```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.Linq;
 4  using Windows.Media.SpeechRecognition;
 5  using Windows.Devices.Spi;
 6  using System.Text;
 7  using System.Threading;
 8  using System.Threading.Tasks;
 9  using Windows.Devices.Enumeration;
10
11  namespace RaspiHomeSpeechNSynthetize
12  {
13      public class Speecher
14      {
15          #region Fields
16          #region Constants
17          #endregion
18
19          #region Variable
20          private Synthetizer _rhSynt;
21          private Commands _rhCommands;
22          private CommunicationWithServer _comWithServer;
23
24          private SpeechRecognizer _recoEngine = null;
25          private SpiDevice _mcp3202;
26
27          private bool _isRaspiCalled = false;
28          #endregion
29          #endregion
30
31          #region Properties
32          public Synthetizer RhSynt
33          {
34              get
35              {
36                  return _rhSynt;
37              }
38
39              set
40              {
41                  _rhSynt = value;
42              }
43          }
44
45          public bool IsRaspiCalled
46          {
47              get
48              {
49                  return _isRaspiCalled;
50              }
51
52              set
53              {
54                  _isRaspiCalled = value;
55              }
56          }
```

```
57
58          public CommunicationWithServer ComWithServer
59          {
60              get
61              {
62                  return _comWithServer;
63              }
64
65              set
66              {
67                  _comWithServer = value;
68              }
69          }
70
71          public Commands RhCommands
72          {
73              get
74              {
75                  return _rhCommands;
76              }
77
78              set
79              {
80                  _rhCommands = value;
81              }
82          }
83          #endregion
84
85          #region Constructor
86          /// <summary>
87          /// Constructor: Initialize
88          /// </summary>
89          public Speecher()
90          {
91              // Initialize the synthetizer
92              this.RhSynt = new Synthetizer(this);
93              this.RhCommands = new Commands();
94
95              // Initialize the communication with the server
96              this.ComWithServer = new CommunicationWithServer(this);
97
98              // Create the speech recognition object
99              this._recoEngine = new SpeechRecognizer();
100
101             // Initialize the recognition
102             InitializeSpeechRecognizer();
103         }
104         #endregion
105
106         #region Methods
107         /// <summary>
108         /// Initialze the Spi communication
109         /// </summary>
110         private async void InitializeSpi()
111         {
112             //using SPI0 on the Pi
```

```
113             var spiSettings = new SpiConnectionSettings(0);//for spi bus index 0
114             spiSettings.ClockFrequency = 3600000; //3.6 MHz
115             spiSettings.Mode = SpiMode.Mode0;
116
117             string spiQuery = SpiDevice.GetDeviceSelector("SPI0");
118
119             var deviceInfo = await DeviceInformation.FindAllAsync(spiQuery);
120             if (deviceInfo != null && deviceInfo.Count > 0)
121             {
122                 _mcp3202 = await SpiDevice.FromIdAsync(deviceInfo[0].Id, spiSettings);
123             }
124         }
125
126         /// <summary>
127         /// Read digital input with SPI
128         /// </summary>
129         /// <returns></returns>
130         private string ReadSpiData()
131         {
132             byte[] transmitBuffer = new byte[3] { 0x01, 0x80, 0 };
133             byte[] receiveBuffer = new byte[3];
134
135             string result = "";
136
137             _mcp3202.TransferFullDuplex(transmitBuffer, receiveBuffer);
138
139             return result = Encoding.UTF8.GetString(receiveBuffer);
140         }
141
142         /// <summary>
143         /// (obsolete on UWP) Set the configuration of the speecher
144         /// </summary>
145         private void InitializeSpeechRecognizer()
146         {
147
148         }
149
150         /// <summary>
151         /// (obsolete on UWP) Enable the speech, used when raspi is not talking
152         /// </summary>
153         public void EnableSpeech()
154         {
155             //this._recoEngine.RecognizeAsync(RecognizeMode.Multiple);
156         }
157
158         /// <summary>
159         /// (obsolete on UWP) Disable the speech, used when raspi is talking
160         /// </summary>
161         public void DisableSpeech()
162         {
163            // this._recoEngine.RecognizeAsyncStop();
164         }
165
```

```
166         /// <summary>
167         /// Used to call raspi
168         /// </summary>
169         /// <param name="nameMentioned"></param>
170         private void CallRaspi(string nameMentioned)
171         {
172             DisableSpeech();
173 
174             this.RhSynt.RaspiCalled(nameMentioned);
175 
176             EnableSpeech();
177         }
178 
179         /// <summary>
180         /// Used to send the command after raspi called
181         /// </summary>
182         /// <param name="brutCommand"></param>
183         public void SendBrutCommand(string brutCommand)
184         {
185             DisableSpeech();
186 
187             this.ComWithServer.SendCommandToServer(brutCommand);

188         }
189 
190         /// <summary>
191         /// Reply message to the sender (reply message error if command
              unknow)
192         /// </summary>
193         /// <param name="messageReply"></param>
194         /// <param name="messageCommand"></param>
195         public void ReplyForSynthetize(string messageReply, string
              messageCommand)
196         {
197             if (messageReply == "ERROR_MESSAGE")
198             {
199                 this.RhSynt.WrongCommand();
200             }
201             else
202             {
203                 this.RhSynt.RaspiSayInformation
                      (this.RhSynt.SetProprelyInformations(messageReply,
                      messageCommand));
204             }
205             this.IsRaspiCalled = false;
206 
207             EnableSpeech();
208         }
209         #endregion
210     }
211 }
212 
```