Fitness systém založený na IoT

Analýza a návrh riešenia

Bc. Viktor Pristaš

1Im 2019 – 2020

Abstrakt. Internet vecí poznáme už niekoľko rokov a aj vďaka nemu sú mnohé činnosti vykonané oveľa efektívnejšie a jednoduchšie. Jedna z oblastí, v ktorej nájdeme ale málo inteligentných zariadení využívajúcich internet vecí, sú posilňovne.

V tejto práci sa venujeme návrhu, implementácií a testovaniu prototypu inteligentného systému, ktorý je schopný automaticky zistiť a zaznamenávať priebeh tréningu pri cvičení na posilňovacej veži so závažovými tehličkami. Tieto už zakúpené stroje budú jednoducho doplniteľné nami vyvinutým zariadením na počítanie opakovaní a zaznamenanie váhy.

Cieľom je poskytnúť podobný komfort ako ponúkajú voľne dostupné aplikácie pre smartfóny na sledovanie outdoorových aktivít ako chôdza alebo beh, pri ktorých používateľ len spustí tréning a o záznam sa postará samotná aplikácia.

Využívať pritom budeme jednoduché a lacné mikrokontroléry, akým je napríklad vývojová doska Arduino, čo zabezpečí, že systém bude cenovo oveľa výhodnejší než kúpa nového inteligentného stroja s podobnými možnosťami.

Kľúčové slová: mikrokontrolér, internet vecí, posilňovanie, Arduino

1. Úvod

Internet vecí (alebo IoT) je dnes bez pochýb jedna z najrýchlejšie vyvíjajúcich sa vetiev. Niet divu, veď ponúka nekonečné možnosti ako zjednodušiť, zrýchliť alebo zefektívniť naše každodenné činnosti. S IoT (a s často k nemu pridruženým prívlastkom „smart“) sa bežne stretávame na rôznych miestach akými sú napríklad domácnosti

(osvetlenie, biela technika, vysávač), mestá (doprava, odvoz odpadu), poľnohospodárstvo (zavlažovanie, postrekovanie) alebo energetika (spotreba plynu, elektriny, vody).

Najväčšou výhodou niektorých týchto systémov je, že dokážu vykonávať isté (hlavne manuálne) činnosti bez interakcie človeka. Kým pri klasickom vodomere je potrebné vyslať v pravidelných intervaloch pracovníka vodárne, aby odčítal spotrebu vody, inteligentné vodomery dokážu automaticky namerať a odoslať namerané údaje do centrálnej databázy. Navyše, pri tomto prístupe sa značne zníži aj pravdepodobnosť chyby.

Jedno z odvetví, ktoré ešte neprešlo touto revolúciou „zinteligentňovania“ sú fitness centrá. Sú síce dostupné fitness appky, no takmer všetky slúžia na sledovanie kardio cvičení pomocou senzorov v smartfónoch a smart hodinkách ako beh, cyklistika, plávanie a pod. Tieto aplikácie sú schopné uložiť dáta o vykonanej aktivite. Okrem toho umožňujú používateľovi sledovať jeho osobné pokroky a tým ho motivovať. Ďalšie aplikácie, ktoré môžeme nájsť pre smartfóny, slúžia na uchovanie manuálne zapísaných údajov o cvičení.

Ako už bolo spomenuté vyššie, cieľom IoT systémov a zariadení je samostatné a automatické vykonávanie istých činností. To znamená, že užívateľ by sa nemal starať o to, aby počítal a zaznamenával počet opakovaní a použitú váhu (ako ani pri behaní nepočíta svoje kroky a nemeria svoj čas), ale toto všetko má na starosti samotný systém. V práci sa zameriame na získanie práve týchto dvoch údajov – použitá váha a počet opakovaní – keďže práve tieto dáta obsahuje (samozrejme okrem samotného cviku) každý poctivý denník cvičiaceho.

2. Prehľad riešení

V roku 1965 Gordon Moore spoluzakladateľ firmy Intel napísal v jednom z jeho článkov, že počet komponentov na integrovanom obvode sa zdvojnásobí každý rok. O 10 rokov neskôr svoje tvrdenie poopravil a zdvojnásobenie predpokladal každý druhý rok. Jeho slová sa naplnili, dokonca fungujú až dodnes a túto jeho (vtedy len) myšlienku poznáme dnes ako Moorov zákon.

Pri tomto zákone je vhodné spomenúť aj Druhý Moorov zákon, taktiež nazývaný aj Rockov zákon, ktorý hovorí o exponenciálnom náraste nákladov na výskum a výrobu polovodičových komponentov. Podobne ako aj ten prvý, aj tento zákon je platný (a podľa predpovedí aj platným zostane) s relatívne vysokou mierou presnosti.

Napriek tomuto pesimisticky vyzerajúcemu trendu sa tieto rastúce vstupné náklady neodzrkadľujú na konečných cenách jednotlivých zariadení, keďže značne narastá aj celkový predaj mikročipov. Smartfóny, smart televízory, inteligentné domáce spotrebiče, hodinky či slúchadla sú už dneska bežnou vecou v živote aj priemerného spotrebiteľa aj domácnosti a tak, samozrejme, náklady na vývoj a výrobu čoraz menších a komplikovanejších mikročipov sa rozdelia na viacej častí, a cena zostane porovnateľná s cenou predchádzajúcej generácie.

Vďaka tomuto faktu, že výpočtové jednotky sú čoraz menšie a pritom ich cena výrazne nerastie, sa vývojári fitness aplikácií zameriavajú na konečných spotrebiteľov ktorí používajú fitness náramky, keďže tieto zariadenia sa stali extrémne populárnymi ako v každodennom živote, tak i v športovom alebo zdravotnom segmente.

K januáru 2021 nám nie je známy žiaden nástroj, ktorý by bol schopný dodať funkcionalitu trackovania cvikov klasickým fitness strojom. Existuje ale množstvo dostupných riešení pre spomínané wearables zariadenia (fitness náramky a inteligentné hodinky), ktoré vedia automaticky identifikovať vykonávané cvičenie. Spoločnosť Samsung umožňuje pomocou náramku a aplikácie Samsung Health[1] detekciu chôdze/behu, cyklistiky, veslovania alebo dynamického cvičenia. Väčšinou pri tom využívajú interné senzory gyroskop a akcelerometer alebo polohu podľa GPS.

Výrobcovia bežeckých pásov, veslovacích strojov alebo stacionárnych bicyklov mali najľahšiu úlohu pri automatickom zaznamenávaní cvičenia. Väčšina týchto strojov totiž zobrazuje dáta (aktuálna rýchlosť, obťažnosť a prejdená vzdialenosť) v reálnom čase a tak bolo pre nich jednoduchou úlohou spárovať stroj s niektorou fitness aplikáciou a pomocou nej uchovávať záznam cvičenia. Napriek jednoduchosti tohto riešenia ťažko nájdeme takéto stroje v bežných posilňovniach, svoje uplatnenie si našli skôr v domácnostiach.

So zaujímavým riešením prišla spoločnosť Tonal[2], ktorá vyvinula univerzálny cvičiaci nástroj, na ktorom je možné vykonávať viac ako 100 rôznych cvikov na posilňovanie. Týmto spôsobom je možné jednoducho nahradiť aj niekoľko bežne používaných posilňovacích strojov a sledovať pohyb cvičiaceho. Navyše ponúka aj neobmedzený počet používateľov, pre ktorých je možné uchovávať všetky údaje

[1] https://www.samsung.com/sk/apps/samsung-health/
[2] https://www.tonal.com/

o cvičení. K používaniu je potrebné mať v mobile stiahnutú aplikáciu Tonal, ktorá ale neposkytuje integrovanie do aplikácií tretej strany ako napríklad Google Fit alebo Strava. Aj vďaka tejto nevýhode – podobne ako to bolo s predchádzajúcimi strojmi – sa Tonal hodí viac do domácností ako do komerčných fitness centier.

Okrem toho posledného ani jedno z dostupných riešení na trhu sa nepribližuje k funkcionalite, ktorú sme chceli pokryť v tejto práci: automaticky zaznamenať cvičenie na jednoduchých strojoch, ktoré sú bezpochybne najpopulárnejšími zariadeniami (popri činkách) pri tradičnom posilňovacom tréningu.

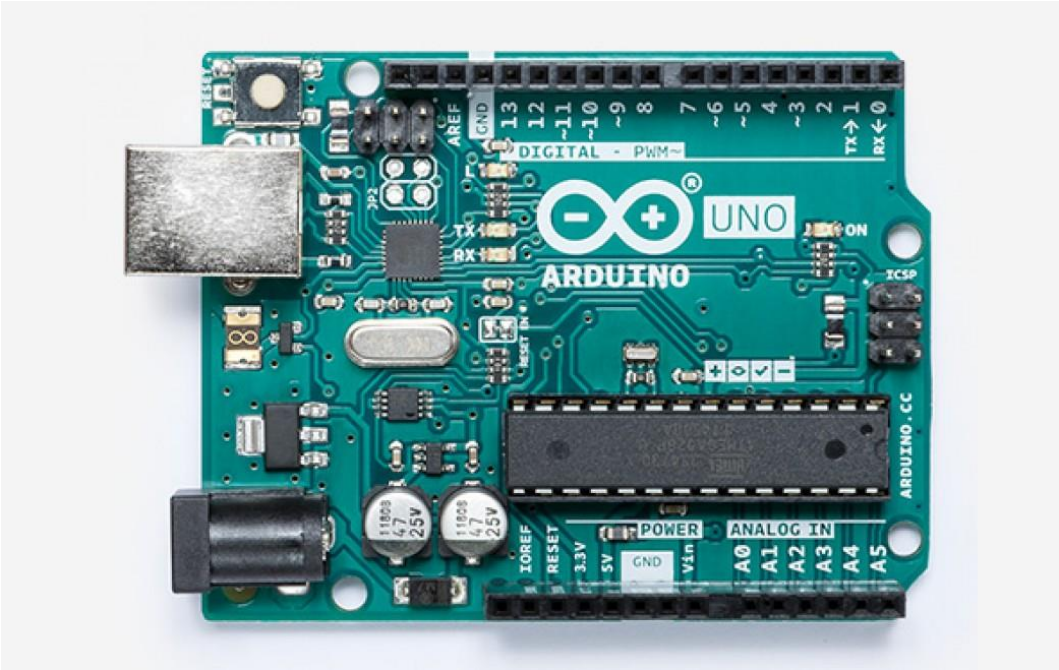
3. Mikrokontrolér a senzory

Základnou riadiacou jednotkou systémov IoT (ako aj mnohých iných zariadení) je malá jednotka nazývaná mikrokontrolér. Mikrokontroléry majú integrované v jadre všetky základné prvky, ktoré mikroprocesor potrebuje k svojej činnosti:

- výpočtové jadro
- pamäť RAM
- pamäť s programom (ROM, EPROM, FLASH)
- časovače
- vstupno-výstupné porty (piny)

Výkon týchto zariadení je oproti bežným stolovým počítačom značne nižší, bežne obsahujú procesor s taktovacou frekvenciou na úrovni 8 MHz a operačnú pamäť okolo 32 kB. Na druhej strane, týmto parametrom zodpovedá aj ich cena, ktorá sa pohybuje v intervale 2 – 25 eúr.

Najznámejším výrobcom vývojových dosiek je bezpochybne spoločnosť Arduino. Vďaka ich open-hardware filozofií (všetky technické údaje mikrokontroléra sú voľne dostupné pre kohokoľvek) sa vyrábajú a vyvíjajú mnohé napodobeniny, vďaka čomu klesla ich cena na spomínanú hladinu.



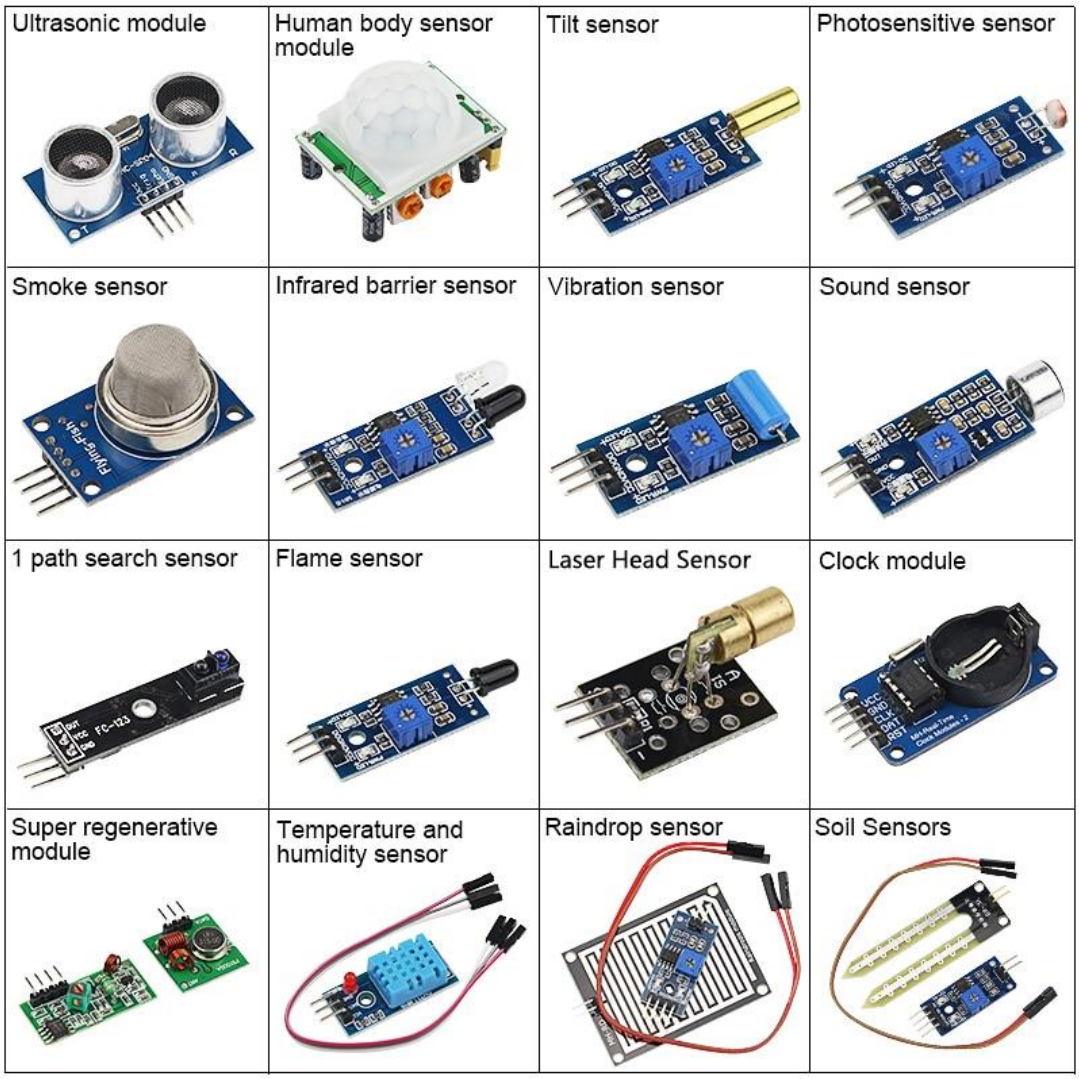
Obrázok 1 Najznámejšia doska od spoločnosti Arduino, model Uno

Väčšina dosiek Arduino používa mikročipy ATmega z rodiny 8-bitových AVR mikročipov postavených na základe RISC architektúry.

Na programovanie Arduino dosiek sa štandardne používa jazyk Arduino Programming Language, ktorého základom je jazyk C++. Je výhodné ho použiť, ak programátor nechce upravovať základné konfiguračné nastavenia, v opačnom prípade je možné použiť čistý C++. Takto získa programátor väčšiu kontrolu nad zariadením, a spravidla pamäťovo menej náročný kód za cenu ťažšieho programovania. Okrem jazyka Arduino Programming Language ponúka spoločnosť na prácu s (nie len Arduino) doskami inteligentné vývojové prostredie nazývané Arduino IDE.

Dosky Arduino (a aj iné vývojové dosky a mikrokontroléry) sú ľahko využiteľné na čítanie senzormi nameraných údajov. Senzory sú súčiastky schopné merať nejakú fyzikálnu veličinu a pretransformovať tieto dáta na elektrický signál, ktorý je možné prenášať a spracovať.

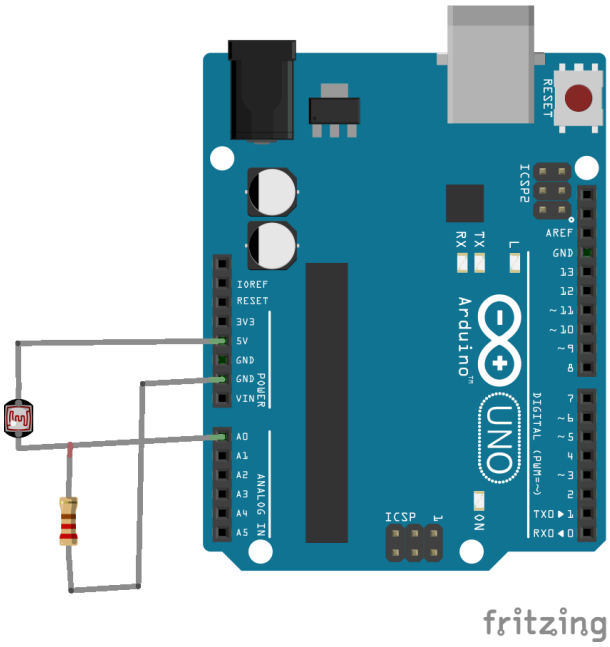
Na nasledujúcom obrázku sú znázornené najčastejšie používané senzory s Arduino doskami.



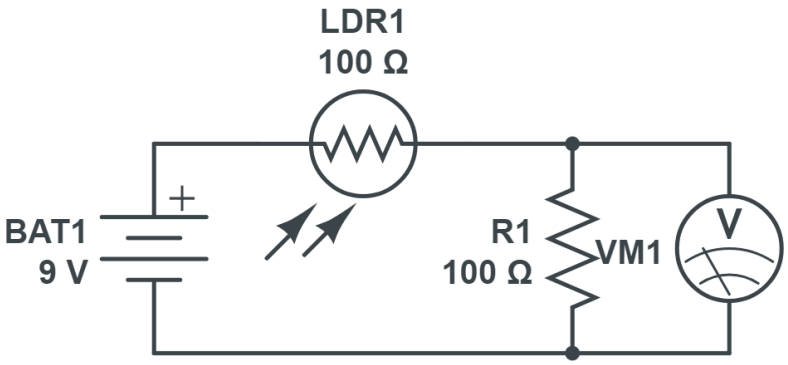
Obrázok 2 Niekoľko najčastejšie používaných senzorov

Ako bolo už vyššie spomenuté, mikrokontroléry sú vybavené vstupno-výstupnými pinmi (angl. general-purpose input/output, skratka GPIO). Tieto porty nemajú vopred definovanú funkcionalitu, je na programátorovi aby ich nakonfiguroval vzhľadom na použitie (prípadne prenechal konfiguráciu na vývojové prostredie). Ak je pin nastavený na vstupný, je schopný merať veľkosť napätia v obvode, v prípade analógového pinu jeho presnú hodnotu a v prípade digitálneho len jeho prítomnosť. Pin konfigurovaný ako výstupný dokáže generovať napätie 0 V alebo 5 V (pri istých doskách 3,3 V) ktoré zodpovedajú digitálnym hodnotám 0 a 1.

Ako ilustračný príklad je vhodné použiť jeden z najjednoduchších používaných senzorov – fotorezistor.



Obrázok 3 Zapojenie fotorezistoru k Arduinu



Obrázok 4 Schéma zapojenia fotorezistoru do obvodu

Na týchto schémach zapojenia vidno, ako sa zapája fotorezistor k analógovému pinu A0 mikrokontroléra. Mikrokontrolér meria v skutočnosti napätie medzi pinom A0 a pinom GND nad rezistorom R1 ako to vidno na pravom obrázku. Rezistor R1 je v zapojení potrebný, pretože základnou vlastnosťou voltmetra je, že sa zapája do obvodu paralelne. Meniacou sa intenzitou svetla sa mení aj napätie nad rezistorom R1 a teda nameria mikrokontrolér rôzne hodnoty.

4. Riešenie z pohľadu hardvéru

Na zaznamenanie cvičenia, ako bolo spomenuté skôr, je potrebné sledovať okrem toho, aký cvik sa cvičil, dva základné atribúty:

- koľko opakovaní sa vykonalo
- akou váhou sa cvičilo

V práci sa zameriame na tzv. posilňovacie veže vybavené závažovými tehličkami, ktoré sa dajú jednoducho pripnúť pomocou kolíka.

Obrázok 5 Posilňovacia veža

4.1 Počítanie opakovaní

Počet opakovaní je z implementačného hľadiska počet koľkokrát sa tehličky zdvihli smerom hore a vrátili zas naspäť. Ich smer pohybu je jednoduchý: vertikálne smerom hore, následne vertikálne smerom dole. Počas analýzy sme skúmali niekoľko možných ciest ako detegovať tento pohyb a snažili sme sa nájsť čím presnejšiu no najmenej poruchovú.

4.1.1 Akcelerometer

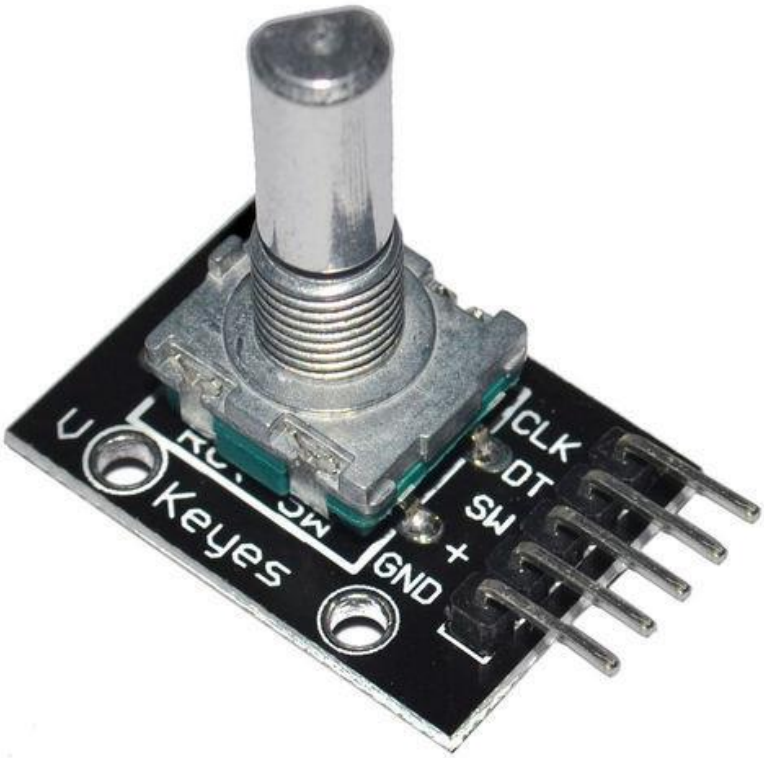
Z pohľadu senzorov je takýto pohyb najjednoduchšie merateľný akcelerometrom. Akcelerometer umiestnený na vrchu závaží je schopný detegovať smer pohybu a tým aj spočítať počet dvihnutí a položení. Okrem toho dokáže zmerať aj zrýchlenie, čo sa javí ako zaujímavý údaj na neskoršiu analýzu cvičenia (dynamika pri opakovaniach), ale ak nie je pohyb dostatočne plynulý alebo je príliš rovnomerný, môžu vzniknúť značné skreslenia a nepresné hodnoty.

4.1.2 Ultrazvukový senzor vzdialenosti

Iný pohľad na tento pohyb je, že tehličky sa každým dvihnutím priblížia ku vrchu posilňovacej veže a naopak pri položení sa oddialia. Toto pozorovanie evokuje možnosť použitia ultrazvukového senzoru vzdialenosti. Ten, umiestnený presne nad závažiami by meral vzdialenosť od vrchnej tehličky. Nevhodnosť tejto myšlienky spočíva v nepresnosti týchto senzorov pri prekročení meranej vzdialenosti niekoľkých desiatok centimetrov.

4.1.3 Rotačný enkodér

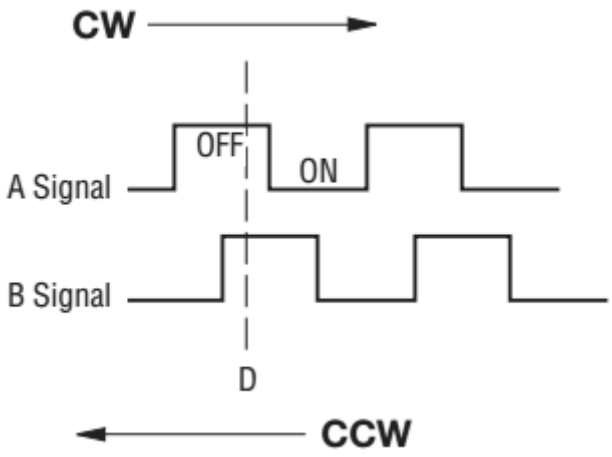
Ďalšou skúmanou možnosťou je sledovať otáčanie kladky, na ktorej sú zavesené závažia. Smer pohybu závaží je jednoznačne identifikovateľný smerom otáčania. Na zistenie otáčania slúži napríklad súčiastka zvaná rotačný enkodér (rotačný snímač, angl. rotary encoder). Je to súčiastka, ktorá umožňuje zistiť polohu pri otáčaní, stretnúť sa s ňou môžeme pri autorádiách alebo pri guľôčkových myšiach. Nemá krajnú polohu, to znamená, že je ním možné otáčať do oboch strán ľubovoľne veľakrát.



Obrázok 6 Rotačný enkodér

Pri otáčaní enkodéra dochádza k pravidelnej zmene dvoch digitálnych signálov, ktoré sú voči sebe mierne posunuté. Pri otáčaní do smeru hodinových ručičiek je možné

namerať na kanáloch (A,B) hodnoty v nasledujúcom poradí: (0, 0), (1, 0), (1, 1), (0, 1), (0, 0)... Analogicky budú tieto hodnoty pri opačnom smere v opačnom poradí. Týmto spôsobom sa dá ľahko detegovať nielen otáčanie ale taktiež jeho smer.

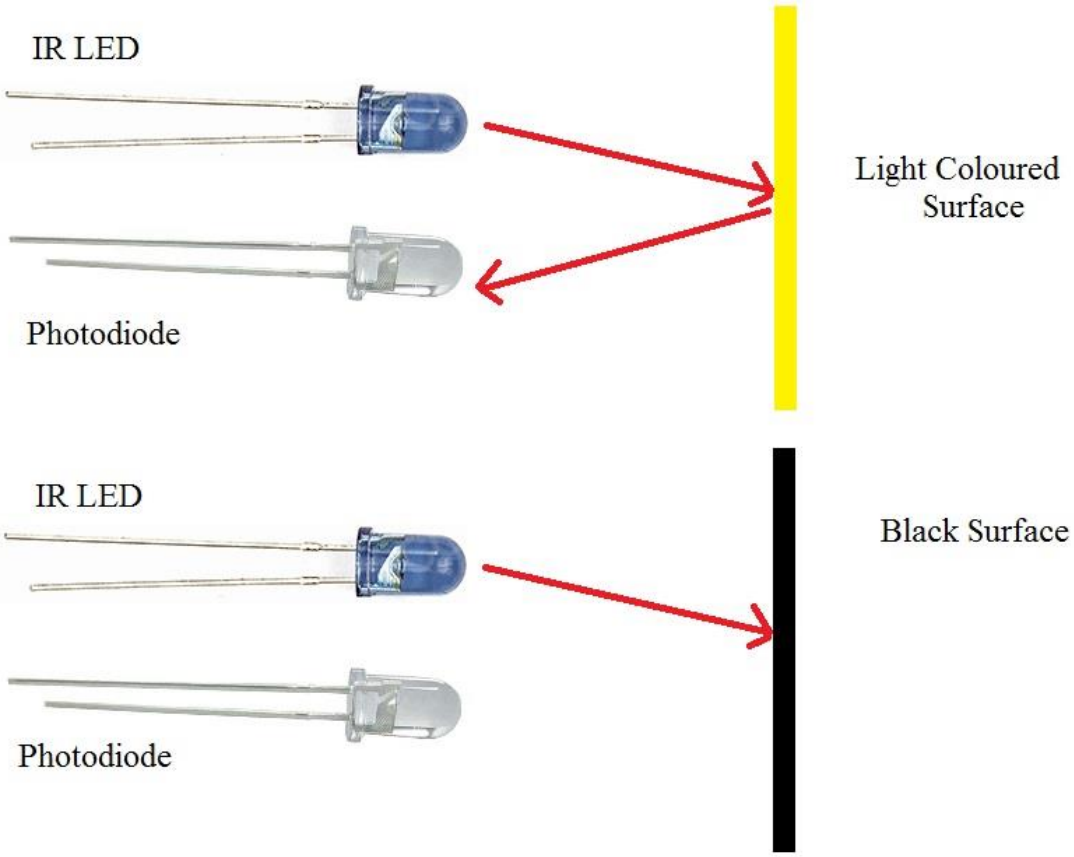


Obrázok 7 Fungovanie rotačného enkodéra

Problém pri tomto riešení spočíva v tom, že táto súčiastka musí byť v priamom kontakte s mechanickými, rotujúcimi časťami stroja, čo môže ľahko viesť k ich rýchlemu opotrebeniu alebo kazeniu. Hardvérová časť tohto systému musí byť odolná voči silným otrasom, aby sa minimalizoval počet potenciálne poruchových miest, čo ale toto riešenie nespĺňa.

4.1.4 Optický rotačný enkodér

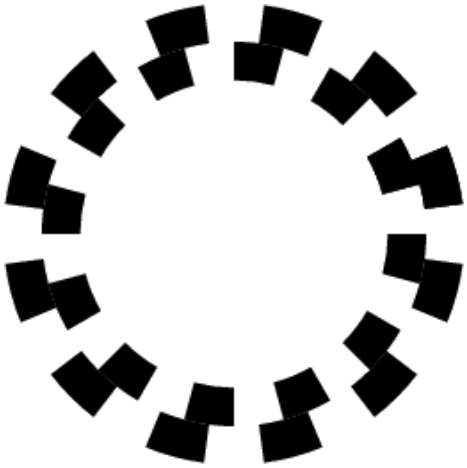
Optický rotačný enkodér funguje podobne ako vyššie predstavený mechanický, pričom pri jeho otáčaní nedochádza k fyzickému kontaktu so súčiastkou. Jeho hlavným hardvérovým komponentom je senzor infračerveného svetla, aký sa často používa v robotike ako senzor sledovania čiary. Princíp jeho fungovania spočíva v tom, že senzor vyžaruje infračervené svetlo a meria, koľko svetla sa odráža späť.



Obrázok 8 Infračervený senzor

Čierna farba (ako je dobre známe) pohlcuje svetlo, teda sa od neho odrazí oveľa menej ako od povrchov inej farby. Senzor musí byť umiestnený na cca. 1-3 cm od snímanej plochy a v závislosti od množstva odrazeného svetla vráti hodnotu v rozmedzí 0-1023 (ako je to obvyklé pri analógových senzoroch Arduina).

V praxi je potrebné nasledovať fungovanie mechanického rotačného enkodéra, t. j. vytvoriť dva signály, ktoré budú mierne posunuté. Toto vieme zabezpečiť tým, že dva takéto senzory budú snímať vytvorenú čierno bielu vzorkovnicu v tvare kruhu.



Obrázok 9 Vzorkovnica v tvare kruhu

Jeden zo senzorov bude snímať vonkajšie sekvenciu čierno bielych políčok, pričom ten druhý tú vnútornú. Nie je ťažké vidieť, že signál (po prevedení analógových hodnôt na digitálne) generovaný týmito senzormi sa zhoduje so signálom aký bol na obrázku 7.

Táto vzorkovnica umiestnená na kladke spolu so senzormi vo vhodnej vzdialenosti značne vylepšuje predošlé riešenie, keďže pri ňom nedochádza k otáčaniu/pohybu senzora a tak je odolnejší voči fyzickému poškodeniu.

4.2 Počítanie váhy

Druhým základným problémom v tejto práci je zistenie váhy akou sa vykonali opakovania. Meranie samotnej hmotnosti je v tomto prípade takmer nemožné, keďže váhy ako senzory pre mikrokontroléry sa nepoužívajú. Riešením by mohlo byť použitie silomera, ktorý by mohol do istej miery nahradiť funkciu váhy, no má značné obmedzenia. Po prvé je náročné jeho umiestnenie na vhodné miesto, aby nebránilo cvičiacemu pri manipulovaní so závažiami, po druhé, tieto prístroje potrebujú pravidelnú kalibráciu. Aj pri bežnom cvičení sa používajú závažia v rozmedzí cca. 10 – 120 kg, čo by ľahko mohlo spôsobiť rozladenie senzora, navyše, keďže ide o relatívne veľké hmotnosti, jeho životnosť (pri zachovaní priaznivej ceny, čo je tiež jeden z cieľov tohto systému) výrazne klesá.

Ak sa pozrieme na problém z inej stránky, to, čo v skutočnosti potrebujeme detegovať, je miesto, kde sa oddelili od seba tehličky. Teda ak zistíme, že k oddeleniu

došlo medzi tehličkami *i* a *i+1*, to znamená, že sa cvičí s prvými i tehličkami. Poznajúc ich nemenné hmotnosti sa jednoducho vypočíta ich celková hmotnosť.

4.2.1 Ohmov zákon

Ohmov zákon definuje vzťah medzi elektrickým prúdom, elektrickým odporom a elektrickým napätím. Jeho definícia znie nasledovne:

Elektrický prúd pretekajúci v uzavretom elektrickom obvode je priamo úmerný napätiu zdroja a nepriamo úmerný elektrickému odporu obvodu.

V matematickom zápise:

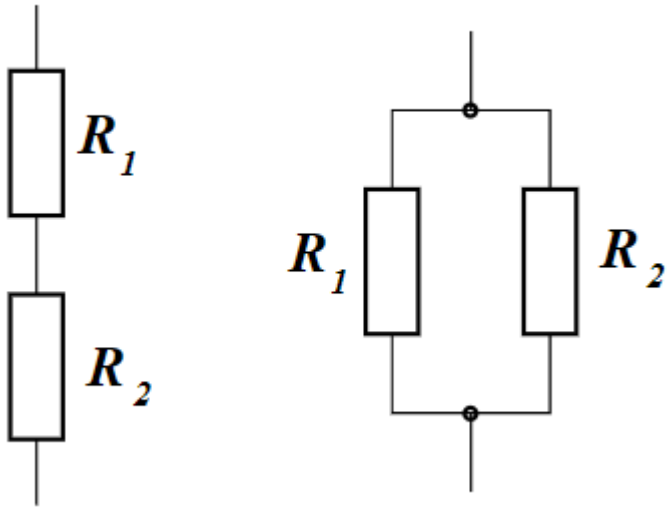
$$I = \frac{U}{R}$$

kde I je veľkosť prúdu, U veľkosť napätia a R veľkosť rezistencie.

Laickejšou rečou by sme mohli jednoducho povedať, že pri konštantnom napätí, s rastúcim odporom klesá veľkosť prúdu v obvode. Teda využijúc schopnosť Arduina merať elektrické napätie vieme zistiť veľkosť (a tým aj zmenu) odporu ktorá je obvode (keď veľkosť vychádzajúceho prúdu a tiež napätia z jeho pinov je konštantná).

4.2.2 Paralelné a sériové zapojenie odporov v obvode

Paralelné zapojenie je zapojenie odporov (alebo iných súčiastok) v elektrickom obvode pod sebou. Na rozdiel od sériového zapojenia, pri ktorom sú súčiastky zapojené v jednom rade za sebou, paralelný obvod obsahuje uzly, v ktorých sa vodiče vetvia a odpory sú umiestnené v rôznych vetvách.



Obrázok 10 Sériové a paralelné zapojenie odporov

Výsledný odpor sériovo zapojených odporov R_1, ... R_n je rovný súčtu ich odporov, teda:

$$R = \sum_{i=1}^{n} R_i = R_1 + R_2 + \cdots + R_n$$

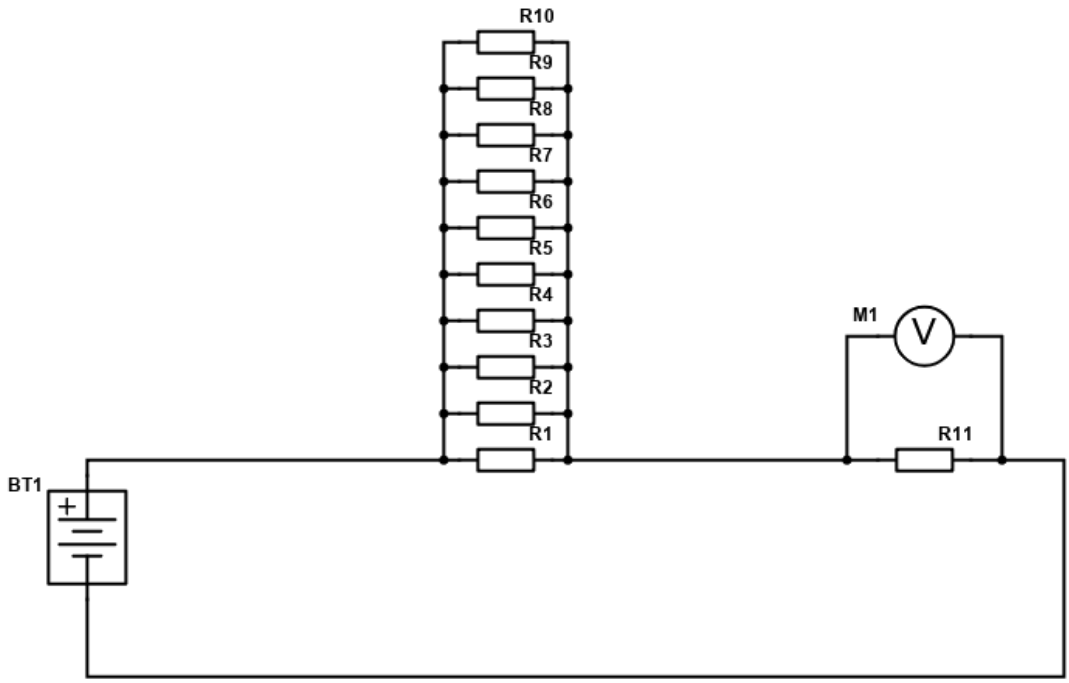
Z Ohmovho zákona je možné odvodiť aj súčet paralelne zapojených odporov R_1, ... R_n pre ktorý platí, že prevrátená hodnota výsledného odporu je daná ako súčet prevrátených hodnôt jednotlivých odporov:

$$\frac{1}{R} = \sum_{i=1}^{n} \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_n}$$

Výsledný odpor pri paralelnom zapojení je teda stále menší ako odpor ktorejkoľvek vetvy.

4.2.3 Spočítanie tehličiek pomocou rezistorov

Znalosti o Ohmovom zákone a výpočte výsledného odporu paralelných odporov využijeme na zistenie počtu dvíhaných tehličiek nasledovne. Na každú tehličku pripojíme rezistor nejakej veľkosti, pričom ich zapojíme do obvodu paralelne. Schému zapojenia pre 10 závažových tehličiek zobrazuje nasledujúci obrázok:



Obrázok 11 Schéma zapojenia odporov

Odpory musia byť zapojené tak, aby v prípade, keď sú tehličky položené na mieste, existovalo medzi rezistormi elektrické spojenie. Naopak ak je horných *n* tehličiek vo vzduchu, teda sa nimi cvičí, tak sú v obvode zapojené len všetky ostatné rezistory, v modelovom príklade spodných *10 – n* odporov.

Ak sú teda známe veľkosti rezistorov, je možné podľa predošlého vzorca vypočítať ich výsledný odpor. Voltmeter M1 znázornený na obrázku 11 je v skutočnosti nami používaný mikrokontrolér, ktorý ako už bolo spomenuté vyššie je schopný merať vchádzajúce napätie. Odpojením niekoľkých odporov z vrchu zapojenia sa zmení výsledný odpor, a tým takisto aj napätie merané mikrokontrolérom.

4.2.3.1 Rozloženie odporov

Na to aby sme vedeli zistiť presný počet odporov je potrebné poznať ich presné hodnoty. V praxi sa budeme snažiť aby postupným odoberaním odporov od vrchu klesal ich výsledný odpor čo najlineárnejšie.

Zaveďme si označenie pre modelový príklad s 10 paralelne zapojenými rezistormi R_1, ... , R_{10} ako je to vidno na obrázku. Nech R_i' je hodnota výsledného odporu spomínaných 10 odporov, ak sú v obvode postupne zdola zapojené odpory $R_1, R_2, \dots, R_{10-i}$.

- $\frac{1}{R_0'} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_{10}}$
- $\frac{1}{R_1'} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_9}$
- $\frac{1}{R_2'} = \frac{1}{R_1} + \frac{1}{R_2} + \cdots + \frac{1}{R_8}$
- ...

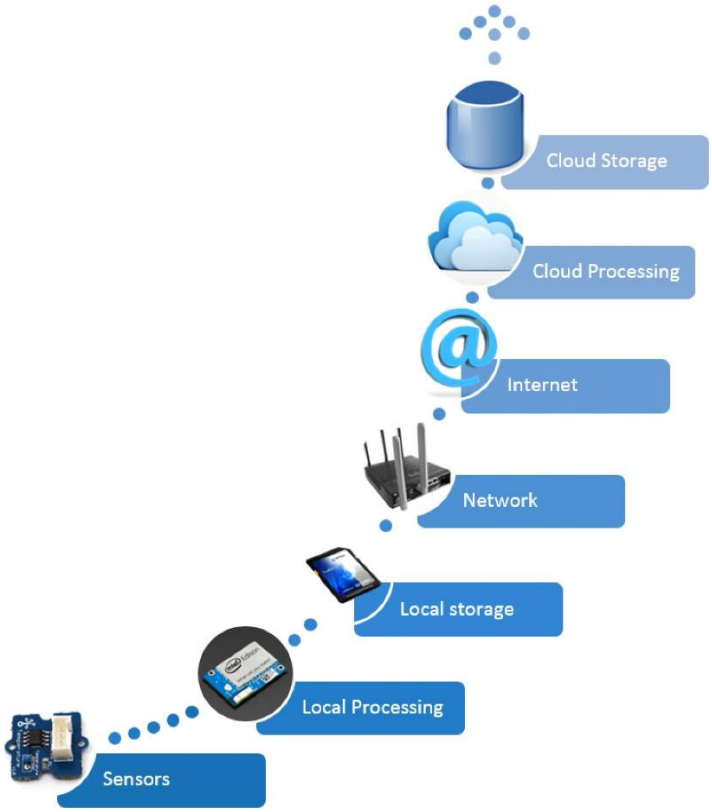
Nech pre $i \in \{2, 3, \dots, 10\}$ označuje premenná R_i^k rozdiel $R_i^k - R_{i+1}^k$, kde k je kód konfigurácie odporov. Pre všetky rôzne konfigurácie k nájdeme najmenšiu hodnotu zo všetkých R_i^k a označíme ju R_{min}^k. Cieľom je nájsť konfiguráciu s maximálnou hodnotou pre R_{min}^k, keďže v tej konfigurácií majú hodnoty R_i^k najmenší rozptyl.

5. Implementácia

V tejto kapitole sa zameriame detailnejšie na implementáciu z pohľadu mikrokontrolérov. Keďže táto práca je postavená na internete vecí, je dobré dodržať pri návrhu niekoľko pravidiel ktoré zabezpečujú efektívne využitie všetkých výpočtových zariadení, siete a senzorov.

5.1 Vrstvenie Internetu Vecí

Internet vecí je komplexný systém pozostávajúci z rôznych zariadení, ktoré majú svoju hierarchickú štruktúru. Nasledujúci obrázok zobrazuje ideálne vrstvenie IoT systému ako prúdia dáta od najjednoduchších senzorov po cloudové spracovanie a úložisko.



Obrázok 12 Vrstvenie ideálne navrhnutého IoT systému

Na spodných vrstvách systému sa nachádzajú senzory, lokálne spracovanie a lokálne úložisko (angl. Sensors, Local processing, Local storage). Tieto vrstvy tvoria jadro Internetu Vecí. Ako bolo už v predošlej kapitole popísané, mikrokontrolér vďaka svojim vstupno-výstupným rozhraniam a integrovanej pamäti je schopný komunikovať so senzormi, získavať z nich dáta, spracovávať a uchovávať ich v značne obmedzenom množstve pre ich prípadnú lokálnu analýzu.

Keďže ide o internet vecí, je na mieste uvažovať o zdieľaní dát s inými mikrokontrolérmi alebo serverom. Toto má za úlohu sieťová vrstva a internet (angl. Network, Internet). Na vrchu tejto štruktúry sa nachádza Cloudové spracovanie a úložisko (angl. Cloud processing, Cloud storage), ktoré fungujú podobne ako tie lokálne s tým rozdielom, že tú máme k dispozícií mnohonásobne väčšiu výpočtovú silu a úložisko na úrovni terabajtov oproti niekoľkým megabajtom.

V mnohých projektoch sa ale napriek tejto osvedčenej schéme vynecháva vrstva lokálneho spracovania a úložiska a všetky údaje získané zo senzorov sa posielajú priamo na server na cloudové spracovanie. Toto môže byť v niektorých prípadoch veľmi

neefektívne. Okrem toho, že môže spôsobiť zbytočné zaťaženie siete posielaním irelevantných dát, môže spôsobiť aj oneskorenú reakciu.

Ako ilustračný príklad týchto problémov by som uviedol autonómne vozidlo. Bezpochybne sa naň môžeme pozerať ako na IoT zariadenie, aj keď je v porovnaní s tými bežnými niekoľkonásobne drahšie, stále pozostáva zo stoviek senzorov a má pripojenie k sieti. V bežnej prevádzke generuje takéto vozidlo aj 5TB dát za hodinu. Ak by všetky vozidlá mali posielať takéto množstvo dát, internetová infraštruktúra by s tým mala značné problémy. Okrem toho, ak by vozidlo nevedelo konať na základe lokálnych rozhodnutí a na každý úkon by muselo čakať na odpoveď servera, v krízových situáciách keď je každá stotina sekundy dôležitá by zbytočné zdržanie mohlo mať vážne následky.

5.2 Vrstvenie nášho riešenia

Prvým cieľom tejto nami vytvoreného systému je zber údajov z dvoch senzorov: z optického rotačného enkodéra a zo série paralelne zapojených rezistorov. Na túto úlohu sme vybrali vývojovú dosku Arduino Nano vyrobenú už spomínanou spoločnosťou Arduino. Poskytuje dostatočné množstvo voľných pinov (8 analógových a 14 digitálnych) ako aj procesor s postačujúcou taktovacou frekvenciou 16 MHz.

V ďalšom kroku je potrebné toto zariadenie pripojiť k internetu, na čo máme hneď niekoľko možností. Prvou možnosťou je pripojenie externého modulu (tzv. shieldu) k vývojovej doske Arduino Nano, ktorý poskytuje káblové pripojenie do siete Ethernet. Toto riešenie ponúka síce relatívne spoľahlivé a rýchle pripojenie, no v našom prípade by bolo zabezpečenie Ethernetového kábla pri každom stroji v posilňovni nepraktické pri inštalácií.

Ďalšou možnosťou je pripojenie WiFi modulu k vývojovej doske. Pri tomto riešení by síce zmizol problém so zabezpečením káblového pripojenia, na druhej strane tieto zariadenia používajú firmvér rôzny od firmvéru Arduina, a preto je ich prvotné spustenie a vzájomná komunikácia komplikovaná.

Kvôli nevýhodám predošlých dvoch riešení sme sa rozhodli použiť tretiu možnosť ktorou je pridanie ďalšej vývojovej dosky NodeMCU s čipom ESP8266 od spoločnosti Espressif, ktorá poskytuje možnosť bezdrôtového pripojenia do siete pomocou WiFi. Eleganciou tohto riešenia je rozdelenie logiky medzi dva nezávislé uzly. Kým prvý z nich

– Arduno Nano – má na starosť čítanie dát zo senzorov, ten druhý obsluhuje pripojenie do siete a komunikáciu so serverom.

6. Komunikačné protokoly

Komunikačný protokol je vopred definovaný štandard na základe ktorého pravidiel si jednotlivé zariadenia posielajú dáta.

Pozorným čitateľom možno neušlo, že pri riešení s dvomi vývojovými doskami predstavenom v predošlej kapitole je potrebné uvažovať hneď dva komunikačné protokoly:

- komunikácia medzi doskami Arduino Nano a NodeMCU
- komunikácia medzi NodeMCU a serverom

Komunikačné protokoly medzi mikrokontrolérmi využívajú väčšinou sériovú komunikáciu, teda dáta sa posielajú za sebou (na rozdiel od paralelnej komunikácie, pri ktorej sa dáta môžu posielať v tom istom čase) a sú charakterizované atribútmi ako rýchlosť, synchrónny/asynchrónny, duplexný a pod.

Rýchlosť komunikácie je daný v bitoch za sekundu a bežné hodnoty pri najpoužívanejších protokoloch sú v medziach od 10 000 bit/s do 10 000 000 bit/s. Rýchlosť komunikácie je dôležitým atribútom ak sa posiela veľké množstvo dát a pomerne často, v našom prípade je to najmenej zaujímavý faktor.

Pri synchrónnych protokoloch je potrebné aby obe strany – teda odosielateľ aj prijímateľ – mali spoločný synchronizovaný čas. Väčšinou sa vyberie jedno zo zariadení, ktoré má za úlohu generovanie a distribúciu časového signálu na základe ktorého komunikujú aj ostatné zariadenia. Pri asynchrónnej komunikácií nie je potrebné aby si komunikujúce strany synchronizovali časový signál, stačí iba aby mali vopred definovanú prenosovú rýchlosť, teda počet prenesených bitov za sekundu.

Takmer všetky známe protokoly majú možnosť obojsmernej komunikácie, čo znamená, že obe strany vedia aj posielať aj prijímať dáta. Pri takejto obojsmernej komunikácií rozlišujeme či ide o plný duplex (angl. full duplex) kde komunikujúce strany dokážu prenášať dáta oboma smermi súčasne alebo polovičný duplex (angl. half duplex) pri ktorom v danom okamihu môže zariadenie buď posielať alebo prijímať dáta.

Typickým príkladom half duplexu môže byť walkie talkie vysielačka ktorá pri vysielaní nedokáže naraz aj prijímať.

Ďalším zaujímavým atribútom pre charakterizáciu komunikačného protokolu je počet pinov (počet káblov medzi zariadeniami) potrebných na komunikáciu. Keďže vývojové dosky majú obmedzený počet vstupno-výstupných rozhraní (pri niektorých to môže byť dokonca aj menej ako 10) v istých prípadoch je potrebné obmedziť výber na taký, ktorý ich využije čo najmenej.

6.1 Prehľad protokolov UART, SPI a I2C

	UART	**SPI**	**I2C**
Synchrónny	Nie	Áno	Áno
Priepustnosť	Do 115 200 bit/s	Do 10 000 000 bit/s	10 000 – 1 000 000 bit/s
Počet pinov	2	4	2
Duplex	Full-duplex	Full-duplex	Half-duplex

6.2 MQTT

Komunikačný protokol MQTT (Message Queuing Telemetry Transport) je jeden z najpoužívanejších protokolov. Je typu M2M (machine to machine), čiže určený na vzájomnú komunikáciu medzi strojmi, pričom väčšinou využíva protokol TCP/IP (alebo akýkoľvek iný sieťový protokol poskytujúci bezstratovú, obojsmernú komunikáciu). MQTT je jednoduchý a nenáročný, preto je použiteľný pri zariadeniach s nízkym výkonom (akými sú aj mikrokontroléry), pričom je navrhnutý tak, aby bol energeticky úsporný a spoľahlivý aj na miestach s pomalým internetovým pripojením. Prvýkrát bol použitý na monitorovanie a riadenie ropovodu v púšti na konci 20. storočia. Neskôr, v roku 2013, ho prerobila spoločnosť IBM.

Protokol funguje na princípe poskytovateľ – odberateľ (angl. publisher – subscriber) a rozlišuje dve hlavné entity: brokera a klienta. Broker je centrálny bod v sieti s ktorým priamo komunikujú klienti dvoma spôsobmi: publikujú (poskytujú) správy alebo ich prijímajú (odoberajú). MQTT protokol je tematický, čiže správy sú priradené do tém (angl. topic) podľa nejakej logickej štruktúry.

Ak napríklad jeden z klientov publikuje správu týkajúcu sa témy *teplota*, broker zabezpečí aby sa táto správa dostala ku všetkým ostatným klientom ktorí sú prihlásení na odoberanie správ v topicu *teplota*. Klienti môžu byť prihlásení k viacerým témam, a pokiaľ k danej téme nie sú prihlásení žiadni odoberatelia, broker zahodí správu, ak nie je definované inak.

Témy sú hierarchicky štruktúrované, pričom jednotlivé úrovne štruktúry sú oddelené lomkou: *Medická/posilňovňaA/miestnosťA1/stroj2/zariadenie0*. Názvy tém sú kódované v UTF-8, takže je možné v nich používať aj diakritiku. Pri prihlásení k téme je možné použiť dva špeciálne znaky: „+“ a „#“. Znak „+“ umožňuje nahradiť jednu úroveň v hierarchií, napríklad prihlásenie sa k téme *Medická/posilňovňaA/+/stroj2/zariadenie0* by znamenalo získanie všetkých správ od zariadení č. 0 na stroji č. 2 vo všetkých miestnostiach v posilňovni A. Znak „#“ nahradzuje jednu alebo viacero nižších úrovní v hierarchií a preto sa píše na koniec reťazca. Ak sa teda klient prihlási k téme *Medická/posilňovňaA/miestnosťA1/#*, znamená to, že očakáva správy týkajúce sa všetkých zariadení každého stroja v miestnosti A1.

Obsah samotnej správy nemá nijaké špecifikácie ani obmedzenia. Odosielať sa môžu binárne údaje alebo textové reťazce ktoré sú najčastejšie vo formáte JSON, BSON alebo XML. Obmedzená je ale veľkosť správy, ktorá nemôže presiahnuť 256 MB a jej minimálna veľkosť je 2 B.

MQTT podporuje aj možnosť potvrdzovania správy. Na základe dôležitosti správy je možné vybrať z troch úrovní QoS (Quality of Service):

- QoS 0 (at most once) – každá správa je odoslaná práve raz a ani klient ani broker nekontrolujú doručenie správy
- QoS 1 (at least once) – pri neúspešnom doručení sa správa odošle znovu niekoľkokrát kým prijímateľ nepotvrdí doručenie
- QoS 2 (exactly once) – odosielateľ a prijímateľ nadviažu spojenie metódou 2-way handshake aby sa zabezpečilo doručenie správy práve raz

Protokol MQTT obyčajne posiela správy ako nešifrovaný text. Šifrovanú komunikáciu a autentifikáciu je možné zabezpečiť použitím protokolu SSL/TLS, pri ktorom sa namiesto bežne používaného portu 1883 použije port 8883 alebo 8884.

Klient nadväzuje spojenie s brokerom správou CONNECT ktorú broker príjme a potvrdí správou CONNACK. Na prihlásenie sa k odberu nejakej témy slúži správa

SUBSCRIBE, na odhlásenie sa z odoberania UNSUBSCRIBE a na publikovanie správy PUBLISH – všetky môžu mať správu potvrdenia ACK (acknowledge).

Jeden z najpoužívanejších, voľne dostupných MQTT brokerov je Mosquitto, ktorý bol vyvinutý spoločnosťou Eclipse. Okrem toho vytvorili aj knižnicu Paho dostupnú pre väčšinu bežne používaných programovacích jazykov, ktorá slúži na programovanie klientskych aplikácií. Napriek tomu, že táto knižnica je použiteľná aj pre Arduino a iné mikrokontroléry, v práci sme sa rozhodli pre jeho jednoduchosť využiť namiesto nej knižnicu PubSubClient.

7. Charakteristika cvičenia

Pohyb závažových tehličiek ktorý je vykonávaný pri cvičení, teda ich dvíhanie a položenie, je spojitý jav. V digitálnom svete potrebujeme všetky spojité javy pretransformovať na diskrétne hodnoty, čím samozrejme prichádzame o istú časť informácií. Mieru straty informácie ovplyvňujú dve vlastnosti:

- Vzorkovacia frekvencia
- Bitová hĺbka

Vzorkovacia frekvencia (angl. sampling rate) definuje počet vytvorených vzoriek za jednotku času, obvykle za 1 sekundu. Platí pre ňu jednoduché pravidlo, vyššia vzorkovacia frekvencia (v rámci rozumných hodnôt) zabezpečí presnejšiu digitalizáciu analógového signálu.

Bitová hĺbka charakterizuje proces digitalizácie z opačného pohľadu. Udáva, počet hodnôt ktoré môže nadobudnúť digitálny signál. Keďže spojitý signál môže nadobudnúť nekonečne veľa hodnôt, z neho vytvorené diskrétne hodnoty (ktorých je ale konečne veľa), budú len ich aproximáciou – vzniknú odchýlky v dôsledku zaokrúhľovania. Zvýšením bitovej hĺbky sa zjemňuje delenie a zmenšuje sa odchýlka spôsobená zaokrúhľovaním. Samozrejme, nadmerne zvyšovaním len jednej z týchto vlastností sa nedosiahne požadovaný efekt získania presnejšieho signálu.

Na popísanie pohybu opakovania budeme v pravidelných časových intervaloch merať polohu dvíhaných závaží, teda výšku, v akej sa nachádzajú. Pod výškou budeme rozumieť počet políčok na vzorkovnici optického rotačného enkodéra, o ktoré sa pootočila kladka, pomocou ktorej sa zdvíhajú tehličky. Pri zjemňovaní tejto hodnoty sme

limitovaní nepresnosťou použitého IR senzora, ktorý nie je schopný spoľahlivo rozlišovať príliš úzke čierne políčka. Bezpečným limitom pre počet čierno-bielych políčok na vzorkovnici s priemerom 11 cm – čo je štandardná veľkosť kladky posilňovacej veže – je 12 až 14 políčok. Pri vzorkovacej frekvencií je limitom rýchlosť mikrokontroléra, ktorá je v danom prípade viac než postačujúca, zabezpečuje totiž viac než 2 000 vzoriek za sekundu.

7.1 Aktuálny stav

Údaje o cvičení sa majú dostať na server, prípadne sa zobrazovať napr. na smartfóne v reálnom čase. To znamená, že je potrebné v reálnom čase posielať charakterizujúce údaje o cvičení ktoré sme nazvali stav (angl. State) a budeme označovať S alebo správy typu S. Pre stav sme našli týchto 5 atribútov ktoré ho charakterizujú:

- *id* – jedinečná hodnota priradená mikrokontroléru.
- *timestamp* – časová pečiatka
- *difficulty* – náročnosť
- *progress* – progres
- *trend* – smer

Timestamp je časová pečiatka od začiatku série cvičenia vyjadrená v milisekundách. Mikrokontolér je schopný vďaka svojim interným hodinám počítať čas od posledného resetu a vrátiť túto hodnotu po zavolaní funkcie `millis()`.

Difficulty reprezentuje počet dvíhaných tehličiek. Z hľadiska mikrokontroléra je to hodnota od 0 do 1023 nameraná vstupným analógovým pinom, ku ktorému sú pripojené paralelné rezistory nachádzajúce sa na závažiach.

Progress zastupuje aktuálnu polohu závaží, teda výšku v akej sa nachádzajú. Táto hodnota je získaná počítaním zmenených políčok vzorkovnice pri otáčaní kladky. Keďže optický rotačný enkodér dokáže identifikovať smer rotácie, pri dvíhaní sa hodnota progress inkrementuje a naopak, pri spúšťaní dekrementuje.

Trend predstavuje aktuálny smer pohybu. Okrem toho, že vyjadruje či sa závažia dvíhajú alebo spúšťajú obsahuje aj informáciu o rýchlosti pohybu. Táto hodnota sa musí dynamicky meniť, teda musí byť neustále prepočítavaná. Trend budeme počítať pomocou posuvného okna dĺžky n (poľa typu `int/byte` dĺžky n) ktoré bude obsahovať posledných n hodnôt progresu.

7.1.1 Výpočet trendu ako rýchlosť

Už vyššie sme spomenuli, že na trend môžeme hľadieť, ako na rýchlosť pohybu, ktorá má kladné aj záporné hodnoty: kladné pri stúpaní závaží, záporné pri klesaní a samozrejme hodnoty blízke nule, ak závažia nie sú v pohybe.

Rýchlosť je z hľadiska fyziky miera pohybu alebo deja za daný časový interval. V našom prípade je časový interval daný rozdielom aktuálnej časovej počiatky a časovej pečiatky keď bola vložená do posuvného okna najstaršia hodnota (pokiaľ sa okno posúva doľava, je to hodnota na poslednej pozícií poľa).

Mieru pohybu (prejdenú dráhu) môžeme získať ako rozdiel aktuálnej hodnoty progresu a najstaršieho záznamu v posuvnom poli, čo sa opäť nachádza na poslednej pozícií poľa. Všimnime si, že v tomto prípade je posuvné okno úplne zbytočné, pristupujeme vždy len k jeho poslednej hodnote, čo môžeme oveľa efektívnejšie uchovávať v jednej primitívnej premennej daného typu.

Pri voľbe veľkosti posuvného okna, resp. pri použití jedinej premennej je dôležité stanoviť ako starý údaj budeme porovnávať s aktuálnym. Aktuálna rýchlosť je totiž citlivá na voľbu časového úseku. Spravidla sa používa čo najmenší interval, aby vypočítaná hodnota nereprezentovala priemernú rýchlosť. V tomto prípade však musíme brať do úvahy aj zlyhanie (presnejšie chybovosť) optických senzorov ktoré by pri zvolení prikrátkeho intervalu mohli skresliť reálne hodnoty.

7.1.2 Výpočet trendu ako smernica regresnej priamky

V štatistike je lineárna regresia lineárnym prístupom k modelovaniu vzťahu medzi nezávislou premennou a jednou alebo viacerými vysvetľujúcimi premennými. Prípad jednej vysvetľujúcej premennej sa nazýva jednoduchá lineárna regresia.

Lineárna regresia predstavuje aproximáciu daných hodnôt priamkou metódou najmenších štvorcov. Pokiaľ túto hľadanú priamku vyjadríme rovnicou $y = b_1 + b_2 * x$ tak riešením je nájdenie optimálnych hodnôt pre parametre (koeficienty) b_1 a b_2. Pre úplnosť je potrebné dodať, že vo všeobecnosti rieši lineárna regresia komplexnejší problém, pričom sa počíta aj s chybou pri aproximácií, no v našom prípade je postačujúce pracovať s touto jej najjednoduchšou verziou.

Metóda najmenších štvorcov je štatistická metóda na aproximáciu riešenia sústavy rovníc, ktorá obsahuje viac rovníc ako neznámych. Najmenšie štvorce znamenajú, že sa pri riešení minimalizuje súčet štvorcov odchýlky voči rovniciam. V praxi to teda znamená, že ak dané body aproximujeme priamkou, optimálna je tá, pri ktorej je súčet štvorcov všetkých bodov najmenší možný.

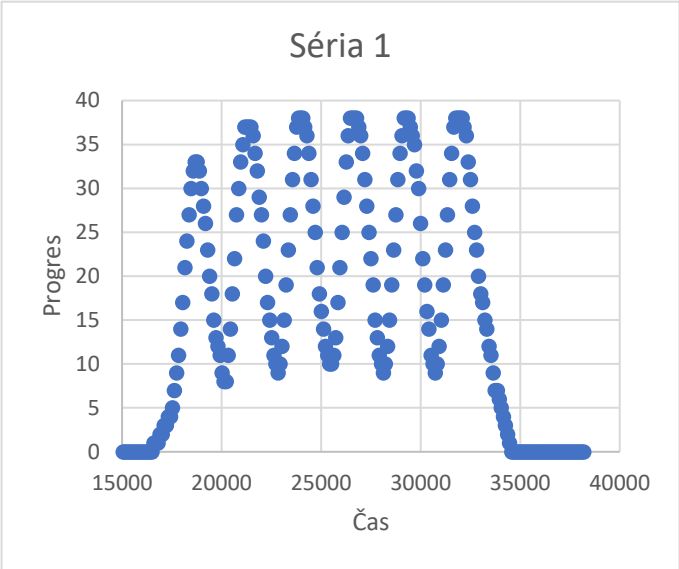
Pre výpočet trendu nepotrebujeme vyjadriť celú rovnicu regresnej (trendovej) priamky, stačí nájsť jej smernicu, teda koeficient b_2. Pri tomto postupe je už ale vhodné pracovať s posuvným oknom obsahujúcim posledných n hodnôt progresu. Kľúčovým bodom je zvoliť vhodnú veľkosť tohto okna. Keďže chceme vyjadriť aktuálnu rýchlosť, zvolením veľkého okna dostaneme podobne ako v predošlom prípade priemernú rýchlosť namiesto aktuálnej. Naopak, príliš malé okno (vzhľadom na možné nepresnosti hodnoty trendu) môže mať za následok skreslenie.

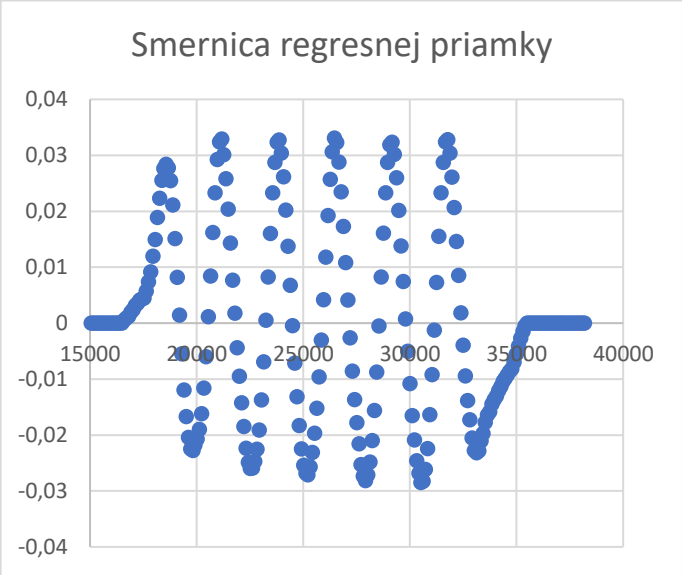
Na výpočet smernice priamky jednoduchej lineárnej regresie sa používa nasledujúci vzorec:

$$b_2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

kde n je veľkosť posuvného okna, teda počet hodnôt pre ktoré riešime regresiu, x_i je nezávislá premenná čas, y_i je náhodná premenná reprezentujúca progres a $\bar{x}$, $\bar{y}$ sú priemermi hodnôt x_i a y_i pre $\forall i \in \{1, 2, \ldots\ n\}$. Takto získaná hodnota smernice b_2 vyjadruje vektor pohybu nasledovným spôsobom:

- Kladné hodnoty znamenajú dvíhanie (čím väčšie, tým je pohyb rýchlejší)
- Záporné hodnoty znamenajú opačný pohyb, čiže spúšťanie (čím menšie, tým je pohyb rýchlejší)
- Hodnoty blízke nule znamenajú, že závažia sú v pokoji (teda nie sú v pohybe)





Obrázok 13 Príklad série a k nej patriace hodnoty regresnej priamky v čase

Samozrejme treba počítať s tým, že takto získaná informácia neodzrkadľuje úplne presne aktuálnu rýchlosť pohybu. Zmena pohybu sa objaví až po istom čase, teda po získaní niekoľkých ďalších hodnôt progresu. Aj keď sa snažíme získať aktuálnu rýchlosť, stále bude predstavovať len akýsi priemer, čo je pre nás síce menejcenná informácia ale rozhodne je zaujímavá.

7.2 Opakovanie

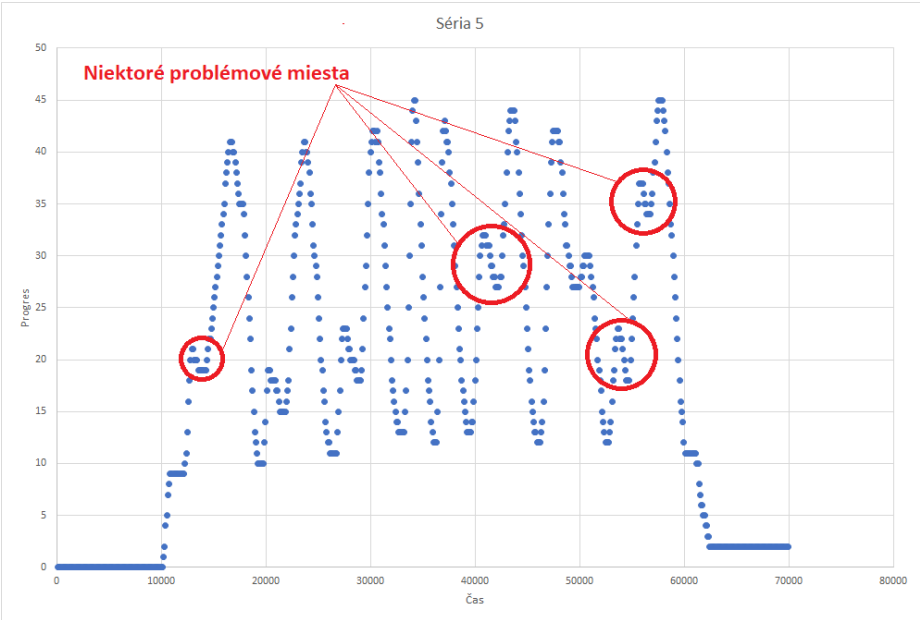
Jeden z cieľov tejto práce je automatické počítanie vykonaných pri cvičení s fitness strojom. Identifikovať jedno opakovanie pre človeka nie je problém. Cvičiaci závažie zdvihol následne ho položil. Môžeme pripočítať opakovanie. Pri tak jednoduchých pohyboch je identifikácia opakovania jednoduchá aj pre stroj, no v momente, keď pohyb nie je taký priamočiary, môže dôjsť k omylom.

Ako príklad si vezmime nasledujúcu ilustračnú situáciu: cvičiaci chce zdvihnúť závažia do výšky 40 cm. Začne vykonávať cvik, no ako sa dostane do výšky 30 cm, prestane vládať. Spraví si preto 2 sekundovú prestávku aby nabral silu počas ktorej mimovoľne spustí závažia o 10 cm. Následne ich trhnutím o 15 cm, kde si potrebuje opäť spraviť krátku prestávku pričom závažia klesnú o 5 cm. Posledným rýchlym pohybom ich dvihne o 10 cm do požadovanej výšky, čím splnil svoj cieľ a môže pokojne spustiť závažia do pôvodnej polohy.

Z pohľadu ľudského zmýšľania išlo jednoznačne o jedno opakovanie aj keď jeho dráha pohybu bola zložitejšia. Z pohľadu stroja, ktorý ale sleduje iba aktuálnu výšku dvíhaných závaží, to až také jednoznačné nie je: najprv dlhý pohyb smerom hore, potom trochu dole, prudko hore, chvíľu dole, opäť prudko hore a následne späť do východiskovej pozície. Nie je ťažké vidieť, že ak by bolo jedno opakovanie definované ako séria za sebou idúcich hodnôt progresu (pozície závažia) pričom tieto hodnoty sú najprv rastúce a potom klesajúce, identifikovali by sme tri opakovania namiesto jedného. Zjavne je potrebné definovať opakovanie inak.

Najjednoduchším a pomerne priamočiarym riešením je chvíľu počkať, čiže neprehlásiť stav za maximum v momente, keď zaznamenáme prvé zníženie hodnoty progresu, podobne v opačnom prípade pri prvom náraste za minimum. Teda ak progres rastie a mikrokontrolér zaregistruje pokles, zapamätá si predposlednú hodnotu ako potenciálne maximum, no za reálne maximum ho prehlási až keď zaznamená ďalších k klesajúcich hodnôt progresu. Analogicky pri opačnom pohybe.

Hodnotu prahovej konštanty k je potrebné vhodne zvoliť. Ak by bola príliš malá, znamenalo by to, že by sa drobné medzipohyby počas vykonávania cviku nevyhodnotili správne, teda by sa nakoniec identifikovalo viacej opakovaní ako sa v skutočnosti vykonalo. Naopak, ak by sa zvolilo za jej hodnotu príliš veľké číslo, opakovanie by sa vôbec nemuselo zaznamenať v prípade ak cvičiaci vykoná menšie pohyby ako nastavená prahová hodnota.



Obrázok 14 Príklad odcvičenej série s vyznačením niektorých, potenciálne zle interpretovaných miest

Podobne ako to bolo aj s rýchlosťou, aj tieto údaje budú mierne neaktuálne. Musí sa spočítať *k* hodnôt progresu z nasledujúceho opakovania, až vtedy sa môže prehlásiť predošlé opakovanie za ukončené. Je to malá nevýhoda, ktorú spôsobmi nie je možné elegantne vyriešiť, musíme sa s ňou jednoducho vysporiadať.

Samotné cvičenie nie je pre každého cvičiaceho len o počítaní opakovaní danou váhou. Niektorí športovci potrebujú získať aj nejaké informácie navyše o svojej aktivite. Zaujímavým údajom pre nich môže byť analýza jednotlivých opakovaní. Tie budeme označovať R (z angl. slova Repetition), a identifikovali sme pre ne nasledovné charakterizujúce atribúty:

- *id* – jedinečné identifikačné číslo zariadenia na ktorom sa zaznamenalo opakovanie
- *timestamp* – časová pečiatka vykonaná v momente ukončenia opakovania
- *repetition* – číslo opakovania v aktuálnej sérií opakovaní
- *difficulty* – obťažnosť, z pohľadu mikrokontroléra hodnota prečítaná na analógovom pine merajúcom počet aktívne zapojených rezistorov v rade
- *up_time* – trvanie dvíhania, časový rozdiel od začiatku cvičenia po začiatok spúšťania
- *up_distance* – výška do ktorej sa dvihli tehličky
- *idle_time* – čas nečinnosti
- *down_time* – vzdialenosť spúšťania
- *down_distance* – trvanie spúšťania

Väčšina týchto atribútov má jednoduchú interpretáciu, dajú sa vyčítať z časovej pečiatky a k nej prislúchajúceho progresu, resp. zo stavov prislúchajúcich danému opakovaniu. Napríklad, *up_distance* je rozdiel progresu v momente, keď sa závažové začali dvíhať a keď dosiahol svoje maximum.

Zaujímavým atribútom je *idle_time*, ktorý reprezentuje čas nečinnosti. Teda keď cvičiaci zdvihne závažia, ale predtým než by ich spustil, drží ich v jednom mieste. Tento údaj vieme získať na základe trendu stavu, ako už bolo vyššie popísané, ak hodnota trendu je blízka nule, znamená to, že závažia nie sú v pohybe. Prahovú hodnotu od ktorej sa pohyb považuje za nulový je potrebné taktiež vhodne zvoliť, aby pri extrémne pomalý pohyb, ktorý je mimochodom bežný pri posilňovaní, sa nevyhodnotil ako nečinnosť.

8. Záver

V článku sme popísali návrh riešenia hardvérovej časti pre systém ktorý je schopný zaznamenávať cvičenie v posilňovni. Zamerali sme sa na konkrétny druh strojov, ktorými sú posilňovacie veže so závažovými tehličkami, na ktorých sledujeme počet vykonaných opakovaní a váhu ktorá bola dvíhaná. Hardvér sme navrhli tak, aby bol odolný voči poškodeniu spôsobeného otrasmi alebo nevhodnou manipuláciou so strojom, pričom sme použili jednoduché senzory, elektrické súčiastky a mikrokontrolér.

V ďalšej časti sa budeme zaoberať vývojom mobilnej aplikácie a taktiež backendu, pomocou čoho bude možné automaticky ukladať údaje o vykonanom cvičení do databázy a sledovať históriu. Systém by mal do plnohodnotne nahradiť denníky, ktoré si cvičiaci vedú a navyše bude obohatený o rôzne štatistické ukazovatele pre lepší prehľad a orientáciu. Je navrhnutý dostatočne všeobecne, a je možné z neho vydolovať aj dodatočné dáta ohľadom dynamiky alebo správnosti vykonania cviku.

Literatúra

1. Binbin Yong et al., IoT-based intelligent fitness system, Journal of Parallel and Distributed Computing, Volume 118, Part 1, 2018, pp 14-21.
2. A. Jain, A Smart Gym Framework: Theoretical Approach, 2015 IEEE International Symposium on Nanoelectronic and Information Systems, Indore, 2015, pp. 191-196.
3. L. Mhatre and N. Rai, Integration between wireless sensor and cloud, 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, 2017, pp. 779-782.
4. S. Fitzgerald, M. Shiloh, T. Igoe, Arduino Projects Book, Arduino LLC, 2012
5. Radovici, A.; Culic, I.; Vaduva, A., „Lecture 1: Introduction to IoT," [Online]. Available: https://ocw.cs.pub.ro/courses/iot/courses/01.
6. Arduino, „Arduino Uno Rev3" [Online]. Available: https://store.arduino.cc/arduino-uno-rev3.