КОНТРОЛИСАЊЕ РОБОТСКЕ РУКЕ ПОМОЋУ *WIIMOTE* КОНТРОЛЕРА
CONTROLLING THE ROBOTIC ARM USING *WIIMOTE* CONTROLLER

Александар Пајкановић, Бранко Докић, Младен Кнежић, *Електротехнички факултет Бања Лука*

Садржај - *У овом раду је описан Wiimote уређај, његова архитектура и начин повезивања са рачунарским системом. Иако је уређај произведен као бежични контролер играчке конзоле Nintendo Wii, његова примјена је знатно шира. Кроз овај рад је реализована и описана апликација која омогућава да се Wiimote контролер користи за управљање роботском руком, конструисаном помоћу Robix Rascal конструкционог сета. При томе, посебна пажња посвећена је рјешавању проблема који су посљедица некомпатибилности уређаја са којима се рачунарски систем, који служи као платформа апликацији, повезује. Поред краћег прегледа примјера линија кода апликације, наведене су имплементиране могућности апликације. Такође, детаљно је описан начин на који корисник треба да рукује Wiimote контролером да би постигао максималну ефикасност при управљању роботском руком.*

Abstract - *In this paper Wiimote device Wiimote is described, it's architecture and it's ways of connecting to a computer system. Even though this device is manufactured as a wireless controller for Nintendo Wii, it's use can be very different. Through this paper an application is created and described, which allows Wiimote to be used as a controller of the robotic arm, constructed using Robix Rascal construction set. Special attention is given to solving problems which are consequence incompatibility of the devices to which a computer system is connected. Besides short overview of the code of the application, the implemented options are listed. Also, the way of using Wiimote for achieving maximum efficiency in controlling robotic arm is described.*

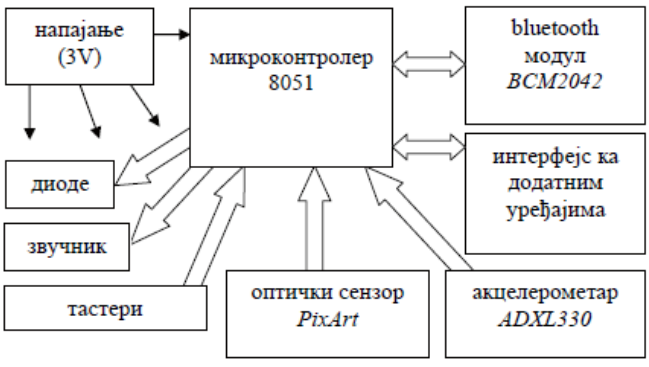
1. УВОД

Посљедња конзола јапанског Нинтенда, под називом *Wii*, унијела је нову димензију у играчко искуство. По први пут корисник је у могућности да на екран у оквиру интерактивне апликације (у овом случају игре), пренесе знање и умијеће везано за одговарајућу дисциплину у којој се такмичи, а не увјежбаност у контролисању виртуелног лика путем џојстика, џојпеда, тастатуре или миша. Све то је омогућено захваљујући уређају који је Нинтендо представио крајем 2005. а 2006. године га пустио у продају под називом *Wii Remote*, односно, скраћено, *Wiimote* [1].

Приказана је и објашњена блок шема уређаја на којој се види које компоненте га чине. Потом су појединачно размотрени и описани акцелерометар и оптички сензор, као компоненте са највећим могућностима примјене. Затим је размотрена технологија која се користи у сврху повезивања са рачунарским системом, као и поступак који је потребно провести да би се успоставила веза. Конструкциони сет *Robix Rascal* је представљен описом хардверских компоненти и начина програмирања конструисаног робота. Апликација *Интерфејс Wiimote ↔ Robix,* која користи све поменуте технологије, представљена је као посљедња потребна компонента у повезивању ова два уређаја у заједнички систем. На крају рада, у закључку, изведени су основни закључци и сумирани добијени резултати.

2. АРХИТЕКТУРА И ХАРДВЕР *WIIMOTE* УРЕЂАЈА

Поједностављена архитектура уређаја дата је блок шемом на сл.1.



Сл.1 *Поједностављена архитектура Wiimote уређаја.*

Микроконтролер је централна компонента *Wiimote* уређаја, преко које су остале компоненте повезане у јединствен систем. Као управљачка јединица користи се осмобитни *RISC* микроконтролер 8051 који управља и обавља размјену података са свим компонентама у

систему. Остале компоненте се могу подијелити на улазне, излазне и улазно-излазне.

Улазне компоненте којима је *Wiimote* опремљен су тастери, оптички сензор и акцелерометар. *Wiimote* је опремљен акцелерометром *ADXL330* и оптичким сензором *PixArt*. Акцелерометар информише рачунарски систем о кретању уређаја по све три осе, а оптички сензор осјетљив на инфрацрвену свјетлост (инфрацрвена камера) информише о кретању објеката у пољу. Овај сензор може да прати до четири тачкаста извора инфрацрвене свјетлости. Акцелерометар и оптички сензор су детаљније описани у секцији 2.1, односно 2.2.

Излазне компоненте су *LED* диоде и звучник. На горњој страни кућишта налазе се четири *LED* диоде које немају фабрички дефинисану сврху него је кориснику остављена могућност њихове произвољне употребе. Обично се користе за сигнализацију о стању уређаја или апликације са којом уређај тренутно комуницира. Звучник је такође на располагању кориснику за емитовање звука на фреквенцијама које се могу произвољно дефинисати.

Улазно-излазни уређаји су *bluetooth* модул и интерфејс ка додатним уређајима. Сва комуникација између *Wiimote* и рачунарског система се обавља бежичним *bluetooth* протоколом. Ово доприноси функционалности уређаја јер га чини контролером на даљину. Ова технологија повезивања, као и протокол по којем се повезивање врши изложени су у трећој секцији. Интерфејс ка додатним уређајима је намијењен повезивању контролера са другим периферијама истог произвођача.

2.1 Акцелерометар

Акцелерометар је уређај који мјери убрзање којем је изложен у односу на убрзање слободног пада. Како је убрзање векторска величина, потребно је поред интензитета имати информацију и о смјеру и правцу дјеловања. Због тога се производе модели који имају могућност детекције убрзања по једној или више оса. Углавном се користе да би детектовали положај уређаја, вибрацију или потрес.

У суштини, акцелерометар се понаша као маса којој је дозвољено слободно кретање. Када уређај добије убрзање, маса се измјести (у складу са законом о инерцији) до позиције у којој је могуће да маса добије исто убрзање као и кућиште уређаја. Тада се то измјештање мјери и прерачунава да би се добило убрзање као излазна величина.

Постоје разни типови акцелерометара, који се разликују како по начину израде, тако и по намјени. У конкретном случају користи се већ поменути *ADXL330*. Из спецификације произвођача [2] може се закључити да се ради о монолитном акцелерометру (све компоненте се налазе у једном интегрисаном колу) мале потрошње намијењеном за мјерење убрзања у распону $\pm 3g$ по свим просторним осама. Овај сензор може да мјери статичко гравитационо убрзање како би на основу тих мјерења могло да се одреди у којем положају у простору у односу на подлогу се уређај налази, али и динамичко убрзање које настаје као резултат кретања уређаја, вибрације или поднесеног ударца. За производњу овог акцелерометра искориштена је технологија кондензатора промјењиве капацитивности. Детекција промјене убрзања се врши по свакој оси појединачно, чиме је осигуран минималан утицај једног сензора на други [3].

2.2 Оптички сензор

Основна функција ове компоненте је да рачунарски систем увијек има информацију о тренутном положају контролера, односно информацију о правцу и смјеру у којем *Wiimote* показује. *Wiimote*, у ствари, помоћу оптичког сензора прати положај референтног осветљења. Као референтно осветљење могу се користити два тачкаста извора инфрацрвене свјетлости на константној међусобној удаљености. Сензор истовремено може да прати до четири таква извора [3], [4].

Прорачунавање положаја се обавља на сљедећи начин. Оптички сензор на контролеру „види“ двије свјетлосне тачке и у сваком тренутку има информацију о њиховој међусобној удаљености која се мијења обрнуто пропорционално са удаљеношћу *Wiimote* уређаја од референтног осветљења. На основу те промјењиве и константе која представља реалну удаљеност ова два свјетлосна извора (њихов размак је константан), могуће је израчунати положај сензора, а самим тим и контролера, у односу на референтно осветљење [3], [4].

3. ПОВЕЗИВАЊЕ *WIIMOTE* УРЕЂАЈА СА РАЧУНАРСКИМ СИСТЕМОМ

За потребе комуникације са рачунарским системом користи се *bluetooth* чип под називом *Broadcom BCM2042*. Модул *BCM2042* садржи интегрисан *bluetooth* профил *Human Interface Device (HID)* и апликације, а *bluetooth* протокол стек који се користи је у потпуности усклађен са препорукама *Bluetooth SIG* за уређаје који чине интерфејс рачунарског система и човјека [4]-[6].

Како *Wiimote* користи већ поменути *HID* профил за комуникацију са рачунарским системом, систем га препознаје као стандардни улазни уређај. Међутим, *Wiimote* не користи стандардне типове података. Свој извјештај (*report*) описује само дужином, односно форматом, док стварни садржај остаје недефинисан, због чега је бескористан уколико се користе стандардни драјвери. Умјесто тога, *Wiimote* користи прилично компликован сет операција за комуникацију, које се преносе кроз *HID* излазне (*output*) извјештаје, а као повратну информацију шаље неколико различитих врста пакета кроз *HID* улазне (*input*) извјештаје, који садрже податке његових периферних уређаја[1] [4].

[1] Називи „излазни“ и „улазни“ извјештаји су релативни, а у овом случају референца је рачунарски систем. Дакле, „улазни“

Комуникација између два уређаја се успоставља у два корака. Прво се уређаји повежу на физичком нивоу а затим започиње размјена информација између два уређаја. За отварање комуникационог канала (физички слој), у зависности од оперативног система, потребно је инсталирати одговарајући софтвер. За потребе овог рада кориштен је *BlueSoleil*. Најједноставнији начин за размјену података је кориштење емулаторског софтвера, какав је, на примјер, *GlovePIE*. Иако је то врло функционална апликација, кориснику не пружа потпуну слободу кориштења свих *Wiimote* ресурса. Проблем оваквог ограничења рјешава се употребом библиотека писаних за програмске језике.

3.1 Успостављање канала

Wiimote не захтјева ни једну од могућности аутентификације или енкрипције које *bluetooth* технологија нуди. Да би се успоставила интеракција са овим уређајем, потребно га је довести у мод рада у којем је „видљив“ рачунару, тј. извршити синхронизацију, тако што се у исто вријеме притисну тастери 1 и 2. Уколико га рачунар не препозна у наредних двадесет секунди, *Wiimote* ће да се искључи. Због тога је потребно држати притиснутим тастере 1 и 2 све док рачунар не препозна и не прихвати комуникацију са уређајем [3].

3.2 Размјена информација

Најзаслужнији за омогућавање алтернативних примјена *Wiimote* контролера је Брајан Пикс. Он је у језику C# написао библиотеку *WiimoteLib* [7], која је омогућила програмерима изузетно једноставан и елегантан механизам комуникације са *Wiimote* уређајем. Постоје и друге библиотеке писане за *Wiimote*, како у *C#* тако и у другим програмским језицима. Мајкл Лафорест је одговоран за портовање Пиксове библиотеке за програмски језик *C++,* под називом *WiiYourself* [8], а *WiiUse* је библиотека писана у *C* програмском језику.

Током развоја апликације за контролисање роботске руке, све ове библиотеке су експериментално тестиране, те је на тај начин утврђено да је *WiimoteLib* најбоља опција. Прије свега, због изузетно добро написане и садржајне документације, али и кориснику на врло једноставан начин приближеног принципа рада библиотеке кроз графичку демо апликацију.

4. *ROBIX RASCAL* КОНСТРУКЦИОНИ СЕТ

Конструкциони сет се одликује једноставношћу конструкције, али и програмирања. За то су првенствено заслужни сервомотори *HS-422* и библиотека *C++* језика, *RascalDLL* коју је могуће преузети заједно са софтвером, *Rascal 0.2.23*, са Интернет странице произвођача [9].

Сервомотори функционишу по врло једноставном принципу. Од три улаза, колико их имају, на један се повезује позитиван једносмјерни напон, на други маса, док је трећи улаз контролни. Напон напајања може бити различит, али у случају *HS-422* користи се 5V [10]. Сигнал који се доводи на контролни улаз одређује да ли се сервомотор креће или не. У овом конкретном случају, облик сигнала који може да покрене сервомотор је низ правоугаоних импулса фреквенције 50Hz. Фактор попуне низа правоугаоних импулса одређује положај у који ће сервомотор да се постави након пријема поворке. Како је могуће да кретање унутар угла од 180°, мотор може да заузме било који положај од 0° до 180°, ако за почетак угаоног координатног система узмемо крајњи лијеви положај, а за крај крајњи десни.

Већ је поменуто да је фреквенција освјежавања $50Hz$, што значи да на сваких $20ms$ треба да се појави нови импулс. Ако висок ниво импулса траје $700\mu s$ мотор ће заузети крајњи лијеви (нулти) положај, а ако је $2300\mu s$ крајњи десни (максимални). Задавањем било које вриједности трајања високог нивоа импулса у оквиру ове двије граничне вриједности, мотор ће заузети угао који је пропорционалан тој вриједности. Тако, на примјер, ако узмемо да висок ниво траје $1500\mu s$ мотор ће заузети централни положај. На описани начин електронски интерфејс је у стању да истовремено контролише до шест сервомотора [10].

Иако је програмирање конструисаног робота могуће и помоћу одговарајућег скрипт језика, у овом раду је кориштена поменута *C++* библиотека, *RascalDLL*. До ове одлуке се дошло такође током израде рада, када је експериментално утврђено да скрипт језик не пружа довољну флексибилност програмирања и приступа ресурсима *Robix Rascal* конструкционог сета.

5. ИНТЕРФЕЈС *WIIMOTE – ROBIX RASCAL*

Да би се могло размишљати о самом повезивању ова два уређаја, потребно је на неки начин омогућити њихово споразумијевање. С обзиром на то да за кориштени робот постоји само једна библиотека писана за програмски језик *C++*, логично је да се за комуникацију са *Wiimote* изабере библиотека *WiiYourself*. Међутим, током израде апликације која би служила као интерфејс, недостатак квалитетне документације за ову библиотеку се показао као пресудан. Како је најквалитетније урађена библиотека *WiimoteLib*, она је сљедећи логичан избор. Међутим, јавља се проблем кориштења *RascalDLL* у развојном окружењу *C#* програмског језика.

Да бисмо могли користити код који није написан у контролисаном *C# .NET* окружењу, него у *C++*, морамо да користимо посредан начин. За то нам је потребна тзв. *Pinvoke* технологија [11]. Ова технологија омогућава позивање функција које се налазе у *dll* фајловима који нису писани у контролисаном *.NET* окружењу. Да би се функције могле позивати у C# језику најприје је потребно импортовати *dll* фајлове које желимо да користимо на сљедећи начин:

```
[DllImport("RascalDLL.dll")].
```

извјештај подразумјева да га шаље *Wiimote*, а „излазни“ шаље рачунарски систем.

Након тога наводимо прототипе функција које ћемо користити, с тим да постоји мала измјена. Неки од типова података су некомпатибилни између ова два програмска језика. Тако, на примјер, умјесто `char *` типа податка у прототип функције користимо `StringBuilder`. Такође, када се промјењљива преноси по референци умјесто `&` користи се `ref int`. Да бисмо илустровали претходно наведено показаћемо како треба да изгледа један прототип функције на конкретном примјеру из *RascalDLL* библиотеке:

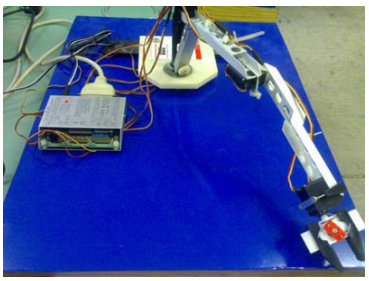
```
public static extern Int32 rbxRobotCreate
(StringBuilder RobotName, ref int
RobotHandle);
```

Битно је напоменути да у случају кориштења *Pinvoke* технологије није потребно позивати функције (које су иначе неопходне да би *RascalDLL* библиотека функционисала у *C++* окружењу):

```
rbxDLLInitialize();
rbxDLLFree();
```

зато што учитавање имплицитно обавља наведена линија кода, а ослобађање ресурса се обавља аутоматски по затварању апликације [11], [12].

С обзиром да *Wiimote*, преко сензора за мјерење убрзања, може да детектује покрет, најприроднији начин контролисања робота је да се креће у складу са покретима руке корисника. Прва мисао која се намеће је да се ограничи простор у којем би корисник помјерао руку, и да се тај ограничени простор сматра координатним системом корисника, а да се простор у којем би се робот кретао сматра координатним системом робота. Сада се трансформацијама координата, које добијамо на основу положаја руке корисника, из једног координатног система, могу прорачунати координате положаја у који би робот требао да дође у другом координатном систему. Како корисник држи *Wiimote* покретна су му два зглоба, раме и лакат. Према томе, и роботу су довољна два зглоба, односно сервомотора, тако да ћемо у даљем тексту умјесто о роботу говорити о роботској руци (сл.2).



Сл.2 *Конструкција описане роботске руке.*

Међутим, овакав начин прерачунавања координата је озбиљан математички проблем који излази из оквира имплементације овог интерфејса. Стога је за реализацију апликације кориштен сљедећи начин комуникације.

У зависности од угла под којим *Wiimote* у руци стоји по *X* и по *Y* оси, роботска рука се креће улијево или удесно, односно нагоре или надоле. На овај начин добијамо да се роботска рука креће у односу на свој релативни положај, а не у односу на неку апсолутну референтну тачку фиксираног координатног система. Другим ријечима, кад год се робот помјери, центар координатног система робота се, такође, помјери у његов тренутни положај.

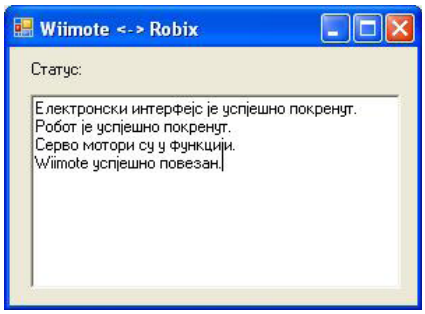
Такође, с обзиром да роботску руку корисник контролише у реалном времену, погодна опција коју треба имплементирати је да се она по обје осе може кретати различитим брзинама. Дакле, у зависности од тога колико корисник нагне *Wiimote* у својој руци, бираће брзину закретања одговарајућег сервомотора – већи угао значи брже закретање, и обрнуто.

Да би робот био функционалнији и да би могао да обавља додатне радње осим помјерања по двије осе (што је довољно за заузимање било којег положаја у тродимензионалном простору) потребна је и хватаљка, видљива на сл.2. Да би она била потпуно функционална, повезујемо је механички са додатним, трећим, сервомотором који омогућава ротацију. Сама хватаљка се реализује помоћу четвртог сервомотора, којим се реализује отварање или затварање. Све укупно, дакле, роботска рука се састоји од четири сервомотора, што је електронски интерфејс у стању да подржи, јер може истовремено да контролише до шест сервомотора [9].

Како је нагињање лијево или десно резервисано за помјерање „раменог“ сервомотора који служи да покреће руку по хоризонталној равни, за ротацију хватаљке користимо тастер 'B' на *Wiimote*. Када тастер није притиснут, нагињање лијево или десно помјера први сервомотор, а када је притиснут, трећи. Четврти сервомотор, односно хватаљку, контролишемо тастерима *PLUS* и *MINUS*, тако што првим скупљамо а другим ширимо хватаљку.

Поред постигнуте функционалности, корисно је имплементирати и могућност активирања и деактивирања утицаја *Wiimote* на кретање роботске руке, како случајни покрет руке корисника не би довео до непредвиђених и неочекиваних кретњи робота. У овом случају, корисник активира везу притиском на тастер 1, а деактивира притиском на тастер 2.

Описани систем имплементиран је у оквиру апликације која нема никакав интерфејс са корисником јер за тим нема потребе. Сва учитавања се обављају аутоматски по њеном покретању, а сви ресурси се ослобађају такође аутоматски по њеном затварању. Једина комуникација са корисником је то што даје извјештај о успјешно обављеном задатку или проблему који се јавио, те о чему се у ствари ради (сл.3.) [12], [13].



Сл.3 Wiimote ↔ Robix.

6. ЗАКЉУЧАК

На претходном описаном примјеру показано је да се *Wiimote* уређај може користити за знатно озбиљније сврхе него што су Нинтендови инжењери планирали. Уређај се показао као врло једноставан и функционалан, као и *Robix* конструкциони сет. Као резултат њиховог повезивања добија се потпуно функционална роботска рука. Примјена овакве даљински управљане роботске руке заиста је широка. На примјер, при раду са радиоактивним елементима у лабораторијама се користе роботи који функционишу на сличном принципу, а цијена им је неупоредиво већа. Додатна предност оваквог система је што се ствара изузетан осјећај виртуелне стварности, управо због тога што *Wiimote* обезбјеђује тренутну реакцију.

Током писања овог рада било је потребно одлучити коју од постојећих библиотека користити за комуникацију са *Wiimote* уређајем, као и на који начин програмирати роботску руку. Искориштена су рјешења која су се у тестној фази развоја апликације показала као најефикаснија. Тако је експериментално утврђено да је најједноставнија за употребу и са највише могућности *WiimoteLib* библиотека Брајана Пикса, те да је за програмирање роботске руке примјеренији директан приступ *RascalDLL* библиотеци.

Занимљив је и проблем који је у овом раду избјегнут. Умјесто директно прерачунавања апсолутних координата тренутног положаја руке корисника, у раду је кориштен принцип помјерања роботске руке у односу на њен тренутни положај, заснован на нагињању контролера под одређеним углом. Овај проблем је математичке природе, али би његовим рјешавањем, те његовом имплементацијом у код постојеће или нове апликације, систем *Wiimote – Robix* добио потпуно нову димензију. Корисно би било и када би се између ове двије компоненте умјесто рачунара користио микроконтролер. С једне стране, то би захтјевало знатне измјене апликације док би с друге стране вишеструко умањило димензије самог система, а тиме увећало његову преносивост и проширило употребљивост.

7. ЛИТЕРАТУРА

[1] www.wikipedia.org посјећено: јануара 2010. године;

[2] www.analog.com/static/imported-files/data_sheets/ADXL330.pdf, посјећено: јануара 2010. године;

[3] Johnny Chung Lee, *Hacking the Nintendo Wii Remote*, Pervasive Computing, бр. 13. јули-септембар 2008. године, стране 39-45;

[4] www.wiibrew.org/wiki/Wiimote, посјећено: јануара 2010. године;

[5] www.broadcom.com/products/Bluetooth/Bluetooth-RF-Silicon-and-Software-Solutions/BCM2042, посјећено: јануара 2010. године;

[6] David Krammer, Gordon McNutt, Brian Senese, Bluetooth Application Developer's Guide, Syngress Publishing Inc. 2002.

[7] www.brianpeeks.net, посјећено: јануара 2010. године;

[8] www.wiiyourself.gl.tter.org, посјећено: јануара 2010. године;

[9] www.robix.com, посјећено: јануара 2010. године;

[10] www.hitecrcd.com/servos/list, посјећено: јануара 2010. године;

[11] www.pinvoke.net, посјећено: јануара 2010. године;

[12] Herbert Shildt, *C# 3.0: A Beginner's Guide*, The McGraw-Hill Companies, 2009.