Организация на вход-изхода в система за паркиране

Светлана Стефанова, Милко Маринов

Input-output organisation of parking systems: *One of the main problems in the modern society is the parking. A good solution is a system managing a lot, garage or parking. One of the tasks when developing such a system is the input/output organisation. This paper presents a module coordinating the input/output area. The major classes and the architecture used for this module are described in details.*

Key words: *Parking systems, driver, communications protocol.*

ВЪВЕДЕНИЕ

Системите за паркиране са неразделна част от съвременото общество. Такава система се проектира и изгражда винаги в зависимост от нуждите и желанията на крайния клиент. Тя трябва да бъде гъвкава и надеждна, подходяща за различните видове паркинги: частни и обществени, летища, хотели, търговски центрове, производствени предприятия и други [4]. Системите за паркиране са предназначени са за управление и контрол на паркинг зоните, като съчетават хардуерни компоненти за контрол на достъпа и съответно програмно осигуряване за управление. Такива системи многократно повишават ефективността при управление на паричния поток [1,2,5].

Целта на настоящата статия е да се представи основния модул, организиращ и управляващ входа и изхода в такава система.

АРХИТЕКТУРА НА МОДУЛ, УПРАВЛЯВАЩ ВХОД-ИЗХОДА В СИСТЕМА ЗА ПАРКИРАНЕ

Във всяка система за паркиране основните устройства, които управляват работата са:

- ⇒ устройство за издаване на билети;
- ⇒ устройство за автоматично плащане;
- ⇒ касиерска станция за плащане
- ⇒ устройство за излизане.

Като основни единици в тези устройства се явяват:

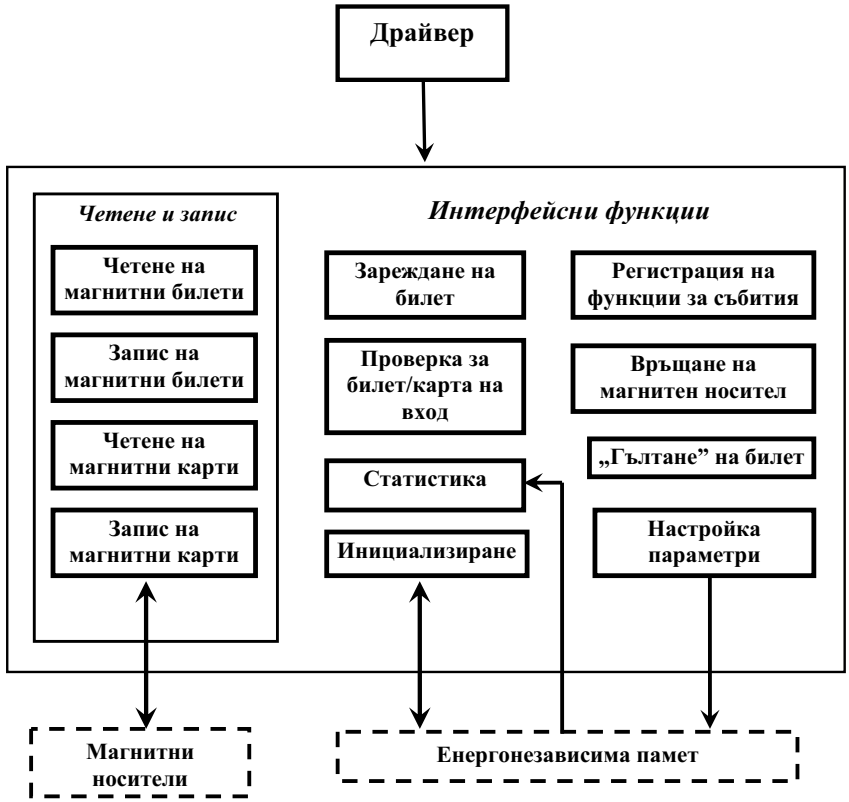
- Магнитна глава – устройство, извършващо запис или четене на информация върху магнитна лента;
- Печатаща глава – устройство, извършващо отрязване на билет и нанасяне на печатна информация върху него.

Всяка от главите има сензори, чиито показания дават реалното състояние към дадения момент.

Архитектурата на модула, управляващ вход-изхода в система за паркиране е представена на Фигура 1. Драйверът реализира вътрешните функции на ниско ниво, които управляват магнитната и печатащата глава. Двете глави могат да се разглеждат като едно цяло и носят общото название валидатор. Тези вътрешни функции не са достъпни за другите модули, които използват драйвера. Специфицирани са в съответствие с особеностите на валидатора, като е изграден протокол за комуникация на главите с основните устройства.

Валидаторът работи в „режим на събитие”. Има дефинирани две събития, които главите изпращат към драйвера: рестартиране и промяна състоянието на сензорите. При такава организация самият драйвер не си комуникира постоянно с главите, а се обръща към тях само при нужда да изпрати команда или при възникване на едно от

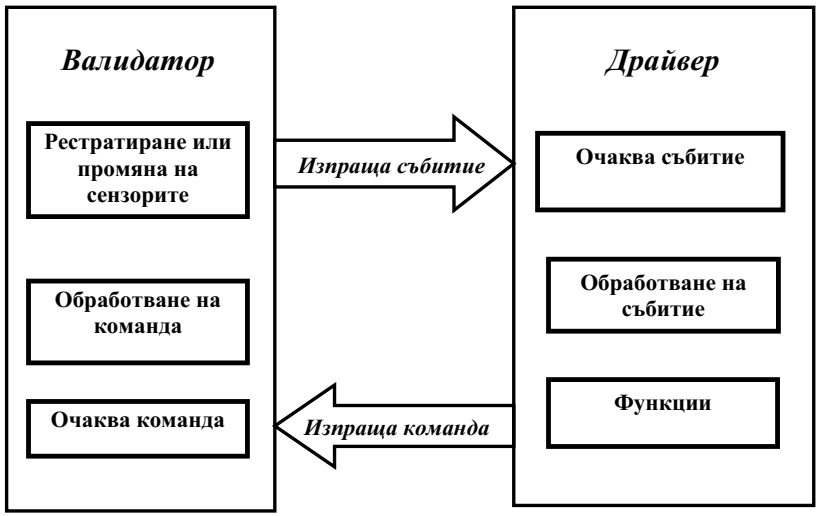
тези две събития. На фигура 2 е показано взаимодействието между валидатора и драйвера.



Фиг. 1 Архитектура на модул, управляващ вход-изхода

Интерфейсните фукции са изградени като краен автомат („машини" на състоянията). Всяка функция разполага с набор от състояния и в едно състояние се изпълнява само една команда. В зависимост от резултата може да се запази същото състояние или да се премине в друго. Основните функции са:

- ⇒ инициализация на валидатора;
- ⇒ връщане информация относно валидатора;
- ⇒ проверяване сензорите на входа на валидатора;
- ⇒ обработка на билет;
- ⇒ връщане на билет от валидатора на клиента;
- ⇒ запис на данни върху магнитен носител;
- ⇒ калиброване на главите;
- ⇒ други спомагателни функции.



Фиг. 2 Взаимодействие „валидатор"-„драйвер"

В основната нишка на драйвера е реализиран краен автомат (State машина). В свободно състояние се очаква обръщение към функция от главно устройство, което използва драйвера. От свободно състояние се преминава в ново състояние, което изпълнява подадената команда и в зависимост от резултата се преминава или отново в свободно състояние или в състояние на блокировка.

Връзката между драйвера и главните устройства е механизъм от тип клиент-сървър. Сървър е драйверът, а клиенти са главните устройства. Драйверът има набор от събития, за които главните устройства могат да регистрират функции. В отделна нишка се следи за събитие, при настъпването на което, ако драйверът е в свободно състояние, той прави обръщение към всички функции, които са се регистрирали за възникналото събитие. Обръщението става последователно, според реда на регистриране на функциите.

СТРУКТУРА НА ПРОТОКОЛА ЗА КОМУНИКАЦИЯ НА ВХОД-ИЗХОДА

Комуникацията се осъществява по RS232 при скорости на обмен 28800 bit/sec. Протоколът е тип Master/Slave с настъпващи събития от Slave устройството. Slave устройството очаква и изпълнява команди, изпратени му от Master устройството като връща съобщение в зависимост от резултата. Slave устройството инициира комуникция само при следните възникнали събития: при неговия рестарт и при промяна състоянието на негов сензор.

Съобщенията в протокола са единични байтове или пакетни съобщения:

- *единични байтове* – могат да бъдат следните типове:
  ACK=0x82 – изпраща се от Master или Slave при всяко коректно получено пакетно съобщение;
  NAK=0x86 – изпраща се от Мастер или Slave при всяко некоректно получено пакетно съобщение и тогава този пакет трябва да се повтори;
  MagEvent=0x80 – изпраща се от Slave, когато е режим на изчакване на събитие и има промяна в състоянието на сензор. Когато Master получи такова съобщение, трябва да изпрати команда за прочитане състоянието

на сензорите и в зависимост от резултата да извърши съответните действия;
MagReset=0x83 – изпраща се към Master след рестарт на Slave.

- *пакетни съобщения* – форматът на пакета е <PS><data><CS>, където
  <PS> - един байт, указващ броя байтове в пакета;
  <data> - данни с дължина до 90 байта;
  <CS> - един байт контролна сума, която е аритметична сума на всички байтове от пакета, без самия контролен.

По отношение на всяко съобщение важат следните изисквания за максимално допустимите закъснения във времето:

- Максимално допустимото времезакъснение от байт до байт в пакетно съобщение е 30ms. При надхвърляне на това време получените байтове се игнорират и се изпраща NAK съобщение към устройството.
- След получаване на пакетно съобщение времето за отговор не трябва да бъде по-голямо от 50ms.
- След изпращане на пакетно съобщение максималното време на очакване на потвърждение е 500ms. Получените данни извън протокола се игнорират.
- Минималното гарантирано време между непакетно съобщение ACK и пакетно съобщение е 50ms.

Нормалната последователност на съобщенията е следната:

1. Master изпраща пакетно съобщение, съдържащо командата към Slave. Master стартира таймер с продължителност 500ms.
2. Slave отговаря с ACK при коректно получено съобщение и стартира изпълнението на получената команда.
3. Slave завършва изпълнението на командата и отговаря с пакетно съобщение.
4. Master отговаря с ACK при коректно получено съобщение. Slave очаква до 500ms получаването на ACK.

Последователността на съобщенията при грешна контролна сума или изтичне на допустимото време в пакета на Master устройството е:

1. Master изпраща пакетно съобщение, съдържащо командата към Slave.
2. Slave отговаря с NACK при грешна контролна сума или таймаут между байтовете в пакета и не стартира изпълнението на получената команда.
3. Master повторно изпраща същото пакетно съобщение.

ОСНОВНИ КЛАСОВЕ, ИЗПОЛЗВАНИ ПРИ РЕАЛИЗАЦИЯТА

При реализация на модула са използвани основни типове и структури от данни, позволяващи следене състоянието на сензорите, своевременно подаване на съобщение и обработването им с команди. ***EValResult*** описва всички възможни кодове за изход от функциите, а ***EValOptions*** описва опциите за изход от функциите. ***EValEvents*** описва набора от събития, за които всяко устройство може да се регистрира. При възникване на определено събитие се извикват функциите, регистрирани от всяко устройство.

Структурата ***ValConfigData*** съдържа данни за инициализация на глава с определени параметри:

- типа на билетите, които могат да се четат;
- броя на билетите за издаване;

- броя на стъпките при рязане на билет;
- броя на стъпките за печатъщата глава;
- формата в брой байтове на отстъпката;

ЗАКЛЮЧЕНИЕ

Предложената обектно-ориентирана реализация на модула, управляващ вход-изхода в система за паркиране е осъществена на езика C. Използвана е интегрирана средата на Zilog – Zilog Developer Studio II, поради използването на процесор eZ80L92 в управляващата платка, за която е разработен драйверът.

Използването на драйвер за управление на валидатора дава възможност за лесното му поддържане. Възникнали проблеми при работа с драйвера от дадено устройство се отстраняват и те не възникват при останалите устройства. Драйверът систематизира необходимия набор от функции, които покриват нуждите на устройствата, ползващи валидатора. По този начин се избягва неконтрулируемото подаване на команди към валидатора от различни устройства.

ЛИТЕРАТУРА

[1]. http://www.miningbg.com
[2]. http://www.birdm.net/bg/products/automated-parking-system
[3]. http://www.kaba.com
[4]. ZiLOG Developer Studio II – eZ80Ecclaim, User manual, UMO14423-0607
[5]. Стефанова Св., Маринов М., Концептуален модел на специализирана картова система, Научни трудове на РУ, 2007, 131-135

За контакти:

Гл.ас. д-р Светлана Стефанова, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, Тел.: (082) 888 356, E-mail: SStefanova@ecs.ru.acad.bg

Доц. д-р Милко Маринов, Катедра “Компютърни системи и технологии”, Русенски университет “Ангел Кънчев”, Тел.: (082) 888 356, E-mail: MMarinov@ecs.ru.acad.bg

Докладът е рецензиран.