**ZÁPADOČESKÁ UNIVERZITA V PLZNI**
**FAKULTA ELEKTROTECHNICKÁ**

**KATEDRA ELEKTROMECHANIKY A VÝKONOVÉ ELEKTRONIKY**

# BAKALÁŘSKÁ PRÁCE

## Demonstrace inverzního kyvadla

**Jan Peroutka** **2017/2018**

ZÁPADOČESKÁ UNIVERZITA V PLZNI
**Fakulta elektrotechnická**
Akademický rok: **2017/2018**

# ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

| | |
|---|---|
| Jméno a příjmení: | **Jan PEROUTKA** |
| Osobní číslo: | **E14B0051P** |
| Studijní program: | **B2612 Elektrotechnika a informatika** |
| Studijní obor: | **Elektrotechnika a energetika** |
| Název tématu: | **Demonstrace inverzního kyvadla** |
| Zadávající katedra: | **Katedra elektromechaniky a výkonové elektroniky** |

Zásady pro vypracování:

Navrhněte a realizujte demonstrační přípravek pro funkci inverzního kyvadla.

1. Porovnejte možná řešení mechanické a pohonné části.
2. Porovnejte možné způsoby algoritmů řízení.
3. Navrhněte elektromechanickou část i řídící elektroniku.
4. Realizujte navržené řešení a ověřte funkčnost.

| | |
|---|---|
| Rozsah grafických prací: | **podle doporučení vedoucího** |
| Rozsah kvalifikační práce: | **30 - 40 stran** |
| Forma zpracování bakalářské práce: | **tištěná/elektronická** |

Seznam odborné literatury:

1. **Student si vhodnou literaturu vyhledá v dostupných pramenech podle doporučení vedoucího práce.**

| | |
|---|---|
| Vedoucí bakalářské práce: | **Ing. Petr Weissar, Ph.D.**<br>Katedra aplikované elektroniky a telekomunikací |
| Datum zadání bakalářské práce: | **10. října 2017** |
| Termín odevzdání bakalářské práce: | **7. června 2018** |

Doc. Ing. Jiří Hammerbauer, Ph.D.
děkan

L.S.

Prof. Ing. Václav Kůs, CSc.
vedoucí katedry

V Plzni dne 10. října 2017

## Abstrakt

Tato bakalářská práce se zabývá teorií a následnou konstrukcí inverzního kyvadla. V první části jsou představena možná konstrukční uspořádání a algoritmy řízení. Druhá část konkrétně popisuje navržené konstrukční řešení fyzického modelu inverzního kyvadla a jeho elektroniku. Zvoleno bylo řešení, kde je kyvadlo umístěno na vozíku poháněném stejnosměrným motorem. Třetí část se zabývá popisem software řídícího mikrokontroléru. V poslední, čtvrté kapitole je popsáno nastavení konstant regulace a funkce kyvadla.



## Klíčová slova

Inverzní kyvadlo, PID, regulátor, jazyk C, STM32, Nucleo, optický inkrementální enkodér

## Abstract

This bachelor thesis deals with the theory of inverted pendulum and describes its construction. First part introduces various construction types and control algorithms. Second part describes specifically the design of the constructed inverted pendulum device. In the selected design the pendulum is placed on the cart, which is driven by a DC motor. In the last, fourth part the regulation constants are set and the functionality of device is depicted.

## Key words

Inverted pendulum, PID, regulator, C programming language, STM32, Nucleo, optical incremental encoder

# Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této bakalářské práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské práce, je legální.

.....

podpis

V Plzni dne 7.6.2018 Jan Peroutka

# Poděkování

Děkuji panu Ing. Petru Weissarovi za pomoc při vedení bakalářské práce. Děkuji také slečně Evě Regnerové za pomoc s korekturou práce.

# Obsah

# Seznam symbolů a zkratek

SAŘ..... Systém automatického řízení

PID ..... Proporcionálně-integračně-derivační (regulátor)

$e(t)$ ..... regulační odchylka

$u(t)$ ..... akční veličina

$K$..... Proporcionální zesílení

$T_I$ ..... integrační časová konstanta

$T_D$ ..... derivační časová konstanta

$K_I$..... integrační konstanta

$K_D$ ..... derivační konstanta

$F_R(p)$..... Laplaceův přenos regulátoru

$\theta$ ..... úhel [rad]

$F, N, P, u$ ..... síla [N]

$M$ ..... moment síly [N.m]

$m$ ..... hmotnost [kg]

$x$ ..... vodorovná vzdálenost [m]

$\dot{a}$ ($\ddot{a}$) ..... první (druhá) derivace veličiny $a$ podle času

$I, J$ ..... moment setrvačnosti [kg.m$^2$]

$\boldsymbol{r}$ ..... polohový vektor

$\hat{\boldsymbol{\imath}}$ , $\hat{\boldsymbol{\jmath}}$ ..... bázové vektory ve směru os x a y

$l$ ..... polovina délky kyvadla [m]

LQR ..... lineárně kvadratický regulátor

$\rho$ ..... hustota

$V$ ..... objem

Φ ..... magnetický indukční tok

$I$ ..... elektrický proud

PWM ..... pulse width modulation (pulzně-šířková modulace)

λ..... krok inkrementálního enkodéru [dílky]

$l$ ..... polovina délky kyvadla [m]

USB ..... universal serial bus

USART ..... synchronní / asynchronní sériové rozhraní

PLL ..... phase locked loop – fázový závěs

Interrupt ..... přerušení

GPIO ..... general purpose input-output (vstup/výstup mikrokontroléru)

# Úvod

Cílem práce je shrnout informace o inverzním kyvadlu a zkonstruovat jeho funkční fyzický model. V první kapitole nejprve popíšu inverzní kyvadlo z obecnějšího hlediska. Poté navrhnu řízení realizované dvojitou regulační smyčkou s PID a PD regulátory, které pro své inverzní kyvadlo použiji. Dále také provedu odvození soustavy diferenciálních rovnic kyvadla a nastíním pokročilejší možnosti řízení.

Ve druhé kapitole seznámím čtenáře s navrženou a zhotovenou konstrukcí inverzního kyvadla. K pohonu jsem vybral stejnosměrný motor s permanentními magnety o dostatečném výkonu a vozík s kyvadlem jsem se snažil navrhnout co nejlehčí. Od toho si slibuji dobré dynamické vlastnosti systému.

Třetí kapitola představí implementaci regulačního algoritmu v mikrokontroléru. Pro tyto účely byl vybrán 32-bitový mikrokontrolér STM32F411, který lze provozovat až na frekvenci 100 MHz. Díky tomu by měl být mikrokontrolér schopen v dostatečně krátkém čase počítat krok algoritmu regulace a též posílat informace do PC.

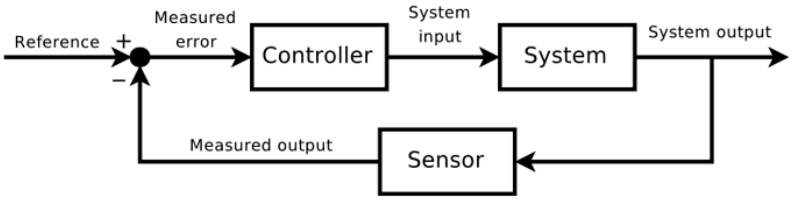
Nakonec popíšu nastavení konstant pro regulátory a demonstruji praktickou funkčnost zhotoveného inverzního kyvadla.

# 1 Inverzní kyvadlo

Inverzní kyvadlo je elektromechanický systém, který slouží jako ukázka stabilizace z fyzikální podstaty nestabilního systému pomocí regulace se zpětnou vazbou.

Zařízení je tvořeno kyvadlem obvykle v podobě tyče, která se může volně otáčet kolem osy procházející kolmo jejím koncem. Kyvadlo je umístěno na pohyblivé základně, jejíž pohyb je realizován elektrickým či jiným aktuátorem. V případě vypnutého řízení kyvadlo pochopitelně visí dolů. Účelem řízení kyvadla je pohybovat základnou s kyvadlem tak, aby se udrželo v nestabilní, tj. vzpřímené poloze.

Pro řízení inverzního kyvadla je nutno snímat regulační odchylku, kterou je úhel natočení kyvadla od svislého směru. Ta je poté zpracovávána regulátorem, který vytváří potřebný signál pro aktuátor.

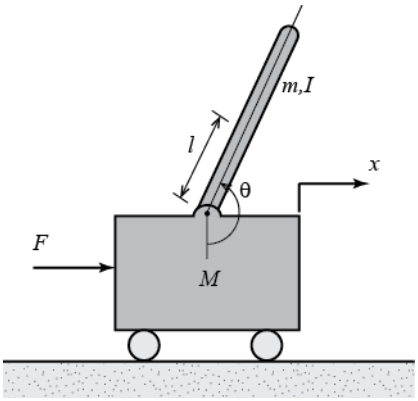



Obr. 1.1 *Jednoduchá zpětnovazební regulace*

Samotné inverzní kyvadlo nemá žádný jiný, než demonstrativní význam, avšak podobný princip je využit u mnohých zařízení, například vozítka Segway.

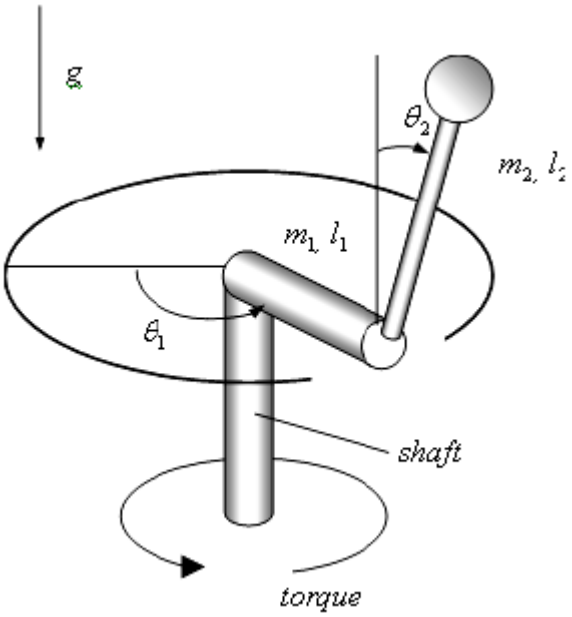
## 1.1 Možná konstrukční řešení inverzního kyvadla

Inverzní kyvadlo lze realizovat různými způsoby. První možností je umístění kyvadla na vozík, který koná přímočarý pohyb (obr. 1.2). Řídící veličinou je pak obvykle síla působící na vozík. Tu může vytvářet například elektrický motor s převody nebo pneumatický aktuátor.




Obr. 1.2 *Inverzní kyvadlo na vozíku* [1]

Další konstrukční možností je rotační inverzní kyvadlo.V tomto uspořádání je kyvadlo umístěno na konci tyče, jež je pevně kolmo připevněná na hřídel. Hřídel je poháněna přes převody elektrickým motorem. Řídící veličinou systému je v tomto uspořádání moment hřídele.




Obr. 1.3 *Rotační inverzní kyvadlo* [2]

Inverzní kyvadlo se dá realizovat i poměrně nekonvenčními způsoby, jako například létající inverzní kyvadlo nesené kvadrokoptérou [3]. Je také možné stabilizovat dvojité a dokonce i trojité kyvadlo (tj. kyvadlo skládající se ze dvou, respektive ze tří článků spojenými klouby) [4].

## 1.2 Možné algoritmy řízení inverzního kyvadla

Základem automatického řízení je regulátor. „Regulátor je část systému automatického řízení (SAŘ), která zajišťuje automatické řízení soustavy ve smyslu optimálních požadavků na řízenou soustavu kladených. Regulátor jako řídicí systém uvažujeme s jedním vstupem a

jedním výstupem, kde vstupní veličina je regulační odchylka $e(t)$ a výstupní veličina je akční veličina $u(t)$ v obvodu SAŘ.“ [5]

V této části práce se budu zabývat možnými algoritmy řízení inverzního kyvadla a odvodím rovnice kyvadla. Podrobněji však popíši pouze způsob řízení použitý v mém modelu, neboť implementace složitějšího regulátoru vyžaduje složitější matematické metody, které jsou nad rámec této práce.

### 1.2.1 PID regulace

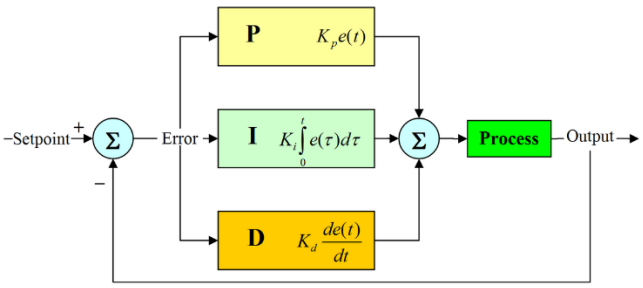
PID regulátor je nepřímý spojitý lineární regulátor se třemi měnitelnými parametry, který se často využívá v průmyslových aplikacích. Nepřímý regulátor se skládá z těchto částí:

a) Měřicí člen – zjišťuje skutečnou hodnotu regulované veličiny a vytváří regulační odchylku

b) Ústřední člen – zpracovává regulační odchylku. V užším smyslu se označuje také jako regulátor a často pod pojmem regulátor myslíme právě ústřední člen.

c) Akční člen (aktuátor) – odpovídajícím způsobem působí na řízenou soustavu. [5]

PID regulátor se skládá ze tří paralelně zapojených regulátorů – proporcionálního (P), integračního (I) a derivačního (D). Lze využít i jiné kombinace, například PI, PD, nebo samostatný P regulátor.




Obr. 1.4 *PID regulátor* [6]

Pro akční veličinu na výstupu PID regulátoru platí

$$u(t) = K\left(e(t) + \frac{1}{T_I}\int_0^{\infty} e(\tau)d\tau + T_D\frac{de(t)}{dt}\right),$$

kde $K$ nazýváme proporcionálním zesílením, $T_I$ integrační časovou konstantou a $T_D$ je derivační

časová konstanta. Místo $T_I$ můžeme též psát $K_I = \frac{K}{T_I}$ a místo $T_D$ lze použít $K_D = KT_D$.

Aplikací Laplaceovy transformace na rovnici PID regulátoru získáme přenos regulátoru

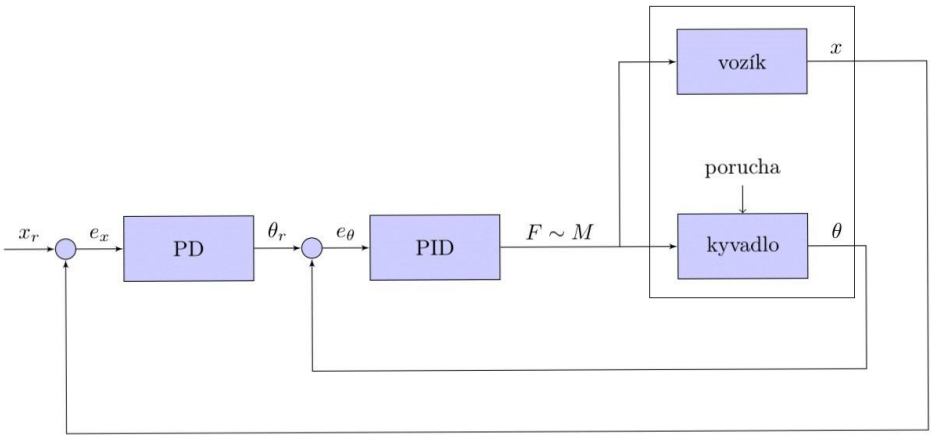
$$F_R(p) = \frac{K_D p^2 + Kp + K_I}{p}$$

Jednotlivé složky mají specifický vliv na vlastnosti regulace:

- P-složka: Zvyšuje přesnost regulace a rychlost odezvy, při příliš vysoké hodnotě $K$ rozkmitává výstup a vede až ke ztrátě stability.
- I-složka: Nezbytná pro dosažení přesné hodnoty výstupní veličiny, zvyšováním $K_I$ roste kmitavost.
- D-složka: zesiluje šum, vylepšuje dynamiku.

### 1.2.2 Navržené řešení řízení inverzního kyvadla

Pro navrhovaný model kyvadla jsem zvolil řízení využívající dvou regulačních smyček (obr. 1.5).




Obr. 1.5 *Regulační schéma inverzního kyvadla*

Vnitřní regulační smyčka reguluje úhel kyvadla $\theta$ a jako akční veličinu využívá sílu působící na vozík $F$, respektive moment motoru $M$, který je síle přímo úměrný. Tato smyčka využívá PID regulátor. K udržování kyvadla v horní poloze postačuje samotná tato smyčka.

Jelikož ale není nijak regulována poloha vozíku, kyvadlo by brzy "ujelo" na konec dráhy, kde by již nemělo prostor ke stabilizaci.

Proto je nutné zavést druhou regulační smyčku pro polohu vozíku. Tato smyčka obsahuje regulátor PD. Integrační složka je vynechána z čistě praktických důvodů popsaných v kapitole 3.7.1.2.

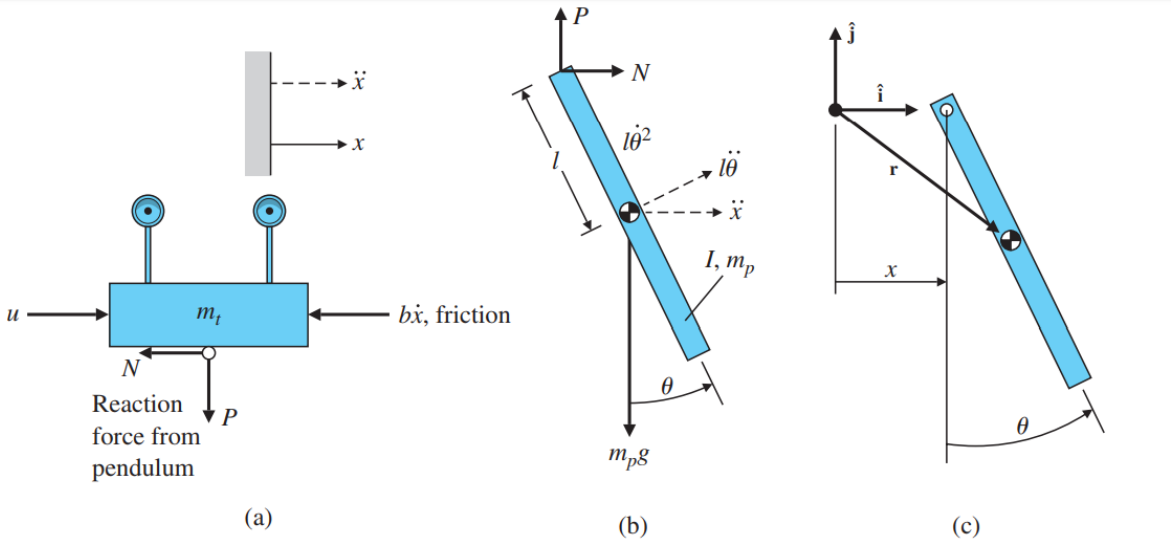
Konstanty obou regulátorů budou nastaveny experimentálně. Postup nastavení bude popsán v kapitole 4.

### 1.2.3 Možnosti řízení využívající matematický model kyvadla

Navržené řízení kyvadla nezahrnuje model systému a konstanty regulátorů jsou nastavovány experimentálně. Složitější a robustnější řízení však potřebuje pracovat s matematickým modelem kyvadla. Nejprve zde proto odvodím rovnice matematického modelu inverzního kyvadla. Dále pak jen stručně nastíním pokročilejší možnosti řízení inverzního kyvadla využívající matematický model.

#### 1.2.3.1 Odvození rovnic inverzního kyvadla

Rovnice inverzního kyvadla odvodím tak, jak je to provedeno v knize Feedback control of dynamic systems [7]. Ke komentářům ze zdroje přidám též své poznámky.




Obr. 1.4 *Ilustrace k odvození rovnic kyvadla* [7].

K odvození využijeme rovnici druhého Newtonova pohybového zákona

$$F = m\ddot{x} \tag{1.1}$$

A rovnici jeho analogie pro rotační pohyb

$$M = I\ddot{\theta} \tag{1.2}$$

Hmotnost kyvadla značme $m_p$ a hmotnost vozíku $m_t$. Dále označme síly reakce kyvadla na vozík ve vodorovném a svislém směru $N$ a $P$ (obr. 1.4 b)).

Mějme souřadnicovou soustavu tvořenou bázovými vektory $\hat{\boldsymbol{\imath}}$ a $\hat{\boldsymbol{\jmath}}$ ve směru os $x$ a $y$. Vektorem $\boldsymbol{r}$ označme polohu hmotného středu kyvadla, které se může otáčet kolem vyznačené osy otáčení (obr. 1.4 c)). Osa otáčení je umístěna na vozíku, jehož pohyb je omezen do směru osy x (obr. 1.4 a)). Pro vektor $\boldsymbol{r}$ pak platí vztah

$$\boldsymbol{r} = x\hat{\boldsymbol{\imath}} + l(\hat{\boldsymbol{\imath}}\, sin\theta - \hat{\boldsymbol{\jmath}}\, cos\theta),$$

kde $l$ je polovina délky kyvadla a $\theta$ úhel od záporné poloosy $y$. Pro první derivaci $\boldsymbol{r}$ podle času máme

$$\dot{\boldsymbol{r}} = \dot{x}\hat{\boldsymbol{\imath}} + l\dot{\theta}(\hat{\boldsymbol{\imath}}\, cos\theta + \hat{\boldsymbol{\jmath}}\, sin\theta)$$

a pro druhou derivaci (podle vztahu pro derivaci součinu a složené funkce)

$$\ddot{\boldsymbol{r}} = \ddot{x}\hat{\boldsymbol{\imath}} + l\ddot{\theta}(\hat{\boldsymbol{\imath}}\, cos\theta + \hat{\boldsymbol{\jmath}}\, sin\theta) - l\dot{\theta}^2(\hat{\boldsymbol{\imath}}\, sin\theta - \hat{\boldsymbol{\jmath}}\, cos\theta)) \tag{1.3}$$

V rovnici (1.3) můžeme vidět, že člen $l\dot{\theta}^2$ leží ve směru kyvadla a míří do osy otáčení a člen $l\ddot{\theta}$ je kolmý na kyvadlo (obr. 1.4 b)).

Nyní pišme rovnici pro vozík s kyvadlem. Podle (1.1) můžeme psát

$$m_t\ddot{x} + b\dot{x} = u - N, \tag{1.4}$$

kde $m_t$ je hmotnost vozíku a $b$ je koeficient tření přímo úměrného rychlosti (později bude zanedbáno). Na pravé straně máme vnější sílu působící na vozík $u$ a sílu od reakce kyvadla ve vodorovném směru $N$.

Uvědomíme si, že použitím (1.1) na polohový vektor kyvadla dle obr 1.4 b) ve vodorovném a svislém směru a (1.2) na rotační pohyb kyvadla bychom získali tři rovnice obsahující neznámé síly $N$, $P$ a neznámý úhel $\theta$. Z těchto tří rovnic bychom mohli vyloučit neznámé síly a získali bychom jednu rovnici popisující pohyb kyvadla – jedinou rovnici pro neznámý úhel $\theta$. Například dosazením vodorovné složky druhé derivace polohového vektoru (1.3), vodorovné reakční síly kyvadla $N$ a hmotnosti kyvadla $m_p$ do rovnice (1.1) dostáváme

$$N = m_p\ddot{x} + m_p l\ddot{\theta}cos\theta - m_p l\dot{\theta}^2 sin\theta \tag{1.5}$$

Jako jednodušší postup, který ušetří mnoho algebraických úprav se však ukazuje aplikovat rovnici (1.1) kolmo na kyvadlo. Tím dostaneme

$$P\ sin\theta + N\ cos\theta - m_p g sin\theta = m_p l\ddot{\theta} + m_p\ddot{x}cos\theta, \tag{1.6}$$

kde na levé straně máme složky sil $P$, $N$ a tíhové síly působící ve směru kolmém na kyvadlo a na pravé straně složky zrychlení v témže směru násobené hmotností kyvadla.

Aplikací (1.2) na rotační pohyb kyvadla pro momenty působící ve středu kyvadla získáme

$$-Pl\ sin\theta - Nl\ cos\theta = I, \tag{1.7}$$

kde $I$ je moment setrvačnosti kyvadla vzhledem k ose procházející jeho hmotným středem.

Vynásobením rovnice (1.6) polovinou délky kyvadla $l$ a jejím sečtením s (1.7) dostáváme pohybovou rovnici pro úhel kyvadla na vozíku:

$$\left(I + m_p l^2\right)\ddot{\theta} + m_p gl\ sin\theta = -m_p l\ddot{x}\ cos\theta. \tag{1.8}$$

Tato rovnice je totožná s rovnicí kyvadla bez vozíku, pouze obsahuje navíc pravou stranu, která má fyzikální význam momentu vytvářeného pohybem vozíku.

Pro získání pohybové rovnice vozíku dosadíme rovnost (1.5) do (1.4) , čímž vyloučíme z (1.4) neznámou sílu a dostaneme

$$(m_t + m_p)\ddot{x} + b\dot{x} + m_p l\ddot{\theta}\, cos\theta - m_p l\dot{\theta}^2\, sin\theta = u \tag{1.9}$$

Tím jsme získali soustavu nelineárních diferenciálních rovnic (1.8) a (1.9) pro kyvadlo a pro vozík. Rovnice nyní ještě přepíšeme pro úhel $\theta' = \theta + \frac{\pi}{2}$, který odpovídá úhlu vychýlení inverzního kyvadla od horní svislé polohy a aplikujeme pravidla pro goniometrické funkce a derivace. Také zanedbáme tření $b$ v rovnici (1.9).

$$\left(I + m_p l^2\right)\ddot{\theta}' - m_p g l\, sin\theta' = m_p l\ddot{x}\, cos\theta'.$$

$$(m_t + m_p)\ddot{x} - m_p l\ddot{\theta}' cos\theta' - m_p l\dot{\theta}'^2\, sin\theta' = u \tag{1.10}$$

Nakonec provedeme linearizaci rovnic v okolí $\theta' = 0$, kde aproximujeme $cos\theta' \cong 1$, $sin\theta' \cong \theta'$ a $\dot{\theta}'^2 \cong 0$. Zároveň můžeme podle Steinerovy věty nahradit $\left(I + m_p l^2\right) = J$, kde $J$ je moment setrvačnosti kyvadla okolo jeho osy otáčení.

$$J\ddot{\theta}' - m_p g l\, \theta' = m_p l\ddot{x}$$

$$(m_t + m_p)\ddot{x} - m_p l\ddot{\theta}' = u \tag{1.11}$$

Tím jsme dostali soustavu lineárních diferenciálních rovnic (1.11) popisujících inverzní kyvadlo na vozíku.

#### 1.2.3.2 Lineárně kvadratický regulátor

Jedním ze způsobů řízení využívajících matematický model systému je lineárně kvadratický regulátor (LQR). LQR je kvadraticky optimální regulátor systému popsaného lineárními diferenciálními rovnicemi. To znamená, že se snaží minimalizovat účelovou funkci (cost function), která je v tomto případě dána tzv. kvadratickým kritériem. Účelová funkce je funkcí stavů systému a řídících vstupů.

„Úkolem LQ řízení je získat stabilní optimální systém s přiměřeně dobrou odezvou. U takového systému se předpokládá a požaduje, aby díky regulátoru byl systém v rovnováze nebo nastaven do daného bodu navzdory rušení. Proto je cílem minimalizovat působení rušení na systém.“ [8]

#### 1.2.3.3 Regulátor využívající Fuzzy logiku

Další možností je použít fuzzy regulátor. Fuzzy logika, na rozdíl od klasické logiky, připouští kromě úplného nebo žádného členství v množině také členství částečné.

Fuzzy systém funguje tak, že jsou nejprve vstupní veličiny převedeny na fuzzy hodnoty (fuzzyfikace). Na tyto hodnoty je aplikována sada pravidel (algoritmus regulace) podle fuzzy logiky. Nakonec je výsledek opět převeden na reálnou hodnotu (defuzzyfikace).

# 2 Návrh elektromechanické části

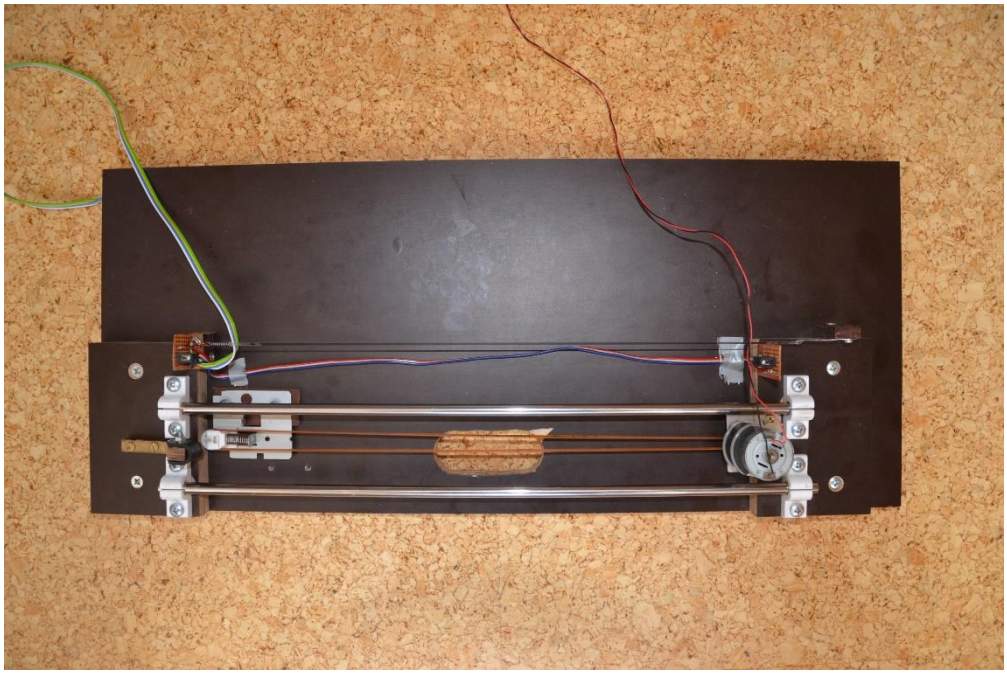
Pro model kyvadla byl zvolen lineární pojezd vozíku s pohonem se stejnosměrným motorem. Úhel natočení kyvadla a poloha vozíku jsou zjišťovány inkrementálními optickými senzory polohy. Převod z motoru na vozík s kyvadlem je realizován ozubeným řemenem.

## 2.1 Návrh mechanické konstrukce

### 2.1.1 Lineární pojezd

Základem modelu inverzního kyvadla je obdélníková deska z překližky o tloušťce 18 mm a rozměrech 57 x 24 cm, ke které jsou přišroubovány dvě menší obdélníkové desky z téže překližky o rozměrech 8 x 12 cm (obr 2.1). Tyto slouží jako podložky pro upevnění vodících tyčí a motoru. Vodící tyče jsou z povrchově kalené nerezové kulatiny o průměru 8 mm. K překližce jsou připevněny pomocí dílů vytištěných na 3D tiskárně.

Na pravé straně zařízení je pomocí příruby z hliníkového plechu o tloušťce 3 mm přišroubován motor. Na levé straně se nachází kladka, která přes tlačnou pružinu napíná řemen. Na obou stranách pojezdu jsou mechanické dorazy, které chrání model před destrukcí v případě najetí vozíku do konce rozsahu v plné rychlosti. Použitý řemen je ozubený a má šířku 4 mm a obvod cca 75 cm. Obvodem řemene je omezená délka pojezdu na zhruba 26 cm od dorazu k dorazu.

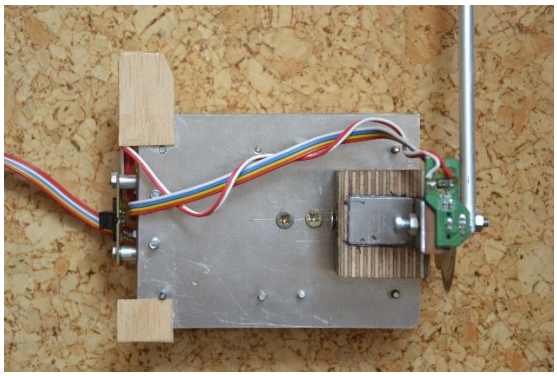


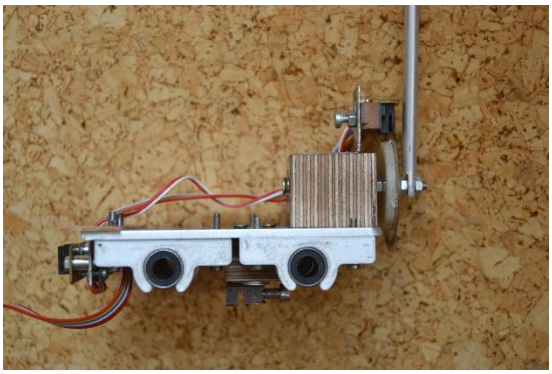
Obr. 2.1 *Lineární pojezd kyvadla shora*

Zezadu je k obdélníkovým podložkám z překližky upevněn pásek optického senzoru polohy. Na koncích dráhy jsou připevněny optické závory, jež zatím nebyly v řízení kyvadla z časových důvodů využity.

Základ vozíku, na kterém je umístěno kyvadlo, tvoří obdélníková hliníková deska o tloušťce 3 mm a rozměrech 85 x 110 mm. K ní jsou zespodu přišroubovány čtyři ložiskové domky vytištěné na 3D tiskárně, ve kterých jsou uložena lineární ložiska. Dále je ze spodní strany desky našroubováno upínání řemenu k vozíku.

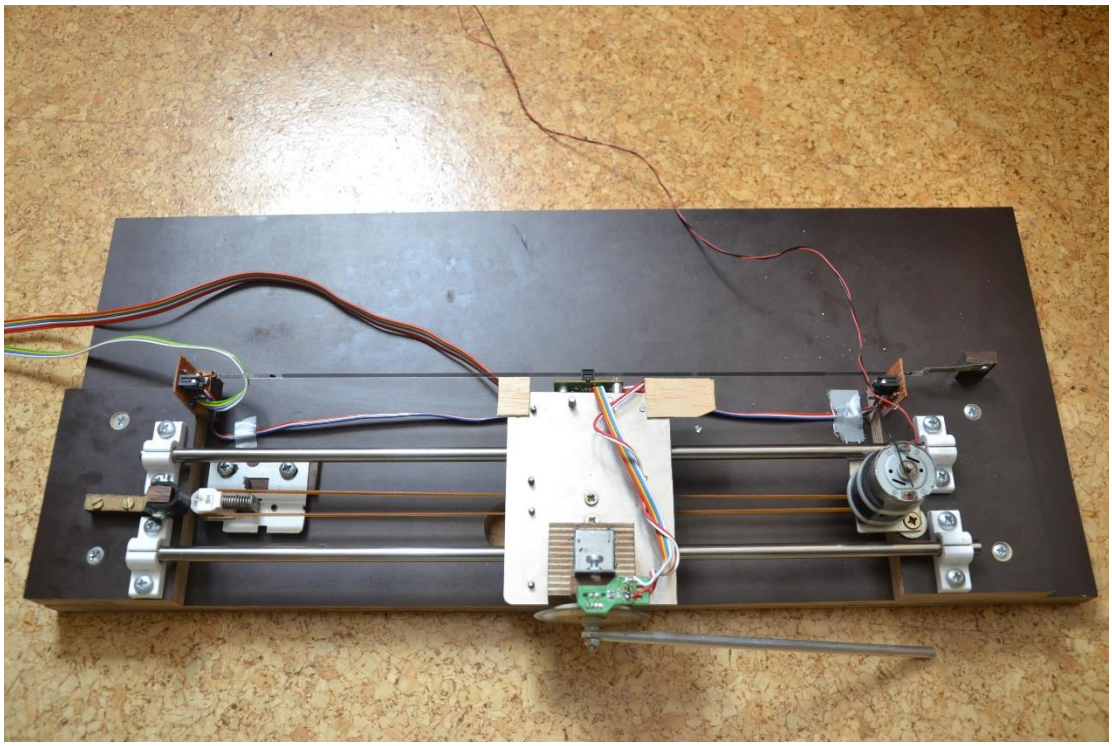
Shora je na vozíku přišroubován kvádr z překližky o rozměrech 40 x 29 x 36 mm, ve kterém je vyvrtán otvor o průměru 13 mm. V tom jsou zasazena ložiska o vnitřním průměru 4 mm. V ložiskách je pomocí matek připevněný šroub M4 x 60 mm, na němž je ze přední strany vozíku umístěna podložka vysoustružená z polyamidu, ke které je přilepen disk senzoru natočení kyvadla. Na konci šroubu je pomocí matek uchyceno samotné kyvadlo, které je tvořeno hliníkovou kulatinou o průměru 6 mm a délce 200 mm. Shora je na kvádru připevněn optický senzor, který z disku získává informaci o natočení kyvadla.





Obr. 2.2 a) *pohled na vozík shora* b) *pohled na vozík z boku*

K zadní části vozíku je připevněn druhý optický senzor, který z pásku čte informaci o poloze vozíku s kyvadlem. Vodiče vedoucí z optických senzorů jsou sloučeny v jeden šestivodičový plochý kabel.



Obr. 2.3 *Pohled na celou mechanickou část shora*

### 2.1.2 Parametry zkonstruovaného zařízení

Pro případné implementování regulátoru založeného na matematickém modelu kyvadla je důležité znát hmotnost kyvadla, moment setrvačnosti kyvadla a hmotnost vozíku. Pro dokonalejší model by bylo třeba zohlednit i například tření vozíku a kyvadla, moment setrvačnosti motoru atd. Tyto veličiny je ale mnohem složitější změřit a bylo by nad rámec této práce to provést.

Hmotnost kyvadla spočteme podle vztahu

$$m = \rho * V,$$

kde V je objem válce o poloměru 3 mm a délce 200 mm a $\rho = 2700\ kg.m^{-3}$ je hustota hliníku. Po převedení jednotek na základní a dosazení obdržíme hmotnost kyvadla $m = 15{,}3\ g$. Spočítaná hmotnost se shoduje s orientačním vážením kyvadla.

Moment setrvačnosti kyvadla J při zanedbání momentu setrvačnosti závitové tyče a senzoru natočení můžeme spočítat podle vztahu pro výpočet momentu setrvačnosti tyče, kde osa otáčení prochází koncem tyče.

$$J = \frac{1}{3} m * l^2 = 2{,}04 \,.\, 10^{-4}\ kg.m^{\,2}$$

Vážením byla zjištěna hmotnost vozíku $m_t$ = 295 g. Většinu hmotnosti vozíku tvoří lineární ložiska, proto by bylo obtížné dosáhnout při stejné konstrukci výrazně menší hmotnosti.

Tabulka 1: *změřené a spočítané parametry modelu*

| | | |
|---|---|---|
| kyvadlo | hmotnost | $m_p$ = 15,3 g |
| | moment setrvačnosti | J = 2,04 . $10^{-4}$ kg.$m^2$ |
| vozík | hmotnost | $m_t$ = 295 g |

## 2.2 Popis pohonu

K pohonu vozíku, na kterém je umístěno kyvadlo, byl zvolen stejnosměrný komutátorový motor s permanentními magnety. Tento pohon má oproti krokovému motoru výhodu ve větší účinnosti, a tudíž ve větším možném výkonu při stejném napájení. Také může na rozdíl od krokového motoru dosahovat podstatně vyšších otáček a nehrozí ztráta kroku. [9]

Nevýhodou je naopak nutnost použití zpětné vazby polohy. Ta je realizována již zmiňovaným optickým inkrementálním senzorem snímajícím polohu vozíku s kyvadlem. Senzor se skládá z pásku připevněného k základně modelu a z čidla umístěného na vozíku.

Motor je řízen přes H-můstek, který umožňuje pulzně šířkovou modulaci napětí v obou polaritách.

### 2.2.1 Zvolený motor

K realizaci kyvadla byl zvolen stejnosměrný komutátorový motor RS-455PA od Mabuchi motor. Udávaný rozsah napájecího napětí motoru je 12 – 42V. Při napájecím napětí 42V a zastaveném motoru udává výrobce proud 1,72A a moment 0,11 Nm. Datasheet motoru je dostupný na webu [10].

*Obr. 2.4: Motor Mabuchi RS-455PA*

Na hřídeli motoru je nasazena řemenice o průměru 6 mm. Ze vztahu pro velikost momentu síly při kolmosti síly a ramena

$$M = F.r$$

si vyjádříme sílu F. Za r dosadíme poloměr řemenice a za M výše udávaný moment a vychází nám, že motor při napájení 42 V může na vozík působit silou až zhruba 36 N. Při napájení 30 V je síla o něco menší, avšak pro účely modelu kyvadla zcela dostačující.

Výrobce udává výkon při maximální účinnosti 8,22 W, kdy motor odebírá proud 0,31 A. Můžeme očekávat, že při větším zatížení bude motor schopen poskytnout výkon až několikrát vyšší.

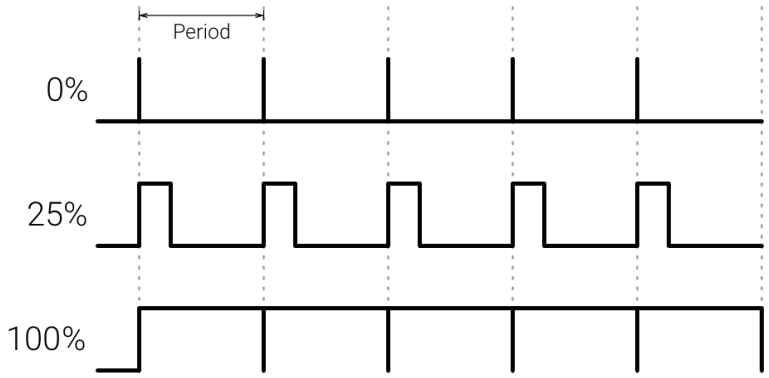
### 2.2.2 Použité řízení motoru

Pro moment stejnosměrného motoru přibližně platí vztah

$$M = k.\Phi.I$$

kde k je konstanta, Φ je magnetický indukční tok a I je proud rotorem motoru. Vzhledem k použití permanentních magnetů je tok Φ konstantní, čili moment motoru je přímo úměrný procházejícímu proudu.

K řízení motoru je využíta pulzně-šířková modulace (PWM). PWM je modulace, kdy se při konstantní periodě obdélníkového signálu mění poměr doby trvání vysoké úrovně signálu ku periodě. Tento poměr se nazývá střída, obvykle se udává v procentech a je přímo úměrný střední hodnotě výstupního signálu (obr. 2.5).



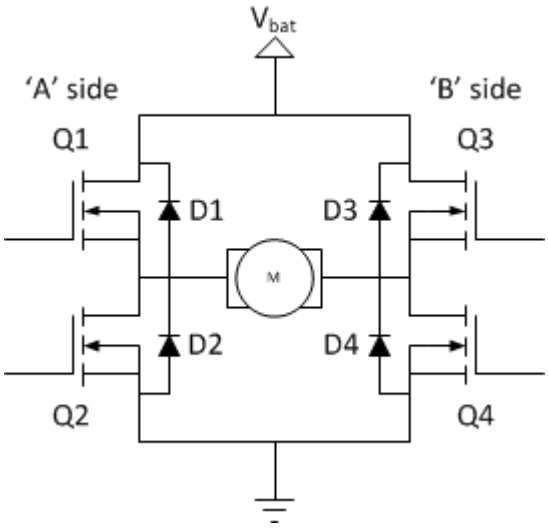

Obr. 2.5: *Pulzně šířková modulace*

Řízení motoru probíhá v otevřené smyčce a je založené na zjednodušujícím předpokladu, že střední hodnota proudu je přímo úměrná střední hodnotě napětí modulovaného PWM. To je velké zjednodušení, protože přesnější model stejnosměrného motoru obsahuje kromě odporu také indukčnost a indukované napětí závislé na otáčkách.

Nicméně bylo experimentálně ověřeno, že je toto zjednodušení pro potřeby modelu kyvadla zcela dostatečné. V případě požadavku na přesnější řízení by bylo vhodné zavést zpětnou vazbu proudu a ten regulovat pomocí PID regulátoru.

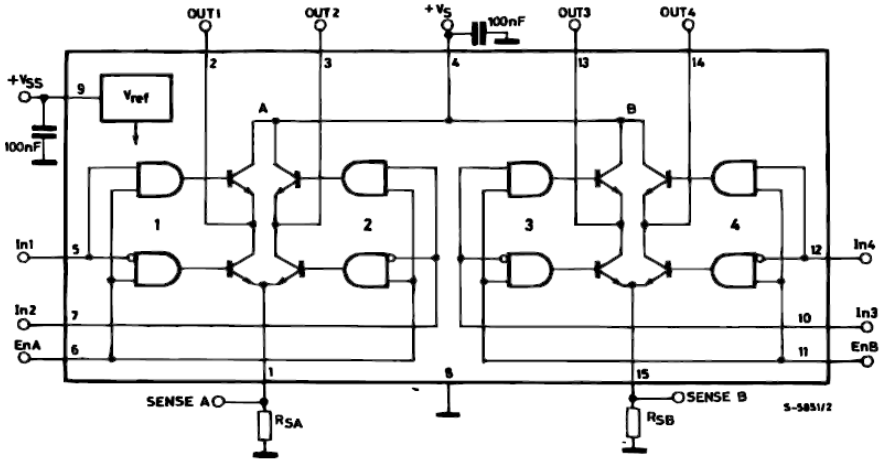
### 2.2.3 Modul s H-můstkem L298N

H-můstek je elektronický obvod tvořený čtyřmi tranzistory, který umožňuje řízení stejnosměrného motoru v obou směrech otáčení. Název je odvozen od podobnosti schématu s písmenem H. Tranzistory pracují ve spínacím režimu a vždy jsou sepnuty pouze 2 diagonálně umístěné (obr 2.4), například Q1 a Q4. H-můstek lze realizovat jak pomocí bipolárních, tak unipolárních tranzistorů. Nezbytným doplněním tranzistorů jsou antiparalelně zapojené diody, které vedou proud v okamžiku po vypnutí tranzistoru a zabraňují tak vzniku přepětí, které by mohlo poškodit tranzistor.




Obr. 2.6: *Schéma H-můstku* [11]

Pro řízení motoru kyvadla je použit modul osazený integrovaným H-můstkem L298N. Tento obvod lze použít pro napájecí napětí do 46V a trvalý stejnosměrný proud do 2A, což je pro zvolený motor dostačující. L298N obsahuje dva nezávislé H-můstky, díky čemuž může být použit k řízení jednoho krokového nebo dvou stejnosměrných motorů. V našem případě je využit pouze jeden z H-můstků. Datasheet obvodu L298N je dostupný na webu [12].




Obr. 2.7: *Blokové schéma integrovaného obvodu L298N*

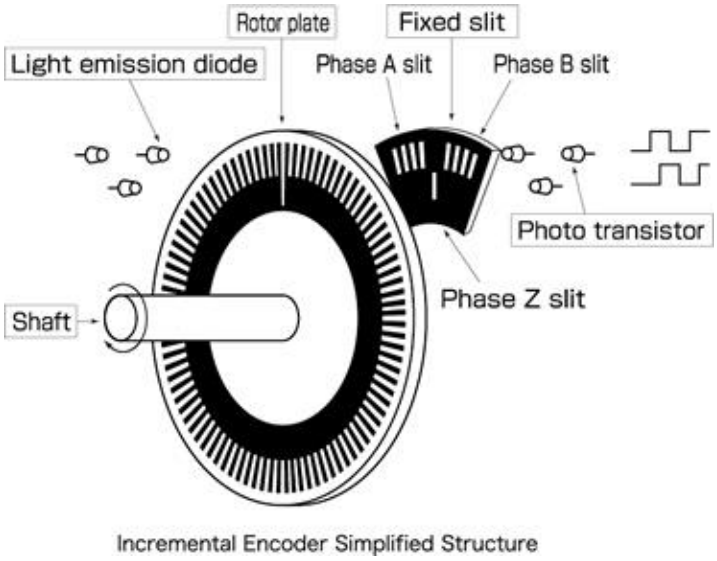
V modulu jsou kromě integrovaného obvodu osazeny také antiparalelní diody a kondenzátor pro omezení zvlnění napájení, takže k modulu stačí pouze připojit svorky motoru, napájení a logické výstupy z mikrokontroléru. Logické vstupy modulu In1 a In2 slouží ke změně polarity výstupního napětí. Ze schématu je vidět, že vstupy In1 a In2 vždy zablokují signál do jednoho z tranzistorů ve větvi, čímž je zabráněno možnosti zkratu větve. Vstup EnA

při logické nule zamezuje spínání všech tranzistorů. Lze ho proto využít na PWM modulaci výstupního napětí můstku.

## 2.3 Použité senzory

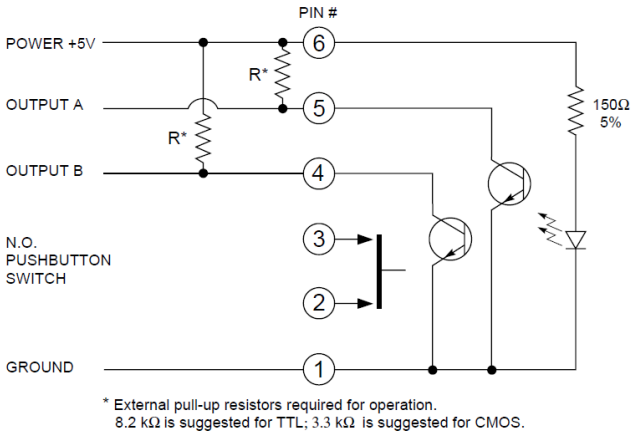
Oba použité senzory jsou optoelektronické inkrementální. "Princip těchto senzorů spočívá v clonění světelného toku mezi zdrojem a fotocitlivými prvky pravítkem (kotoučem) pravidelně rozděleným na úseky pro světlo propustné a nepropustné (kroky). Posuv pravítka (rotoru) o 1 krok (λ) vyvolá přerušení světelného svazku a výstupní signál fotocitlivého detektoru po úpravě na impuls unifikovaného tvaru inkrementuje obsah čítače." [13]

Fotocitlivé prvky senzoru tvoří dva fototranzistory, které jsou navzájem posunuty o $n\lambda + \frac{\lambda}{4}$, kde n je přirozené číslo. Tím pádem při otáčení disku dochází ke vzniku dvou obdélníkových signálů posunutých o 90° (takzvaných kvadraturních signálů). Jejich sled závisí na směru otáčení, což můžeme využít pro zjištění směru rotace (posunu). Před fototranzistory je umístěno stínítko, které zajišťuje, že při určitém úhlu je všechno světlo přicházející na senzor odstíněno.




*Obr. 2.8: Inkrementální senzor úhlu natočení [14]*

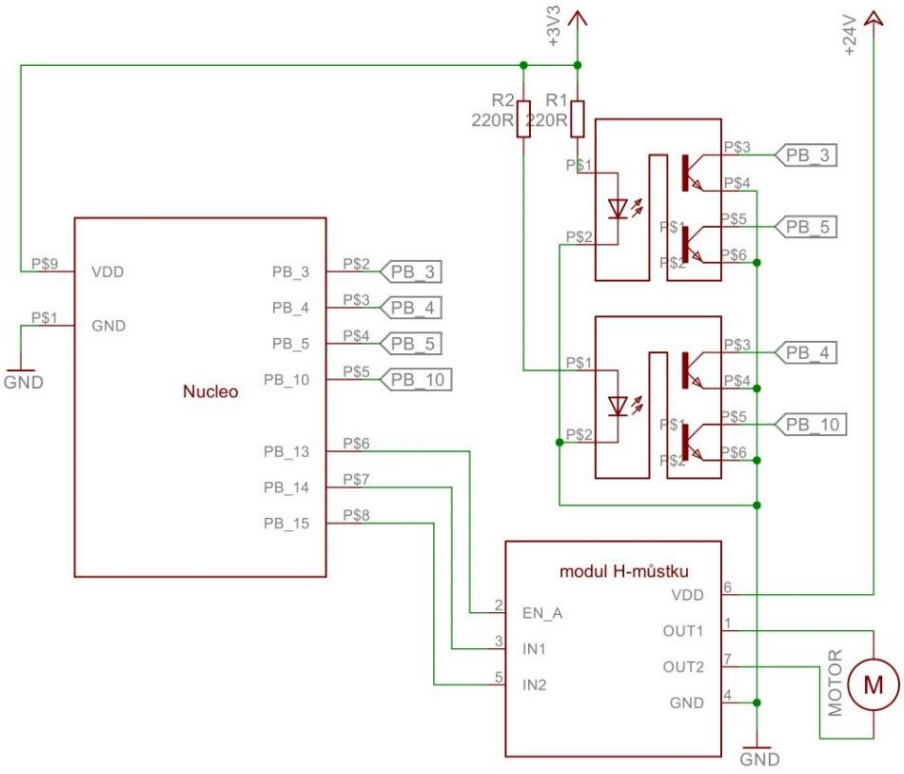
Bohužel kvůli původu senzorů, které pochází ze staré tiskárny, nebylo možné dohledat jejich katalogové listy. Pomocí měření polovodičových přechodů a z polohy vývodů se však podařilo správně zjistit zapojení vývodů. K napájení senzorů je použito napětí 3,3 V, stejně jako v tiskárně, kde byly předtím použity.




Obr. 2.9: *Schéma senzoru (jde o jiný senzor, který je napájen 5V)*

## 2.4 Celkové schéma zapojení

Na obrázku (2.10) je znázorněno zjednodušené celkové schéma zapojení inverzního kyvadla, kde jsou zobrazeny pouze ty výstupy desky Nucleo, které jsme použili.




Obr. 2.10: *Zjednodušené celkové schéma zapojení*

# 3 Řídící program

## 3.1 Použitý mikrokontrolér a vývojové prostředí

Inverzní kyvadlo je řízeno mikrokontrolérem STM32F411RE na vývojové desce Nucleo od firmy STM. STM32F411RE je 32-bitový mikrokontrolér využívající jádro Cortex-M4.

Obr. 3.1: *Deska Nucleo F411RE*

Parametry mikrokontroléru:

- Taktovací frekvence až 100 MHz
- 512 KB flash
- 128 KB SRAM
- 6 16-bitových časovačů, z nichž jeden má speciální funkce pro PWM
- 2 32-bitové časovače
- 3x USART
- Až 3x I2C
- 5x SPI
- Další viz datasheet Nucleo dostupný na webu [15]

Na desce je kromě samotného mikrokontroléru osazen také programátor ST-link, který umožňuje komunikaci s počítačem prostřednictvím portu USB. Toho je využito jak při programování mikrokontroléru, tak při provozu kyvadla, které posílá hodnoty některých

proměnných do PC.

Program je psaný a odlaďovaný ve vývojovém prostředí Atollic TrueSTUDIO, které je pro mikrokontroléry od STM dostupné zdarma. Toto prostředí umožňuje komunikaci počítače s mikrokontrolérem přes programátor ST-link. Také obsahuje mnoho odlaďovacích funkcí.

V programu jsou využity knihovny `stm_core.h` a `nucleo_usart.h` vytvořené a používané pro předmět KAE/MPP. Materiály k tomuto předmětu jsou dostupné na Courseware předmětu[16]. První zmiňovaná knihovna obsahuje funkce pro nastavení výstupů mikrokontroléru, čtení a zápis na výstupech a také nastavení časování. Druhá knihovna slouží ke komunikaci přes sériové rozhraní USART.

## 3.2 Nastavení taktu procesoru

Mikrokontrolér lze taktovat buď vnitřním RC oscilátorem, nebo vnějším zdrojem hodin. Na desce zabudovaný ST-link je taktován krystalovým oscilátorem o frekvenci 8MHz. Tento hodinový signál může využít i mikrokontrolér STM32F411RE, čehož je v programu využito. Signál lze dále pomocí PLL upravit až na 100 MHz.

K nastavení taktu mikrokontroléru je použita funkce `SetClock100MHz()` z knihovny `stm_core.h`. Tato funkce vybere vnější zdroj hodin a vhodně nastaví násobicí a dělicí konstanty PLL tak, aby byl takt na sběrnicích 100 MHz (s výjimkou sběrnice pro periferie APB1, která umožňuje maximální frekvenci 50 MHz).

## 3.3 Čtení polohy ze senzorů

Jak již bylo zmíněno v kapitole 2.3, výstupem inkrementálních snímačů polohy jsou dva kvadraturní obdélníkové signály. Jejich vyhodnocování mikrokontrolérem je založeno na využití externího interruptu. Jelikož je princip obou senzorů stejný, budu dále popisovat pouze senzor snímající polohu vozíku s kyvadlem.

### 3.3.1 Nastavení proměnných, vstupů a interruptů

Nejprve je nutné nadefinovat globální proměnnou typu `volatile` pro polohu vozíku, kterou lze měnit v obsluze interruptu. Hodnota proměnné může být jak kladná, tak záporná.

```
volatile int32_t pos = 0;
```

Dále potřebujeme nastavit výstupy mikrokontroléru pro senzor polohy. Zvolil jsem

výstupy PB_3 a PB_5 na portu GPIOB, které je třeba nastavit jako vstupy s pull-up rezistory. Jejich nastavení je realizováno funkcí z knihovny vytvořené na předmětu MPP, která nastaví potřebné registry GPIO.

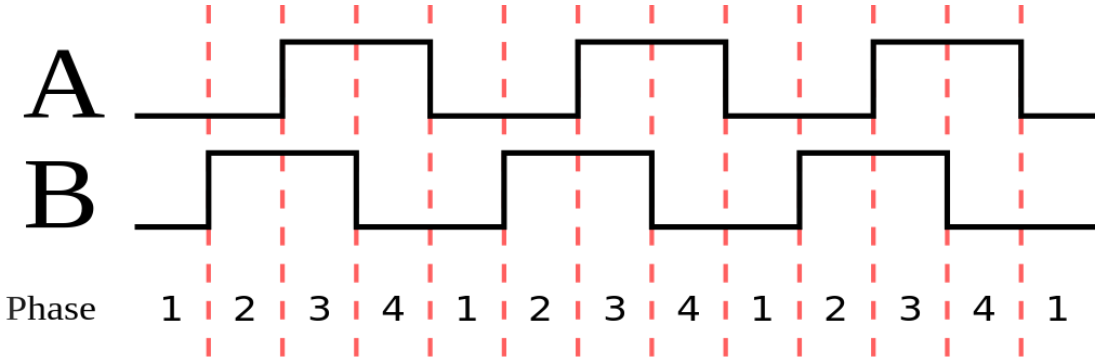
```
/*PINY PRO SENZOR POLOHY*/

  Nucleo_SetPinGPIO(GPIOB, 3, ioPortInputPU);

  Nucleo_SetPinGPIO(GPIOB, 5, ioPortInputPU);
```

Zbývá nastavit externí interrupt na PB_3 aktivovaný náběžnou hranou. Nastavení si můžete prohlédnout ve zdrojovém kódu programu, který je dostupný v příloze.

### 3.3.2 Obsluha externího interruptu




Obr. 3.2: *Kvadraturní signály*

PB_3 je připojen na jeden z kvadraturních signálů, pro ilustraci zvolme signál A (obr. 3.1). V obsluze interruptu pak testujeme logickou úroveň signálu B, který je připojen na PB_5. Je-li signál B při náběžné hraně A v úrovni H, došlo k posunutí vozíku v kladném směru (ten byl zvolen při pohledu zepředu jako směr doprava) a v obsluze interruptu inkrementujeme proměnnou `pos`. V opačném případě proměnnou dekrementujeme.

```
void EXTI3_IRQHandler (void)

{

  if (EXTI->PR &= EXTI_PR_PR3)

    EXTI->PR |= EXTI_PR_PR3; //shodim priznak preruseni nastavenim tohohoto
bitu do 1
```

```
  if(GPIORead(GPIOB, 5))
    pos++;
  else
    pos--;
}
```

## 3.4 Nastavení řízení motoru

K řízení motoru je použita pulzně-šířková modulace. Ta je v mikrokontroléru realizována pomocí šestnáctibitového čítače TIM1, který je pro tyto účely uzpůsoben.

### 3.4.1 Nastavení výstupů

Pro řízení motoru potřebujeme dva výstupy na řízení směru motoru a jeden pro PWM. Výstupy opět nastavíme pomocí funkce z knihovny `stm_core.h`.

```
Nucleo_SetPinGPIO(GPIOB, 13, ioPortAlternatePP); // MOTOR PWM PIN
Nucleo_SetAFGPIO(GPIOB, 13, 1); // AF01 odpovida TIM1_CH1N
Nucleo_SetPinGPIO(GPIOB, 14, ioPortOutputPP); // MOTOR DIRECTION PIN1
Nucleo_SetPinGPIO(GPIOB, 15, ioPortOutputPP); // MOTOR DIRECTION PIN1
```

Příkaz na druhém řádku ukázky slouží k nastavení alternativní funkce pinu PB_13. V tabulce s alternativními výstupy v datasheetu STM32F411RE můžeme dohledat, že číslo alternativní funkce PB_13 odpovídající časovači TIM1 se rovná jedné. Všechny výstupy jsou typu push-pull, což znamená, že dokáží protlačit proud oběma směry.

### 3.4.2 Nastavení čítače TIM1

V části programu, kde probíhá nastavení čítače, je nejprve nastavena předdělička tak, aby se čítač inkrementoval každých 0,1 µs. Čítač ve výchozím nastavení čítá směrem nahoru. Do registru `TIM1->ARR` se nastaví hodnota, při které se čítač vynuluje, v našem případě 255. Tím je dána perioda čítání T = 25,6 µs. Podle

$$f = \frac{1}{T}$$

dopočítáme frekvenci, která činí přibližně 39 kHz. Frekvence vyšší, než horní hranice

akustického pásma je vhodná z důvodu omezení nežádoucích akustických projevů motoru.

## 3.5 Nastavení časovače SysTick

Pro časování regulátorů je potřeba vytvořit časovou základnu. Ta je realizována pomocí časovače zvaného SysTick. SysTick je tvořen 24-bitovým časovačem, který čítá směrem dolů. Při přechodu z 1 do 0 se nastavuje příznak dopočítání a při přechodu z 0 se do čítače načte obsah reload registru.

SysTick nastavíme pomocí funkce `SysTick_Config()` z knihovny CMSIS (Cortex Microcontroller Software Interface Standard). Tato funkce vyžaduje jako parametr počet impulzů, které má SysTick napočítat. Pro regulační smyčky byla zvolena časová základna 1 ms. Potřebný počet hodinových impulzů získáme z taktu procesoru děleného počtem milisekund v jedné sekundě:

```
SysTick_Config(GetBusClock(busClockAHB) / 1000);
```

V přerušení od časovače SysTick nastavíme proměnnou `action` (typu volatile bool) do hodnoty 1. Tato proměnná nese informaci o tom, zda došlo k přerušení a je opakovaně testována v hlavní smyčce.

## 3.6 Nastavení USART

Ke komunikaci mikrokontroléru s PC slouží sériové rozhraní USART. Komunikace respektive funguje tak, že pomocí USART mikrokontrolér předává informace ST-linku, který komunikuje s PC přes USB.

K nastavení rozhraní USART použijeme funkci `Usart2Init()` z knihovny `nucleo_usart.h`. Tato funkce požaduje jako vstup hodnotu modulační rychlosti udávanou v Bd, v našem případě je použita hodnota 38400 Bd. Funkce `Usart2Init`() za nás nastaví výstupy mikrokontroléru pro USART, nastaví vhodnou dělicí konstantu podle frekvence hodin a zvolené modulační rychlosti a nakonec povolí periferii.

## 3.7 Hlavní smyčka programu

V hlavní smyčce programu se neustále testuje proměnná `action`. Tato proměnná je nastavena do hodnoty 1 každou milisekundu prostřednictvím časovače SysTick. Byla-li proměnná `action` nastavena do hodnoty 1, jsou vykonány potřebné výpočty a poslány některá

data do PC. Zároveň je proměnná `action` vynulována. Je-li proměnná action rovna nule, testování se opakuje.

```
if (action == 1)
    {
      action = 0; //shodit priznak akce
      /* vypocty regulace; */
    }
```

Tím je dosaženo toho, že časový interval mezi výpočty regulace je konstantní a o definované délce. Nutnou podmínkou přesného časování je, že veškeré operace vykonané v bloku `if` musí trvat méně než 1 ms.

### 3.7.1 Výpočty regulace a řízení motoru

Jak bylo již vysvětleno v kapitole (1.2.2), jsou použity dvě regulační smyčky. Vnitřní smyčka slouží k regulaci úhlu kyvadla, jako akční veličinu používá hodnotu PWM, kterou zjednodušeně považujeme za úměrnou momentu motoru. Vnější regulační smyčka reguluje polohu vozíku prostřednictvím upravování požadovaného úhlu kyvadla.

#### 3.7.1.1 PID regulace úhlu kyvadla

V části programu pro PID regulaci je nejprve vypočítána chyba úhlu `angleError` jako rozdíl požadovaného a skutečného úhlu. Chyba úhlu je celočíselná, protože úhel je měřen v celých číslech. Požadovaný úhel je výstupem druhé regulační smyčky, popřípadě ho lze zadat jako konstantu odpovídající úhlu vzpřímeného kyvadla.

Dále je spočítána integrační složka regulátoru. Integrační složka je realizována nejjednodušším možným způsobem jako prostá suma. Proměnná `angleIntegral` se vždy inkrementuje o chybu úhlu vynásobenou časovým elementem regulace. Integrální složka je následně limitována, aby nemohla přesáhnout patřičné meze.

```
angleIntegral = angleIntegral + (float)angleError* 10E-3;
```

```
if(angleIntegral > integralLimit)
  angleIntegral = integralLimit;
if(angleIntegral < - integralLimit)
  angleIntegral = - integralLimit;
```

Derivační složka je počítána jako rozdíl chyby úhlu v aktuálním a v posledním uplynulém časovém kroku, proto je nutné mít proměnnou na uložení minulé hodnoty chyby `lastTimeAngleError.`

Derivační složka představovala problém. Senzor úhlu má 2150 dílků na otáčku. Změna úhlu o jeden dílek za 1 ms odpovídá úhlové rychlosti 0,465 ot/s. Při této a nižší rychlosti otáčení by byla derivační složka menší nebo rovna jedné. Jelikož je úhel počítán v celých číslech (z principu funkce inkrementálního senzoru), byla by derivační složka při malých rychlostech otáčení zatížena velkou zaokrouhlovací chybou.

Řešením tohoto problému bylo počítat derivační složku ne každou jednu milisekundu, ale v delším časovém intervalu. Proto byla přidána proměnná `cycleCount,` která se během každého průběhu výpočtů regulace inkrementuje.

Před výpočtem derivační složky pak testujeme, je-li proměnná `cycleCount` beze zbytku dělitelná zvoleným časem mezi výpočty derivace. V kladném případě je spočítána derivační složka.

```
if (cycleCount % angleDerivativeTime == 0)
{
  angleDerivative = angleError - lastTimeAngleError;
  lastTimeAngleError = angleError;
}
```

Nakonec je spočítána výstupní veličina regulátoru, kterou je střídy PWM pro motor (o které předpokládáme, že je přímo úměrná proudu). Ta je vypočtena jako součet výstupů jednotlivých složek P, I a D násobených jejich konstantami. Tato hodnota je poté omezena na rozsah -255 až 255 a ve znaménku nese též informaci o požadovaném směru otáčení.

Stabilizace kyvadla funguje i v případě, že je použita pouze tato vnitřní regulační smyčka, jen má kyvadlo tendenci „ujíždět“ pryč. Z toho důvodu je nutné zavést druhou regulační smyčku polohy vozíku.

#### 3.7.1.2 PD regulace polohy vozíku

PD regulace je implementována obdobně, jako PID popsaná výše. Požadavek na polohu je nastaven na nulu, což znamená, že se regulátor snaží udržovat vozík v poloze, ve které byl při restartu. Výstup je dán součtem:

```
requiredAngle = DivPerRevolution/2 + (-posError*posKp)/1000 +
                (-posDerivative*posKd)/1000;
```

kde však máme navíc člen `DivPerRevolution/2.` Mínusy jsou způsobeny opačným zapojením kvadraturních výstupů senzoru polohy. Výstupem regulace je požadovaný úhel sloužící jako vstup pro PID regulaci.

V proměnné `DivPerRevolution` je uložen počet dílků na otáčku inkrementálního senzoru úhlu kyvadla. Při zapnutí, respektive restartu programu, musí kyvadlo volně viset dolů. Horní poloha, ve které chceme kyvadlo stabilizovat je o polovinu otáčky pootočená. Proto musíme k požadovanému úhlu přičíst počet dílků senzoru na otáčku vydělenou dvěma. Příspěvky od P a D složek regulátorů jsou děleny tisícem, protože je potřeba použít malých čísel a chceme omezit množství výpočtů s plovoucí řádovou čárkou (z důvodů rychlosti).

K regulaci polohy vozíku je použita regulační smyčka bez integrační složky. Integrační složka regulátoru je vynechána z důvodu nepřesnosti senzoru úhlu kyvadla. Ten totiž občas ztratí několik kroků, a tak se změní hodnota úhlu odpovídajícího horní poloze kyvadla.

Představme si situaci, kdy regulace polohy vozíku obsahuje i integrační složku a dojde k chybě měření úhlu kyvadla. Vozík by nejprve najel do polohy, kde by proporční složka regulace úhlu byla rovna chybě úhlu a v této poloze by kyvadlo bylo stabilizované. S rostoucím časem by ale rostla integrační složka regulace polohy, která by se čím dál víc „snažila“ vozík posunout do výchozí polohy. V této poloze by ale byl požadovaný úhel na kyvadlo rozdílný od úhlu odpovídajícímu svislé poloze, čímž by se kyvadlo začalo převažovat a došlo by k nestabilitě.

Je-li však integrační složka vynechána, pak při chybě senzoru úhlu dojde k posunutí

rovnovážné polohy vozíku s kyvadlem. Tím sice vozík nesetrvává stále na stejném místě, avšak zabráníme tím vzniku nestability.

Nejlepším řešením zmíněného problému by bylo použití uzavřeného absolutního optického enkodéru, který eliminuje rušivé vlivy a nemůže se mu stát, že ztratí krok.

Chybovost použitého senzoru lze ukázat jednoduchým pokusem, kdy se kyvadlo ručně otočí o několik otáček jedním a o stejný počet otáček druhým směrem. Výsledný úhel potom není nulový, jak bychom očekávali, ale liší se o několik jednotek dílků. Chybovost senzoru je pravděpodobně způsobena diskem, který je na několika místech poškrábaný. Dalšími možnými důvody nepřesnosti jsou ne zcela přesná konstrukce senzoru a také dopadající okolní záření na senzor.

#### 3.7.1.3 Řízení motoru

Pro samotné řízení motoru je nutné pomocí informace z proměnné `motPWM` nastavit výstupy mikrokontroléru tak, aby byly „srozumitelné“ pro H-můstek. Nejprve je třeba nastavit výstupy PB_14 a PB_15, které jsou připojeny na vstupy In1 a In2 H-můstku a ovládají směr otáčení. Ty nastavíme vzájemně do opačné hodnoty na základě znaménka `motPWM.`

```
GPIOWrite(GPIOB, 14, (motPWM > 0));
GPIOWrite(GPIOB, 15, (motPWM < 0));
```

Absolutní hodnota proměnné `motPWM` je dále přepočítána z rozsahu 0 – 255 do rozsahu `PWM_min - PWM_max`. Rozsah po přepočítání je zvolený na 80 – 255. Přepočítání je použito z důvodu, že při hodnotě střídy nižší než 80 nemá motor ještě dost momentu, aby dokázal překonat třecí síly vozíku. Použití minimální meze pro střídu motoru mělo na řízení kyvadla znatelně pozitivní vliv. Přepočítaná hodnota PWM je poté uložena do registru `CCR1` časovače `TIM1`.

```
TIM1->CCR1 = motPWMLimited;
```

Výstup čítače v PWM režimu zůstává v aktivní úrovni, dokud je hodnota čítače nižší, než hodnota uložená v `TIM1->CCR1.` Vzhledem k tomu, že čítač čítá směrem nahoru, je hodnota v tomto registru přímo úměrná velikosti střídy, kdy hodnota 255 znamená střídu 100%.

### 3.7.2 Tisk hodnot do PC

Ve hlavní smyčce probíhá také posílání několika hodnot do PC. Komunikace je realizována přes sériové rozhraní USART. K tisku dochází jednou za 200 ms, k čemuž je opět využita proměnná `cycleCount` a příkaz modulo. Tisk výstupních hodnot byl nepostradatelný zejména při ladění programu.

```
if (cycleCount%200 == 0) //kazdy dvousty cyklus odeslat po USART (5x za
                          sekundu)

      {

        printf("%d\t%d\t%d\t%d\t%d\r\n",  (int)angle,  (int)requiredAngle,
(int)pos, (int)motPWM, (int)posDerivative);

      }
```

# 4 Naladění konstant regulátorů a provoz modelu kyvadla

Jako první byla laděna vnitřní smyčka regulující úhel kyvadla. Regulace polohy vozíku byla přitom zcela vypnuta a požadovaný úhel byl tedy neměnný. PID regulátor byl nastavován manuálně podle postupu uvedeném na webu [17]. Nejprve byly všechny konstanty nastaveny na nulu. Poté jsme zvyšovali proporcionální konstantu tak dlouho, až začalo být kyvadlo nestabilní a oscilovat. Proporcionální konstanta pak byla nastavena zhruba na polovinu hodnoty při oscilacích. Dále se nastavila integrační konstanta. Při jejím zvyšování se zrychlovala reakce kyvadla až do úrovně, kdy docházelo k mírným oscilacím. Nakonec byla zvyšována derivační konstanta tak dlouho, až oscilace vymizely. Hodnoty konstant byly zaznamenány do tabulky.

Tabulka 2: *experimentálně nastavené konstanty PID regulace úhlu*

| | |
|---|---|
| $K$ | 30 |
| $K_I$ | 10 |
| $K_D$ | 45 |

Poté byla nastavována vnější smyčka, tedy regulace polohy vozíku. Ladění probíhalo obdobně, pouze chyběla integrační složka. Hodnoty byly zaznamenány do tabulky 3.

Tabulka 2: *experimentálně nastavené konstanty PD regulace polohy vozíku*

| | |
|---|---|
| $K$ | 0,045 |
| $K_D$ | 0,14 |

Pro demonstraci funkce inverzního kyvadla nejprve zapojíme mikrokontrolér přes USB do PC, kde spustíme sériový terminál o modulační rychlost 38400 Bd na příslušném portu. Vozík s kyvadlem nastavíme do středu dráhy a necháme kyvadlo ustálit se ve visu. Poté restartujeme mikrokontrolér, čímž vynulujeme hodnoty senzorů. Kyvadlo pak ručně pootočíme

o půl otáčky proti směru hodinových ručiček a zapneme napájení. Motor pak pohybuje s vozíkem tak, aby udržoval kyvadlo v horní poloze (obr 4.1) a je schopen ho stabilizovat i při případném působení rušivých sil na kyvadlo.

Obr. 4.1: *Model inverzního kyvadla v provozu*

# Závěr

V práci jsem nejprve shrnul konstrukční možnosti řešení modelu inverzního kyvadla. Dále jsem provedl odvození rovnic inverzního kyvadla a nastínil možnosti řízení. Podrobněji jsem popsal regulaci pomocí dvou regulačních smyček PID a PD, kterou v kyvadle implementuji. Následně jsem se zabýval praktickým řešením konstrukce inverzního kyvadla.

Během vývoje kyvadla jsem jsem nejprve vytvořil prototyp pojezdu se stejnosměrným motorem, abych ověřil možnost regulace polohy pomocí PID se zpětnou vazbou od inkrementálního senzoru. Tento pojezd však nebyl příliš přesný, protože vodící tyče nebyly zcela rovnoběžné. Vinou toho kladl vozík příliš velký mechanický odpor. Proto jsem vytvořil nynější exemplář s využitím přesnějších dílů vytištěných na 3D tiskárně. Vozík se na tomto pojezdu posunuje již zcela lehce. Až po odladění pojezdu jsem na vozík připevnil kyvadlo a začal experimentovat s jeho stabilizací.

Software byl nejprve z důvodu jednoduššího programování implementován na desce Arduino. Regulace zde fungovala, ale čas výpočtu a tedy i časový krok regulace byl pravděpodobně mnohem delší. Po ověření principiální funkčnosti byl kód přepsán do nynější podoby na 32-bitový mikrokontrolér STM32F411RE s jádrem Cortex-M4. Velkou výhodou oproti osmibitovému Arduinu je mnohem větší rychlost, která je způsobena jednak 32-bitovou aritmetikou jádra M4 a také vyšší taktovací frekvencí. Díky tomu bez problémů funguje krok algoritmu regulace 1 ms, a to i v případě, že mikrokontrolér posílá hodnoty do PC po sériové lince. Další výhodou použitého mikrokontroléru je větší kontrola nad programem.

Model kyvadla po mechanické stránce funguje výborně, díky dostatečnému výkonu motoru a hladce fungujícímu pojezdu je vozík s kyvadlem na pohled velmi rychlý. Regulace je schopna se vypořádat i s vnějšími silami působícími na kyvadlo (například “šťouchnutí” prstem do kyvadla).

Model však také vykazuje některé nedokonalosti a poskytuje mnoho příležitostí pro případná vylepšení. Hlavním nedostatkem zařízení je inkrementální snímač úhlu kyvadla, který je zatížen chybou rostoucí s časem provozu kyvadla. Bylo by vhodné jej nahradit absolutním optickým enkodérem, u kterého je nárůst chyby v čase vyloučen.

Dále se nabízí možnost použití složitějšího algoritmu řízení založeného na matematickém modelu, který by zcela jistě vylepšil dynamiku systému. Zvláště se nabízí možnost experimentovat s regulátorem pracujícím ve fuzzy logice, která se na FEL ZČU vyučuje v předmětu KAE/ANF. Pro kyvadlo by také bylo zajímavé vytvořit programově řízené automatické „vyšvihnutí“ do horní polohy.

# Seznam literatury a informačních zdrojů

[1] obrázek dostupný z: http://ctms.engin.umich.edu/CTMS/Content/InvertedPendulum/System/Modeling/figures/pendulum.png

[2] obrázek dostupný z: https://www.researchgate.net/profile/Tang_Teng_Fong/publication/285143485/figure/fig1/AS:301414048911360@1448874117111/Simplified-rotary-inverted-pendulum12-Table-1-Mechanical-and-electrical-system.png

[3] A Flying Inverted Pendulum. *Youtube* [online]. [cit. 2018-06-06]. Dostupné z: https://www.youtube.com/watch?v=15DIidigArA

[4] Triple Pendulum on a Cart. *Youtube* [online]. [cit. 2018-06-06]. Dostupné z: https://www.youtube.com/watch?v=cyN-CRNrb3E

[5] TŮMA, František. *Automatické řízení 1: lineární spojité dynamické systémy*. 3., upr. vyd. V Plzni: Západočeská univerzita, 2007. ISBN 978-80-7043-568-7. s.206

[6] obrázek dostupný z: https://upload.wikimedia.org/wikipedia/commons/thumb/4/40/Pid-feedback-nct-int-correct.png/1200px-Pid-feedback-nct-int-correct.png

[7] FRANKLIN, Gene F., J. David POWELL a Abbas EMAMI-NAEINI. *Feedback control of dynamic systems*. Upper Saddle River: Prentice Hall, c2002. Chronological history of feedback control. ISBN 9780135001509. s.55

[8] OŽAŇA, Štěpán. *NAVRHOVÁNÍ A REALIZACE REGULÁTORŮ* [online]. Ostrava, 2012 [cit. 2018-06-06]. Dostupné z: http://www.person.vsb.cz/archivcd/FEI/NRR/Navrhovani%20a%20realizace%20regulatoru.pdf. Učební text. Vysoká škola báňská – Technická univerzita Ostrava.

[9] Krokový motor. *Pohonná technika* [online]. [cit. 2018-06-06]. Dostupné z: http://www.pohonnatechnika.cz/skola/motory/krokovy-motor

[10] RS-455PA-15205. *Mabuchi motor* [online]. [cit. 2018-06-06]. Dostupné z: https://product.mabuchi-motor.com/detail.html?id=107

[11] obrázek dostupný z: http://www.modularcircuits.com/blog/articles/h-bridge-secrets/h-bridges-the-basics/

[12] DUAL FULL-BRIDGE DRIVER. *Sparkfun* [online]. [cit. 2018-06-06]. Dostupné z: https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf

[13] RIPKA, Pavel. *Senzory a převodníky*. 2. vyd. V Praze: České vysoké učení technické, 2011. ISBN 978-80-01-04696-8. s.32

[14] obrázek dostupný z: https://www.electronicshub.org/wp-content/uploads/2015/02/structure-of-an-incremental-encoder.jpg

[15] NUCLEO-F411RE. *ST* [online]. [cit. 2018-06-06]. Dostupné z: http://www.st.com/en/evaluation-tools/nucleo-f411re.html

[16] KAE/MPP. *CourseWARE* [online]. [cit. 2018-06-06]. Dostupné z: https://courseware.zcu.cz/portal/studium/courseware/kae/mpp

[17] PID controller. *Wikipedia* [online] [cit. 2018-06-06]. Dostupné z: https://en.wikipedia.org/wiki/PID_controller

# Přílohy

## Příloha A – Hlavní funkce programu řídícího inverzní kyvadlo

```
/* Includes */
#include "mpp_shield.h"
#include <nucleo_usart.h>

/* Private macro */
/* Private variables */

volatile int32_t pos = 0;
volatile int32_t angle = 0;
volatile bool action = 0;

int32_t cycleCount = 0;


int32_t motPWM = 0;
int32_t motPWMLimited = 0;
int32_t PWM_min = 80;
int32_t PWM_max = 255;
int32_t direction = 0;


int32_t angleError = 0;
int32_t lastTimeAngleError = 0;
float angleIntegral = 0;
int32_t angleDerivative = 0;


int32_t posError = 0;
int32_t lastTimePosError = 0;
int32_t posDerivative = 0;


int32_t requiredPosition = 0;
int32_t requiredAngle = 0;

// PROMENNE PD REGULATORU polohy
int32_t posKp = 45; // skutečná posKp = posKp/1000
int32_t posKd = 140; // // skutečná posKd = posKd/1000
int32_t DivPerRevolution = 2150;


// PROMENNE PID REGULATORU UHLU
int32_t angleKp = 30; //30
float angleKi = 10;
int32_t angleKd = 45; //45

int32_t angleIntegralLimit = 20;
int32_t angleDerivativeTime = 5; //v ms

/* Private function prototypes */
/* Private functions */


void SysTick_Handler (void)
{
  action = 1;
}


void EXTI3_IRQHandler (void)
{

  if (EXTI->PR &= EXTI_PR_PR3)
    EXTI->PR |= EXTI_PR_PR3; //shodim priznak preruseni nastavenim tohohoto bitu do 1

  if(GPIORead(GPIOB, 5))
    pos--;
  else
    pos++;
```

```
}

void EXTI4_IRQHandler (void)
{

  if (EXTI->PR &= EXTI_PR_PR4)
    EXTI->PR |= EXTI_PR_PR4; //shodim priznak nahozenim tohohle bitu

  if(GPIORead(GPIOB, 10))
    angle++;
  else
    angle--;
}

int main(void)
{

  SetClock100MHz(clockSourceHSE);
  SystemCoreClockUpdate();

  /*PINY PRO SENZOR POLOHY*/
  Nucleo_SetPinGPIO(GPIOB, 3, ioPortInputPU);
  Nucleo_SetPinGPIO(GPIOB, 5, ioPortInputPU);

  /*PINY PRO SENZOR UHLU*/
  Nucleo_SetPinGPIO(GPIOB, 4, ioPortInputPU);
  Nucleo_SetPinGPIO(GPIOB, 10, ioPortInputPU);

  /*PINY PRO MOTOR*/
  Nucleo_SetPinGPIO(GPIOB, 13, ioPortAlternatePP); // MOTOR PWM PIN
  Nucleo_SetAFGPIO(GPIOB, 13, 1); // AF01 odpovida TIM1_CH1N
  Nucleo_SetPinGPIO(GPIOB, 14, ioPortOutputPP); // MOTOR DIRECTION PIN1
  Nucleo_SetPinGPIO(GPIOB, 15, ioPortOutputPP); // MOTOR DIRECTION PIN1

  SysTick_Config(GetBusClock(busClockAHB) / 1000); //nastavit systick

  /*POVOLENI APB2 PRO TIM1*/
  if (!(RCC->APB2ENR & RCC_APB2ENR_TIM1EN))
  {
    RCC->APB2ENR |= RCC_APB2ENR_TIM1EN;
    RCC->APB2RSTR |= (1<<0); //RCC_APB2RSTR_TIM1RST z nějakýho důvodu nejde
    RCC->APB2RSTR &= ~(1<<0);
  }

  /*NASTAVENI TIM1*/
  TIM1->CR1 = 0 // DIR = 0 - upcounter
    | TIM_CR1_ARPE; // bufferovany zapis do ARR

  TIM1->CR2 = 0;
  TIM1->PSC = 10 - 1; // 10MHz - 0,1 us
  TIM1->ARR = 256 - 1; // 25,6us preteceni
  TIM1->CR1 = TIM_CR1_CEN;
  TIM1->BDTR |= TIM_BDTR_MOE;
  TIM1->CCMR1 &= ~TIM_CCMR1_OC1M;
  TIM1->CCMR1 |= TIM_CCMR1_OC1M_2 | TIM_CCMR1_OC1M_1; // 110 – PWM 1
  TIM1->CCER |= TIM_CCER_CC1NE; // povol komplementarni vystup
  TIM1->CCR1 = 0;

  /* POVOLENI HODIN PRO SYSCFG, KDE SE NASTAVI EXTERNI INTERRUPTY */
  if (!(RCC->APB2ENR & RCC_APB2ENR_SYSCFGEN))
    {
      RCC->APB2ENR |= RCC_APB2ENR_SYSCFGEN;
      RCC->APB2RSTR |= RCC_APB2RSTR_SYSCFGRST;
      RCC->APB2RSTR &= ~RCC_APB2RSTR_SYSCFGRST;
    }

  /* NASTAVENI EXTERNIHO INTERRUPTU NA PB3 */
  SYSCFG->EXTICR[0] |= SYSCFG_EXTICR1_EXTI3_PB; //external interrupt na PB3

  EXTI->IMR |= EXTI_IMR_MR3; //vstup 3 neni maskovany
  EXTI->RTSR |= EXTI_RTSR_TR3; //rising edge detekce na 3

  NVIC_EnableIRQ(EXTI3_IRQn); //dovol ext. interrupt 3
```

```
/* NASTAVENI EXTERNIHO INTERRUPTU NA PB4 */
SYSCFG->EXTICR[1] |= SYSCFG_EXTICR2_EXTI4_PB;

EXTI->IMR |= EXTI_IMR_MR4;
EXTI->RTSR |= EXTI_RTSR_TR4;

NVIC_EnableIRQ(EXTI4_IRQn);

Usart2Init(38400);

while(1)

{

  if (action == 1)
  {
    action = 0; //shodit priznak akce

    /*POSITION PD REGULATION*/

    posError = requiredPosition - pos;

    if (cycleCount%100 == 0)
    {
      posDerivative = posError - lastTimePosError;
      lastTimePosError = posError;
    }

    requiredAngle = DivPerRevolution/2 - (posError*posKp)/1000 - (posDerivative*posKd)/1000;

    /*ANGLE PID REGULATION*/

    angleError = requiredAngle - angle;

    angleIntegral = angleIntegral + (float)angleError/1000;

    if(angleIntegral > angleIntegralLimit)
      angleIntegral = angleIntegralLimit;
    if(angleIntegral < -angleIntegralLimit)
      angleIntegral = -angleIntegralLimit;

    if (cycleCount % angleDerivativeTime == 0)
    {
      angleDerivative = angleError - lastTimeAngleError;
      lastTimeAngleError = angleError;
    }

    motPWM = (angleError*angleKp + angleIntegral*angleKi + angleDerivative*angleKd);

    /*orezani rychlosti do rozsahu -255 - 255*/
    if(motPWM < -255)
    {
      motPWM = -255;
    }

    if(motPWM > 255)
    {
      motPWM = 255;
    }

    GPIOWrite(GPIOB, 14, (motPWM > 0));
    GPIOWrite(GPIOB, 15, (motPWM < 0));

    // prepocet PWM z rozsahu 0-255 do rozsahu PWM_min - PWM_max

    if (motPWM>0)
      motPWMLimited = (motPWM*(PWM_max - PWM_min)/(PWM_max) + PWM_min);
    else
      motPWMLimited = ((-motPWM)*(PWM_max - PWM_min)/(PWM_max) + PWM_min);

    TIM1->CCR1 = motPWMLimited;

    cycleCount++;
```

```
      if (cycleCount%200 == 0) //kazdy dvousty cyklus vytisknout uhel a polohu (5x za sekundu)
      {
        printf("%d\t%d\t%d\t%d\t%d\r\n", (int)angle, (int)requiredAngle, (int)pos, (int)motPWM,
(int)motPWMLimited);
      }

    }

  }


}
```