Bartosz PAWŁOWICZ [1], Michał SIERPIŃSKI [2], Bartosz TRYBUS [3]
[1] DEPARTMENT OF ELECTRONIC AND COMMUNICATION SYSTEMS
[2] STUDENTS SCIENTIFIC CIRCLE OF INFORMATION TECHNOLOGY
[3] DEPARTMENT OF COMPUTER AND CONTROL ENGINEERING
Rzeszow University of Technology, 12 Powstańców Warszawy St., 35-959, Rzeszow, Poland

Controlling a robot swarm with EV3 modules for mobile indoor mapping

Abstract

The goal of the paper is to present a lab application controlling a swarm of mobile robots built with Lego Mindstorms bricks of EV3 series. The application named *BrickCenter* is based on Windows Runtime (WinRT) architecture and allows the user to manage robots by controlling selected motors through a keyboard or a gamepad. The application is able to read measurements acquired from standard sensors connected to EV3 controllers. It is a basis for a concept of supervisory control of a robot swarm designed for indoor mapping with RFID readers.

Keywords: robot swarm, RFID, WinRT, EV3.

1. Introduction

Nowadays, one of the most interesting and common topics in literature is the subject of “intelligent” autonomous robots designed to conduct mapping of indoors, to help people in wheelchairs or the blind or simply to bring coffee from the kitchen to the dining room.

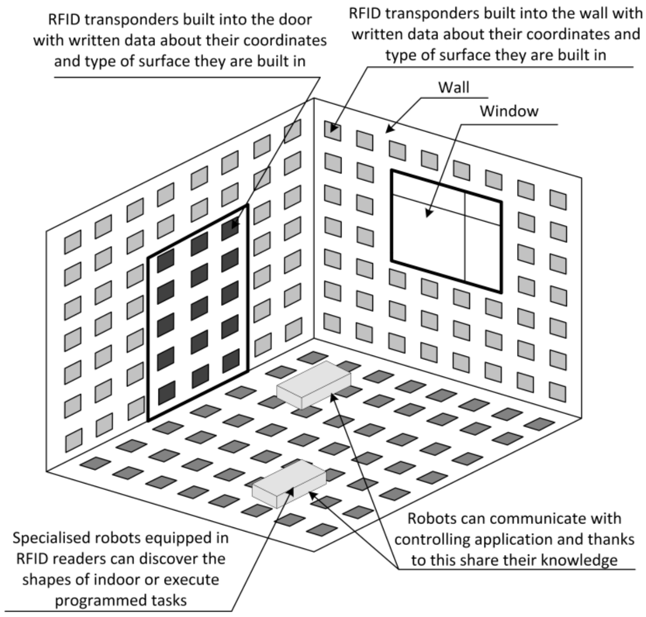
RFID transponders built into the door with written data about their coordinates and type of surface they are built in
RFID transponders built into the wall with written data about their coordinates and type of surface they are built in
Wall
Window
Specialised robots equipped in RFID readers can discover the shapes of indoor or execute programmed tasks
Robots can communicate with controlling application and thanks to this share their knowledge

Fig. 1. The idea of indoor mapping using programmed RFID transponders

One of the most important functions of this kind of robots is to gain knowledge about maps of indoors where the robot is moving or to move according to a programmed trajectory. One of the known ideas of indoor mapping is usage of RFID transponders built into walls, floors, doors and furniture [1, 2]. In this concept, the robot equipped with RFID readers can map the indoor area using information and coordinates stored in a net of RFID transponders (Fig. 1). In the proposed approach, a single robot is replaced by a swarm of robots, which can be specialized in their functioning. The concept requires to control the swarm of robots by a central supervisory controller with adequate computing power. Such a central point can gain knowledge about indoors by involving multiple robots in the swarm at the same time, thus faster, more efficiently and more reliably. By transmitting the acquired data to a central store, the robots can share the knowledge to improve the mapping process. The robots can work in different ways as shown in Fig. 2. In comparison to an example surface of single robot mapping (Fig. 2a), a swarm can be used to verify measurements of a single robot (Fig. 2b), independent mapping of two or more areas (Fig. 2c), and mapping multiple areas with automatic verification (Fig. 2d).

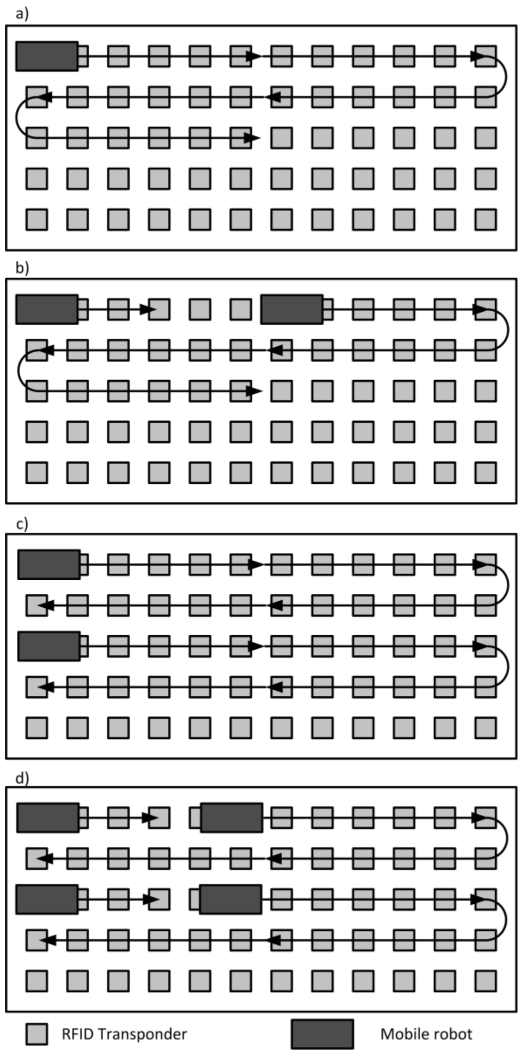

Fig. 2. Robot swarm work modes: a) single robot, b) verification of measurements, c) parallel scanning, d) parallel scanning with verification

2. Components of the robot swarm system

The schematic diagram of the robot swarm control system with central controlling application is presented in Fig. 3. The application BrickCenter is tightly connected with the hardware platform on which robots of the swarm are built, i.e. Lego Mindstorms EV3. The choice of the platform was determined by a low price of elements (bricks) and flexibility in prototyping various robots in different shapes and functions. An important advantage of the platform is the open Hardware Development Kit which allows building custom devices depending on the needs, not only using peripherals designed by Lego. Lego Mindstorms EV3

(Evolution) is a microcontroller-based system equipped with a set of input/output ports being able to connect with up to 4 sensors and servomechanisms. The set also contains a LCD display, a speaker and five illuminated buttons. It can be connected to a computer using USB or Bluetooth. As an USB host, the EV3 controller is able to connect to a Wi-Fi network via an USB card. By default, EV3 uses the Linux operating system, making it a fully operational microcomputer. EV3 is also equipped with a microSD card reader, allowing running an alternative operating system from a memory card. The users are able to design application programs in a graphical manner based on the LabView programming language. There is also a possibility to write programs in various programming languages such as C# or in the assembly language. Aside from working as a single device, the EV3 controllers may work in a daisy-chain system consisting of 2 up to 8 devices connected via Bluetooth or USB. The transmission speed in this mode is roughly 12 Mb per second [3, 4].

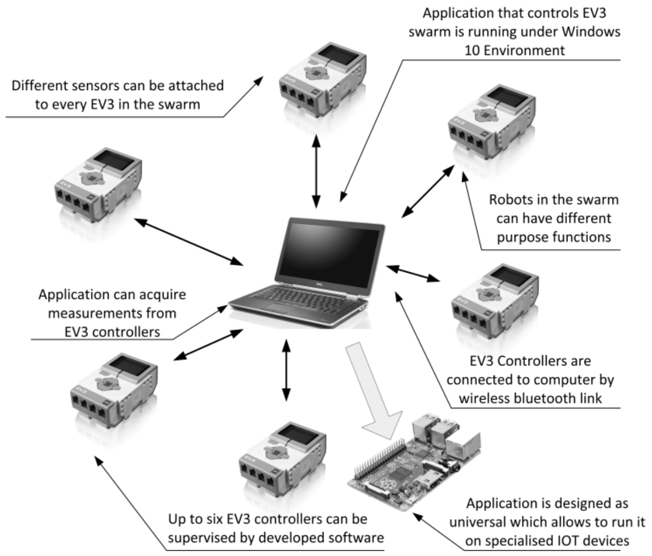
Application that controls EV3 swarm is running under Windows 10 Environment
Different sensors can be attached to every EV3 in the swarm
Robots in the swarm can have different purpose functions
Application can acquire measurements from EV3 controllers
EV3 Controllers are connected to computer by wireless bluetooth link
Application is designed as universal which allows to run it on specialised IOT devices
Up to six EV3 controllers can be supervised by developed software

Fig. 3. Elements of the robot swarm control system

3. Application design

The application BrickCenter which controls the swarm of robots is based on the new Windows Runtime platform (WinRT) [5, 6]. The main idea behind using this architecture was to design a universal application which would run on different hardware targets, especially various IoT (Internet of Things) devices. As the result, the control software may be run not only on a classic PC but also on Raspberry Pi 2/3.

The application is designed with flexibility in mind and is able to control different robots based on EV3 controllers. The robots in the swarm can be also controlled manually. Every action of robots in the swarm is supervised by configurable events which are defined in the application by the user.

BrickCenter is able to access all the data acquired from standard sensors included in Lego Mindstorms EV3, e.g. ultrasonic, infrared, touch, light and color sensors. The sensor readouts are presented on the operating screen (touch panel) as easy-to-read visual components. Furthermore, the application has the functionality to connect to multiple EV3 controllers simultaneously by either of available transmission mediums e.g. USB, Bluetooth, Wi-Fi. The user can create custom configuration profiles to handle multiple setups.

An important feature of BrickCenter is a data server which allows passing the data collected from sensors to external software. Such behavior widens the functionality of the system and allows data analysis with some advanced intelligent processing package. Sharing the data with other electronic control systems is also possible.

The application BrickCenter provides management of the robot swarm and handling of complex actions. For this purpose, it was necessary to create a flexible data model compatible with dynamic changes occurring in the application. The application structure consists of a set of classes specialized to perform only a portion of necessary logic. Such an approach greatly improved the clarity of the implementation from the programmer’s point of view. Due to the distribution of code, further changes and future improvements do not require code factorization [7]. The main classes of BrickCenter can be categorized as follows.

- Data models – designed to create a transparent model of the acquired data. Objects of these classes include all necessary information to send the data to external programs.
- Controllers – providing application management functions and data recording.
- Pages – providing the user the ability to interact with the application.
- Converters – implementing the conversion between types thanks to preset rules.
- Option panels – responsible for creation and management of options and settings.
- Template selectors – allowing for dynamic changes in the interface with the ability to select the right model for visual components (controls);
- Controls – visual components allowing the user to properly interact with the program.

The application BrickCenter was written in C# and C++ programming languages. Two external libraries were used to properly interact with the hardware components, namely:

- LEGO Mindstorms EV3 API – providing basic connectivity features and EV3 controller support.
- GameController – C ++ library used to connect and to capture events using Xbox 360 gamepad.

4. Data structures

BrickController is one of the fundamental classes of the application, designed to hold all the information about a single EV3 controller, such as its name, identifier, currently used manual input method (gamepad or keyboard) and list of all peripherals connected to input/output ports of EV3.

```
[DataMemberAttribute]
public string Name {get; set; }
[DataMemberAttribute]
public string Id { get; set; }
[DataMemberAttribute]
public Brick Brick;
[DataMemberAttribute]
Public Dictionary<string,
ObservableCollection<InputEvent>> EventList { get;
set; }
[DataMemberAttribute]
public ObservableCollection<Sensor> Sensors { get;
set; }
```

Fig. 4. Structure of the serializable part of *BrickController* class

To allow serializing the information, data contracts have to be created, indicating which variables and properties should undergo serialization and are used later to recreate the serialized object. Here, the serialization involves JSON Data Interchange Format [8,9]. For this reason, the *BrickController* class is extended with *DataMemberAttribute* (Fig. 4**Błąd! Nie można odnaleźć źródła odwołania.**) for variables and properties which will be serialized in the final JSON string, e.g. *Name*, *Id* or collection of *Sensors*.

```
[DataMemberAttribute]
public string Name { get; set; }
[DataMemberAttribute]
public ObservableCollection<Motor> Motors
{get;set;}
```

Fig. 5. Basic properties of the *InputEvent* class

The handling of multiple events invoked by the user required to create a class standardizing the way different types of input devices are managed. This task is performed by the class *InputEvent* which also may be serialized (like *BrickController*). For an event causing an action, it contains its *Name* and a list of all the motors engaged when the event gets fired (Fig. 5).

The class *Sensor* stores the data collected from one of the sensors connected to the EV3 controller. For the purpose of transferring the sensor data outside of the application, the class contains serializable variables and properties (Fig. 6).

```
[JsonConverter(typeof(StringEnumConverter))]
[DataMemberAttribute]
public InputPort BrickInputPort { get; set; }
[JsonConverter(typeof(StringEnumConverter))]
[DataMemberAttribute]
public DeviceType _sensorType;
```

Fig. 6. Data structures containing the information acquired from the input port of the EV3 controller

The attribute *JsonConverter* in Fig. 7 was used to serialize the values of enumerable types (such as *InputPort* or *DeviceType*) as strings instead of numbers.

```
public InputPort BrickInputPort { get; set; }
[JsonConverter(typeof(StringEnumConverter))]
[DataMemberAttribute]
public OutputPort BrickOutputPort { get; set; }
[JsonConverter(typeof(StringEnumConverter))]
[DataMemberAttribute]
public DeviceType motorType { get; set; }
[DataMemberAttribute]
public int PowerRatingMovement { get; set; }
```

Fig. 7. Data structures containing the information about a motor

The class *Motor* holds the data acquired from one of the motor ports of the EV3 controller and is designed to allow easy management of its resources. The class contains three properties (Fig. 7):

- *BrickInputPort* of the type *InputPort* indicates the controller port to which the data is sent.
- *BrickOutputPort* of the type *OutputPort* indicates the controller port from which data can be received.
- *motorType* of the type *DeviceType* stores the type of the motor connected to the controller;

5. Communications

In order to communicate with the EV3 controller, Lego Mindstorms EV3 API was used in the BrickCenter application. The library provides basic communication functions and management of input/output ports and connected devices. Three communication technologies are available in the application, namely Bluetooth, Wi-Fi and USB. It is possible to communicate with multiple EV3 controllers simultaneously.

```
RfcommDeviceService.GetDeviceSelector(RfcommServic
eId.SerialPort);
DeviceInformationCollection devices =
  await DeviceInformation.FindAllAsync(selector);
DeviceInformation device = (from d in devices
where d.Id == _deviceId select
d).FirstOrDefault();
```

Fig. 8. Selecting the desired EV3 controller from all available devices

Unfortunately, the library is not able to distinguish the controllers, which are all named "EV3" by default. To bypass this, there was introduced a new variable *Id* containing a unique identifier, which allows differentiating them. The code excerpt from Fig. 8 is used by the application to find the connected EV3 module with the given *_deviceId*.

If any of the controllers gets disconnected, the application is experiencing an exception. To handle such a situation, the additional event <*BrickDisconnected*> was implemented [10]. It is used as shown in Fig. 9 to inform the user about the disconnection.

```
catch(System.Exception)
{BrickDisconnected?.Invoke(this, new
BrickDisconnectedEventArgs() {Details =  "Brick
disconnected due to unexpected behavior" });
```

Fig. 9. Notifying the user about an unexpected disconnection of EV3

BrickCenter allows the binding of motor engaging or disengaging with specific events, e.g. a keyboard key press. For this purpose, a mechanism was developed to verify whether an event should result in some action.

```
foreach (BrickController brick in MainPage.Bricks)
{
if (brick.isConnected && brick.Type ==
    Enums.ControllerType.Keyboard)
{
  InputEvent inputEvent =
    brick.EventList["Keyboard"]
    .FirstOrDefault(x => x.Name == key);
  if (inputEvent != null)
    foreach (Motor motor in inputEvent.Motors)
   await
brick.Brick.DirectCommand.TurnMotorAtPowerAsync(mo
tor.BrickOutputPort, motor.PowerRatingMovement);
}}
```

Fig. 10. Invoking motors by *InputEvent* object

Two functions were implemented to handle keyboard events, i.e. *KeyDown* and *KeyUp*, which accepts a string identifier of the key which has been pressed. The code in Fig. 10 shows how the key events are mapped to the motor operations.

First, status of the connection with EV3 is checked (using the property *isConnected*). If an appropriate key has been pressed, the *inputEvent* object is used to turn on all the motors bound to the event. Similar instructions are executed in case of other input devices (gamepad). In the case of an analog user input (e.g. gamepad with a joystick), a conversion is performed to map the tilt value to the speed of motors.

As said, BrickCenter is able to work as a data server for external client applications. All the data acquired from EV3 devices and their sensors is stored in a common format (JSON). The data can be transmitted to external applications using a client-server connection over WebSocket protocol or via a simple access file with adjustable frequency of data update.

6. Conclusions

The BrickCenter application presented in the paper was successfully tested in the laboratory. It was able to manage a swarm of six robots based on EV3 controllers. The robots had different characteristics and tasks. It seems that such specialization can improve the process of indoor mapping. The multitasking operating environment based on WinRT is also an important factor, allowing incorporating various functions, such as data sharing, handling user input or operator interface.

7. References

[1] Kalita W., Skoczylas M.: Application of RFID Technique for Object Location on Plane Area. XXXIII International Conference of IMAPS CPMT IEEE Poland Chapter, Gliwice-Pszczyna, pp. 396-400, 21-24.09.2009.
[2] Kalita W., Skoczylas M., Węglarski M.: Application of RFID Transponders Integrated with Sensors in Navigation System of Autonomous Object. 34th International Microelectronics and Packaging IMAPS-CPMT Poland Conference, p. 201, Wrocław, 22-25.09.2010.
[3] Garber Gary: Learning LEGO MINDSTORMS EV3. Packt Publishing Ltd, 2015.
[4] Rollins M.: Beginning LEGO MINDSTORMS EV3. Apress, 2014.
[5] Mayberry M.: WinRT Revealed. Apress, 2012.
[6] Likness J., Garland J.: Programming the Windows Runtime by Example. Addison-Wesley Professional, 2014.
[7] Powers L., Snell M.: Microsoft Visual Studio 2015 Unleashed. (3rd Edition), Sams, 2015.
[8] Standard ECMA-404: The JSON Data Interchange Format, 1st Edition, 2013, http://www.ecma-international.org/publications/ files/ ECMA-ST/ECMA-404.pdf
[9] Johnson B.: Professional Visual Studio 2015. 1st Edition, 2015.
[10] Albahari B., Albahari J.: C# 5.0 in a Nutshell. 5th Edition, O'Reilly Media, 2012.

Received: 07.05.2016 Paper reviewed Accepted: 01.07.2016

Bartosz PAWŁOWICZ, PhD, eng.

Currently working at Rzeszow University of Technology in Department of Electronic and Communications Systems. Mainly involved in a research team focusing on the development of RFID contactless identification techniques and coordination of the work of the Students Scientific Circle of Electronics and Information Technology at the Faculty of Electrical and Computer Engineering.

e-mail: barpaw@prz.edu.pl

Michał SIERPIŃSKI, eng.

Currently studying towards MSc degree in computer science at Rzeszow University of Technology. Leader of Student Scientific Circle of Programmers at the Faculty of Electrical and Computer Engineering. Currently involved in swarm intelligence and robot control.

e-mail: michal.sierpinski@gmail.com

Bartosz TRYBUS, PhD, eng.

Assistant professor at the Department of Computer and Control Engineering, Rzeszów University of Technology. He received PhD in computer engineering from AGH University of Science and Technology, Cracow. His research involves real-time systems and runtime environments of control applications.

e-mail: btrybus@prz.edu.pl