Urządzenia Peryferyjne

**Ćwiczenie 14 – Bluetooth -> komunikacja z telefonem komórkowym**

Prowadzący: Dr Inż. Jan Nikodem

Grupa: Poniedziałek tydzień parzysty godz. 10:15

Data wykonania ćwiczenia: 13.11.2017r.

Wykonali : Paweł Biel 225949

Oskar Szubert 213624

1. Zagadnienia
   1. Mechanizm rejestracji funkcji callback’owych

Funkcje callback’owe inaczej wywołania zwrotne są odwrotnością normalnych funkcji, które wywołujemy podczas programowania. W tym przypadku użytkownik jedynie rejestruje funkcję do późniejszego wywołania, natomiast funkcje biblioteki wywołają ją w stosownym dla siebie czasie.  Kiedy następuje wywołanie zwrotne, funkcja, która je wykonuje, nie wie nic o tym, co się zdarzy. To zależy od tego, co zostało zarejestrowane do wywołania.  Rejestrowanie funkcji polega na stworzeniu własnej klasy na bazie innej klasy oraz przedefiniowaniu w niej odpowiedniej metody tak, aby wykonywała ona inne czynności, niż to jest założone w jej klasie bazowej. Wywołanie zwrotne odbywa się wtedy w ten sposób, że inna funkcja woła metodę na podanym obiekcie, ale faktyczna procedura, która zostanie wywołana, może być różna, w zależności od tego, jakiej klasy jest podany obiekt.

* 1. Komunikacja poprzez Bluetooth

Standard komunikacji bezprzewodowej opisany normami IEEE 802.15.1. Określane są trzy klasy mocy nadawczej:

* Klasa 1 – 100mW do 100m
* Klasa 2 – 2,5 MW do 10m
* Klasa 3 – 1mW do 1m

Standard korzysta z [fal radiowych](https://pl.wikipedia.org/wiki/Fale_radiowe) w paśmie [ISM](https://pl.wikipedia.org/wiki/Pasmo_ISM) 2,4 [GHz](https://pl.wikipedia.org/wiki/Herc). Na terenie Polski akceptowalne przedział częstotliwości wynosi 2,4 - 24,25 GHz . Bluetooth jest standardem cały czas rozwijanym:

* Bluetooth 1.0 – 21 [kb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99)
* Bluetooth 1.1 – 124 [kb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99)
* Bluetooth 1.2 – 328 [kb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99)
* Bluetooth 2.0 + EDR – wprowadzenie Enhanced Data Rate zwiększyło transfer teoretyczny do 2,1 [Mb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99) (około 3 [Mb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99) wliczając narzut protokołu)
* Bluetooth 2.1 + EDR - uproszczenie i ujednolicenie procesu parowania urządzeń BT, wsparcie dla przyszłych implementacji [NFC](https://pl.wikipedia.org/wiki/Near_Field_Communication), zmniejszenie zużycia energii[[4]](https://pl.wikipedia.org/wiki/Bluetooth#cite_note-4)
* Bluetooth 3.0 + HS (High Speed) – 24 [Mb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99) (3 MB/s)
* Bluetooth 3.1 + HS (High Speed) – 40 [Mb/s](https://pl.wikipedia.org/wiki/Bit_na_sekund%C4%99) (5 MB/s)
* Bluetooth 4.0 + LE (Low Energy) – 1 Mb/s znacząco ograniczono pobór energii, kosztem obniżonego transferu oraz zwiększono realny zasięg działania do 100 m
* Bluetooth 4.1 - standard opracowany do zastosowania  
  w tzw. "intrenecie rzeczy", umożliwiający bezpośrednią łączność przedmiotów z internetem
* Bluetooth 4.2 - w stosunku do poprzednich wersji: szybszy transfer, wyższy poziom bezpieczeństwa, nawiązanie  
  łączności z przedmiotami - łatwiejsze
* Bluetooth 5.0 - ujednolicenie wersji, szybszy transfer – 2 Mb/s   
  dla przedmiotów łączących się z „internetem rzeczy” i 50 Mb/s   
  do normalnych, realny zasięg działania do 140m

Każde urządzenie ma 48 bitowy adres IEEE MAC (Bluetooth Device Address, BD\_ADDR) i jest on używany do inicjowania pewnych operacji oraz obliczania kodu dostępu.

Moduł Bluetooth wyposażony jest w 28-bitowy wewnętrzny zegar, który determinuje synchronizację i skakanie po częstotliwościach. Nigdy nie jest on dostrajany, ani wyłączany. Do synchronizacji z innym modułem Bluetooth wykorzystywana jest różnica (offset) pomiędzy zegarami jednostek chcących  
się komunikować. Częstotliwość zegara wynosi 3,2 kHz.

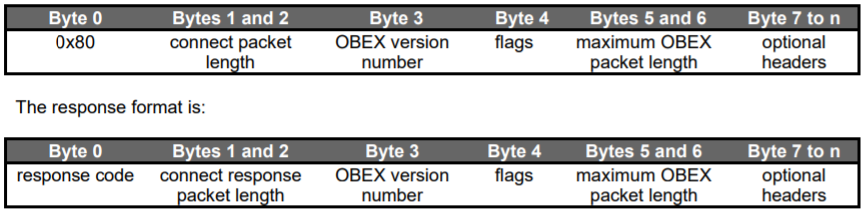
* 1. Protokół transferu plików OBEX i przygotowane polecenia

**OBEX - Object Exchange** - protokół komunikacyjny, określający procedury wymiany danych binarnych między urządzeniami.

Specyfikacja protokołu OBEX opiera się architekturze klient - serwer. Klient wykorzystuje zaufane medium transportowe do połączenia z serwerem w celu zażądania transmisji obiektów. Przesyłane obiekty są zapisane w formacie binarnym.

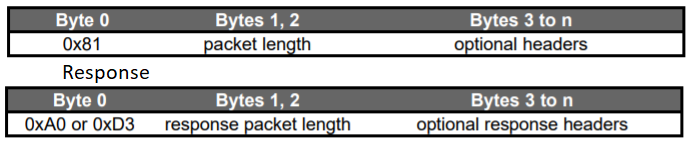
* Polecenie CONNECT

Ta operacja inicjuje połączenie i określa podstawowe oczekiwania każdej strony łącza.

**sass**

Żądanie i odpowiedź CONNECT muszą się mieścić w jednym pakiecie.

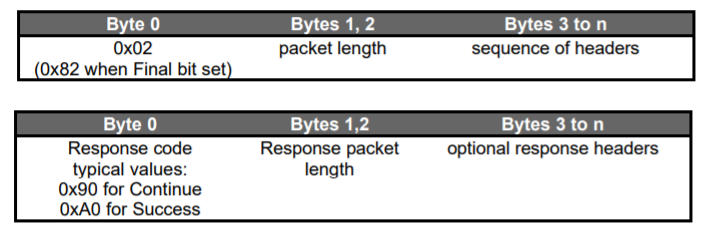
* Polecenie DISCONNECT

Ten kod operacyjny sygnalizuje koniec sesji OBEX. Może zawierać dodatkowy nagłówek Description- czyli informacje czytelne dla użytkownika. Żądanie i odpowiedź DISCONNECT muszą się mieścić w

-0xA0 Operacja udana

- 0xD3 Service Unavailable

* PUT

Operacja ta wysyła obiekt z klienta do serwera. Request standardowo powinien zawierać nagłówki: nazwa i długość. Może występować także nagłówek Data/Time. Jednakże istnieje możliwość pominięcia wszystkich headerów- jeżeli urządzenie docelowe jest bardzo proste i może odpierać/obsługiwać tylko jeden typ obiektu a komunikacja z niezaufanych źródeł jest niemożliwa.

1. Kod programu
2. using System;
3. using System.Collections.Generic;
4. using System.ComponentModel;
5. using System.Data;
6. using System.Drawing;
7. using System.Linq;
8. using System.Text;
9. using System.Threading.Tasks;
10. using System.Windows.Forms;
11. using InTheHand.Net;
12. using InTheHand.Net.Bluetooth;
13. using InTheHand.Net.Sockets;
14. using System.Threading;
15. namespace Bluetooth
16. {
17. public partial class Form1 : Form
18. {
19. public const string DEVICE\_PIN = "000000";
20. private bool continueScanning = false;
21. private bool isPaired = false;
22. private bool isUnpaired = false;
23. private List<BluetoothDeviceInfo> devices = new List<BluetoothDeviceInfo>();
24. BluetoothDeviceInfo deviceToPair = null;
25. public Form1()
26. {
27. InitializeComponent();
28. textBoxConsole.ReadOnly = true;
30. }
31. // Funkcja wyszukująca urządzenia oraz ich typ i adres
32. private void ScanDevices()
33. {
34. try
35. {
36. while(continueScanning)
37. {
38. var bluetoothClient = new BluetoothClient();
39. var bluetoothDevices = bluetoothClient.DiscoverDevices();
40. foreach(var bluetoothDevice in bluetoothDevices)
41. {
42. var bluetoothInfo = string.Format("Nazwa urządzenia: {0}\r\nTyp urządzenia: {1} Adres: {2}",
43. bluetoothDevice.DeviceName, bluetoothDevice.ClassOfDevice.Device,bluetoothDevice.DeviceAddress);
44. textBoxConsole.Text += bluetoothInfo + "\r\n";
45. bool onList = false;
46. if(bluetoothDevice.DeviceName!=null)
47. {
48. foreach (string item in listBoxDevices.Items)
49. if (item == bluetoothDevice.DeviceName)
50. onList = true;
51. }
52. if(!onList)
53. {
54. listBoxDevices.Items.Add(bluetoothDevice.DeviceName);
55. devices.Add(bluetoothDevice);
56. }
57. }
58. }
59. }
60. catch(Exception error)
61. {
62. textBoxConsole.Text += "Wystąpił błąd!\r\n" + error.Message + "\r\n";
63. }
64. }
65. // funkcja która wywołuje szukanie urządzeń po kliknięciu przycisku
66. private void buttonFindDevices\_Click(object sender, EventArgs e)
67. {
68. Thread findDevices = new Thread(ScanDevices);
69. findDevices.Priority = ThreadPriority.Highest;
70. findDevices.IsBackground = true;
71. findDevices.Start();
72. if(buttonFindDevices.Text == "Szukaj urządzeń")
73. {
74. buttonFindDevices.Text = "Zatrzymaj wyszukiwanie";
75. continueScanning = true;
76. textBoxConsole.Text += ("Szukam urządzeń.\r\n");
77. }
78. else
79. {
80. buttonFindDevices.Text = "Szukaj urządzeń";
81. continueScanning = false;
82. textBoxConsole.Text += "Wyszukiwanie zakończone.\r\n";
83. }
84. }
85. // Funkcja pozwalająca na odbieranie plików
86. private void ReceiveFiles()
87. {
88. while (isPaired)
89. {
90. var listener = new ObexListener(ObexTransport.Bluetooth);
91. listener.Start();
92. ObexListenerContext ctx = listener.GetContext();
93. ObexListenerRequest req = ctx.Request;
94. String[] pathSplits = req.RawUrl.Split('/');
95. String file = pathSplits[pathSplits.Length - 1];
96. req.WriteFile(file);
97. textBoxConsole.Text += "Odebrano plik.\r\n";
98. listener.Stop();
99. }
100. }
101. //Funkcja parująca urządzenia
102. private void PairWithDevice()
103. {
104. foreach(var device in devices)
105. {
106. if (device.DeviceName == (string)listBoxDevices.SelectedItem)
107. deviceToPair = device;
108. }
109. if(deviceToPair==null)
110. {
111. textBoxConsole.Text += "Nie zaznaczyłeś z którym urządzeniem chcesz się połączyć!\r\n";
112. }
113. else
114. {
115. textBoxConsole.Text += "Łączenie z: " + (string)deviceToPair.DeviceName + "\r\n";
116. deviceToPair.Update();
117. deviceToPair.Refresh();
118. deviceToPair.SetServiceState(BluetoothService.ObexObjectPush, true);
119. isPaired = BluetoothSecurity.PairRequest(deviceToPair.DeviceAddress, DEVICE\_PIN);
120. if(isPaired)
121. {
122. textBoxConsole.Text += "Sparowano urządzenie.\r\n";
123. listBoxConnected.Items.Add(deviceToPair.DeviceName);
124. buttonPairWithDevice.Enabled = false;
125. buttonUnpair.Enabled = true;
126. Thread receiveFiles = new Thread(ReceiveFiles);
127. receiveFiles.Priority = ThreadPriority.Highest;
128. receiveFiles.IsBackground = true;
129. receiveFiles.Start();
130. }
131. else
132. {
133. textBoxConsole.Text += "Nie sparowano.\r\n";
134. }
135. }
136. }
137. //Funkcja rozparowująca urządzenia
138. private void Unpair()
139. {
140. if (deviceToPair.DeviceName == listBoxConnected.SelectedItem.ToString())
141. isUnpaired = BluetoothSecurity.RemoveDevice(deviceToPair.DeviceAddress);
143. if(isUnpaired)
144. {
145. textBoxConsole.Text += "Rozłączono.\r\n";
146. listBoxConnected.Items.Remove(deviceToPair.DeviceName);
147. isPaired = false;
148. }
149. else
150. {
151. textBoxConsole.Text += "Nie udało się rozłączyć urządzenia.\r\n";
152. }
153. }
154. private void buttonPairWithDevice\_Click(object sender, EventArgs e)
155. {
156. Thread pairWithDevice = new Thread(PairWithDevice);
157. pairWithDevice.Priority = ThreadPriority.Highest;
158. pairWithDevice.IsBackground = true;
159. pairWithDevice.Start();
160. }
161. private void buttonUnpair\_Click(object sender, EventArgs e)
162. {
163. Unpair();
164. buttonPairWithDevice.Enabled = true;
165. buttonUnpair.Enabled = false;
166. }
167. private void buttonAddFile\_Click(object sender, EventArgs e)
168. {
169. openFile.ShowDialog();
170. }
171. //Funkcja pozwalająca na wybranie pliku który chcemy wysłać
172. private void openFile\_FileOk(object sender, CancelEventArgs e)
173. {
174. bool exist = false;
175. foreach (var fileName in openFile.FileNames)
176. {
177. if (fileName != null)
178. {
179. foreach (string item in listBoxFiles.Items)
180. if (item == fileName)
181. exist = true;
182. }
183. if (!exist)
184. {
185. listBoxFiles.Items.Add(fileName);
186. }
187. else
188. {
189. MessageBox.Show("");
190. }
191. }
192. }
194. // Funkcja wysyłająca plik do sparowanego urządzenia
195. private void SendFiles()
196. {
197. foreach(string file in listBoxFiles.Items)
198. {
199. try
200. {
201. if(deviceToPair!=null)
202. textBoxConsole.Text += ("Wysyłanie pliku\r\n");
203. var uri = new Uri("obex://" + deviceToPair.DeviceAddress + "/" + file);
204. var request = new ObexWebRequest(uri);
205. request.ReadFile(file);
206. var response = (ObexWebResponse)request.GetResponse();
207. textBoxConsole.Text += ("Pomyślnie wysłano plik:" + file + "\r\n");
208. }
209. catch(Exception error)
210. {
211. textBoxConsole.Text += "Wystapił błąd! " + error.Message + "\r\n";
212. }
213. }
214. listBoxFiles.Items.Clear();
215. }
216. private void buttonSendFile\_Click(object sender, EventArgs e)
217. {
218. Thread sendFiles = new Thread(SendFiles);
219. sendFiles.Priority = ThreadPriority.Highest;
220. sendFiles.IsBackground = true;
221. sendFiles.Start();
222. }
223. private void textBoxConsole\_TextChanged(object sender, EventArgs e)
224. {
225. textBoxConsole.SelectionStart = textBoxConsole.Text.Length;
226. textBoxConsole.ScrollToCaret();
227. }
228. Uwagi

Nasz program niestety nie ma możliwości wyboru z jakiego adaptera chcemy skorzystać.