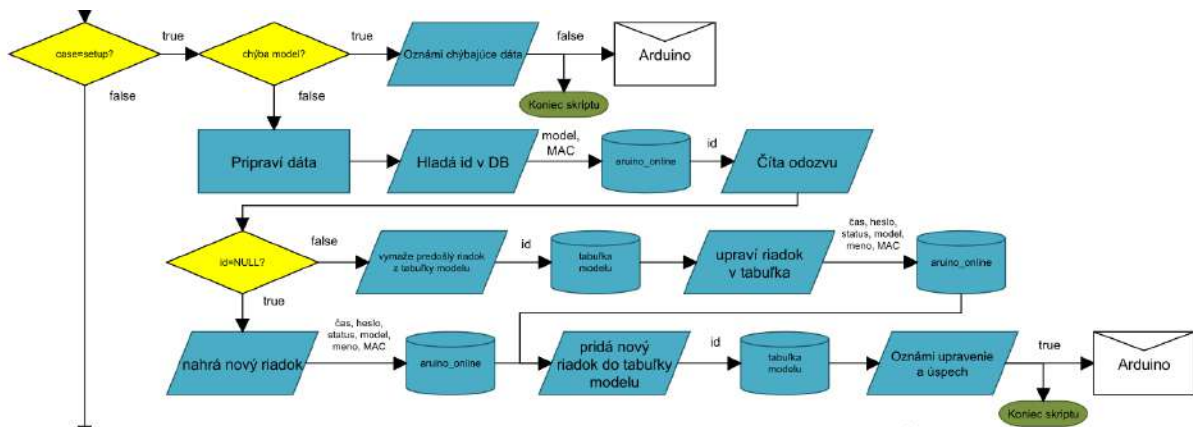
Arduino po spustení alokuje pamäť pre všetky premenné, načíta knižnice a inicializuje sa. Následne vstúpi do funkcie **setup**, kde sa ako prvé pripojí na WiFi, pomocou údajov, ktoré jej dal používateľ pred



Obr. 4.4: Detail grafu pre Arduino časť setup 1.

nahráním, a zároveň je získaná MAC adresa z Arduina, ktorá je neskôr posielaná na web server. Na Obr. 4.4 môžete vidieť, že následne Arduino prejde do slučky registrácie Arduina na server.

Dáta sú poslané na PHP skript stránky `arduino_online.php`. Tieto dáta obsahujú užívateľom zadané meno, model, adresu MAC a premennú `case`, ktorá v tomto prípade nesie hodnotu **setup**. Na Obr. 4.5 môžete vyčítať, že PHP skript po prečítaní dát vstúpi do časti kódu registrácie Arduina.



Obr. 4.5: Detail grafu pre PHP časť setup 1.

Dáta sú overované a ako prvé sa zisťuje, či už sa daná adresa MAC nenachádza v tabuľke `arduino_online`.

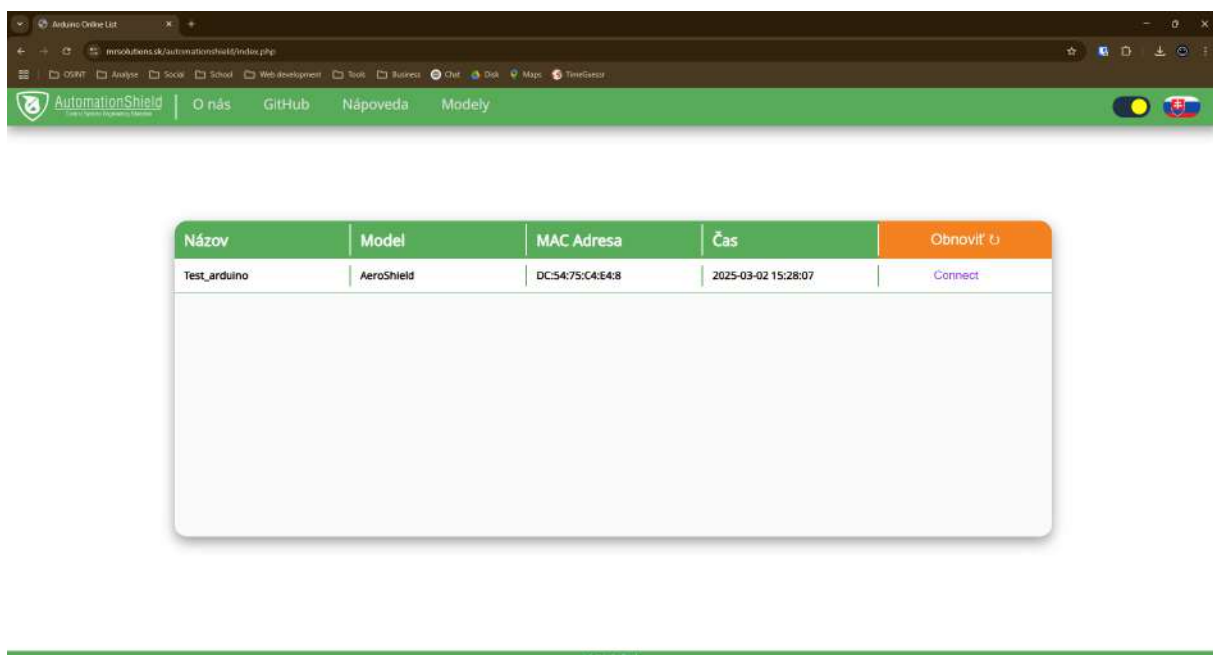
- Ak sa táto MAC adresa už v tabuľke nachádza, znamená to, že na konkrétnom Arduine už v minulosti prebiehal experiment a je potrebné vymazať riadok tohoto Arduina z tabuľky predošlého modelu, ktorý sa nachádzal na tomto Arduine, napríklad z tabuľky `AeroShield`. Následne je upravená tabuľka `aero_shield` novými aktuálnymi dátami.
- Ak sa MAC adresa tohoto Arduina v tabuľke `arduino_online` nenachádza, nemusí sa vymazávať riadok zo žiadnej tabuľky modelu a treba len nahráť nový riadok do tabuľky `arduino_online` s dátami pre toto Arduino.

Po vykonaní jednej z týchto dvoch možností sa nahrá nový riadok do tabuľky aktuálneho modelu. Pre lepšie pochopenie uvediem príklad:

1. Daná MAC adresa bola nájdená v tabuľke `arduino_online` a jej starý model bol **FurutaShield**.
2. Z tabuľky **FurutaShield** je vymazaný riadok, kde sa nachádza táto MAC adresa.
3. Tabuľka `arduino_online` je upravená novými aktuálnymi dátami a teda hlavne novým modelom **AeroShield**.
4. Do tabuľky modelu **AeroShield** je nahraný nový riadok pre túto MAC adresu.
5. PHP skript odpovedá Arduinu, že registrácia bola úspešná.

Tento proces je rovnaký aj v prípade, že starý a nový model sú rovnaké. Celý proces trvá len pár sekúnd. Ak prebehne úspešne a nemusí sa opakovať, tak sa v tabuľke dostupných Arduín zobrazí zaregistrované Arduino, ako vidieť na Obr. 4.6.

Tento proces prebieha veľmi rýchlo a spoľahlivo, no pri nestabilnom WiFi signále sa môže Arduino nepodať zaregistrovať. V tomto prípade treba nájsť iné WiFi pripojenie.



The screenshot shows a web browser window with the URL `arduinoonline.sk/automationshield/index.php`. The page has a green header with the 'AutomationShield' logo and navigation links. Below the header is a table with the following data:

Názov	Model	MAC Adresa	Čas	Obnoviť ↻
Test_arduino	AeroShield	DC:54:75:C4:E4:8	2025-03-02 15:28:07	Connect

Obr. 4.6: Tabuľka dostupných Arduín.

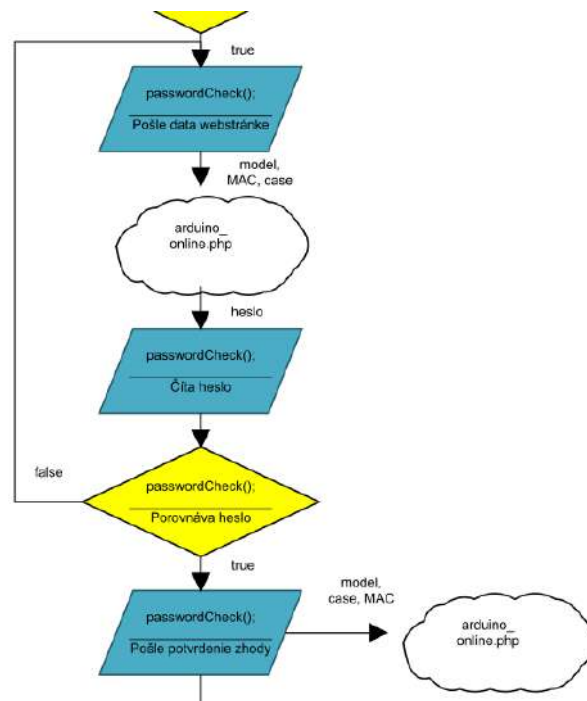
## 4.2.2 Prihlásenie používateľa cez webstránku

Arduino je stále v tomto momente ešte stále vo funkcii **setup**, no prejde do cyklu overovania zadaného hesla **password\_check**, ktorý je znázornený na Obr. 4.7.

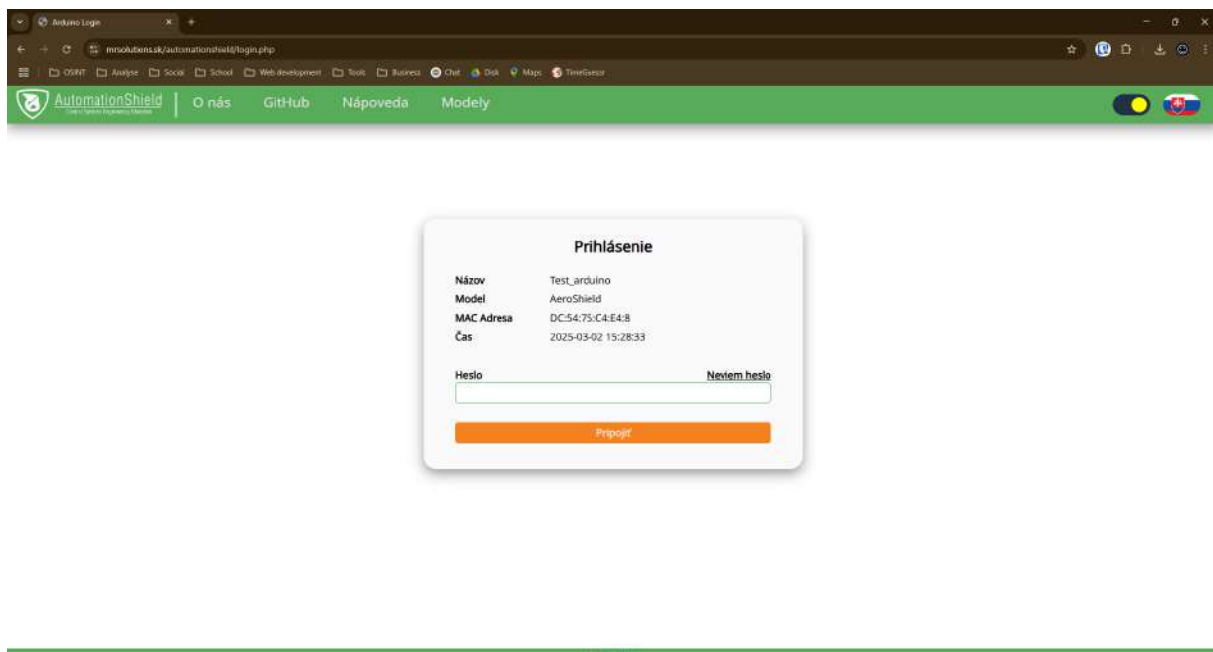
V tejto časti posielajú Arduino dáta ako indikátory na rovnaký PHP skript **arduino\_online.php**. Dotazuje sa tak na to, aké heslo používateľ zadal v prihlasovacom formulári, na ktorý sa dostane kliknutím na tlačidlo **Connect** z Obr. 4.6. Tento formulár môžete vidieť na Obr. 4.8. Zadaním hesla sa namiesto formulára objaví načítavacia animácia z Obr. 4.9, pretože formulár čaká na odpoveď od Arduina.

Medzi časom, čo užívateľ vykonáva prihlasovanie, v Arduine prebieha vyššie spomínaný cyklus **password\_check**. Ten znovu kontaktuje PHP skript **arduino\_online.php**, ktorý teraz číta, aké heslo užívateľ zadal, a posielajú ho späť Arduinu na kontrolu, ako vidieť na Obr. 4.10.

Skript pomocou MAC adresy a modelu získava heslo, ktoré používateľ zadal v prihlasovacom formulári, toto heslo následne posielajú späť Arduinu, ktoré ho porovnáva s heslom, ktoré používateľ nastavil pred nahrať kódu do Arduina. V prípade, že sa heslá nezhodujú, PHP skript prihlasovacej stránky prečíta z tabuľky nezmenenú hodnotu premennej **status** a teda hodnotu **disconnected**. Používateľovi sa následne



Obr. 4.7: Detail grafu pre Arduino časť setup 2.



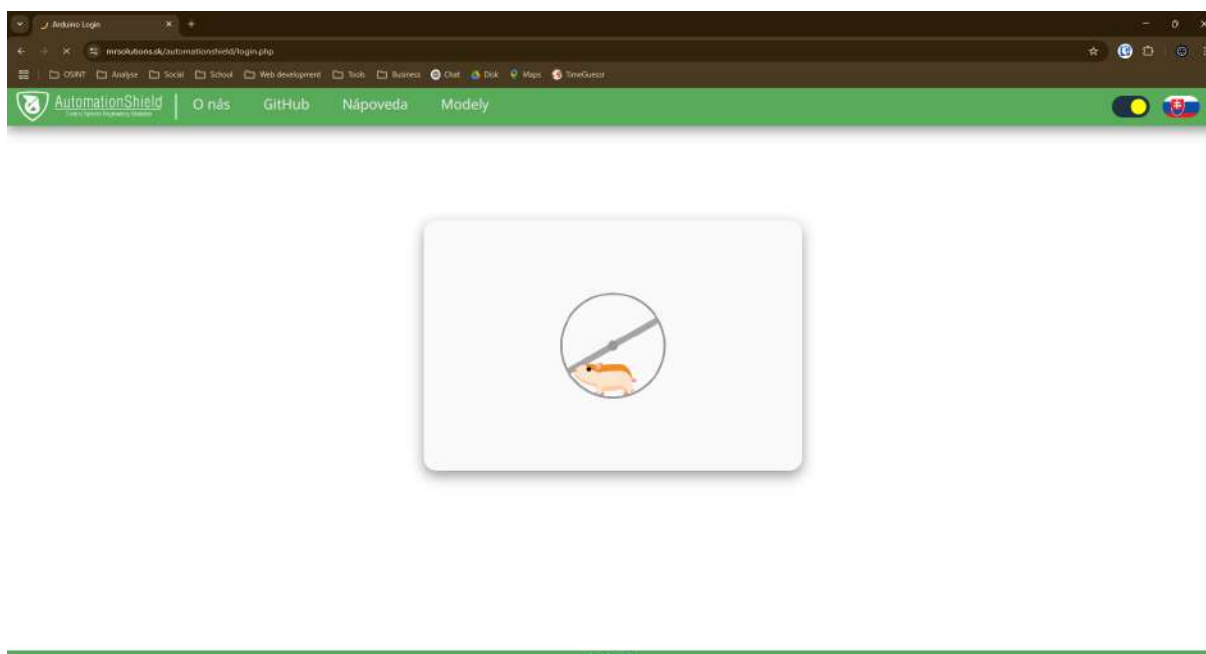
Obr. 4.8: Prihlasovací formulár.

opäť zobrazí prihlasovací formulár s hláškou, že sa heslo nezhoduje, ako vidieť na Obr. 4.11 a môže zadať heslo znovu.

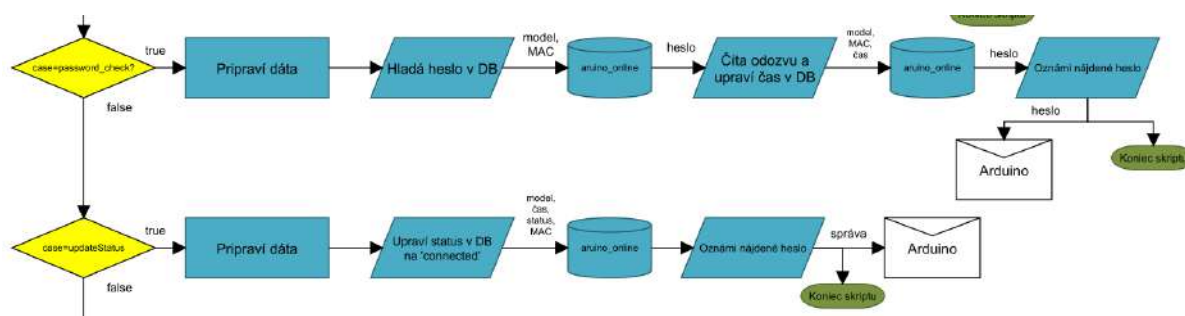
V prípade, že sa heslá zhodujú, Arduino odošle novú požiadavku na stránku **arduino\_online.php**, kde sa teraz skript pomocou prijatej premennej **case** dostane do cyklu oznámenia zhody hesiel, ktorý je vidieť na Obr. 4.10.

PHP skript teraz upraví premennú **status** na **connected**. Prihlasovací formulár v tomto prípade





Obr. 4.9: Prihlasovací formulár čakajúci na odpoveď.



Obr. 4.10: Detail grafu pre PHP skript časť setup 2.

### Prihlásenie

<b>Názov</b>	Test_arduino
<b>Model</b>	AeroShield
<b>MAC Adresa</b>	DC:54:75:C4:E4:8
<b>Čas</b>	2025-03-06 16:35:19

**Heslo**
**Neviem heslo**

Password doesn't match

Pripojiť

Obr. 4.11: Prihlasovací formulár po nezhode hesiel.



prečíta premennú **status**, a zistí, že heslá sa zhodujú, to vyvolá presmerovanie užívateľa na GUI stránku konkrétneho modelu Shieldu.

### 4.2.3 Experimentovanie pomocou GUI

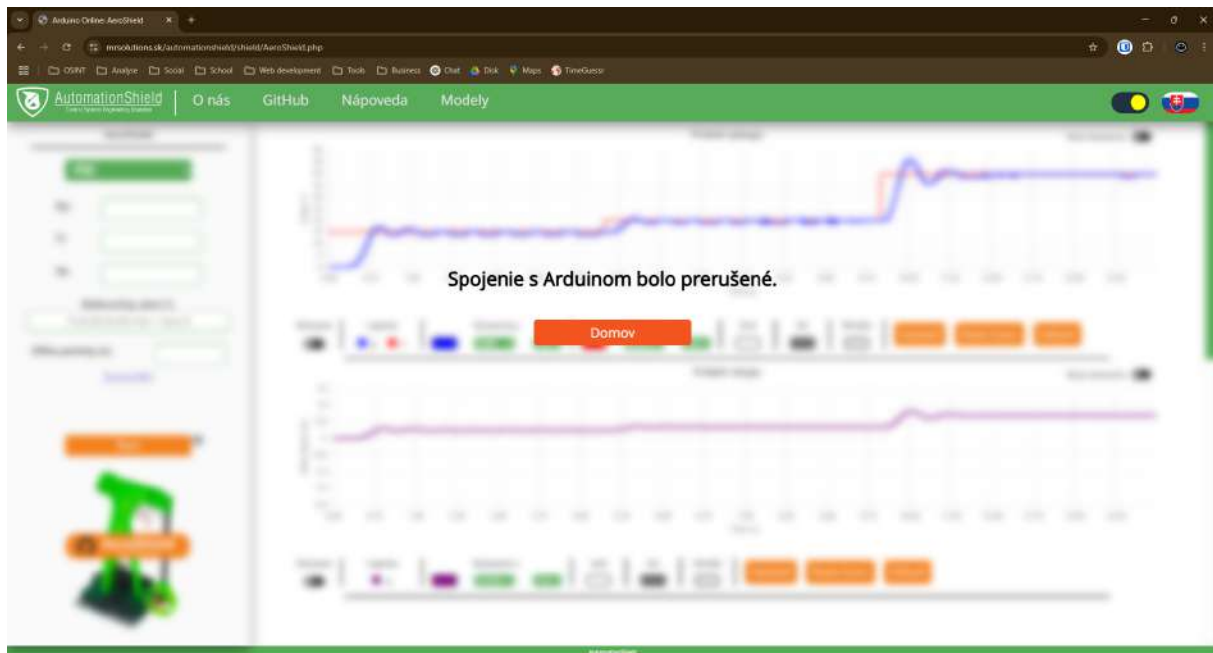
Do tohto kroku je postup pre **každý model rovnký** a jediná vec, ktorá sa mení, je hodnota premennej **model**. Model, ale nemení postup registrácie a prihlásenia. Časť experimentovania je robená tak, aby postupy zadávania a čítania experimentov boli rovnaké, ale nebudú identické. Ide o to, že každý Shield potrebuje iné parametre na vykonanie experimentov a zároveň sú merané iné parametre pri ich priebehu. Pri niektorých modeloch sa napríklad meria len jeden uhol, ako pri AeroShielde. Pri niektorých sa meria viac parametrov, ako napríklad pri FurutaShielde.

#### GUI stránka modelu

GUI má rovnakú funkčnosť a podobný výzor pre všetky modely. Na komunikáciu používajú rovnaké princípy a všetky formuláre fungujú na rovnakom princípe. Rozdiel je prakticky len v dátach, ktoré sú vykresľované v grafoch alebo zadávané vo formulároch, keďže sa môže jednať o iné jednotky alebo iný typ dát. Všetky dáta sa odosielaajú a prijímajú pomocou AJAXu, PHP a JavaScriptu, ktorých vývojový diagram pre GUI je nepotrebný, nakoľko nejde o jeden kontinuálny proces ako pri Arduino IDE kóde alebo stránok `arduino_online.php` či `arduino_experiment_data.php`, ktorý si ukážeme v nasledujúcich podkapitolách. Jeden vývojový diagram som však uviedol a to `AeroShield_update.php` A.4, ktorý slúži na vkladanie parametrov z GUI formuláru do DB. Jeho princíp je jednoduchý a nepotrebný pre túto kapitolu. Všetky stránky pre GUI nesú meno podľa modelu, takže samotné GUI má stránku napríklad `AeroShield.php`, no spolu s ňou pracuje niekoľko iných stránok a to:

- **AeroShield\_session.php** - tento skript sa načíta ešte pred načítaním stránky GUI. Nesie v sebe **id** pre dané Arduino z tabuľky **arduino\_online** a zároveň vytvára tabuľku nameraných dát pre dané Arduino (tabuľku pomenuje podľa MAC adresy Arduina).
- **AeroShield.js** - skript v ktorom sa nachádzajú všetky funkcie pre daný model od zobrazovacej logiky HTML elementov a výstražných správ, volanie a odosielanie dát pomocou AJAXu pre rôzne funkcie stránky GUI, zobrazovanie a upravovanie vkreslovacích grafov nameraných dát a v podstate každý logická funkcia je volaná v tomto skripte.
- **AeroShield\_fetch.php** - volané java skriptom pomocou AJAXu. Tento skript získava namerané dáta z tabuľky nameraných dát (tabuľka ktorá je pomenovaná podľa MAC adresy Arduina).
- **AeroShield\_DataCheck.php** - volané java skriptom pomocou AJAXu, ide o kontrolu pripojenia Shieldu. Ak sa spojenie preruší, stránka GUI ukáže správu o prerušení a nedovolí ďalšie interakcie s GUI, ako vidíte na Obr. 4.12. Prerušenie vzniká v momente, kedy v tabuľke **arduino\_online** nebola upravená kolónka **datetime** posledných 15 sekúnd a zároveň neprebíha žiadny experiment, keďže počas trvania experimentu sa táto kolónka neupravuje.
- **AeroShield\_update.php** - volané java skriptom pomocou AJAXu - slúži na vkladanie dát na vykonanie experimentu z GUI formuláru do tabuľky modelu (tabuľka pomenovaná modelom na príklad tabuľka `AeroShield`, vývojový diagram A.4).
- **AeroShield-dark.css** a **AeroShield-light.css** - kaskádové štýly pre dané GUI. Kaskádové štýly a globálne funkcie celej webstránky sú preberané v ďalšej kapitole, GUI stránka má však viac kaskádových štýlov, okrem globálneho kaskádového štýlu, je na GUI nahraný aj globálny GUI kaskádový štýl **A\_automation-shield-workbench-dark.css** a **A\_automation-shield-workbench-light.css** a už spomínané kaskádové štýly priamo pre daný model. Tieto štýly sú prepínané pomocou tlačidla, ktoré si preberieme v ďalšej kapitole. Rozdelenie kaskádových štýlov týmto spôsobom zaručuje lepšiu modularitu.

Všetky vymenované stránky sú navrhnuté tak, aby sa dali jednoducho upraviť a použiť znovu aj pre ostatné modely, sú riadne zakomentované a robené modálne, tak ako aj zvyšok práce. Pridanie nového modelu je teda len otázkou recyklovania už existujúcich kódov.



Obr. 4.12: GUI oznámenie prerušeného spojenia s Arduino.

## Kontrola experimentu

Arduino po úspešnom prihlásení vstúpi do funkcie **loop**. Táto funkcia v sebe obsahuje 3 rôzne funkcie a to:

- **experimentCall.**
- **experimentBegin.**
- **sendExperimentData.**

Podľa Obr. 4.13 je vidieť, že Arduino je najprv v nekonečnej slučke, kde sa pýta stránky `arduino_online.php`, či používateľ zadal nejaký experiment. Toto dotazovanie prebieha každé dve sekundy do nekonečna, až pokiaľ nepríde pozitívna odpoveď.

Dotaz v PHP skripte prebieha podobne ako predošlé dotazy zo stránky `arduino_online.php`. Na Obr. 4.14 možno vidieť, ako sa tentokrát vďaka premennej **case** skript dostal do časti, kde skript najprv nájde **id** pre konkrétne Arduino v tabuľke `arduino_online` a následne vďaka tomuto nájdenému **id** skontroluje hodnotu premennej **START** z tabuľky modelu. Podľa toho, či je hodnota **START** alebo **STOP** posíla odpoveď Arduino.

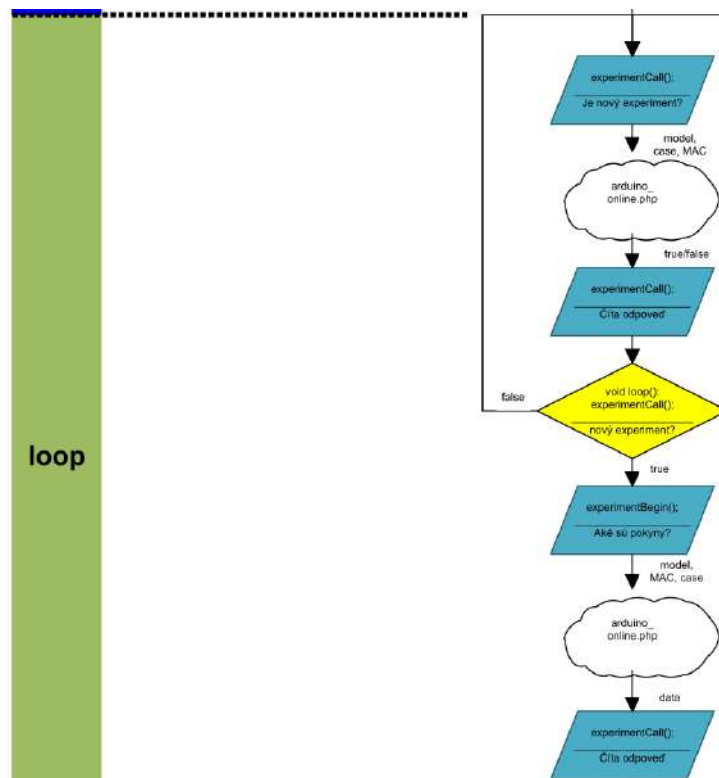
Ak užívateľ zadal na GUI stránke experiment, tak sa proces presunie na funkciu **experimentBegin**, kde podľa Obr. 4.13 možno vidieť, že Arduino sa tentokrát nedotazuje stránky `arduino_online.php`, ale stránky `arduino_online_experiment_data.php`. Celý sekvenčný diagram sa nachádza vo výpise z kódu A.3. Na Obr. 4.15 môžete vidieť, že tento PHP skript si podobne ako v skripte `arduino_online.php` najprv musí nájsť **id** z tabuľky `arduino_online`.

Následne skript vojde do cyklu, kde porovnáva, aký model na sebe toto Arduino má. Ak je to napríklad AeroShield (Obr. 4.16) tak ako prvé zistí, aký experiment bol zadáný. To je potrebné k tomu, aby skript vedel, ktoré dáta má následne z databázy načítať.

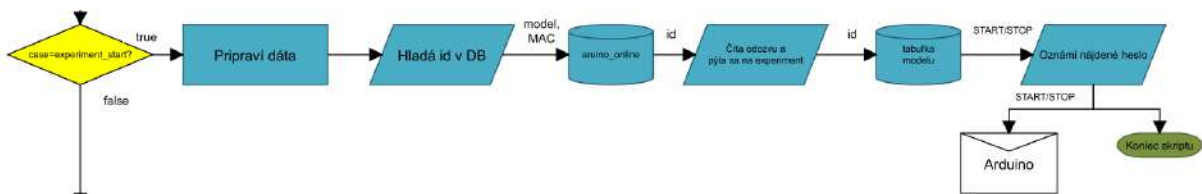
Na Obr. 4.17 je cyklus pre **PID** pri modeli **AeroShield**. Skript nájde tie správne parametre a odosiela ich na Arduino.

Arduino dostáva dáta v podobe stringu, z ktorých vyčíta najprv, o aký experiment sa jedná. Následne podľa neho nájde ten správny cyklus, ako vidno na Obr. 4.18.

Cyklus spracuje dáta a prevedie experiment. Po ukončení experimentu, či už predčasným kvôli veľkému uhlu naklonenia alebo splnením časovej podmienky, sa všetky parametre reštartujú na východiskové



Obr. 4.13: Detail grafu pre Arduino časť experiment 1.



Obr. 4.14: Detail grafu pre PHP časť experiment 1.

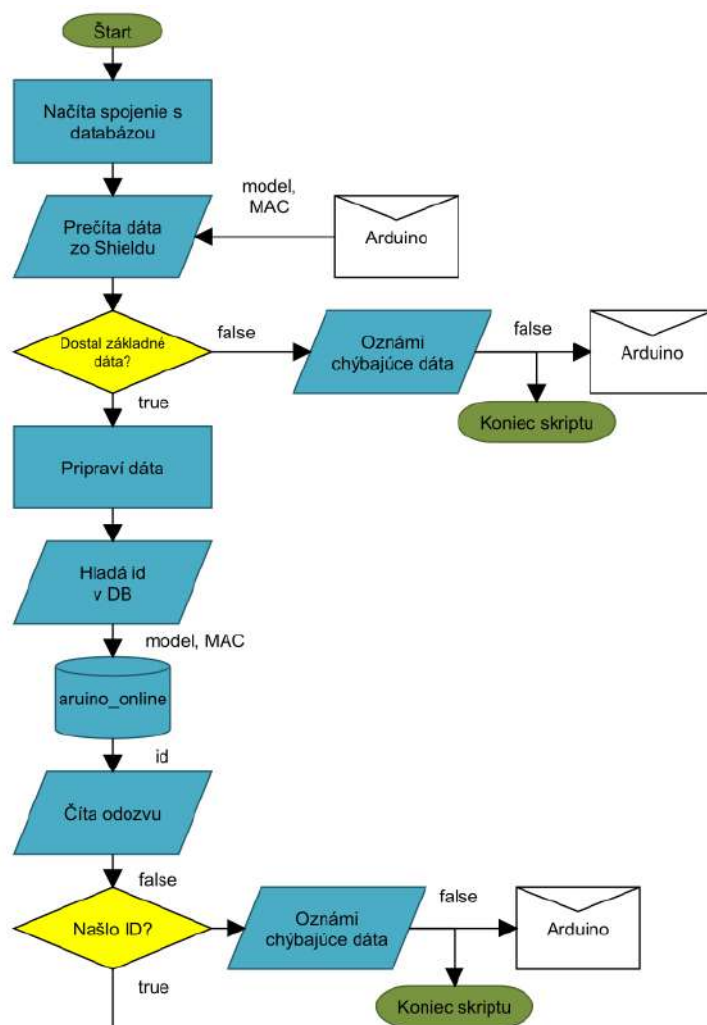
hodnoty.

Počas experimentu Python aplikácia SerialMonitor.exe alebo .app číta všetky merané parametre, ktoré sú iteratívne vypisované do serialovej komunikácie a posíla ich do databázy nameraných dát, ktorá je pomenovaná podľa adresy MAC. Python aplikácia zároveň priebežne ukladá všetky namerané dáta a po tom, čo prečíta, že experiment bol ukončený, vygeneruje **experiment\_data.mat** súbor, kde sú všetky tieto dáta uložené. Tento vygenerovaný súbor sa nachádza vždy v rovnakom priečinku, ako .exe alebo .app aplikácia a jeho hodnoty sú prepísané každým novým experimentom. Pokiaľ si chce užívateľ tieto dáta uložiť, stačí, že súbor premenuje alebo ho presunie do iného priečinku v počítači.

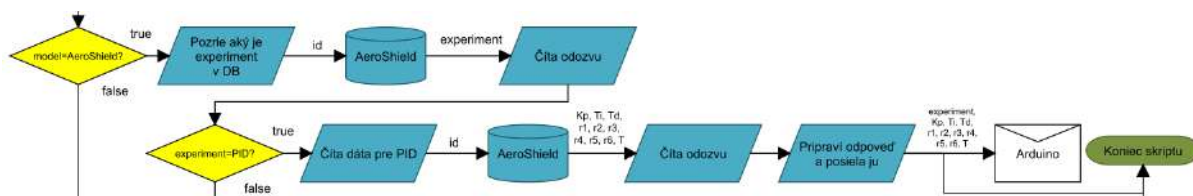
Arduino po vykonaní experimentu a reštartovaní parametrov prejde do funkcie **sendExperiment-Data**, ktorá posíla správu o ukončení experimentu, čo je znázornené na Obr. 4.19.

Na Obr. 4.20 je spracovanie tohto dotazu na stránke **arduino\_online.php** a jedná sa o poslednú funkciu celého PHP skriptu.

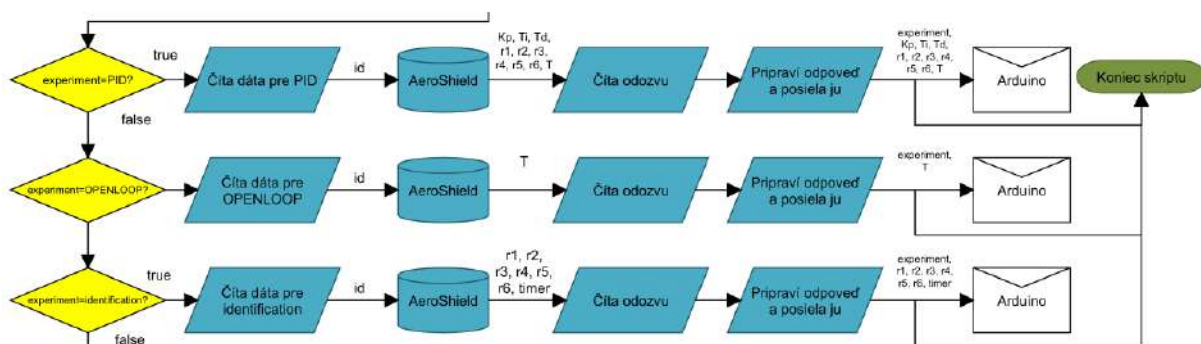
Arduino teraz prejde naspäť na začiatok funkcie **loop** a znovu sa dotazuje na to, či používateľ zadal nejaký nový experiment.



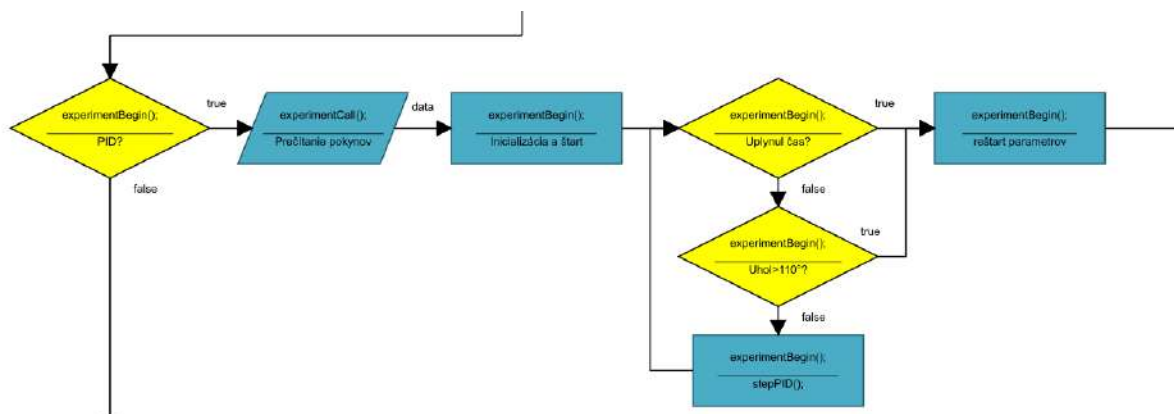
Obr. 4.15: Detail grafu pre PHP časť experiment 2.



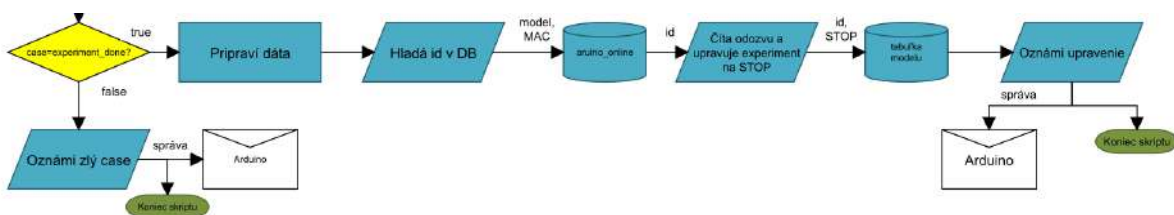
Obr. 4.16: Detail grafu pre PHP časť experiment 3.



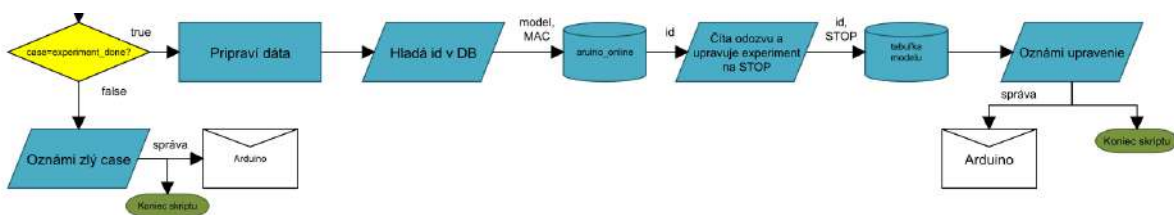
Obr. 4.17: Detail grafu pre PHP časť experiment 4.



Obr. 4.18: Detail grafu pre Arduino časť experiment 2.



Obr. 4.19: Detail grafu pre Arduino časť ukončenie.



Obr. 4.20: Detail grafu pre PHP časť experiment 5.

## 5 Webstránka

Stránka ako celok má mnoho rôznych skriptov, kódov alebo iných zdrojov, ktoré sa v nej vyskytujú. Stránky môžeme rozdeliť do dvoch hlavných kategórií a to **prezentačné** a **pomocné**. Prezentačné môžeme ďalej rozdeliť na podkategórie **experimentné** a **podporné**.

Experimentné sme si rozobrali v predošlých kapitolách, sú to:

- **index.php** – domovská stránka.
- **login.php** – prihlásenie do Arduina.
- **model.php** (napr. AeroShield.php) – GUI pre experimenty.

Podporné stránky:

- **about-us.php** – informačná stránka o AutomationShield iniciatíve.
- **how-to.php** – stránka plná návodou na Arduino, AutomationShield a AutomationShield online.
- **models.php** – stránka pre stiahnutie súborov pre AutomationShield online.

Všetky vymenované stránky majú svoje pomocné stránky, môže sa jednať o AJAX, zobrazovacie skripty a podobne. Pre celú stránku, ale platia určité pravidlá, ktoré si teraz prejdeme.

### 5.1 Kaskádové štýly

Kaskádové štýly sú grafický kódovací jazyk pre HTML, je to základná časť každej webstránky na svete. Aj napriek tomu, že je tento jazyk používaný dodnes a nemá žiadnu alternatívu, má svoje nedostatky, ktoré môžu byť doplnené napríklad JavaScriptom. Je veľa spôsobov, ako nastaviť systém pre kaskádové štýly. Celá stránka môže mať napríklad len jeden globálny kaskádový štýl, napríklad style.css, ktorý je použitý na každej stránke a nachádzajú sa v ňom všetky grafické elementy. Ďalšou možnosťou je nepoužívať externý súbor a štýly písať na každej stránke do hlavičky, alebo priamo do HTML atribútov. Tieto riešenia sú však veľmi náročné na úpravu a veľmi zlé na škálovanie stránky a znovu použitie. Keď si developer nedá pozor, ľahko pomenuje viac elementov rovnako a štýly sa potom medzi sebou bijú [27]. Pre moju webstránku som teda navrhol tieto pravidlá:

- Každý štýl má svetlý a tmavý variant, ktoré sú prepínané JavaScriptom podľa požiadavky používateľa prepínačom na pravej strane topbaru.
- Globálny kaskádový štýl, v ktorom sa nachádzajú prvky, ktoré sú na každej stránke, ako napríklad definícia topbaru, footbaru a celého tela stránok. Tento štýl je len jeden súbor a je priložený ku každej webstránke, ako vidieť vo výpise z kódu 5.1 pomenovaný **A\_light-mode\_style.css** a **A\_dark-mode\_style.css**.
- Lokálny kaskádový štýl - tento štýl obsahuje grafické kódy priamo pre danú stránku. Vždy sú nazvané podľa stránky, na ktorej sa nachádzajú, nasledované pomlčkou a slovom dark/light, napríklad **index-dark.css** a **index-light.css**, príklad rozdielu denného a nočného štýlu je vidieť na Obr. 5.1.
- GUI stránky ako jediné zdieľajú spoločné štýly **A\_automation-shield-workbench-light.css** a **A\_automation-shield-workbench-dark.css**.

Výpis z kódu 5.1: Príklad hlavičky HTML kódu.

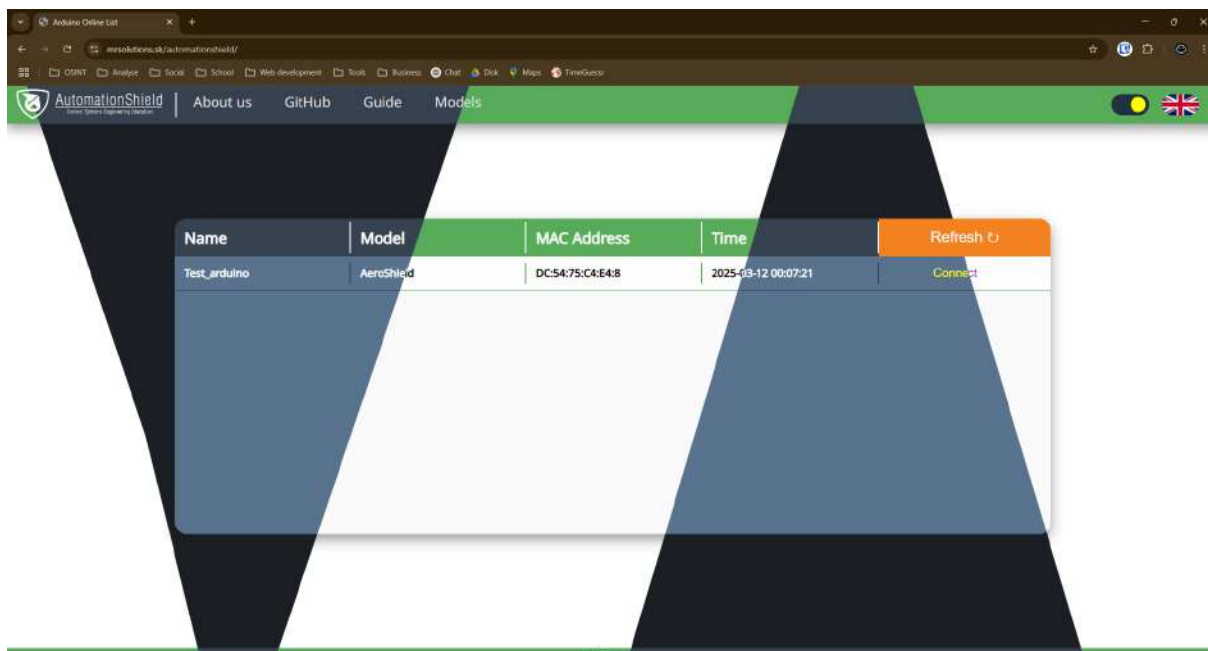
```
<head>
<!-- meta -->
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Arduino Online List</title>

<!-- Global CSS + fonts -->
<link id="day-css" rel="stylesheet" href="A_light-mode-style.css"
type="text/css">
<link id="night-css" rel="stylesheet" href="A_dark-mode-style.css"
type="text/css" disabled>
<link href='https://fonts.googleapis.com/css?family=Noto+Sans'
rel='stylesheet'>

<!-- Page-Specific CSS -->
<link id="page-day-css" rel="stylesheet" href="index-light.css"
type="text/css">
<link id="page-night-css" rel="stylesheet" href="index-dark.css"
type="text/css" disabled>

<!-- JS -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script src="A_script.js"></script> <!-- Global Script -->
<script src="index.js"></script> <!-- Page-Specific Script -->
</head>
```



Obr. 5.1: Príklad rozdielu denného a nočného štýlu.

Týmto systémom je zaručené, že sa nebudú jednotlivé štýly z rôznych stránok dostávať do konfliktov. Zároveň sa v štýloch ľahšie orientuje a tým sa aj ľahšie modifikujú a pridávanie nových stránok je jednoduché.

## 5.2 JavaScript kód

Rovnaké pravidlá, aké som zvolil pre kaskádové štýly, platia aj pre skripty v JS. JavaScript je vo webdevelopmente použitý hlavne na miestach, kde samotné PHP a CSS nestačia. Jedná sa napríklad o rôzne zobrazovacie logiky, časovania či prácu s AJAXom [28]. V mojej práci majú skripty teda tieto pravidlá:

- Globálny JavaScript - taktiež je možné ho vidieť vo výpise z kódu 5.1. Obsahuje najmä logiku na prepínanie kaskádových štýlov medzi denným a nočným režimom, spravovanie Cookies a logika prekladania stránky, ktorú si prejdeme neskôr.
- Lokálny skript - skripty pre jednotlivé stránky, väčšinou sa jedná o AJAX alebo zobrazovaciu logiku na jednotlivých stránkach. Nie vždy je lokálny skript vôbec potrebný, napríklad na stránke models.php sa v lokálnom skripte nič nenachádza.

### 5.2.1 Cookies

Cookies sú malé dátové informácie uložené priamo v zariadení užívateľa [29]. Cookies samotné sú veľmi zložitý právny aspekt webdevelopmentu. Ich oficiálne rozdelenia sú obširné a idú do veľkých hĺbok. Nakoľko táto práca nie je na túto tému vôbec zameraná, tak ďalej opisujem len potrebné informácie, ktoré som uznal za vhodné pre pochopenie mojej webstránky.

Pri cookies sa často jedná hlavne o logické hodnoty pre skripty na pozadí, ako napríklad aký jazyk si užívateľ zvolil a podobne. Takéto cookies bývajú na stránke uložené neobmedzený časový interval. Iný typ cookies sú takzvané session cookies, ktoré fungujú rovnako ako normálne cookies, ale sú uložené len určitý čas, a to dovtedy, kým nie je prehliadač zatvorený [30].

Cookies podliehajú zákonom a developer si musí vyžiadať od užívateľa povolenie na ich používanie. Zákon o oboznámení a súhlase používania cookies podlieha hlavne týmto pravidlám [31]:

- Aké cookies stránka používa.
- Prečo tieto cookies používa.
- Kto má ku cookies prístup.
- Možnosti nastavenia cookies.

Zákon vyžaduje, aby mohol užívateľ stránku používať aj keď odmietne niektoré cookies a užívateľ by mal mať možnosť vybrať si. Na Obr. 5.2 vidíte moju správu o nastaveniach cookies. Vysvetľujem v nej aké cookies používam a prečo ich používam. Zároveň podávam informáciu o tom, ako stránka s cookies narába a teda, že sa nezdieľajú s treťou stranou ani nezbierajú na štatistické alebo iné účely.

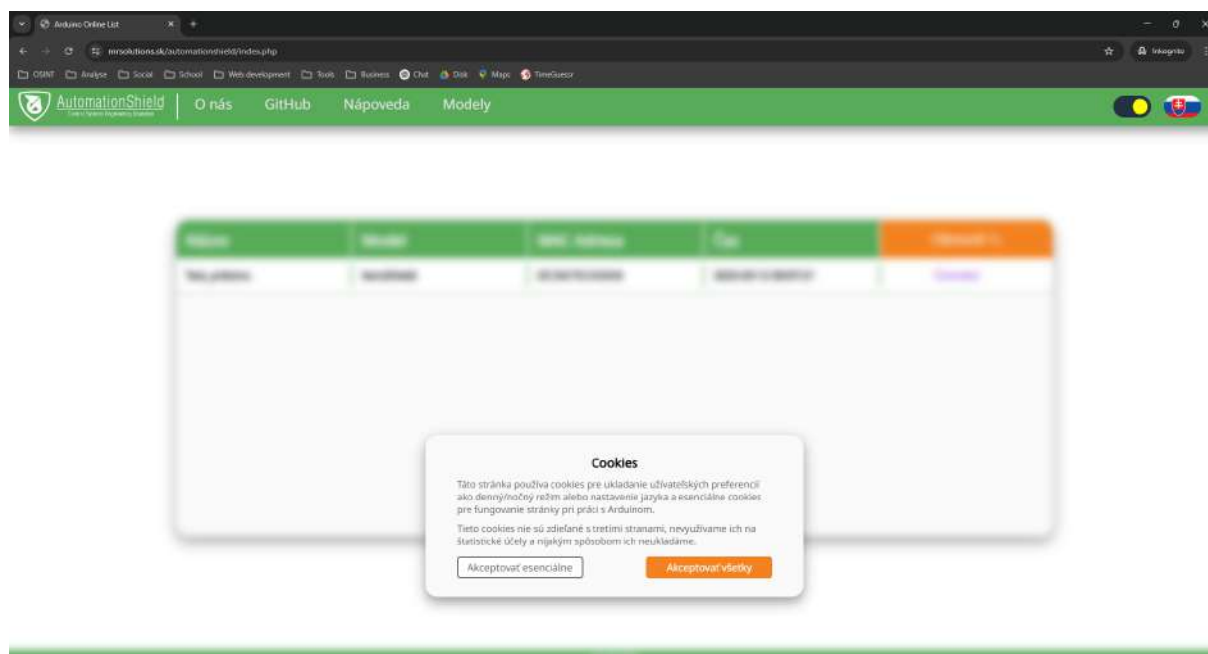
Zvolené nastavenie je na stránke uložené na dobu neurčitú a užívateľ si môže nastavenia cookies otvoriť kliknutím na **Nastavenia cookies**, na spodnej lište, ako vidno na Obr. 5.3.

Počas zobrazenia cookies oznamu má užívateľ stále možnosť prepínať jazyky, denný a nočný režim či kliknúť na odkazy v topbare, nebude však vedieť kliknúť na žiadny iný obsah konkrétnych stránok. Oznamu sa však nezabaví, keďže v rámci zaručenia zobrazenia oznamu cookies, je celý html kód pre tento oznam vpísaný do globálneho skriptu. Tento oznam tak užívateľ dostane nezávisle od toho, na akú stránku sa dostane.

## 5.3 Preklady

Logika prekladov sa vo webdevelopmente robí rôzne, takmer vždy sa však používajú indikátory v HTML, ktoré ukazujú, kde má byť aký text. Takéto indikátory som použil aj ja. V každom HTML elemnte, kde chcem zobraziť nejaký text, napíšem atribút "**data-txt=XXX**", kde XXX je indikátor daného textu a data-txt je atribút, ktorý hľadá globálny JavaScript. JavaScript potom nájde všetky požadované elementy,





Obr. 5.2: Ohlásenie cookies používateľovi.

prečíta ich hodnoty ako napríklad `'data-txt=txtAvailable'` a následne v súbore `translations.json`, ktorého časť je ukázaná v obrázku 5.4 nájde správny text pre `"txtAvailable"` [32].

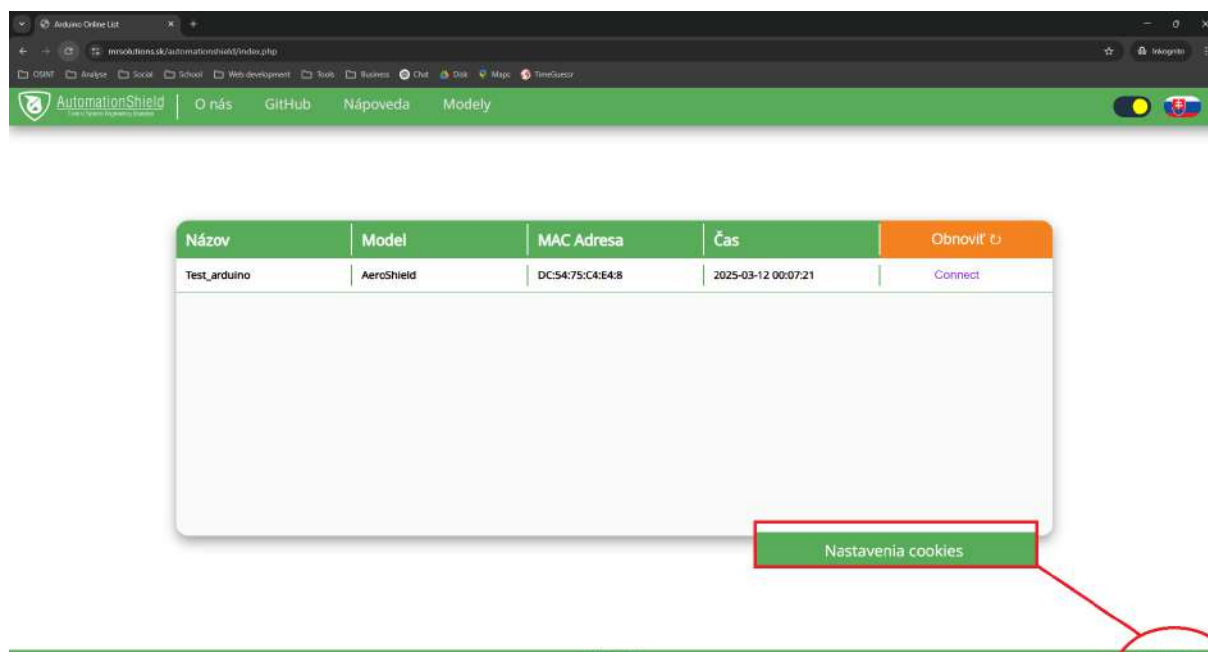
Skript zvolí text podľa toho, aký jazyk si užívateľ vyberie. Tento systém je veľmi dobre škálovateľný, keďže nové texty sa ľahko prekládajú a existujúce sa ľahko upravujú, ale je dokonca pomerne jednoduché pridať kompletne nový jazyk. Stačí len pridať novú možnosť do HTML ponuky jazykov v topbare a potom pridať daný jazyk do všetkých prekladov. Tento proces je síce zdĺhavý, ale pridanie nového jazyka je otázkou hodiny.

## 5.4 Iné

V rámci webstránky som zaviedol ešte niekoľko pravidiel a postupov. Napríklad existuje len jeden topbar a jeden footer. Oba sú nakódované v samostatných súboroch a na každú stránku, kde sa vyskytujú, sú len volané pomocou PHP príkazu **include**. To umožňuje footer aj topbar rýchlo upravovať a meniť, keďže stačí upraviť len ich zdrojový súbor a na všetkých stránkach sa takto upraví automaticky [33].

Ďalej sú tu pravidlá pre HTML atribúty. HTML atribúty slúžia ako ukazovatele pre ostatné jazyky ako kaskádové štýly, atribúty môžu byť ľubovoľné, ale medzi základné všeobecne používané sa zaraďujú **id**, **name** a **class**. Rôzne jazyky vedia pracovať s rôznymi atribútmi. Pre lepší prehľad a ako prevencia proti omylom, som zaviedol tieto pravidlá [34]:

- **class** je použitý ako indikátor pre kaskádové štýly.
- **id** je použité len ako ukazovateľ pre JavaScript, prípadne pre formuláre.
- **name** je použitý vždy len pre formuláre.
- **data-txt** - už spomínaný mnou navrhnutý atribút, ktorý používa JavaScript na indikáciu prekladov.
- **iné atribúty** - vo väčšine fungujú ako zobrazovacia logika pre kaskádové štýly, kde atribút môže byť napríklad **show** s hodnotami **TRUE** a **FALSE**. Hodnoty sú menené JavaScriptom, ktorý tieto atribúty nachádza len pomocou **id** a v kaskádových štýloch je pripravený štýl pre daný **class** pre hodnotu **TRUE** a **FALSE**.



Obr. 5.3: Znovu otvorenie cookies.

```

"txtAvailable": {
  "svk": "Žiadne dáta k dispozícii. Automatická obnova každých 15 sekúnd.",
  "eng": "No data available. Auto-refresh every 15 seconds.",
  "ukr": "Дані відсутні. Автоматичне оновлення кожні 15 секунд."
},

```

Obr. 5.4: Príklad z prekladového súboru.

Takto rozdelené atribúty umožňujú rýchlejší development, keďže nemôže prísť ku žiadnym nezhodám, alebo nedorozumeniam a ľahšie sa pochopia aj inému developerovi, keby má pracovať s mojimi kódmi. Výhodou je napríklad aj to, že iné atribúty sa v CSS volajú inak. Napríklad **class** sa v CSS volá pomocou body a názvu classu, napríklad ".index-body{kód pre grafiku}". Keby chceme v CSS však volať **id**, tak ho nevoláme bodkou, ale mriežkou "#index-body{kód pre grafiku}". Ak nie je určené, ktoré atribúty sa na čo používajú, tak sa spomalí development, lebo developer by si nedával pozor na to, čo akým atribútom pomenoval.

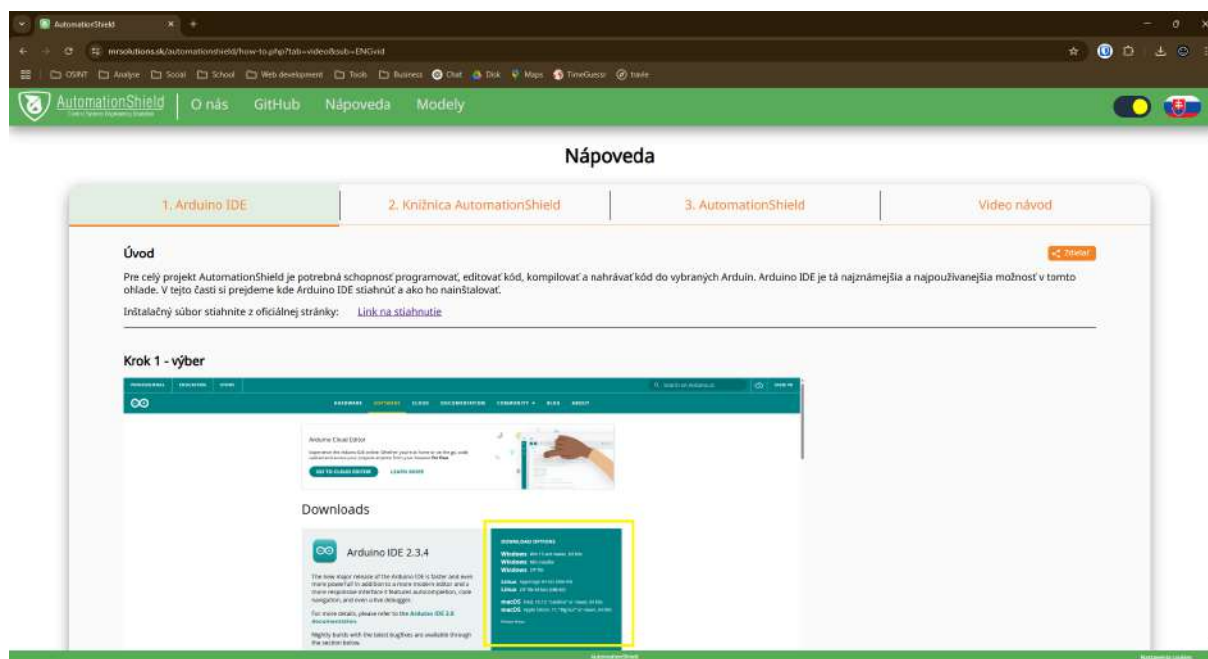
## 5.5 Podporné stránky

Ako som už spomínal, na webstránke sa nachádzajú aj stránky pre podporu používateľa a sú pri nich dodržané všetky vyššie uvedené pravidlá. Je možné, že v budúcnosti pribudnú ku týmto stránkam nové, ktorých účel však ešte nepoznáme. Môže sa napríklad jednať o nejaké kontaktné formuláre.

### 5.5.1 how-to.php

Táto webstránka je na rozdiel od ostatných webstránok s výnimkou webstránok GUI pomerne zložitá. Na Obr. 5.5 možno vidieť, že na prvý pohľad táto stránka pripomína typické všeobecné online návody, ako práca s operačným systémom alebo rôznymi programami. V jednoduchosti je krása, a preto som túto stránku nakódoval modúlárne. Nachádza sa tu množstvo návodov, ako inštalácia programu Arduino IDE, implementácia knižníc, ale aj návody na prácu so zariadeniami AutomationShield alebo návody na to,

ako riešiť prípadné problémy. Keďže sa jedná o stránku, kde budem s najväčšou pravdepodobnosťou v budúcnosti pridávať ďalší obsah, snažil som sa ju naprogramovať čo najjednoduchšie. Zvolil som preto modulárny prístup, kde na stránku **howto.php** volám moduly pomocou PHP funkcie **include**. Kód je tak rozdelený do menších častí a pridávanie alebo editácia existujúceho obsahu je ľahšia vďaka zvýšenej prehľadnosti.



Obr. 5.5: Úvodný náhľad na stránku how-to.php.

Tretia sekcia návodov, **3. AutomationShield**, v sebe skrýva niekoľko podsekcí, ako vidno na Obr. 5.6. Pri takomto množstve obsahu by finálny kód tejto webstránky prestal byť prehľadný. Možno si tiež všimnúť aj to, že každý jeden návod má v sebe ukryté tlačidlo **Zdieľať**. Po jeho kliknutí sa používateľovi skopíruje link na danú sekciu návodov a môže tak niekoho ľahšie naviesť na vyriešenie daného problému.

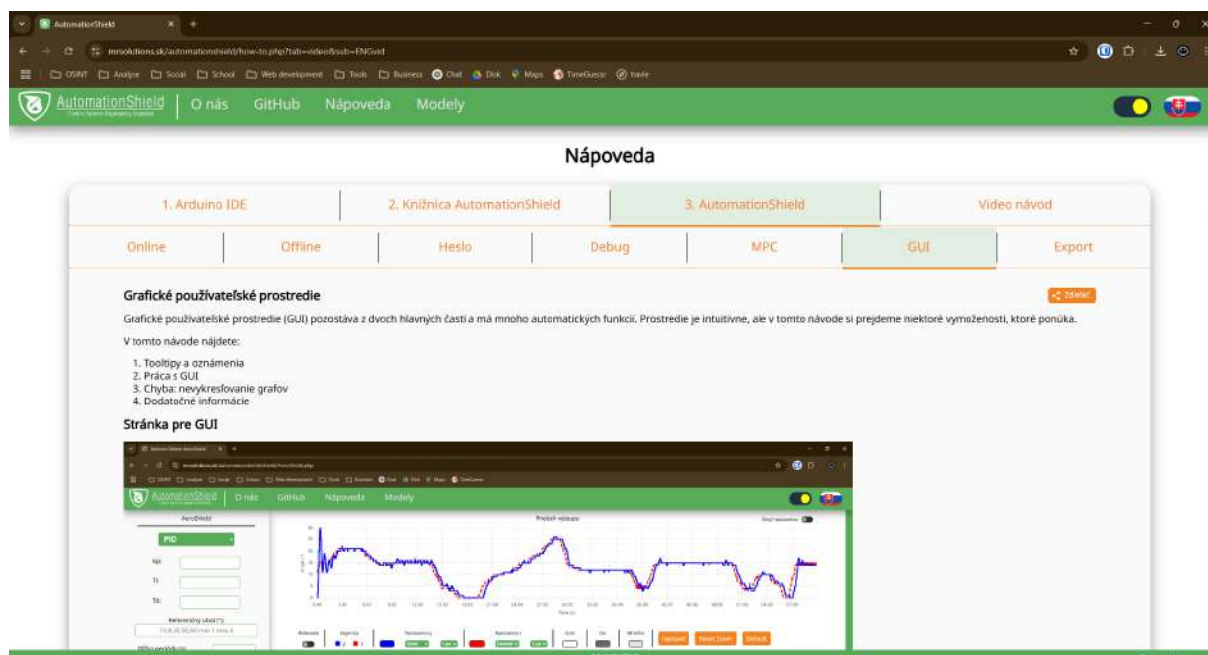
V neposlednom rade sa na tejto stránke nachádzajú aj videá, ako môžete vidieť na Obr. 5.7. Ide o kompletný návod inštalácie programu Arduino IDE, implementácia potrebných knižníc, otestovanie správnosti inštalácie, pripojenie AutomationShield zariadenia, získanie potrebného kódu na prácu s AutomationShield online a následné experimentovanie pomocou webového GUI. Momentálne sú dostupné dve videá s kompletným hlasovým komentárom v slovenčine a angličtine, s dostupnými ukrajinskými titulkami.

### 5.5.2 models.php

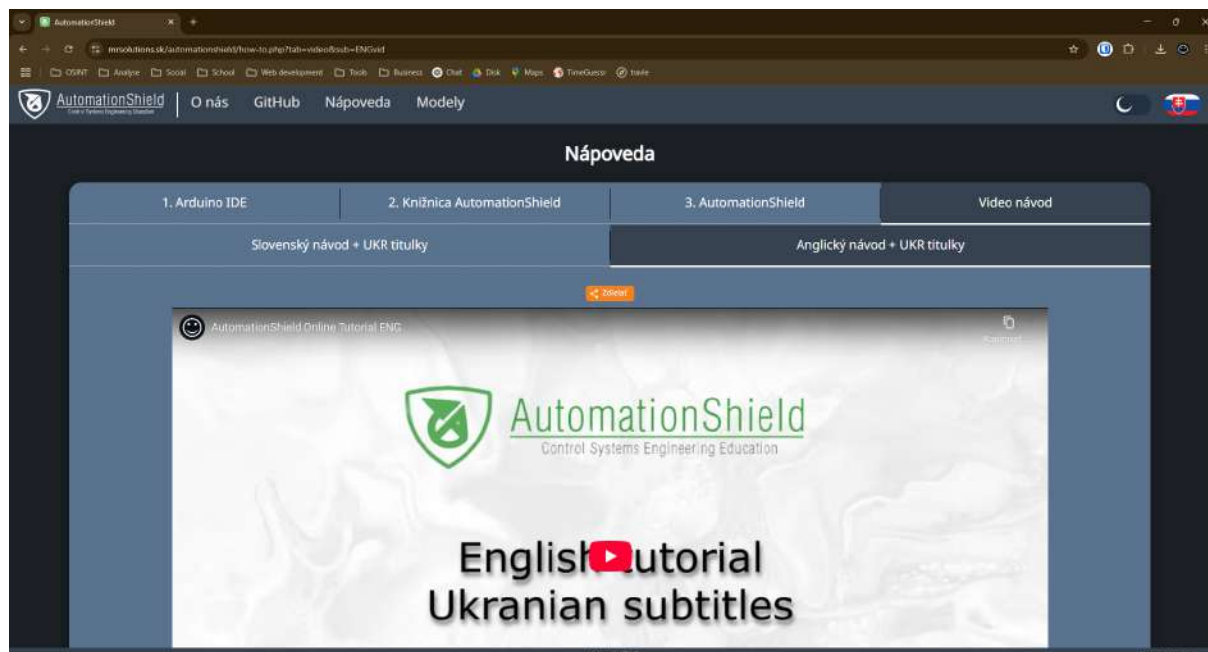
Na tejto stránke sa nachádzajú aktuálne podporované modely AutomationShieldu pre prácu s mojím GUI prostredím. Ponúka tu aj súbory na stiahnutie ako môj SerialMonitor.exe alebo Arduino IDE kód pre prácu online. Náhľad tejto stránky možno vidieť na Obr. 5.8.

### 5.5.3 aboutus.php

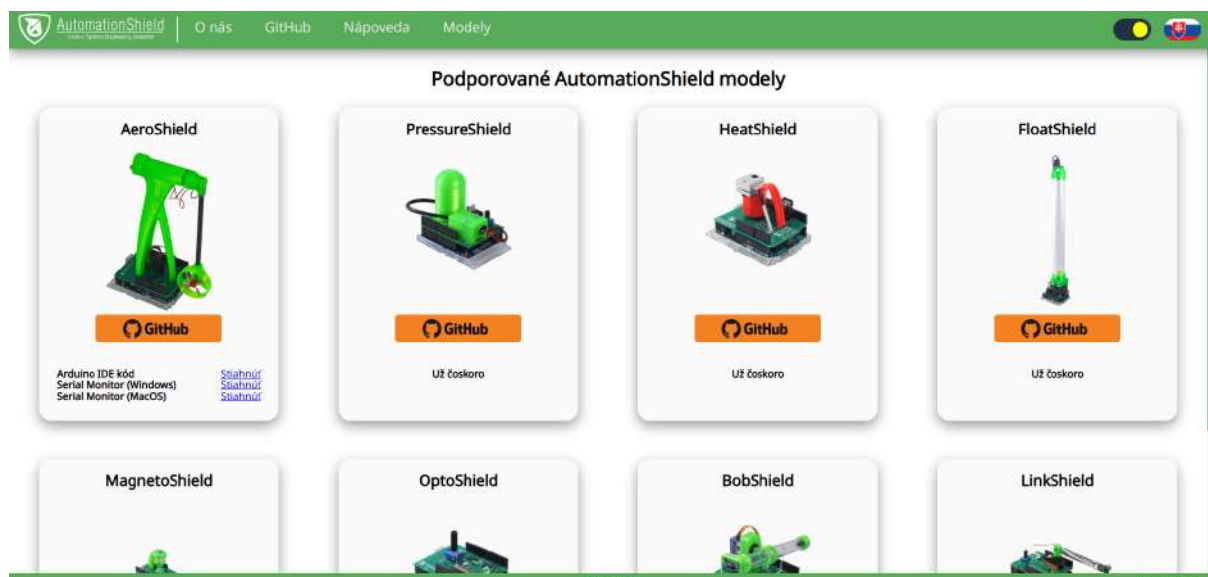
Nachádzajú sa tu informácie o iniciatíve AutomationShield, o mojom online GUI a podobne. Náhľad tejto stránky môžete vidieť na obrázku 5.9



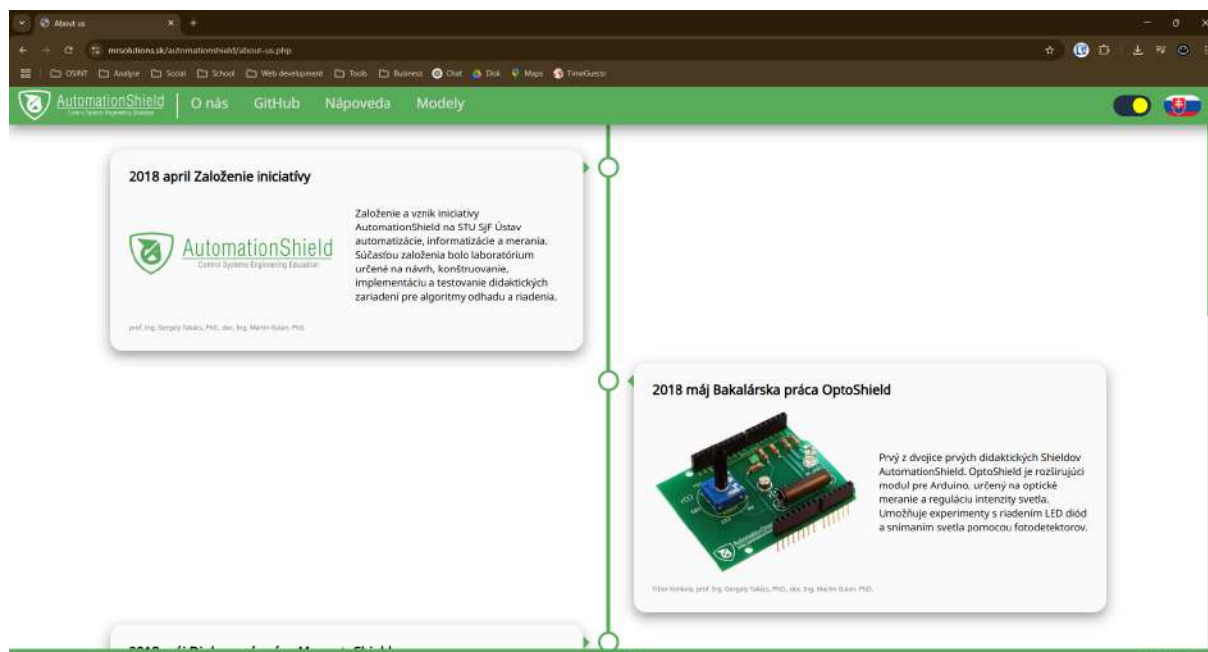
Obr. 5.6: Náhľad na tretiu sekciu v how-to.php.



Obr. 5.7: Náhľad na video sekciu v how-to.php [35].



Obr. 5.8: Náhľad na stránku models.php.



Obr. 5.9: Náhľad na stránku aboutus.php.

## 6 Podpora pre ďalšie AS zariadenia

Medzi cieľmi tejto práce je aj škálovateľnosť a modifikovateľnosť. Do tohto momentu som sa snažil urobiť automatizovaný systém na základe premenných, ktorý by ovládal dianie na stránke, ako som už vysvetlil v predchádzajúcich kapitolách. Na rad prišlo pridanie ďalších didaktických zariadení od AutomationShieldu.

### 6.1 Pridané zariadenia

Zariadenie, na ktorom prebiehal celý výskum a vývoj, bolo AeroShield, ktoré pracuje na princípe kyvadla, kde snímame uhol naklonenia kyvadla ovládaného vrtuľkou poháňanou motorom. Zatiaľ čo veci ako registrácia, prihlásenie a všetka komunikácia ostávajú rovnaké, iné zariadenia snímajú iné veličiny a sledujú iné fyzikálne princípy. Na výber sú mnohé zariadenia, ale pre dobrú reprezentáciu rôznych typov experimentov som sa rozhodol pridať:

- **FloatShield** – ventilátorom poháňaný tunel, kde sa za pomoci vzduchu vznáša sledovaná guľička.
- **MagnetoShield** – levitujúci magnet ktorého výšku sa snažíme kontrolovať.
- **FurutaShield** – najnovšie zariadenie, ktoré je vo vývoji súčasne s mojou prácou. Ide o invertované kyvadlo, pri ktorom je snímaných viac parametrov.

Treba však podotknúť, že iniciatíva AutomationShield pri vývoji každého Shieldu dbá na unifikáciu. Tento potvrdil aj pri implementácii spomenutých zariadení do môjho projektu, ktoré sú opísané v nasledujúcich podkapitolách.

#### 6.1.1 FloatShield

Už spomenuté zariadenie, ktoré sleduje levitáciu guľičky, ktorá je vytláčaná v priesvitnom tuneli za pomoci ventilátora [36], ktorý je vidno na Obr. 6.1.

Postup implementácie zariadenia do môjho systému bol teda nasledovný:

1. Skopírovanie môjho Arduino IDE kódu pre AeroShield. Kód som skopíroval a zmenil niektoré veci ako napríklad zmena premennej model na FloatShield, ktorý ukazuje stránke, o aký model sa jedná.
2. Skontrolovanie funkcií na registráciu a prihlásenie, v ktorých, ale žiadna zmena nebola nutná.
3. Implementácia všetkých regulačných modelov riadenia, ktoré FloatShield ponúka. Pri porovnávaní s AeroShieldom, som zistil, že vstupné a výstupné premenné sú skoro rovnaké. Vždy tam je jedna referencia, jedna sledovaná hodnota a jeden akčný zásah. Vstupné parametre ako napríklad  $K_p$ ,  $T_d$  a  $T_i$  parametre pre PID regulátor ostávajú rovnaké a jediné rozdiely boli len v tom, ako vyzerajú jednotlivé funkcie **step**. Editácia pôvodného AeroShield kódu pre FloatShield prebehla teda celkom rýchlo. Zatiaľ čo AeroShield bol vo vývoji mesiace a aj jeho IDE kód prešiel mnohými fázami, FloatShield bol nachystaný v priebehu niekoľkých hodín aj s testovaním jednotlivých regulačných modelov riadenia. Metódy riadenia sú teda rovnaké ako pre AeroShield a to:

- PID,
- MPC,



Obr. 6.1: Didaktické zariadenie FloatShield od iniciatívy AutomationShield

- MPC s manuálnym nastavením referencie,
- LQ,
- LQ s manuálnym nastavením referencie,
- Identifikácia,
- Otvorená slučka,

Po príprave Arduino IDE kódu bolo treba pridať FloatShield case do PHP kódov. Hlavný PHP skript **arduino\_online.php** nebolo treba meniť vôbec. Vzhľadom na podobnosť zariadení som v skripte **arduino\_experiment\_data.php** len skopíroval celý AeroShield case a pomenoval ho na FloatShield. Jediná úprava, ktorá bola nutná, bolo premenovanie databázovej tabuľky z AeroShield na FloatShield. Už spomínanú tabuľku som tiež len skopíroval z AeroShieldu.

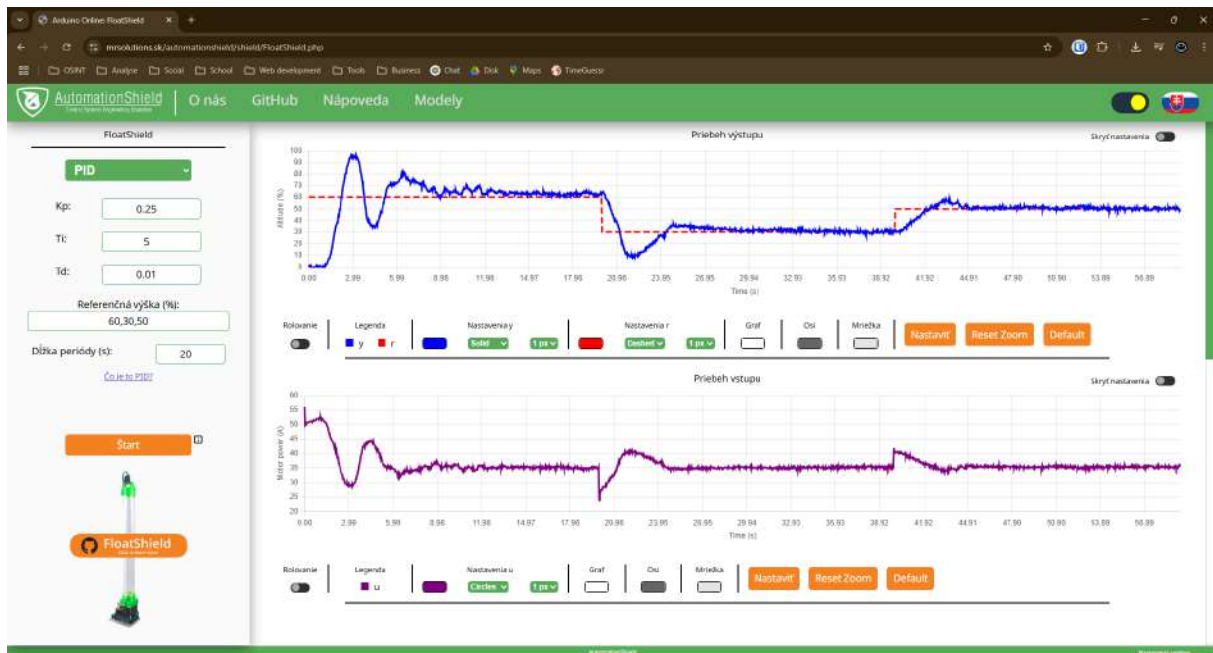
Samotné GUI, ktoré môžete vidieť na Obr. 6.2, je tiež len kópia pôvodného AeroShield GUI. Aj v ňom som z logickej stránky musel len zmeniť všetky premenné AeroShield na FloatShield.

Po grafickej stránke však potrebovalo viac zmien, hlavne zo strany textov. Implementácia celého FloatShieldu trvala jeden deň a následný deň prebiehalo testovanie a bug-fixing, hlavne v Arduino IDE kóde, kde som musel opraviť logiku niektorých step funkcií alebo opraviť prepočty časov a parametrov.

### 6.1.2 MagnetoShield

Ako ďalšie som pridal didaktické zariadenie MagnetoShield, ktoré je vidieť na Obr. 6.3. Postup bol úplne rovnaký, ako pri FloatShielde, keďže ide o experimentovanie s magneticky levitujúcim diskom,





Obr. 6.2: GUI pre FloatShield.

podobne ako pri FloatShielde sledujeme jeho polohu v milimetroch, dávame mu referenciu v milimetroch a vykonávame akčný zásah na silu magnetu vo voltoch [37].

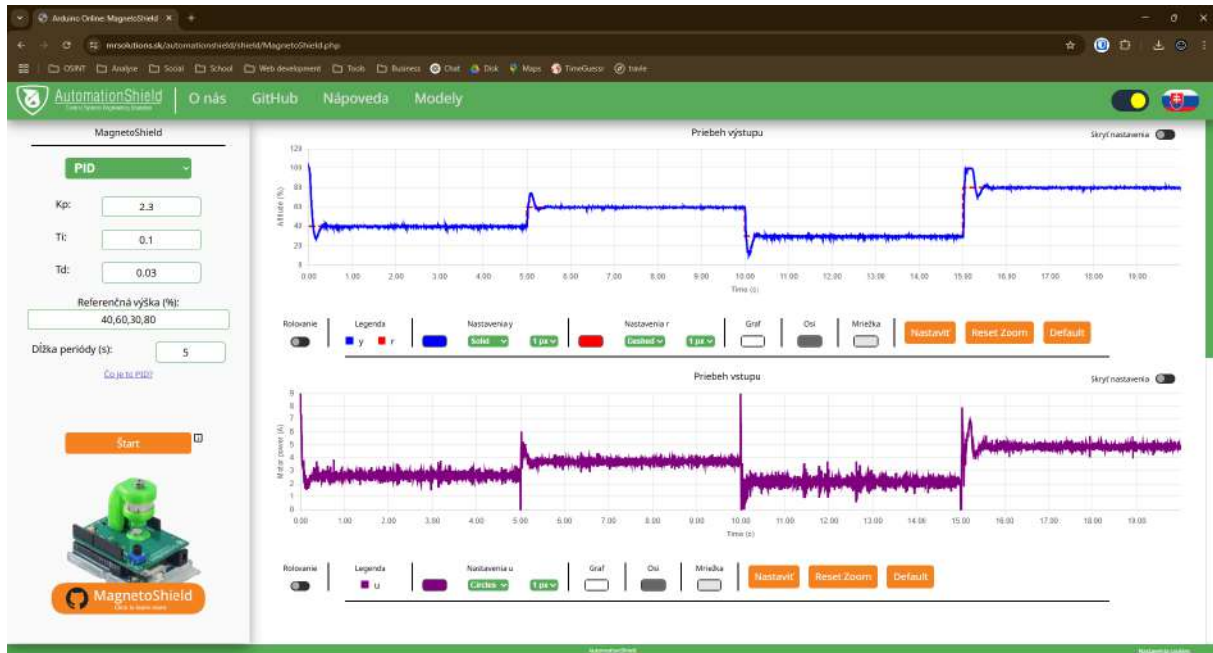


Obr. 6.3: Didaktické zariadenie MagnetoShield od iniciatívy AutomationShield.

Štruktúra všetkých kódov ostáva teda rovnaká. Musel som len pridať novú databázovú tabuľku MagnetoShield a doplniť skripty o case magneto, čo bolo len skopírovanie a upravenie predchádzajúcich skriptov. Hlavný skript AutomationShield\_online ostal nedotknutý. Výsledné funkčné užívateľské prostredie môžete vidieť na Obr. 6.4.

Zásah do MagnetoShieldu, ktorý musím spomenúť, že MagnetoShield pracuje s referenciou v milimetroch a to v rozsahu od 12 do 17 milimetrov. Aby som predišiel nedorozumeniam a uľahčil užívateľom prácu s týmto didaktickým zariadením, tak v GUI užívateľ zadáva referenciu v percentách od 0 do 100. Arduino túto referenciu vezme a upraví ju podľa na milimetre podľa vzťahu:





Obr. 6.4: GUI pre MagnetoShield.

$$\text{rmm} = \left( \frac{\text{rper}}{100} \right) \cdot (17.0 - 12.0) + 12 \quad (6.1)$$

kde: **rmm** je konvertovaná referencia na milimetre a **rper** je zadaná referencia v percentách.

Nameranú výšku a referenciu pred odoslaním na stránku konvertujem späť na percentá pomocou podľa vzťahu:

$$\text{xper} = \left( \frac{\text{xmm} - 12.0}{17.0 - 12.0} \right) \cdot 100.0 \quad (6.2)$$

kde: **xmm** je nameraná výška a referencia v milimetroch a **xper** - konverovaná referencia a výška na percentá.

Okrem tejto konverzie som musel do Arudino IDE kódu len spraviť úpravy MagnetoShield experimentov na moju štruktúru komunikácie a funkcií `step()`;

### 6.1.3 FurutaShield

Posledné zariadenie, ktoré som implementoval do prvej verzie mojej práce. FurutaShield je invertované kyvadlo pozostávajúce z otočného ramena, na ktorom sa nachádza kyvadlo, ktoré chceme dostať do hornej kolemej polohy, a môžete ho vidieť na obrázku 6.5. Toto didaktické zariadenie je v čase vývoja mojej práce takisto vo vývoji, takže sa podoba výsledného GUI aj celkové možnosti ovládania v budúcnosti možno zmenia.

Oproti ostatným didaktickým zariadeniam, ktoré som pridal, bola jeho implementácia o niečo zložitejšia, keďže samotný FurutaShield je omnoho zložitejšie zariadenie oproti ostatným. Verzia FurutaShieldu, s ktorým som pracoval, mala pripravené zatiaľ iba EMPC a LQR riadenie. Tieto riadenia som do mojej štruktúry IDE kódu aj databáz implementoval bez problémov. Oproti ostatným zariadeniam však pribudli štyri nové premenné súvisiace s ladením vyšvihnutia kyvadla a to:

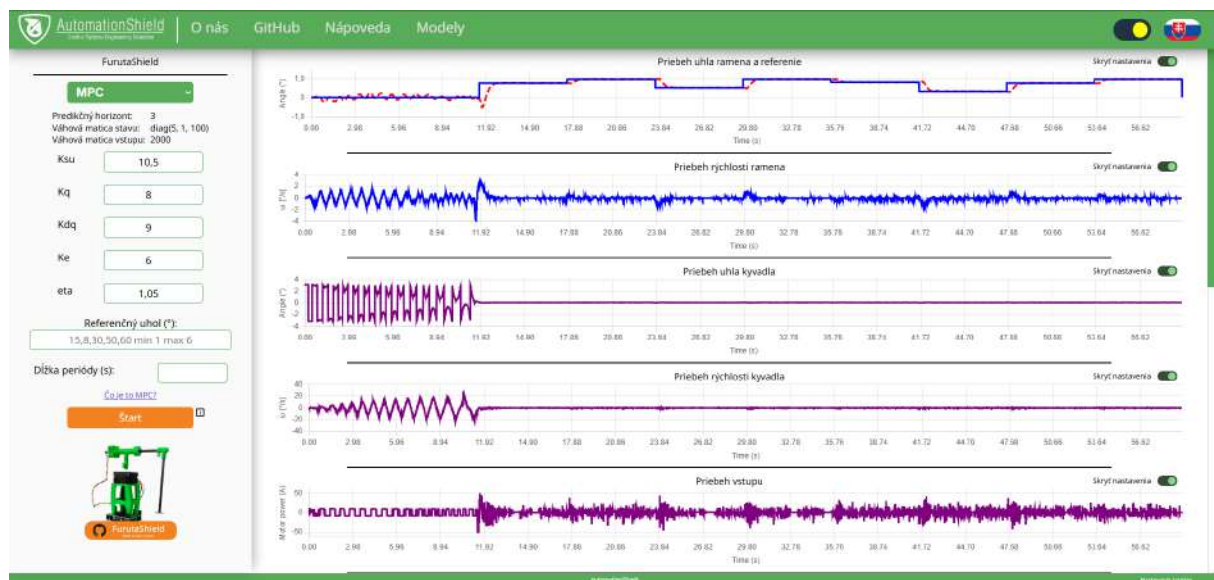
- **Ksu** – určuje základné zosilnenie systému.
- **Kq** – ovplyvňuje stabilitu uhla ramena.



Obr. 6.5: Didaktické zariadenie FurutaShield z open-source iniciatívy AutomationShield.

- **Kdq** – ovplyvňuje maximálnu rýchlosť systému.
- **Ke** – ovplyvňuje prírastok energie.

Tieto premenné treba ladiť podľa konkrétneho FurutaShieldu, pretože ich fyzikálne parametre sa môžu odlišovať. Preto bolo potrebné, aby používateľ mohol v rámci ladenia tieto parametre meniť. Výsledný formulár aj s novými vstupnými premennými môžete vidieť na Obr. 6.6, na ľavej strane.



Obr. 6.6: GUI pre FurutaShield

Samozrejme, že s novými parametrami bolo treba upraviť aj tabuľku **FurutaShield** o dané kolónky. Štruktúru tejto tabuľky môžete vidieť na Obr. 6.7.

Všetky ostatné skripty som len skopíroval a pridal nový **case** pre FurutaShield, s úpravou o nové parametre. V Arduino IDE kóde som zároveň upravil komunikáciu, aby Arduino prijalo aj tieto parametre, rovnako ako už ste videli komunikačné kódy v kapitolách vyššie.

Na Obr. 6.6 vyššie možno vidieť päť grafov oproti klasickým dvom, ako tomu je v GUI pre ostatné zariadenia. Pri FurutaShielde sledujeme viac parametrov a to:

Názov	Dátový typ
id	INT
arduino_onlin...	INT
Experiment	VARCHAR
START	VARCHAR
TIMER	INT
Kp	DECIMAL
Ti	DECIMAL
Td	DECIMAL
r1	DECIMAL
r2	DECIMAL
r3	DECIMAL
r4	DECIMAL
r5	DECIMAL
r6	DECIMAL
T	INT
Ksu	DECIMAL
Kq	DECIMAL
Kdq	DECIMAL
Ke	DECIMAL

Obr. 6.7: Databázová tabuľka pre FurutaShield.

- Polohu a referenciu ramena.
- Rýchlosť a referenciu ramena.
- Polohu a referenciu kyvadla.
- Rýchlosť a referenciu kyvadla.
- Akčný zásah.

Musel som pre to zmeniť grafické zobrazenie pre toto GUI, upraviť aj aplikáciu **SerialMonitor** a tabuľku nameraných dát o tieto parametre.

Rozdiely pre GUI sú predvolene skryté nastavenia grafov a zmenené veľkosti tak, aby bolo zobrazených všetkých päť grafov na jednej obrazovke. Na Obr. 6.8 môžete vidieť GUI pre FurutaShield s otvorenými nastaveniami.

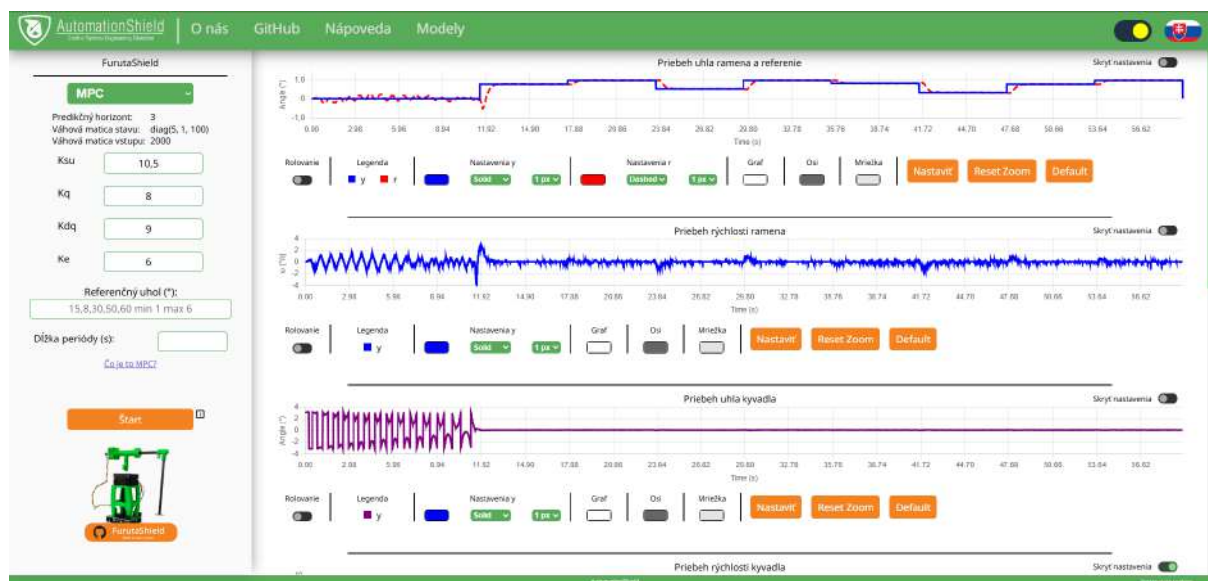
Štandardne už bolo treba doplniť preklady, nastaviť správne výpočty časov v Arduino IDE kóde, doplniť nové tooltipy, upraviť logiku step funkcií a podobne. Aj zložitý zariadenie ako FurutaShield sa však bez väčších problémov podarilo implementovať do mojej práce bez zmien v kódach a skriptoch komunikačnej logiky. Tento úspech potvrdil škálovateľnosť a modifikovateľnosť mnou navrhnutého systému.

### 6.1.4 Zhrnutie

Aj keď sa všetky didaktické zariadenia zdajú byť rovnaké, pracujú s rôznymi typmi riadiacich metód. Zhrnutie môžete vidieť v tabuľke 6.1.

Kde:

- **Man** - Manuálne zadávanie referencie pomocou potenciometra.
- **CL Ident** - Closed Loop identifikácia pomocou PID.
- **Kal** - Kalmanov filter.

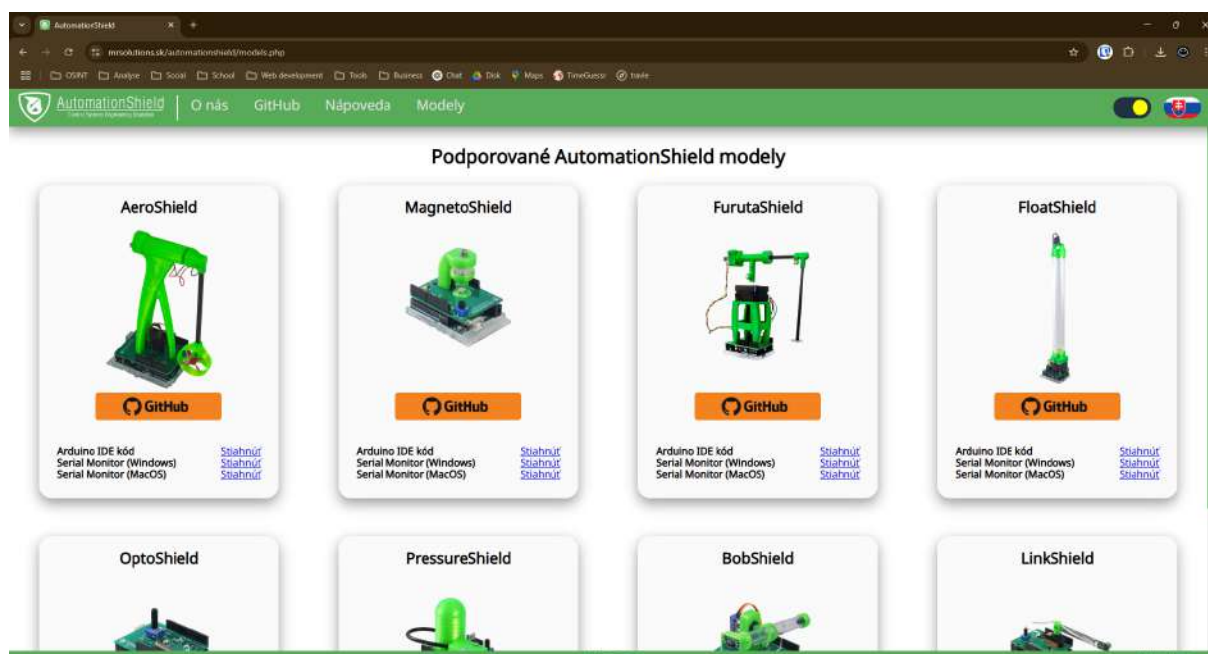


Obr. 6.8: GUI pre FurutaShield s otvorenými nastaveniami.

Tabuľka 6.1: Prehľad dostupných metód pre jednotlivé zariadenia.

Metóda	Aero	Float	Magneto	Furuta
PID	✓	✓	✓	✗
MPC	EMPC	MPC+Kal	EMPC	EMPC
MPC Man	✓	✓	✓	✗
LQ	LQI	LQ+Kal	LQ	LQR
LQ Man	✓	✓	✓	✗
Identifikácia	✓	✗	✗	✗
CL Ident	✗	✓	✓	✗
OpenLoop	✓	✓	✗	✗

Všetky štyri modely majú už svoje kódy na stránky s modelmi, ako môžete vidieť na obrázku 6.9 spolu s novou verziou aplikácie SerialMonitor. Zvyšné modely na stránke Modelov majú popis **Už čoskoro**, pretože ich implementácia je naplánovaná v blízkej budúcnosti.



Obr. 6.9: Finálna stránka s dostupnými súbormi na stiahnutie pre jednotlivé modely.

## 7 Testovanie AutomationShield Online

Na proces hromadného testovania som pozval dvoch spolužiakov a jednu doktorandku. Spolu so mnou sme testovali moju prácu v rovnakom čase. Dokumentačnú fotku môžete vidieť na Obr. 7.1. Testovanie v prípade pozvanej výpomoci zahŕňalo stiahnutie potrebných súborov z mojej webstránky, nasledovanie mojich návodov a testovanie zariadení pomocou GUI.



Obr. 7.1: Fotodokumentácia z hromadného testovania AutomationShield Online.

Výsledky testovania boli pozitívne. Môj systém bez problémov zvládol prácu s viacerými zariadeniami v rovnakom čase. Celkovo v komunikáciách medzi Arduino a serverom neprišlo k žiadnym problémom. Pri ovládaní zariadení takisto neprišlo k žiadnym problémom a všetko fungovalo tak, ako to bolo zamýšľané. Testovaný bol AeroShield, FloatShield a MagnetoShield. Testovanie prebehlo na rôznych verziách Arduino IDE. Webové prehliadače boli Mozilla Firefox, Chrome a Edge. Všetky počítače išli na systéme Windows 11.

Jediné problémy vznikli pri funkčnosti samotnej webstránky ako takej. Jednému z testujúcich počítač odmietal nahrať súbor s prekladmi, a tak mal celú stránku bez textov. Ďalší mal zase problém so zobrazením oznamu o cookies. Okrem toho som dostal od testujúcich niekoľko dobrých poznámok a požiadaviek, čo zmeniť alebo zlepšiť.

### 7.1 Problém s prekladmi

Pred opísaním tejto chyby a jej riešenia musím deklarovat' prečo je tento projekt na mojej doméne. Projekt beží na doméne <https://www.mrsolutions.sk>, keďže vrámci vývoja potrebujem úplnú kontrolu nad serverom, jeho nastaveniami. Po schválení vedením iniciatívy AS bude web premiestnený na AutomationShield doménu.

Chybu som riešil priamo na počítači, kde sa problém vyskytoval. Z developerskej konzoly som zistil, že problém je spôsobený politikou CORS (Cross-Origin Resource Sharing), čo je mechanizmus, ktorý chráni užívateľa aj webstránky pred načítavaním obsahu z iných domén [38].

Chybové hlásenie, ktoré som získal, je uvedené v prílohe 7.1:



## Výpis z kódu 7.1: Chyba načítania prekladov.

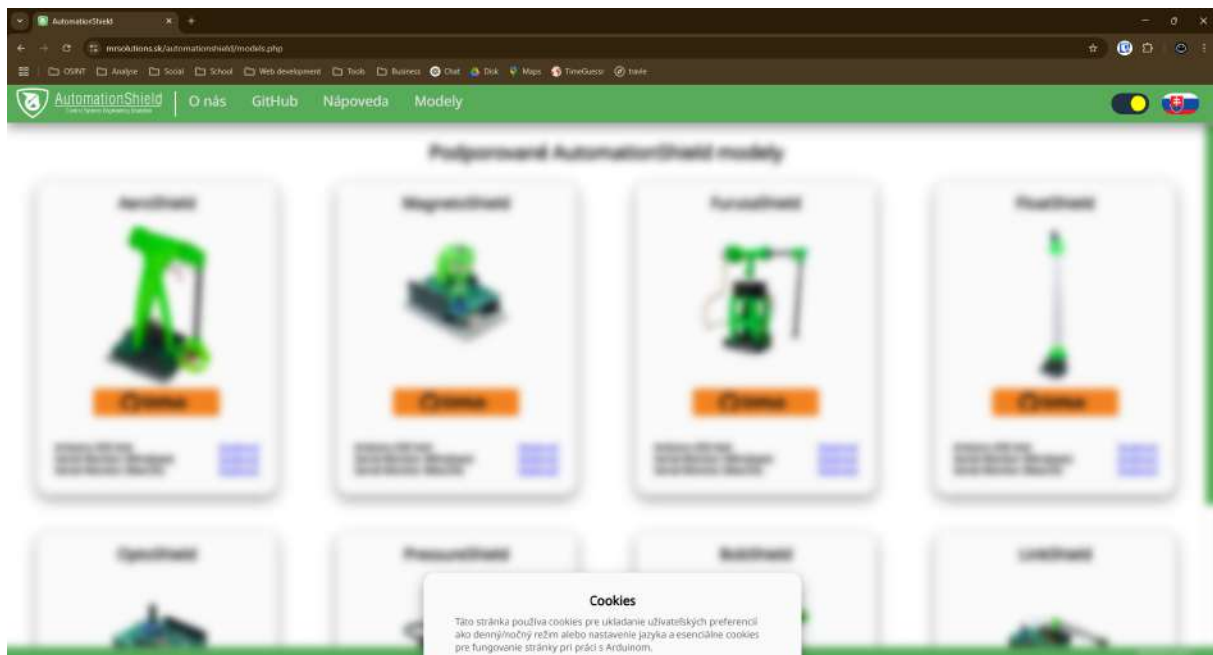
```
Access to XMLHttpRequest at
'https://mrsolutions.sk/automationshield/A_translations.json'
from origin 'https://www.mrsolutions.sk'
has been blocked by CORS policy
```

Problém nastal preto, že môj web beží na doméne **https://www.mrsolutions.sk**, ale JSON súbor s prekladmi som sťahoval z **https://mrsolutions.sk** (bez „www“). Aj keď tieto domény technicky smerujú na ten istý server, z pohľadu prehliadača ide o rôzne origíny (domény). Väčšina prehliadačov preto vyhodnotí požiadavku ako porušenie CORS politiky, ak server výslovne nepovolí prístup z iného originu.

Alternatívnym riešením by bolo upraviť konfiguráciu servera tak, aby povoľoval zdieľanie (CORS header **Access-Control-Allow-Origin**). Z bezpečnostných dôvodov som sa však rozhodol radšej zmeniť požiadavku tak, aby smerovala na tú istú doménu (pridať „www“). Tento jednoduchý zásah problém vyriešil.

## 7.2 Zlé zobrazenie Cookies

Tento problém bol čisto grafický. Celú stránku som po grafickej stránke kódoval dynamicky, aby sa stránka prispôbovala užívateľovi. Táto chyba, ktorú môžete vidieť na obrázku 7.2, bola pozostatok z môjho testovania. Pri hľadaní správneho rozloženia som použil grafické jednoty vzdialenosti px (pixel), ktoré sa neprispôbujú obrazovke.



Obr. 7.2: Zlé zobrazenie Cookies.

Oprava bola jednoduchá a vyžadovala len zmenu jednotiek z px (pixel) na vh (viewer height), ktorá udáva percentuálnu vzdialenosť podľa veľkosti použitej obrazovky.

## 7.3 Poznámky a požiadavky

Ako som spomenul, počas testovania som dostal aj niekoľko vstupov na zmeny a vylepšenia, ktoré si teraz v krátkosti prejdeme.

### 7.3.1 Pomenovanie kariet a Favicon.ico

Pred testovaním som nazýval každú stránku podľa jej obsahu. Zároveň som si neuvedomil, že webstránka nemá ikonku v kartách. Testéri sa sťažovali, že nevedeli nájsť otvorenú kartu, lebo v nej nebol názov stránky, ale len pomenovanie obsahu. Na Obr. 7.3 môžete vidieť karty webstránky pred a po zmene.



Obr. 7.3: Staré a nové karty.

### 7.3.2 Drobné zmeny textov

Tieto zmeny nie je potrebné detailnejšie dokumentovať. Vo väčšine sa jedná len o zmenu jedného slova, ako napríklad zmena **Akceptovať** **esencialne** na **Akceptovať** **základné** a iné podobné zmeny v návodoch alebo v tooltipoch.

### 7.3.3 Zmeny v GUI formulároch

Na formuláre som dostal viacero vstupov. Ako prvý bolo nedostatočné navedenie na chybu. Pri programovaní formulárov som nepoužil pseudoelement **required**. To malo za príčinu, že ak používateľ zabudol vyplniť nejaké pole, stránka ho nepožiadala, aby toto pole vyplnil, ale len vypísala chybovú hlášku "**niečo je zle**", ktorú som spomenul v kapitole vyššie. Na Obr. 7.4 môžete vidieť, ako sa formulár správa v prípade, keď používateľ nevyplní nejaké pole.



Obr. 7.4: Chybná hláška na vstupných poliach formulárov.

Ako ďalšie bolo zadávanie doby trvania experimentu alebo doby trvania jednej referencie v cykle. Pri mojich testoch som vždy používal celé čísla a podľa nich som aj pripravil programy v Arduino IDE. Skript na vkladanie hodnôt z formuláru do databázy som naprogramoval s funkciou **intval**, ktorá prípadné desatinné čísla zmenila na celé zaokruhlovaním dole. Pri testovaní sa však jeden testujúci pokúsil nastaviť dobu na **0.55**. Kvôli vyššie spomenutej funkcii **intval**, táto hodnota ani neprišla do databázy a užívateľ dostal mnou pripravenú hlášku "**Niečo je zle**". Problém som opravil tým, že som nastavil minimálnu dĺžku kroku v týchto poliach na 1 a pripravil skript, ktorý užívateľovi nedovolí v týchto poliach napísať desatinnú čiarku a bodku. Zároveň som upravil tooltip, aby lepšie užívateľovi opísal funkčnosť a požadovaný vstup, ktorý môžete vidieť na Obr. 7.5.

V neposlednom rade bola požiadavka na nastavenie základných parametrov do polí Kp, Ti a Td pre PID experimenty. Požiadavke som vyhovel a v aktuálnej verzii sa užívateľovi pri PID otvorí formulár už s vyplnenými parametrami, ako môžete vidieť na obrázku 7.6

## 7.4 Testovanie projektu na MacOS

Celý projekt bol vyvíjaný na operačnom systéme Windows 11. Cieľom však bolo zabezpečiť čo najširšiu kompatibilitu s rôznymi zariadeniami. MacOS je rozšírený natívny operačný systém pre produkty spo-



Doba trvania jednej referencie v sekundách. Len jedno celé číslo (nie desatinné), ktoré platí pre všetky referencie.

Dĺžka periódy (s): 5

[Čo je to PID?](#)

Obr. 7.5: Oprava vstupných polí zadávania času.

MagnetoShield

PID

Kp: 2,3

Ti: 0,1

Td: 0,03

Referenčná výška (%): 15,8,30,50,60 min 1 max 6

Dĺžka periódy (s): 5

[Čo je to PID?](#)

Obr. 7.6: Polia vstupných parametrov pre PID s prednastavenými hodnotami.

ločnosti Apple, a preto bola celá práca navrhnutá tak, aby bola použiteľná aj na týchto zariadeniach — s výnimkou aplikácie **SerialMonitor**.

Aplikácia **SerialMonitor** bola naprogramovaná v jazyku Python a pomocou knižnice Tkinter exportovaná do formátu **.exe**, určeného pre Windows. Tento formát však nie je kompatibilný s MacOS a aplikáciu tak nie je možné na zariadeniach Apple spustiť. Samotný Python kód, vrátane použitia knižnice Tkinter, je však plne kompatibilný s MacOS [39]. Na vytvorenie natívnej aplikácie vo formáte **.app**, ktorý je určený pre MacOS, je potrebné vykonať export priamo na zariadení s týmto operačným systémom.

K zariadeniu s MacOS som získal prístup až tesne pred odovzdaním práce, a preto je verzia pre MacOS momentálne ešte vo fáze testovania. Ostatné komponenty projektu *AutomationShield Online* sú však plne kompatibilné a otestované aj na tomto operačnom systéme.

## 8 Záver

V rámci mojej diplomovej práce sa mi podarilo vytvoriť modifikovateľný a škálovateľný systém databáz, skriptov a kódov, ktoré spolu umožňujú ovládanie zariadení AutomationShield pomocou grafického používateľského rozhrania (GUI). Počas vývoja som musel vyriešiť množstvo problémov a často som narazil aj na slepé uličky pri hľadaní správnej komunikácie medzi Arduino a serverom.

Výsledný systém je používateľsky priateľský a intuitívny. Webstránka obsahuje množstvo návodov, tooltipov a poskytuje široké možnosti modifikácie. Arduino využíva iba knižnicu `AutomationShield` a oficiálnu knižnicu od Arduina `ArduinoHttpClient`, čím som znížil potenciálne problémy s kompatibilitou v budúcnosti. Celý projekt som navrhol bez akýchkoľvek hardvérových úprav. Používateľ môže spúšťať experimenty bez nutnosti opätovnej kompilácie a nahrávania programu, vrátane možnosti meniť parametre experimentu cez GUI.

Jedinou komplikáciou bolo odosielanie nameraných dát na server. Kvôli obmedzenej pamäti a štruktúre Arduina nebolo možné odosielať dáta priamo na server. Tento problém som vyriešil pomocou mojej aplikácie `SerialMonitor`, ktorú používateľ spustí a ktorá sama odosiela dáta na server. Zároveň dáta zapisuje do súboru `experiment_data.mat`, čím sa vyriešil aj export experimentov.

Webová aplikácia dokáže obslúžiť neobmedzený počet používateľov. Databázy sú plne automatizované, Arduino sa samo zaregistruje a používateľ má plynulý priebeh interakcie so systémom, keďže všetko je spracované na pozadí.

GUI spolu s aplikáciou `SerialMonitor` poskytujú množstvo informácií o dianí v systéme, vrátane prípadných chýb, ktoré sú podrobne opísané a vyriešené v návodoch. Táto práca predstavuje aj potenciálny prínos pre priemysel, keďže Arduino R4 je pomerne nové zariadenie a na webe, vrátane oficiálnej stránky Arduina, je málo návodov na jeho použitie, najmä v takto rozsiahlych projektoch. Výskum preto vyžadoval náročnú prácu a samostatné hľadanie riešení.

Projekt je veľmi kompatibilný s rôznymi zariadeniami, keďže beží vo webovom prostredí, a tak nie je problém ho spustiť aj na slabších počítačoch. Podarilo sa mi vykonať aj hromadné testovania, ktoré potvrdili funkčnosť systému. Projekt je pripravený na používanie na operačnom systéme Windows, čo potvrdili testovania. Pri týchto testovaniach sa objavili iba drobné nedostatky, ako nepresné texty alebo chybné nastavená grafika, ktoré sa podarilo rýchlo opraviť. Projekt je plne kompatibilný aj s operačným systémom MacOS, až na aplikáciu `SerialMonitor`, na ktorej pracujem.

Celý projekt so všetkými zdrojovými kódmi, vrátane Arduino IDE, PHP, HTML, CSS, SQL, JavaScriptu a Pythonu, sa nachádza v mojom GitHub repozitári [40].

### 8.1 Motivácia na ďalšiu prácu

Projekt však otvára priestor aj pre ďalšie rozšírenia a možnosti:

- Rozšíriť projekt o plnú podporu aj pre operačný systém MacOS.
- Ako prvé by bolo vhodné doplniť podporu pre všetky didaktické zariadenia z iniciatívy AutomationShield a implementovať aj možnosť nahrávania súboru `EMPC_ctrl.h` (regióny) cez GUI priamo do Arduina.
- Preskúmať aj riešenia, ktoré by si vyžadovali hardvérový zásah, no mohli by eliminovať potrebu aplikácie `SerialMonitor.exe`.

- Implementovať tento systém aj na iné zariadenia ako AutomationShield, napríklad priamo v priemyselnom prostredí.
- Integrovať do projektu umelú inteligenciu, ktorá by mohla nájsť uplatnenie napríklad pri realizácii PID regulácie.
- Rozšíriť GUI o animované vykresľovanie príslušného zariadenia v reálnom čase.
- Rozšíriť webovú stránku v sekcii návodov a vytvoriť podrobnejšie video návody — ako pre AutomationShield Online, tak aj pre klasickú prácu so zariadeniami AutomationShield.
- Vytvoriť didaktické cvičenia s využitím AutomationShield Online, v ktorých by študenti museli identifikovať systém, spracovať exportované dáta a vypočítať PID parametre vhodne zvolenou metódou.
- Vytvoriť projekt zameraný na rozšírenie iniciatívy AutomationShield aj na stredné odborné školy a ďalšie vysoké školy, s využitím tejto nadstavby.

# Literatúra

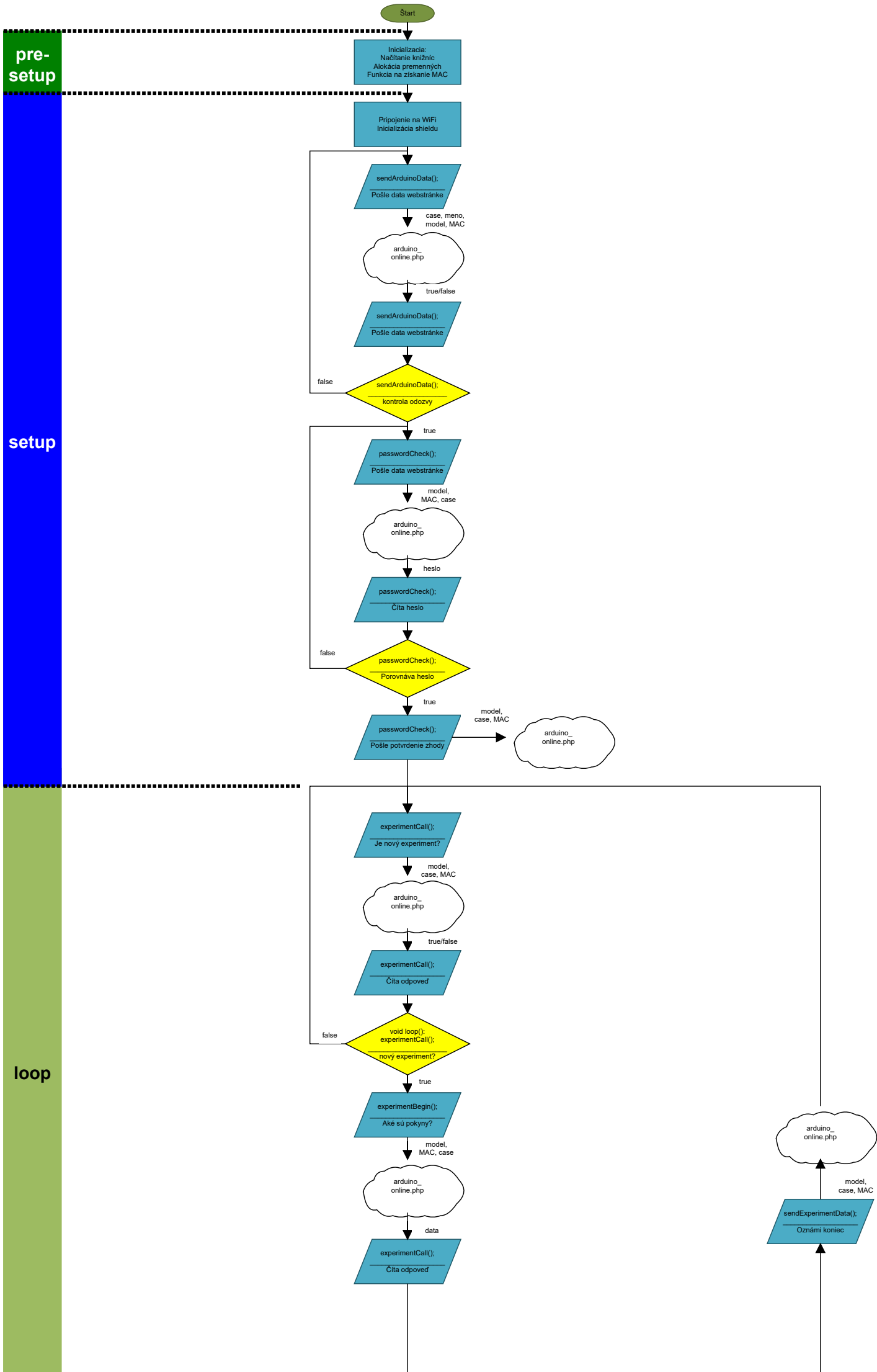
- [1] G. Takács and M. Gulán, *Základy Prediktívneho Riadenia*, 1st ed. Bratislava, Slovakia: Spektrum STU, 2018, in Slovak language. (Fundamentals of Predictive Control).
- [2] P. Horáček, “Laboratory experiments for control theory courses: A survey,” *Annual Reviews in Control*, vol. 24, pp. 151–162, 2000.
- [3] Gergely Takács et al., “AutomationShield Wiki,” <https://github.com/gergelytakacs/AutomationShield/wiki>, 2024, online; prístup 24. mája 2025.
- [4] A. Choubey, S. Jain, and P. Padhy, “Comprehensive tuning of frequency shifted imc-pid controller for boost converter using harris hawk optimization,” *Engineering Research Express*, vol. 7, 05 2025.
- [5] N. K. Prabowo and I. Irwanto, “The implementation of arduino microcontroller boards in science: A bibliometric analysis from 2008 to 2022,” *arXiv preprint arXiv:2312.10840*, 2023. [Online]. Available: <https://arxiv.org/abs/2312.10840>
- [6] Arduino Support, “What’s the difference between UNO R3 and UNO R4 boards?” 2025, accessed: 2025-02-28. [Online]. Available: <https://support.arduino.cc/hc/en-us/articles/9350551575964-What-s-the-difference-between-UNO-R3-and-UNO-R4-boards>
- [7] Arduino, “Arduino UNO R4 WiFi board image,” Online Image, 2025, accessed: 2025-02-28. [Online]. Available: <https://store.arduino.cc/en-sk/products/uno-r4-wifi>
- [8] —, “Why Arduino pro,” 2025, accessed: 2025-02-28. [Online]. Available: <https://www.arduino.cc/pro/why-pro>
- [9] R. Pashic, I. Kuzmanov, and K. Atanasovski, “Espressif esp32 development board in wifi station communication mode,” pp. 1–6, 05 2020.
- [10] Andreas, C. R. Aldawira, H. W. Putra, N. Hanafiah, S. Surjarwo, and A. Wibisurya, “Door security system for home monitoring based on esp32,” *Procedia Computer Science*, vol. 157, pp. 673–682, 2019, the 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050919311378>
- [11] A. Zhaxalikov, A. Mombekov, and Z. Sotsial, “Surveillance camera using wi-fi connection,” *Procedia Computer Science*, vol. 231, pp. 721–726, 2024, 14th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 13th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (EUSPN/ICTH 2023). [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1877050923021592>
- [12] I. Anshori, G. F. Mufiddin, I. F. Ramadhan, E. Ariasena, S. Harimurti, H. Yunkins, and C. Kurniawan, “Design of smartphone-controlled low-cost potentiostat for cyclic voltammetry analysis based on esp32 microcontroller,” *Sensing and Bio-Sensing Research*, vol. 36, p. 100490, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2214180422000198>
- [13] W. Rothschof, “Arduino webserver receive post data and parameters,” 2020. [Online]. Available: [https://werner.rothschof.net/202003\\_arduino\\_webserver\\_post\\_en.htm](https://werner.rothschof.net/202003_arduino_webserver_post_en.htm)
- [14] Arduino, “Wifi chat server - arduino,” 2016. [Online]. Available: <https://www.arduino.cc/en/Tutorial/WiFiChatServer>

- 
- [15] Arduino, “Arduinohttpclient library documentation,” 2025, accessed: 2025-02-28. [Online]. Available: <https://docs.arduino.cc/libraries/arduinohttpclient/>
- [16] Apais, “Part 1 - send arduino data to the web (php/mysql/d3.js),” 2015. [Online]. Available: <https://www.instructables.com/PART-1-Send-Arduino-data-to-the-Web-PHP-MySQL-D3js/>
- [17] Wikipedia contributors, “MAC address,” 2024, [Online; accessed 2024-03-02]. [Online]. Available: [https://en.wikipedia.org/wiki/MAC\\_address](https://en.wikipedia.org/wiki/MAC_address)
- [18] Espressif Systems, *Miscellaneous System API — ESP-IDF Programming Guide*, 2024, [Online; accessed 2024-03-02]. [Online]. Available: [https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/system/misc\\_system\\_api.html](https://docs.espressif.com/projects/esp-idf/en/stable/esp32s3/api-reference/system/misc_system_api.html)
- [19] N. Provos and D. Mazieres, “Bcrypt password hashing algorithm,” in *USENIX Annual Technical Conference, FREENIX Track*, 1999.
- [20] A. Biryukov, D. Dinu, and D. Khovratovich, “Argon2: The memory-hard function for password hashing and other applications,” *IACR Transactions on Symmetric Cryptology*, 2016.
- [21] N. I. of Standards and Technology, “Advanced encryption standard (aes),” <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>, 2001, FIPS PUB 197.
- [22] P. Tibenský, “AeroShield: Miniaturný experimentálny modul aerokyvadla,” Bakalárska práca, Slovenská technická univerzita v Bratislave, Strojnícka fakulta, 2022, evidenčné číslo: Sjf-13432-106440.
- [23] A. Documentation, “Arduino uno r4 wifi cheat sheet,” 2024. [Online]. Available: <https://docs.arduino.cc/tutorials/uno-r4-wifi/cheat-sheet/>
- [24] ESPBoards.dev, “A dive deep guide on the new arduino uno r4 and programming,” 2024. [Online]. Available: <https://www.espboards.dev/blog/how-to-program-arduino-uno-r4-mcus/>
- [25] R. Controls, “Tkinter with serial,” <https://robotic-controls.com/learn/python-guis/tkinter-serial>, 2013, online tutorial.
- [26] Chart.js Community, *Chart.js Documentation*, 2025, accessed: 2025-03-05. [Online]. Available: <https://www.chartjs.org/docs/latest/>
- [27] Kinsta. (2024) 14 css best practices for beginners. Najlepšie praktiky pre používanie CSS. [Online]. Available: <https://kinsta.com/blog/css-best-practices/>
- [28] H. Alk. (2024) Javascript best practices. Odporúčania pre písanie čistého a udržiavateľného JavaScript kódu. [Online]. Available: <https://dev.to/harshanalk/javascript-best-practices-5dc6>
- [29] Cloudflare, Inc., “What are cookies?” 2023, accessed: 2025-03-28. [Online]. Available: <https://www.cloudflare.com/learning/privacy/what-are-cookies/>
- [30] European Commission, “Cookies policy,” 2024, accessed: 2025-03-28. [Online]. Available: [https://commission.europa.eu/cookies-policy\\_en](https://commission.europa.eu/cookies-policy_en)
- [31] Termly, “Do i need a cookie policy on my website?” 2024, accessed: 2025-03-28. [Online]. Available: <https://termly.io/faq/do-i-need-a-cookie-policy-on-my-website/>
- [32] A. Remdt. (2021) Building a super small and simple i18n script in javascript. Popis jednoduchého i18n skriptu využívajúceho atribúty `data-i18n` a JSON súbory. [Online]. Available: <https://andreasremdt.com/blog/building-a-super-small-and-simple-i18n-script-in-javascript/>
- [33] S. APM. (2020) How to use php include and require statements. Popis použitia PHP funkcií include a require na modularizáciu webových komponentov. [Online]. Available: <https://www.scoutapm.com/blog/how-to-use-php-include-and-require-statements>
- [34] W. contributors. (2025) Html attribute. Prehľad HTML atribútov vrátane id, class, name a data-\*. [Online]. Available: [https://en.wikipedia.org/wiki/HTML\\_attribute](https://en.wikipedia.org/wiki/HTML_attribute)
- [35] AS, “Automationshield online – návody: Video návody (anglické video),” <https://mrsolutions.sk/automationshield/how-to.php?tab=video&sub=ENGvid>, 2025, online; navštívené 28. mája 2025.

- [36] G. Takács, P. Chmurčiak, M. Gulán, E. Mikuláš, J. Kulhánek, G. Penzinger, M. Podbielančík, M. Lučan, P. Šálka, and D. Šroba, “Floatshield: An open source air levitation device for control engineering education,” in *Proceedings of the 21st IFAC World Congress*. Berlin, Germany: IFAC, July 12–17 2020, pp. 1–8, supplementary files: Presentation file, Annotated audio/video presentation.
- [37] G. Takács, J. Mihalík, E. Mikuláš, and M. Gulán, “Magnetoshield: Prototype of a low-cost magnetic levitation device for control education,” in *Proceedings of the 2020 EDUCON IEEE Global Engineering Education Conference*. Porto, Portugal: IEEE, April 27–30 2020, pp. 1516–1525, supplementary files: Presentation file, Annotated audio/video presentation.
- [38] Mozilla Developer Network. (2024) CORS (Cross-Origin Resource Sharing). Accessed: 2025-05-24. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>
- [39] Python Software Foundation, “Tkinter — python interface to tcl/tk,” 2024, accessed: 2025-05-29. [Online]. Available: <https://docs.python.org/3/library/tkinter.html>
- [40] M. Repka, “Automationshield online: Complete source code repository for automationshield online project,” <https://github.com/MatusRepkaSolutions/AutomationShieldOnline>, 2025, accessed: 2025-05-30. [Online]. Available: <https://github.com/MatusRepkaSolutions/AutomationShieldOnline>

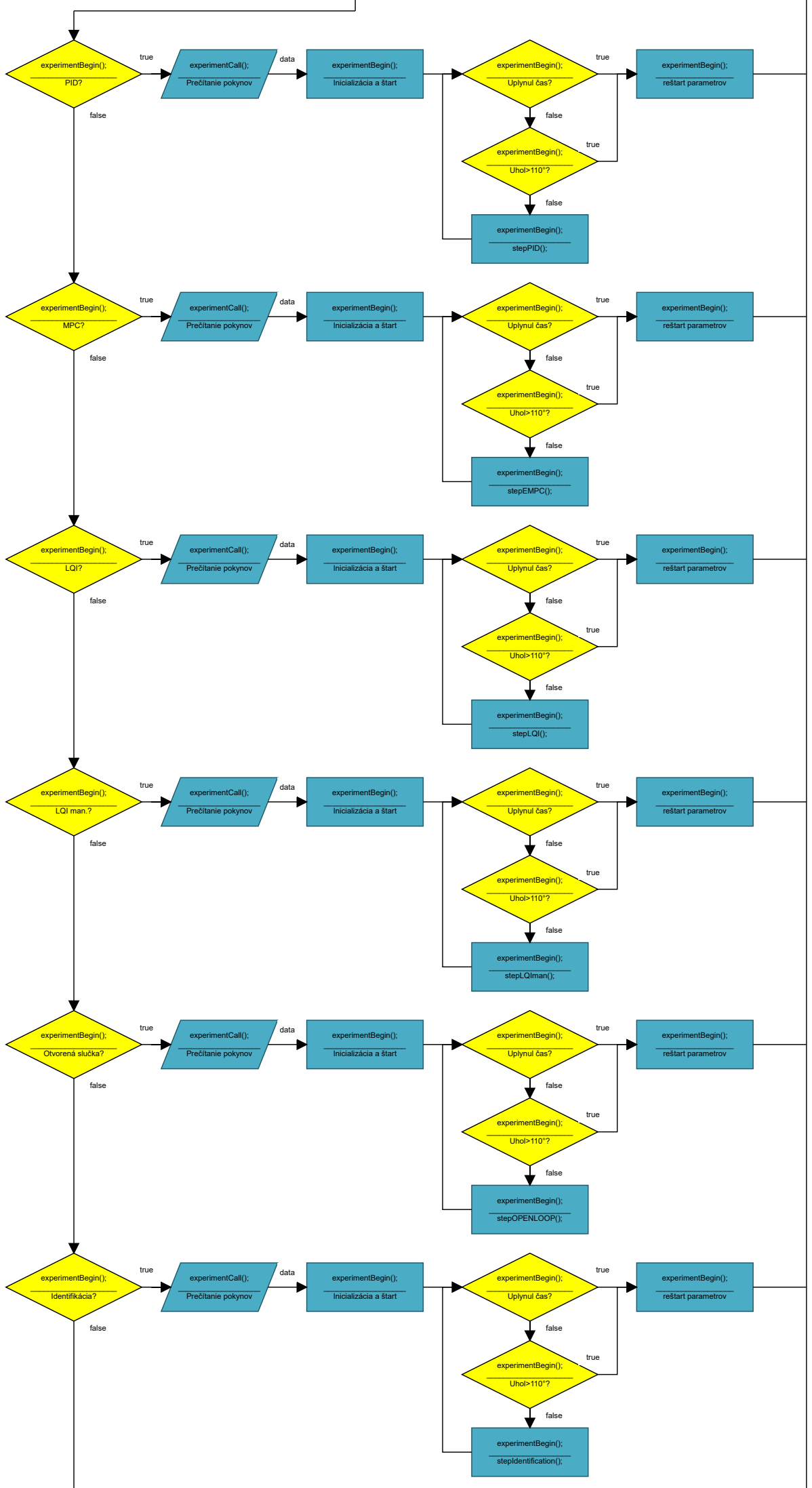
# A Prílohy

## A.1 Arduino INO Flowchart

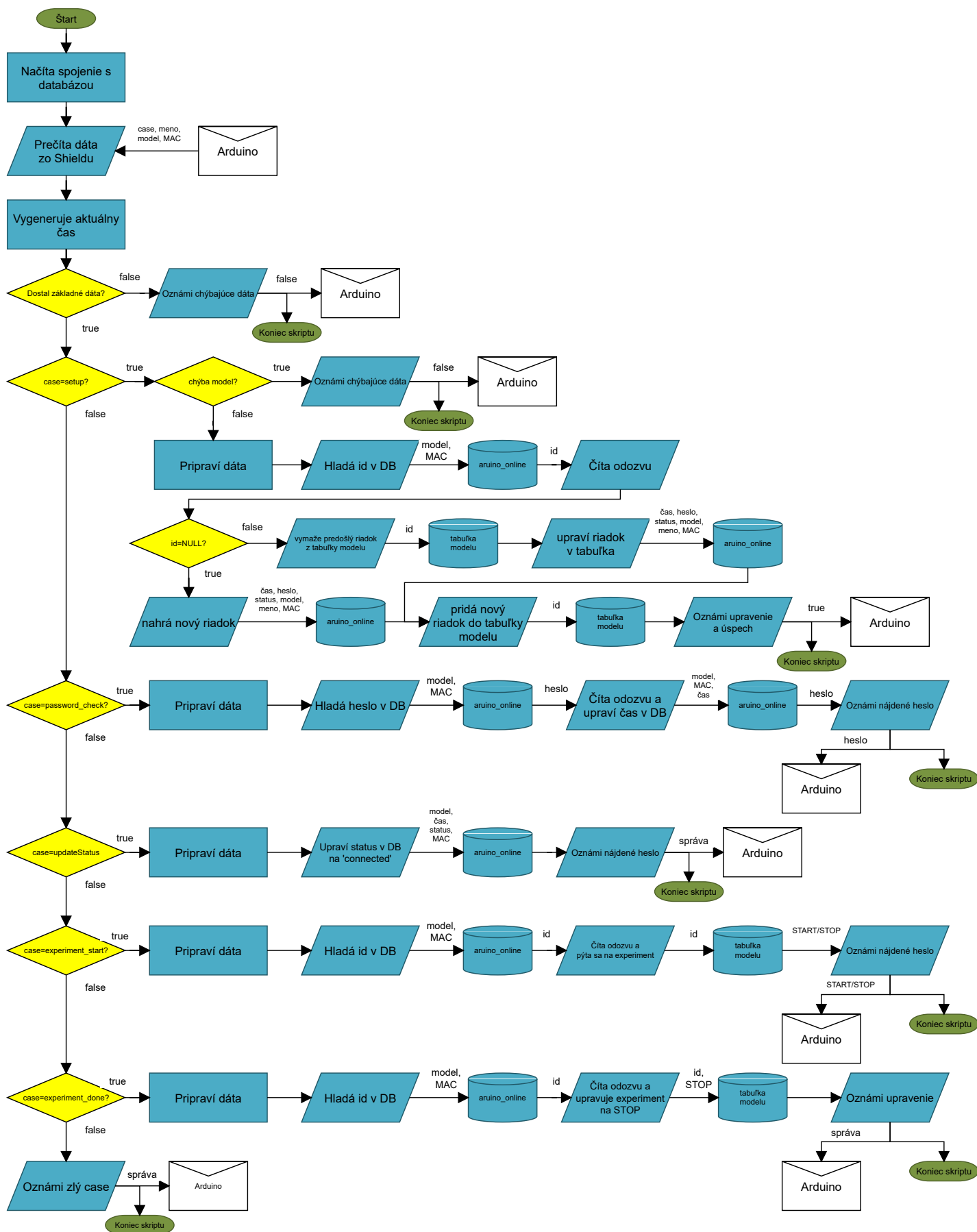




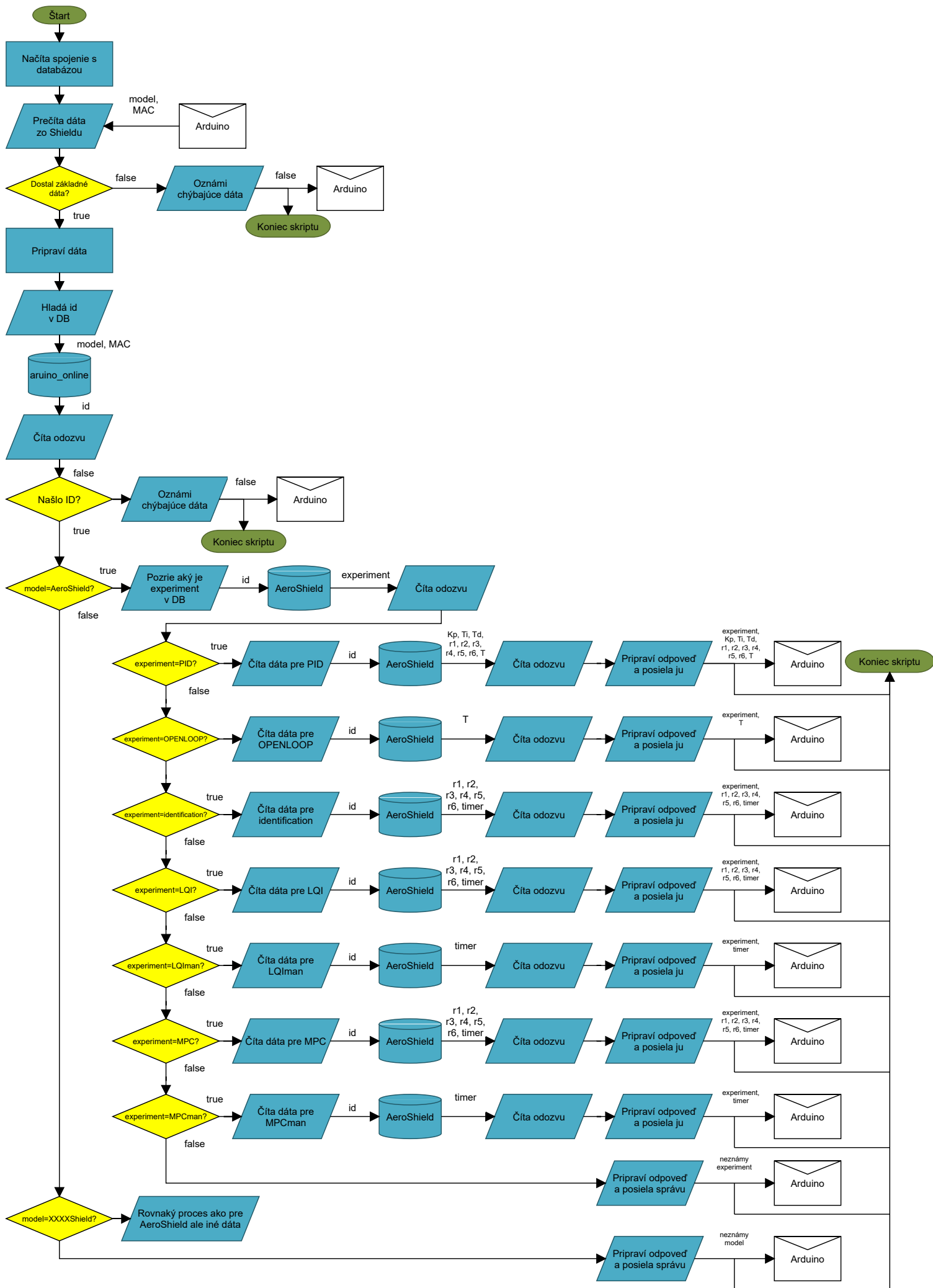
loop



## A.2 Arduino Online PHP Flowchart



### A.3 Arduino Experiment Data PHP Flowchart



## A.4 Flowchart of submitting experiment data to the DB

