

# **Dálkové ovládání brány s GSM modulem**

Bc. Tomáš Smékal



# ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš Smékal**  
Osobní číslo: **A12479**  
Studijní program: **N3902 Inženýrská informatika**  
Studijní obor: **Informační technologie**  
Forma studia: **kombinovaná**

Téma práce: **Dálkové ovládání brány s GSM modulem**  
Téma anglicky: **The Remote-control of Gates Using a GSM Unit**

Zásady pro vypracování:

1. Zpracujte literární rešerši na téma elektricky ovládaných bran.
2. Vyberte vhodnou kombinaci technických prostředků určených k realizaci zadání práce.
3. V rámci teoretické části zpracujte podrobný přehled vybraných HW prostředků
4. Navrhněte model brány, na kterém lze vyzkoušet a odladit navrhovaný systém řízení brány.
5. Navrhněte způsob zabezpečení brány proti neoprávněnému vstupu a poškození.
6. Na základě navrženého způsobu zabezpečení brány vyberte vhodné senzory zajišťující požadovanou míru zabezpečení.
7. Navržený model realizujte.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **GARG, Vijay Kumar a Joseph E WILKES.** Principles and applications of GSM. Upper Saddle River, N.J.: Prentice Hall PTR, c1999, xxiii, 481 p. ISBN 01-394-9124-4.
2. **Sensor technology handbook.** Editor Jon S Wilson. Burlington: Elsevier, c2005, ix, 691 s. Newnes. ISBN 978-0-7506-7729-5.
3. **MAZIDI, Muhammad Ali, Sarmad NAIMI a Sepehr NAIMI.** The AVR microcontroller and embedded systems: using Assembly and C. Editor Jon S Wilson. Upper Saddle River, N.J.: Prentice Hall, c2011, xiv, 776 p. Newnes. ISBN 01-380-0331-9.
4. **BARRETT, Steven F.** Arduino microcontroller: processing for everyone!. 2nd ed. San Rafael, Calif.: Morgan, 2012. ISBN 978-160-8458-592.
5. **NICOLESCU, Gabriela.** Model-based design for embedded systems. Boca Raton: CRC Press, c2010, xxiv, 739 s. ISBN 978-1-4200-6784-2.
6. **GSM, GPRS, and edge performance: evolution towards 3G/UMTS.** 2nd ed. Editor Timo Halonen, Javier Romero, Juan Melero. Chichester: John Wiley, 2003, xxxvii, 615 s. ISBN 04-708-6694-2.

Vedoucí diplomové práce:

**Ing. Martin Pospíšilík, Ph.D.**

Ústav počítačových a komunikačních systémů

Datum zadání diplomové práce:

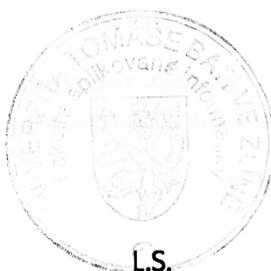
**6. února 2015**

Termín odevzdání diplomové práce:

**15. května 2015**

Ve Zlíně dne 6. února 2015

doc. Mgr. Milan Adámek, Ph.D.  
*děkan*



doc. Mgr. Roman Jašek, Ph.D.  
*ředitel ústavu*


### Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnaní případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

### Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

..........  
podpis autora

## **ABSTRAKT**

Práce pojednává o branách používaných v současnosti a o technologiích, které se u nich využívají. V projektu byl proveden návrh ovládání brány pomocí GSM modulu, otevírání a zavírání brány, návrh senzorů pro zjišťování stavu brány a její zabezpečení pomocí odesílání zprávy majiteli na základě tohoto stavu. Funkčnost projektu byla ověřena návrhem a stavbou modelu brány. Jako ovládací jednotka brány je využito mikroprocesorové desky Arduino UNO a GSM modul s čipem SIM900. Pro připojení senzorů a elektroniky pro řízení motoru, byl vytvořen shield, který je připojen k desce s mikroprocesorem.

Klíčová slova: GSM, Brána, Arduino

## **ABSTRACT**

In this thesis we discuss recent gates and used technology. GSM module for opening and closing the gate, design of sensors for detecting the state of the gate and it's safety via sending messages to the owner based on these states were created as a parts of the project. Functionality was tested by designing and creating the model of gate. Arduino UNO microprocessor board and GSM module with SIM900 chip were used as gate control unit. For connection of sensors and electronics of engine control the shield connected to the microprocessor board was created.

Keywords: GSM, Gate, Arduino

Děkuji panu Ing. Martinu Pospíšilíkovi, Ph.D. za odborné vedení bakalářské práce a mnoho cenných rad a podnětů. A také své rodině za trpělivost.

## OBSAH

<b>ÚVOD .....</b>	<b>9</b>
<b>I    TEORETICKÁ ČÁST .....</b>	<b>9</b>
<b>1    POUŽITÉ TECHNOLOGIE .....</b>	<b>11</b>
1.1    JAZYK C++ .....	11
1.2    GSM .....	11
1.2.1    Historie [10] .....	12
1.2.2    Struktura sítě [13].....	13
1.3    GSM SHIELD S ČÍPEM SIM900 .....	16
1.4    ARDUINO .....	17
1.4.1    Arduino UNO .....	18
1.5    VÝVOJOVÉ PROSTŘEDÍ.....	19
1.5.1    Arduino 1.0.5.....	19
1.5.2    Visual Studio Community 2013 s Visual Micro.....	19
1.6    TARGET 3001!.....	19
<b>2    EXISTUJÍCÍ ŘEŠENÍ.....</b>	<b>21</b>
2.1    TYPY BRAN .....	21
2.1.1    Posuvné brány .....	21
2.1.2    Křídlové brány .....	22
2.2    FOTOBUŇKY .....	23
2.3    POHONNÁ JEDNOTKA .....	24
2.4    MAJÁK .....	24
2.5    DÁLKOVÉ OVLÁDÁNÍ [20] .....	24
2.5.1    GSM.....	24
<b>II    PROJEKTOVÁ ČÁST.....</b>	<b>25</b>
<b>3    MODEL BRÁNY .....</b>	<b>27</b>
3.1    MECHANIKA.....	27
3.2    ELEKTRONIKA.....	27
3.2.1    Napájení .....	28
3.2.2    Ovládání motoru .....	29
3.2.3    Senzory .....	29
3.3    DESKA PLOŠNÝCH SPOJŮ .....	32
<b>4    SOFTWARE .....</b>	<b>35</b>
4.1    LOGICKÉ ROZVRŽENÍ .....	35

4.1.1	ICommunication.....	35
4.1.2	GSMCommunication.....	35
4.1.3	SerialCommunication .....	36
4.1.4	Gate .....	36
4.1.5	EepromAddresses.h.....	38
4.1.6	Hlavní smyčka kódu.....	38
4.1.7	Knihovna GSM_GPRS_GPS_Shield .....	40
4.2	ZABEZPEČENÍ BRÁNY.....	40
4.3	STAV BRÁNY .....	41
4.4	UŽIVATELSKÉ PŘÍKAZY .....	41
4.5	MOŽNÁ VYLEPŠENÍ.....	42
<b>ZÁVĚR.....</b>		<b>43</b>
<b>SEZNAM POUŽITÉ LITERATURY .....</b>		<b>44</b>
<b>SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK .....</b>		<b>47</b>
<b>SEZNAM OBRÁZKŮ .....</b>		<b>49</b>
<b>SEZNAM TABULEK .....</b>		<b>50</b>
<b>SEZNAM PŘÍLOH .....</b>		<b>51</b>



## ÚVOD

Člověk musí nejprve vystoupit z vozidla aby si mohl otevřít vjezdová vrata nebo musí mít sluhu, který mu je otevře. Tato situace už dnes není příliš moderní. A proto existují různé pohony pro brány, které se ovládají dálkově a umožní tak uživateli pohodlně ovládat bránu ze sedačky auta. Většina Dálkových ovládání pracuje na principu vysílání kódu, na volných frekvencích 433 MHz nebo 868 Mhz. V této práci se budu zabývat, jakým způsobem by bylo možné realizovat ovládání brány pomocí GSM modulu. Bude popsán princip a návrh modelu brány, na kterém se modul vyzkouší v praxi.

Práce se v teoretické části zabývá technologiemi, které se v průběhu vývoje využily. Patří mezi ně GSM modul s čipem SIM900. Popisuje technologii Arduino a jejich desky. V druhé teoretické části se řeší typy existujících bran a technologie, jaké jsou v současnosti na branách využívány.

Projektová část je rozdělena také na dvě části. Na část s návrhem modelu brány, jaká je zvolena konstrukce a použité senzory. V této části je i popis a návrh elektronické části, včetně návrhu desky plošných spojů. Druhá část se pak zabývá softwarovou stránkou této práce a ukazuje jak je projekt strukturován. Na konci práce je pak pár nápadů, kterými by bylo možné do budoucna celý systém vylepšit.

# I. TEORETICKÁ ČÁST

## 1 POUŽITÉ TECHNOLOGIE

### 1.1 Jazyk C++

Verze jazyka C++ je založena na verzích klasického C a ANSI C. V roce 1980 byl rozšířen o typové kontroly, konverzi funkčních argumentů a o třídy. Původně se rozšíření jmenovalo pouze jako C s třídami, až později v roce 1983 se ujal název C++.

Nejpodstatnější rozšíření jazyka C jsou:

- Definice, kontrola a konverze datových typů z funkčních argumentů
- Přetížení jmen funkcí a operátorů
- Třídy s datovými abstrakcemi, virtuální funkce
- Správa volné paměti s operátory new a delete
- Vícenásobná dědičnost
- Prvek třídy s přístupem protected
- Ukazatel na instanci třídy
- Zpracování výjimek

Jazyk C je vhodný nástroj pro psaní programů pro mikroprocesory. Oproti strojově bližšímu assembleru je mnohem přívětivější pro programátora, protože podporuje strukturované programování. Tím se významně minimalizuje riziko chyb i čas na vývoj aplikací. C++ pak nabízí podporu objektově orientovaného programování. V případě, kdy potřebujeme dosáhnout ještě vyšší rychlosti programu a mít kontrolu nad jeho přesným chodem, lze do funkcí jazyka C i C++ vepsat inline assembler a tím propojit výhody jak jazyka C, tak i assembleru. Můžeme přesně určit do kterých registrů se kdy a co zapíše nebo jaká podmínka se provede.

Jazyky C a C++ mají velkou výhodu v rychlosti provádění a nízkou paměťovou náročnost. [23]

### 1.2 GSM

GSM (Global System for Mobile Communications, původně Groupe Special Mobile) je standard vyvinutý ETSI (European Telecommunications Standards Institute) a popisuje protokoly pro druhou generaci (2G) digitální buňkové sítě a její použití pro mobilní telefony. Od svého vzniku v roce 1982 se stal výchozím celosvětovým standardem pro mobilní komunikaci.

Síť 2G nahradila síť první generace (1G), která byla analogovou mobilní sítí. GSM standard je digitální systém s přepojováním okruhů sítě, jenž je optimalizován pro plně duplexní hlasovou telefonii. Přepojování okruhů je v telekomunikacích rozšířený systém, kdy komunikace probíhá po předem sestaveném okruhu, na rozdíl od počítačových sítí, kde převládá systém v přepojování paketů.

Následně byl vyvinut systém třetí generace (3G) UMTS a standard čtvrté generace (4G)

### 1.2.1 Historie [10]

V roce 1982 byl zahájen vývoj evropského standardu pro digitální mobilní hlasovou telefonii, kdy CEPT (Evropská konference poštovních a telekomunikačních správ) vytvořila Groupe Special Mobile a později stálou skupinu technické podpory se sídlem v Paříži. O pět let později, v roce 1987, 15 zástupců z 13 evropských zemí podepsalo memorandum o porozumění v Kodani k vývoji a nasazení společného mobilního telefonního systému v celé Evropě. Byla stanovena pravidla, aby GSM byla v EU povinnou normou. Toto rozhodnutí nakonec vyústilo v jednotnou, otevřenou a standardní síť.

Ještě téhož roku byla první technická specifikace vydána a schválena ministry ve čtyřech největších zemích EU (Německo, Velká Británie, Francie a Itálie). Systém tak dostal i politickou podporu. Během necelých 9 měsíců byl výbor Groupe Special Mobile přenesen z CEPT na Evropský institut telekomunikačních standardů ETSI.

Souběžně však Francie a Německo podepsaly dohodu o společném vývoji již v roce 1984 a až v roce 1986 se připojily Itálie a Velká Británie. v roce 1986 Evropská komise navrhla vyhrazení pásma 900 MHz pro GSM. První GSM hovor na světě byl uskutečněn 1. července 1991 bývalým finským premiérem Harri Holkeri s Kaarina Suonio (starostka ve městě Tampere).

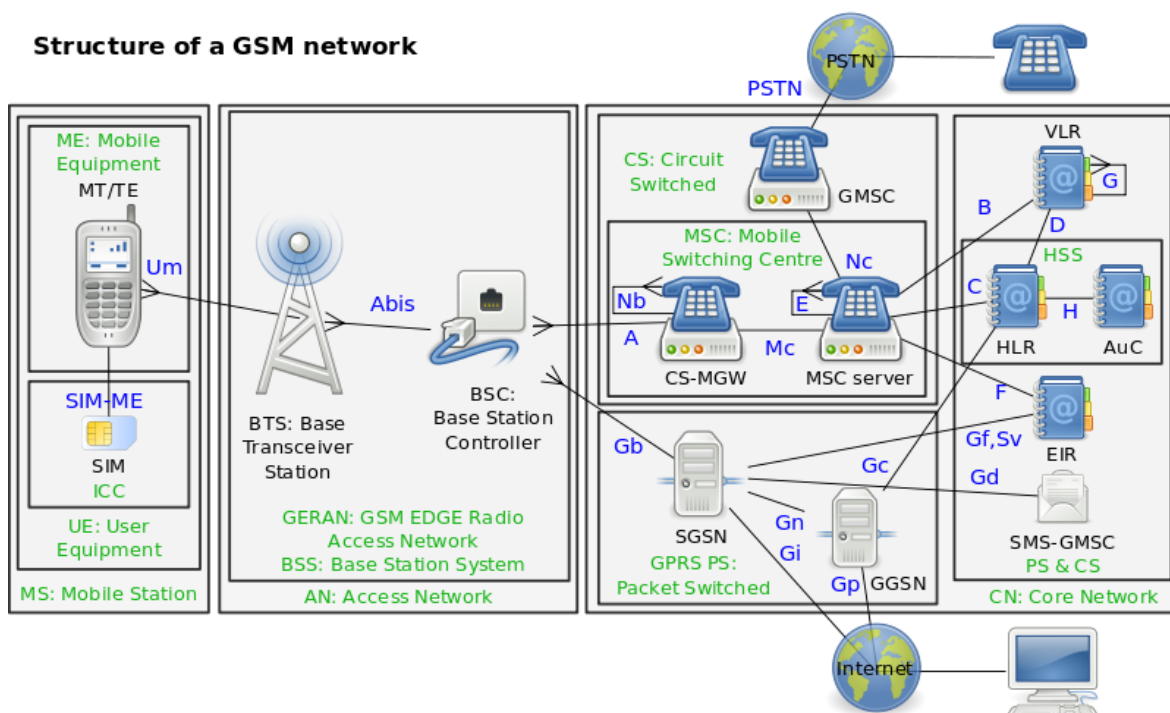
V roce 1991 byl standard rozšířen o normu GSM na frekvenčním pásmu 1800 MHz a první taková síť začala fungovat ve Velké Británii roku 1993. Tento rok, Telecom Austrálie se stal prvním operátorem, který nasadil síť GSM mimo Evropu a byl dán do provozu první dostupný a praktický ruční GSM mobilní telefon.

V roce 2000 byly zahájeny první komerční GPRS služby a prodej prvních mobilních telefonů GPRS.

V roce 2001 byla spuštěna první UMTS (W-CDMA) síť s technologií 3G, která není součástí GSM. Celosvětově počet GSM zařízení přesáhl 500 milionů. v roce 2002 byly zavedeny první multimediální zprávy (Multimedia Messaging Service – MMS).

Do roku 2005 již síť GSM představovaly více než 75% celosvětového trhu s celulárními sítěmi a sloužily pro 1,5 miliardy uživatelů. Tento počet se o pět let později zvýšil více než třikrát a ve více jak 212 zemích dosáhl počet uživatelů 5 miliard.

## 1.2.2 Struktura sítě [13]



Obr. 1.1 Struktura sítě GSM [14]

**ME (Mobile Equipment) Mobilní uživatelská stanice** Mobilní telefon je v podstatě transciever (vysílač/přijímač) komunikující se základnovou stanicí BTS doplněný řídicími obvody a vstupně/výstupními zařízeními (klávesnice, display, sluchátko, mikrofon, porty). Mobilní stanice je jednoznačně identifikována pomocí čísla IMEI (International Mobile Equipment Identity) uloženého v její paměti. Sám účastník je identifikován pomocí SIM karty (Subscriber Identification Module).

**Base Station Subsystem — subsystém základnových stanic a jejich řízení** GSM je buňková síť, což znamená, že mobilní telefony se k ní připojují vyhledáním buňky v jejich bezprostřední blízkosti.

Buňkové sítě mají nesporné výhody oproti jiným řešením:

- Jsou dostatečně flexibilní, aby používaly vlastnosti a funkce dostupné ve většině veřejných i soukromých sítí.
- Mají větší kapacitu přenosu
- Stačí menší vysílací výkon
- Mají větší oblast pokrytí

- Jsou odolnější vůči rušení jinými signály

Existuje pět různých velikostí buněk v GSM síti: makro, mikro, piko, femto a vykrývací buňky. Oblast pokrytí každé buňky se liší podle implementačního prostředí. Makro buňky mohou být považovány za buňky, kde je anténa základnové stanice instalována na stožáru nebo budově nad okolním terénem nebo městskou zástavbou. Mikrobuněk jsou buňky, jejichž výška antény je pod průměrnou střešní úrovní – obvykle v městských oblastech. Pikobuněk jsou malé buňky, jejichž průměr pokrytí je jen pár desítek metrů – používají se hlavně v interiéru. Femtobuněk jsou buňky určené pro použití v obytných nebo malých podnikových prostředích. Vykrývací buňky slouží k vyplnění mezer mezi ostatními buňkami.

Vodorovný dosah buněk se liší v závislosti na výšce antény, zisku antény, a podmínkách šíření od několika set metrů až po desítky kilometrů. Nejdelší vzdálenost specifikuje standard GSM na 35 km.

Celosvětově pracují GSM sítě v mnoha různých frekvenčních rozsazích, v Evropě nejčastěji na 450 MHz, 950 MHz, 1800 MHz a 2100 MHz. Se zvyšující se frekvencí klesá dosah rádiového signálu od 49 km (450 MHz) po 12 km při nejvyšší frekvenci (2100 MHz).

Bez ohledu na frekvenci zvolené operátorem je pásmo rozděleno do časových kanálů (Timeslot) pro jednotlivé telefony. To umožňuje osm kanálů pro plnou šířku pásma nebo šestnáct pro poloviční šířku pásma. Těchto osm rozhlasových timeslotů je seskupeno do TDMA rámce. Poloviční kanály používají alternativní rámce ve stejném časovém úseku. Rychlost datového kanálu pro všech 8 kanálů je 270,833 kbit/s a trvání rámce je 4,615 ms.

Vysílací výkon telefonu je omezen na maximálně 2 watty v GSM 850/900 a 1 watt v GSM 1800/1900.

***Network and Switching Subsystem — síťový spojovací subsystém – páteřní síť***  
Jde vlastně o systém mobilních respektive radiotelefonních ústředen. Na rozdíl od klasických telefonních ústředen celý tento systém vykonává kromě obvyklých přepojovacích funkcí ještě mnoho dalších činností vyplývajících z mobility účastníků (určování polohy, přidělování kanálů apod.).

***Operations support system (OSS)*** Podpora systému, jeho údržba, diagnostika, monitoring a reporting sítě.

***GPRS Core Network*** Volitelná součást, která umožňuje připojení do externích sítí a především k Internetu pomocí paketově orientovaných služeb (IP).

**Hlasové kodeky** GSM používá řadu hlasových kodeků, které dokáží hlasový rozsah 0,3 – 3,4 kHz přenést datovou rychlostí 6,5 nebo 13 kbit/s.

Tento systém je založený na lineárním prediktivním kódování (LPC). Kodeky jsou velmi efektivní při identifikaci důležitých částí zvuku pro srozumitelnost hovoru, což umožňuje snížit datovou rychlost i při velmi dobré srozumitelnosti.

**Subscriber Identity Module (SIM)** Jednou z klíčových vlastností GSM je Subscriber Identity Module, běžně známá jako SIM karta. SIM karta je odnímatelná čipová karta obsahující informace o uživateli, volitelné osobní identifikační číslo (PIN), neměnné číslo osobní odblokovávací klíč (PUK) a další údaje potřebné pro provoz:

- IMSI (International Mobile Subscriber Identity)
- Autentikační klíč (Ki)
- Šifrovací klíč (Kc)
- TMSI (Temporary Mobile Subscriber Identity)
- LAI (Location Area Identity)

Může obsahovat i uživatelský telefonní seznam. To umožňuje poskytovateli služeb (operátorovi) při připojení uživatele do sítě jeho ověření a identifikaci ihned po zapnutí telefonu. Alternativně uživatel může měnit operátory beze změny telefonu, jen změnou příslušné SIM.

**Bezpečnostní služby GSM** GSM má pouze omezenou úroveň zabezpečení služeb. Systém byl navržen tak, aby bylo provedeno ověření účastníka pomocí pre-shared klíče a odpovědi na výzvu. Komunikace mezi účastníkem a základnovou stanicí může být šifrována. Vývoj UMTS zavádí volitelné Universal Subscriber Identity Module (USIM), který používá delší autorizační klíč, což zvyšuje bezpečnost při vzájemné komunikaci a ověření sítě a uživatele.

Ověření totožnosti uživatele SIM karty probíhá pomocí zadávání číselných kódů PIN (PIN2) a PUK. Ověření totožnosti mobilní stanice může probíhat (v závislosti na operátorovi) pomocí IMEI (International Mobile Equipment Identity), což je číslo uložené v mobilní stanici a dále v registru EIR. Do registru VLR je zasláno toto číslo, které je pak ověřeno.

Po zapnutí požádá mobilní stanice o přístup do sítě. Zašle tedy své IMSI a toto je jediný okamžik, kdy je toto číslo používáno. Anonymita je poté zajištěna přidělením tzv. dočasné identifikace TMSI (Temporary Mobile Subscriber Identity), která je uložena

na SIM kartě a v registru Visitor Location Register (VLR). Tento registr obsahuje každá ústředna MSC, kde se ukládají pouze data účastníků, kteří se pohybují přechodně přes oblast dané MSC. Takový účastník se obvykle pohybuje přes dvě oblasti s různými MSC. Po opuštění oblasti první MSC budou data v VLR smazána a uložena v druhé MSC.

GSM používá několik kryptografických algoritmů pro bezpečnost. Nejčastěji se používají proudové šifry A5/1, A5/2 a A5/3 pro zajištění hlasového soukromí při přenosu hlasu po rádiu. A5/1 byl vyvinut jako první a je to silnější algoritmus, který se používá v Evropě a ve Spojených státech, šifra A5/2 byla vyvinuta později pro použití v jiných zemích a má slabší algoritmus. Obě šifry lze prolomit, A5/2 dokonce v reálném čase.

Pro datové přenosy používá GSM General Packet Radio Service (GPRS). Nejčastěji nasazené GPRS šifry byly veřejně rozděleny v roce 2011.

### 1.3 GSM shield s čipem SIM900

GSM modul použitý v této práci je postaven na čipu SIM900 od firmy SIMCOM. Jedná se o čtyřpásmový GSM/GPRS modul, který umí pracovat na frekvencích 850, 900, 1800 a 1900 MHz. Správná frekvence se vybere automaticky. Deska s čipem je navržena jako shield pro Arduino, takže je velmi jednoduše použitelná. Komunikace s arduinem probíhá pomocí AT příkazů přes sériovou linku. Pomocí zkratovacích propojek si můžeme nastavit, se kterými piny arduina bude modul komunikovat, na výběr máme buď hardwarový sériový port (D0, D1) nebo můžeme využít sw sériový port a to D3 pro Rx a D2, D8 nebo D10 pro Tx. V této práci je využita kombinace se softwarovým portem (D2, D3).

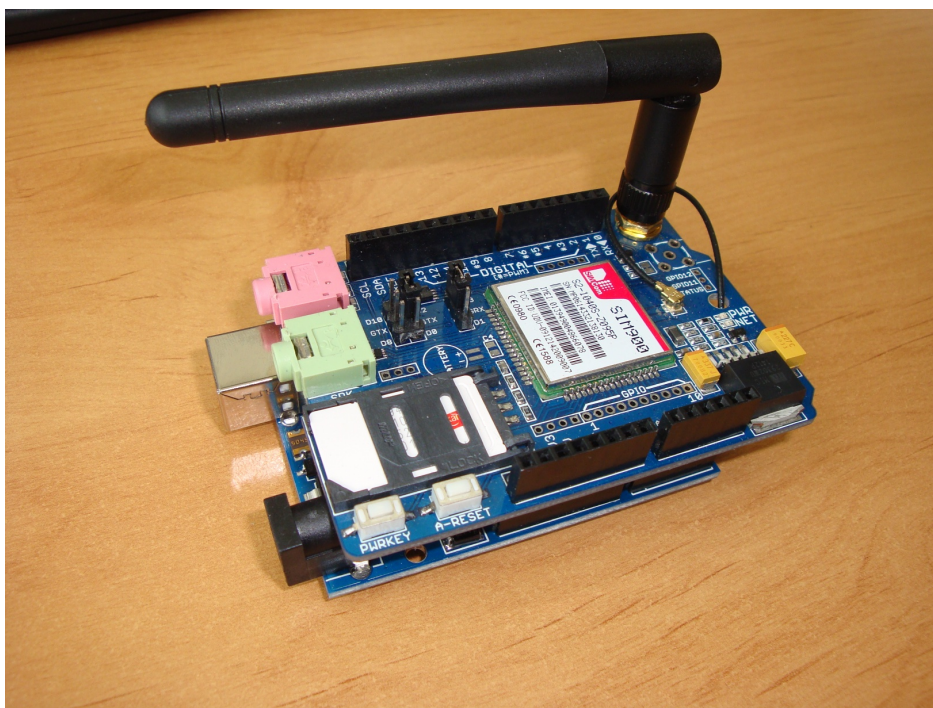
Na modulu můžeme nastavit a případně použít 2 PWM výstupy, 10 digitálních vstupně výstupních pinů. Chování jednotlivých pinů se nastavuje, jako všechny součásti čipu, pomocí AT příkazů.

Na desce jsou zapojeny dva audio konektory, pro sluchátka a mikrofon, které se používají pro spojený hovor a tlačítko pro spuštění GSM čipu. Spuštění se dá řešit i pomocí Arduina a to přes digitální pin 7. Dále pak tlačítko pro reset, anténu, slot pro SIM kartu a dvě LED diody. Pokud svítí červená LED, tak je připojeno napájení. Zelená LED pak podle blikání určuje 4 stavy.

Tab. 1.1 Indikace stavu GSM modulu [12]

64ms On/800ms Off	SIM900 není přihlášen do sítě
64ms On/3000ms Off	SIM900 je přihlášen do sítě
64ms On/300ms Off	GPRS komunikace
OFF	SIM900 je vypnuto





Obr. 1.2 GSM shield připojený na Arduino UNO

## 1.4 Arduino

Arduino je open-source projekt. Jeho historie sahá do roku 2005 do italského města Ivrea, kde se v Interaction Design Institute rozhodli vytvořit jednoduchý a hlavně i levný vývojový set pro studenty, který by nahradil tou dobou velmi rozšířenou a drahé desky Basic stamp. Povedlo se, a proto autoři nyní své desky nejenom prodávají, ale také zdarma sdílí všechna potřebná schémata a návody.

Arduino je tedy fyzická počítačová platforma založená na jednoduché mikrokontrolní desce a vývojovém prostředí pro psaní softwaru pro desku – Wiring. Tato programová část byla založena na programovacím jazyku s editorem Processing, hojně využívaný k výuce programování. Toto prostředí je tedy vhodné pro začátečníky, ale také zároveň flexibilní pro zkušené uživatele. [4]

Arduino se stále vyvíjí a rozhodně je to systém stále používaný. Vznikají nové desky a díky open-source také různé další neoficiální typy založené na těch původních.

Každá deska Arduino má procesor od firmy Atmel (mikroovladače ATMEGA8 a ATMEGA168) a další elektronické komponenty na jednotném grafickém zpracování a modrém podkladu. Když se něco nevyskytuje přímo na desce, stačí si vybrat z rozsáhlé nabídky tzv. shieldů a vybraný shield poté zasunout do zdířek k tomu určených (např. Ethernet Shield, WiFi shield aj.).

Software Arduino funguje v operačních systémech Windows, Macintosh OSX a Linux, což je velká výhoda oproti ostatním ovladačům, které většinou fungují jen ve Win-

dows. Díky open-source může být programátory rozšířen přes knihovny C++ nebo přímo přepnout z Arduina do programovacího jazyka AVR C, na kterém je založen. [3]

#### 1.4.1 Arduino UNO

Arduino UNO je dnes nejspíše nejčastěji používaným typem ze všech desek Arduino. Deska je založena na procesoru ATmega328p. Má 14 digitálních vstupně výstupních pinů, z čehož 6 podporuje PWM, 6 analogových vstupů, 16 MHz krystal. Na desce nalezneme napájecí konektor, USB a ICSP. Po stranách desky jsou konektory, do kterých lze zasunout shield, který může rozšířit modul o další periferie: Ethernet, bluetooth, LCD displej, GSM modul a další.

Tab. 1.2 Parametry Arduino UNO

Vstupní napětí (doporučené)	7 V – 12 V
Vstupní napětí (limit)	6 V – 20 V
Digitální IO	14 (6 podporuje PWM)
Analogové vstupy	6
Maximální proud IO pinů	40 mA
Flash paměť	32 kB
SRAM	2 kB
EEPROM	1 kB (100 000 přepisů)
Takt	16 MHz
Délka	68,6 mm
Šířka	53,4 mm
Váha	25 g

UNO má mnoho rozhraní pro komunikaci s počítačem, jinou deskou arduina nebo jiným mikrokontrolérem. Poskytuje UART TTL (5 V) dostupné na digitálních pinech 0 (RX) a 1 (TX). Tato sériová linka poskytuje pomocí dalšího mikroprocesoru ATmega16U2 komunikaci přes USB jako virtuální COM port. Aplikace Arduino obsahuje COM monitor, který lze použít pro jednoduchou textovou komunikaci počítače s deskou. Dvě LED kontrolky RX a TX svítí ve chvíli, kdy jsou data přenášena přes USB, ale ne pokud je komunikace pouze přes digitální piny 0 a 1.

Na kterýchkoliv jiných digitálních pinech lze UART komunikaci řešit softwarově, pomocí knihovny `SoftwareSerial`.

Mikroprocesor dále podporuje I2C (TWI) a SPI komunikaci. Aplikační balíček Arduino obsahuje `Wire` knihovnu, která zjednodušuje práci s I2C rozhraním. Pro SPI existuje knihovna se stejným názvem `SPI`. [5]

## 1.5 Vývojové prostředí

### 1.5.1 Arduino 1.0.5

Arduino vyvíjí své vlastní vývojové prostředí. To obsahuje kompilátor avr-gcc a knihovny, které lze pro procesory použít. Z tohoto prostředí lze psát kompletní programová řešení pro desky Arduino i nahrávat zkompilevané binární soubory přímo do mikroprocesorů.

Pro komunikaci s deskami Arduino se využívá COM portů. Pokud máme v programu využití sériové linky, lze tak komunikovat s deskou přes počítač s pomocí COM monitoru, který Arduino software obsahuje.

Uživatelská přívětivost psaní kódu v tomto prostředí však není nijak pokročilá a prakticky se neliší od psaní v obyčejném textovém editoru. Nevýhodou je velmi omezené zvýraznění syntaxe a nemožnost využít automatické navigace, které většina novějších vývojových prostředí poskytuje.

Používám verzi Arduino 1.0.5 i když už existuje nejnovější verze 1.6.4., která však většinou zlepšuje pouze podporu pro novější typy procesorů.

### 1.5.2 Visual Studio Community 2013 s Visual Micro

Visual Studio již dlouhou dobu podporuje C++. Umí navigovat na zdroj proměnných, metod i konstant. Navíc obsahuje funkci intellisense, která dokáže napovídat, jaké metody mohu napsat k příslušné třídě. Psaní je tedy mnohem jednodušší i rychlejší.

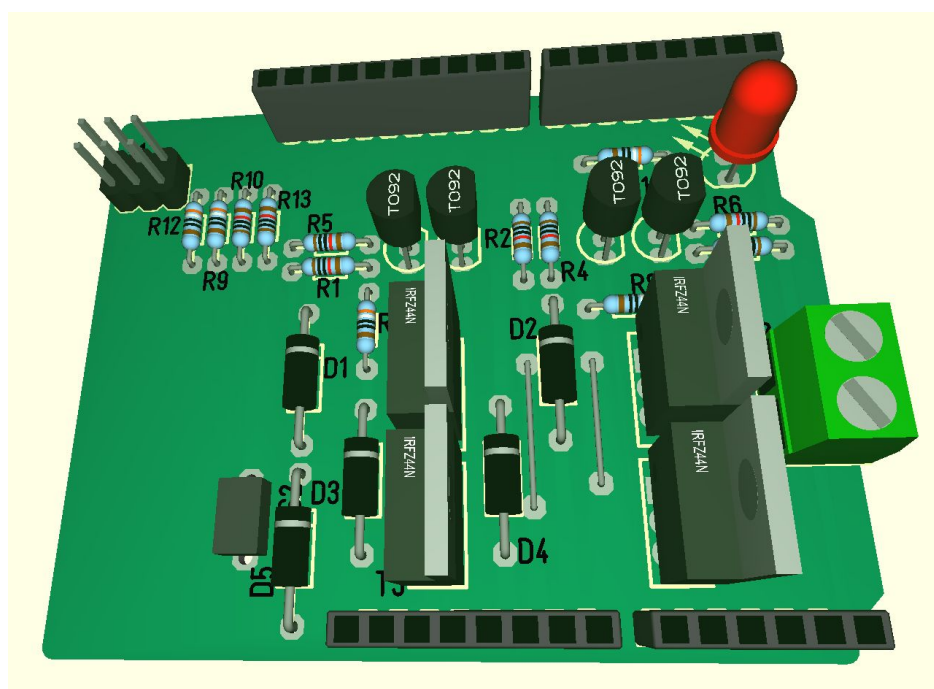
Nevýhodou je potřeba stále kompilovat kód v software Arduino. Tento problém však lze vyřešit doinstalováním Visual Micro, což je doplněk do Visual Studia. Tomuto doplňku zadáte umístění prostředí Arduino a můžete tak provádět i kompilaci a nahrávání binárního souboru do desky Arduino přímo z prostředí Visual Studia. Obsahuje též COM monitor, takže spouštění Arduino software již není vůbec potřeba.

V placené verzi Visual Micro je možné využít i debugger, který umí ladit běžící kód přímo na desce.

## 1.6 Target 3001!

Když jsem potřeboval navrhnout schéma a desku plošných spojů, hledal jsem software, ve kterém bych mohl rychle a snadno nakreslit jednoduché zapojení. Aplikací tohoto typu je spousta, ale nejvíc kladných ohlasů jsem našel na Eagle. Zkoušel jsem verzi Eagle 6.5 a hodně mě zklamalo uživatelské rozhraní, ve kterém se, dle mého mínění, nedá rozumně pracovat. Absence zkratk a nutnost pořád klikat do menu pro přepínání jednoduchých režimů mě donutila najít aplikaci jinou.

Proto jsem vybral Target 3001!, který je pro nekomerční použití a 250 pinů zdarma. Umožňuje kreslení schémat, navrhovat plošné spoje i automaticky, simulaci elektronické



Obr. 1.3 3D zobrazení navržené desky v Target 3001!

kých obvodů, generování kódu pro CNC stroje. Obsahuje rozsáhlou knihovnu součástek a spousta se jich dá stáhnout od tvůrců třetích stran. Další možností je importovat knihovnu určenou pro Eagle. Spousta součástek je k dispozici i s 3D náhledem a po rozmístění součástek na desku je možné si zvolit zobrazení v 3D režimu a tím se ujistit, že je pro součástky dostatek místa. [24]

## 2 EXISTUJÍCÍ ŘEŠENÍ

### 2.1 Typy bran

Existují dva typy bran – posuvné a křídlové. Křídlové mohou být ještě jednokřídlové nebo dvoukřídlové. Tyto typy se liší pouze konstrukcí a z toho plynoucími výhodami či nevýhodami.

#### 2.1.1 Posuvné brány

Konstrukce posuvných bran může být buď kolejnicová nebo samonosná.

**Kolejnicová** – je založena na kolejnici ležící na spodní straně brány, po které se brána na kolečkách posouvá. Na horní straně bočního sloupku pak bývá dvojice koleček proti sobě, mezi kterými prochází horní hrana brány. Tím se zajistí stabilní poloha brány. Většinou se brána staví podél plotu, či zdi. Kolejnice bývá typu U, která se zabetonuje do země a v ní se pohybují kolečka brány. Tento profil zajistí, že se brána udrží ve správném směru. Je ovšem náchylná k zapadávání nečistotami.

**Samonosná** – je držena na speciálních C profilech uchycených na bráně. Mimo průjezdnou část vozovky jsou pak umístěny dva vozíky, které jezdí v C profilu brány. Díky tomuto systému je průjezdná část bez jakéhokoliv konstrukčního prvku a není tak náchylná na nečistoty jako kolejnicový typ. [22]

Nejčastěji se posouvá jeden blok brány do strany, čímž se může schovat za plot nebo zeď. Existují i brány nebo spíš vrata od garáží, která jsou sekční. Složená jsou z několika plátů, které se navzájem spojují panty. Kolejnice nebo nosný profil pak může do určité míry zatačacet. Používá se v místech s omezeným prostorem garáže, kde se vrata zasouvají ke kolmé stěně. (obr. 2.1)

Automatický pohyb se u obou typů řeší motorem s ozubeným kolečkem, který je umístěn u krajního sloupku,

#### *Výhody*

- Nezabírá prakticky žádné místo v prostoru.
- Dá se otevřít jenom částečně, například pro průchod chodců.
- Sekční typ nepotřebuje mít podél brány rovný plot (zeď).



Obr. 2.1 Sekční posuvná vrata LOMAX [17]

### *Nevýhody*

- Musí se udržovat kolejnice čisté od většího množství nečistot, například sněh, led, šterk nebo listí.
- Potřebuje tolik místa podél plotu (zdi), jak je dlouhá brána.

### **2.1.2 Křídlové brány**

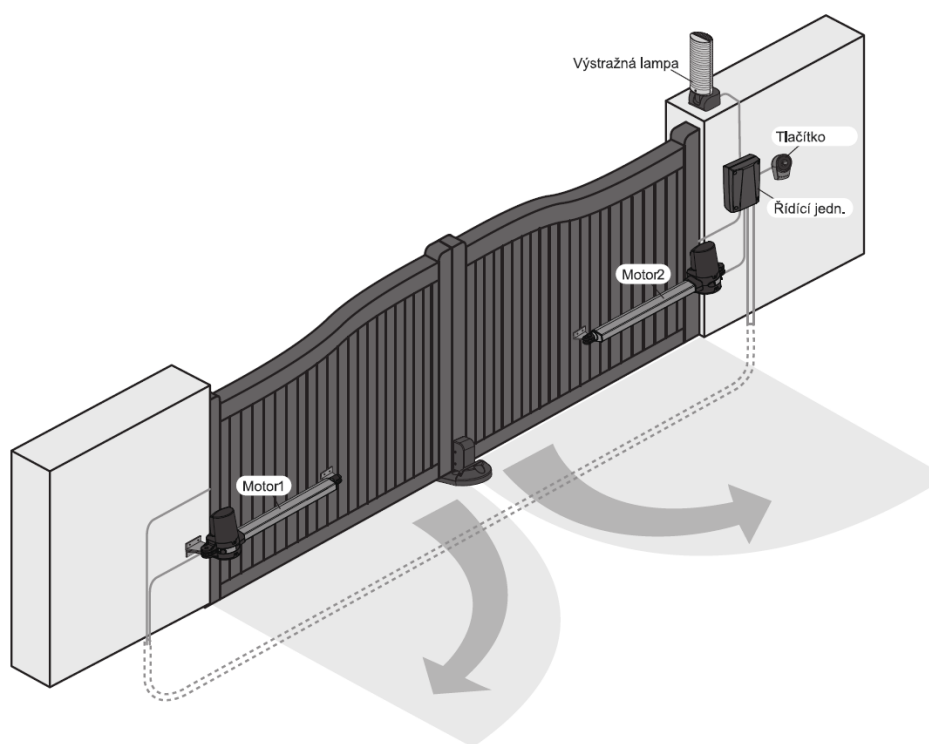
Křídlové brány se také dělí na dvě skupiny, jednokřídlové a dvoukřídlové. Jejich princip a použití je prakticky totožný. Jednokřídlová brána má na jednom bočním sloupku panty, na kterých je brána zavěšena a otevírá se dovnitř nebo ven. Šířka brány je dána velikostí průjezdního prostoru. Naopak dvoukřídlová má 2 brány o šířce poloviny průjezdního místa. Panty jsou na sloupcích z obou stran a obě poloviny se pohybují. Nejčastěji stejným směrem, buď dovnitř nebo ven.

Automatizace otevírání se nejčastěji řeší výsuvným ramenem na kloubech. Rameno se připevňuje na sloupek a křídlo brány. Toto rameno se prodlužuje nebo zkracuje a podle toho se brána zavírá nebo otevírá.

Další možností je podzemní pohon. Ten je umístěn v zemi hned u sloupku tak, aby osa motoru byla pod panty. Motor pak pohybuje spodní částí křídla brány.

### *Výhody*

- Řeší problém s nedostatkem místa podél plotu (zdi).
- Odolná vůči nečistotám. Problémem tak bývá pouze větší množství sněhu.



Obr. 2.2 Pohon křídlové brány [21]

### *Nevýhody*

- Při otvírání zaberou hodně prostoru. Mohou být překážkou i otevřené
- Při dvoukřídlovém řešení musí být natažena k druhému sloupku i silová kabeláž pro motory.

## 2.2 Fotobuňky

Ke každé bráně se dnes přidávají fotobuňky, které hlídají prostor brány mezi sloupky. V případě, že se v tomto prostoru někdo nebo něco vyskytuje, čidlo to pozná a řídicí jednotka nedovolí bránu zavřít. Případně pokud tato situace nastane při zavírání, tak se brána zastaví a některé se znovu úplně otevrou.

Dělí se na jednocestné nebo dvoucestné, přičemž nejčastější jsou jednocestné, u které se na jedné straně nachází vysílací buňka, která vysílá paprsek infračerveného světla a na druhé straně vjezdu je přijímací buňka, která když přijímá světlo, tak je vjezd volný a pokud nepřijímá, pak něco stojí v cestě.

Dvoucestná fotobuňka funguje podobně, ale vysílací i přijímací buňka se vykytuje na stejné straně a na druhé straně je pouze odrazka, která signál vrátí zpět. [9]

## 2.3 Pohonná jednotka

Pohonné jednotky se umísťují přímo na konstrukci brány. Motory jsou na 230 V nebo 24 V. Řídící jednotka udává kdy a kterým směrem se má motor spustit. Pokud je pohonná jednotka v nečinnosti, pak se motor uzamkne, aby nešlo bránu samovolně otevřít. Na těchto pohonných jednotkách bývá systém odblokování a otevření brány ručně. Toto je důležité v případě výpadku proudu. Větší křídlové brány se doplňují ještě o elektromechanické zámky, aby nemohlo dojít k poškození uzamykacího systému v pohonné jednotce. [27]

## 2.4 Maják

Slouží k informaci, že je brána v pohybu nebo se k pohybu chystá. Chodci na chodníku se tak mohou připravit na blížící se auto. Řidič také vidí, že se ovládání brány sepnulo.

## 2.5 Dálkové ovládání [20]

Dálkové ovladače fungující na frekvencích 433 MHz nebo 868 Mhz. Vysílají kód, podle kterého přijímač rozezná správný vysílač a provede nastavenou akci. Existují 3 typy ověření.

**Pevný kód** – vysílač vysílá pokaždé stejný kód. Přijímač je nastavený tak, že přijímá stejný kód jako je s ním spárovaný vysílač. Bohužel jsem nikde nenašel, jak dlouhý tento kód je. Ale kombinací by mělo být tisíce, takže není pravděpodobné, že byste po nainstalování takového zařízení omylem otevřely vrata u souseda. Nevýhoda je, že tento kód může někdo odposlouchávat a poté jej kdykoliv odeslat ze stejného zařízení. Tento problém řeší plovoucí kód.

**Plovoucí kód** – Ovladač při každém stisku tlačítka odešle pokaždé jiný kód a přijímací jednotka je nastavena tak, aby nepřijala dvakrát stejný kód. Kód bývá 52 bitový, tzn. kombinací je zhruba  $4,5 \cdot 10^{15}$ .

**Plovoucí kód & Antiscan** – Útočník může kromě odposlouchávání kódu využít ještě náhodného vysílání všech možných kódů a doufat, že se jednou trefí do toho správného. Přijímač je v tomto režimu nastaven tak, že pokud v krátké době přijme více různých kódů, pak se zablokuje a další kódy přijme až za stanovenou dobu nebo po ručním odblokování.

### 2.5.1 GSM

Další možností je využití GSM modulů. Obrovskou výhodou je, že lze nastavit, na která telefonní čísla bude reagovat. Existují typy, jako například PGM-03C Eco, které pracují



pouze na jednoduchém spínání na základě prozvonění tohoto modulu a sepnutí jednoho relé. Nevyužívají tak vůbec potenciálu, který GSM nabízí.

Pak existují systémy mnohem propracovanější, například "GSM Klíč" od firmy Sectron. Existuje v několika verzích, lišících se v množství výstupů a možnosti nastavení. V nejvyšší verzi umí spínat až 4 výstupy a jejich spouštění je ovládáno podle délky prozvonění. Všechny lze nastavovat pomocí mobilu přes SMS. A také umí odesílat SMS o stavu na vstupech. [11]

## II. PROJEKTOVÁ ČÁST

### 3 MODEL BRÁNY

#### 3.1 Mechanika

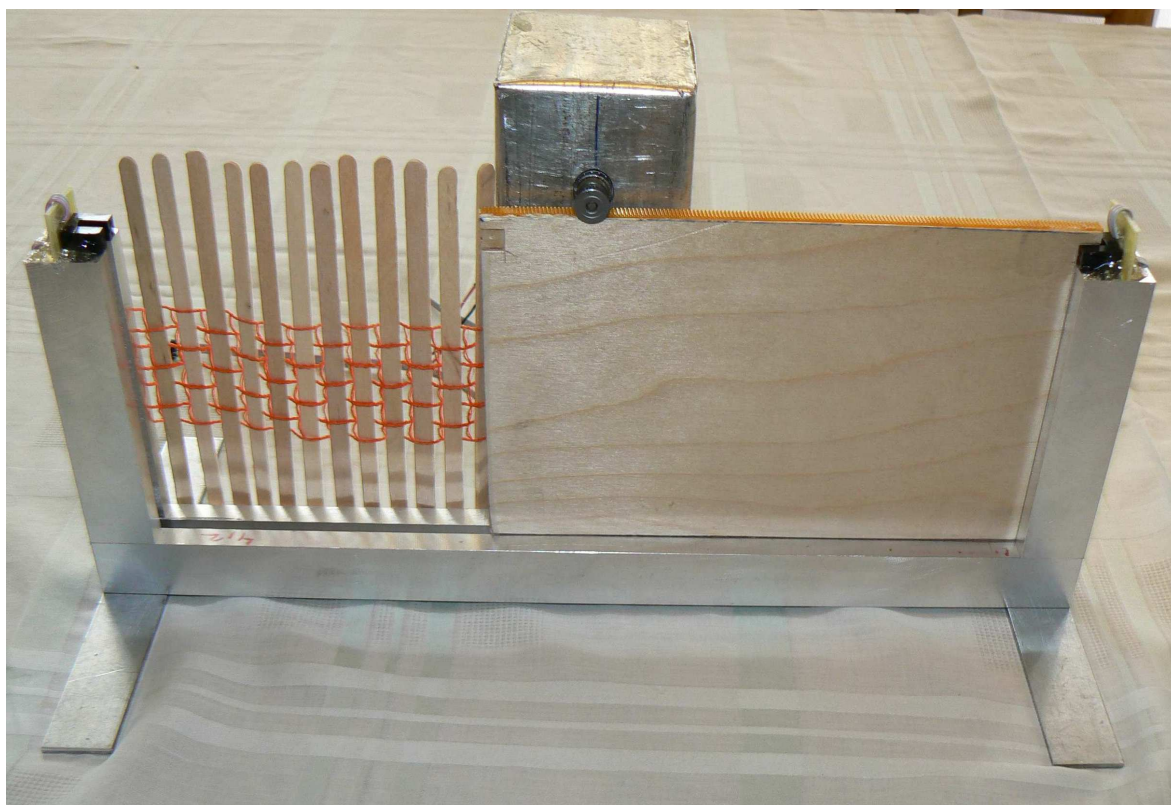
Skelet je složen ze čtyř částí:

- Základny z hliníkového profilu 20 x 20 x 300 mm, v kterém je vyfrézována drážka 4,2 mm široká a 9 mm hluboká.
- Koncové sloupky jsou vyrobeny stejně jako základna z hliníkového profilu a mají výšku 120 mm. Každý sloupek je přišroubován dvojicí šroubů M4 x 25 k základně přes stabilizační plošky z hliníku tloušťky 2 mm, které udržují bránu ve stabilní poloze. Obě mají velikost 20 x 150.
- Sloupek motorové jednotky je spájen z pocínovaného plechu, plní funkci středového sloupku a držáku motoru ve správné poloze vůči bráně. K základně je přišroubován dvojicí šroubů M4 x 8.
- Pro posun brány jsem využil motoru s ozubeným kolečkem a řemínkem, které jsem použil ze staré tiskárny. Na přívodním kabelu napájení je feritové jádro, které snižuje rušení vznikající při práci motoru při řízení signálem PWM. Brána je vyrobena ze 4 mm překližky, na kterou je z vrchní strany přilepený řemínek. Ve spodní části je překližka zeslabena, aby se snížil třecí odpor na bocích brány v drážce. Obdobně jsou zkoseny hrany brány pro nájezd do koncových sloupků. V místě, kde se brána zasouvá do optických čidel je upravena na tloušťku 3 mm, protože optická čidla mají štěrbinu pouze 3,1 mm širokou.

Na obou sloupcích jsou přilepena optická čidla, která po zasunutí brány dávají signál o dojezdu brány do zavřené nebo otevřené polohy. Po dojezdu brány do některé z krajních poloh je motor řídící jednotkou vypnut. Brána zůstává v této poloze do příchodu signálu pro otevření `Open` (pokud byla zavřena) nebo pro uzavření `Close` (pokud byla otevřena). Kabely optických čidel jsou přilepeny na konstrukci a přivedeny ke středovému sloupku.

#### 3.2 Elektronika

Jako základ jsem využil Arduino Uno s procesorem Atmel ATmega328P. K tomu jsem pořídil GSM modul s čipem SIM900. Jedná se o takzvaný shield. Shield je rozšiřující deska, která se pouze zasune do konektorů v arduinu a tím je s ním bezpečně spojen a sdílí s ním veškerou svoji konektivitu i napájecí část. Hlavní výhodou je, že jsou obě desky spojeny do jednoho celku. Navíc je navržený tak, že do něj lze vložit další shield a tím ještě více rozšířit jeho možnosti.



Obr. 3.1 Model brány

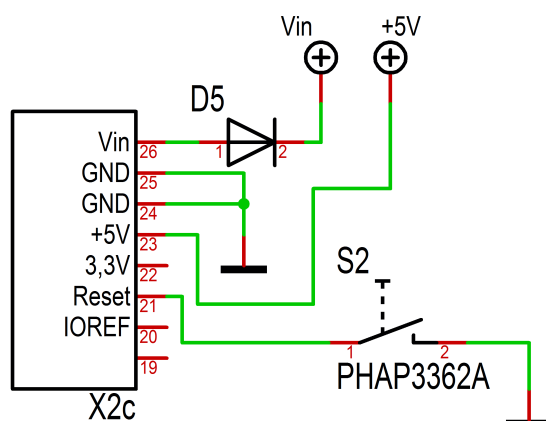
Proto jsem ovládání brány navrhl také jako shield. Není už dále rozšiřitelný, protože obsahuje transistory, které jsou příliš vysoké a není tedy možnost zařízení rozšířit například o displej.

Shield obsahuje dvě části obvodu. První část, která ovládá motor a druhá, která zpracovává signál z čidel na koncích brány.

### 3.2.1 Napájení

Arduino UNO je možné napájet od 6 V do 20 V (doporučeno 7 V – 12 V) [1]. Mnou použitý motor je na 12 V, a proto je potřeba obvod napájet tímto napětím. Za konektorem napájení arduina je jedna ochranná dioda [2]. Na vstup obvodu jsem vložil ještě jednu další diodu, abych ochránil arduino od případných zpětných pulzů, které mohou vzniknout při přechodových jevech při provozu motoru. (obr. 3.2) Na obou diodách zůstane přechodové napětí a motory tedy nejsou napájeny plným napětím 12V, ale podle měření pouze 10,5V. Jelikož rychlost motoru stejně omezují pomocí PWM, tak toto nižší napětí postačuje.

Na desku s modulem jsem přidal i tlačítko pro reset arduina. Pro restart mikrokontroléru stačí pouze na pin "Reset" přivést zem (logickou 0).



Obr. 3.2 Schéma napájení modulu z arduina

### 3.2.2 Ovládání motoru

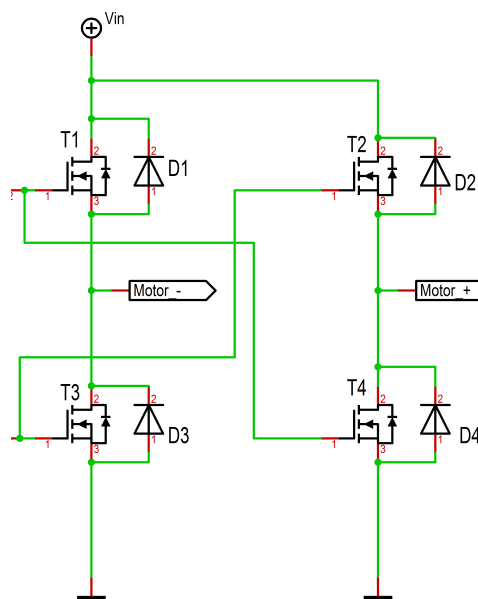
Pro ovládání motoru jsem chtěl prvně využít jenom relé, které bych ovládal a jeho kontakty by byly připojeny k motoru. Jenže použitý motor na bráně byl při plném otevření příliš rychlý a bránu spíše vystřeloval než otevíral. Proto jsem se nakonec rozhodl využít potenciálu procesoru a použil jsem pulsní šířkovou modulaci (PWM), a tím jsem omezil celkový výkon, který jde do motoru. V tomto případě nešlo použít relé, kvůli jeho pomalému spínání.

Využil jsem tedy výkonové tranzistory typu MOSFET, které zvládají rychlé spínání a mají velmi nízký odpor při otevřeném stavu, takže zvládají velmi vysoké proudy. Mnou vybrané IRFZ44 mají v otevřeném stavu pouze  $24m\Omega$ , otevírací čas je maximálně  $40ns$  a zavírací čas maximálně  $140ns$ , z čehož plyne, že zvládá pracovat až v řádu jednotek Mhz.

Z tranzistorů jsem vytvořil H-můstek, abych mohl spouštět motor v obou směrech. Na schématu (obr. 3.3) vidíme, že pokud otevřeme zároveň tranzistor T1 i T4, tak nám bude proud protékat právě přes tranzistor T1, motor a T4. Naopak pokud otevřeme T2 a T3, pak bude proud protékat v opačném směru T2, motorem a T3. Velmi důležité v tomto zapojení je, aby nikdy nebyly otevřeny dva tranzistory, které jsou jakoby nad sebou, tzn. T1 a T3, nebo T2 a T4. Pak by proud procházel pouze přes tyto tranzistory a ne přes motor, takže by došlo prakticky ke zkratu a mohli bychom tak tranzistory zničit. Při změně stavu motoru dávám 100 ms pauzu pro ustálení přechodových jevů v obvodech motoru.

### 3.2.3 Senzory

K vnitřnímu ovládání brány je potřeba znát stav, ve kterém se brána nachází. Tento stav se mění v závislosti na požadavku uživatele, funkci motoru a stavu čidel na koncích



Obr. 3.3 Schéma H-můstku

brány. Podrobné vysvětlení vlivu bude vysvětleno v kapitole 4.3.

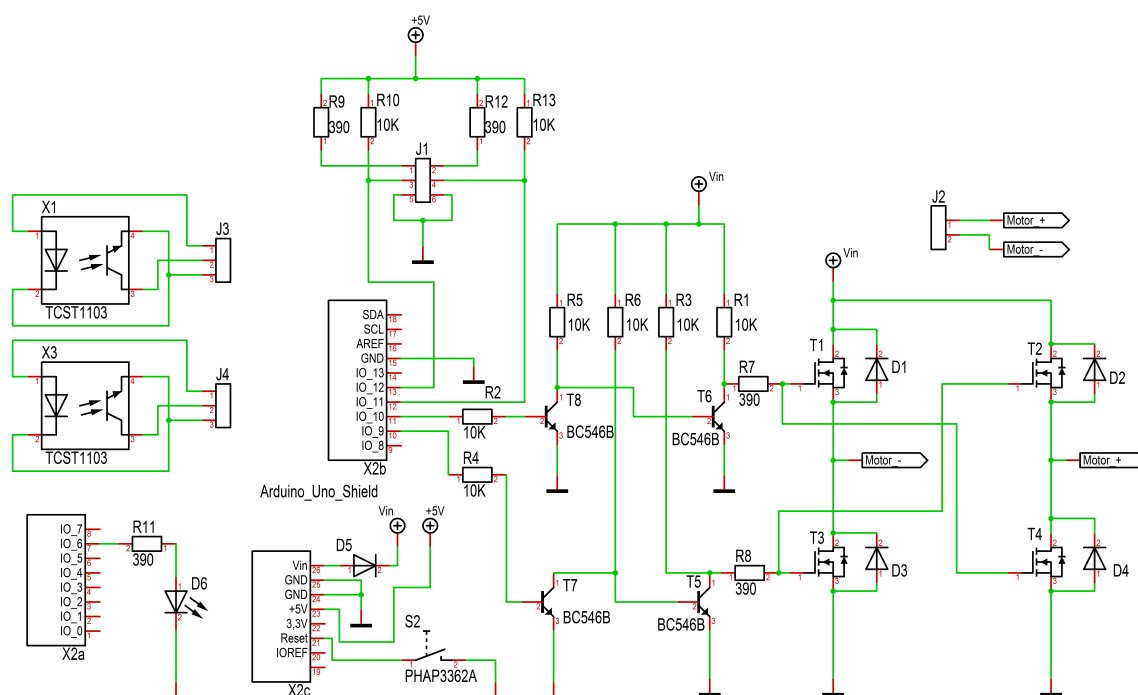
V této kapitole budu popisovat aplikaci jednotlivých senzorů na konci brány. V zásadě je jedno, jaký typ čidla použijeme. Procesor arduina je naprogramován tak, že očekává na digitálních vstupech 11 (otevřeno) a 12 (zavřeno) logickou hodnotu.

1 sepnuto – brána se vyskytuje v otevřeném/zavřeném stavu.

0 rozepnuto – brána není v otevřeném/zavřeném stavu

Vzhledem k použití posuvné brány jsem využil optozávory TCST1103 [25]. Součástka je složena z jedné infračervené LED diody a fototranzistoru, které jsou umístěny proti sobě v pouzdře ve tvaru U. Jakmile diodou prochází proud, tak se rozsvítí a fototranzistor se otevře. Pokud do mezery v pouzdře umístíme cokoliv neprůhledného, tak se nám fototranzistor zase zavře. Tohoto jednoduchého principu využívám tak, že je optozávora umístěna na konstrukci tak, aby se do ní brána zasunula a tím zavřela fototranzistor.

Ze schématu (obr. 3.5) je zřejmé, že diodou protéká stálý proud nezávislý na stavu brány. Při nezastíněném senzoru je fototranzistor v otevřeném stavu a na výstupu SIG-OUT je logická 0. Pokud ovšem fototranzistor zastíníme zasunutím brány, tak se zavře, neprochází jím tedy žádný proud a na rezistoru R10 je celé napětí 5 V, které se nám projeví na výstupu SIG-OUT jako logická 1. Velikost odporu R9 jsem nastavil tak, aby proud byl co nejnižší, ale zároveň aby při nezakrytém čidle byl tranzistor otevřený (SIG-OUT = logická 0).

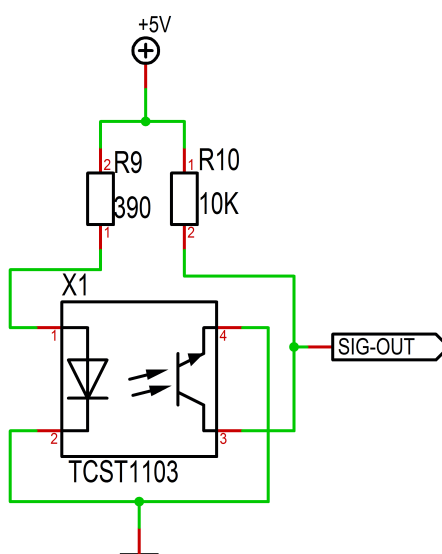


Obr. 3.4 Schéma zapojení shieldu pro Arduino Uno

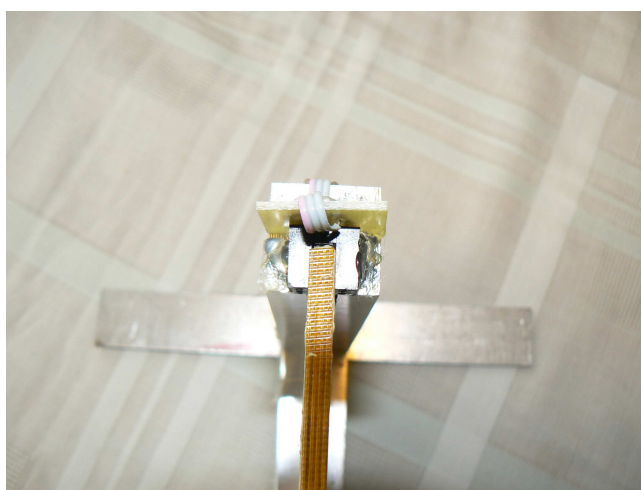
Jako další alternativu jsem zkoušel využití jazýčkového relé, které by bylo na konstrukci rámu a na bráně by byl magnet, který by při přiblížení toto relé sepnul. V tak malém modelu jaký jsem sestavil, byla tato varianta těžko použitelná, protože obyčejné magnety jsou příliš velké a na bránu by se těžko přidělávali. Testoval jsem neodymové magnety které jsou menší, ale zase jsou příliš silné a relé spínali už na vzdálenost jednotek centimetrů. Na skutečné veliké bráně by použití bylo jednodušší kvůli většímu prostoru pro nastavení správné vzdálenosti k sepnutí relé.

Jako další typ čidla můžeme použít klasické tlačítko, které by brána při dojezdu sepnula. Jeden kontakt by byl připojen na 5 V. Druhý kontakt pak na digitální vstup Arduino a přes rezistor  $10\text{ k}\Omega$  na zem. Při nestlačeném tlačítku tento rezistor zajistí na vstupu Arduino logickou 0. Po stisknutí tlačítka se na vstup dostane logická 1.

Převážně na závorách, kde není sloupek na obou stranách, se využívá ultrazvuková čidla, která měří vzdálenost od objektů. Sensor vyšle vysokofrekvenční tón a čeká, až se vrátí zpět. Zpět se vrátí díky odrazu od překážky. Změří se čas od vyslání po příjem a spočítá se vzdálenost, kterou zvuk za změřenou dobu urazil. Vzdálenost ještě vydělíme 2, protože spočítaná délka je pro cestu tónu k překážce i zpět. Rychlost šíření vzduchu je závislá na teplotě a vlastnostech látky, ve kterých se šíří. V tomto případě však stačí pouze informativní zjištění, zda se v oblasti vyskytuje nějaká překážka a k tomu postačí počítat s rychlostí šíření ve vzduchu  $340\text{ ms}^{-1}$ .



Obr. 3.5 Schéma aplikace senzoru TCST1103



Obr. 3.6 Umístění čidla na bráně

### 3.3 Deska plošných spojů

Deska plošných spojů je navržena v aplikaci Target 3001! (viz. 1.6). Rozmístění součástek i navržení plošného spoje je provedeno ručně, bez použití automatiky. Použita je jednostranná deska plošného spoje. Princip technologie je založen na přenosu toneru z laserové tiskárny přímo na kuprexitovou destičku a to z důvodu jednoduchosti a dostatečné přesnosti.

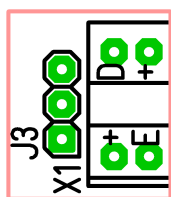
Na speciální přenosový papír se pomocí laserové tiskárny vytiskne zrcadlově převrácený vzor spojů ve velikosti 1:1. Poté se papír přiloží na důkladně očištěnou a odmaštěnou kuprexitovou desku, natištěnou stranou k desce a pečlivě se papír přezehlí žehličkou. Po vychladnutí se destička i s papírem ponoří do vody, kde se papír postupně



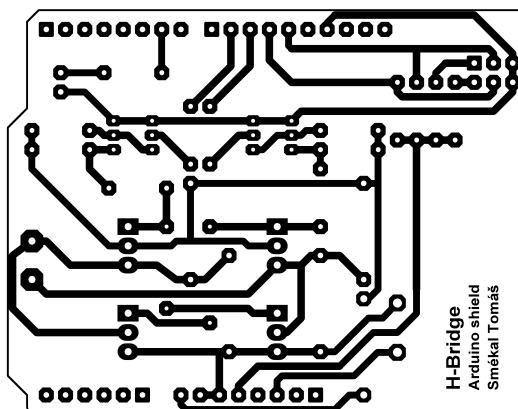
uvolní. Poté můžeme již vyleptat destičku v chloridu železitém, kde se odstraní všechna měď mimo natištěnou oblast.



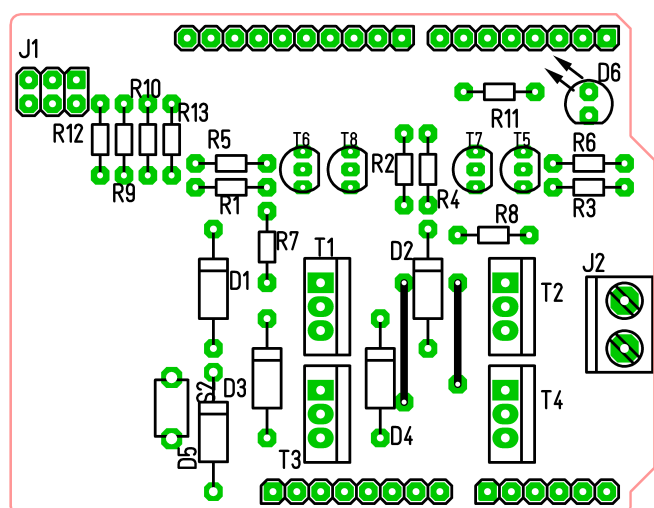
Obr. 3.7 Deska plošných spojů senzoru 1:1



Obr. 3.8 Osazení  
senzoru



Obr. 3.9 Deska plošných spojů 1:1



Obr. 3.10 Osazení

## 4 SOFTWARE

### 4.1 Logické rozvržení

#### 4.1.1 ICommunication

Vzhledem k ceně SMS a četnosti testování a nutnosti komunikace jsem vytvořil třídu `ICommunication`, která slouží jako interface pro třídy obsluhující konkrétní komunikační rozhraní. V tomto případě GSM `GSMCommunication` a pro ladění jsem využíval sériovou linku `SerialCommunication`. Nebyl by však problém dopsat ovládací třídy i pro jiné způsoby komunikace, přes bluetooth, ethernet a podobně.

Třída obsahuje prakticky pouze tři metody a to inicializační metodu `void Init()`, pro příjem zprávy `rcvMessage` a jednu přetíženou pro odeslání zprávy `sendMessage`

Definice metody pro příjem `byte rcvMessage(char** message, char** phoneNumber)`. Tato metoda vrací hodnotu 1, pokud vrací zprávu a 0, pokud není žádná zpráva k dispozici. Výstupní parametry `message`, ve které se vrací načtená zpráva a `phoneNumber`, kde je uložen identifikátor na odesílatele.

Metoda pro odesílání je přetížená. Jednou ve tvaru `void sendMessage(char* message)`, kde jako vstupní parametr slouží pouze zpráva `message`, kterou chceme odeslat a příjemce se zvolí výchozí nastavený. Po druhé `void sendMessage(char* message, char* phoneNumber)`, je k odesílané zprávě přidán ještě identifikátor `phoneNumber` na příjemce, kterému budeme zprávu odesílat.

#### 4.1.2 GSMCommunication

Je potomkem třídy `ICommunication`. Využívá knihovny `GSM_GPRS_GPS_Shield` (4.1.7), ze které využívá třídu `MSGSM`.

V inicializační metodě `Init()` se zavolá `gsm.begin()`, což zajistí, že se GSM modul spustí a začne se přihlašovat do sítě. Metoda trvá, v důsledku přihlašování do sítě a navazování a ověřování komunikace, několik sekund. Po přihlášení modulu do sítě můžeme ověřit, že stavová LED dioda na GSM modulu blikne jednou za 3 sekundy.

Metoda pro příjem je implementována tak, že zjistí, zda je na SIM kartě uložena nová nepřečtená zpráva `sms.IsSMSPresent`. Pokud není, tak se pouze nastaví 0 jako návratová hodnota a ukončí se. Pokud zpráva existuje, tak se zavolá metoda `sms.GetAuthorizedSMS`, která přečte ze SIM karty zprávu a uloží do proměnné `message`, zároveň do proměnné `phoneNumber` uloží číslo odesílatele. Výhoda této metody oproti `sms.GetSMS` je, že umí porovnat číslo odesílatele s čísly uloženými na SIM a tím vrátit stav, jestli zpráva pochází od někoho ze seznamu nebo ne. Tohoto jsem využil k blokování zpráv od cizích čísel a zabránil tak otevření brány cizí osobou. Zpráva se následně smaže ze SIM. Při neautorizované zprávě se vrátí 0, čímž se v dalším zpracování zpráva

ignoruje.

Odesílací metoda `sendMessage` opět využívá knihovny `MSGSM` konkrétně metody `sms.SendSMS`. Tuto metodu můžeme volat buď s číslem, na které budeme zprávu odesílat nebo s pozicí kontaktu uloženého na SIM. Pro odeslání zprávy jako odpovědi na zprávu od uživatele tak použijí číslo, ze kterého zpráva přišla. Naopak pro poslání zprávy na základě nějaké události vycházející ze stavu brány se posílá zpráva prvnímu kontaktu uloženého na SIM.

#### 4.1.3 SerialCommunication

Třída `SerialCommunication` je taktéž potomkem `ICommunication`. Použita byla pouze z důvodu testování. V konečném binárním kódu není použita. Pro komunikaci využívá sériové linky.

Třída samozřejmě implementuje metodu pro odesílání s parametrem `phoneNumber`, jelikož sériová linka není adresovatelná, tak se parametr ignoruje.

#### 4.1.4 Gate

Třída `Gate` ovládá mechanickou část brány a vyhodnocuje stavy senzorů. Na základě hodnot na senzorech si udržuje hodnotu stavu, ve kterém se zrovna brána nachází (4.3).

Pro výchozí nastavení je použita metoda `Init()`. V této metodě jsou nastaveny piny 10 a 11 jako vstupní s pull up rezistorem. Slouží pro připojení senzorů určujících konečné polohy brány. Pull up rezistor zajistí, aby při nepřipojené vstupní napěťové úrovni procesor vždy přečetl logickou 1. Pro ovládání motorů jsem použil PWM. Využil jsem k tomu 16 bitový timer 1, který má pro výstup PWM nastaveny piny 9 a 10, které jsou tak nastaveny jako výstupní.

Pro nastavení časovače existují různé knihovny, ve kterých pak stačí zavolat metodu s frekvencí, jakou má časovač pracovat. Metoda pak zařídí nastavení všech registrů. Výsledný binární kód by ale zbytečně narostl. Pokud bychom chtěli často v průběhu programu měnit frekvenci, se kterou bude časovač pracovat, použití knihovny může být výhodné. V našem případě se však frekvence měnit po celou dobu běhu nebude. Proto jsem nastavení registrů provedl ručně.

V první řadě je potřeba určit dělič, který určuje, kolik taktů vstupního krystalu je zapotřebí k přičtení 1 do čítače časovače. Dále pak režim, v jakém bude časovač pracovat. Máme nastaveno Fast PWM, které je určeno pro ovládání PWM. Režimů je více, všechny jsou popsány v datasheetu k procesoru ATmega328P [6].

Frekvence je vypočtena podle vzorce:

$$f = \frac{f_{clk\_IO}}{N \cdot (1 + TOP)}$$

Kde:

- $f_{clk\_IO}$  je frekvence vstupního krystalu – 16 MHz
- $N$  je nastavení děliče, může nabývat hodnot (1, 8, 64, 256, 1024)
- $TOP$  je hodnota čítače, při které se čítač vynuluje a počítá od nuly

Postupně jsem testoval různé frekvence pro PWM. V první řadě jsem chtěl dostat frekvenci nad 20 kHz, protože při nižších frekvencích motor nepříjemně píská. Motor má však nižší točivý moment. Nešlo tak dobře regulovat rychlost, protože se motor točil buď příliš rychle nebo se netočil vůbec. Proto jsem postupně frekvenci snižoval. Výsledná konfigurace je bez děliče (dělič nastaven na 1) a čítač se nuluje při své maximální hodnotě tj.  $2^{16} - 1 = 65535$

Dosazením do předchozího vzorce dostáváme výslednou frekvenci:

$$f = \frac{16000000}{1 \cdot (1 + 65535)} = 244,14Hz$$

Motor sice píská, ale tón je relativně hluboký, takže není nepříjemný a naopak akusticky signalizuje spuštění motoru. Při takto zvolené frekvenci je točivý moment dostatečný. Samotná rychlost se určí střídou PWM. Ta se určí nastavením registru pro každý pin zvlášť. OCR1A pro pin 9 a OCR1B pro pin 10. [6] Pin je nastaven na 1 po dobu hodnoty čítače od 0 po OCR1x, kdy se vynuluje. Na 1 se nastaví zase při přetečení čítače z TOP na 0.

Hodnota OCR1x se nastavuje při inicializaci třídy a načítá se z vnitřní paměti EEPROM, kde je uložena na adresách 0 a 1, protože do této paměti se ukládají čísla po jednotlivých bytech a potřebujeme uložit 2. Jako rozumná hodnota pro použitý model brány je kolem 17000. Tato hodnota se dá měnit zavoláním metody pro zvýšení `IncreaseDuty(unsigned int offset)` a snížení `DecreaseDuty(unsigned int offset)`, kde za proměnnou `offset` dosadíme, o kolik se má střída změnit. Každá tato změna se automaticky zapíše do EEPROM paměti, takže i po restartu zůstane poslední nastavená rychlost. K tomu slouží metoda `savePwmDuty(unsigned int duty)`.

Spouštění motoru a jeho zastavení se řeší, jak již bylo zmíněno, přes PWM. Tento stav se musí pro každý pin nastavit zvlášť. Děje se tak v registru TCCR1A bit `COM1A1` pro pin 9 nebo `COM1B1` pro pin 10.

V metodách pro spuštění motoru `Open()` a `Close()` se aktivuje a to pouze pokud není již ve stavu `OPENED`, respektive `CLOSED`. K tomu se pak rozsvítí LED, připojena

na pinu 6. V metodě `StopMotor()` se deaktivuje PWM pro oba piny a zhasne LED.

#### 4.1.5 EepromAddresses.h

V hlavičkovém souboru `EepromAddresses.h` jsou zapsány adresy do EEPROM paměti procesoru. V poznámce je zapsána délka v bytech, o kolik musí být další adresa posunuta, aby nepřepisovala hodnotu předchozího záznamu.

#### 4.1.6 Hlavní smyčka kódu

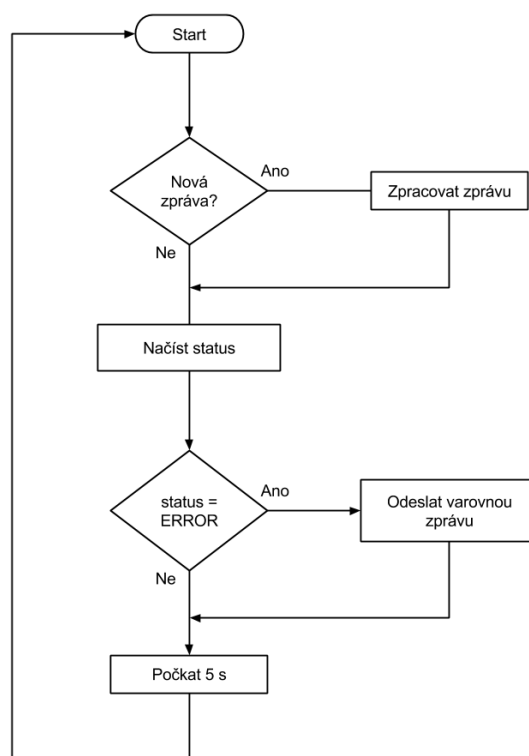
Začátek aplikace psané pro arduino nezačínají funkcí `main`, ale jsou zde dvě základní funkce `setup()` a `loop()`. Velkou výhodou Arduina je, že již v základu obsahuje inicializační funkce pro různé typy mikroprocesorů. Po spuštění překladač se podle zvoleného typu procesoru vyplní různé konstanty, které se používají pro nastavení hardwarových částí. Patří mezi ně například `F_CPU`, ve které je uložen takt procesoru. Pro Arduino UNO je to 16 MHz. Nebo se určuje základní nastavení časovačů, pro Arduino UNO to jsou 3 časovače, ale třeba pro Arduino MEGA je těchto časovačů 6. Těchto různých konstant jsou pro různé procesory stovky.

Soubor `GateGSM.ino` je počáteční soubor se základními funkcemi `setup()` a `loop()`. Slouží jako hlavní řídicí část kódu, která zpracovává a obsluhuje jednotlivé třídy, jež jsou popsány v předchozích kapitolách.

Ve funkci `setup()` probíhá inicializace všech hardwarových i softwarových součástí. V první řadě se vytvoří instance komunikační třídy `GSMCommunication`. V konstruktoru probíhající inicializace může trvat relativně dlouho, protože GSM modul se přihlašuje do mobilní sítě. Poté se nastaví timer 2 na krekvenci 60 Hz a povolí se přerušení při přetečení. Následně se zavolá inicializace brány a její zavření. Tedy po každém případném restartu bude brána zavřená nebo se začne uzavírat, aby se do tohoto stavu dostala.

V hlavní aplikační smyčce `loop()` se kontroluje existence nových zpráv z komunikačního objektu `ICommunication()`. V případě, že zavolaná metoda `rcvMessage` vrátí `true`, pak je jistota, že odesílatel je již ověřen, protože zprávy od neoprávněného odesílatele se automaticky smažou.

Obdržená zpráva se předá funkci `control`, která zpracuje obdržený požadavek. Následně se načte aktuální stav brány, kdy se přečte status metodou `gate.getStatus()`. Neopakuje se volání o kontrolu stavu z čidel brány. Pokud je tento status `ERROR`, pak se pošle uživateli zpráva o chybném stavu brány, který nebyl vyžádán. Tato zpráva je odeslána pouze jednou, takže pokud i při dalším průchodu cyklem bude pořád status `ERROR`, zpráva se znovu neposílá. Pokud ale byl mezitím status správný, zpráva se znovu poslat může.



Obr. 4.1 Vývojový diagram aplikační smyčky

V případě debug módu se nyní vypíše přes sériovou linku některé registry pro kontrolu stavu. V produkčním nastavení se nic nevypisuje a zavolá se funkce `delay(5000)` (čekání 5 s) než začne nový cyklus. Tohoto zpoždění jsem využíval při vývoji systému pro snadnější ladění. Po ukončení vývoje jsem zpoždění ponechal, protože jsem usoudil, že načítání SMS zpráv z GSM modulu stačí co 5 sekund. V případě reálné brány lze tento čas případně zkrátit nebo úplně vynechat.

`ISR(TIMER2_OVF_vect)` je obslužná funkce přerušení při přetečení timeru 2. Z nastavené frekvence 60 Hz vyplývá, že toto přerušení nastane každých cca. 17 ms. Z důvodu vyšší priority obsluhy přerušení před hlavním programem, obsluha řeší kritické operace, na které se nemůže čekat, než se provedou jiné operace. Například zpracování zprávy z GSM modulu, může trvat příliš dlouho a pokud by se neprovedla kontrola brány, nemusel by se vypnout motor při dojezdu do koncových stavů na jedné či druhé straně. Je však nutné, aby byla obsluha co nejkratší, aby se hlavní smyčka dostala na řadu. Ve funkci se tedy zajišťuje stav brány a její případné zastavení. Zavoláním metody `gate.Update()` se zajistí aktualizace stavu brány a případné zastavení motorů v závislosti na tomto stavu.

#### 4.1.7 Knihovna GSM\_GPRS\_GPS\_Shield

Knihovna `GSM_GPRS_GPS_Shield` je open source a obsahuje soubor tříd pro práci s GSM moduly s čipy SIM900 nebo SIM908. Poskytují možnost zpracovávat SMS, volání, e-mail, GPRS. Vše samozřejmě funguje pouze s vloženou SIM kartou.

Inicializace modulu se spustí zavoláním metody `gsm.begin(9600)`, čímž se nejprve vytvoří sériová komunikace s rychlostí baudrate, která je vložena v parametru. V inicializaci se pak provede samotné spuštění GSM modulu, aby se začal přihlašovat do sítě. To se provádí pomocí digitálního pinu 7, na který se přivede na dobu 1 s logická 1 a tím se začne modul přihlašovat do sítě. Procesor se snaží navázat spojení s modulem pomocí AT příkazů. Jakmile se spojí, tak je tato metoda ukončena.

Třída `gsm` obsahuje i metody pro zvedání a pokládání hovoru, příjem SMS a další, ale většina už je deprecated a používají se konkrétní třídy, které obsluhují konkrétní operace. Třída `MSGSM` obsluhuje SMS zprávy, `CallGSM`, hovory a další.

Pro příjem SMS zpráv se tedy používá třída `MSGSM`, kdy se nejprve zjistí, jestli existuje nová zpráva pomocí metody `IsSMSPresent(SMS_UNREAD)`. Pokud najde zprávu, tak vrátí pozici, na které se zpráva nachází. Pozici pak použijeme pro načtení zprávy v metodě `GetSMS`, která vrátí text zprávy a číslo odesílatele.

V této metodě je však jedna chyba, která za určitých podmínek nedovolí přečíst celou zprávu o délce 160 znaků. Přečte jenom část. Vzhledem ke komunikaci s modulem pomocí AT příkazů se i odpověď vrací v tomto speciálním formátu. V jedné odpovědi se nevrací pouze text, ale i číslo odesílatele, jméno odesílatele, pokud je uloženo na SIM, čas příchodu a stav, že zpráva proběhla v pořádku. Buffer použitý v metodě, do kterého se ukládá celá odpověď je však pouze na 200 znaků, což často nemusí stačit. Řešením je nastavit tento buffer na větší. Dříve byla kapacita na SIM velmi malá a pro jméno se mohlo používat pouze 20 znaků. V novějších SIM kartách je místa již více a tedy i jména mohou být delší. Proto není zrovna jednoznačně určitelné, jak velký buffer je potřeba nastavit, aby byla jistota příjmu celé zprávy z modulu.

Buffer jsem zvětšil na velikost 230 znaků, což stačí na jména do délky 20 znaků.

Pro posílání zprávy se pak používá metoda `SendSMS`, která může používat pro odeslání jak číslo příjemce, komu se má zpráva poslat, nebo jenom pozice na SIM, ze které si číslo sama získá.

## 4.2 Zabezpečení brány

K zabezpečení brány je využito GSM modulu jako aktivního zprostředkovatele stavu brány. V hlavním cyklu kódu se neustále kontroluje stav brány, jakmile je zjištěn chybový stav, tak se okamžitě odešle varovná zpráva na číslo uloženo na první pozici SIM



karty. V případě trvání chyby se zpráva při dalších průchodech cyklem programu nepošle. Znovu by se poslala až ve chvíli, kdy by se stav brány znovu vrátil do kteréhokoliv nechybového.

Uživatel si může stav brány kdykoliv zkontrolovat pomocí odeslání požadavku **Status** na bránu, která vzápětí vrátí stav, ve kterém se brána v aktuální chvíli vyskytuje.

Pro ještě vyšší zabezpečení by šlo bránu doplnit o elektronický zámek, který by ji zajistil, jakmile by dosáhla koncových pozic.

### 4.3 Stav brány

Brána se může vyskytovat ve stavech:

**OPENED** – otevřeno – na čidle pro otevření je logická 1 a motor stojí

**CLOSED** – zavřeno – na čidle pro zavření je logická 1 a motor stojí

**OPENING** – otevírání – na čidle pro otevření i zavření je logická 0 a motor je spuštěn ve směru otevírání

**CLOSING** – zavřeno – na čidle pro otevření i zavření je logická 0 a motor je spuštěn ve směru zavírání

**ERROR** – chyba – pokud je na obou čidlech logická 0 a motor stojí nebo je na obou čidlech 1

Mezi jednotlivými stavy se přechází automaticky. Pouze pokud přijde požadavek od uživatele, pro otevření nebo zavření, tak se spustí motory a tím se zároveň aktualizuje status.

### 4.4 Uživatelské příkazy

Příkazy se posílají pomocí SMS a musí být napsány přesně a bez jakýchkoliv bílých znaků.

**Open** – otevřít – zavolá se metoda pro otevření brány `gate.Open()` (4.1.4)

**Close** – zavřít – zavolá se metoda pro zavření brány `gate.Close()` (4.1.4)

- – snížit rychlost – zavolá se metoda pro snížení rychlosti brány `gate.DecreaseDuty(duty)`. Můžeme zavolat 1 až 3 pomlčky napsané hned po sobě bez mezer, čím více pomlček, tím více se motor zpomalí. Rychlost se ukládá do paměti trvale. I po restartu si pamatuje poslední nastavenou rychlost.

+ – zvýšit rychlost – zavolá se metoda pro zvýšení rychlosti brány `gate.IncreaseDuty(duty)`. Můžeme zavolat 1 až 3 + napsaná hned po sobě bez mezer, čím více pomlček, tím více se motor zrychlí. Rychlost se ukládá do paměti trvale. I po restartu si pamatuje poslední nastavenou rychlost.

**Status** – stav – zavolá se metoda `gate.Status()`, která vrátí aktuální stav brány (4.3) a odešle jej zpět uživateli, který dotaz poslal.

#### 4.5 Možná vylepšení

Další možností, kterou by šlo realizovat, je systém otevírání a zavírání na pouhé prozvonění. Uživatel by zavolal na číslo brány, brána by telefon položila a následně otevřela nebo zavřela v závislosti na předchozím stavu.

Rozumnou možností vylepšení bych viděl logování některých stavů, jako je příjem příkazu od nevalidního kontaktu. Šlo by řešit například použitím čtečky paměťové karty a logovat SMS i s kontaktem na paměťovou kartu. Nebo použít ethernet modul, který by se připojil do sítě LAN a logovalo by se na nějaký server.

Při použití ethernet modulu, by šlo řešit i samotné ovládání přes internet. Vytvořila by se nová třída zděděná od `ICommunication`. Muselo by se však vymyslet ověření odesílatele příkazu.

Při realizaci skutečné brány by bylo vhodné doplnit bránu o další senzor na každé straně brány, které by sloužily jako dojezdové senzory. Na procesoru je dostatek volných vstupů pro jejich připojení. Po dosažení brány do tohoto bodu by se snížila rychlost posunu brány, pomocí změny střídý na výstupech k motoru. Významně by se tím dal snížit čas k otevírání a zavírání brány, protože by počáteční rychlost mohla být větší.

## ZÁVĚR

Náplní této práce bylo ukázat, jakým způsobem je možné ovládat brány pomocí GSM modulu.

Elektronická část byla navržena pro GSM modul s čipem SIM900 od firmy SIMCOM. Pro řízení byla vybrána platforma Arduino. Konkrétně se jednalo o modul Arduino UNO s mikroprocesorem ATMEGA328p. Podařilo se navrhnout jednoduchý a použitelný model brány na hliníkových profilech. Pro pohon byl použit motor s ozubeným řemínkem ze staré tiskárny. Cena modelu byla proto minimální.

K ovládání modelu brány byl vytvořen shield s H-můstkem pro řízení motoru a vstupy pro příjem signálu z optických čidel na konstrukci brány. Při odzkoušení zapojení na nepájivém poli vše fungovalo správně. Při zapojení na desku plošných spojů však přestal fungovat jeden z výkonových tranzistorů. Po jeho výměně vše funguje, jak má.

Součástky H-můstku vyšly zhruba na 100 Kč, kdy nejdražší byly MOSFET tranzistory. Arduino UNO se dá v Česku sehnat za cenu kolem 500 Kč. GSM modul pak vyjde na zhruba 1000 Kč. Já využil možnosti objednání jak Arduina, tak GSM modulu z Číny. Cena obou desek byla do 1000 Kč.

Při konstrukci brány jsem řešil výběr senzorů a po testech jsem vybral optická čidla typu TCST1103. Dále byly testovány jazýčková relé a jejich spínání pomocí magnetů. Pro takto malý model tento druh senzoru nebyl vhodný. Avšak testy bylo prokázáno, že na reálných branách by byl vhodný. V reálných branách by bylo také vhodné použití mechanického tlačítka. V modelu nebylo dostatek místa pro použití více druhů senzorů najednou.

Díky možnosti GSM modulu odesílat zprávy uživateli, je software vybaven automatickým odesláním zprávy při detekci chybového stavu. Uživatel tak okamžitě vidí, že je brána narušena.

Použitá řídicí deska umožňuje další rozšíření vlastností brány. Ze třinácti vstupně výstupních digitálních pinů, zůstaly 4 nevyužity. Tyto piny tak můžeme využít například k přidání dalších vstupních senzorů. V případě, že by bylo potřeba použít ještě více vstupů, pak lze program snadno použít v Arduino MEGA 2560, který nabízí 54 digitálních pinů.

## SEZNAM POUŽITÉ LITERATURY

- [1] Arduino - ArduinoBoardUno. (cit. 19.4.2015), [Online].  
URL <http://www.arduino.cc/en/Main/ArduinoBoardUno>
- [2] Arduino - ArduinoBoardUno Schéma. (cit. 20.4.2015), [Online].  
URL [http://www.arduino.cc/en/uploads/Main/Arduino\\_Uno\\_Rev3-schematic.pdf](http://www.arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf)
- [3] Co je to Arduino? | Arduino.cz. (cit. 4.5.2015), [Online].  
URL <http://arduino.cz/co-je-to-arduino/>
- [4] Co je to Arduino? | Arduino.cz. (cit. 4.5.2015), [Online].  
URL <http://arduino.cz/seznameni-s-arduinem/>
- [5] Arduino - ArduinoBoardUno. (cit. 4.5.2015), [Online].  
URL <http://www.arduino.cc/en/Main/ArduinoBoardUno>
- [6] Datasheet ATmega328P. (cit. 2.5.2015), [Online].  
URL <http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48-64-128-256-512-1024-2048-4096-8192-16384-32768-bits.pdf>
- [7] Barrett, S. F.: *Arduino microcontroller: processing for everyone! 2nd ed.* Morgan, 2012, ISBN 978-160-8458-592.
- [8] Evans, B.: *Beginning arduino programming: writing code for the most popular microcontroller board in the world.* Apress, 2011, ISBN 978-1-4302-3777-8.
- [9] Bezpečnostní fotobuňky - Teleskopické a jiné brány. (cit. 25.4.2015), [Online].  
URL <http://www.vjezdove-brany.eu/teleskopicka-brana/bezpecnost-a-ovladani/bezpecnostni-fotobunky/>
- [10] technictest.com - GSM - (Global System for Mobile) historie. (cit. 2.5.2015), [Online].  
URL [http://www.technictest.com/gsm\\_historie.php](http://www.technictest.com/gsm_historie.php)
- [11] GSM Klíč - mobilem zdarma otevři vrata, brány, závory. (cit. 25.4.2015), [Online].  
URL <http://www.gsmklic.cz/>
- [12] SIM900 Quad-Band GPRS shield with Micro SD card slot - Epalsite Wiki. (cit. 10.5.2015), [Online].  
URL [http://wiki.epalsite.com/index.php?title=SIM900\\_Quad-Band\\_GPRS\\_shield\\_with\\_Micro\\_SD\\_card\\_slot](http://wiki.epalsite.com/index.php?title=SIM900_Quad-Band_GPRS_shield_with_Micro_SD_card_slot)

- [13] Základní struktura GSM. (cit. 2.5.2015), [Online].  
URL <http://tomas.richtr.cz/mobil/gsm-strukt.htm>
- [14] GSM structure. (cit. 2.5.2015), [Online].  
URL [http://upload.wikimedia.org/wikipedia/commons/d/d1/Gsm\\_structures.svg](http://upload.wikimedia.org/wikipedia/commons/d/d1/Gsm_structures.svg)
- [15] John-David Warren, H. M., Josh Adams: *Arduino robotics*. Apress, 2011, ISBN 978-1-4302-3183-7.
- [16] a Joseph E. Wilkes, V. K. G.: *Principles and applications of GSM*. Prentice Hall, 1999, ISBN 01-394-9124-4.
- [17] Posuvná garážová vrata LOMAX. (cit. 24.4.2015), [Online].  
URL <http://www.lomax.cz/cs/garazova-vrata/posuvna-vrata/>
- [18] Margolis, M.: *Arduino cookbook*. O'Reilly, 2011, ISBN 978-0-596-80247-9.
- [19] Muhammad A. Mazidi, S. N., Sarmad Naimi: *The AVR microcontroller and embedded systems: using Assembly and C*. Prentice Hall, 2011, ISBN 01-380-0331-9.
- [20] Co je plovoucí kód? | Levné Alarmy.cz - autoalarmy, centrální zamykání, parkovací senzory, LED světla a další. (cit. 25.4.2015), [Online].  
URL <http://www.levnealarmy.cz/recenzie-a-clanky/slovník-pojmu/co-je-plovouci-kod.html>
- [21] Pohon pro křídlové brány - manuál P200. (cit. 24.4.2015), [Online].  
URL [http://www.epohony.cz/admin/files/e\\_product\\_files/0/76/src\\_P200manul\\_CZ\\_2.1.pdf](http://www.epohony.cz/admin/files/e_product_files/0/76/src_P200manul_CZ_2.1.pdf)
- [22] Posuvná garážová vrata LOMAX. (cit. 24.4.2015), [Online].  
URL <http://www.ceskestavby.cz/clanky/posuvne-brany-setri-misto-20205.html>
- [23] Renner, G.: *Borland C++ Kompendium znalostí a zkušeností*. Unis, 1992, ISBN 80-238-0315-8.
- [24] IBF-Wiki. (cit. 19.4.2015), [Online].  
URL <http://server.ibfriedrich.com/wiki/ibfwikien/index.php>
- [25] TCST1130 Datasheet. (cit. 21.4.2015), [Online].  
URL <http://www.vishay.com/docs/83764/tcst1103.pdf>
- [26] Timo Halonen, J. M., Javier Romero: *GSM, GPRS and edge performance: evolution towards 3G/UMTS 2nd ed.* John Wiley, 2003, ISBN 04-708-6694-2.

- 
- [27] Elektrické posuvné brány vám usnadní život - Brána bydlení.cz. (cit. 25.4.2015), [Online].  
URL <http://www.brana-bydleni.cz/elektricke-posuvne-brany-vam-usnadni-zivot/>
- [28] Wilson, J. S.: *Sensor technology handbook*. Elsevier, 2005, ISBN 978-0-7506-7729-5.

**SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK**

DPS	Deska plošných spojů
PWM	Pulse width modulation
GSM	Global System for Mobile Communications
ETSI	European Telecommunications Standards Institute
UMTS	Universal Mobile Telecommunications System
CEPT	European Conference of Postal and Telecommunications Administrations
EU	Evropská unie
GPRS	General Packet Radio Service
W-CDMA	Wideband Code Division Multiple Access
MMS	Multimedia Messaging Service
ME	Mobile Equipment
BTS	Base Transceiver Station
IMEI	International Mobile Equipment Identity
SIM	Subscriber Identification Module
TDMA	Time Division Multiple Access
OSS	Operations support system
IP	Internet protokol
PIN	Personal identification number
PUK	Personal unblocking code
IMSI	International Mobile Subscriber Identity
Ki	Autentikační klíč
Kc	Šifrovací klíč
TMSI	Temporary Mobile Subscriber Identity
LAI	Location Area Identity
USIM	Universal Subscriber Identity Module
VLR	Visitor Location Register
MSC	Mobile Switching Centre
USB	Universal Serial Bus
ICSP	In-Circuit Serial Programming
SRAM	Static Random Access Memory
EEPROM	Electrically Erasable Programmable Read-Only Memory
RX	Receive data
TX	Transmit data
COM	Component Object Model
LED	Light-Emitting Diode

UART	universal asynchronous receiver/transmitter
TWI	Two Wire Interface
SPI	Serial Peripheral Interface
IDE	Integrated Development Environment
CNC	Computer Numeric Control
SMS	Short message service
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
LAN	Local area network



**SEZNAM OBRÁZKŮ**

Obr. 1.1	Struktura sítě GSM [14] . . . . .	13
Obr. 1.2	GSM shield připojený na Arduino UNO . . . . .	17
Obr. 1.3	3D zobrazení navržené desky v Target 3001! . . . . .	20
Obr. 2.1	Sekční posuvná vrata LOMAX [17] . . . . .	22
Obr. 2.2	Pohon křídlové brány [21] . . . . .	23
Obr. 3.1	Model brány . . . . .	28
Obr. 3.2	Schéma napájení modulu z arduina . . . . .	29
Obr. 3.3	Schéma H-můstku . . . . .	30
Obr. 3.4	Schéma zapojení shieldu pro Arduino Uno . . . . .	31
Obr. 3.5	Schéma aplikace senzoru TCST1103 . . . . .	32
Obr. 3.6	Umístění čidla na bráně . . . . .	32
Obr. 3.7	Deska plošných spojů senzoru 1:1 . . . . .	33
Obr. 3.8	Osazení senzoru . . . . .	33
Obr. 3.9	Deska plošných spojů 1:1 . . . . .	33
Obr. 3.10	Osazení . . . . .	34
Obr. 4.1	Vývojový diagram aplikační smyčky . . . . .	39

**SEZNAM TABULEK**

Tab. 1.1	Indikace stavu GSM modulu [12] . . . . .	16
Tab. 1.2	Parametry Arduino UNO . . . . .	18

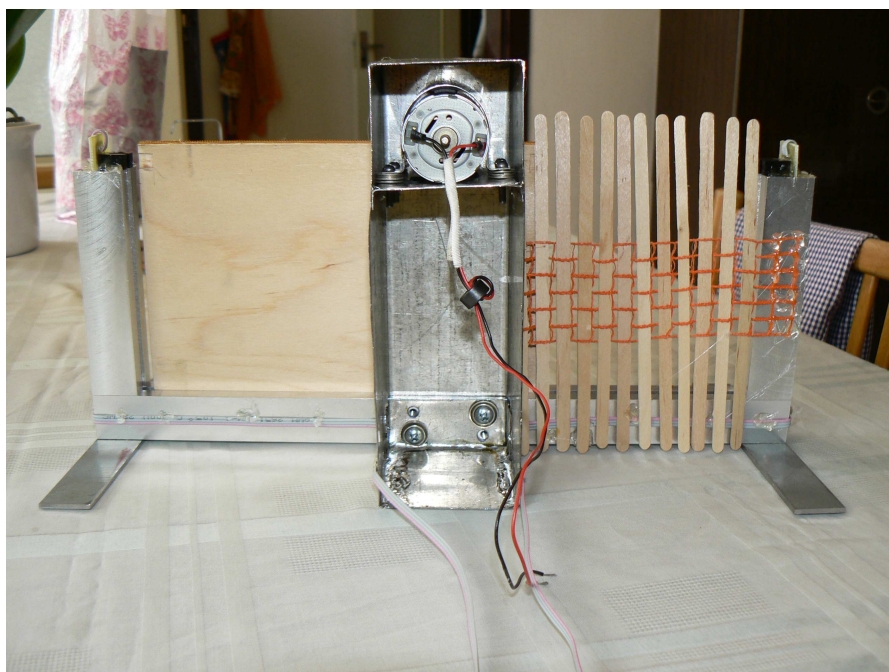
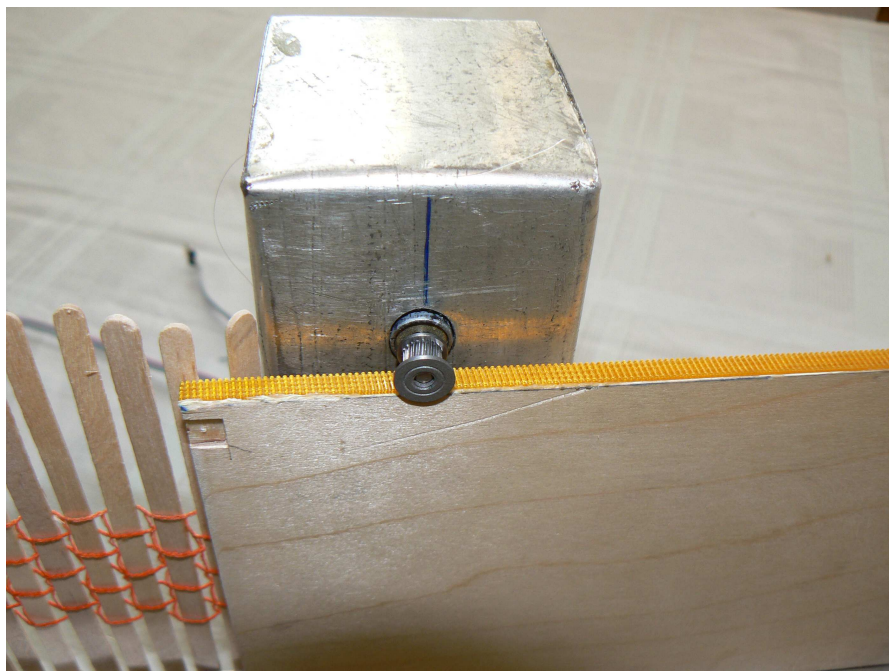
## SEZNAM PŘÍLOH

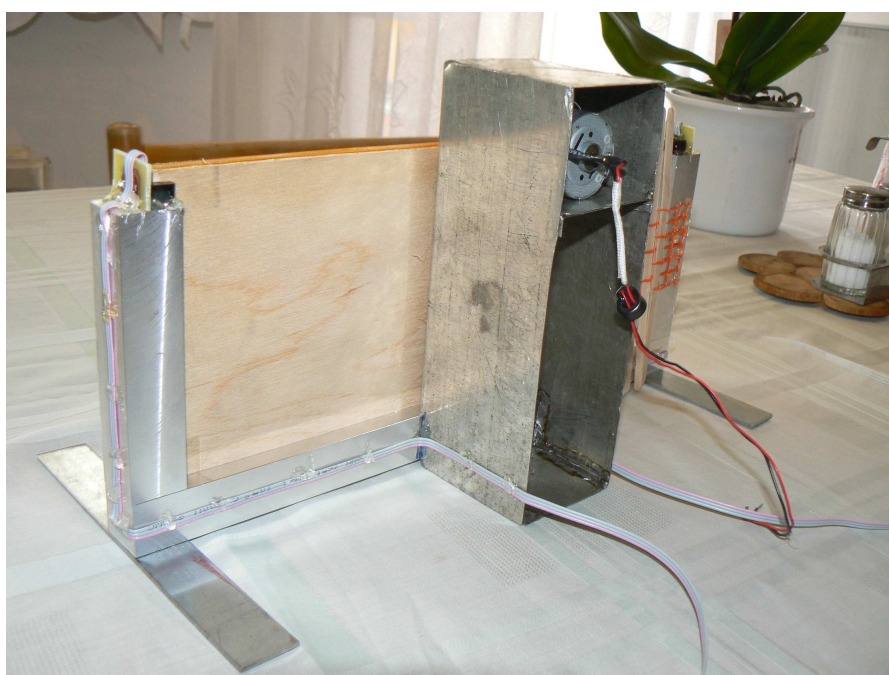
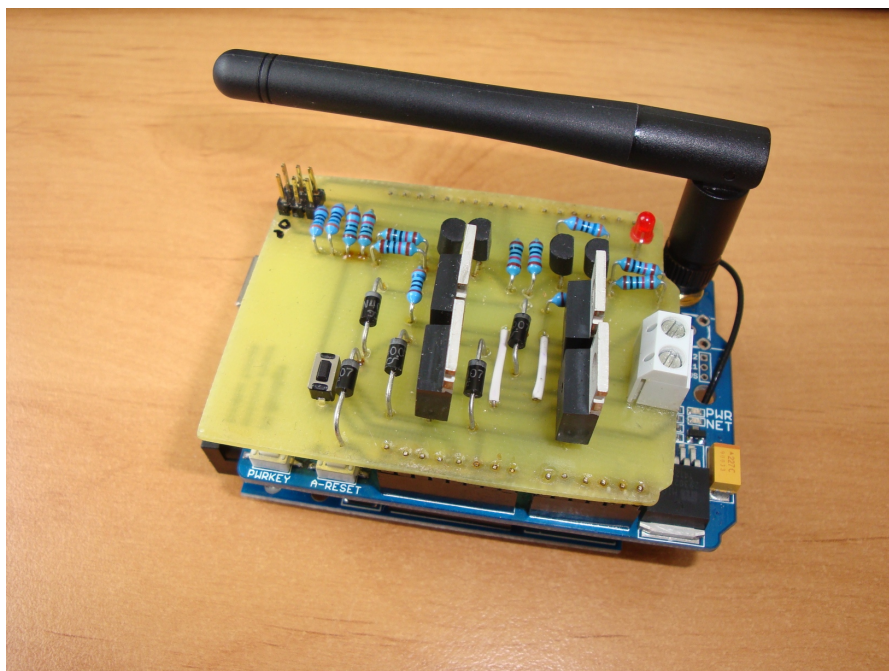
- P I. Seznam použitých součástí
- P II. Fotodokumentace
- P III. Zdrojové kódy

## PŘÍLOHA P I. SEZNAM POUŽITÝCH SOUČÁSTEK

	Počet kusů	Typ	Označení ve schématu
1	8	10k $\Omega$	R1, R2, R3, R4, R5, R6, R10, R13
2	5	390 $\Omega$	R7, R8, R9, R11, R12
3	4	IRFZ44N	T1, T2, T3, T4
4	4	BC546B	T5, T6, T7, T8
5	5	IN4007	D1, D2, D3, D4, D5
6	1	LED červená 3mm	D6
7	1	Tlačítko PHAP3362	S2
8	2	TCST1103	X1, X3
9	1	Svorkovnice	J2
10	2	Lámací kolíková lišta jednořadá 3 piny	J3, J4
11	1	Lámací kolíková lišta dvouřadá 3 piny	J1
12	2	Lámací kolíková lišta jednořadá 6 pinů	X2a, X2d
13	1	Lámací kolíková lišta jednořadá 7 pinů	X2c
14	1	Lámací kolíková lišta jednořadá 10 pinů	X2b

## PŘÍLOHA P II. FOTODOKUMENTACE





## PŘÍLOHA P III. ZDROJOVÉ KÓDY

### Listing 1 GateGSM.ino

```
#include <SoftwareSerial.h>
#include "ICommunication.h"
// #define DEBUG_SERIAL_COMMUNICATION // Disable GSM Module - uses Serial
// #define _DEBUG
// #define _DEBUG
#ifdef _DEBUG
#define DEBUG_PRINT(MESSAGE) SerialPrint(MESSAGE)
#else
#define DEBUG_PRINT(MESSAGE)
#endif
#define DEBUG_SERIAL_COMMUNICATION
#include "SerialCommunication.h"
#else
#include "GSMCommunication.h"
#endif
#include "Gate.h"
#include <SIM900.h>
#include <sms.h>
#include <MemoryFree.h>
#include <pgmStrToRAM.h>
#define ERROR_MESSAGE "The gate is not closed. No command was send."
Gate gate;
ICommunication* communication;
bool errorSended = false;
char* GateStatusLabel[] = {
    "OPENED",
    "CLOSED",
    "OPENING",
    "CLOSING",
    "ERROR",
    "STOPPED"
};
void setup()
{
    #ifndef DEBUG_SERIAL_COMMUNICATION
        Serial.begin(9600);
        Serial.println(F("Begin Init"));
    #endif
    #ifdef DEBUG_SERIAL_COMMUNICATION
        communication = new SerialCommunication();
    #else
        communication = new GSMCommunication();
    #endif
    // init timer2
    noInterrupts(); // disable all interrupts
    TCCR2A = 0;
    TCCR2B = 0;
    TCNT2 = 0;
    // TCCR2A |= _BV(WGM21); // CTC mode
    TCCR2B |= _BV(CS22) | _BV(CS21) | _BV(CS20); // 1024 prescaler - c. 60Hz
    TIMSK2 |= _BV(TOIE2); // enable overflow interrupt
    gate.Init();
    interrupts(); // enable all interrupts
    Serial.println(F("Close on Start"));
    gate.Close();
    Serial.println(F("End Init"));
}
void loop()
{
    char* message;
    char* phoneNumber;
    #ifdef _DEBUG
        Serial.print(F("RamBeforeRead= "));
        Serial.println(freeMemory(), DEC);
    #endif
    if(communication->rcvMessage(&message, &phoneNumber))
    {
        control(message, phoneNumber);
    }
    #ifdef _DEBUG
        Serial.print(F("RamAfterRead= "));
        Serial.println(freeMemory(), DEC);
    #endif
    free(message);
    free(phoneNumber);
    // Serial.print(F("RamLoopEnd= "));
    // Serial.println(freeMemory(), DEC);
    GateStatus status = gate.getStatus();
    if(status == ERROR)
    {
        if(!errorSended)
        {
            communication->sendMessage(ERROR_MESSAGE);
            errorSended = true;
        }
    }
    else
        errorSended = false;
    #ifdef DEBUG_SERIAL_COMMUNICATION
```

```

        Serial.print(F("Status: "));
        Serial.println(GateStatusLabel[status]);
    #endif
    #ifdef _DEBUG
        Serial.print(F("ICR1: "));
        Serial.println(ICR1);
        Serial.print(F("OCR1A: "));
        Serial.println(OCR1A);
        Serial.print(F("OCR1B: "));
        Serial.println(OCR1B);
        Serial.println(F("LoopEnd"));
    #endif
    #ifdef DEBUG_SERIAL_COMMUNICATION
        delay(2000);
    #else
        delay(5000);
    #endif
}
ISR(TIMER2_OVF_vect)
{
    gate.Update();
    //GateStatus status = gate.getStatus();
}
void control(char* message, char* phoneNumber)
{
    Serial.println(message);
    if(strcmp(message, "Open") == 0)
    {
        Serial.println(F("Opening"));
        gate.Open();
    }
    else if(strcmp(message, "Close") == 0)
    {
        Serial.println(F("Closing"));
        gate.Close();
    }
    else if(strcmp(message, "Stop") == 0)
    {
        Serial.println(F("Stopping"));
        gate.StopMotor();
    }
    else if(strcmp(message, "+") == 0)
    {
        Serial.println(F("Increase"));
        gate.IncreaseDuty(100);
    }
    else if(strcmp(message, "++") == 0)
    {
        Serial.println(F("Increase"));
        gate.IncreaseDuty(1000);
    }
    else if(strcmp(message, "+++") == 0)
    {
        Serial.println(F("Increase"));
        gate.IncreaseDuty(5000);
    }
    else if(strcmp(message, "-") == 0)
    {
        Serial.println(F("Decrease"));
        gate.DecreaseDuty(100);
    }
    else if(strcmp(message, "--") == 0)
    {
        Serial.println(F("Decrease"));
        gate.DecreaseDuty(1000);
    }
    else if(strcmp(message, "---") == 0)
    {
        Serial.println(F("Decrease"));
        gate.DecreaseDuty(5000);
    }
    else if(strcmp(message, "Status") == 0)
    {
        gate.Update();
        GateStatus status = gate.getStatus();
        communication->sendMessage(GateStatusLabel[status], phoneNumber);
    }
    else
    {
        Serial.print(F("Wrong command: "));
        Serial.println(message);
    }
}
}

```

## Listing 2 EepromAddresses.h

```

#define EEPROM_ADDRESS_PWM_DUTY 0 // 2 bytes

```



### Listing 3 Gate.h

```
#ifndef GATE_H
#define GATE_H
#include <arduino.h>
#include <wiring_private.h>
#include <avr/eeprom.h>
#include "EepromAddresses.h"
#define TIMER1_A_PIN 9
#define TIMER1_B_PIN 10
#define PIN_GATE_MOTOR_POWER_OPEN_DUTY OCR1A // pin 9
#define PIN_GATE_MOTOR_POWER_CLOSE_DUTY OCR1B // pin 10
#define PIN_GATE_OPENED 11
#define PIN_GATE_CLOSED 12
#define PIN_MOTOR_ON_LED 6
#define ICR_TOP 65535
#define PWM_DUTY 17500
enum GateStatus { OPENED, CLOSED, OPENING, CLOSING, ERROR, STOPPED};
class Gate{
private:
    GateStatus _status;
    bool initialized;
    unsigned int _duty;
    void loadPwmDuty();
    void savePwmDuty(unsigned int duty);
    //GateStatus CheckStatus();
public:
    Gate();
    ~Gate();
    void Init();
    void Open();
    void Close();
    void StopMotor();
    void Update();
    GateStatus getStatus();
    void SetMotorDuty(unsigned int duty);
    unsigned int IncreaseDuty(unsigned int offset);
    unsigned int DecreaseDuty(unsigned int offset);
    void savePwmDuty();
};
#endif
```

### Listing 4 Gate.cpp

```
#include "Gate.h"
//#include <arduino.h>
Gate::Gate()
{
    initialized = false;
}
Gate::~~Gate()
{
}
void Gate::Init()
{
    loadPwmDuty();
    TCCR1A = 0;
    TCCR1B = 0;
    TCNT1 = 0;
    ICR1 = ICR_TOP;
    TCCR1A |= _BV(WGM11) | _BV(COM1A1) | _BV(COM1B1);
    TCCR1B |= _BV(CS10); // prescaler 1
    TCCR1B |= _BV(WGM12) | _BV(WGM13); // fast PWM - TOP ICR_TOP
    pinMode(TIMER1_A_PIN, OUTPUT);
    pinMode(TIMER1_B_PIN, OUTPUT);
    pinMode(PIN_MOTOR_ON_LED, OUTPUT);
    pinMode(PIN_GATE_OPENED, INPUT_PULLUP);
    pinMode(PIN_GATE_CLOSED, INPUT_PULLUP);
    Update();
    initialized = true;
    StopMotor();
}
void Gate::Update()
{
    if(initialized)
    {
        bool isOpened = digitalRead(PIN_GATE_OPENED) == 1; //Pull up resistor negate
        //↳ logic
        bool isClosed = digitalRead(PIN_GATE_CLOSED) == 1; //Pull up resistor negate
        //↳ logic
        GateStatus result = ERROR;
        if(isOpened)
        {
            if(isClosed)
            {
                result = ERROR;
                StopMotor();
            }
        }
    }
}
```

```

        else
        {
            if(_status == OPENING)
                StopMotor();
            result = OPENED;
        }
    }
    else if(isClosed)
    {
        if(_status == CLOSING)
            StopMotor();
        result = CLOSED;
    }
    else
    {
        if(PIN_GATE_MOTOR_POWER_OPEN_DUTY > 0 || PIN_GATE_MOTOR_POWER_CLOSE_DUTY >
            ↪ 0)
        {
            if(PIN_GATE_MOTOR_POWER_OPEN_DUTY > 0)
                result = OPENING;
            else
                result = CLOSING;
        }
        else
            result = ERROR;
    }
    _status = result;
}

GateStatus Gate::getStatus(){
    return _status;
}

void Gate::Open()
{
    if(initialized)
    {
        if(_status != OPENED)
        {
            StopMotor();
            delay(100);
            sbi(TCCR1A, COM1A1);
            PIN_GATE_MOTOR_POWER_OPEN_DUTY = _duty;
            digitalWrite(PIN_MOTOR_ON_LED, 1);
        }
    }
}

void Gate::Close()
{
    if(initialized)
    {
        if(_status != CLOSED)
        {
            StopMotor();
            delay(100);
            sbi(TCCR1A, COM1B1);
            PIN_GATE_MOTOR_POWER_CLOSE_DUTY = _duty;
            digitalWrite(PIN_MOTOR_ON_LED, 1);
        }
    }
}

void Gate::StopMotor()
{
    if(initialized)
    {
        cbi(TCCR1A, COM1A1);
        cbi(TCCR1A, COM1B1);
        PIN_GATE_MOTOR_POWER_OPEN_DUTY = 0;
        PIN_GATE_MOTOR_POWER_CLOSE_DUTY = 0;
        digitalWrite(PIN_MOTOR_ON_LED, 0);
    }
}

void Gate::SetMotorDuty(unsigned int duty)
{
    if(initialized)
    {
        _duty = duty;
        if((TCCR1A >> COM1A1) & 1)
            PIN_GATE_MOTOR_POWER_OPEN_DUTY = _duty;
        else if((TCCR1A >> COM1B1) & 1)
            PIN_GATE_MOTOR_POWER_CLOSE_DUTY = _duty;
        savePwmDuty();
    }
}

unsigned int Gate::IncreaseDuty(unsigned int offset)
{
    if((_duty - offset) <= ICR_TOP)
        SetMotorDuty(_duty + offset);
    return _duty;
}

unsigned int Gate::DecreaseDuty(unsigned int offset)

```

```

{
    if((_duty - offset) > 0)
        SetMotorDuty(_duty - offset);
    return _duty;
}
void Gate::loadPwmDuty()
{
    while(!eeprom_is_ready())
    {
        delay(1);
    }
    _duty = eeprom_read_word(EEPROM_ADDRESS_PWM_DUTY);
    if(_duty == 0)
    {
        _duty = PWM_DUTY;
        savePwmDuty(_duty);
    }
    Serial.print(F("Loaded duty: "));
    Serial.println(_duty);
}
void Gate::savePwmDuty(unsigned int duty)
{
    while(!eeprom_is_ready())
    {
        delay(1);
    }
    eeprom_write_word(EEPROM_ADDRESS_PWM_DUTY, duty);
    Serial.print(F("Saved duty: "));
    Serial.println(duty);
}
void Gate::savePwmDuty()
{
    savePwmDuty(_duty);
}

```

Listing 5 ICommunication.h

```

#ifndef ICOMMUNICATION_H
#define ICOMMUNICATION_H
#include <arduino.h>
class ICommunication
{
public:
    virtual void Init() = 0;
    /// <summary>
    /// Returns 1 if new message exists else 0. New message from numbers on SIM card
    /// </summary>
    virtual byte rcvMessage(char** message, char** phoneNumber) = 0;
    /// <summary>
    /// Send message to first number on SIM card
    /// </summary>
    virtual void sendMessage(char* message) = 0;
    /// <summary>
    /// Send message to phoneNumber
    /// </summary>
    virtual void sendMessage(char* message, char* phoneNumber) = 0;
};
#endif

```

Listing 6 GSMCommunication.h

```

#ifndef GSMCOMMUNICATION_H
#define GSMCOMMUNICATION_H
#include <arduino.h>
#include "ICommunication.h"
#include <SIM900.h>
#include <sms.h>
#define GSM_POWER_PIN 7
class GSMCommunication : public ICommunication
{
private:
    MSGSMS sms;
public:
    GSMCommunication();
    ~GSMCommunication();
    virtual void Init();
    virtual byte rcvMessage(char** message, char** phoneNumber);
    virtual void sendMessage(char* message);
    virtual void sendMessage(char* message, char* phoneNumber);
};
#endif

```

Listing 7 GSMCommunication.cpp

```

#include "GSMCommunication.h"
#define SMS_LENGTH 161 // 160 znaku sms a 1 ukoncovaci

```

```

#define NUMBER_LENGTH 15          // 14 znaku sms a 1 ukoncovaci
#define AUTH_POS_FIRST 1          //
#define AUTH_POS_LAST 20         //
GSMCommunication::GSMCommunication()
{
    Init();
}
GSMCommunication::~GSMCommunication()
{
}
void GSMCommunication::Init()
{
    //gsm.begin(9600);
    gsm.begin(14400);
    //gsm.begin(19200);
}
byte GSMCommunication::rcvMessage(char** message, char** phoneNumber)
{
    int smsPosition;
    if((smsPosition = sms.IsSMSPresent(SMS_UNREAD)) > 0)
    {
        *phoneNumber = (char*)malloc(NUMBER_LENGTH * sizeof(char));
        *message = (char*)malloc(SMS_LENGTH * sizeof(char));
        char authStatus = sms.GetAuthorizedSMS(smsPosition, *phoneNumber, *message,
        ↪ SMS_LENGTH, AUTH_POS_FIRST, AUTH_POS_LAST);
        sms.DeleteSMS(smsPosition);
        if(authStatus != GETSMS_AUTH_SMS)
        {
            return 0;
        }
        return 1;
    }
    return 0;
}
void GSMCommunication::sendMessage(char* message)
{
    char n[20];
    sms.SendSMS((byte)1, message);
}
void GSMCommunication::sendMessage(char* message, char* phoneNumber)
{
    char n[20];
    sms.SendSMS(phoneNumber, message);
}

```

### Listing 8 GSMCommunication.h

```

#ifndef GSMCOMMUNICATION_H
#define GSMCOMMUNICATION_H
#include <arduino.h>
#include "ICommunication.h"
#include <SIM900.h>
#include <sms.h>
#define GSM_POWER_PIN 7
class GSMCommunication : public ICommunication
{
private:
    MSGSMS sms;
public:
    GSMCommunication();
    ~GSMCommunication();
    virtual void Init();
    virtual byte rcvMessage(char** message, char** phoneNumber);
    virtual void sendMessage(char* message);
    virtual void sendMessage(char* message, char* phoneNumber);
};
#endif

```

### Listing 9 GSMCommunication.cpp

```

#include "GSMCommunication.h"
#define SMS_LENGTH 161          // 160 znaku sms a 1 ukoncovaci
#define NUMBER_LENGTH 15       // 14 znaku sms a 1 ukoncovaci
#define AUTH_POS_FIRST 1       //
#define AUTH_POS_LAST 20       //
GSMCommunication::GSMCommunication()
{
    Init();
}
GSMCommunication::~GSMCommunication()
{
}
void GSMCommunication::Init()
{
    //gsm.begin(9600);
    gsm.begin(14400);
    //gsm.begin(19200);
}

```

```

byte GSMCommunication::rcvMessage(char** message, char** phoneNumber)
{
    int smsPosition;
    if((smsPosition = sms.IsSMSPresent(SMS_UNREAD)) > 0)
    {
        *phoneNumber = (char*)malloc(NUMBER_LENGTH * sizeof(char));
        *message = (char*)malloc(SMS_LENGTH * sizeof(char));
        char authStatus = sms.GetAuthorizedSMS(smsPosition, *phoneNumber, *message,
        ↪ SMS_LENGTH, AUTH_POS_FIRST, AUTH_POS_LAST);
        sms.DeleteSMS(smsPosition);
        if(authStatus != GETSMS_AUTH_SMS)
        {
            return 0;
        }
        return 1;
    }
    return 0;
}

void GSMCommunication::sendMessage(char* message)
{
    char n[20];
    sms.SendSMS((byte)1, message);
}

void GSMCommunication::sendMessage(char* message, char* phoneNumber)
{
    char n[20];
    sms.SendSMS(phoneNumber, message);
}

```

#### Listing 10 SerialCommunication.h

```

#ifndef SERIAL_COMMUNICATION_H
#define SERIAL_COMMUNICATION_H
#include <arduino.h>
#include "ICommunication.h"
class SerialCommunication : public ICommunication
{
private:
public:
    SerialCommunication();
    ~SerialCommunication();
    virtual void Init();
    virtual byte rcvMessage(char** message, char** phoneNumber);
    virtual void sendMessage(char* message);
    virtual void sendMessage(char* message, char* phoneNumber);
};
#endif

```

#### Listing 11 SerialCommunication.cpp

```

#include "SerialCommunication.h"
#define SMS_LENGTH 161 // 160 znaku sms a 1 ukoncovaci
#define NUMBER_LENGTH 15 // 14 znaku sms a 1 ukoncovaci
SerialCommunication::SerialCommunication()
{
    Init();
}

SerialCommunication::~SerialCommunication()
{
}

void SerialCommunication::Init()
{
    Serial.begin(9600);
    Serial.println(F("Begin Init"));
}

byte SerialCommunication::rcvMessage(char** message, char** phoneNumber)
{
    byte i = 0;
    if(Serial.available() > 0)
    {
        *phoneNumber = (char*)malloc(NUMBER_LENGTH * sizeof(char));
        *message = (char*)malloc(SMS_LENGTH * sizeof(char));
        **message = 0;
        **phoneNumber = 0;
        while(Serial.available() > 0)
        {
            *(*message + i) = (char)Serial.read();
            i++;
            delayMicroseconds(1000);
        }
        *(*message + i) = 0;
        return 1;
    }
    return 0;
}

void SerialCommunication::sendMessage(char* message)
{
    Serial.println(message);
}

```

```
}  
void SerialCommunication::sendMessage(char* message, char* phoneNumber)  
{  
    Serial.println(message);  
}
```