Програмиране за вградени системи – Тема 16

1. **Дефинира понятията: електричество, променлив ток (AC), прав ток (DC), широчинно-импулсна модулация и др.**

- Електричеството е физически феномен, свързан с присъствието и движението на електрически заряди. То може да се проявява в различни форми, като електрически ток, електрическо поле, електромагнитни вълни и други.
- Променлив ток (AC) е вид електрически ток, който променя посоката и размера си на редовни интервали от време. Този вид ток се използва за транспортиране на енергия в електропреносните мрежи и за захранване на много електрически уреди и машини.
- Прав ток (DC) е вид електрически ток, който протича в една посока и е с постоянна стойност. Този вид ток се използва за захранване на много електронни уреди, включително компютри, мобилни телефони и други.
- Широколентова импулсна модулация (PWM) е техника за управление на електрически мощности, която използва импулсни сигнали с висока честота за регулиране на средната мощност на електрическия ток. Тази техника се използва във много приложения, включително за управление на скоростта на моторите, за управление на яркостта на светодиодните лампи и за контрол на температурата във вентилационни и климатични системи.

2. **Различава видовете развойни платки и микроконтролери.**

- Развойните платки са платки, на които се монтират електронни компоненти и които се използват за прототипиране, тестване и разработване на електронни устройства. Те предоставят основни функционалности като захранване, комуникация, програмируемост и други, които позволяват на потребителите да тестват и да разработват нови електронни устройства.
- Има много видове развойни платки, като някои от най-популярните са Arduino, Raspberry Pi, ESP32 и STM32. Всяка от тези платки има свои характеристики и спецификации, които я правят подходяща за различни приложения.
- Микроконтролерите са електронни компоненти, които съдържат централен процесор, памет и периферни устройства, като например A/D конвертори, PWM генератори, таймери и други. Те се използват за управление на електронни устройства и имат много приложения в различни области, като автомобилната индустрия, домашните електроуреди, промишлеността, медицината и други. Микроконтролерите могат да бъдат програмирани за различни функции и задачи, като контрол на мотори, сензори, светлини, комуникация и други. Има много видове микроконтролери, като някои от най-известните са AVR, PIC, ARM и STM32. Всяка от тези серии микроконтролери има свои характеристики и спецификации, които я правят подходяща за различни приложения.

3. **Обяснява основните характеристики и особености на вградените системи. Дава пример за блокова схема на вградена система.**

- Вградените системи са компютърни системи, които са специално проектирани и програмирани за конкретни задачи и приложения. Те са обикновено по-малки и по-евтини от общоползваните компютърни системи и имат ниска консумация на енергия. Вградените системи могат да се намират в различни устройства, като автомобили, домашни електроуреди, мобилни устройства, медицински уреди и други.

Ето някои от основните характеристики и особености на вградените системи:

- Те имат специализирани функции и приложения.
- Те са обикновено проектирани да работят в реално време.
- Те имат ограничени ресурси, като ограничена памет, процесорна мощност и скорост на комуникация.
- Те са обикновено много ефективни в управлението на ресурсите, като консумация на енергия и памет.
- Те могат да бъдат високо надеждни и с малко поддръжка.
- Те могат да бъдат трудни за програмиране поради ограниченията на ресурсите и хардуерните ограничения.

4. **Разработете и анализирайте кода на следната задача**

```
int pinSvetodiod = 13; // пин 13

void setup() {

 pinMode(pinSvetodiod, OUTPUT);


}


void loop() {

 digitalWrite(pinSvetodiod, HIGH);// светване на диод

 delay(1000);// забавяне с 1 секунда

 digitalWrite(pinSvetodiod, LOW);// изгасяне на диод

 delay(1000);// забавяне с 1 секунда

}
```

5. **Проектирането на компютърни системи и техните компоненти преминава през няколко етапа. Пояснете етапите на проектиране.**

Проектирането на компютърни системи и техните компоненти е сложен процес, който включва няколко етапа, които обикновено се извършват последователно. Основните етапи на проектирането на компютърни системи:

Анализ на изискванията - този етап включва анализ на нуждите и изискванията на клиента или потребителя. Трябва да се определи целта на системата, функциите, които тя трябва да изпълнява и какви ще бъдат изискванията за производителност и надеждност.

- Проектиране на архитектурата - този етап включва проектиране на основната архитектура на системата. В този етап се избират технологиите, хардуерните компоненти и софтуерните платформи, които ще бъдат използвани в системата. Проектира се архитектурата на хардуера и софтуера, както и начина на комуникация между тях.
- Проектиране на компонентите - този етап включва проектиране на всеки един компонент от системата. В този етап се проектират печатни платки, микроконтролери, сензори, актуатори и други компоненти. Основната цел на този етап е да се уверите, че всички компоненти ще работят заедно и че ще отговарят на изискванията.
- Проектиране на софтуера - този етап включва проектиране на софтуера, който ще управлява системата. В този етап се програмират микроконтролерите, който ще управляват хардуерните компоненти. Също така се проектират и приложенията, които ще използват потребителите.
- Тестване и отстраняване на грешки - след като системата е проектирана, тя трябва да бъде тествана, за да се уверим, че функционира правилно. В този етап се извършват различни видове тестове, за да се провери надеждността и функционалността на системата. Ако се открият грешки, те трябва да бъдат отстранени и тестовете да се повторят, докато системата работи правилно.
- Производство - след като системата е проектирана и тествана, тя може да бъде произведена. Този етап включва изграждане на печатни платки, монтаж на компонентите и изграждане на кутиите и други части на системата.
- Монтаж и инсталация - след като системата е произведена, тя трябва да бъде монтирана и инсталирана в желаното място. Това може да включва инсталация на софтуера, свързване на кабели и конфигуриране на системата.
- Поддръжка и обновяване - един път като системата е инсталирана, тя може да се нуждае от поддръжка и редовно обновяване на софтуера. Това може да включва поправка на грешки, подмяна на компоненти или добавяне на нови функции.

6. **Описва специфичните изисквания и особености на софтуера, предназначен за вградени системи.**

Софтуерът, предназначен за вградени системи, има някои специфични изисквания и особености, които го правят различен от софтуера, който се използва на обикновени компютърни системи. Някои от тези изисквания и особености включват:

- Ниският брой на ресурсите - вградените системи обикновено имат ограничени ресурси, като например ограничена оперативна памет, ограничен брой входове/изходи и ограничен процесорен капацитет. Поради тази причина, софтуерът, който се използва в тези системи, трябва да бъде написан, за да използва минимални ресурси и да бъде оптимизиран за максимална ефективност.
- Реално време - много от вградените системи трябва да функционират в реално време, като например системите за контрол на температурата, управление на двигатели, системите за контрол на промишлени производствени процеси и др. Това означава, че

106

софтуерът трябва да бъде способен да реагира на събития в реално време, като например входни сигнали или външни събития, и да изпълнява задачи в реално време.

- Надеждност - вградените системи обикновено се използват за мисии-критични приложения, като например авионика, медицински устройства, автомобилни системи и други. Това означава, че софтуерът трябва да бъде написан с висока степен на надеждност и безопасност, за да избегне грешки и неизправности, които могат да доведат до сериозни последици.
- Системи за управление на енергията - много вградени системи изискват управление на енергията, за да могат да се използват с батерии или други източници на енергия. Това означава, че софтуерът трябва да бъде написан с висока енергийна ефективност и да използва малко енергия.

7. Обяснява структурата на програма за вградена система.

Програмите за Arduino се пишат на програмен език, базиран на C++, който е оптимизиран за вградени системи. Структурата на програмата за Arduino се състои от следните основни части:

Функциите setup() и loop() са основните функции в програмата за платформата Arduino. Те се използват за инициализиране на системата и за основния цикъл на програмата.

- Функцията setup() се изпълнява веднъж, когато системата се стартира. Тя се използва за инициализиране на всички необходими компоненти и за конфигуриране на входно-изходните портове. Например, ако програмата използва светодиод, функцията setup() ще конфигурира порта на микроконтролера, където светодиода е свързан, като избере режима на работа на порта (вход или изход).
- Функцията loop() се изпълнява непрекъснато след изпълнението на функцията setup(). Тя се използва за основния цикъл на програмата и съдържа основните задачи, които трябва да се изпълняват от системата. Например, ако програмата контролира светодиод, функцията loop() ще изпраща сигнали към порта, когато светодиода трябва да бъде включен или изключен, като използва подходящите команди за работа с портовете.

8. Демонстрира код за управление на състоянието на изводите.

```
int ledPin = 13;  // номер на порта, където е свързан светодиода
void setup() {
 pinMode(ledPin, OUTPUT);  // конфигуриране на порта като изход
}
void loop() {
 digitalWrite(ledPin, HIGH); // включване на светодиода
 delay(1000);              // изчакване 1 секунда
 digitalWrite(ledPin, LOW);  // изключване на светодиода
 delay(1000);              // изчакване 1 секунда
}
```

9. Демонстрира код за управление на електронни компоненти с развойна платка по зададена задача.

```
// Използваните изводи на Arduino Uno пиновете 9 и 10
int ledPin = 9;         // Пинът, към който е свързана крушката
int ledState = LOW;     // Състояние на крушката (включено/изключено)

// Подготовка на пиновете
void setup() {
 pinMode(ledPin, OUTPUT);  // Задаване на пина за изходен
}

// Главна програма
void loop() {
 // Превключване на състоянието на крушката
 if (ledState == LOW) {
  ledState = HIGH;
 } else {
  ledState = LOW;
 }
```

```
digitalWrite(ledPin, ledState);  // Задаване на състоянието на крушката
delay(1000);                     // Закъснение от 1 секунда (мигане на крушката)
}
```

10. Прави заключения и изводи за серийната комуникация

Серийната комуникация е изключително полезна техника за обмен на данни между различни електронни устройства. Това е особено важно във вградените системи, където микроконтролерът често се използва за събиране на данни от сензори, управление на актуатори и комуникация с други устройства.

Серийната комуникация може да се осъществи по множество начини, но един от най-широко използваните методи е употребата на протокола UART.

Ключовите елементи на серийната комуникация са baud rate (скорост на предаване), дължина на думата (битове, използвани за представяне на една информационна дума), контролни битове (за потвърждаване на приемането и защита от грешки), протоколни формати (протоколи за данни, контролни протоколи и др.).

Серийната комуникация може да бъде използвана за различни приложения, като сензори, дистанционно управление, протоколи за управление на мотори, дисплеи и други.

Серийната комуникация е важен елемент от многобройните приложения на вградените системи. Нейното използване може да улесни и ускори разработката на софтуер и хардуер за вградени системи, като осигурява по-бърз обмен на данни и управление, отколкото друга. Важно е да се избере подходящ протокол за комуникация и да се зададат правилните настройки, за да се осигури правилното функциониране на вградената система.

Serial.Begin(speed) - определя скоростта на серийната комуникация

Serial.Read() - получава данни от серийния порт

Serial.Write(Val) - изпраща данни към серийния порт. Параметърът val може да бъде единична променлива, низ или масив.

11. Открива грешки в програмен код и го модифицира, така че да реши поставената задача.

```
Import processing.serial.*;
string buff = "";
string temp = "";
float temporary = 0.0;
float screenwidth = 0.0;
Float xCoordinate = 0;
Float yCoordinate = 0;
```

```
Int val = 0;
Int NEWLINE = 10;
Serial port;
void setup() {
 Size(200,200);
strokeWeight(10);
Stroke(255);
Smooth();
Port = new Serial(this, “COM2”, 9600); // променете COM2 с верният порт
}
void Draw() {
fill(0,2);
rectMode(CORNER);
rect(0,0,width, height);
while(port.available() > 0)
 serialEvent(port.read());
}
point(xCoordinate, yCoordinate);
}
void serialEvent(int serial){
if(serial != NEWLINE) {
buff += char(serial);
} else {
if (buff length() > 1){
temp = buff.substring(0, buff.length()-(buff length()-1));
if(temp.equals("A") == true){
temp = buff. substring(1, buff.length());
temporary = float(temp);
xCoordinate= width/(1024/temporary);
println(xCoordinate);
}
```

```
if(temp.equals("B") ==true) {
temp =buff.substring(1, buff.length());
temporary =float(temp);
yCoordinate=height/(1024/temporary);
println(yCoordinate);
}
buff= "";
}
}
}
```

12. Анализира, определя и допълва програмен код, така че да реши поставената задача.

```
int xPin = A0; // аналогов вход за управление на х-координатата
int yPin = A1; // аналогов вход за управление на у-координатата
int ledPin = 13; // изход за светодиод

void setup() {
 pinMode(ledPin, OUTPUT); // настройка на изхода за светодиода
 Serial.begin(9600); // стартиране на серийната комуникация
}

void loop() {
 int xValue = analogRead(xPin); // четене на стойността на х-координатата
 int yValue = analogRead(yPin); // четене на стойността на у-координатата

 Serial.print(xValue); // извеждане на х-координатата
 Serial.print(",");
 Serial.println(yValue); // извеждане на у-координатата
```

```
  if (yValue < 500) {
    digitalWrite(ledPin, HIGH); // включване на светодиода, ако курсора е горе
  } else {
    digitalWrite(ledPin, LOW); // изключване на светодиода, ако курсора е долу
  }

  delay(100); // забавяне на изпълнението за по-голяма стабилност
}
```

Този код използва двата потенциометъра за управление на x-координатата и y-координатата на курсора на екрана. Когато курсорът се придвижи нагоре, светодиодът на платката също се включва.

Когато се стартира програмата, тя отваря серийна комуникация със скорост 9600 бода. При всяко измерване на стойността на потенциометъра за x и y координати, тези стойности се извеждат на серийния порт, като вместо знака за нов ред се използва запетаята.

Когато курсорът е придвижен нагоре, програмата включва светодиода на платката, за да уведоми потребителя, че курсорът се движи нагоре. Когато курсорът е придвижен надолу, светодиода се изключва.