Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.3
DOI: https://doi.org/10.24867/26IH03Tirnanic

ПРИМЕНА *FREERTOS* ОПЕРАТИВНОГ СИСТЕМА ЗА УПРАВЉАЊЕ ИНДУСТРИЈСКИМ СИСТЕМИМА

APPLICATION OF THE *FREERTOS* OPERATING SYSTEMS FOR CONTROLLING INDUSTRIAL SYSTEMS

Алекса Тирнанић, *Факултет техничких наука, Нови Сад*

Област – МЕХАТРОНИКА

Кратак садржај – *Рад се састоји из две целине - теоријске основе и физичке реализације. У оквиру теоријског дела биће објашњени основни појмови, основне карактеристике као и дефиниције embedded система у реалном времену, као и осталих појединости везане за реализацију самог мастер рада. Након тога, у практичном делу биће приказан рад са системима у реалном времену на PIC32MZ1024EFH микроконтролеру у ком се симулира рад једне индустријског система са покретном траком на коју се поставља флашица која се пуни са течношћу из резервоара и након што се напуни креће до краја покретне траке.*

Кључне речи: *RTOS (системи у реалном времену), embedded, PIC, Arduino микроконтролери*

Abstract – *The paper consists of two units - theoretical basis and physical realization. In the framework of the theoretical unit, the basic concepts, basic characteristics and definitions of embedded systems in real time will be explained, as well as other details related to the realization of the Master thesis itself. After that, in the practical part, the work with real-time systems on the PIC32MZ1024EFH microcontroller will be shown, in which the work of an industrial system with a conveyor belt is simulated, on which a bottle is placed, which is filled with liquid from a tank and, after being filled, continues to move to the end conveyor belts.*

Keywords: *RTOS (Real Time Systems), Embedded, PIC, Arduino Microcontrollers*

1. УВОД

Овај рад се бави симулацијом рада једног индустријског система са покретном траком са свим неопходним елементима и уређајима потребним за представљње аутоматизованог процеса рада. У наредним поглављима биће представљен начин програмирања, као и управљања целокупним системом за пуњење резервоара са течношћу.

Главни циљ рада јесте упознавање са системима у реалном времену, конкретно са *FreeRTOS* програм и његовом применом у аутоматизованим индустријским системима.

НАПОМЕНА:
Овај рад је проистекао из мастер рада чији је ментор био др Миодраг Бркић, ван. проф.

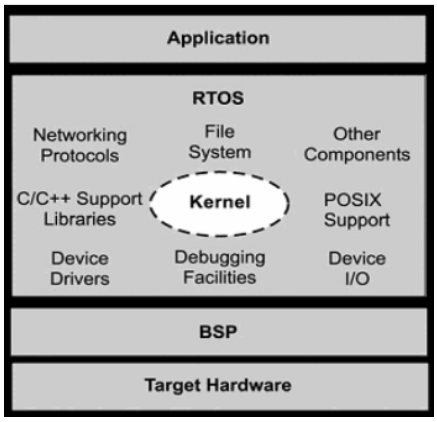
Процес који се аутоматизује јесте пуњење флашице течношћу и њено кретање кретње по покретној траци. У самом процесу аутоматизације учествују Arduino и PIC32 који комуницирају преко серијских пинова.

На почетку и на крају траке постоје гранични сензори који детектују присуство предмета на траци, рад мотора је симулиран помоћу диоде, чијом осветљеношћу се управља помоћу енкодера и PWM сигнала. Ниво течности у резервоару се прати помоћу потенциометра који симулира пловак.

2. ТЕОРИЈСКЕ ОСНОВЕ

2.1. Архитектира система у реалном времену

RTOS се пројектује тако да има модуларну и слојевиту архитектуру. Може да садржи комбинацију више различитих модула: језгро (eng. *kernel*), фајл систем (eng. *file system*), подршку за мрежне протоколе (нпр. подршку за *TCP/IP* протокол) и друге компоненте, што је илустровано на Слици 1. Покренута апликација приступа ресурсима система кроз *API* функције оперативног система. (*API – Application programming interface*) [1].



Слика 1: *Уопштена архитектура RTOS [1]*

Језгро представља средишњу и неопходну компоненту сваког *RTOS-a*. Унутар њега је имплементирана логика која је одговорна за распоређивање програмских задатака у времену, као и основни објекти за синхронизацију и комуникацију између програмских задатака [1].

Основне компоненте кернела су: Планер (eng. *Scheduler*) - компонента у сваком језгру која на основу алгоритама одређује који таск треба да се изврши у датом тренутку; Објекти - стандардни објекти

укључују подршку за задатке (*task*-ове), семафоре и редове порука;

Сервиси - представљају операције које језгро извршава над објектима или опште операције као што су обрада прекида и управљање ресурсима [1].

2.2. *Scheduler*

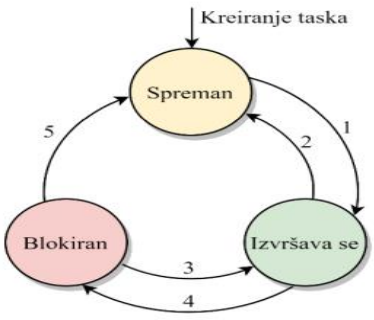
Scheduler је срце кернела и обезбеђује алгоритам за распоређивање задатака на систему. Основни појмови су:

- Распоредљиви ентитети (eng. schedulable entities)
- Модел извршавања програмских послова (eng. multitasking)
- Замена контекста (eng. context switching)
- Диспечер (eng. dispatcher)
- Алгоритми распоређивања (eng. *scheduling algorithm*) [2].

2.3. *Task*-ови

Task је јединствена програмска целина која види CPU као свој „приватни“ ресурс. *Сваки* задатак обично представља једну бесконачну програмску целину, односно постоји упоредно са другим задацима [1].
Задатак може бити у три основна стања

- (Спреман – eng. *ready state*) Задатак је спреман за извршавање, али тренутно се извршава задатак вишег приоритета.
- (Блокиран - eng. *blocked state*) Задатак захтева ресурс који није доступан.
- (Извршава - eng. *running state*) Задатак се тренутно извршава јер је тренутно задатак са највишим приоритетом [3].



Слика 2 *Коначна машина стања таскова [3]*

2.4. Бинарни семафори

Бинарни семафор је сигнални објекат који служи за синхронизацију између групе задатака. Он има две вредности - 0 и 1, односно заузет и слободан [3]. У овом раду, бинарни семафори се користе да онемогуће извршавање таска за покретање мотора уколико нису испуњени услови за старовање, а то су да је притиснут почетни тастер, да је ниво течности у резервоару у прописаном нивоу и да је брзина мотора нула.

2.5. Бројачки семафори

Бројачки семафор користи бројач да би се омогућило више пута заузимање или ослобађање семафора [3]. У овом раду се користе за прикупљање података са граничних тастера јер се на тај начин гарантује да ће се сваки сигнал сачувати и да ће систем одреаговати, односно зауставити процесе рада.

2.6. Мутекс семафори

Мутекс семафори представљају специјалну верзију бинарних семафора који подржавају особине власништва над семафором [3]. У овом задатку сви процеси који чекају да стигне наредба *“START”* і користе заједнички мутекс семафор како би процес свих операција кренуо истовремено.

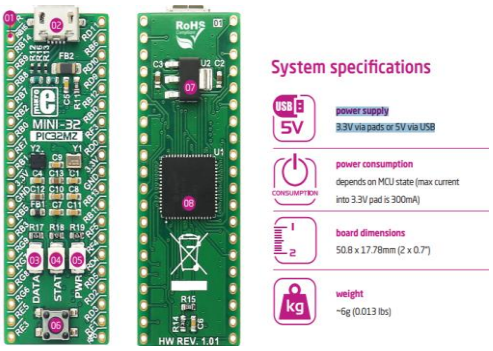
2.7. Редови порука

Редови порука се користе за размену инормација између таскова. Тако на пример, из таска за праћење нивоа течности се шаљу подаци таску за управљање мотором да уколико ниво течности није на прописаном нивоу, да не покреће мотор, као и што таск у ком су дефинисани гранични тастери шаље информацију таску за управљање мотором да ли је неки тастер притиснут

2.7. Основне карактеристике *PIC32MZ*

PIC32MZ1024EFH064 је микроконтролер из серије *Microchip* PIC32MZ, који се темељи на архитектури *MIPS32* што му омогућава високе перформансе и ефикасност.. Основне карактеристике овог микроконтролера су:

- 1*MB* флеш меморије, 512*KB RAM-a*.
- Брзина процесора преко 200*MHz*.
- Комуникациони интерфејси: *UART, SPI, I2C, USB, Ethernet.*
- Периферије: PWM модули, ADC конвертори, *GPIO* пинови.
- Максимална струја ако је на 3.3V је 300mA
- Овај модел има 64 пинова у одређеном кућишту [4].



Слика 3. *PIC32MZ* плочица *[4]*

3. ФИЗИЧКА РЕАЛИЗАЦИЈА СИСТЕМА

3.1. Програмирање PIC32 микроконтролера

PIC32MZ1024EFH микроконтролер се програмира у *MPLAB X IDE* развојном окружењу. Како би се користиле библиотеке везане за периферијске уређаје, вршила иницијализација пинова графичким путем (*GUI* – eng. *Graphical user interface*), као и многе друге могућности неопходно је инсталирати *MPLAB Code Configurator* (*MCC*), као и *Harmony 3 plugin*-ове. У оквиру овог рада, главне библиотеке су биле везане за серијску комуникацију, као и за *FreeRTOS* програма у реалном времену.

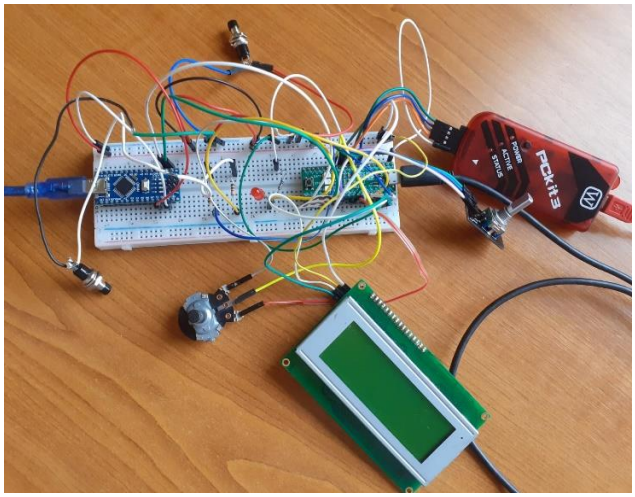
Неопходно је имати и *PICKit3* програматор, и повезати његове пинове на микроконтролер. У оквиру *MPLAB*-а је још потребно у подешавањима пројекта

изабрати одговарајући програматор и онда је могуће испрограмирати микроконтролер.

3.2. Хардверске компоненте система

У наставку биће наведени и касније детаљно описани сви потребни кораци за реализацију овог рада као и сама физичка реализација, која је приказана на слици 4. Све компененте су повезане на заједничку нулу, а начин повезивања компененти у систему је описан у даљем тексту.

1. Потребно је повезати излазе програматора редом на MCLR, 3.3V, GND, PGED1 и PGEC1 пинове PIC32MZ микроконтролера
2. Повезати Arduino и PIC32MZ микроконтролере преко серијске комуникације.
3. LCD I2C екран се повезује на извор напајања од 5V и на SDL и SCL пинове на PIC32MZ микроконтролеру.
4. Енкодер се такође повезује на напајање од 5V са ардуина, а два канала А и Б на дигиталне улазе микроконтролера.
5. Потенциометар се повезује на 3.3V, као и на ADC сигнал са микроконтролера
6. LED диода је са стране катоде повезана на отпорник од 100Ω, па на PWM сигнал са микроконтролера, док са стране аноде на нулу.
7. Тастери су повезани *pull-up* методом.



Слика 4. *Физичка реализација индустријског система*

3.3. Принцип рада система

Основни циљ пројекта је упознавање са радом у *FreeRTOS* и *Harmony 3*, са харверском имплементацијом на PIC32 микроконтролера. У ту сврху се развија систем који симулира рад индустријског система који се у даљем развоју може имплементирати и надоградити како би радио у реалним условима. Због тога су изабрани системи у реалном времену јер нам они омогућавају да се више таскова извршава „истовремено“ , односно омогућава да се задаци извршавају по приоритету што значи да ће се задатак са већим приоритетом извршити у тачно одређеном временском тренутку, независно од осталих процеса у систему. Овим методом се дакле поузданост и безбедност радне станице поставља на највиши ниво.

Почетак рада система започиње слањем команде *“START”* са ардуина на PIC32MZ путем серијске комуникације. Тада, уколико је брзина мотора нула, уколико је ниво течности већи од прописаног минимума, што се прати помоћу потенциомтра, који симулира пловак, и уколико је активан почетни гранични тастер који служи да детектује када је флашица у почетном положају, чека се три секунде да се флашица напуни течношћу, па затим покретна трака полако убрзава док не стигне до максималне номиналне брзине, креће се одређено време том брзином и затим полако успорова док се потпуно не заустави. Само кретање мотора се симулира PWM сигналом који управља вредношћу осветљености LED диоде. Покретна трака се заустављa уколико је предмет прешао одређени пут, што се прати помоћу енкодера или уколико је притиснут тастер на крају покретне траке. Такође, с обзиром да радимо са 12-обитним потенциометром, вредност A/D конверзије је број у оквиру прописаног минималном и максималног нивоа. Тако да уколико је вредност конверзије мања од минималне или већа од максималне сматра се да је ниво течности у резервоару у првом случају испод, а у другом изнад дозвољеног нивиа и цео систем престаје са радом.

Све ове информације се прате преко серијског монитора у ардуино програму, и уколико се детектује било која од информација за заустављање покретне траке, шаље се порука *“STOP”* и тада се цео процес зауставља. Освежавање екрана се врши на одређени предефинисани временски период.

Такође, на LCD екрану се прате најважније информације, као што су вредност нивоа течности у резервоару, брзина обртања мотора, пређени пут, као и време од почетка процеса, односно од постављања предмета у почетни положај.

```
Pocetni taster pritisnut:       DA
Nivo tecnosti u rezervoaru:     1500
Vreme trajanja procesa:         0 s
Predjeni put:                   0 cm
Brzina motora:                  0 Hz
Krajni taster pritisnut:        NE

"START"

Pocetni taster pritisnut:       NE
Nivo tecnosti u rezervoaru:     1200
Vreme trajanja procesa:         5 s
Predjeni put:                   75 cm
Brzina motora:                  30 Hz
Krajni taster pritisnut:        NE
```

Слика 5. *Изглед серијског екрана у Arduinu за праћење процеса*

3.4. Организација таскова

Програмско решење рада је организовано у шест таскова. То су праћење вредности енкодера, управљање мотор, праћење нивоа течности резервоара, LCD екран, гранични тастери и серијска комуникација. Гранични тастери и енкодер су таскови највишег приоритета, јер се они извршавају у кратким временским интервалима, и неопходни су да прекину рад остатка система. Затим следи таск у коме се управља мотором у зависности од енкодера. Као таскови са најнижим приоритетом су серијска комуникација са *Arduino* чипом и исписивање информација на LCD екрану.

Стартовање система организовано је тако да таск за серијску комуникацију са ардуином добије команду и преко редова порука их прослеђује осталим задацима. LCD екран је заједнички ресурс за праћење нивоа течности и праћење брзине и позиције предмета на траци, односно користи се мутекс семафори за синхронизацију исписа података. Бинарни семафори се користе као блокирајући елементи којима се онемогућава покретање мотора уколико ниво течности није у прописаним границама или предмет није присутан или уколико је предмет стигао до краја траке, када је потребно зауставити мотор.

3.5. Важне коришћене библиотеке

На основу потреба за реализацију задатка изабран је рад у *MPLAB X IDE* развојном окружењу, са *plugin-ом Harmony 3* који се покреће преко *MPLAB Code Configurator-a*. Овај *plugin* омогућава генерисање многобројних библиотека помоћу којих је олакшано програмирање микроконтролера. Главна библиотека због које је изабран *Harmony 3* јесте *FreeRTOS*. Осим ње, подржава и рад са свим периферијама контролера, као што су тајмери, *ADCHS, I2C, UART* и друге, али и графичку иницијализацију пинова и подешавање такта рада система као и такта рада периферија уређаја.

У наставку ће бити укратко појашњене неке од важнијих библиотека:

- *Core* – у оквиру ње можемо графички креирати таскове, доделити им називе, приоритет и поставити основна подешавања везане за њих
- *FreeRTOS* – додавањем ове библиотеке дефинишемо тип планера и омогућавамо коришћење објеката кернела попут семафора, мутекса и редова.
- *OCMP* (eng. *Output Compare*) – омогућава креирање PWM сигнала тако што је подесимо да ради у том режиму и повежемо са одговарајућим тајмером
- *ADCHS* (eng. *ADC High Speed*) – помоћу ње пратимо вредности аналогног сигнала. Неопходно је у подешавањима одабрати одговарајући ADC периферију и пин.
- *TMR* – додавање хардверских тајмера који се могу користити за праћење различитих процеса или за генерисање PWM сигнала.
- *I2C* – библиотека за серијску комуникацију са LCD екраном.

4. ЗАКЉУЧАК

Истраживање и рад са *embedded* системима у реалном времену показао се као веома захтеван задатак, који изискује знање из различитих научних дисциплина, али је и веома добра основа за даље стицање знања из те области. Главна предност ове теме јесте непходност и потенцијал оваквих поузданих система у свим гранама индустрије, али и у свакодневном животу.

5. ЛИТЕРАТУРА

[1] Јован Матић *„Real Time оперативни системи за мале embedded системе“,* Универзитет у Нишу, скрипта

[2] Поповић Иван *„Системи у реалном времену, II део*“, Електротехнички факултет, Универзитет у Београду, презентација

[3] Поповић Иван *„Наменски рачунарски системи за рад у реалном времену“*, “, Електротехнички факултет, Универзитет у Београду, електронска књига

[4] „Упутсво за коришћење PIC32MZ микроконтролер“, линк: https://download.mikroe.-com/documents/starter-boards/mini/pic32mz/mini-32-pic32-manual-v101.pdf (pristupljeno u avgustu 2023.)

Кратка биографија:

Алекса Тирнанић рођен је 20.07.1999. године у Смедереву где је завршио основну школу и гимназију. ОАС је завршио на Факултету техничких наука у Новом Саду на смеру мехатроника. Исте године је уписао МАС на мехатроиници.

Контакт: aleksa.tirnanic@gmail.com