

UNIVERZITET U BEOGRADU
ELEKTROTEHNIČKI FAKULTET

MASTER RAD

Implementacija

mentor:
Prof. dr Lazar Saranovac

student:
Lazar Caković
br. indeksa: 3083/2016

Beograd, septembar 2018.

Sadržaj

1	Apstrakt	3
2	Uvod	4
3	Protokol	5
3.1	OSI model	5
3.1.1	Sloj 1: Fizicki sloj (Physical Layer)	6
3.1.2	Sloj 2: Sloj veze (Data Link Layer)	6
3.1.3	Sloj 3: Mrezni sloj (Network Layer)	7
3.1.4	Sloj 4: Transportni sloj (Transport Layer)	7
3.1.5	Sloj 5: Sloj sesije (Session Layer)	8
3.1.6	Sloj 6: Sloj prezentacije (Presentation Layer)	8
3.1.7	Sloj 7: Aplikativni sloj (Application Layer)	9
3.2	TCP/IP model	9
3.3	Precision Time Protocol	10
3.4	Algoritam najboljeg sata (Best master clock algorithm)	12
3.5	Sinhronizacija (Synchronization)	13
4	Operativni sistem	15
4.1	FreeRTOS	15
4.2	lwIP	18
5	Hardverska implementacija	21
5.1	Time stamping unit - TSU	23
6	Softverska implementacija	25
7	Zaključak	26

Slike

5.1	Razvojna ploca SAMA5D27-SOM1-EK1 (pogled odozgo) . . .	21
5.2	Moduli - Razvojna ploca SAMA5D27-SOM1-EK1	22
5.3	Moduli - SAMA5D27-SOM1	23

Glava 1

Apstrakt

Glava 2

Uvod

Glava 3

Protokol

3.1 OSI model

Open Systems Interconnection model (OSI model) je konceptualni model koji karakterise i standardizuje komunikacione funkcije u telekomunikacionim ili kompjuterskim sistemima, i to bez obzira na unutrašnju strukturu uređaja ili njihovu tehnologiju. Cilj ovog modela je da se postigne kompatibilnost različitih komunikacionih sistema sa standardnim protokolima komunikacije. OSI model razdvaja komunikacione sisteme u apstraktne slojeve. Originalna verzija modela ima sedam slojeva.

Sloj unutar modela služi sloj iznad njega, i koristi sloj ispod njega u hijerarhiji. Na primer, sloj koji postize komunikaciju preko mreže bez gresaka, služi aplikacijama iznad koje ga koriste, i to dok poziva jednostavne funkcije za prijem i predaju paketa na mreži. Dve instance istog sloja su vizualizovane tako što su povezane horizontalno u istom sloju.

Ovaj model je proizvod Open Systems Interconnection projekta u International Organization for Standardization (ISO), i ima oznaku ISO/IEC 7498-1.

[lazar] slika svih slojeva u OSI modelu

Na svakom nivou N, dva entiteta na komunikacionim uređajima razmenjuju jedinice protokola (PDU - protocol data units) pomoću sloja N protokola. Svaki PDU sadrži podatke od interesa (payload) (SDU - service data unit), zajedno sa zaglavlјima koji odgovaraju protokolu.

Obrada podataka između dva uređaja koji su OSI-kompatibilni se odvija u sledećim koracima:

- Podaci koji se prenose se formiraju na najvišem sloju u uređaju koji predaje podatke na mreži (sloj N) u jedinicu protokola (PDU).
- PDU se prosledjuje sloju N-1, gde je poznat kao SDU.

- Na sloju N-1 se na SDU dodaju zaglavlja, na osnovu cega se formira PDU za sloj N-1. Nakon cega se prosledjuje na sloj N-2.
- Ovaj postupak se ponavlja sve dok se ne dostigne najnizi sloj u modelu, nakon cega se podaci prenose ka uredjaju koji prima podatke.
- Na strani prijemnog uredjaja se podaci prenose od najnižeg sloja u modelu, do najvišeg, gde se serije SDU struktura uspesno obradjuju, pri cemu se skidaju zaglavlja sa svakog sloja, dok se ne dostigne najvisi sloj u modelu, nakon cega su dostupni sirovi podaci.

[lazarc] dodati jos nesto o ovome ukoliko se nadje.

3.1.1 Sloj 1: Fizicki sloj (Physical Layer)

Fizicki sloj je odgovoran za prenos i prijem nestrukturiranih sirovih podataka izmedju uredjaja i fizickog medijuma za prenos. On pretvara digitalne bitove u elektricne, radio ili opticke signale. Specifikacije sloja definisu karakteristike poput nivoa napona, fizicke brzine prenosa podataka, maksimalne udaljenosti prenosa i fizickih konektora. Ovo ukljucuje raspored pinova, napona, linijske impedanse, specifikacije kablova, vremenskih signala i frekvencije za bezicne uredjaje. Kontrola brzine bitova se vrši na fizickom nivou i moze definisati nacin komunikacije kao simpleks, polu dupleks ili dupleks komunikaciju. Komponente fizickog sloja mogu se opisati u smislu topologije mreze. Bluetooth, Ethernet i USB, sve imaju specifikacije za fizicki sloj.

3.1.2 Sloj 2: Sloj veze (Data Link Layer)

Sloj veze podataka obezbedjuje prenos podataka izmedju dva cvora u komunikaciji - vezu izmedju dva direktno povezana uredjaja na mrezi. Ovaj sloj otkriva i eventualno ispravlja greske koje se mogu javiti u fizickom sloju. On definise protokol za uspostavljanje i prekid veze izmedju dva fizicki povezana uredjaja. Takodje, definise protokol za kontrolu protoka izmedju njih.

IEEE 802 standard deli sloj veze na dva podsloja:

- **Kontrola pristupa medijumu (MAC - Medium access control):** ## [lazarc] [prevod] odgovorna je za kontrolu nacin na koji uredjaji na mrezi dobijaju pristup medijumu i dozvolu za prenos podataka.
- **Kontrola logicke veze (LLC - Logical link control):** ## [lazarc] [prevod] odgovorna je za identifikaciju i enkapsulaciju slojeva mreznog protokola, i kontrolu gresaka i sinhronizaciju paketa koji se salju.

MAC i LLC slojevi IEEE 802 mreznog standarda kao sto su 802.3 Ethernet, ili 802.11 Wi-Fi su slojevi veze (data link layer).

Point-to-Point Protocol (PPP) - je protokol sloja veze koji radi na nekoliko razlicitih fizickih slojeva, kao sto su sinhroni ili asinhroni serijske linije.

3.1.3 Sloj 3: Mrežni sloj (Network Layer)

Mrežni sloj obezbedjuje funkcionalno i proceduralno sredstvo prenosa sekvenci podataka promenljive duzine, koji se nazivaju jos i paketi, od jednog cvora do drugog, povezanih u "razlicite mreze". ## [lazarc] [prevod] ## Mreza je medijum na koji moze biti povezano vise cvorova, u kom svaki cvor ima adresu i koji dozvoljava cvorovima povezanim sa njim da prenose poruke ka ostalim cvorovima, i to samo dajuci sadrzaj poruke i adresu cvora na koji poruka treba da bude dostavljena, omogucavajući mrezi da nadje nacin da isporuci poruku odredisnom cvoru, eventualno ga usmeravajući kroz sredisnje cvorove (uredjaje koji su izmedju dva uredjaja koji pokusavaju da komuniciraju). Ako je poruka prevelika da bi se prenela sa jednog uredjaja na drugi samo koriscenjem sloja veze (Data Link Layer), mreza moze preneti podatke tako sto ce ih podeliti u nekoliko delova na jednom uredjaju, poslati delove nezavisno, i onda spojiti delove na drugom uredjaju. Pri cemu moze, iako nije uvek potrebno, prijaviti greske u isporuci.

Isporuka poruka na mreznom sloju nije garantovano pouzdana. Mrežni sloj moze pruziti pouzdanu isporuku poruka, ali nije obavezno da to mora biti ispunjeno.

Neki broj protokola koji upravljaju slojevima, imaju funkciju koja je definisana u aneksu upravljanja, ISO 7498/4, i pripadaju mreznom sloju. Oni ukljucuju protokole rutiranja, upravljanja grupomsa vise uredjaja, informacije o mreznom sloju, o greskama, kao i dodeljivanje adresa mreznog sloja medju uredjajima. Ustvari, to je funkcija podataka koji se prenose uz pomoc protokola, sto ih cini da pripadaju mreznom sloju, a ne protokolu. ## [lazarc] poslednja recenica [prevod] ##

3.1.4 Sloj 4: Transportni sloj (Transport Layer)

Transportni sloj obezbedjuje funkcionalno i proceduralno sredstvo prenosa sekvenci podataka promenljive duzine od predajnog do prijemnog uredjaja, uz odrzavanje kvaliteta.

Transportni sloj kontrolise pouzdanost date konekcije kroz kontrolu protoka, segmentaciju/desegmentaciju i kontrolu gresaka. Neki protokoli su orjentisani na stanje mreze, a neki na konekciju u mrezi. ## [lazarc] [prevod] ## Ovo znaci da Transportni sloj moze da prati segmente i ponovo preneti one

koji nisu isporučeni prijemnom uređaju. Transportni sloj takođe omogućava i potvrdu uspešnog prenosa podataka i šalje naredne podatke ako nije došlo do greske prilikom prenosa. Transportni sloj formira i segmente koji su primljeni iz viših slojeva, npr Aplikativnog sloja (Application Layer). Segmentacija je proces podele dugih poruka u kraće poruke kako bi se lakše prenele preko nizih slojeva u modelu.

OSI model definiše pet klasa transportnih protokola za povezivnje od klase 0 (koja je takođe poznata i kao TP0 i ima najslabije karakteristike) do klase 4 (TP4, koja je dizajnirana za manje pouzdane mreže, slične Internetu). Klasa 0 (TP0) nema mogućnost oporavka od greske i bila je dizajnirana za korišćenje na mrežnim slojevima koji pružaju konekciju bez gresaka. Klasa 4 (TP4) je najbliža TCP, iako TCP sadrži neke funkcije koje se u OSI modelu dodeljuju višim slojevima. Takođe, sve klase u OSI modelu omogućavaju brzu upotrebu podataka i očuvanje granica podataka ## [lazarc] [prevod] ##. Detalje karakteristika svih klasa prikazane su u sledećoj tabeli:

[lazarc] tabela sa interneta ## [lazarc] prebaci tabelu na srpski

3.1.5 Sloj 5: Sloj sesije (Session Layer)

Sloj sesije kontroliše dijaloge (veze) između uređaja. Uspostavlja, upravlja i uklanja veze između lokalnih i udaljenih aplikacija. Obezbeđuje funkcije Full-Duplex, Half-Duplex ili Simplex i uspostavlja procedure Checkpoint-a, prekida ili ponovnog pokretanja procedura. OSI model je učinio ovaj sloj odgovornim za dobro završavanje sesija, što je osobina TCP (Transmission Control Protocol), i takođe proveru sesija i oporavak, što se obično ne koristi u Internet Protocol Suite. Sloj sesije se obično eksplicitno primenjuje u sredinama aplikacija koje koriste proceduralne pozive na udaljenim uređajima.

3.1.6 Sloj 6: Sloj prezentacije (Presentation Layer)

Sloj prezentacije uspostavlja kontekst između dva entiteta aplikativnog sloja, u kom entiteti aplikativnog sloja mogu koristiti različitu sintaksu i semantiku ukoliko sloj prezentacije pruža mapiranje između njih. Ukoliko je dostupno mapiranje, jedinice protokola su enkapsulirane u jedinice sesije i prosledjene na nize slojeve.

Ovaj sloj obezbeđuje nezavisnost od predstavljanja podataka u različitim aplikacijama i mrežnim formatima. Sloj prezentacije pretvara podatke u oblik koji prihvata zadata aplikacija. Ovaj sloj formatira podatke koji se šalju preko mreže. Ponekad se naziva i sintakсни sloj. Takođe, može uključivati i funkcije kompresije.

3.1.7 Sloj 7: Aplikativni sloj (Application Layer)

Aplikativni sloj je OSI sloj najblizi krajnjem korisniku, što znači da i OSI aplikativni sloj i korisnik interaguju direktno sa aplikacijom. Ovaj sloj komunicira sa softverskom aplikacijom koja sadrži komponentu za komunikaciju. Takve aplikacije ne spadaju u okvir OSI modela. Funkcije u aplikativnom sloju obično uključuju identifikovanje partnera u komunikaciji, određivanje dostupnosti resursa i sinhronizaciju komunikacije. Prilikom identifikacije uređaja za komunikaciju, aplikativni sloj se razlikuje od samih aplikacija. Na primer, internet aplikacija (web strana) može imati dva entiteta - dve aplikacije: jedna koja koristi HTTP za komunikaciju sa korisnicima, i drugu za udaljenu bazu podataka koja čuva podatke. Ni jedan od ovih protokola nemaju nista sa podacima koji se čuvaju, to se nalazi samo u aplikaciji. Aplikacijski sloj nema načina za određivanje resursa u mreži.

3.2 TCP/IP model

Internet protocol suite:

Internet protocol suite je konceptualni model i set komunikacionih protokola koji se koriste za Internet i slične kompjuterske mreže. Opšte je poznat kao TCP/IP zbog toga što su osnovni protokoli u ovom modelu TCP (Transmission Control Protocol) i IP (Internet Protocol). Ponekad se naziva i DoD (Department of Defense) model zbog toga što je razvijanje ovog modela pomoglo Ministarstvo odbrane SAD-a kroz DARPA.

Internet protokol suite ([lazarc] [prevod]) omogućava razmenu podataka između dva uređaja na mreži, i to specificirajući kako će se podaci deliti u pakete, adresirati, prenositi, rutirati, i primati. Ove funkcionalnosti su organizovane u četiri apstraktna sloja, koja klasifikuju sve protokole s obzirom na to u kom delu povezivanja se nalaze ([lazarc] [prevod] scope of networking involved). Od najnižeg do najvišeg, slojevi se dele na Link Layer (Sloj povezivanja), koji sadrži komunikacione metode za podatke koji ostaju unutar jednog segmenta mreže; Internet Layer (Sloj interneta), koji omogućava povezivanje između nezavisnih mreža; Transport Layer (Sloj prenosa), koji omogućava komunikacione servise za aplikacije na uređajima u mreži; i Application Layer (Sloj aplikacije), koji omogućava servise korisnicima i sistemskim aplikacijama.

Tehnički standardi koji specificiraju Internet protocol suite ([lazarc] [prevod]) i mnogi od protokola koji čine IPS održava Internet Engineering Task Force (IETF). IPS je model koji prethodi OSI modelu, koji je dosta detaljniji i opisuje više u mrežnom sistemu.

Key architectural principles: (##[lazarc] nastavi ##)

3.3 Precision Time Protocol

Precision Time Protocol (PTP) (prevod: protokol preciznog vremena) je protokol koriscen za sinhronizaciju satova preko kompjuterske mreze. U lokalnoj kompjuterskoj mrezi (local area connection), postize se precisnost sata i u rangi ispod mikrosekunde, sto ga cini pogodnim za merenja i kontrolne sisteme.

PTP je originalno definisan u IEEE 1588-2002 standardu, i zvanično nazvan "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" i objavljen 2002 godine. U 2008 godini, IEEE 1588-2002 je objavljen kao preradjeni standard, poznat i kao PTP Version 2, sa poboljšanom tačnošću, preciznošću i robusnošću, međutim nije kompatibilan sa prethodnom verzijom koja je objavljena 2002 godine.

"IEEE 1588 je dizajniran da popuni prazninu koja nije dobro obradjena ni jednim od dva dominantna protokola, NTP i GPS. IEEE 1588 je dizajniran za lokalne sisteme u kojima je potrebna preciznost izvan one koja je dostupna NTP protokolom. Takodje je dizajniran za aplikacije koje se ne mogu nositi sa cenom GPS prijemnika na svakoj uredjaju, ili sa onima u kojima nije moguće dobijanje GPS signala."==== "IEEE 1588 is designed to fill a niche not well served by either of the two dominant protocols, NTP and GPS. IEEE 1588 is designed for local systems requiring accuracies beyond those attainable using NTP. It is also designed for applications that cannot bear the cost of a GPS receiver at each node, or for which GPS signals are inaccessible.-Eidson, John C. (April 2006). Measurement, Control and Communication Using IEEE 1588. Springer. ISBN 1-84628-250-0.

Arhitektura: IEEE 1588 standard opisuje hijerarhijsku master-slave arhitekturu za distribuciju vremena. Pod ovom arhitekturom podrazumeva se distribucija vremena u sistemu koji se sastoji od jednog ili više komunikacionih medijuma (segmenta koji su povezani na mrežu), i jednog ili više izvora tačnog vremena. #Obični uredjaj# Izvor "običnog" vremena ("ordinary clock") je uredjaj sa jednim pristupom mreži i ima jednu od dve uloge, ili je izvor (master) tačnog vremena, ili čeka na tačno vreme (slave) u komunikaciji na mreži. #Sporodni uredjaj# Granicni sat (Boundary clock) ima više pristupa, na različite mreže, i može precizno sinhronizovati jedan segment mreže na drugi. Master sinhronizacije se bira za svaki segment mreže u sistemu. #Glavni uredjaj# Referentno vreme koje se uzima za izvor sinhronizacionog sata se zove Grandmaster clock. Grandmaster dostavlja sinhronizacione informacije do svih uredjaja koji su povezani na istu mrežu sa

njim. Ukoliko se u nekom delu mreze nalazi Boundary clock on prosledjuje tacno vreme ka ostalim uredjajima koji su direktno na njega povezani.

mora slika ovde kako izgleda arhitektura tacno

Simplifikovano, PTP sistem se sastoji od Ordinary clocks #Obicnog uredjaja# povezanih na jednostavnu mrezu, i bez Boundary clocks #Sporodnih uredjaja#. Grandmaster se bira, i svi ostali uredjaji se direktno sinhronisu na njega.

IEEE 1588-2008 predstavlja Clock koji je povezan sa mreznom opremom koja prenosi PTP poruke. Transparent clock #Transparentni uredjaj# modifikuje PTP poruke koje prolaze kroz uredjaj. Vremenski pecati (TIMESTAMPS) u porukama su modifikovani tako da se uzme u obzir i vreme za koje poruka prolazi kroz dodatne uredjaje u komunikaciji. Ova sema komunikacije povecava distribuciju preciznosti tako sto se kompenzuje promenljivost dostave podataka preko mreze.

PTP tipicno koristi EPOCH vreme, standardno vreme za UNIX sisteme (1 Januar 1970 kao pocetak racunanja vremena). Dok je UNIX vreme bazirano na Univerzalnom vremenu UTC, i mora da postoji sekunda preskoka (#Ovo dodati, mozda u uvod#), PTP je baziran na Medjunarodnom Atomskom Vremenu (TAI - International Atomic Time). PTP Grandmaster daje trenutnu razliku izmedju UTC i TAI, kako bi UTC vreme moglo da se izracuna od primljenog PTP vremena.

mora slika da se vidi razlika izmedju UTC i PTP, TAI

Detalji protokola: Sinhronizacija i obrada u PTP sistemu se postize razmenom poruka preko komunikacionog medijuma. Do sad, PTP standard propisuje samo ove tipove poruka.

- Sync, Follow_Up, Delay_Req i Delay_Resp poruke se koriste u Ordinary i Boundary uredjajima i sluze samo za komunikaciju informacija o vremenu koje se koriste za sinhronizaciju uredjaja na mrezi. - Pdelay_Req, Pdelay_Resp i Pdelay_Res_Follow_Up se koriste u Transparent Clock uredjajima da mere kasnjenje kroz uredjaj tako da se moze iskoristiti u kompenzaciji vremena u sistemu. Transparent Clock i definicija ovih poruka nisu dostupne u IEEE 1588-2002 standardu. - Announce poruke se koriste i Best master clock algorithm u IEEE 1588-2002 standardu za algoritam odredjivanja najtacnijeg sata na mrezi, i to kako bi se izgradila hijerarhija uredjaja i kako bi se odredio Grandmaster. - Management poruke se koriste u upravljanju mrežom za posmatranje performansi na mrezi, konfiguraciju mreze i odrzavanje PTP sistema. - Signalne poruke se koriste u komunikaciji izmedju uredjaja koje nisu vremenski kritične. Signalne poruke su uvedene u IEEE 1588-2002 standard.

Poruke se karakterizuju kao Event i General, odnosno poruke dogadjaja i opste poruke. Event poruke su vremenski kritične i to u preciznosti predaje

i prijema preciznosti vremenskih pečata (TIMESTAMPS) i direktno uticu na distribuciju preciznosti vremena. (JOS JEDNOM POGLEDAJ OVAJ PREVOD). Sync, Delay_Req, Pdelay_Req i Pdelay_resp su poruke dogadjaja. Opste poruke su ubicajene jedinice protokola, zato sto podaci u ovim porukama su od znacaja za PTP, ali njihovi vremenski pecati za predaju i prijem nisu. Announce, Follow_Up, Delay_Resp, Pdelay_Resp_Follow_Up, Management i Signalne poruke su opste poruke.

Prenos poruka: PTP poruke mogu da koriste UDP (User datagram protocol) preko Internet protokola (UDP/IP) za prenos poruka. IEEE 1588-2002, koristi samo IPv4 prenos, ali je ovo prosireno da ukljucuje i IPv6 u IEEE 1588-2008 standardu. U IEEE 1588-2002, sve PTP poruke se salju u Multicast (modulu objavljivanja na mrezi) (#pogledaj opet ovaj prevod#), dok se u IEEE 1588-2008 je uveo kao opciju

3.4 Algoritam najboljeg sata (Best master clock algorithm)

BMC algoritam obavlja deljenu selekciju najboljeg kandidata za tacno vreme prema sledecim karakteristikama:

- Identifikator: Univerzalni jedinstveni identifikator za sat. Tipicno je baziran na MAC adresi uredjaja.
- Kvalitet: Obe verzije IEEE 1588 standarda pokusavaju da kvantifikuju kvalitet sata na osnovu ocekivanih devijacija u vremenu, tehnologije koja je koriscena za implemntaciju vremena ili lokacije u hijerarhiji satova, u semi kvaliteta satova (clock stratum scheme).

- Prioritet: Administrativno dodeljen prioritetni znak koji BMC koristi kako bi sto bolje odredio Grandmaster u PTP domenu. Dok je IEEE 1588-2002 standard imao samo jednu logicku promenljivu kako bi odredio prioritet, IEEE 1588-2008 ima dva 8-bitna polja prioriteta.

- Varijansa: Procena stabilnosti sata zasnovana na zapazanju njegovog ucinka prema PTP refernci.

IEEE 1588 koristi hijerarhijski algoritam selekcije zasnovan na sledecim osobinama, u naznacenom redosledu:

- Prioritet 1: korisnik moze dodeliti specifican staticki dizajniran prioritet svakom satu, pre svega odredjujuci prioritet medju njima. Manje vrednosti prioriteta oznacavaju veci prioritet.
- Klasa: Svaki sat je clan odredjene klase, svaka klasa dobija svoj prioritet.
- Preciznost: Preciznost izmedju sata i UTC, u nanosekundama.
- Varijansa: Varijabilnost sata.
- Prioritet 2: Definisan prioritet, definisuci redosled rezervne kopije u slucaju da drugi kriterijumi nisu dovoljni. Manje vrednosti prioriteta oznacavaju veci priori-

tet. - Jedinstveni identifikator: selekcija zasnovana na MAC adresi se koristi kao metod odlucivanja kada su sve ostale osobine iste.

Svojstva sata se daju u IEEE 1588-2002 porukama za sinhronizaciju (Sync messages) i u IEEE 1588-2008 u porukama za oglasavanje (Announce messages). Trenutni Master clock (##[lazar] [prevod]##) prenosi sve informacije u rednovnim intervalima. Sat koji sebe smatra boljim od trenutnog Master sata prenosi ove informacije kako bi se pozvali svi uredjaji za porucnu Master sata. Kada trenutni Master prepozna bolji sat, tada Master sat zaustavlja emitovanje poruka za sinhronizaciju (Sync Messages), ili poruke oglasavanja (Announce messages), u zavisnosti od verzije protokola, i bolji sat preuzima ulogu Master sata. BMC algoritam uzima u obzir samo osobine koje su vec poznate, i koje su deklarirali sami satovi, i ne uzima u obzir kvalitet veze na mrezi.

3.5 Sinhronizacija (Synchronization)

Koristeci BMC algoritam, PTP bira Master sat za IEEE 1588 domen i za svaki segment mreze unutar tog domena. Satovi odredjuju razliku izmedju njih (offset) i Master-a na mrezi. Neka promenljiva t predstavlja fizicki vreme. Za dati Slave uredjaj, razlika $o(t)$ u vremenu t se definise kao: $o(t) = s(t) - m(t)$ gde $s(t)$ predstavlja vreme mereno satom na Slave uredjaju u vremenu t , dok $m(t)$ predstavlja vreme mereno satom na Master uredjaju u vremenu t .

Master uredjaj periodicno objavljuje (Broadcasts) trenutno vreme kao poruku ostalim uredjajima na mrezi. IEEE 1588-2002 protokolom je definisana objava vremena na svaku sekundu. Dok je IEEE 1588-2008 protokolom dozvoljeno i do 10 objava vremena u jednoj sekundi.

[lazar] [slika] sync

Svaka objava vremena krece u vremenskom trenutku T_1 , i to Sync porukom koju salje Master uredjaj svim uredjajima u domenu. Uredjaj koji prima ovu poruku pamti vreme T'_1 u kom je primio Sync poruku. Master moze naknadno poslati Follow_up poruku u kojoj ce se nalaziti tacno vreme T_1 u kom je poslata prethodna poruka. Nemaju svi Master uredjaji sposobnost da posalju tacne vremenske oznake unutar Sync poruke. Tek nakon sto je prenos završen, oni mogu dobiti tacne vremenske trenutke stizanja Sync poruke iz hardvera za povezivanje na mrezu. Master uredjaji sa ovim ograncenjima salju Follow_up poruke kako bi preneli vreme T_1 . Master uredjaji koji poseduju PTP mogucnosti unutar hardvera za povezivanje mogu ubaciti tacne vremenske oznake unutar Sync poruka, i ne moraju koristiti Follow_up poruke.

Kako bi se tacno sinhronizovali na Master uredjaj, satovi moraju individualno odrediti vreme prenosa poruka kroz medijum za povezivanje. Vreme progresije poruke kroz medijum za povezivanje se radi merenjem vremena koje je potrebno da poruka ode od svakog uredjaja do njihovog Mastera u domenu, i da se vrati nazad. Ovu razmenu iniciraju Slave uredjaji i pri tome mere vreme progresije poruke d . Razmena poruka pocinje tako sto Slave uredjaj salje Delay_Req poruku u vremenskom trenutku T_2 ka svom Master uredjaju. Master uredjaj primi ovu poruku, i kao odgovor posalje tacnu vremensku oznaku kada je primio Delay_Req poruku. Poruka odgovora Delay_Resp sadrzi tacno vreme T'_2 u kome je primljena poruka Delay_Req.

Nakon razmene ovih poruka Slave uredjaj ima spoznaju o cetiri vremenska trenutka T_1, T'_1, T_2 i T'_2 .

Ukoliko je d vreme koje je potrebno Sync poruci da prodje kroz medijum za povezivanje, a \tilde{o} konstantna razlika satova izmedju Master i Slave uredjaja, onda je:

$$T'_1 - T_1 = \tilde{o} + d \quad (3.1)$$

i

$$T'_2 - T_2 = -\tilde{o} + d \quad (3.2)$$

Odakle je:

$$\tilde{o} = \frac{1}{2}(T'_1 - T_1 - T'_2 + T_2) \quad (3.3)$$

Sada dva uredjaja znaju koliki je ofset \tilde{o} prilikom prenosa i mogu se ispraviti tako da budu u skladu sa Master uredjajem.

Jedna pretpostavka je da se prenos poruka odvija u periodu vremena koji je tako mali, da se razlika moze smatrati konstantnom u tom periodu. Jos jedna pretpostavka je da je vreme koje je potrebno da poruka stigne od Master do Slave uredjaja ista kao i u obrnutom smeru. I na kraju, pretpostavka je da i Master i Slave uredjaji mogu da precizno mere vremenske trenutke u kojima salju ili primaju poruke. Stepem primene ovih pretpostavki utice na to koliko ce se dobro sinhronizovati dva uredjaja.

[lazarc] Pogledaj jos da se ubaci sa Teams za Valeo

Glava 4

Operativni sistem

4.1 FreeRTOS

FreeRTOS je kernel operativnog sistema koji radi u realnom vremenu, i to za namenske sisteme, i moze se koristiti na preko 35 mikrokontrolera.

(Wiki) Implementacija: FreeRTOS je dizajniran tako da bude mali i jednostavan. Kernel (srce operativnog sistema) se sastoji od samo 3 fajla, i pisan je u C programskom jeziku. Kako bi se kod napravi da bude citljiv, lako portabilan, i kako bi se lako održavao projekat, pisan je uglavnom u C programskom jeziku, sa izuzetkom da su neke funkcionalnosti napisane u assembleru, gde je to bilo potrebno, i to uglavnom rutine u Scheduler-u (Rasporedjivacu??? `##[lazarc] [prevod]##`) koje su specificne za samu arhitekturu.

FreeRTOS omogućava koriscenje metoda za stvaranje vise programskih niti, ili Taskova, stvaranje mehanizama za Sinhronizaciju niti, Mutexa, Semafora i softverskih tajmera. Takodje, postoje mogucnosti koriscenja FreeRTOS-a i za aplikacije niske potrosnje. Aplikacije koje se koriste FreeRTOS mogu biti kompletno staticki alocirane. Alternativno RTOS objekti mogu dinamički alocirane sa 5 sema alokacije memorije i one cine:

- samo alocirati;
- alocirati i osloboditi sa jednostavnim, brzim algoritmom;
- kompleksnija ali brza alokacija i oslobadjanje uz algoritam spajanja susednih memorijskih blokova;
- alternativa za jos kompleksiniju semu koja ukljucuje spajanje susednih memorijskih blokova koja omogućava da hip (HEAP) bude podeljen na vise memorijskih delova;

- i na kraju C biblioteka za alociranje i oslobađanje sa zaštitom međusobnog isključivanja.

Unutar FreeRTOS-a ne postoji ni jedan od složenijih svojstava operativnih sistema koji se uobičajeno mogu naći u operativnim sistemima poput Linux-a ili Microsoft Windows-a, kao što su drajveru uređaja, napredno upravljanje memorijom, korisnički nalozi, i umrežavanje. Akcenat ovog operativnog sistema je na kompaktnosti i brzini izvršavanja. O FreeRTOS-u se može misliti kao o "biblioteci niti" više nego kao o "operativnom sistemu". (## [lazar] , although command line interface and POSIX-like I/O abstraction add-ons are available ##)

FreeRTOS implementira više niti tako što postoji jedan program koji poziva metode niti u jednakim kratkim vremenskim intervalima. Metoda promene niti zavisi od prioriteta niti i uključuje round-robin semu promene niti. Uobičajen interval promene je do 1/1000 sekunde do 1/100 sekunde, i to kroz prekid hardverskog tajmera, ali interval promene se često menja tako da zadovolji potrebe specifične aplikacije.

FreeRTOS Documentation: FreeRTOS je idealno sklopljen za duboke namenske aplikacije u realnom vremenu koje koriste mikrokontrolere ili male mikroprocesore. Ovaj način projektovanja aplikacija uključuje kombinaciju kako strogih zahteva za realnim vremenom u aplikaciji, tako i manje strogih.

Strogi zahtevi za aplikacijama realnog vremena su oni u kojima postoji vremenski rok u izvršavanju, i ako se taj rok probije, doći će do apsolutnog pada funkcionalnosti sistema. Na primer, airbag u kolima ima potencijal da napravi više štete nego dobrog ukoliko je odziv sistema samo malo sporiji nego što treba.

FreeRTOS je kernel realnog vremena (ili rasporedjivac(##[lazar] [prevod]##) realnog vremena) na koji se nadograđuje aplikacija tako da ispuni stroge zahteve za realnim vremenom aplikacije. To dozvoljava da aplikacija bude organizovana kao kolekcija nezavisnih programskih niti. Na procesoru koji ima samo jedno jezgro, samo jedna programska nit se može izvršavati u jednom trenutku. Kernel odlučuje koja nit se izvršava tako što određuje prioritet koji se dodeljuje svakoj niti. U najjednostavnijem slučaju, dizajner aplikacije može odrediti više prioritete nitima koje implementiraju stroge zahteve za realnim vremenom, a nize prioritete onim nitima koje nemaju tako stroge zahteve za izvršavanjem. Ovim bi se osiguralo da niti koje imaju strožije zahteve, imaju prioritete izvršavanja i pristupa resursima nad ostalim nitima, ali odluke za dodelu izvršavanja nisu uvek tako jednostavne.

Napomena: Unutar FreeRTOS-a se programska nit naziva "task". Tako da će se u daljem tekstu i koristiti naziv Task za programsku nit.

U projektovanju aplikacija za namenske sistema postoji ustaljena praksa

projektovanja aplikacija koja ne zahteva koriscenje kernela za realno vreme, i ove tehnike mogu dati bolje resenje problema. Mada, u kompleksnijim slucajevima, verovatnije je koriscenje kernela za aplikacije u realnom vremenu, i takodje moze biti kombinacija koriscenja kernela, i drugih tehnika projektovanja aplikacije.

Kao sto je vec opisani, prioriteti taskova mogu pomoci da se osigura da aplikacija ispunji sve zahteve, ali kernel moze doneti i neke manje ocigledne beneficije. Neke od njih su navedene ispod:

- **Skracivanje informacija o vremenskom rasporedu (Abstracting away timing information):** Kernel je odgovoran za vreme izvršavanja i dodeljuje API kojim se unutar aplikacije moze upravljati vremenom. Ovim se omogucava jednostavnija strukturiranost koda, i ukupna velicina koda je manja.
- **Održavanje/Prosirivanje (Maintainability/Extensibility):** Uskracivanjem informacija o vremenskom rasporedu rezultuje u manjim zavisnostima izmedju modula, i dozvoljava aplikaciji da evoluiru u kontrolisanom i predvidjenom nacinu. Takodje, kernel je odgovoran za rasporedjivanje vremena, tako da performanse aplikacije manje mogu biti promenjene u hardveru na kome se pokrecu.
- **Modularnost (Modularity):** Taskovi su nezavisni moduli, pri cemu svaki od njih mora imati dobro definisanu svrhu.
- **Timski razvoj (Team development):** Taskovi bi trebalo da imaju dobro definisane interfejse, kako bi se lakse razvijali u timovima.
- **Lakse testiranje (Easier testing):** Ako su taskovi dobro definisani kao nezavisni moduli sa cistim interfejsima, mogu biti testirani nezavisno.
- **Ponovno koriscenje koda (Code reuse):** Veca modularnost sa vecom nezavisnoscu koda koji se moze ponovo koristiti sa manje ulozenog truda.
- **Poboljsana efikasnost (Improved efficiency):** Koriscenjem kernela softver se u potpunosti moze prebaciti na opkretanje dogadjajima (event driven programming), i time bi se uštedelo procesorsko vreme koje se trosi na poliranje dogadjaja koji se ne dogadjaju. Kod se pokrece samo ukoliko postoji nesto sto je potrebno uraditi.

Protiv poboljsane efikasnosti stoji to da je potrebno pocesuirati RTOS prekid, i promeniti izvršavanje sa jednog taska na drugi. Kako god, i

aplikacije koje ne koriste RTOS normalno uključuju neku formu prekida.

- **Idle time (##[lazarc] [prevod]##):** Idle task je task koji se automatski kreira prilikom startovanja Rasporedjivaca (##[lazarc] [prevod]##). I izvršava se kad nema taskova unutar aplikacije koji bi se izvršavali. Ovaj task se može koristiti za merenje procesorske moci koja se troši, za izvršavanje provera u pozadini, ili da jednostavno pokrene režim smanjene potrošnje u sistemu.
- **Upravljanje snagom (Power management):** Efikasnost koja se dobija koriscenjem RTOS-a dozvoljava procesoru da provede više vremena u režimu smanjenje potrošnje.

Potrošnja se može značajno smanjiti time što procesor odlazi u režim smanjenje potrošnje kad god je pokrenut Idle task. FreeRTOS takodje ima i specijalni tick-less mod, u kome procesor odlazi u režim smanjene potrošnje na duže.

- **Fleksibilno upravljanje prekidima (Flexible interrupt handling):** Upravljanje prekidima se može držati veoma kratko tako što se odlaze obrada bilo kog taska koji je kreirao sam dizajner, ili taska unutar FreeRTOS-a.
- **Razliciti zahtevi za obradom (Mixed processing requirements):** Jednostavni oblici dizajniranja programa mogu se postići mesanjem periodičnog, kontinualnog i procesiranja pokretanog događajima. Pored toga, ispunjavanje strogih i manje strogih zahteva za realnim vremenom u aplikacijama može se postići izborom odgovarajućih taskova i prioriteta prekida.

4.2 lwIP

lwIP (light-weight IP) je implementacija TCP/IP komplet-a (## [lazarc] [prevod] suite ##) je originalno napisao Adam Dunkels u Computer and Networks Architectures (CNA) laboratoriji na SHvetskom institutu za kompjuterske nauke (Swedish Institute of Computer Science) ali ga sad aktivno razvija tim inženjera sirom sveta kojim rukovodi Kieran Mansley.

lwIP je open-source projekat koji je besplatan za preuzimanje i koriscenje (pod BSD licencom), pisan u C programskom jeziku i može se preuzeti sa internet stranice tima koji ga razvija.

Fokus lwIP implementacije TCP/IP je da se smanji koriscenje RAM memorije i da se i dalje dobija potpuna funkcionalnost TCP. Ovim lwIP postaje interesantan za koriscenje u namenskim sistemima koji raspolazu sa RAM memorijom od nekoliko desetina kilobajta kB i prostorom od oko 40 kilobajta u ROM memoriji.

Od kada je prvi put objavljen, lwIP izaziva dosta interesovanja, i danas se koristi u dosta komercijalnih projekata. lwIP je do sad iskoriscen na mnogim platformama i operativnim sistemima, i moze se koristiti bez i sa operativnim sistemom. U ovoj implementaciji, lwIP se koristi u okviru FreeRTOS operativnog sistema, kao jedan njegov deo.

LwIP je veoma modularan i ima podrsku za dosta protokola, od kojih vecina moze da se ukloni za manju velicinu koda.

- *Mrežni protokoli i protokoli veze: (Link and network protocols)*
 - **ARP**: protokol veze koji se koristi za prevod prirodne hardver adrese ("MAC adresa") u IP adresu
 - **IPv4**: dominantni mrežni protokoli koji se koristi danas, posebno za Internet
 - **IPv6**: naslednik IPv4, koji, narocito, prosiruje velicinu IP adrese na 128 bita
 - **ICMP**: kontrolni protokol za IP
 - **IGMP**: protokol za upravljanje grupa unutar IP-a
- *Transportni protokoli: (Transport protocols)*
 - **UDP**: protokol bez prikljucka, i bez mehanizma pouzdanosti
 - **TCP**: protokol orjentisan ka konekciji, za kontinualni tok podataka (streaming)
- *Protokoli visokog nivoa: (High-level protocols)*
 - **DHCP**: dobijanje IP adrese sa podrskom servera
 - **AUTOIP**: dobijanje IP adrese bez podrške servera
 - **SNMP**: koriscen za nadgledanje stanja mreže
 - **PPP**: koriscen za stvaranje direktne konekcije izmedju dva cvora na mrezi

IPv4: (##[lazar] opisati u delu za softversku implementaciju ##)

lwIP pruza tri API-a (Application Program's Interface) za programe koji komuniciraju sa TCP/IP kodom:

- low-level čore"/čallback"ili "raw"API
- dva API-a viseg nivoa (sekvencijalni API-i):
 - netconn API
 - socket API

Sekvencijalni API pruža način za obično, sekvencijalno programiranje koje koristi lwIP stek (`##[lazar] stack ##`). Model izvršavanja je baziran na blokirajućoj otvori-pročitaj-upisi-zatvori paradigmi. Kako je TCP/IP stek baziran na događajima, TCP/IP kod i aplikativni program, moraju da se pozivaju sa razlicitim kontekstima izvršavanja, u razlicitim nitima.

Prilikom mesanja sekvencijalnog i širovog"API-a u programima, treba biti pazljiv. Funkcije koje pripadaju nesekvencijalnom API-u u stvari mogu biti pozvane iz glavne tcpip_thread niti. Takodje, registrovanje programski rutina (ili inicijalizovanje delova u lwIP) mora biti odradjeno unutar tog konteksta (na primer, u vreme startovanja aplikacije u tcpip_init_callback rutini ili u vreme izvršavanja unutar tcpip_callback rutine).

Jos neke cinjenice o API-ima koje uticu na koriscenje lwIP steka:

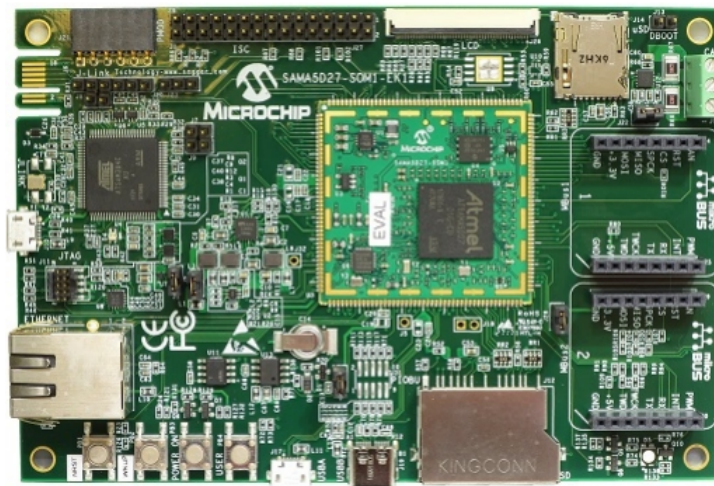
- **netconn- i raw-API su samo unutar lwIP-a** : kod koji koristi ovaj API se ne može koristiti u drugim stekovima koji imaju iste mogućnosti kao lwIP (na primer uIP i td.)
- **socket API** je u suprotnosti sa gore navedenom stavkom, napravljen je tako da je kompatibilan i može se koristiti u drugim stekovima.
- **socket- i netconn-API** su sekvencijalni API-i koji zahtevaju programske niti (jedna nit je za aplikaciju koja koristi API, jedna nit upravlja tajmerima unutar steka, paketima koji dolaze, i td.)
- **raw API** koristi mehanizam povratnih rutina (na primer. aplikacija poziva rutinu kada dodje novi podatak). Ukoliko se koristi u programu koji radi na sekvencijalni način, može biti teže korišćenje.
- **raw API** daje bolje performanse kako ne zahteva promenu izvršavanja programskih niti.
- **raw API i netconn API** podržavaju zero-copy ¹ kako za TX tako i za RX. (kako za predaju, tako i za prijem)

¹predstavlja operaciju pri kojoj procesor ne vrši kopiranje podataka iz jedne memorijske oblasti u drugu. Ovo se često koristi kako bi se sačuvali ciklusi procesora i propusnog opsega memorije prilikom prenosa kontinualnih podataka (datoteka, fajlova) preko mreže.

Glava 5

Hardverska implementacija

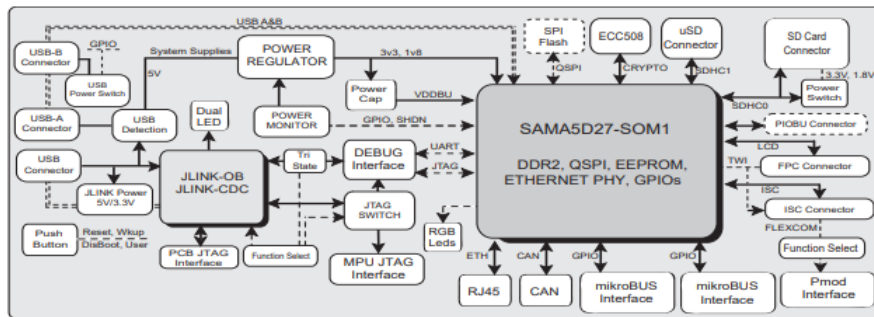
Razvojno okruženje koje je korisno za hardversku implementaciju je razvojna ploca SAMA5D27-SOM1-EK1 proizvođača Microchip. Na ploči se nalazi SAMA5D27 SOM (System on Module) modul koji je ključan za implementaciju. Na modulu se nalazi SAMA5D27-D1G-CU SIP (System in Package) koji sadrži 1 Gbit DDR2 SDRAM memorije. Modul nudi pouzdanu i niskobudžetnu platformu za razvoj namenskih računarskih sistema koji će na kraju i završiti u finalnoj proizvodnji, kao i malu formu, dopunjenu sa velikim brojem interfejsa koji se mogu koristiti u delu projektovanja krajnjeg sistema.



Slika 5.1: Razvojna ploca SAMA5D27-SOM1-EK1 (pogled odozgo)

SOM je potpuno opremljen industrijski sertifikovan kompjuter dizajniran za integraciju korisničke aplikacije. SOM modul je namenski napravljen kao

mala hardverska platforma opremljena širokim spektrom modula za brzo povezivanje kako bi podržali projektovanje podrške za razne IoT (Internet of Things) aplikacija, prenosnih uređaja, ali i aplikacija u industrijske svrhe. SOM integrise 1Gbit DDR2 SDRAM i QSPI memoriju kao 10/100 Mbit Ethernet interfejs. Takodje, SOM poseduje i 128 GPIO pinova ([#] [lazarc]) koji obezbeđuju pristup SOM-a za različite upotrebe. Svi GPIO pinovi su nezavisni, i mogu se konfigurisati kao ulazi ili izlazi, sa ili bez PULL-UP otpornika. Razvojna ploca poseduje i širok spektar periferija, kao i korisnicki interfejs i nacin za prosirenje funkcionalnosti, ukljucujuci i dva mikroBUS Click interjesa firme Mikroelektronika kojim se dobija mogucnost za prosirenje funkcionalnosti svim modulima koje ova firma nudi u svom asortimanu.

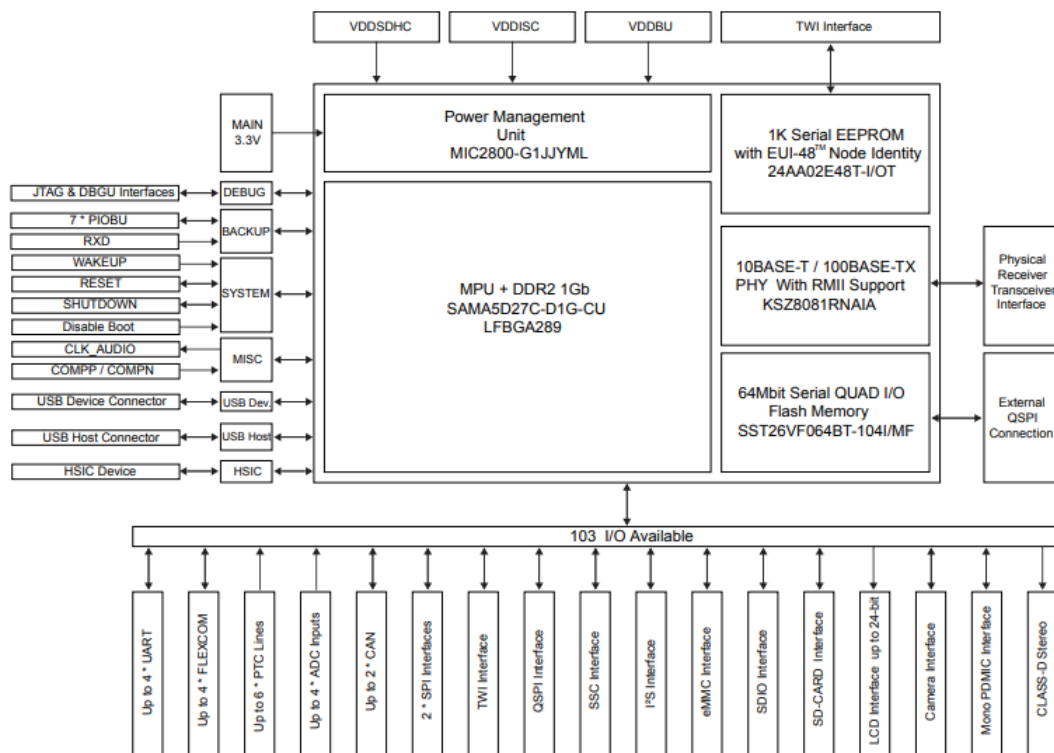


Slika 5.2: Moduli - Razvojna ploca SAMA5D27-SOM1-EK1

Na samom SAMA5D27-SOM1 postoje i:

- Ultra mali SIP (SAMA5D27-D1G-CU) koji sadrzi stedljivi SAMA5D27 Arm Cortex A5 procesor i 1Gbit DDR2 SDRAM memoriju
- SST26VF064 64 Mb QSPI Flash memoriju
- 24AA02E48 2 Kb serijski E2PROM (Electrically Erased Programmed Read Only Memory) sa programiranom EUI identifikacijom pristupa
- MIC2800 cip za kontrolu napajanja
- KSZ8081RNA Ethernet Phy 10/100 MHz RMII

Razvojno okruženje SAMA5D27-SOM1-EK1 je veoma mocno i moze se koristiti za širok spektar aplikacija. Najcesce je izbor za razvoj aplikativnog softvera u namenskim racunarskim sistemima, uz koriscenje Embedded Linux operativnog sistema, i s tim ciljem se i bira. Za ovu implementaciju nije koriscen u tom smislu, vec je koriscen sa drugim operativnim sistemom



Slika 5.3: Moduli - SAMA5D27-SOM1

kojim je moguće istaci neke druge njegove karakteristike. Ali ono što je najviše interesantno za ovu primenu je postojanje TSU (Timestamping unit) za harversku podršku PTP-a. I to u sklopu periferije za povezivanje preko Ethernet-a, čime je omogućeno prepoznavanje PTP poruka koje dolaze na interfejs.

5.1 Time stamping unit - TSU

IEEE-1588 je standard za preciznu sinhronizaciju vremena u lokalnim mrežama. Radi se o razmeni preciznog vremena između dva uređaja u lokalnoj mreži. PTP poruke se mogu prenositi preko IEEE802.3/Ethernet, IPv4 ili IPv6 protokola kako je to već opisano u odeljcima pre. Periferija unutar razvojnog okruženja GMAC označava tacku vremenske oznake poruke (na početku slanja i na kraju slanja) poruke. IEEE 802.1AS je podskup IEEE 1588 standarda. Jedina razlika je u adresi koja služi za slanje PTP poruka svim uređajima u lokalnoj mreži (Multicast). GMAC periferija je dizajnirana tako da prepoznaje poruke koje su ključne za PTP protokol.

Kao što je već navedeno, sinhronizacija između dva uređaja se izvodi u dva stadijuma, određivanje razlike (offset) između dva sata (na strani Master i Slave uređaja), nakon čega se šalje tačno kasnjenje na linijama za prenos podataka, čime se tačno sinhronizuju dva sata. Hardverski moduli koji pomazu u ovoj razmeni poruka određuju tačno vreme kada je poruka stigla, i kada je poruka poslata. Sto je ključno za određivanje vremena kojim se izračunavaju razlika i kasnjenje. Pojava ovih poruka uzrokuje prijavljivanje hardverskih prekida, tako da je moguće operisati sa vremenima koja se dobijaju tako da se dobija fina sinhronizacija vremena dva uređaja. Podrška ovom protokolu u hardveru se ogleda u postojanju TSU (Timestamping unit) periferije koja se sastoji od tajmera i registara u koje se smestaju tačna vremena u trenucima kada stignu ili odu poruke koje su ključne za dobijanje tačnog vremena. Prekid se prijavljuje kada se ovi registri osveže vremenom slanja ili primanja poruke bitne za PTP protokol. Tajmer je implementiran kao 94-bitni broj, u kome visih 48 bita broje sekunde, narednih 30 bita nanosekunde, i preostalih 16 bita broje vreme ispod nanosekunda. Nizih 46 bita se prevrte (roll-over) kad se izbroji tačno jedna sekunda. Takođe, prijavljuje se hardverski prekid nakon što se izbroji jedna sekunda. Vrednosti tajmera, kao i parametri za njegovo upravljanje, se mogu menjati softverski preko APB interfejsa. Tajmer se pokreće MCK taktom, koji je glavni takt unutar procesora, i njime se takođe može manipulirati. Promene podešavanja tajmera unutar TSU su od velike važnosti za tačnu sinhronizaciju dva uređaja. Ova periferija najviše utiče na tačno vreme koje je potrebno sinhronizovati, i takođe daje informacije o tačnom vremenu na uređaju koji se koristi. Više detalja o samoj implementaciji i korišćenju ove periferije biće dato u delu softverske implementacije.

Glava 6

Softverska implementacija

Glava 7

Zaključak