VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ČASOMÍRA PRO OBHAJOBY PROJEKTŮ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL NAVRÁTIL
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ČASOMÍRA PRO OBHAJOBY PROJEKTŮ

PROJECT DEFENCE TIMEKEEPING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL NAVRÁTIL
AUTHOR

VEDOUCÍ PRÁCE Doc. Ing. RICHARD RŮŽIČKA, Ph.D.
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této práce je navrhnout a vyrobit ethernetový modul pro časoměrný panel obsahující hodiny a stopky, který panelu bude poskytovat síťové rozhraní a nápajet jej přes ethernet. Bylo potřeba naprogramovat operační program pro mikroprocesor ovládající hodiny a stopky, který pomocí tohoto ethernetového modulu načte přesný čas z internetu a nastaví jej hodinám. Stopky jsou ovladatelné přes síť obslužnou aplikací, která v rámci této práce také vznikla. Pro tento účel jsem na mikroprocesoru naprogramoval server, který interaguje s obslužnou aplikací. Spustitelné soubory obslužné aplikace jsem vygeneroval pro operační systém Windows a systém Android. Spustit ji lze snadno i na operačním systému Linux. Přínosem této prace je možnost stopování času pohodlně z počítače nebo telefonu, kde jak student tak zkoušející bude mít přesnou informaci o průběhu času.

Abstract

Goal of this thesis is to design and assemble ethernet module for time measuring panel containing watch and stopwatch which will provide network interface to this panel and will power it through ethernet. One of requirements was to develop a control application for microprocessor controlling watch and stopwatch which loads time via this module from the internet and it sets the watch. Stopwatch is controllable via network through control application which was created as part of this thesis. For this purpose there was created a server running on the microprocessor which interacts with control application. Executables of control application are generated for operating systems Windows and Android. It is easy to run it on Linux as well. Contribution of this thesis is possibility to easily track time from the pc or the phone where student and examiner will have a precise information about time.

Klíčová slova

Stopky, synchronizace času, Wiznet 5500, Power over Ethernet, MC9S08, LTC4267, multiplaformní aplikace

Keywords

Stopwatch, time synchronization, Wiznet 5500, Power over Ethernet, MC9S08, LTC4267, cross platform application

Citace

Michal Navrátil: Časomíra pro obhajoby projektů, bakalářská práce, Brno, FIT VUT v Brně, 2015

Časomíra pro obhajoby projektů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Richarda Růžičky, Ph.D.

. . . . . .
Michal Navrátil
19. května 2015

Poděkování

Touto cestou bych rád poděkoval pánovi doc. Ing. Richardu Růžičkovi, Ph.D, který mi poskytl všechny potřebné informace a zázemí pro vytvoření této práce a pomohl mi s realizací a opravou vytvořeného ethernetového modulu. Dále bych rád poděkoval pánovi Ing. Václavu Šimkovi za ochotu a cenné rady při návrhu a testování desky plošných spojů, bez kterých by modul nebylo možné vyrobit.

Obsah

Úvod

Při obhajobách projektů, zkoušení studentů u státních závěrečných zkoušek nebo při jiných podobných událostech je potřeba stopovat přesný čas, po který student bude vědět v jaké fázi se nachází a kolik času do konce mu ještě zbývá.

Na naší alma mater Fakultě informačních technologií, na Vysokém učení technickém v Brně, se pro tuto činnost používá běžná kuchyňská minutka nebo se čas stopuje zběžným stopováním času na klasických hodinách.

Vznikla tedy potřeba pro realizaci něčeho praktičtějšího, co by bylo vždy dostupné na pracovišti a poskytovalo by určitý komfort pro ovládání.

Pro tento účel vznikl pod vedením pana doc. Ing. Richarda Růžičky, Ph.D. na fakultě projekt, při kterém byl vytvořený časoměrný panel zobrazující přesný čas ve formátu HH:MM:SS a pod ním stopky ve formátu MM:SS. Časoměrný panel je možné ovládat pomocí dálkového ovládače a je na něm připraven 20 pinový vývod se sběrnicí pro rozšíření jej o ethernetový modul, pomocí něhož by bylo možné hodiny ovládat přes síť.

Také díky tomuto ethernetovému modulu, by si časoměrný modul mohl po zapojení synchronizovat čas a nemusely by se hodiny nastavovat ručně.

Pokud by byl na pracovišti zdroj Power over Ethernetu zaveden, bylo by velmi praktické vést k hodinám jen jeden kabel, který by zastával jak síťovou komunikaci, tak samotné napájení. Proto další návrh při realizaci projektu je, že by se časoměrný panel mohl přes ethernetový modul napájet elektrickou energií pomocí IEEE 802.3af, což by se také realizovalo pomocí ethernetového modulu.

Úkolem této práce je tento ethernetový modul navrhnout, vyrobit a naprogramovat síťovou komunikaci přes něj pro synchronizaci času. Také naprogramovat obslužnou aplikaci, pomocí níž se stopky budou moct ovládat.

Struktura práce

V první kapitole nastíním možnosti síťové komunikace, dostupné přenosové média, způsoby synchronizace času a také hardwarový TCP stack, který pro vestavěné systémy tuto komunikaci umožňuje.

V druhé kapitole popíšu způsob přenosu elektrické energie pomocí Ethernetu a čip LTC4267, který po správném navrhnutí obvodu vyjedná se zdrojem dané napětí a pustí jej do spotřebiče.

Ve třetí kapitole se zaměřím na nyní dostupný časoměrný panel a na něm použitý mikroprocesor.

Ve čtvrté kapitole popíšu návrh ethernetového modulu, operačního programu pro mikroprocesor a návrh obslužné aplikace s protokolem pro komunikaci s mikroprocesorem.

V poslední, páté kapitole, popíšu realizaci ethernetového modulu a způsob naprogramování obslužné aplikace s operačním programem pro mikroprocesor na časoměrném panelu. Bude zde také popsáno testování výsledků práce.

Kapitola 1

Síťová komunikace a synchronizace času

Tato kapitola nastiňuje pohled na dnešní komunikační médium. Zaměřuje se na sítě typu Ethernet a jejich možnosti při napájení přes kroucenou dvojlinku pomocí jej rozšiřujícího standardu IEEE 802.3af. Dále popisuje protokol pro synchronizaci a přenos času po sítí pomocí standardu NTP/SNTP.

1.1 Úvod do Ethernetu

V modelu ISO/OSI Ethernet reprezentuje fyzickou a linkovou vrstvu. Je definovaný standardem 802.3, vydaným společností IEEE v roce 1983. Od této doby vznikla řada dodatků a revizí. Je postavený na Carrier Sense Multiple Access with Collision Detection (CSMA/CD), tedy na metodě sloužící pro řízení přístupů k přenosovému médiu a řešení případných kolizí.

Specifikace Ethernetu popisují použití různých topologií a přenosových medií. V dnešní době jsou nejrozšířenější přenosová média kroucená dvojlinka a optický kabel.

Metoda CSMA/CD je navrhnuta tak, že počítá s tím, že signál se po přenosovém mediu šíří nekonečně rychle a nenastává při přenosu dat zpoždění. Proto standardy definují maximální délky přenosového media, tzn. vzdálenosti mezi aktivními prvky v závislosti na použitých materiálech a typu přenosového média. Standard 802.3 definuje mnoho norem pro různé přenosové rychlosti, které definují rychlost přenosu, použitou signalizační metodu a použité přenosové médium. [1]

1.1.1 Varianty Ethernetu z pohledu použitého přenosového média

Názvy variant Ethernetu mají složenou strukturu podle třech základních vlastností. Číslice na začátku názvu určuje rychlost přenosu, následuje klíčové slovo určující signalizační metodu, za kterou je poslední část názvu, která určuje použité přenosové médium.

Dnes nejpoužívanější dvě skupiny variant jsou 100BASE-X pro 100MBit/s přenos a 1000BASE-X pro tzv. gigabitový přenos.

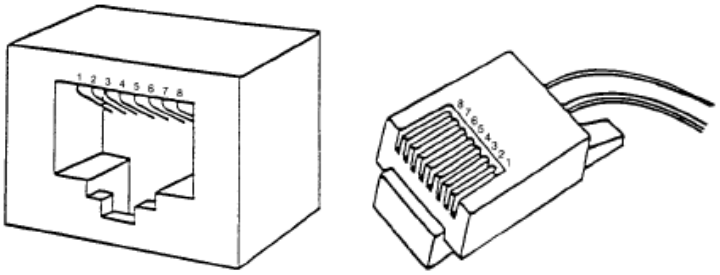
- **100BASE-X** – Jak už název napovídá, jedná se o standardy fyzické vrstvy dosahující rychlostí až 100Mb/s a používající jako metodu pro řízení přístupů k přenosovému mediu CSMA/CD. Současně přední specifikace jsou dvě: 100BASE-TX a 100BASE-FX. Specifikace 100BASE-TX definuje přenos přes dva páry kroucené dvojlinky kategorie 5 s maximální délkou segmentu 100 metrů. Specifikace 100BASE-FX definuje přenos

pomocí dvou optických vláken, dlouhé maximálně 412 metrů pro vícevláknové vodiče a až 10 kilometrů pro jedno-vláknový vodič. [2]

- **1000BASE-X** – Standardy fyzické vrstvy v třídě 1000BASE-X dosahují rychlosti až 1Gb/s s metodou přístupu k přenosovému médiu CSMA/CD. Byly navržené převážně pro optická vlákna. Délka segmentu je určena vlnovou délkou světla a tloušťkou vlákna od 220 metrů (1000BASE-SX) až po 5 kilometrů (1000BASE-LX). [3]

1.1.2 Způsoby zapojení kroucené dvojlinky

Rozhraní závislé na médiu (MDI) u kroucené dvojlinky popisuje jak elektrickou tak fyzickou část přenosu. Standard specifikuje pořadí žil v konektoru a ve zdířce jak zobrazuje obrázek 5.2 a jaké signály vede která žíla.



Obrázek 1.1: Pořadí žil v konektoru a ve zdířce. Zdroj: [1].

Rozlišuje se na dva typy. Prvním je MDI kde je nutné použít křížený síťový kabel a MDI-X, kde je křížení integrováno do jedné ze zdířek a je potřeba použít kabel přímý. Dnešní zdířky obsahují automatické rozpoznávání typu křížení a podle něj se chovají jako MDI nebo MDI-X.

Signály při využití pro přenos dvou páru kroucené dvojlinky se zabudovaným křížením v přípojce, tedy MDI-X, jsou znázorněny v tabulce 1.1, kde TD je označení páru pro odesílání a jako RD je označen příjímací pár.

Žíla	**MDI signál**
1	TD+
2	TD-
3	RD+
4	nepoužité
5	nepoužité
6	RD-
7	nepoužité
8	nepoužité

Tabulka 1.1: Signály na jednotlivých žílách. Zdroj: [1].

1.2 Síťová synchronizace času

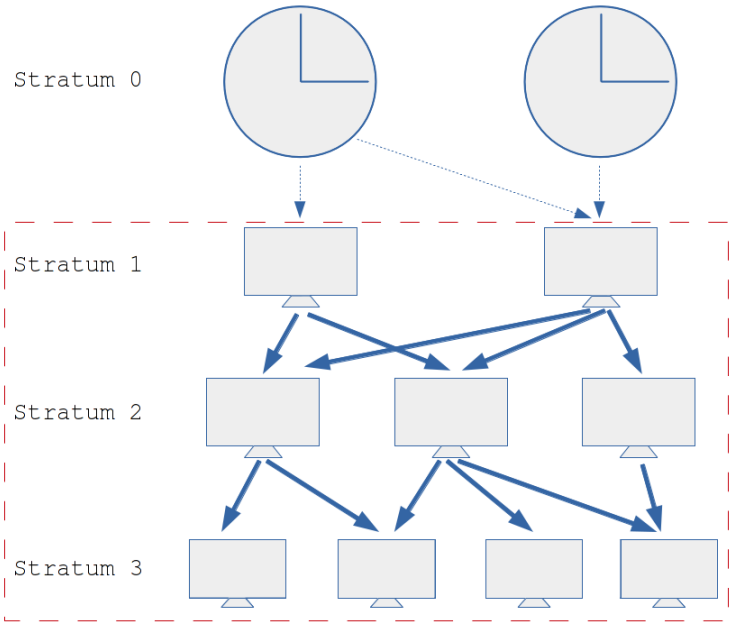
1.2.1 Network time protocol

Network Time Protocol (NTP) je široce využívaný protokol pro synchronizaci hodin počítačů v internetu. Definuje komplexní mechanismy pro přístup k informaci o času, vytváření NTP podsítí klientů a serverů a nastavování jejich systémových hodin.

NTP implementace funguje ve třech režimech, jako primární server nebo sekundární server nebo jako klient. Primární server se synchronizuje s referenčními hodiny přímo spojenými k UTC (GPS, Galilelo, atd.). Klient se synchronizuje pomocí jednoho nebo více nadřazených serverů, ale neposkytuje synchronizaci dalším klientům. Sekundární server má jednoho nebo více nadřazených serverů a také jednoho nebo více podřazených serverů nebo rovnou klientů[4]. Obsáhleji jsou tyto režimy popsané v následující kapitole 1.2.1.

Hierarchie

Struktura topologie NTP sítě je stromová. Kořen stromu je vždy primární server, který poskytuje co nejpřesnější UTC čas. Další patra stromu jsou tvořeny sekundárními servery. Listy stromu jsou klienti. Úroveň pater stromu se udáva v tzv. stratum number. Kořen stromu (primární server) má stratum number rovno 1, každá další nižší úroveň má vždy o jedno větší stratum number než předchozí úroveň. Referenční zdroj, ze kterého bere primární server přesný čas má stratum 0. Příklad rozestavění topologie NTP sítě a její hierarchie je demonstrována na obrázku 1.2. [4]



Obrázek 1.2: Ukázka NTP hierarchie.

Režimy protokolu

K dispozici jsou tři režimy, ve kterých NTP může běžet: symetrický, klient/server a broadcast. Každý režim má vlastní typ vazby a role komunikujících prvků. Každá vazba má definované hodnoty, kterými se označují komunikující prvky při komunikaci. V tabulce 1.2 jsou tyhle hodnoty uvedeny.

Typ vazby	Označení vazby	Hodnota paketu
Symetrická aktivní	1	1 nebo 2
Symetrická pasivní	2	1
Klient	3	4
Server	4	3
Broadcast server	5	5
Broadcast klient	6	nedefinována

Tabulka 1.2: Hodnoty vazeb režimů a paketů NTP. Zdroj [4].

- **Klient/server režim** – server poskytuje synchronizaci jednomu nebo více klientům, ale nepřijímá synchronizaci od nich. Server se může synchronizovat pomocí referenčních hodin připojenými přímo k UTC nebo jinému zdroji garantujícímu přesný čas.
- **Symetrický režim** – komunikující prvek je zároveň jak server tak klient. Komunikující stanice si synchronizují čas navzájem od sebe. Jsou dvě varianty: aktivní a pasivní. Varianty se odsebe odlišují tím, že při aktivní vazbě je spojení trvalé zato při pasivní vazbě časově omezené.
- **Broadcast režim** – server periodicky posílá broadcast pakety, které můžou být přijaté více klientmi. Režim je jinak v zásadě stejný, jako při režimu klient/server.

1.2.2 Simple network time protocol

Simple Network Time Protocol (SNTP) je podmnožinou Network Time Protokolu, používajícího se k synchronizaci hodin zařízení v Internetu. SNTP muže být použit, když není nutné využívat plné vlastnosti NTP. SNTP neimplementuje minimalizaci odchylky způsobenou rozptylem a zpožděním paketů a také klient nemůže sloužit i jako server [4]. SNTP se používá výhradně ve vestavěných systémech a pro tuhle potřebu byl navržený.

1.2.3 Daytime protocol

Daytime protocol specifikovaný v RFC 867[5] definuje způsob, jak jednoduše poskytnout informaci o aktuálním čase a datu. Informaci odesílá ve formě ASCII znaků. Protokol se využívá hlavně v případech, kdy se informace použije jako textový výstup.

Daytime komunikace mezi klientem a serverem může probíhat buď na TCP portu 13 nebo na UDP portu 13. Po připojení klienta k serveru, server okamžitě, bez ohledu na vstupu od klienta, odešle informaci o čase a ukončí spojení.

Syntaxe časové informace není pevně určená. Je doporučeno, aby byla limitována na rozsah ASCII tabulky a měla by být maximální velikosti osmdesáti znaků. RFC 867 nabízí

jako jednu z možných syntaxí: Den v týdnu, Měsíc Den, Rok Čas-Časová zóna. Příklad: Středa, Květen 20, 2015 07:00:00-GMT+2.

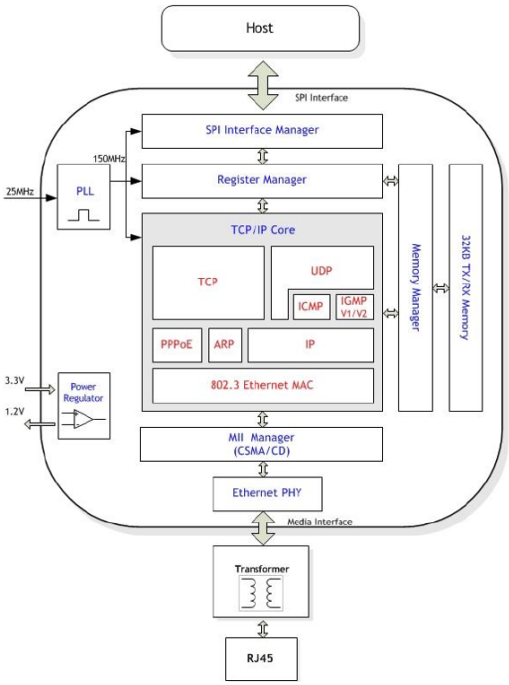
1.2.4 Time protocol

Time protocol specifikovaný v RFC 868[6] definuje způsob, jak jednoduše poskytnout informaci o aktuálním čase a datu, kterou je poté možno strojově zpracovat. Informace je reprezentovaná 32 bitovou hodnotou, která udává uplynulý počet sekund od referenčního data 00:00 1. Ledna 1900 GMT. Tento datum bude referenční do roku 2036, poté 32bitové pole přeteče.

Time server naslouchá buď na TCP portu 37 nebo na UDP portu 37. Po připojení na TCP port 37 okamžitě odešle time server 32bitovou informaci o čase klientovi. Po úspěšné komunikaci obě strany spojení ukončí. Při UDP komunikaci server naslouchá na UDP portu 37. Klient odešle prázdný datagram. Server ho přijme a obratem odešle 32 bitovou informaci o čase.

1.3 Hardwarový TCP Stack Wiznet 5500

Wiznet 5500 (w5500) je hardwarový TCP/IP čip k použití do vestavěných systémů pro připojení do internetu. W5500 podporuje TCP, UDP, IPv4, ICMP, ARP, IGMP a PPPoE. Podporuje 10BASE-X nebo 100BASE-X. Umožňuje naráz spravovat až 8 soketů. Obsahuje 32kB paměť která se rozděluje mezi 8 soketů, tzn. na jeden soket 4kB a tj. 2kB pro odesílané a 2kB pro přijímané data. Čip komunikuje s obslužným mikroprocesorem pomocí rozhraní SPI 1.3.2 kde vždy figuruje jako slave. Pro tuto kapitolu byl použitý zdroj [7].



Obrázek 1.3: Blokové schéma w5500. Zdroj: [7].

1.3.1 Referenční schéma pro zapojení Wiznet 5500

Referenční schéma pro zapojení w5500 s galvanickým odizolovaním v rj45 je přiloženo v Příloze A.

1.3.2 Sériové periferní rozhraní

W5500 posktuje SPI (Serial Peripheral Interface) Bus Interface s 4 signály (SCSn, SCLK, MOSI, MISO) jako interface pro komunikaci s jiným zařízením. Wiznet 5500 operuje přes SPI jako slave.

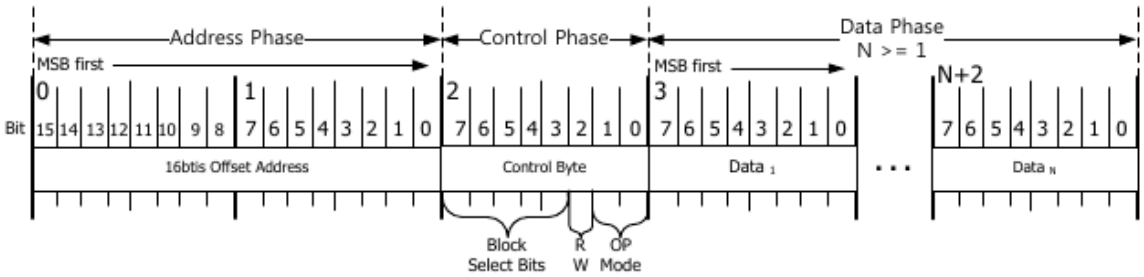
SPI protokol definuje čtyři řídící módy. Každý mód se liší podle polarity a fáze SCLK signálu. W5500 poskytuje SPI mód 0 a mód 3. Oba MOSI a MISO signaly používají přenosnou sekvenci od MSB do LSB když MOSI signal vysílá a MISO signal přijíma. MOSI a MISO signaly vždy vysílají nebo přijímají v sekvenci od MSB do LSB.

1.3.3 Přerušení

W5500 také poskytuje přerušení pro předem definovanou událost. Pro tento účel slouží signál INTn který při nastaní přerušení klesne na nízkou logickou úroveň. Při situaci kdy není přerušení je na signál na vysoké logické úrovni.

1.3.4 Komunikační rámec SPI

SPI rámec se skládá ze tří částí jak ukazuje obrázek 1.4.



Obrázek 1.4: Struktura komunikačního SPI rámce. Zdroj: [7].

- **Adresová část** – na obrázku 1.4 jako Address Phase, je složen ze dvou bajtů a adresuje instrukce w5500.
- **Kontrolní část** – na obrázku 1.4 jako Control Phase, jeden řídící baj ve kterém prvních 5 bitů (bity 7-3) udávají, jestli se bude pracovat s běžnými registry nebo s kterým soketem a jakou jeho částí, tzn. jestli s registry soketu, nebo s jeho bufferem na odesílání dat nebo s bufferem pro přijímání dat. Bit číslo 2 určuje jestli se bude číst z w5500 nebo do něj zapisovat. Poslední dva bity (1-0) udávají o jaký mód komunikace půjde, jestli s předem neznámou délkou dat a nebo fixní o velikostech 1 bajt, 2 bajty nebo 4 bajty.
- **Datová část** – na obrázku 1.4 jako Data Phase je samotný proud datové informace.

1.3.5 Ovládání a konfigurace Wiznetu 5500

Kooperace s w5500 probíhá pomocí nastavování registrů a práce s nimi.

W5500 má dvě skupiny registrů. První skupina slouží pro globální konfiguraci w5500, druhá skupina pro konfiguraci a ovládání síťových soketů.

Obecné registry

- **MR - Mode** – spuštění softwarového resetu, povolení odezvy na ping, povolení Wake on LAN, vynuceného ARP módu nebo nastavení PPPoE módu.
- **GAR - Gatway Address** – pro nastavení IP adresy výstupní brány. Defaulně 192.168.0.1.
- **SUBR - Subnet Mask Address** – pro nastavení masky sítě. Defaultně 255.255.255.0.
- **SHAR - Source Hardware Address** – pro nastavení MAC adresy zařízení. Defaultně 00:08:DC:01:02:03.
- **SIPR - Source IP Address** – pro nastavení IP adresy zařízení. Defaultně 192.168.0.2.
- **IR - Interrupt** – signalizuje stav přerušení, které je způsobeno konfliktem IP adres, nedosažitelným síťovým cílem, PPPoE ukončením spojení nebo příchodem magického paketu pro Wake on Lan. Každý bit z IR lze smazat zapsáním hodnoty '1' na jeho pozici. Když IR registr není roven '0x00', INTn PIN je nastaven na nízkou úroveň, dokud se tak nestane.
- **IMR - Interrupt Mask** – povoluje (hodnota '1') nebo zakazuje (hodnota '0') přerušení na příslušném bitu. Souvisí s registrem IR, kde bit v IMR povoluje přerušení na bitu v IR. Defaultně jsou zakázány všechny přerušení.
- **SIR - Socket Interrupt** – signalizuje přerušení od jednotlivých soketů signalizované hodnotou '1'. Jeden bit v registru reprezentuje jeden soket. Pokud SIR registr není roven '0x00', INTn pin je nastaven na nízkou úroveň, dokud se tak nastane. Nulování registru je za pomocí registru Sn_IR, v skupině registru pro konkrétní soket.
- **SIMR - Socket Interrupt Mask** – povoluje (hodnota '1') nebo zakazuje (hodnota '0') přerušení od jednotlivých soketu. Souvisí s registrem SIR, kde příslušný bit v SIMR povoluje přerušení na příslušném bitu v IR. Defaultně jsou zakázány všechny přerušení.

Registry soketů

- **Sn_MR Socket n Mode** – první polovina bajtu povoluje/zakazuje multicasting, blokování broadcastu, nezpožďování ACK a blokování unicastu, druhá polovina určuje variací čtyř bitů protokol - TCP, UDP, MACRAW nebo soket uzavřen.
- **Sn_CR Socket n Command** – slouží pro ovládání soketu, nastavení jeho nynějšího statusu, tzn. OPEN, LISTEN, CONNECT odeslat ACK paket, DISCON odeslat FIN/ACK paket, CLOSE zavřít soket bez odeslání FIN/ACK paket, SEND odeslat obsah TX zásobníku, SEND_MAC odeslat obsah TX zásobníku bez automatického ARP procesu, SEND_KEEP odeslat keep-alive paket, RECV čekáni na příchozí data.

- **Sn_IR Socket n Interrupt** – signalizuje stav přerušení jako je spojení navázáno, ukončeno, data přijaty, data odeslány a timeout. Každý bit z IR lze smazat zapsáním hodnoty '1' na jeho pozici. Když registr není roven '0x00' je nastaven příslušný bit v registru SIR na hodnotu '1' a tím INTn pin se nastaví na nízkou logickou hodnotu.
- **Sn_SR Socket n Status** – signalizuje stav soketu, změní se při dokonání činnosti nastavené pomoci registru Sn_CR nebo při příchozích paketech ACK, SYN, FIN v TCP. Stavy soketu můžou být v kontextu s předchozími registry například SOCK_CLOSED, SOCK_INT, SOCK_LISTEN, SOCK_ESTABLISHED, SOCK_CLOSE_WAIT nebo v neposlední řadě SOCK_MACRAW. Dále jsou stavy nastávající při měnění stavu SOCK_SYNSENT, SOCK_SYNRECV, SOCK_FIN_WAIT, SOCK_CLOSING nebo SOCK_TIME_WAIT a SOCK_LAST_ACK.
- **Sn_PORT Socket n Source Port** – nastavuje zdrojový port zařízení pro komunikaci.
- **Sn_DHAR Socket n Destination Hardware Address** – nastavuje cílovou MAC adresu.
- **Sn_DIPR Socket n Destination IP Address** – nastavuje cílovou IP adresu.
- **Sn_DPORT Socket n Destination Port** – nastavuje cílový port.
- **Sn_IMR Socket n Interrupt Mask** – povoluje (hodnota '1') nebo zakazuje (hodnota '0') přerušení při jednotlivých případech komunikace. Souvisí s registrem Sn_IR, kde příslušný bit v Sn_IMR povoluje přerušení na příslušném bitu v Sn_IR. Defaultně jsou zakázány všechny přerušení.

Kapitola 2

Napájení zařízení pomocí Ethernetu

2.1 Power over Ethernet

Standard IEEE 802.3af, lidově nazývaný Power over Ethernet (PoE), byl navržený a v roce 2003 vydaný pro způsob napájení spotřebičů pomocí kroucené dvojlinky. Všechny informace popisované v této kapitole jsou shrnutím zdroje [8].

PoE rozděluje napájení zařízení na dvě komunikující strany - zdroj a spotřebič. Jako zdroj napětí je zde PSE (Power Sourcing Equipment), který řídí komunikaci se spotřebičem a na základě odpovědí posílá určitý rozsah napětí a proudu spotřebiči. PD (Powered Device) je zde spotřebič a má za úkol definovat sám sebe, správně reagovat na dotazy od PSE, a také rozhodnout o puštění nebo nepuštění napětí a proudu do spotřebiče.

2.1.1 Varianty přenosu elektrické energie podle Ethernetového standardu

Standard IEEE 802.3af popisuje varianty napájení spotřebiče přes kroucenou dvojlinku jak pro 10Mbit/100Mbit tak pro 1Gbit.

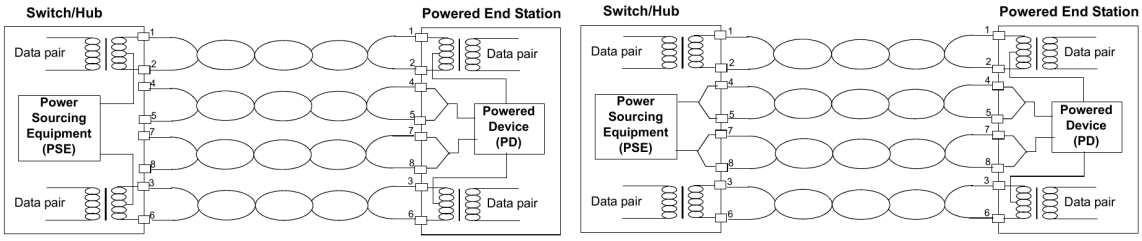
Jsou dva způsoby, jak napájení vložit do vodičů. Prvním způsobem je, že PSE je integrováno přímo do switche/hubu. Druhý způsob popisuje alternativu, kdy switch/hub nepodporuje standard IEEE 802.3af a PSE se vloží mezi switch/hub a spotřebič.

Každý z těchto způsobů má dvě alternativy, které se liší v tom, po kterých vodičích se napájecí napětí vede. Alternativa A definuje způsob vedení napětí a proudu pomocí vodičů 1,2 a 3,6, alternativa B zase vedení pomocí vodičů 4,5 a 7,8, které jsou v určitých variantách 10Mbit a 100Mbit sítích nevyužité.

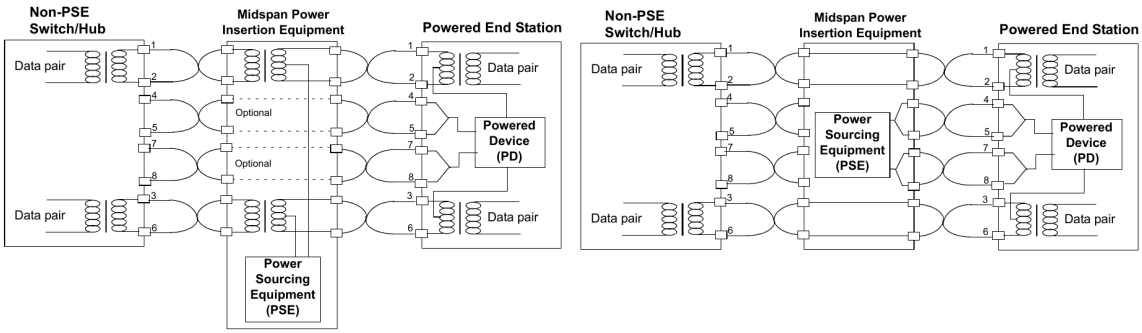
PoE u 10Mbit/100Mbit sítí

Na obrázcích 2.1 je znázorněno napájení spotřebiče způsobem, kdy je PSE integrováno přímo do switche/hubu. Levým obvod popisuje alternativu A, která vede napájení přes datové vodiče a pravý alternativu B která vede napájení pomocí nevyužitých páru.

Způsob napájení spotřebiče, když switch/hub nepodporuje standard IEEE 802.3af, je podle standartu znázorněn obrázkem 2.3. Při téhle variantě je nutné přidat mezi aktivní prvek a spotřebič externí zdroj napětí a proudu, který standard podporuje.



Obrázek 2.1: Alternativa A a B, se zabudovaným PSE v switchi/hubu. Zdroj: [2].



Obrázek 2.2: Alternativa A a B, s použitím externího PSE. Zdroj: [2].

PoE u 1Gbit sítí

U 1Gbit sítí je způsob napájení spotřebiče totožný se způsobem napájení při 10Mbit/100Mbit sítí. Rozdíl je v tom, že když se uvažuje alternativa B, tak se napájení posílá také pro datových vodičích, protože v 1Gbit sítí jsou využité všechny žíly pro přenos dat, tzn. vodiče 4,5 a 7,8 nejsou volné.

2.1.2 Protokol pro komunikaci mezi zdrojem a spotřebičem

Komunikace probíhá na základě nastavování rozsahu napětí na PSE a následné odezvě od PD. PoE definuje 5 stavů, ve kterých může komunikace mezi PD a PSE probíhat.

Nejprve začíná detekce toho, jestli PD podporuje standard IEEE 802.3af, poté se určuje výkonnostní třída napájeného zařízení. Následuje aktivace napájeného zařízení pomocí PSE který je po ustálení odebíraných hodnot následován cíleným stavem, kdy je koncové zařízení plně napájeno. Pátým stavem je stav, kdy PSE detekuje, že na druhé straně je zařízení, které nepodporuje IEEE 802.3.af. V tomhle režimu je PSE v tzv. klidovém režimu.

Hodnoty napětí na zdroji při komunikaci jsou shrnuty v tabulce 2.1.

Způsob komunikace pomocí napěťových stavů nahrazuje požití jiného, jinak nutného komunikačního kanálu.

Detekce podpory PoE na PD

V případe že napájené zařízení podporuje standard IEEE 802.3af, tak má na vstupu podle standardu odpor 25kΩ. Pro zjištění tohoto odporu PSE posílá na PD stejnosměrné napětí v rozsahu 2.8 - 10 V. Detekce probíhá nezávisle na polaritě napájecího napětí. Po

Režim	Napětí [V]
Detekce standardu na PD	2.8-10
Identifikace třídy	15.5-20.5
Aktivace PD	30-44
Plné napájení	44-57
Klidový stav	0-2.8

Tabulka 2.1: Napětí ze zdroje při komunikaci. Zdroj [8].

zjištění přítomnosti 25kΩ odporu, začíná fáze druhá - určení výkonnostní třídy napájeného zařízení.

Určení výkonnostní třídy PD

Standard IEEE 802.3af definuje 4 výkonnostní třídy spotřebičů, které zobrazuje tabulka 2.2. Navzájem se liší velikostí výkonů, které je PSE schopen dodávat. Jsou čtyři stupně napájení, kde první až třetí určuje napájecí výkon při plném napájení PD. Nízký výkon o 4.0W, střední výkon 7.0W a 15.42W jako vysoký, respektive plný výkon. Z nichž je ještě třída nula, která je třídou defaultní. Používá se, když nelze identifikovat třídu spotřebiče.

Identifikace výkonnostní třídy se provádí při vytvoření napětí na zdroji v rozsahu 15,5 - 20 V. Při tom se měří odebíraný proud spotřebičem ze zdroje a rozhoduje se podle jeho velikosti, do jaké třídy PD spadá. Pokud je proud menší než 0.4 mA, tzn spotřebič prakticky neodebírá žádný proud, pak se nastaví defaultní výkonnostní třída 0. Pokud je odebíraný proud vetší než 44mA pak se spotřebič nepovažuje za navržený podle standardu IEEE 802.3af a PSE nebude PD napájet. Rozsah mezi 0.4mA do 44mA odebíraného proudu se rozděluje podle úrovní popsaných v tabulce 2.2.

Třída	Odebíraný proud [mA]	Příkon spotřebiče [W]	Napětí z PSE [V]
0	0-4	0.44-12.94	15,5-20
1	9-12	0.44-3.84	15,5-20
2	17-20	3.84-6.49	15,5-20
3	26-30	6.49-12.95	15,5-20

Tabulka 2.2: Výkonostní třídy. Zdroj [8].

Aktivace PD a plné napájení

Po určení výkonnostní třídy může začít plné napájení spotřebiče. Přechod do tohoto stavu má předfázi, kdy se nastaví napětí ze zdroje na na velikost v rozsahu 30-44 V a tím se dá spotřebiči najevo, že se zdroj chystá plně napájet PD. Tento přechod, kdy PD začne okamžitě odebírat proud a tím skokově poklesne napětí, může být vyhodnocený na straně PSE jako zkrat, který se ošetřuje odpojením napájecího napětí a opakovaní komunikace od bodu detekce. Tohle chování je nežádoucí a ošetřuje se právě stavem pro aktivaci PD, kde

je dovolený pokles napětí, který když se po určité době ustálí na dovolené hranici, nebude vyhodnocený jako zkrat.

Při plném napájení spotřebiče PSE dodává do sítě 44-57 V. Proud nesmí klesnout pod 10 mA, jinak PSE vyhodnotí tento pokles jako odpojení spotřebiče a přestane napájet spotřebič.

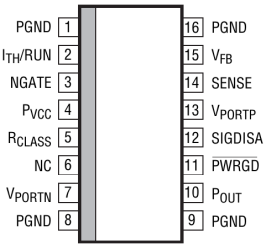
Klidový režim

Posledním stavem, ve kterém se může komunikace nacházet, je stav klidu, který nastává když PSE nedetekuje na druhé straně žádný síťový prvek, který by podporoval standard IEEE 802.3af. V této fázi zdroj dodává do obvodu pouze zbytkové napětí o velikosti 0-2,8 V.

2.2 Interface pro Power over Ethernet IEEE 802.3af na straně spotřebiče LTC4267

LTC4267 je hardwarová implementace strany spotřebiče pro napájení přes Ethernet podle standardu IEEE 802.3af. Obsahuje v sobě ověřovací 25kΩ odpor pro ověření že cílové zařízení PoE podporuje, senzor pro klasifikaci zdroje a také pojistku proti přehřátí čipu. Pro tuto kapitolu jsem čerpal z datasheetu pro LTC4267 [9].

2.2.1 Popis pinů LTC4267



Obrázek 2.3: LTC4267 v pouzdře CNG a IGN. Zdroj: [9].

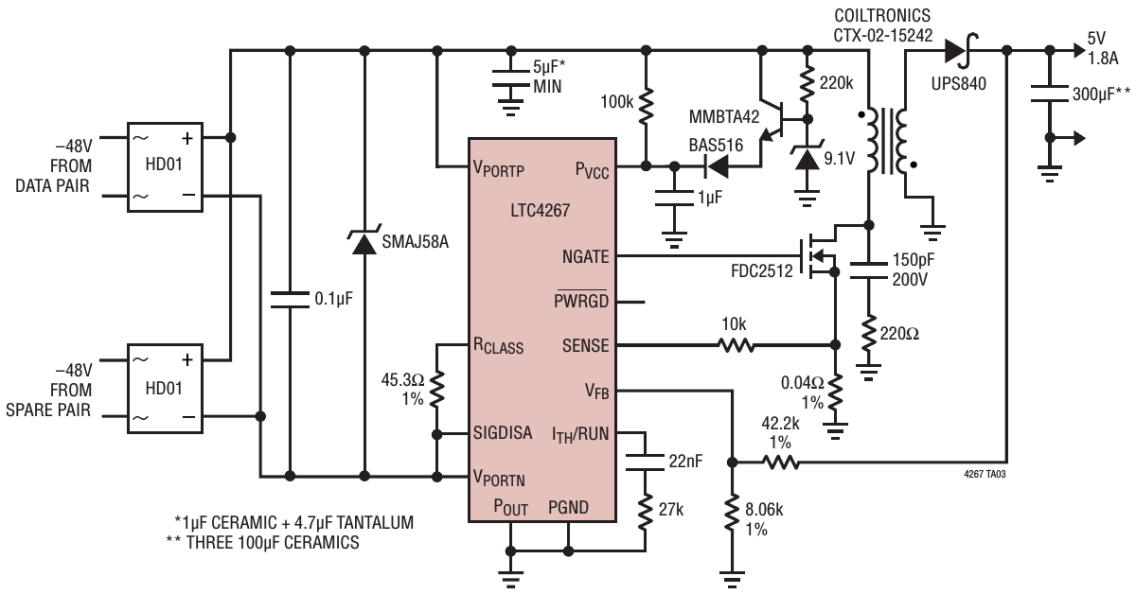
- **VPORTP** – vstup +48V napětí do LTC4267 z PSE.
- **VPORTN** – vstup -48V napětí do LTC4267 z PSE.
- **PVCC** – kladný výstup z LTC4267.
- **POUT** – záporný výstup z LTC4267, vyvádí -48V na PGND pin napěťového regulátoru.
- **PGND** – uzemnění LTC4267
- **RCLASS** – vstup pro určení výkonnostní třídy. Ta se určuje odporem zapojeným mezi RCLASS a VPORTN.
- **PWRGD** – signalizuje nízkou impedanci, že napájecí proud je v pořádku. Při detekci PD, určování výkonnostní třídy nebo při přehraní čipu je na tomto pinu vysoká impedance.

- **SIGDISA** – umožňuje předložit PSE jinou velikost ověřovacího odporu který určuje, jestli dané zařízení podporuje PoE a tím vyřadit celý PD obvod pro IEEE 802.3af. Pokud není tento pin použitý, připojí se na V_{PORTN}.
- **I_{TH}/RUN** – zastává dvě funkce. Slouží jako kompenzační zesilovač regulátoru napětí a také jako spouštěč/vypínač řídícího vstupu. Když hodnota napětí klesne pod 0,28V vůči PGND, způsobí vypnutí kontroléru.
- **SENSE** – monitoruje proud a napětí na výstupu napěťového regulátoru.
- **V_{FB}** – pomocí tohoto pinu získává LTC4267 zpětnou vazbu z výstupu transformátoru přes dělič napětí, který ovlivňuje velikost kompenzace.
- **NGATE** – je výstupní řídící brána, která reguluje průtok proudu regulačním MOSFET tranzistorem a tím i výslednou velikost napětí na výstupu.
- **NC** – nevyužité piny.

2.2.2 Nastavení výstupního napětí

Výstupní napětí do spotřebiče se určuje děličem napětí zapojeným mezi výstup transformátoru a pinem V_{FB} viz. 2.4. Poměr odporů potřebný k žádoucímu napětí na výstupu se vypočítává pomocí vzorce $R2 = R1 \cdot \frac{V_{OUT}-V_{REF}}{V_{REF}}$ kde V_{REF} je u neizolovaného zdroje napětí v LTC4267 0,8V.

2.2.3 Referenční schéma zapojení pro 5V na výstupu



Obrázek 2.4: Neizolované referenční schéma zapojení pro 5V na výstupu. Zdroj: [9].

Kapitola 3

Časoměrný panel a mikroprocesor MC9S08LG16

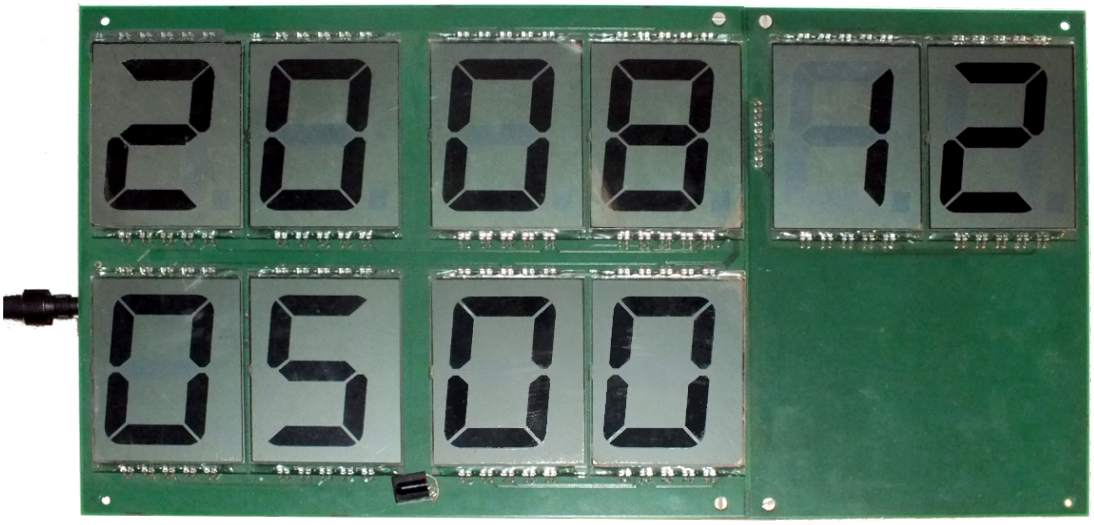
Tato kapitola popisuje digitální panel s hodinami a stopkami, který byl navržený a vyrobený pro stopování času při obhajobách projektů na VUT FIT.

Dále kapitola popisuje panel ovládající mikroprocesor MC9S08LG16 z mikroprocesorové rodiny HCS08.

3.1 Panel s digitálními hodinami a stopkami

Panel s digitálními hodinami a stopkami 3.1 je navržený a vyrobený pro zobrazování hodin a stopování času. Jak hodiny, tak i stopky lze nastavovat a ovládat dálkovým ovládačem.

Na hodinách je připravený vývod se sběrnicí SPI a s napájením 3.3V, který byl připraven pro rozšíření panelu o ethernetový modul. Ethernetový modul bude panelu umožňovat pro hodiny načítat přesný čas z internetu a dále se přes něj budou pomocí počítače ovládat stopky.



Obrázek 3.1: Časoměrný panel s digitálními hodinami a stopkami.

3.1.1 Hardwarové prvky panelu

Hodiny byly vytvořeny studenty v rámci projektu v předmětu Návrh externích adaptérů a vestavěných systémů. Hodiny navrhli a sestavili z komponentů popsaných níže.

Mikroprocesor

Modul obsahuje MCU Freescale MC9S08LG16, který řídí jak časomíru tak posléze síťovou komunikaci. Obsahuje vestavěný LCD řadič, který umožňuje řídit až 8x29 segmentů u na panelu použitého 64 vývodového pouzdra.

LCD panely

Na desce se nachází deset LCD displayů. Pro zobrazování aktuálního času je na desce v horní polovině vyhrazeno šest LCD displayů v časovém schématu hh:mm:ss. V dolní polovině jsou vyhrazené čtyři LCD displaye v časovém schématu mm:ss pro zobrazení stopovaného času.

Piezoelektrický bzučák

Pro signalizaci konce časování je na panelu zaveden piezoelektrický bzučák PT-2040P-PQ. Využívá se také pro zvukové efekty jako zpětnou vazbu při nastavovaní hodin nebo časomíry.

Infračervený přijímač

Pro zpracování signálu z dálkového ovladače, je v panelu zabudován infračervený přijímač TSOP31236. Ten na výstup generuje již demodulovaný a vyfiltrovaný digitální signál, který poté zpracovává mikroprocesor.

Napájení obvodu

Pro možnosti napájení z elektrické sítě je v panelu zabudován souosý 2,1mm konektor. Obod vyžaduje velikost napětí 5V.

Konektor pro programování mikroprocesoru

Pro programování mikroprocesoru je deska osazena standardizovaným šestí pinovým BDM konektorem.

Ethernet

Při návrhu hodin se počítalo s použitím ethernetového modulu pro FITkit, který je popsaný v [10], pro možnost připojení k internetové síti. Hodiny jsou osazeny sběrnicí SPI s dvouřadým dvaceti pinovým pinheader konektorem. O příchozích datech informuje mikroprocesor pomocí přerušení IRQ. Tento modul je napájen 3.3V, který konvertuje z 5V integrovaný stabilizátor napětí MC33269D-3.3.

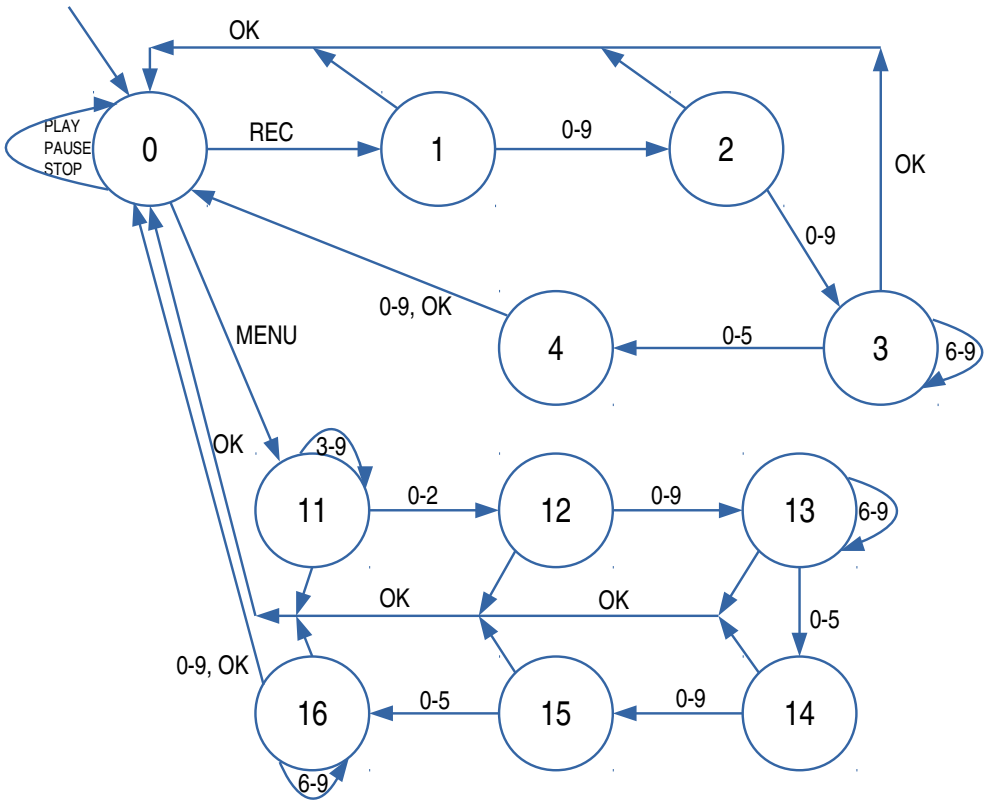
3.1.2 Ovládání pomocí dálkového ovládače

Ovládání hodin je možné pomocí programovatelného infračerveného dálkového ovládače nastaveného na kód 0607.

- **Nastavování hodin** – se z výchozího stavu aktivuje stisknutím tlačítka MENU na dálkovém ovladači. Tím se aktivuje horních šest LCD displayů sloužících pro zobrazení aktuálního času. Vybíráním jednotlivých číslic na ovladači se postupně nastavují hodiny, minuty a vteřiny.
- **Nastavování stopek** – se aktivuje stisknutím tlačítka REC z výchozího stavu. Tím se aktivují dolní čtyři LCD displaye, sloužící pro zobrazení časomíry. Po zadání konkrétního času se zobrazí čas ve stopnutém módu. Odpočet řídíme pomocí tlačítek PLAY, PAUSE a STOP.

Pro přechod mezi jednotlivými LCD display při nastavování času je možné použít šipky doprava a doleva. Z nastavování lze odejít po stisknutí tlačítek PLAY, STOP nebo MENU. Poté se vrátí k původnímu času, který byl na hodinách/stopkách před nastavováním.

Konečný automat pro ovládání hodin dálkovým ovladačem

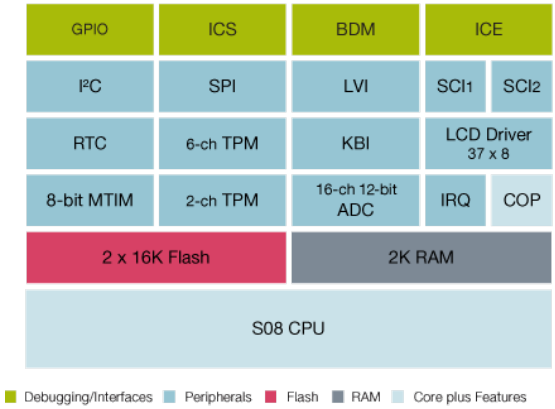


Obrázek 3.2: Stavový automat tlačítek ovládače.

3.2 Mikroprocesor Freescale MC9S08LG16

Osmibitový mikroprocesor použitý v časoměrném panelu je založený na jádře rodiny HCS08. Jádro obsahuje komplexní instrukční sadu CISC upravenou pro dosáhnutí vetší efektivity programů zkompilovaných z jazyka C. Mikroprocesor je Von Neumannovy architektury. Informace pro tuto podkapitolu jsem získal z referenčního manuálu pro mikroprocesor [11].

Pro tento projekt byl vybraný hlavně kvůli možnosti připojit k němu až 37 LCD displayů s osmi segmenty.



Obrázek 3.3: Architektura mikroprocesoru MC9S08LG32. Zdroj [12].

3.2.1 Technické specifikace

- **Frekvence CPU** – až 40 MHz.
- **Achitektura** – 8-bitová s Von Neuman rozložení paměti.
- **RAM** – 1984B.
- **Flash paměť** – Flash A - 2048B, Flash B - 16384B.
- **Instrukční sada** – HCS08
- **LCD řadič** – možnost připojit a ovládat až 29 LCD displayů s osmi segmenty.
- **Počet vývodů pouzdra** – 64

3.2.2 Popis částí mikroprocesoru MC9S08LG16

- **AD převodník a komparátor** – 12 kanálů, automatická porovnávací funkce, 2,5μ
- **LCD řadič** – až 37x8 nebo 41x4 segmentů
- **SCI** – obousměrné asynchronní sériové komunikační rozhraní.
- **SPI** – obousměrné synchronní sériové komunikační rozhraní.
- **IIC** – sériová sběrnice s maximální 100kbps propustností, podporuje broadcast mód a 10-bitovou adresaci.

- **TPMx** – jeden šesti-kanálový a jeden dvou-kanálový čítač/časovač
- **MTIM** – 8-bitový časovač, čtyři časové zdroje
- **RTC** – 8-bitové hodiny reálného času s binární nebo dekadickou násobičkou. Tři interní zdroje času včetně možnosti externího zdroje.
- **KBI** – řízení ovládacích kláves podporující 8x8 klávesovou matici.
- **IRQ** – jeden pin pro přerušení od externího zařízení.

3.2.3 Přerušení IRQ

IRQ pin slouží pro možnost zachycení upozornění externího zařízení na událost která se v externím zařízení udála.

U mikroprocesoru MC9S08LG16 má procesor pro tenhle možnost vyveden jeden pin na který je externí zařízení možno připojit.

3.2.4 Rozdělení paměti

- **DIRECT PAGE REGISTERS**

  Registry pro ovládání modulů mikroprocesoru.
- **RAM**

  Paměti RAM o velikosti 1984B.
- **LCD Registers**

  Registry ovládající LCD displaye.
- **UNIMPLEMENTED 4000B**

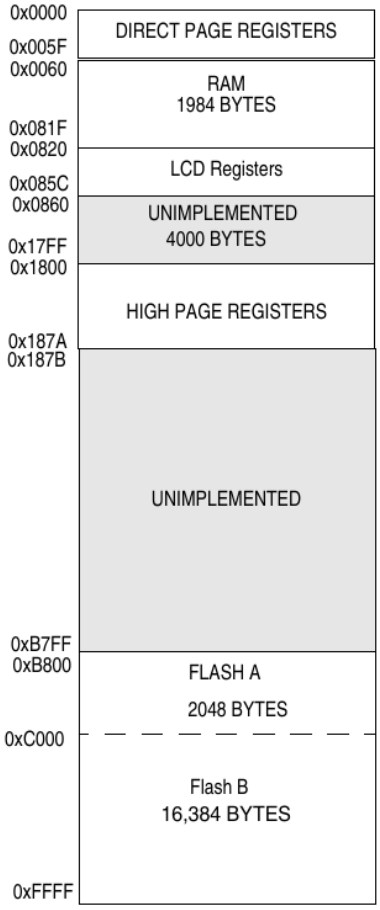
  Nevyužitý adresovací prostor.
- **HIGHT PAGE REGISTERS**

  Méně často používané registry, ponechávají více místa v "Direct page registers"pro častěji
- **UNIMPLEMENTED**

  Nevyužitý adresovací prostor.
- **FLASH A 2048B**

  Paměť konstantních dat a programu.
- **FLASH B 16384B**

  Paměť konstantních dat a programu.



Obrázek 3.4: Průřez pamětí. Zdroj: [11].

Kapitola 4

Návrh řešení

V této kapitole bude popsán návrh schématu ethernetového modulu, návrh desky plošného spoje a výběr součástek pro jeho budoucí fyzickou realizaci.

Bude se také zaobírat návržením operačního programu pro mikroprocesor Freescale MC9S08LG16, který bude síťově synchronizovat čas a nastavovat jej hodinám na časoměrném panelu a také návrhem protokolu pro ovládání stopek na časoměrném panelu pomocí obslužné aplikace.

V poslední části představí návrh vzhledu obslužné aplikace.

4.1 Ethernetový modul

4.1.1 Výběr mikrokontrolérů pro Power over Ethernet a síťovou komunikaci

Power over Ethernet

Aby ethernetový modul dokázal napájet hodiny požadovaným napětím 5V a přitom splňoval specifikaci standardu IEEE 802.3af, rozhodl jsem se pro použití LTC4267 popsaného v kapitole 2.2.1. Ten je plnohodnotný napájecí interface pro IEEE 802.3af na straně spotřebiče, který zastane celou komunikaci se zdrojem napětí.

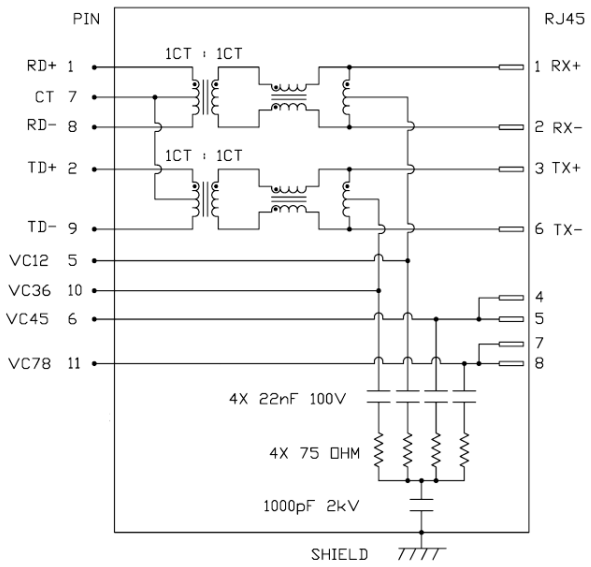
Dále je potřeba vybrat síťový konektor RJ45, který bude umožňovat vývod přivedeného napětí. Ten jsem vybral od firmy BEL FUSE typ MagJack 0813-1X1T-57-F. Jak je patrné z obrázku 4.1, konektor vyvádí přivedené fantomové střídavé napětí pomocí pinů VC12, VC36 a napětí přivedené pomocí volných datových vodičů piny VC45 a VC78.

Síťová komunikace

Při návrhu časoměrného panelu se počítalo s tím, že panel se rozšíří o ethernetový modul pro FITkit, který bude umožňovat synchronizaci času z internetu.

Po nastudování mikroprocesoru MC9S08LG16 popsaného v kapitole 3.2 použitého v časoměrném panelu a zvážení jeho možností, jsem došel k názoru, že kvůli malé paměti RAM by nezvládl plnohodnotnou implementaci TCP stacku.

Protože se ethernetový modul bude muset navrhnout a vyrobit nový kvůli rozšíření o Power over Ethernet, rozhodl jsem se, že se použije plnohodnotná hardwarová implementace TCP stacku, která celou síťovou komunikaci obsáhne.



Obrázek 4.1: Schéma zapojení MagJack 0813-1X1T-57-F. Zdroj [13].

Pro toto řešení jsem vybral hardwarový TCP stack Wiznet 5500. Síťová komunikace se poté bude moct ovládat mikroprocesorem MC9S08LG16, který ji ale nebude muset vytvářet od samotného základu, ale bude nastavovat sokety a pracovat s nimi.

4.1.2 Návrh schématu a desky plošného spoje ethernetového modulu

Pro návrh schématu ethernetového modulu a desky plošného spoje použiji program Altium Designer, který umožňuje kreslení schémat a z nich poté vygenerování a nakreslení desky plošného spoje. Dále umožňuje navrhnout si vlastní součástky a nakreslit jejich footprint, což je pro tuto práci důležitá vlastnost.

Obsahuje také spousty utilit pro ověření správnosti návrhu desky plošného spoje a možnosti jako je autorouting, který umožňuje automatické vygenerování cest mezi součástky.

4.2 Operační program mikroprocesoru MC9S08LG16

Mikroprocesor Freescale MC9S08LG16 je hlavní výpočetní jednotkou celého LCD panelu bude ovládat také ethernetový modul a jeho síťovou komunikaci. LCD panel budu programovat v jazyce C pomocí vývojového studia Freescale CodeWarrior IDE verze 5.9.0 který poskytuje i realtime debugger operačního programu spuštěného na mikroprocesoru.

4.2.1 Komunikace s Wiznetem 5500

Pro síťovou komunikaci slouží hardwarový TCP stack Wiznet 5500 (w5500), se kterým mikroprocesor MC9S08LG16 komunikuje pomocí sériového periferního rozhraní. V této komunikaci figuruje vždy w5500 jako slave a MC9S08LG16 jako master.

Když při síťové komunikaci nastane předem definovaná událost, oznámí to w5500 mikroprocesoru MC9S08LG16 pomocí IRQ signálu.

4.2.2 Synchronizace času

Rozhodl jsem se synchronizovat časovou informaci pomocí protokolu Time 1.2.4 kvůli jeho jednoduchosti a snadné dostupnosti.

Pro získání informace o času od Time serveru vytvořím TCP soket, který se připojí na server na port 37, který po připojení klienta odešle klientovi čtyři bajty s informací o čase. Po správném doručení, spojení uzavře jak server, tak mikroprocesor MC9S08LG16. Dále ze získané hodnoty mikroprocesor MC9S08LG16 vypočítá aktuální čas a nastaví ho hodinám.

4.2.3 Ovládání stopek pomocí počítače

Stopky na časoměrném panelu budou moct být ovládány pomocí počítače. K ovládání bude naprogramována obslužná aplikace která časoměrnému panelu bude odesílat potřebné informace, které časoměrný panel bude zpracovávat.

Návrh síťového protokolu pro ovládání stopek obslužnou aplikací

Protokol pro ovládání stopek bude muset obsáhnout odeslání a nastavení času stopkám, jeho spuštění nebo pozastavení a také jejich vynulování.

Server bude naslouchat na portu 4242 a čekat na příchozí spojení. Po jeho ustanovení přijme informaci o tom, co má provést. Danou událost vykoná a odešle přijatou informaci zpět obslužné aplikaci. Jak server tak obslužná aplikace spojení ukončí.

1. **Ustanovení spojení**
   (a) Server naslouchá na portu 4242.
   (b) Obslužná aplikace odešle SYN paket.
   (c) Server odešle SYN ACK paket.
   (d) Obslužná aplikace odešle ACK paket.
2. **Informace o činnosti**
   (a) Aplikace odešle informaci o události.
      - **TEST** - ASCII hodnotu '1'.
      - **TIME** - ASCII hodnotu '2' + čtyři čísla, které se nastaví na stopkách.
      - **PLAY** - ASCII hodnotu '3'.
      - **PAUSE** - ASCII hodnotu '4'.
      - **STOP** - ASCII hodnotu '5'.

   (b) Server odešle nazpět informaci o události.
3. **Ukončení spojení**
   (a) Aplikace odešle FIN/ACK paket.
   (b) Server odešle FIN/ACK paket.

4.3 Návrh obslužné aplikace

Stopky na časoměrném panelu musí být ovládány pomocí počítače. Pro tento účel na implementuji aplikaci, která s hodinami bude komunikovat.

4.3.1 Programovací jazyk

Aplikaci jsem se rozhodl naprogramovat v jazyce Python, ve kterém využiji frameworku Kivy, pomocí kterého se vytvoří grafické rozhraní GUI. Díky tomuhle výběru bude možné aplikaci spustit jak na operačních systémech Windows, Linux, Mac OS tak i na mobilních telefonech se systémy Android nebo iOS.

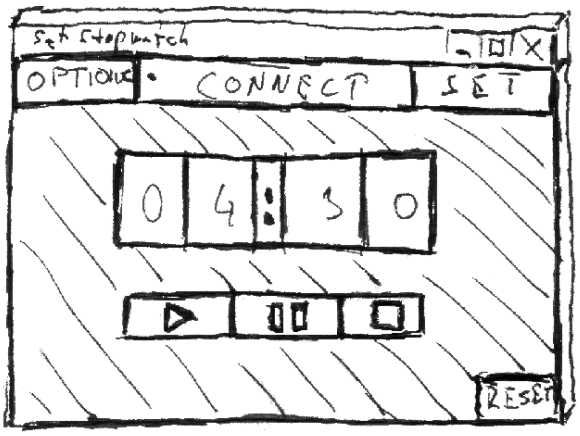
4.3.2 Návrh vzhledu aplikace

Aplikace musí graficky obsáhnout nastavování čísel pro jednotlivé LCD displaye stopek a jejich ovládání.

Pro tento účel jsem navrhl čtyři samostatné input pole, do kterých se postupně napíšou čísla času a pomocí buttonu "SET"se hodinám odešlou.

Pod čísly budou klasické START, PAUSE, STOP ikonky použité s GNU licencí, pomocí kterých se hodiny budou ovládat. Dále aplikace musí umět nastavit si IP adresu hodin. K tomuto účelu po kliknutí na button "OPTIONS"se zobrazí popup okno do kterého se IP adresa hodin zapíše a uloží.

Návrh vzhledu aplikace je ukázán na obrázku 4.2.



Obrázek 4.2: Návrh vzhledu aplikace, kterou se budou stopky ovládat.

Kapitola 5

Realizace

V této kapitole bude popsáno, jak jsem vytvořil schéma ethernetového modulu s deskou plošných spojů. Bude popsána také jeho výroba a testování jeho funkčnosti po zapojení.

Popíši zde, jak jsem naprogramoval mikroprocesor MC9S08LG16, aby obsluhoval vytvořený ethernetový modul, přes který by si dokázal načíst informaci o čase, kterou následně zpracuje a nastaví hodinám. Dále aby dokázal interagovat s obslužnou aplikací.

Poslední část kapitoly bude věnována obslužné aplikaci samotné. Bude v ní popsán způsob jakým jsem ji naprogramoval a otestoval.

5.1 Hardware ethernetový modul

K realizaci byla potřeba vybrat a nakoupit součástky se kterými škola nedisponovala. Všechny použité součástky jsou v tabulce v příloze C.

5.1.1 Schéma ethernetového modulu

Pro realizaci hardwaru s Wiznetem 5500 (w5500) pro síťovou komunikaci a s LTC4267 pro Power over Ethernet, jsem spojil jejich referenční schémata a připojil k nim síťový konektor RJ45, který oddělil od sebe napájecí napětí a signály síťové komunikace.

Aby bylo schéma kompletní, vytvořil jsem součástky i s footprinty které chyběly v dostupných knihovnách se součástky pro Altium Designer. Mnou vytvořené součástky jsou Wiznet 5500, LTC4267, konektor RJ45 MagJack 0813-1X1T-57-F, shottkyho dioda B1100, transformátor 1:1, unipolární tranzistor FDC2512, bipolární tranzistor MMBTA42, shottkyho dioda SS10P6-M3/86A a faston pro vývod napájecího napětí do časoměrného panelu.

Hotové schéma vytvořené pomocí programu Altium Designer je přiloženo v příloze B.

5.1.2 Deska plošných spojů

Desku plošných spojů jsem také vytvořil pomocí programu Altium Designer a je přiložena z obou stran v příloze B.

Po uváženém rozmístění součástek na desku plošného spoje o rozměrech 50x75 jsem se pokusil automaticky vygenerovat cesty utilitou autorouting, kterou Altium Designer poskytuje. Tahle verze ale porušovala zásady správného vedení cest, a tak jsem se rozhodl, že cesty plošných spojů udělám ručně.

Součástky na desce jsou rozmístěny tak, aby filtrační kondenzátory byly co nejblíž přívodu napětí. Stejně tak i usměrňovače Power over Ethernetem přivedeného napětí jsou co nejblíže k vývodům síťového konektoru.

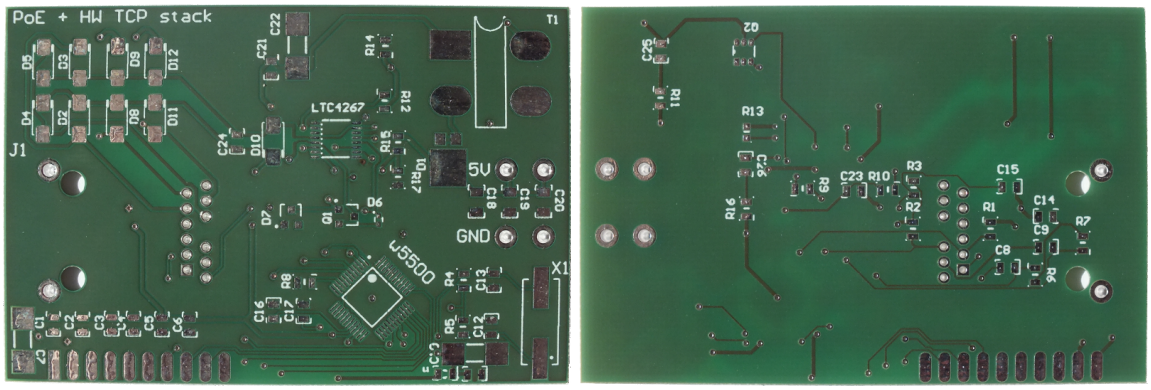
Cesty, které vedou napájecí napětí nebo napětí Power over Ethernet jsou o větší šířce o velikosti 0.5mm. Power over Ethernet je touto šířkou veden přes transformátor až do výstupního fastonu. Ostatní cesty jsou o standardní velikosti o šířce 0.2mm.

Na přední straně desky jsem rozlil vodící polygon, který vede uzemnění GND.

Na desce mám použité 4 velikosti vrtáků o velikostech 3.3mm, 1.6mm, 0.9mm a 0.4mm.

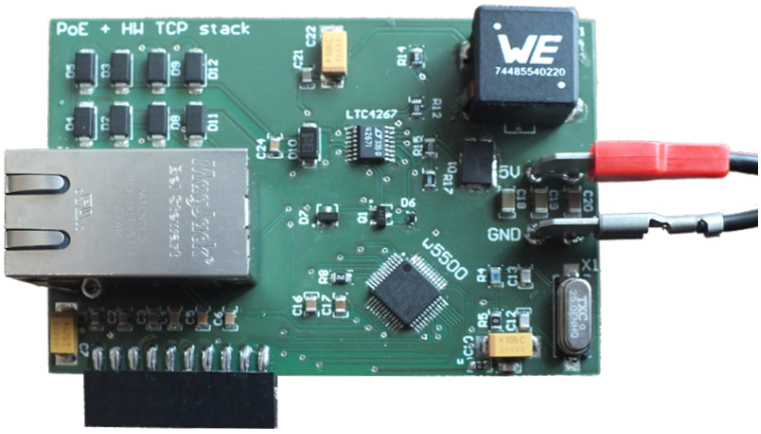
5.1.3 Výroba a osazení desky plošných spojů

Kvůli tomu, že fréza na výrobu plošných spojů je momentálně pokažená, tak jsme desku zadali do výroby externí firmě Gatema s.r.o., která vyrobila a dodala dva kusy.



Obrázek 5.1: Vyrobená deska plošných spojů.

Desku jsem osadil pomocí stroje, který je k dispozici na naší fakultě v laboratoři aplikovaných mikrokontrolérů L306. Horní strana byla osazena za pomocí pasty a se součástky vložena do pece, která cín roztavila a ten přilnul k vývodům součástek. Spodní vrstva byla osazena a napájená ručně za pomocí elektrické mikropájky.



Obrázek 5.2: Osazená deska plošných spojů.

5.1.4 Testování funkčnosti ethernetového modulu

Power over Ethernet

Po osazení desky jsem ethernetový modul zapojil do sítě poskytující Power over Ethernet podle normy IEEE 802.11af, abych změřil výstupní napětí na fastonech a tím otestoval obvod pro generování 5V. Požadované napětí jsem naměřil o přesné velikosti 5.02V, ale po chvíli napětí kleslo, a začalo se pohybovat v rozmezí 0.5V - 2V. Po sáhnutí na čip LTC4267 jsem zjistil, že jeho teplota byla velmi nepřiměřená a odpojil jsem ethernetový kabel.

Při analýze toho co se stalo jsem přišel na to, že jsem špatně navrhl footprint součástky Q2 (Unipolární tranzistor CMOS FDC2512), který je navržený zrcadlově otočený podél vertikální osy. To způsobilo že tranzistor má prohozený pin Gate s pinem Source, což způsobilo, že se přes Gate pustilo vysoké napětí na pin NGATE na čipu LTC4267, který se tímto zničil.

Následně jsem pečlivě proměření celý obvod, při čemž jsem nenarazil na žádný jiný problém. Odhaduji tedy, že po výměně čipu LTC4267 a otočením tranzistoru Q2 bude obvod fungovat správně.

Síťová komunikace

Po zapojení desky do klasické Ethernetové sítě bez PoE jsem otestoval síťovou komunikaci. Ta za celou dobu testování ovládání hodin při programování a testování nové verze obslužné aplikace obohacené o nové požadavky nevykázala jediné nestandardní chování. Modulu jsem přiřadil MAC adresu o hodnotě 00:08:DC:1D:4F:0A, kterou jsem pro ní vypůjčil z modulu WIZ550io, který má fakulta k dispozici pro testování.

5.2 Operační program mikroprocesoru MC9S08LG16

Implementaci programu ovládajícího ethernetový modul a jeho síťovou komunikaci jsem logicky rozdělil do čtyřech zdrojových souborů s jejich případnými hlavičkovými soubory.

V první dvojici `eth.c` a `eth.h` je nastavení komunikace mezi Freescale MC9S08LG16 a w5500 a také inicializace síťového nastavení IP adres a masky přerušení na w5500.

Druhá dvojice souborů `time.c` a `time.h` obsahuje obsluhu soketu pro získání času a následný výpočet času, který se nastaví hodinám na časoměrném panelu.

V třetí dvojici souborů `server.c` a `server.h` je naimplementován server, který zajišťuje obsluhu obslužného programu a podle přijatých dat nastavuje stopky na časoměrném panelu.

Dále jsem v souboru `MCUinit.c` naimplementoval zpracování IRQ přerušení ve funkci `__interrupt void isrVirq(void)`, která poté vyvolává příslušné funkce pro jejich další zpracování v souborech `time.c` nebo `server.c`.

Také jsem využil pro nastavování w5500 driver, který poskytutuje výrobce WIZnet Co. implementující v jazyce C komunikaci a adresaci registrů v w5500 a také funkce pro přijetí a odeslání dat pomocí soketu.

5.2.1 Implementace komunikace mezi Freescale MC9S08LG16 a Wiznetem 5500 a jeho nastavení

Pro komunikaci mezi mikroprocesorem MC9S08LG16 (dále jen mikroprocesorem) a Wiznetem 5500 (dále jen w5500) bylo připravené sériové periferní rozhraní (dále SPI),

které bylo potřeba nastavit na mikroprocesoru pro požadavky w5500.

Puštění mikroprocesorových hodin do SPI

V mikroprocesoru bylo nutné pustit do modulu poskytující SPI procesorové hodiny pomocí nastavení bitu `SCGC2_SPI` na hodnotu '1'.

Nastavení registru `SPIC1`

V tomto registru jsem povolil SPI, nastavil režim SPI přenosu, jeho fázi a polaritu hodinového signálu a také směr posuvu data registru. Je zakázáno každé SPI přerušení.

- **bit SPIE = 0** - zakázání přerušení při doručení bajtu.
- **bit SPE = 1** - povolení SPI.
- **bit SPTIE = 0** - zakázání přerušení při odeslání bajtu.
- **bit MSTR = 1** - nastavení SPI do režimu master.
- **bit CPOL = 1** - SPI klidová úroveň přenosu je log. 1.
- **bit CPHA = 1** - SPI hodnota je čtena při sestupné hraně.
- **bit SSOE = 1** - automatické vybírání výstupu pro slave.
- **bit LSBFE = 0** - posílání bitu od MSB do LSB.

Nastavení registru `SPIC2`

V tomto registru jsem povolil automatické vybírání výstupu pro slave a zakázal obousměrný provoz po jednom vodiči.

- **bit MODFEN = 1** - automatické vybírání výstupu pro slave.
- **bit BIDIROE = 0** - zakázání obousměrného výstupu.
- **bit SPISWAI = 0** - zakázání pozastavení stopek, když je mikroprocesor ve wait režimu.
- **bit SPC0 = 0** - řízení směru toku dat pří obousměrném výstupu.

Naprogramování funkcí pro odeslání a přijetí bajtu pomocí SPI

Driver pro ovládání w5500 musí mít k dispozici funkce které pro daný mikroprocesor odešlou a přijmou datový bajt. K tomuto účelu jsem naprogramoval funkci `void SpiWriteByte( uint8_ dat)` která odešle data w5500 a funkci `uinit8_t SpiReadByte(void)`, která data z w5500 přijme.

Další obslužné funkce jsou `void SpiSlaveSelect(void)` a `void SpiSlaveDeselect(void)` ve kterých povolují nebo zakazuji pomocí SS vývodu přenos mezi w5500 a mikroprocesorem.

5.2.2 Nastavení řídících registrů Wiznetu 5500

Kvůli zpoždění inicializace w5500 po zapnutí časoměrného panelu je nastavení registrů w5500 a síťová synchronizace času opožděno o jednu sekundu.

Nastavení masky přerušení pro určité sokety se provádí pomocí registru `SIMR`, který jsem nastavil na hodnotu 0x05, což povoluje přerušení od soketů 0 a 2.

Síťové nastavení w5500 je řešeno pomocí registrů popsaných v kapitole 1.3.5.

Síťové nastavení Wiznetu 5500

- **SIPR** - IP adresa modulu: 192.168.0.42.
- **SUBR** - Maska sítě: 255.255.255.0.
- **GAR** - Výchozí brána sítě: 192.168.0.1.

5.2.3 Implementace síťové synchronizace času

Synchronizace času je naimplementovaná v souborech `time.c`, `time.h` a `MCUinit.c`.

Inicializace soketu

V souboru `time.c` se spustí inicializační funkce `void Time_init(void)`. Ta zinicializuje pro synchronizaci času soket číslo 2 na w5500. Nastaví masku přerušení síťové komunikace soketu na `Sn_IMR` na hodnotu `0x0F`, díky čemuž povolí přerušení pří připojení soketu, doručení dat přes soket, timeout a ukončení spojení.

Dále nastavím soket pro protokol TCP zápisem hodnoty `0x01` do registru `Sn_MR` a nastavím náhodné hodnoty portu pro příjem dat do registru `Sn_PORT`. Následně zapíšu do registru `Sn_CR` hodnotu `0x01`, která soket zinicializuje a otevře.

Do registru zapíšu `Sn_DIPR` IP adresu Time serveru a do `Sn_DPORT` port serveru, na kterém naslouchá.

Pomocí zápisu hodnoty `0x04` do `Sn_CR` pošlu w5500 příkaz připojit se k serveru.

Zpracování následných přerušení

Následně mikroprocesor vyčkává na přerušení, která soket vyvolá. Když přerušení nastane, v souboru `MCUinit.c` jej funkce `__interrupt void isrVirq(void)` zpracuje. Nejprve pomocí registru `SIR` ověří, který soket přerušení vyvolal a poté pomocí registru `Sn_IR` zjistí jaká přesná událost nastala a adekvátně k ní spustí obslužnou funkci.

Každé přerušení se smaže zápisem hodnoty 1 na bit onoho přerušení v registru `Sn_IR`.

1. **Spojení ustanoveno** - když spojení je ustanoveno a přerušení zachyceno, spustí se funkce `void timeEstablished(int socket_number)` v souboru `time.c`, která vraci rovnou return a je připravena pro debugování nebo kdyby bylo případně něco potřeba udělat.
2. **Data přijaty** - když jsou data přijaty, spustí se funkce `void timeDataRecieved int socket_number` která nastaví registr `Sn_CR` hodnotou `0x08` do režimu disconnect, což zpusobí, že se odešle serveru FIN/ACK paket.

3. **Ukončení spojení** - po řádném ukončení spojení se serverem, přerušení spustí funkci `void timeDisconnected(int sn)`, která přečte data ze zásobníku pro příchozí data na w5500 a zkonkatenuje všechny čtyři bajty dohromady aby jsme dostali celistvou hodnotu o počtu sekund které uběhli od data 00:00 1.1.1900. Tuhle hodnotu předá funkci `setTime(uint32_t time)`.

4. **Timeout** - když primární server není dostupný a vyprší čas pro pokus o ustanovení spojení, spustí se alternativní inicializační funkce `Time_timeout_init()`, která pracuje úplně stejně jako primární funkce `Time_init()` jen se sekundárním Time serverem. Tím se spustí stejný proces jako při prvním pokusu o ustanovení spojení.

Zpracování časové informace a nastavení času časoměrnému panelu

Pro zpracování časové informace jsem naprogramoval funkci `void setTime(uint32_t seconds)`, která postupně odečítá z sekund roky, čímž zajistí správný výpočet let přestupných a dále postupně měsíce, dny, hodiny a minuty, čímž máme postupně v proměnných `year`, `month`, `day`, `hour`, `min` a `seconds` informaci o aktuálním čase v GMT časovém pásmu.

Čas na hodinách poté nastavim funkcí `void LCD_setTime (uint8_t hours, uint8_t minutes, uint8_t seconds)` kterou jsem vytvořil v `lcd.c`.

5.2.4 Implementace serveru pro komunikaci s obslužnou aplikací

Pro implementaci souboru jsem vytvořil soubory `server.c` a jeho hlavičkový soubor `server.h`. Dále v souboru `MCUinit.c` je ve funkci `__interrupt void isrVirq(void)` naimplementovaná obsluha přerušení.

Inicializace soketu

V souboru `server.c` se spustí inicializační funkce `void Server_init()`. Ta nastaví registr přerušení soketu `Sn_IMR` na hodnotu `0x1F`, čímž se nastaví vyvolání přerušení při připojení soketu, doručení dat přes soket, úspěšné odeslání dat, timeout a ukončení spojení.

Dále nastavím soket pro protokol TCP zápisem hodnoty `0x01` do registru `Sn_MR` a nastavím hodnotu portu pro příjem dat pomocí registru `Sn_PORT` na port 4242. Následně zapíšu do registru `Sn_CR` hodnotu `0x01`, která soket zinicializuje a otevře.

Poslední část inicializace se provede pomocí zápisu hodnoty `0x02` do registru `Sn_CR`, která přepne soket do módu Listen a tím zajistí naslouchání na portu 4242.

Zpracovávání následných přerušení

Následně soket vyčkává na případné připojení obslužné aplikace a jeho požadavky. Když přerušení nastane, v souboru `MCUinit.c` jej funkce `__interrupt void isrVirq(void)` zpracuje. Nejprve pomocí registru `SIR` ověří, který soket přerušení vyvolal a poté pomocí registru `Sn_IR` zjistí jaká přesná událost nastala a adekvátně k ní spustí obslužnou funkci.

Příznak přerušení se smaže zápisem hodnoty 1 na bit onoho přerušení v registru `Sn_IR`.

Při přijetí hodnoty, která spadá do navrhženého soketu ji obratem odešle nazpět načtením ji do odesílacího zásobníku soketu pomocí funkce `wiz_recv_data(socket_number, data, size)`. Poté nastavím registru `Sn_CR` na hodnotu `0x20` oznámí w5500, že má data odeslat.

1. **Spojení ustanoveno** - když spojení je ustanoveno a přerušení zachyceno, jen se smaže jeho příznak a program pracuje dál. Je to připraveno pro přehlednost a pro debugování.

2. **Data přijaty** - když jsou data přijaty, spustí se funkce `void serverDataRecieved()` která je přečte, a podle nich spustí další rutinu.

   (a) **TEST** - když přijde ASCII hodnota 1 - jen ji odešle nazpět.

   (b) **TIME** - když přijde ASCII hodnota 2 - hodnotu odešle nazpět a server načte další čtyři bajty s informaci o čase. Tu pošle funkci `void setStopwatchTime( uint8_t)`, která zní vypočítá čas který se má nastavit stopkám. Vypočtený čas předá funkci `LCD_setStopwatch(uint8_t first_num, uint8_t second_num)`, kterou jsem naimplementoval v souboru `lcd.c`. Funkce `LCD_setStopwatch(uint8_t first_num, uint8_t second_num)` nastaví čas stopkám a nastaví stav stopek do defaultního stavu.

   (c) **PLAY** - když přijde ASCII hodnota 3 - hodnotu odešle nazpět a spustí funkci `LCD_play()` v souboru `lcd.c` která nastaví proměnnou `tbeep` na hodnotu 2 a tím se spustí stopky.

   (d) **PAUSE** - když přijde ASCII hodnota 4 - hodnotu odešle nazpět a spustí funkci `LCD_pause()` v souboru `lcd.c` která nastaví proměnnou `tbeep` na hodnotu 0 a tím se stopky zastaví.

   (e) **STOP** - když přijde ASCII hodnota 5 - hodnotu odešle nazpět a spustí funkci `LCD_stop()` v souboru `lcd.c` která nastaví proměnnou `tbeep` na hodnotu 0 a tim se stopky zastaví a dále zapíše jako aktuální čas hodnotu `00:00`.

3. **Data odeslané** - když jsou data v odeslány, zavolá se funkce `serverDataSended()`, která nastaví registr `Sn_CR` na hodnotu `0x08`. To zapříčiní odeslání FIN/ACK paketu od serveru k obslužné aplikaci a tím se spojení ukončí.

4. **Ukončení spojení** - po řádném ukončení spojení s klientem server znova spustí funkci `Server_init()` Server se tak znovu připraví na nový požadavek od obslužné aplikace.

5.3 Obslužná aplikace

5.3.1 Implementace aplikace

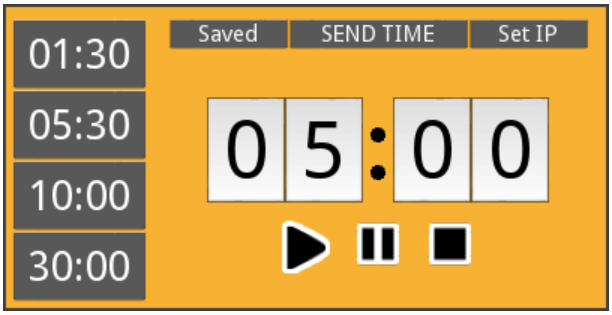
Obslužnou aplikaci jsem naimplementoval v jazyce Python verze 2.7 a ve frameworku pro GUI Kivy verze 1.9.0. Oddělil jsem grafickou část od části vykonávající síťovou komunikaci a zpracovávání dat.

Front end

Grafickou část jsem rozdělil na tři části. Na část horní - `MenuBar` - ve které jsou zasazené nastavovací buttony, na část dolní - `MainStopwatch` - do které jsem zasadil textové pole pro zadání času a ovládací tlačítka. Po konzultaci jsem přidal část třetí na levý bok, ve které jsou zasazeny programovatelné tlačítka s časem, které když se na ně klikne, odešlou a rovnou i spustí nastavený čas.

Při kliknutí na button "Set IP"se objeví popup okno `SetIpPopup`. V něm je políčko, ve kterém uživatel zadá IP adresu časoměrného panelu a uloží. Zadání IP adresy je ošetřeno tak, aby byla zadaná pouze validní IPv4 adresa.

Při kliknutí na button "Saved"se objeví popup okno `SavedClock`, ve kterém uživatel může nastavovat časy na buttonech v bočním panelu.



Obrázek 5.3: Grafické rozhraní obslužné aplikace.

Back end

V back endu aplikace se pomocí objektu `MenuBar` načítají čísla z textových polí a zpracovávají se stisknuté buttony pod velkými hodiny. Data poté objekt `MenuBar` předá objektu `Sender`, který je zpracuje a odešle přes síť ethernetovému modulu. Komunikace probíhá podle navrženého protokolu, posaného v kapitole 4.2.3. Podobně je naimplementováno i chování tlačítek na bočním panelu.

Jak časy nastavené buttonům na boční straně, tak i IP adresa časoměrného modulu se po nastavení uloží do pomocného souboru, z něhož při dalším zapnutí aplikace si data zpět načte. Díky tomu jsou tyto data perzistentní.

5.3.2 Testování obslužné aplikace

Obslužnou aplikaci jsem naimplementoval a tím i otestoval na systému Gentoo Linux. Pro potřeby školy jsem vygeneroval spustitelnou aplikaci i pro systémy Windows a Andorid, kde jsem na těchto platformách vyladil aplikaci tak, že funguje bezproblémově.

Závěr

V této práci jsem navrhl a vyrobil ethernetový modul o který jsem rozšířil časoměrný panel. Pomocí něj je možné připojit panel do sítě a napájet jej přes Power over Ethernet.

Vytvořil jsem k nim obslužnou grafickou aplikaci v jazyce Python s pomocí frameworku Kivy pro vytvoření grafického uživatelského rozhraní. Touto aplikací je možné ovládat stopky na časoměrném panelu přes počítač nebo mobilní telfon. Pro tento účel jsem na mikroprocesoru na časoměrném panelu naprogramoval server, který zpracovává dotazy obslužné aplikace, a adekvátně pracuje se stopkami na panelu. Obslužná aplikace je díky vhodně zvolenému jazyku a frameworku Kivy multiplatformní a lze ji používat jak na systémech Windows, Linux, Mac OS, tak i na mobilních telefonech se systémy Android nebo iOS. Aplikaci jsem vygeneroval pro Windows OS a Android, na Linuxu je snadno spustitelná.

Dále jsem na hodinách naprogramoval synchronizaci času, pro kterou jsem použil protokol Time, pomocí něhož si časoměrný modul načte přesný čas a nastaví jej hodinám.

Testování

Testování obvodu pro Ethernetovou komunikaci proběhlo ve všech směrech bezproblémově. Při četném testování komunikace obslužné aplikace s časoměrným panelem, síťová komunikace nevykazovala žádné nestandardní chování.

Při realizaci desky plošných spojů jsem špatně navrhl footprint pro tranzistor Q2 FDC2512, který je díky tomu na desce vyroben se zrcadlově otočenými piny. Kvůli této chybě přešel přes tranzistor vysoký proud do čipu LTC4276, který se tímto zničil. Kvůli tomuto, obvod poskytující Power over Ethernet je nefunkční, dokud se čip LTC4267 nevymění za nový a tranzistor nepřipájí opačně. Součástka je v době odevzdání práce objednána, ale zásilka nestihla přijít. Proto v době odevzdání této práce není obvod s Power over Ethernetem řádně otestovaný.

Při prvním testování obvodu pro Power over Ethernet, obvod správně vyváděl na výstup 5V potřebných pro napájení hodin, poté se čip LTC4267 spálil a obvod se stal nefunkčním. Tahle událost neovlivnila majoritní obvod pro Ethernetovou komunikaci.

Pokračování projektu

Další práce, která může být na časoměrném panelu realizována, je rozšíření jej o modul s flash pamětí. Do té by si program mohl ukládat a trvale modifikovat své síťové nastavení, IP adresy časových serverů, nebo třeba uložit DHCP paket, který pro svou velikost není možné mít uložený v kódu. Tak se zlepší jejich komfort a funkcionalita, protože hodiny začnou být programovatelné jako jiné sofistikovanější síťové prvky.Tyto změny by mohla přinést nová generaci hodin, která by mohla být osazena i krystalem pro přesnější generování času a také LED displayem pro lepší viditelnost zobrazovaného času.

Literatura

[1] Ieee standard of ethernet. [online]. December 2012 [Cited 2014-12-03]. Dostupné na: <http://standards.ieee.org/getieee802/download/802.3-2012_section1.pdf>.

[2] Ieee standard of ethernet - section two. [online]. December 2012 [Cited 2014-12-03]. Dostupné na: <http://standards.ieee.org/getieee802/download/802.3-2012_section2.pdf>.

[3] Ieee standard of ethernet - section three. [online]. December 2012 [Cited 2014-12-03]. Dostupné na: <http://standards.ieee.org/getieee802/download/802.3-2012_section3.pdf>.

[4] D. Mills , U. Delaware, J. Martin, J. Burbank, W. Kasch. Network time protocol version 4: Protocol and algorithms specification. [online]. 2010 [cit. 2015-01-10]. Dostupné na: <http://tools.ietf.org/html/rfc5905>.

[5] J. Postel. Daytime protocol. [online]. 1983 [cit. 2015-03-26]. Dostupné na: <http://tools.ietf.org/html/rfc867>.

[6] J. Postel , K. Harrenstien. Time protocol. [online]. 1983 [cit. 2015-03-26]. Dostupné na: <http://tools.ietf.org/html/rfc868>.

[7] WIZNET. W5500 datasheet version 1.0.6. [online]. 2014 [cit. 2015-04-04]. Dostupné na: <http://wizwiki.net/wiki/lib/exe/fetch.php?media=products:w5500:w5500_ds_v106e_141230.pdf>.

[8] Galit Mendelson. All you need to know about power over ethernet (poe) and the ieee 802.3 af standard. [online]. 2004 [cit. 2015-04-10]. Dostupné na: <http://kondorsecurity.com/store/media/pdf/PoE_and_IEEE802_3af.pdf>.

[9] LINEAR TECHNLOGY. Ltc4267 power over ethernet ieee 802.3af pd interface with integrated switching regulator. [online]. [cit. 2015-04-12]. Dostupné na: <http://www.linear.com/docs/5437>.

[10] Zdeněk VAŠÍČEK. Rozšiřující modul - ethernet - fitkit. [online]. 2010 [Cited 2015-04-26] <http://merlin.fit.vutbr.cz/FITkit/docs/hardware/hw_mod_eth.html>.

[11] FREESCALE. Mc9s08lg32 mc9s08lg16 reference manual hcs08 microcontrollers. [online]. 2009 [cit. 2015-04-27]. Dostupné na: <http://cache.freescale.com/files/microcontrollers/doc/ref_manual/MC9S08LG32RM.pdf>.

[12] FREESCALE. S08lg: 8-bit segment lcd s08lg32 and s08lg16 mcus. [online]. [cit. 2015-04-27]. Dostupné na: <http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=S08LG>.

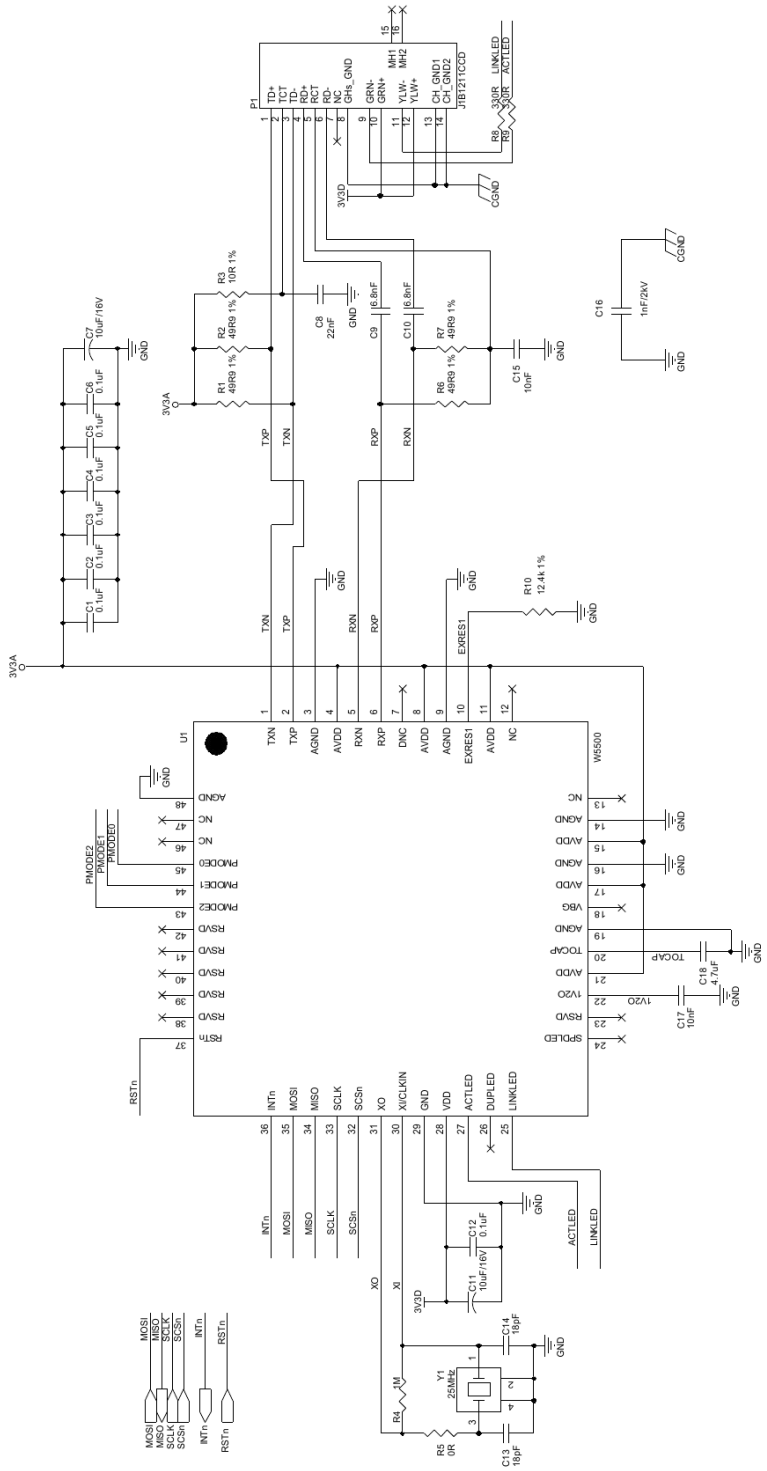
[13] BEL FUSE. 0813-1x1t-57-f power over ethernet single port magjack®. [online]. [cit. 2015-05-05]. Dostupné na: <www.farnell.com/datasheets/1385870.pdf>.

[14] WIZNET. Reference schematic rj45 with transformer type. [online]. [cit. 2015-05-10]. Dostupné na: <http://wizwiki.net/wiki/lib/exe/fetch.php?cache=&media=products:w5500:rj45_with_transformer.jpg>.

Seznam příloh

Příloha A

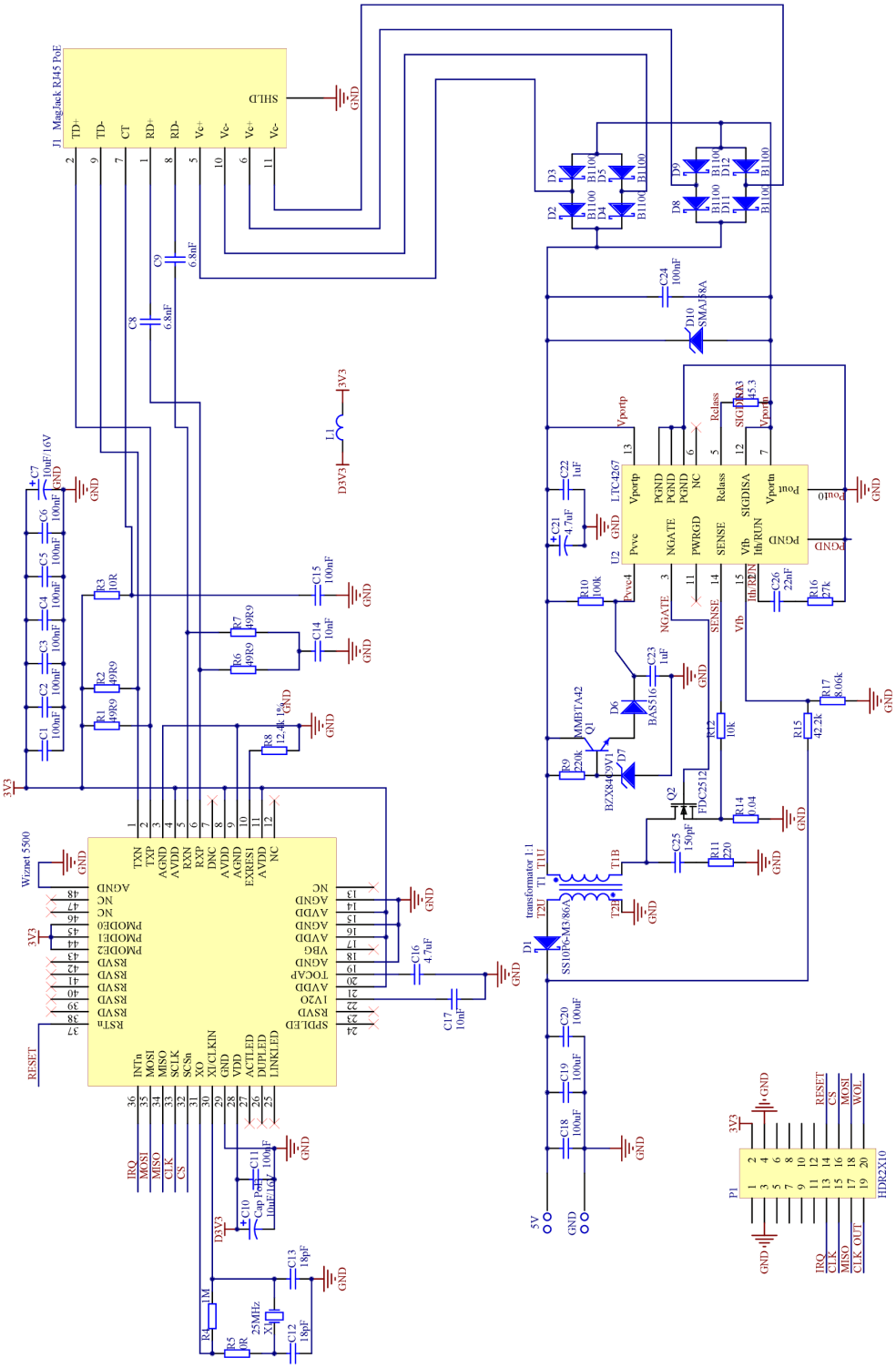
Referenční schéma zapojení TCP HW stack Wiznet W5500



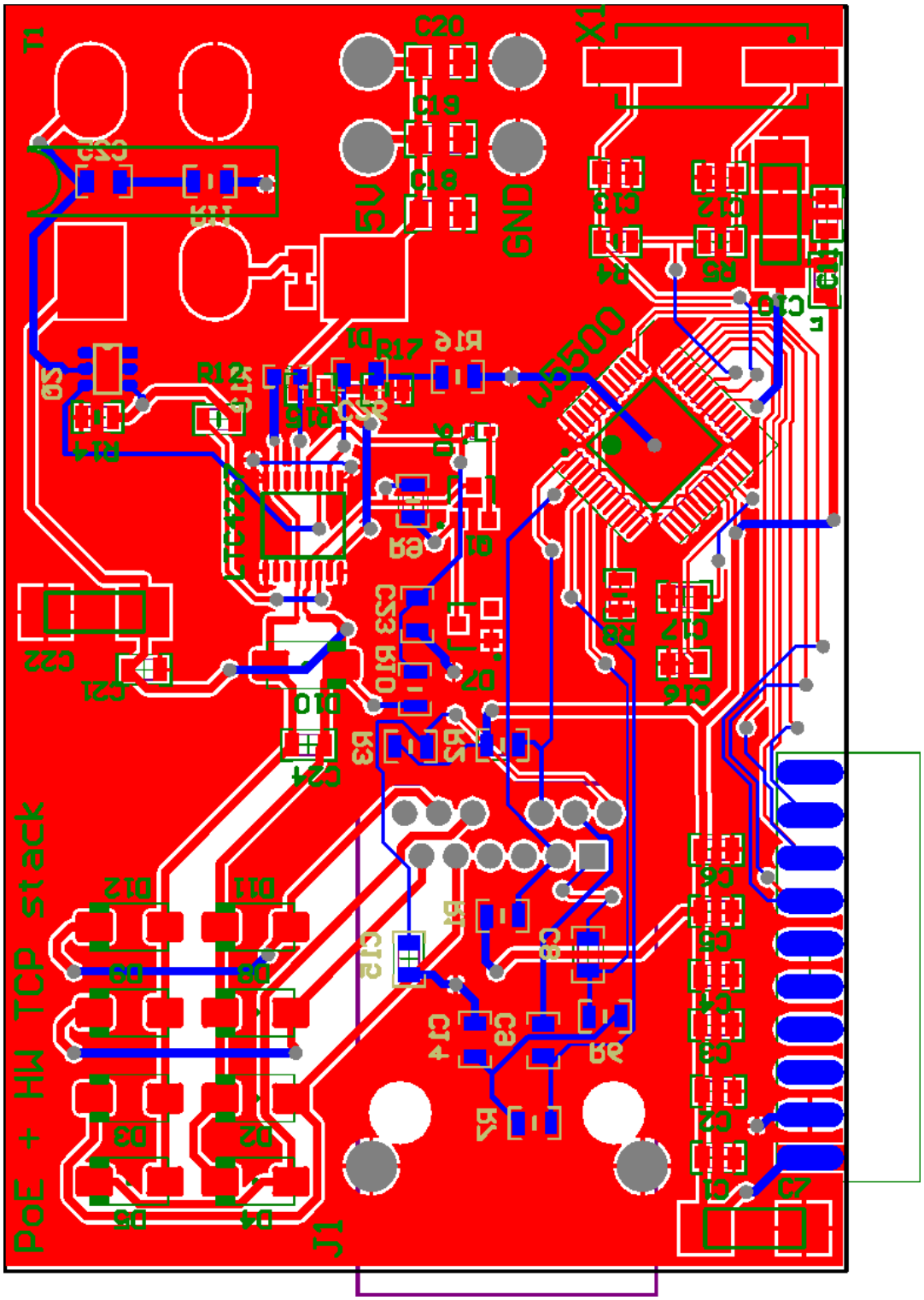
Obrázek A.1: Referenční schéma w5500 s integrovaným galvanickým odizolováním v RJ45. Zdroj [14].

Příloha B

Schéma a deska plošných spojů ethernetového modulu



Obrázek B.1: Schéma zapojení ethernetového modulu.



Obrázek B.2: Deska plošných spojů, horní (červená) a spodní (modrá) vrstva.

Příloha C

Seznam použitých součástek

množství	**součástka**	**vysvětlení**
4	odpor 49.9Ω	udržení proudu mezi TX, RX a w5500
1	odpor 10Ω 1%	puštění Ucc do CT
1	odpor 1MΩ	rozpojení pro vedení kmitočtu krystalu
1	odpor 0Ω	zkrat pro vedení kmitočtu krystalu
1	odpor 12.4kΩ	externí referenční rezistor
1	odpor 220kΩ	odpojení báze NPN tranzistoru od PoE
1	odpor 100kΩ	usměrnění proudu ze zesilovače
1	odpor 220RΩ	RC článek u transformátoru
1	odpor 10kΩ	dělič napětí senzoru
1	odpor 45.3Ω 1%	referenční odpor pro určení výkonnostní třídy PoE
1	odpor 0.04Ω 1%	dělič napětí senzoru
1	odpor 42.2kΩ 1%	dělič napětí výstupu 5V
1	odpor 8.06kΩ 1%	dělič napětí výstupu 5V
1	odpor 27kΩ	RC článek Ith/RUN
9	kondenzátor 100μF	filtry Ucc
2	kondenzátor 10μF	filtr Ucc
2	kondenzátor 6.8nF	filtry RX
2	kondenzátor 18pF	filtry oscilátoru
2	kondenzátor 10nF	1V20 vůči GND a RX vůči GND
2	kondenzátor 4.7μF tantalový	filtr Ucc a PoE
1	kondenzátor 4.7μF	w5500 TOCAP vůči GND
3	kondenzátor 100μF	filtr výstupu PoE obvodu
1	kondenzátor 1μF	filtr PoE
1	kondenzátor 150pF	RC článek u transformátoru PoE
1	kondenzátor 22nF	RC článek Ith/RUN vůči GND PoE
1	dioda schottkyho SS10P6-M3/86A	výstup z transformátoru do odporového děliče a filtru
8	dioda schottkyho B1100	usměrňovače PoE
1	dioda BAS516	záklopka k emitoru NPN tranzistoru
1	dioda zenerova BZX84C9V1	záklopka k bázi NPN tranzistoru od Pvvc LTC4267
1	dioda zenerova SMAJ58A	stabilizační zenerova dioda PoE
1	tranzisotr bipolární NPN MMBTA42	zesilovač
1	tranzistor unipolární FDC2512	regulátor výstupního napětí
1	krystal 25MHz	krystal pro generování kmitočtu pro w5500
1	čip LTC4267	PoE mikrokontrolér
1	čip Wiznet 5500	TCP hardwarový stack
1	konektor RJ45 MagJack 0813-1X1T-57-F	připojení ethernetu a PoE
1	konektor HDR2X10	dvaceti-pinový konektor k připojení k panelu