Билет №9

Вградени системи

Точка 1: Дефинира и обяснява основни понятия във вградените системи. Посочва и различава основни компоненти във вградените системи. Обяснява характеристиките и особеностите на вградените системи. (16т.)

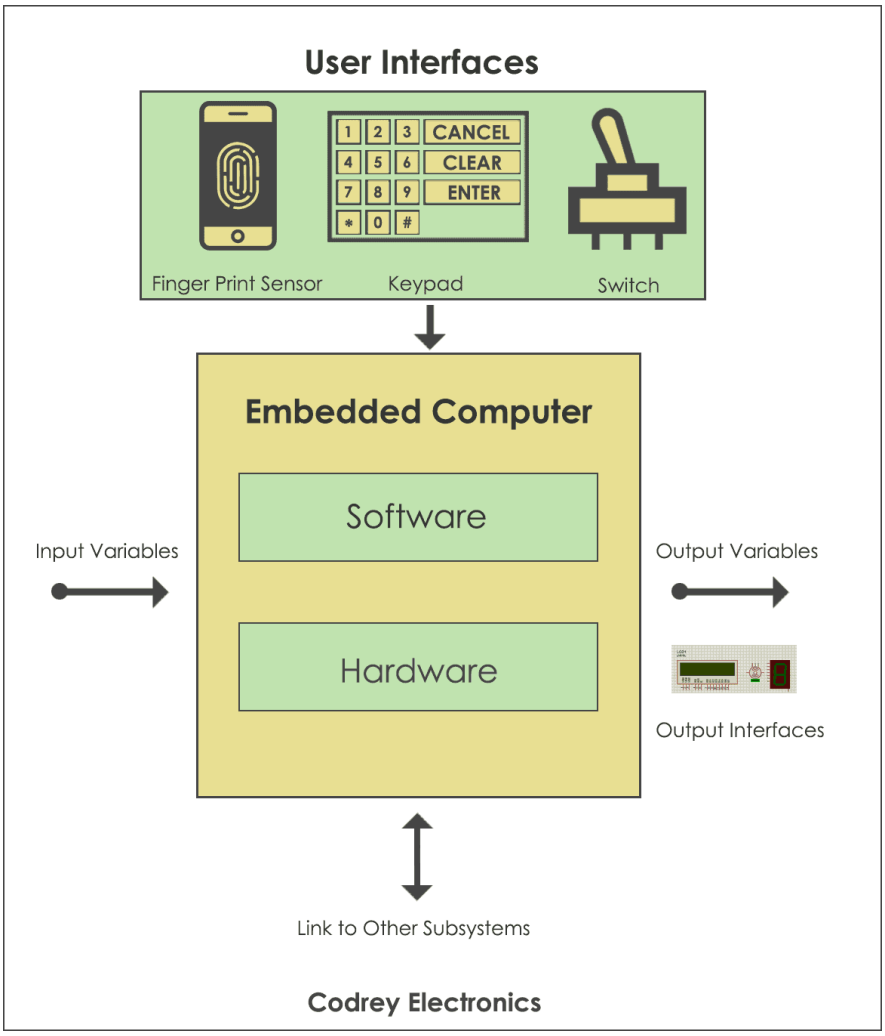
Точка 1.1: Дефинира основни понятия във вградените системи. (2т.)

Вградена система е част от цялостна система от устройства, която включва хардуер, като електрически и механични компоненти. Вградената система е обикновено проектирана да има една или две функции, за разлика от компютъра. Възможно е да бъде както самостоятелна система, така и част от по-голяма такава.

Една микрокомпютърна система може да бъде реализирана на базата на микропроцесор или на базата на микроконтролер. Когато се реализира на базата на микропроцесор, са необходими и допълнителни устройства – памети, входно/изходни портове, схеми за управление на прекъсванията и др.

Основните компоненти в една вградена система са микропроцесор, захранване, памет, таймери, серийни портове за комуникация и др. Някои основни негови характеристики са усъвършенствана функционалност, операция в реално време, ниски производствени разходи, процесор и памет и др.

Вградените системи се използват навсякъде около нас. Примери за тях могат да бъдат прахосмукачките, колите (техните системи като ABS и др.), смартфоните, светофарите, охранителни системи (СОТ) и много други. Превърнати са в неразделна част от живота ни, тъй като са необходими не само в ежедневието ни, но и в медицината, в промишлеността. Чрез тях ние улесняваме живота си. В медицината вградените системи са от изключителна необходимост, както за измерване на пулс, така и за изследвания, животоспасяващи системи и мн. др.



Точка 1.2: Обяснява основни понятия във вградените системи. (2т.)

Микрокомпютърът е най – важният компонент във вградената система. Той е малък, сравнително евтин компютър, в който ролята на процесор (CPU) се изпълнява от микропроцесор. Микрокомпютърът представлява програмируемият модул, в който е заредена програмата за изпълнение на цялата система. В зависимост от изискванията за управление и мониторинг на системата микрокомпютърът може да бъде едноплатков компютър, микроконтролер (MCU), цифров сигнален процесор (DSP) и др.

Сензорите (датчиците) са устройства, които преобразуват определена физическа величина в електрически сигнал, който може да бъде разчетен от програмируемото устройство. Пример за сензори са сензори за температура, сензор за влажност, сензор за анализ на газове, pH сензор...

Микропроцесорът е процесор, затворен в интегрална схема. Казано с други думи микропроцесорът е физическа реализация на процесор под формата на електронен компонент. Микропроцесорът също не е завършена компютърна система, а е само част от нея

Микроконтролерът е едночипова система, съчетаваща в себе си микропроцесор, тактов генератор, оперативна памет. Той е един от най – използваните модули в съвременните вградени системи. При него консумацията на енергия е ниска, притежава малки размери и е с ниска цена, за сметка на малката му изчислителна мощност. Основните параметри на микроконтролерът са разреденост [bits], тактова честота [MHz], производителност [MIPS], размер на оперативната памет [RAM, kB], размер на програмната памет [Flash, kB], размер на постоянната памет [EEPROM, kB], входно – изходни портове, комуникационни интерфейси, захранващо напрежение [V], цена.

Изпълнителните устройства извършват определена полезна работа под управление на програмируемото устройство. Пример за такива устройства са електродвигатели, дисплеи, светлинни и звукови индикации, нагреватели, бъркалки, компресори, помпи, пневматични цилиндри...

Преобразователните схеми служат за допълнителна обработка на електрическите сигнали. Напр. изходите на някои сензори имат много ниски стойности на изходното напрежение и за да могат да бъдат прочетени от програмируемото устройство е необходимо електрическият сигнал предварително да се усили. Пример за такива схеми са: усилватели, филтри, делители на ток/напрежение, схеми за смяна нивото на сигнала, драйвери за електромотори.

Пин – извод на микропроцесора със съответни възможности за приемане или подаване на цифрови или аналогови сигнали.

АЛУ (аритметично – логическо устройство) – извършва всички аритметични операции в микрокомпютъра.

Точка 1.3 Посочва и различава основни компоненти във вградените системи. (4т.)

Една вградена система основно се нуждае от захранване, процесор, памет, серийни комуникационни портове и специфични вериги за системно приложение.

Захранването е ключовият компонент за осигуряване на захранване към веригата на вградената система. Обикновено вградената система изисква захранване от 5 V или може да бъде в диапазон от 1,8 до 3,3. V. Източникът на захранване може да бъде батерия или може да бъде осигурен от стенен адаптер. Захранването се избира според изискванията на потребителя и изискванията на приложението. Електрозахранването трябва да бъде гладко и ефективно, така че да може да се осигури непрекъснато захранване на вградена система. Захранването също трябва да позволява разсейване и трябва да бъде възможно най-ефективно.

За всяка вградена система **процесорът** действа като мозък на системата. Процесорът е отговорен за решаването на производителността на вградената система. На пазара има множество видове процесори, които могат да бъдат избрани според изискванията на потребителя. Вградената система може да действа като микроконтролер и микропроцесор. Процесорът може да бъде 8-битов процесор, 16-битов процесор и 32-битов процесор. Колкото по-малък е битът, толкова по-малко е приложението за вградени системи. Когато се използват големи приложения, във вградената система е необходим процесор с по-висок бит. Процесорът

трябва да е много бърз, цената трябва да е минимална, производителността трябва да е добра, така че функциите да могат да се изпълняват много бързо във вградена система.

Тъй като във вградената система се използват различни микроконтролери, **паметта** присъства в самия микроконтролер. Има основно два вида памет RAM (памет с произволен достъп) и ROM (памет само за четене). Тъй като RAM паметта е променлива памет, данните могат да се съхраняват временно в паметта и когато системата се изключи, данните се губят от паметта. Паметта само за четене се класифицира като кодова памет. ROM се използва за съхраняване на програмата и когато системата е включена, вградената система извлича код от ROM паметта.

В някои от приложенията винаги има изискване за забавяне, което трябва да се предостави в приложението. Например, в приложенията за LED дисплеи има изискване за известно забавяне, така че светодиодът да може да продължи да мига. И за това **таймер и брояч** могат да се използват във вградената система. Програмирането може да се извърши по такъв начин, че забавянето да генерира вградената система. Времевият интервал на забавяне може да бъде решен чрез използване на кристалния осцилатор и системната честота, така че забавянето да може да се генерира според изискванията на потребителя.

Комуникационният порт е типът интерфейс, който се използва за комуникация с други видове вградени системи. Във вградената система има множество видове комуникационни портове като UART, USB, Ethernet, RS-485 и много други. Когато вградена система се използва в приложение с малък мащаб, комуникационните портове могат да се използват от микроконтролера. Има и серийни протоколи, които могат да се използват за изпращане на данни от една системна платка към друга платка.

Когато се използва вградената система, **входът** е необходим за взаимодействие със системата. Входът към вградената система може да бъде предоставен от сензора или от самия потребител. Процесорът, използван във вградената система, може да бъде базиран на вход и изход. Трябва да се направи правилната конфигурация за използване на входния и изходния порт. Във вградената система има фиксирани входни и изходни портове, така че устройствата да могат да се свързват само към посочените портове. Например P0, P1, P2 и много други.

Когато вградената система е проектирана, има няколко хардуерни компонента, които могат да се използват за целите на проектиране. Изборът на **верига** зависи изцяло от приложението, използвано за вградените системи. Например, в приложенията с температурни сензори има изискване за температурни сензори за измерване на температурата.

Това са основните **хардуерни компоненти** при вградените системи.

Основни **софтуерни изисквания** при вградените системи:

Асемблер – той е съден, когато езикът за програмиране, който е съден за проектиране на приложението, е асемблер. След това програмата на асемблерния език се преобразува в HEX код, така че да може да бъде допълнително обработена. И след като напише кода, програмистът се използва за запис на програмата в чипа.

Емулаторът е софтуерен инструмент, който се използва за изпълнение на функциите на хост системата. Всички компоненти могат да се контролират от инструмента за емулатор. Емулаторът също се използва за намиране на грешки и за отстраняване на грешки в кода. Също се използва за прехвърляне на кода от хост системата към целевата система.

Компилаторът е вид софтуер, който се използва за преобразуване на езика за програмиране в някакъв език, който целевата машина може да разбере и да изпълни функциите. Основната употреба на компилатора е да прехвърли кода от високо ниво на някакъв език от ниско ниво. Езиците от ниско ниво включват машинен код, обектен код и асемблер.

Точка 1.4: Обяснява характеристиките и особеностите на вградените системи. (4т.)

Вградените системи изискват изпълнение в реално време. Това означава, че извършват някакъв вид изчисления или задачи в момента на изпълнение. Поради тази причина е необходимо да имат висока

надеждност, съответно се разработват около операционна система в реално време. Пример за това може да бъде анти – блокиращата система при автомобилите. В реалния свят ние няма как да знаем предварително кога ще ни бъде необходимо рязкото спиране. За целта вградената система ни помага, като е програмирана да извърши изпълнението в реално време.

Вградените системи обикновено се проектират за конкретна задача/задачи. Трябва да бъдат свързани с периферни устройства. Те са по – бавни от обикновените компютри, но по – практични заради малкия си размер. Използват се в ел. уреди, програмират се по – лесно и притежават по – малко оперативна памет. Поради тази причина са достъпни, тъй като имат по – ниска цена.

Точка 2: Прави заключения и изводи за разликите между микропроцесор и микроконтролер. (8т.)

Микропроцесор:	**Микроконтролер:**
Микропроцесорът изпълнява функцията на централен процесор (CPU), разположен в една интегрална схема (IC). За да се изгради или проектира система (компютър), микропроцесорът трябва да бъде свързан външно към някои други компоненти като памет (RAM и ROM) и входно / изходни портове.	Микроконтролерът може да се разглежда като едночипов компютър, който съдържа микропроцесор и други необходими компоненти. ИС на микроконтролера има вградена памет (както RAM, така и ROM), заедно с някои други компоненти като I / O устройства и таймери.
Микропроцесорите се използват главно при проектирането на системи с общо предназначение от малки до големи и сложни системи като суперкомпютри.	Микроконтролерите се използват в устройства с автоматично управление , за обработка на задачи в реално време, тъй като са еднократно програмирани, самодостатъчни и ориентирани към изпълнение на конкретни задачи устройства.
Микропроцесорите са основни компоненти на персоналните компютри.	Микроконтролерите обикновено се използват във вградени системи
Изчислителният капацитет на микропроцесора е много висок. Следователно може да изпълнява сложни задачи. Микропроцесорите имат интегриран математически копроцесор. Сложни математически изчисления, които включват плаваща запетая, могат да се извършват с голяма лекота.	По-малък изчислителен капацитет в сравнение с микропроцесорите. Обикновено се използва за по-прости задачи. Микроконтролерите нямат математически копроцесори. Те използват софтуер за извършване на изчисления с плаваща запетая, което забавя действието на устройството.
Микропроцесорна система може да изпълнява множество задачи.	Система, базирана на микроконтролер, може да изпълнява единични или много малко задачи.
Основната задача на микропроцесора е да изпълнява цикъла с инструкции многократно. Това включва извличане, декодиране и изпълнение.	В допълнение към изпълнението на задачите за извличане, декодиране и изпълнение, микроконтролерът също така контролира своята среда въз основа на изхода от цикъла на инструкциите.
Като цяло консумацията на енергия и разсейването са високи поради външните устройства. Следователно се изисква външна охладителна система.	Консумацията на енергия е по-малка.
Тактовата честота е много висока, обикновено от порядъка на Giga Hertz.	Тактовата честота е по-рядко в порядъка на мега херца.

От разликите между микропроцесора и микроконтролера, става ясно, че микропроцесорът не може да замени микроконтролера и обратно. И двете технологии имат своя уникален начин за използване и приложение.

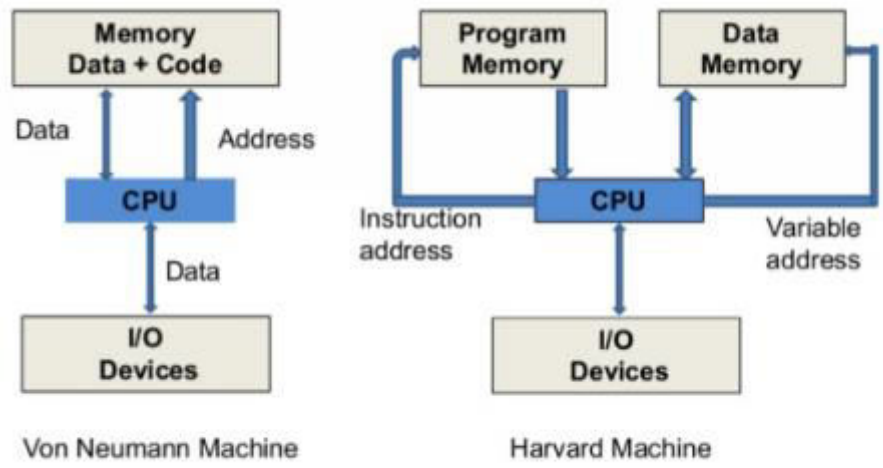
Точка 3: Обяснява и различава видовете архитектури. (8т.)

Точка 3.1: Обяснява видовете архитектури. (4т.)

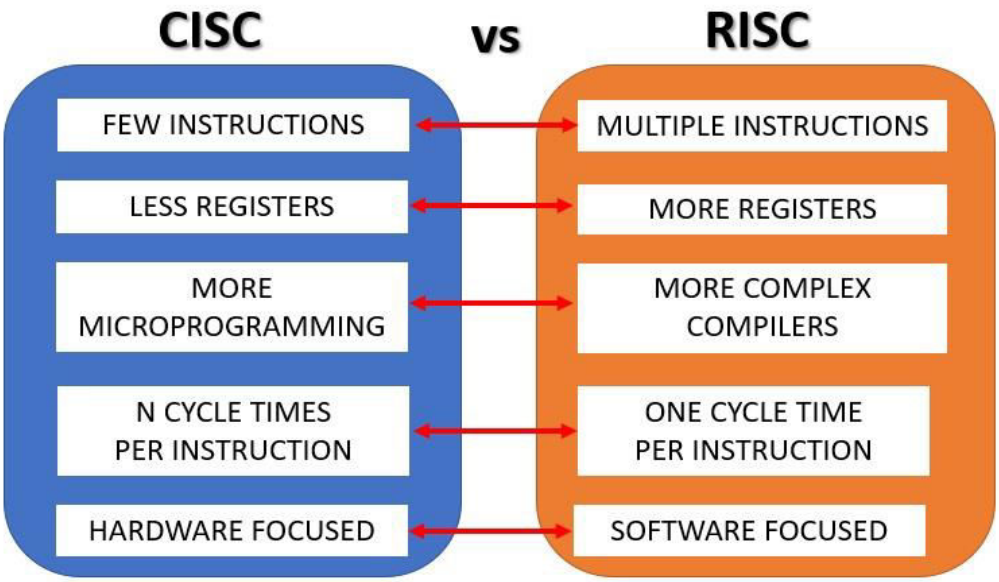
Точка 3.2: Различава видовете архитектури. (4т.)

Фон Ноймановата е първата архитектура, описана от Фон Нойман през 1945 година и по късно създадена. Идеята и е да имаме една основна памет в която да пазим информация и команди за централния процесор (central processing unit), той да ги преработва и да ги праща обратно в тази памет чрез така наречените шини (busses). Взима се информация отвън, преработва се през централния процесор и той я праща при паметта. Проблема при това, е че тази цел отнема две стъпки. Пращане на команда от паметта през шината за данни към централния процесор, той я обработва и я връща пак през шината за данни. Това прави тази архитектура по бавна.

Харвардската архитектура оправя този проблем, като разделя командите от основната памет. Така има памет за инструкции и памет за данни. За да може инструкциите да стигнат до central processing unit, им трябва шина, която е наречена control bus. Така може в една стъпка(един clock импулс) да се прати команда от паметта за инструкции към централния процесор, да се обработи и да се прати към основната памет.



CISC (Complex Instruction Set Computing) архитектурата се състои от пълен набор от инструкции, които са сложни, по-големи, имат повече изчислителна мощ и т.н. Една инструкция CISC може да се използва за изпълнение на няколко операции на ниско ниво, многостепенни операции и множество режими на адресиране. Времето за изпълнение на тези инструкции е дълго. Intel X86 е пример за CISC архитектура.

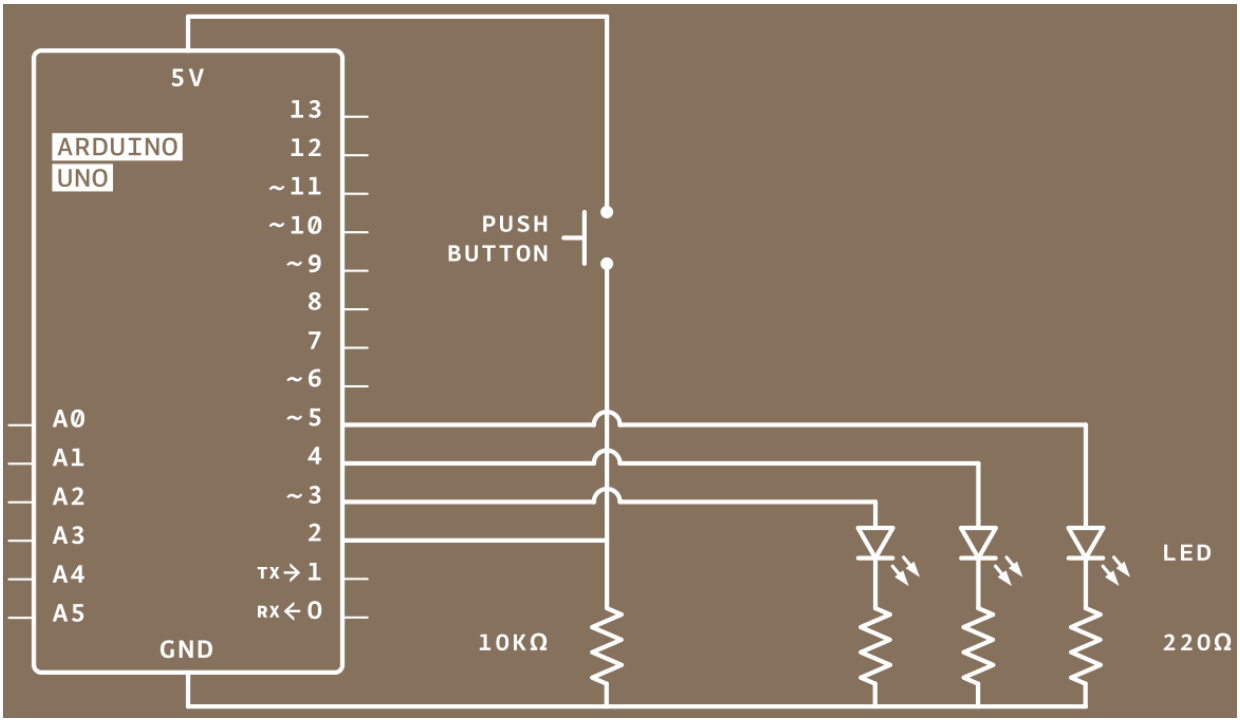


RISC (Reduced Instruction Set Computing) архитектура е вид архитектура на компютърен процесор, която има малък и силно оптимизиран набор от инструкции. Основната идея зад RISC архитектурата е да опрости набора от инструкции и дизайна на процесора, за да подобри производителността и да намали консумацията на енергия. RISC архитектурата е разработена през 80-те години на миналия век като отговор на нарастващата сложност на дизайна на компютърните процесори по това време. RISC процесорите обикновено имат малък набор от прости и силно оптимизирани инструкции, които могат да бъдат изпълнени бързо. Инструкциите в RISC архитектурата обикновено са с фиксирана дължина, което опростява дизайна на процесора.

Точка 4: Демонстрира знания за архитектурата на съвременен микропроцесор. (6т.)

Архитектурата x86, разработена от Intel, е доминиращата архитектура за персонални компютри от няколко десетилетия. Използва се в много настолни и преносими компютри, както и в сървъри и работни станции. Най-новата версия на архитектурата x86 е x86-64, която поддържа 64-битови изчисления и се използва в повечето съвременни настолни и преносими компютри. Архитектурата ARM, разработена от ARM Holdings, се използва в широка гама от устройства, включително смартфони, таблети, смарт часовници и други вградени системи. ARM процесорите са известни с ниската си консумация на енергия и обикновено се използват в устройства, захранвани от батерии.

Точка 5: Модифицира принципна електрическа схема на микроконтролер по зададена задача. (6т.)



Точка 7.3: Свързва компоненти в схема при зададена задача за моделиране на вградена система. (8т.)

Точка 8: Избира необходимите хардуерни компоненти при реализирането на вградена система, така че да реши поставената задача. (8т.)

Нарича се „Командно табло на космически кораб“. Управляваната система са светодиодите. Първоначално свети само зеленият светодиод. При натискане на ключето(докато е натиснато) свети по един от червените светодиоди, като се сменят(мигат, единият като свети, другият е тъмен и обратно). За целта използваме Arduino Uno, базирано на микроконтролер. Разполагаме с четири резистора, три от които са за светодиодите. Показаната картинка по-горе е електрическата схема на поставената задача. Пиновете могат да четат само две състояния: когато протича напрежение и когато няма такова. Този вид input обикновено се нарича дигитално (или бинарно, заради two-state-а). В нашата ситуация са HIGH и LOW. HIGH е така да се каже „тук има напрежение“, съответно LOW означава „тук няма напрежение“. Когато даденият ни output пин е HIGH, използвайки командата digitalWrite(), ние го включваме.

```
int switchState=0;
void setup(){
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
pinMode(2, INPUT);
                }
void loop() {
switchState=digitalRead(2);
if (switchState==LOW){
digitalWrite(3, HIGH);//green led
digitalWrite(4, LOW);//red led
digitalWrite(5, LOW);//red led
                }
```

```
else  {
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, HIGH);
delay(250);
digitalWrite(4, HIGH);
digitalWrite(5, LOW);
delay(250);
    }
}
```

Точка 6: Изброява базови и периферни компоненти на микроконтролер. (2т.)

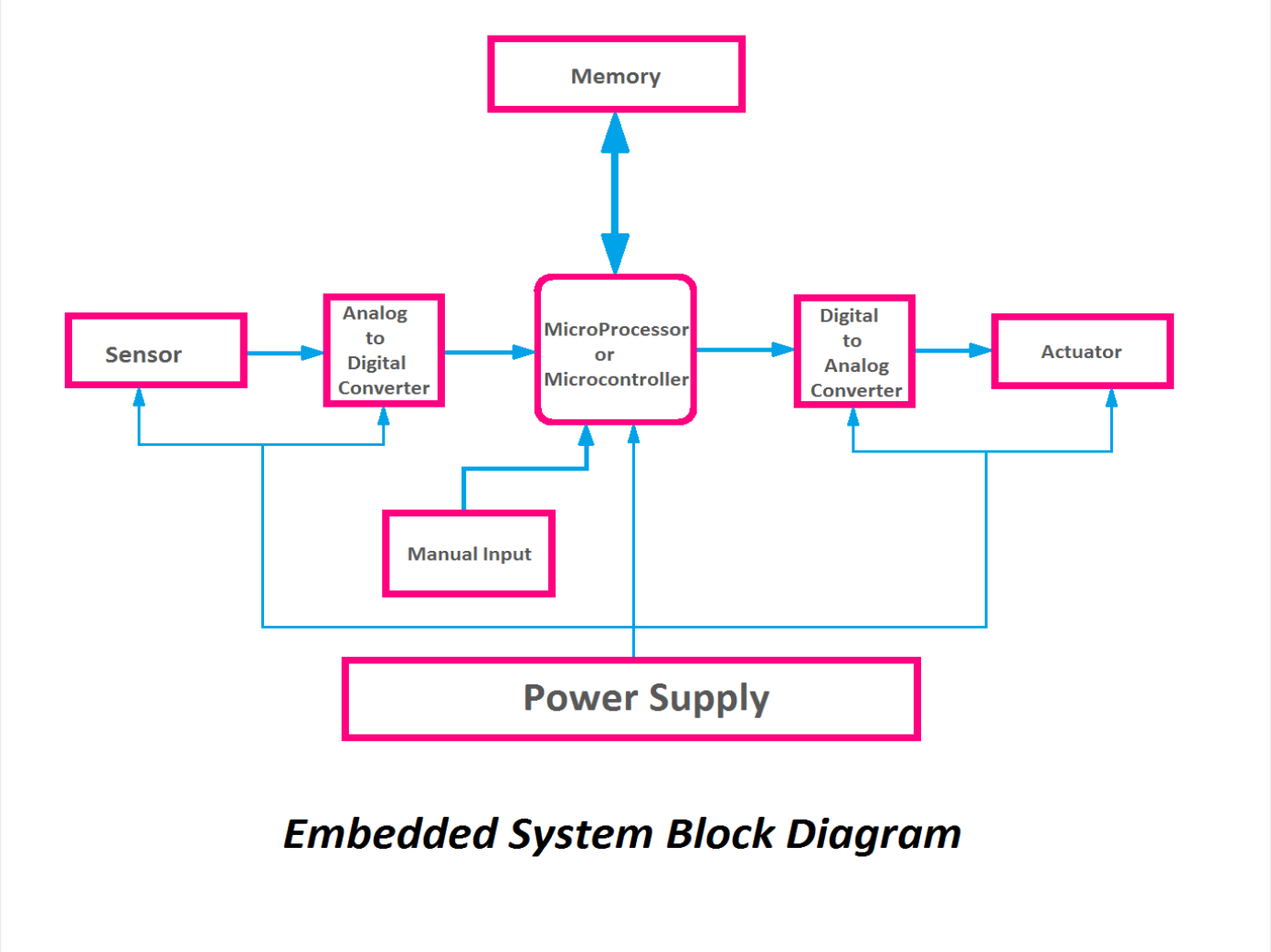
Базови компоненти	Периферни компоненти
процесор	сензори
АЛУ	изходни у-ва
преобразователни схеми	капацитети
RAM памет	
регистри	
програмна памет	
външна памет	
I/O портове	
тактов генератор	

Точка 7: Обяснява и модифицира блок-схема на вградена система. Свързва компоненти в схема при зададена задача за моделиране на вградена система. Представя графично блок-схема на вградена система. (24т.)

Точка 7.1: Обяснява блок-схема на вградена система. (4т.)

Точка 7.2: Модифицира блок-схема на вградена система. (6т.)

Точка 7.4: Представя графично блок-схема на вградена система. (6т.)



Сензори:

Сензорът е устройство, което усеща физическите промени около себе си и генерира електрически или електронен сигнал според физическите промени. Повечето от сензорите генерират сигнали с аналогов характер. Термостат, светлозависими резистори (LDR), превключвател на потока са примери за сензори, използвани във вградени системи. В една практическа вградена система има толкова много устройства и вериги, които се използват със сензори за увеличаване на тяхната чувствителност, контрол на мощността, шум или премахване на изкривяването.

Analog to Digital Converter:

Както знаем, повечето сензори генерират само аналогови сигнали. Но процесорът не може да чете и обработва аналоговите сигнали. Аналогово-цифров преобразувател за преобразуване на аналоговите сигнали, генерирани от сензора, в цифрови сигнали, които ще бъдат обработени от микропроцесора или микроконтролера.

Процесор:

Микропроцесор, микроконтролер, цифров сигнален процесор (DSP), специфични за приложението интегрални схеми (ASIC), гейт масиви, полеви програмируеми гейт масиви (FPGA), GPU са примери за устройства, които могат да се използват като процесор във вградена система. Всеки един от тях се използва за обработка. Това също зависи от целта на работата или операцията. Ако работата, разпределена от вградената система, е по-сложна, тогава се използва висококачествен процесор.

Digital to Analog Converter:

Процесорът може да генерира само цифрови сигнали. Така че различните типове изходни устройства като изпълнителни механизми, високоговорители, зумери, които работят с аналогови сигнали, не могат да бъдат директно свързани с процесора. Така че е необходим цифров към аналогов преобразувател за преобразуване на цифровия сигнал, генериран от процесора, в аналогови сигнали, които ще се използват за работа с аналогови изходни устройства. Устройствата за цифров изход като дисплеи не изискват цифрово-аналогов преобразувател, той може да бъде свързан директно към процесора с помощта на драйверна схема.

Акутатор:

Актуаторът е устройство, което прави физически промени, когато към него се приложи електрически или електронен сигнал. Соленоид, стъпков двигател, електромагнит и т.н. са примери за задвижващи механизми, използвани във вградена система.

9. Прави заключения и изводи за захранването и енергийната ефективност. (8т.)

Техники за пестене на енергия за микроконтролери

- Режими на заспиване (sleep mode)

Режимите на заспиване (обикновено наричани режими с ниска мощност) са може би най-популярната техника за намаляване на консумацията на енергия в микроконтролерите. Те обикновено включват деактивиране на определени схеми или таймери, които захранват определени периферни устройства на микроконтролерите. В зависимост от архитектурата и производителя, микроконтролерите обикновено имат различни видове режими на заспиване, като всеки режим притежава способността да деактивира повече вътрешни схеми или периферни устройства в сравнение с другия. Режимите на заспиване обикновено варират от дълбок сън или изключен, до режим на празен ход и режим на сън. Характеристиките, както и името на тези режими могат да варират при различните производители.

- Динамична модификация на честотата на процесора

Това е друга широко популярна техника за ефективно намаляване на количеството енергия, консумирана от микроконтролера. Това е най-старата техника и малко по-сложна от режимите на заспиване. Това включва фърмуера(софтуер от производителя, перманентен софтуер, програмиран в read-only памет), който динамично управлява тактовия генератор на процесора, като се редува между висока и ниска честота, тъй като връзката между честотата на процесора и количеството консумирана енергия е линейна (както е показано по-долу). Изпълнението на тази техника обикновено следва този модел; когато системата е в неактивно състояние, фърмуерът задава тактовата честота на ниска скорост, което позволява на устройството да спести малко енергия и когато системата трябва да извърши тежки изчисления, тактовата честота се възстановява. Има непродуктивни сценарии за промяна на честотата на процесора, което обикновено е резултат от лошо развит фърмуер. Такива сценарии възникват, когато тактовата честота се поддържа на ниска, докато системата извършва тежки изчисления. Ниската честота в този сценарий означава, че системата ще отнеме повече време, отколкото е необходимо, за да изпълни поставената задача и по този начин натрупано ще консумира същото количество енергия, което дизайнерите са се опитвали да спестят. Следователно, трябва да се внимава при прилагането на тази техника в критични във времето приложения.

- Оптимизиран по отношение на захранването фърмуер

Един от най-добрите начини за намаляване на количеството енергия, консумирана от микроконтролера, е чрез писане на ефективен и добре оптимизиран фърмуер. Това пряко влияе върху обема на работата, извършена от процесора за време и това, в допълнение, допринася за количеството мощност, консумирана от микроконтролера. По време на писането на фърмуера трябва да се положат усилия, за да се гарантира

намален размер на кода и цикли, тъй като всяка ненужна инструкция, изпълнена, е част от енергията, съхранявана в батерията, която се губи.

10. Анализира и модифицира програмен код, така че да реши поставена задача. (14т.)

Нека използваме горната схема за космическия кораб. Задачата, която искаме да направим обаче, ще бъде светофар. Трябва да разполагаме с три диода – червен, жълт и зелен. Модифицираме код на Arduino C, така че вградената система да представлява светофар, съответно моделиран с червен, зелен и жълт светодиод, които да се сменят на определен брой секунди(например 20). Щом се натисне ключето, да светва зеления светодиод, пак за същия брой секунди, а след това действието на светофара да продължава по обикновения начин.

```
int red = 10;

int yellow = 9;

int green = 8;

int button = 12; // switch is on pin 12

void setup(){

pinMode(red, OUTPUT);

pinMode(yellow, OUTPUT);

pinMode(green, OUTPUT);

pinMode(button, INPUT);

digitalWrite(green, HIGH);

}

void loop() {

    if (digitalRead(button) == HIGH){

        delay(15); // software debounce

        if (digitalRead(button) == HIGH) {

            // if the switch is HIGH, ie. pushed down - change the lights!

            changeLights();

            delay(15000); // wait for 15 seconds

        }

    }

}
```