**Авторы**: К.М. Головко, И.Р. Агамирзян

**Название программы**: Беспроводная система управления домашними устройствами на платформе Android и Arduino

**Исходный код**:

//C# код, проект IRemote

//Файл: RemotesDatabase.cs

using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using SQLite;  
using Newtonsoft.Json;  
namespace IRemote  
{  
 public class RemotesDatabase  
 {  
 readonly SQLiteAsyncConnection database;  
  
 public RemotesDatabase(string path)  
 {  
 database = new SQLiteAsyncConnection(path);  
 RemoteToSave lol = new RemoteToSave();  
 lol.Category = "LOL";  
 database.CreateTableAsync<RemoteToSave>().Wait();  
 //database.DropTableAsync<RemoteToSave>().Wait();  
 }  
 /// <summary>  
 /// Gets all the remotes from database  
 /// </summary>  
 /// <returns>The remotes list</returns>  
 public async Task<List<Remote>> GetRemotesAsync()  
 {  
 List<RemoteToSave> JSONRemotes = await database.Table<RemoteToSave>().ToListAsync();  
 List<Remote> remotes = new List<Remote>();  
 await Task.Run(() =>  
 {  
 {  
 foreach (RemoteToSave rem in JSONRemotes)  
 {  
 Remote nextRemote = JsonConvert.DeserializeObject<Remote>(rem.JSONRemote);  
 nextRemote.ID = rem.ID;  
 nextRemote.Category = rem.Category;  
 remotes.Add(nextRemote);  
 }  
 }  
 } );  
 return remotes;  
 }  
 /// <summary>  
 /// Gets the remotes with mancioned category.  
 /// </summary>  
 /// <returns>The remotes with mancioned category.</returns>  
 /// <param name="category">Category.</param>  
 public async Task<List<Remote>> GetRemotesWithCategoryAsync(string category)  
 {  
 List<RemoteToSave> JSONRemotes = await database.Table<RemoteToSave>().ToListAsync();  
 List<Remote> remotes = new List<Remote>();  
 await Task.Run(() =>  
 {  
 {  
 foreach (RemoteToSave rem in JSONRemotes)  
 {  
 if (rem.Category == category)  
 {  
 Remote nextRemote = JsonConvert.DeserializeObject<Remote>(rem.JSONRemote);  
 nextRemote.ID = rem.ID;  
 nextRemote.Category = rem.Category;  
 remotes.Add(nextRemote);  
 }  
 }  
 }  
 } );  
 return remotes;  
 }  
 /// <summary>  
 /// Gets the categories.  
 /// </summary>  
 /// <returns>The categories list</returns>  
 public async Task<List<string>> GetCategoriesAsync()  
 {  
 List<RemoteToSave> JSONRemotes = await database.Table<RemoteToSave>().ToListAsync();  
 HashSet<string> categories = new HashSet<string>();  
 List<string> categoriesList = new List<string>();  
 await Task.Run(() =>  
 {  
 foreach (RemoteToSave rem in JSONRemotes)  
 if (!String.IsNullOrWhiteSpace(rem.Category))  
 categories.Add(rem.Category);  
 foreach (string cat in categories)  
 categoriesList.Add(cat);  
 } );  
 return categoriesList;  
  
 }  
 /// <summary>  
 /// Gets the remote by identifier.  
 /// </summary>  
 /// <returns>The remote by identifier.</returns>  
 /// <param name="id">Identifier.</param>  
 public async Task<Remote> GetRemoteByIDAsync(int id)  
 {  
 RemoteToSave retrem = await database.Table<RemoteToSave>().Where(i => i.ID == id).FirstOrDefaultAsync();  
 return JsonConvert.DeserializeObject<Remote>(retrem.JSONRemote);  
  
 }  
 /// <summary>  
 /// Saves the remote async.  
 /// </summary>  
 /// <returns>int</returns>  
 /// <param name="rem">Remote</param>  
 public async Task<int> SaveRemoteAsync(Remote rem)  
 {  
 RemoteToSave save = new RemoteToSave  
 {  
 ID = rem.ID,  
 Category = rem.Category,  
 JSONRemote = JsonConvert.SerializeObject(rem)  
 };  
 if (save.ID != 0)  
 {  
 return await database.UpdateAsync(save);  
 }  
 else  
 {  
 return await database.InsertAsync(save);  
 }  
  
 }  
 /// <summary>  
 /// Removes the remote async.  
 /// </summary>  
 /// <returns>int</returns>  
 /// <param name="rem">Remote</param>  
 public async Task<int> RemoveRemoteAsync(Remote rem)  
 {  
 RemoteToSave del = new RemoteToSave  
 {  
 ID = rem.ID,  
 Category = rem.Category,  
 JSONRemote = JsonConvert.SerializeObject(rem)  
 };  
 return await database.DeleteAsync(del);  
 }  
 }  
}  
//Файл: IBlueConnection.cs

using System;  
using System.Threading.Tasks;  
using System.Collections.Generic;  
using System.Threading;  
namespace IRemote  
{  
 public interface IBlueConnection  
 {  
 List<string> BoundedDevicesNames { get; }  
 bool IsBluetoothOn { get; }  
 int DeviceToWorkSetByNumber { set; }  
 bool AnyBluetooth { get; }  
 Task<bool> ConnectToSelectedDevice();  
 bool IsConnected { get; }  
 void Disconnect();  
 string ConnectedDeviceName { get; }  
 void SendIR(IRSignal signal);  
 Task<IRSignal> ReciveIR();  
 void CalancellReciving();  
 }  
}  
//Файл: IDbFileProvider.cs

using System;  
namespace IRemote  
{  
 public interface IDbFileProvider  
 {  
 string GetLocalFilePath(string filename);  
 }  
}  
//Файл: IMakeToast.cs

using System;  
  
namespace IRemote  
{  
 /// <summary>  
 /// Interface to make user notifyes  
 /// </summary>  
 public interface IMakeToast  
 {  
  
 /// <summary>  
 /// Shows the toast message  
 /// </summary>  
 /// <param name="Message">Message</param>  
 /// <param name="IsLong">If set to <c>true</c> long showing, else short</param>  
 void ShowMessage(string Message, bool IsLong);  
  
 }  
}

//Файл: ConnectionPage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.ConnectionPage" Title="Connect to IRemote">  
 <ContentPage.ToolbarItems>  
 <ToolbarItem Text="Connect" Clicked = "OnConnectClicked"/>  
 <ToolbarItem Text="Disconnect" Clicked = "OnDisconnectClicked"/>  
 </ContentPage.ToolbarItems>  
 <ContentPage.Content>  
 <StackLayout>  
 <Label Text="Select a device:" Margin="20,10,0,20" FontSize="18">  
 </Label>  
 <ListView x:Name="Devices" Margin="0,10,0,0" SeparatorColor="#ECEFF1">  
 <ListView.ItemTemplate>  
 <DataTemplate>  
 <ViewCell>  
 <StackLayout>  
 <Label Text="{Binding}" TextColor="#37474F" HorizontalTextAlignment="Start" VerticalTextAlignment="Center" Margin="25,10,25,0" />  
 </StackLayout>  
 </ViewCell>  
 </DataTemplate>  
 </ListView.ItemTemplate>  
 </ListView>  
 <Label x:Name = "status" Margin="20,10,0,20" FontSize="18">  
 </Label>  
 </StackLayout>  
 </ContentPage.Content>  
</ContentPage>  
//Файл: : ConnectionPage.xaml.cs

using System;  
using System.Collections.Generic;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class ConnectionPage : ContentPage  
 {  
  
 public ConnectionPage()  
 {  
 InitializeComponent();  
 if (App.BlueCon.AnyBluetooth)  
 {  
 Devices.ItemsSource = App.BlueCon.BoundedDevicesNames;  
 }  
 else  
 {  
 StackLayout errorStack = new StackLayout();  
  
 errorStack.Children.Add(new Label  
 {  
 HorizontalOptions = LayoutOptions.Center,  
 VerticalOptions = LayoutOptions.Center,  
 Text = "Your device have no Bluetooth\ n\n\n\nThis app wouldn't work, sorry((",  
 FontSize = 30,  
 //TextColor = Color.FromHex("#F44336"),  
 VerticalTextAlignment = TextAlignment.Center,  
 HorizontalTextAlignment = TextAlignment.Center,  
 Margin = new Thickness(20, 50)  
 } );  
 Content = errorStack;  
 ToolbarItems.Clear();  
 }  
 selectedNumber = -1;  
 Devices.ItemSelected += OnItemSelected;  
  
 }  
  
 int selectedNumber;  
  
 protected override void OnAppearing()  
 {  
 base.OnAppearing();  
 if (App.BlueCon.AnyBluetooth)  
 {  
 if (App.BlueCon.AnyBluetooth && !App.BlueCon.IsBluetoothOn)  
 {  
 App.ToastMaker.ShowMessage("Please, turn on the bluetooth", true);  
  
 }  
 Devices.ItemsSource = App.BlueCon.BoundedDevicesNames;  
 selectedNumber = -1;  
 status.Text = App.BlueCon.ConnectedDeviceName;  
 }  
 }  
  
 protected async void OnConnectClicked(object sender, System.EventArgs e)  
 {  
 if (selectedNumber != -1)  
 {  
 App.BlueCon.DeviceToWorkSetByNumber = selectedNumber;  
 bool result = await App.BlueCon.ConnectToSelectedDevice();  
 if (result)  
 {  
 App.ToastMaker.ShowMessage($"Connected to {Devices.SelectedItem as string} ", false);  
 status.Text = $"Connected to {Devices.SelectedItem as string} ";  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage($"Can not connect to {Devices.SelectedItem as string} ", false);  
 status.Text = "Not Connected";  
 }  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("Select a device first", false);  
  
 }  
  
 }  
  
 protected void OnItemSelected(object sender, SelectedItemChangedEventArgs e)  
 {  
 int i = 0;  
 foreach (object dev in Devices.ItemsSource)  
 {  
 if (dev.Equals(Devices.SelectedItem))  
 {  
 selectedNumber = i;  
 }  
 i++;  
 }  
 }  
  
 protected void OnDisconnectClicked(object sender, EventArgs e)  
 {  
 App.BlueCon.Disconnect();  
 App.ToastMaker.ShowMessage("All connections are closed", false);  
 status.Text = "Not Connected";  
 }  
 }  
}  
//Файл: : EditPage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.EditPage">  
 <ContentPage.ToolbarItems>  
 <ToolbarItem Text = "Save" Clicked = "OnSaveClicked"/>  
 </ContentPage.ToolbarItems>  
  
 <ContentPage.Content>  
 <ScrollView>  
 <StackLayout >  
 <StackLayout Orientation="Horizontal">  
 <Label Text="Name " HorizontalOptions="Start" FontSize="Medium" Margin="15,15,5,5"/>  
 <Entry x:Name="RemoteNameEntery" HorizontalOptions="FillAndExpand" Margin="0,0,5,5" />  
 </StackLayout>  
 <StackLayout Orientation="Horizontal">  
 <Label Text="Category " HorizontalOptions="Start" FontSize="Medium" Margin="15,15,5,5"/>  
 <Entry x:Name="CategoryEntery" HorizontalOptions="FillAndExpand" Margin="0,0,5,5" />  
 <Picker x:Name="CategoryPicker" HorizontalOptions="End" Title="Select"/>  
 </StackLayout>  
 <Label Text = "Main" Margin = "10,10,10,10" FontSize = "Medium"/>  
 <Grid x:Name = "MainGrid" >  
 <Grid.RowDefinitions>  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
   
 </Grid>  
 <Label Text = "Additional" Margin = "10,10,10,10" FontSize = "Medium"/>  
 <Grid x:Name = "AdditionalGrid" >  
 <Grid.RowDefinitions>  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
   
 </Grid>  
 <Label Text = "Numbers" Margin = "10,10,10,10" FontSize = "Medium"/>  
 <Grid x:Name = "NumbersGrid">  
 <Grid.RowDefinitions>  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
   
 </Grid>  
 <StackLayout x:Name = "CustomButtonsEdit">  
   
 </StackLayout>  
 </StackLayout>  
 </ScrollView>  
 </ContentPage.Content>  
</ContentPage>  
  
<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.EditPage">  
 <ContentPage.ToolbarItems>  
 <ToolbarItem Text = "Save" Clicked = "OnSaveClicked"/>  
 </ContentPage.ToolbarItems>  
  
 <ContentPage.Content>  
 <ScrollView>  
 <StackLayout >  
 <StackLayout Orientation="Horizontal">  
 <Label Text="Name " HorizontalOptions="Start" FontSize="Medium" Margin="15,15,5,5"/>  
 <Entry x:Name="RemoteNameEntery" HorizontalOptions="FillAndExpand" Margin="0,0,5,5" />  
 </StackLayout>  
 <StackLayout Orientation="Horizontal">  
 <Label Text="Category " HorizontalOptions="Start" FontSize="Medium" Margin="15,15,5,5"/>  
 <Entry x:Name="CategoryEntery" HorizontalOptions="FillAndExpand" Margin="0,0,5,5" />  
 <Picker x:Name="CategoryPicker" HorizontalOptions="End" Title="Select"/>  
 </StackLayout>  
 <Label Text = "Main" Margin = "10,10,10,10" FontSize = "Medium"/>  
 <Grid x:Name = "MainGrid" >  
 <Grid.RowDefinitions>  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
   
 </Grid>  
 <Label Text = "Additional" Margin = "10,10,10,10" FontSize = "Medium"/>  
 <Grid x:Name = "AdditionalGrid" >  
 <Grid.RowDefinitions>  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
   
 </Grid>  
 <Label Text = "Numbers" Margin = "10,10,10,10" FontSize = "Medium"/>  
 <Grid x:Name = "NumbersGrid">  
 <Grid.RowDefinitions>  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 <RowDefinition Height="Auto" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
   
 </Grid>  
 <StackLayout x:Name = "CustomButtonsEdit">  
   
 </StackLayout>  
 </StackLayout>  
 </ScrollView>  
 </ContentPage.Content>  
</ContentPage>  
// Файл EditPage.xaml.cs

using System;  
using System.Collections.Generic;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class EditPage : ContentPage  
 {  
 Remote bindedRemote;  
 List<Button> allButtons;  
 public EditPage()  
 {  
 InitializeComponent();  
 RemoteNameEntery.TextChanged += OnRemoteNameChanged;  
 CategoryEntery.TextChanged += CategoryEntery\_TextChanged;  
 CategoryPicker.SelectedIndexChanged += CategoryPicker\_SelectedIndexChanged;  
 }  
  
 protected async override void OnAppearing()  
 {  
 base.OnAppearing();  
 bindedRemote = BindingContext as Remote;  
 Title = bindedRemote.Name;  
 RemoteNameEntery.Text = Title;  
  
 GenerateGrid(MainGrid, 0, 19);  
 GenerateGrid(AdditionalGrid, 20, 43);  
 GenerateGrid(NumbersGrid, 44, 59);  
 SetAllButtonsList();  
 GenerateCustomKeysEditor();  
 CategoryPicker.Items.Clear();  
 CategoryEntery.Text = bindedRemote.Category;  
 List<string> categories = await App.Database.GetCategoriesAsync();  
 foreach (string cat in categories)  
 {  
 CategoryPicker.Items.Add(cat);  
 }  
 }  
  
 protected void OnRemoteNameChanged(object sennder, EventArgs e)  
 {  
 if (RemoteNameEntery.Text.Length > 32)  
 {  
 RemoteNameEntery.Text = RemoteNameEntery.Text.Substring(0, 32);  
 }  
 bindedRemote.Name = RemoteNameEntery.Text;  
 Title = RemoteNameEntery.Text;  
 }  
 protected async override void OnDisappearing()  
 {  
 //await DisplayAlert("Don't forget to save changes",  
 // "Untill you press save button, changes won\'t save to" +  
 // " database and will reset after remote closing", "Got it");  
 base.OnDisappearing();  
  
 }  
 protected async void OnSaveClicked(object sender, EventArgs e)  
 {  
 await App.Database.SaveRemoteAsync(bindedRemote);  
 App.ToastMaker.ShowMessage($"Remote {bindedRemote.Name} saved", false);  
 }  
  
 void CategoryPicker\_SelectedIndexChanged(object sender, EventArgs e)  
 {  
 if (CategoryPicker.SelectedIndex >= 0)  
 {  
 CategoryEntery.Text = CategoryPicker.Items[CategoryPicker.SelectedIndex];  
 CategoryPicker.SelectedIndex = -1;  
  
 }  
 }  
  
 void CategoryEntery\_TextChanged(object sender, TextChangedEventArgs e)  
 {  
 if (CategoryEntery.Text.Length > 24)  
 {  
 CategoryEntery.Text = CategoryEntery.Text.Substring(0, 24);  
 }  
 bindedRemote.Category = CategoryEntery.Text;  
 }  
  
 private void GenerateGrid(Grid grid, int istart, int ifinish)  
 {  
 grid.Children.Clear();  
 for (int i = 0; i < ifinish - istart + 1; i++)  
 {  
 int remoteButtonNumber = istart + i;  
 Button nextButton = new Button()  
 {  
 Text = $"{bindedRemote.Buttons[remoteButtonNumber].Text} ",  
 BorderRadius = 0,  
 //Margin = new Thickness(-1, -1, -1, -1),  
 BackgroundColor = Color.FromHex("EEEEEE"),  
 BindingContext = bindedRemote.Buttons[remoteButtonNumber]  
 };  
 switch (remoteButtonNumber)  
 {  
 case 3:  
 nextButton.BackgroundColor = Color.FromHex("F44336");  
 nextButton.FontFamily = Device.OnPlatform(null, "Unicode\_IEC\_symbol.ttf#IEC-symbols-Unicode", null);  
 break;  
 case 20:  
 nextButton.BackgroundColor = Color.FromHex("F44336");  
 break;  
 case 21:  
 nextButton.BackgroundColor = Color.FromHex("4CAF50");  
 break;  
 case 22:  
 nextButton.BackgroundColor = Color.FromHex("FFEB3B");  
 break;  
 case 23:  
 nextButton.BackgroundColor = Color.FromHex("2196F3");  
 break;  
 default:  
 break;  
 }  
 nextButton.Clicked += NextButton\_Clicked;  
 grid.Children.Add(nextButton, i % 4, i / 4);  
 }  
  
 }  
 private void GenerateCustomKeysEditor()  
 {  
 CustomButtonsEdit.Children.Clear();  
 int i = 0;  
 foreach (int keyId in Remote.CustomKeys)  
 {  
 i++;  
 StackLayout next = new StackLayout { Orientation = StackOrientation.Horizontal };  
 next.Children.Add(new Label  
 {  
 Text = $"Custom Key {i} ",  
 HorizontalOptions = LayoutOptions.Start,  
 Margin = new Thickness(10, 10, 10, 10),  
 FontSize = 18  
 } );  
 next.Children.Add(new Entry  
 {  
 BindingContext = keyId,  
 HorizontalOptions = LayoutOptions.FillAndExpand,  
 Text = allButtons[keyId].Text,  
 } );  
 (next.Children[1] as Entry).TextChanged += OnCustomKeyTextChanged;  
 CustomButtonsEdit.Children.Add(next);  
 }  
 }  
 private void SetAllButtonsList()  
 {  
 allButtons = new List<Button>();  
 foreach (View child in MainGrid.Children)  
 {  
 allButtons.Add(child as Button);  
 }  
 foreach (View child in AdditionalGrid.Children)  
 {  
 allButtons.Add(child as Button);  
 }  
 foreach (View child in NumbersGrid.Children)  
 {  
 allButtons.Add(child as Button);  
 }  
 }  
 protected void OnCustomKeyTextChanged(object sender, EventArgs e)  
 {  
  
 Entry customKeyEntry = sender as Entry;  
 int customKeyId = (int)customKeyEntry.BindingContext;  
 if (customKeyEntry.Text.Length > 16)  
 {  
 customKeyEntry.Text = customKeyEntry.Text.Substring(0, 16);  
 }  
 allButtons[customKeyId].Text = customKeyEntry.Text;  
 bindedRemote.Buttons[customKeyId].Text = customKeyEntry.Text;  
  
 }  
  
 void NextButton\_Clicked(object sender, EventArgs e)  
 {  
 if (App.BlueCon.AnyBluetooth)  
 {  
 if (App.BlueCon.IsBluetoothOn)  
 {  
 if (App.BlueCon.IsConnected)  
 {  
 Navigation.PushAsync(new RecivePage { BindingContext = (sender as Button).BindingContext as RemoteButton } );  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("You are not connected", false);  
 }  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("Turn on the Bluetooth", false);  
 }  
  
  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("There is no Bluetooth on your device", false);  
 }  
 }  
 }  
}  
// Файл HelpPage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.HelpPage">  
 <ContentPage.Content>  
 <ScrollView>  
 <StackLayout>  
 <Label x:Name="about" FontSize="Medium" HorizontalTextAlignment="Center" VerticalTextAlignment="Center" Margin="30,0,30,0" LineBreakMode="WordWrap"/>  
   
 </StackLayout>  
 </ScrollView>  
 </ContentPage.Content>  
</ContentPage>  
// Файл HelpPage.xaml.cs

using System;  
using System.Collections.Generic;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class HelpPage : ContentPage  
 {  
 public HelpPage()  
 {  
 InitializeComponent();  
 about.Text = "Thank you for using IRemote app. \ n\n If you want to build the receive-sending device by yourself," +  
 " please, contact kirillgolowko@gmail.com for more details and the protocol description. \ n\n" +  
 "To connect to your devise use connection manager. You should have a Bluetooth turned on on your device." +  
 " Then you could use the app. \ n\nTo use a virtual remote just press a button.There are three tabs with " +  
 "different keys.There is an Edit tab to edit your remote.Here you can choose name and category for your " +  
 "remote, also some captures on buttons may be changed. To record IR signal press a button on virtual " +  
 "remote in edit mode and then press a button on your real remote.App would record the signal automaticly. " +  
 "To cancel recording just press the back button. \ n\nThere is a + for adding new remote and space under" +  
 " Show all remotes for categories. If app works unproper restart it.\ n\n This is CC app, use it for your own needs for free. " +  
 "\ n\nBy Kirill Golovko, 2017.";  
 Title = "Help and About";  
 }  
 }  
}  
// Файл HomePage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.HomePage">  
 <ContentPage.Content>  
 <StackLayout>  
 <Label Text="HI!" FontSize="72" HorizontalTextAlignment="Center" Margin="0,30,0,30">  
 </Label>  
 <Label x:Name="aboutLabel" FontSize="Medium" HorizontalTextAlignment="Center" VerticalTextAlignment="Center" Margin="30,0,30,0" LineBreakMode="WordWrap" />  
 </StackLayout>  
 </ContentPage.Content>  
</ContentPage>

// Файл HomePage.xaml.cs  
  
using System;  
using System.Collections.Generic;  
  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class HomePage : ContentPage  
 {  
 string a { get; set; }  
 public HomePage()  
 {  
 a = "123";  
  
  
  
 InitializeComponent();  
  
 aboutLabel.Text = "This app will prowide you full-functional remote for using with ArduinoIR. You" +  
 " can easily manage your remotes an devide them by zones where you would like to use them. To find more " +  
 "abaut bluetooth protocol and other additional informaton use HELP AND ABOUT BUTTON in left menu. \u23FB";  
  
 aboutLabel.FontFamily = Device.OnPlatform(null, "Unicode\_IEC\_symbol.ttf#IEC-symbols-Unicode", null);  
  
 }  
 }  
}

// Файл MainPage.cs

using System;  
using System.Diagnostics;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public class MainPage : MasterDetailPage  
 {  
 MasterPage master;  
  
 public MainPage()  
 {  
 Detail = new NavigationPage(new HomePage { BindingContext = null } );  
 master = new MasterPage();  
 Master = master;  
 master.ConnectButtonClicked += OnConnectButtonClicked;  
 master.ShowAllButtonClicked += OnShowAllButtonClicked;  
 IsPresentedChanged += MainPage\_IsPresentedChanged;  
 master.CategorySelected += OnCategorySelected;  
 master.HelpButtonClicked += OnHelpButtonClicked;  
 }  
  
 void MainPage\_IsPresentedChanged(object sender, EventArgs e)  
 {  
 master.UpdateCategories();  
 }  
  
 public void OnConnectButtonClicked(object sender, EventArgs e)  
 {  
 Detail = new NavigationPage(new ConnectionPage());  
 IsPresented = false;  
 }  
 public void OnShowAllButtonClicked(object sender, EventArgs e)  
 {  
 Detail = new NavigationPage(new SelectionPage { BindingContext = null } );  
 IsPresented = false;  
 }  
  
 void OnCategorySelected(object sender, CategorySelectedEventArgs e)  
 {  
 Detail = new NavigationPage(new SelectionPage { BindingContext = e.Category } );  
 IsPresented = false;  
 }  
 protected override bool OnBackButtonPressed()  
 {  
 return true;  
 }  
 void OnHelpButtonClicked(object sender, EventArgs e)  
 {  
 Detail = new NavigationPage(new HelpPage());  
 IsPresented = false;  
 }  
 }  
}  
// Файл MasterPage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.MasterPage">  
 <ContentPage.Content>  
 <StackLayout>  
 <ContentView x:Name="LogoView">  
 <Label x:Name="LogoLabel" Margin="20,40,0,10" TextColor="White" FontSize="Large">IRemote</Label>  
 </ContentView>  
 <Button x:Name="ConnectButton" Text="Connect manager" BackgroundColor="White" TextColor="#546E7A" />  
 <Button x:Name="HelpButton" Text="Help and about" BackgroundColor="White" TextColor="#546E7A" />  
 <Button x:Name="ShowAllButton" Text="Show all remotes" BackgroundColor="White" TextColor="#546E7A" />  
 <ListView x:Name="CategoryList" SeparatorColor="#ECEFF1">  
 <ListView.ItemTemplate>  
 <DataTemplate>  
 <ViewCell>  
 <StackLayout>  
 <Label Text="{Binding}" Margin="10,5,10,5" TextColor="#546E7A">  
 </Label>  
 </StackLayout>  
 </ViewCell>  
 </DataTemplate>  
 </ListView.ItemTemplate>  
 </ListView>  
 </StackLayout>  
 </ContentPage.Content>  
</ContentPage>  
  
// Файл MasterPage.xaml.cs  
  
using System;  
using System.Collections.Generic;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public delegate void CategorySelectedEventHandler(object sender, CategorySelectedEventArgs e);  
  
 public partial class MasterPage : ContentPage  
 {  
 public MasterPage()  
 {  
 InitializeComponent();  
  
 Title = "IRemote";  
  
 LogoLabel.BackgroundColor = (Color)App.Current.Resources["primaryColor"];  
 LogoView.BackgroundColor = (Color)App.Current.Resources["primaryColor"];  
  
 ConnectButton.Clicked += OnNavigation;  
 ShowAllButton.Clicked += OnNavigation;  
 CategoryList.ItemTapped += OnNavigation;  
 HelpButton.Clicked += OnNavigation;  
  
 }  
  
 protected async override void OnAppearing()  
 {  
 base.OnAppearing();  
  
 CategoryList.ItemsSource = await App.Database.GetCategoriesAsync();  
 }  
  
 public event CategorySelectedEventHandler CategorySelected;  
 public event EventHandler HelpButtonClicked;  
 public event EventHandler ConnectButtonClicked;  
 public event EventHandler ShowAllButtonClicked;  
  
 public void OnNavigation(object sender, EventArgs e)  
 {  
 if (sender == ConnectButton && ConnectButtonClicked != null)  
 ConnectButtonClicked(this, EventArgs.Empty);  
  
 if (sender == HelpButton && HelpButtonClicked != null)  
 HelpButtonClicked(this, EventArgs.Empty);  
  
 if (sender == ShowAllButton && ShowAllButtonClicked != null)  
 ShowAllButtonClicked(this, EventArgs.Empty);  
 if (sender == CategoryList && CategorySelected != null)  
 {  
 CategorySelected(this, new CategorySelectedEventArgs((e as ItemTappedEventArgs).Item as string));  
 }  
  
 }  
 public async void UpdateCategories()  
 {  
 CategoryList.ItemsSource = await App.Database.GetCategoriesAsync();  
 }  
 }  
  
 public class CategorySelectedEventArgs : EventArgs  
 {  
 public CategorySelectedEventArgs(string category) : base()  
 {  
 Category = category;  
 }  
 public string Category { get; }  
 }  
  
}  
// Файл RecivePage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.RecivePage">  
 <ContentPage.Content>  
 <StackLayout Orientation = "Vertical">  
 <Label Text="Press a button on your real remote. If app not working properly while reciving, please, restart it" Margin = "50" HorizontalTextAlignment = "Center" VerticalTextAlignment = "Center" FontSize="20"/>   
 <Button Text="Calancell reciving" Clicked="Handle\_Clicked" />  
 </StackLayout>  
 </ContentPage.Content>  
</ContentPage>  
  
  
// Файл RecivePage.xaml.cs  
using System;  
using System.Collections.Generic;  
using System.Threading.Tasks;  
using System.Threading;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class RecivePage : ContentPage  
 {  
  
 public RecivePage()  
 {  
 InitializeComponent();  
  
 }  
 protected async override void OnAppearing()  
 {  
  
 IRSignal sig = await App.BlueCon.ReciveIR();  
 if (sig.Ok)  
 {  
 (BindingContext as RemoteButton).Signal = sig;  
 App.ToastMaker.ShowMessage($"Recorded {sig} ", false);  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("An error accured, try again", false);  
 }  
 if (IsVisible && IsEnabled)  
 {  
 try  
 {  
 await Navigation.PopAsync();  
 }  
 catch  
 {  
 App.ToastMaker.ShowMessage("Please, restart the app", false);  
 }  
 }  
  
 }  
 protected override bool OnBackButtonPressed()  
 {  
 App.BlueCon.CalancellReciving();  
  
 //bool res = base.OnBackButtonPressed();  
 return true;  
 }  
  
 void Handle\_Clicked(object sender, System.EventArgs e)  
 {  
 App.BlueCon.CalancellReciving();  
 }  
  
 }  
}  
// Файл RemotePage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.RemotePage">  
 <ContentPage.ToolbarItems>  
 <ToolbarItem Text="Delete" Clicked="OnDeleteClicked" />  
 <ToolbarItem Text="Edit" Clicked="OnEditClicked"/>  
 </ContentPage.ToolbarItems>  
 <ContentPage.Content>  
 <Grid x:Name="ButtonsGrid" ColumnSpacing="1" RowSpacing="-2">  
 <Grid.RowDefinitions>  
 <RowDefinition Height="\*" />  
 <RowDefinition Height="\*" />  
 <RowDefinition Height="\*" />  
 <RowDefinition Height="\*" />  
 <RowDefinition Height="\*" />  
 <RowDefinition Height="\*" />  
 </Grid.RowDefinitions>  
 <Grid.ColumnDefinitions>  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 <ColumnDefinition Width="\*" />  
 </Grid.ColumnDefinitions>  
 </Grid>  
 </ContentPage.Content>  
</ContentPage>  
// Файл RemotePage.xaml.cs

using System;  
using System.Collections.Generic;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class RemotePage : ContentPage  
 {  
 protected Remote bindedRemote;  
  
 public int IStart { get; set; }  
  
 public int IFinish { get; set; }  
  
  
  
 public RemotePage()  
 {  
 InitializeComponent();  
  
  
 }  
 protected override void OnAppearing()  
 {  
 base.OnAppearing();  
 bindedRemote = BindingContext as Remote;  
 ButtonsGrid.Children.Clear();  
 for (int i = 0; i < IFinish - IStart + 1; i++)  
 {  
 int remoteButtonNumber = IStart + i;  
 Button nextButton = new Button()  
 {  
 Text = $"{bindedRemote.Buttons[remoteButtonNumber].Text} ",  
 BorderRadius = 0,  
 //Margin = new Thickness(-1, -1, -1, -1),  
 BackgroundColor = Color.FromHex("EEEEEE"),  
 BindingContext = bindedRemote.Buttons[remoteButtonNumber]  
 };  
 switch (remoteButtonNumber)  
 {  
 case 3:  
 nextButton.BackgroundColor = Color.FromHex("F44336");  
 nextButton.FontFamily = Device.OnPlatform(null, "Unicode\_IEC\_symbol.ttf#IEC-symbols-Unicode", null);  
 break;  
 case 20:  
 nextButton.BackgroundColor = Color.FromHex("F44336");  
 break;  
 case 21:  
 nextButton.BackgroundColor = Color.FromHex("4CAF50");  
 break;  
 case 22:  
 nextButton.BackgroundColor = Color.FromHex("FFEB3B");  
 break;  
 case 23:  
 nextButton.BackgroundColor = Color.FromHex("2196F3");  
 break;  
 default:  
 break;  
 }  
 nextButton.Clicked += NextButton\_Clicked;  
 ButtonsGrid.Children.Add(nextButton, i % 4, i / 4);  
 }  
  
 }  
  
  
  
 protected async void OnDeleteClicked(object sender, EventArgs e)  
 {  
 var confirm = await DisplayAlert("Delete the remote?", "Are you shure to delete" +  
 $"{(BindingContext as Remote).Name} remote?", "Delete", "Clancel");  
 if (confirm)  
 {  
 await App.Database.RemoveRemoteAsync(BindingContext as Remote);  
 await Navigation.PopAsync();  
 }  
 }  
 protected async void OnEditClicked(object sender, EventArgs e)  
 {  
 await Navigation.PushAsync(new EditPage { BindingContext = this.BindingContext } );  
 }  
  
 void NextButton\_Clicked(object sender, EventArgs e)  
 {  
 if (App.BlueCon.AnyBluetooth)  
 {  
 if (App.BlueCon.IsBluetoothOn)  
 {  
 if (App.BlueCon.IsConnected)  
 {  
 App.BlueCon.SendIR(((sender as Button).BindingContext as RemoteButton).Signal);  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("You are not connected", false);  
 }  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("Turn on the Bluetooth", false);  
 }  
  
  
 }  
 else  
 {  
 App.ToastMaker.ShowMessage("There is no Bluetooth on your device", false);  
 }  
  
 }  
 }  
}  
  
// Файл SelectionPage.xaml

<?xml version="1.0" encoding="UTF-8"?>  
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.SelectionPage">  
 <ContentPage.ToolbarItems>  
 <ToolbarItem Text="+" Clicked="AddNewClicked" />  
 </ContentPage.ToolbarItems>  
 <ContentPage.Content>  
 <StackLayout>  
 <ListView x:Name="RemotesList" Margin="0,20,0,0" SeparatorColor="#ECEFF1">  
 <ListView.ItemTemplate>  
 <DataTemplate>  
 <ViewCell>  
 <StackLayout>  
 <Label Text="{Binding Name}" TextColor="#37474F" HorizontalTextAlignment="Start" VerticalTextAlignment="Center" Margin="25,10,0,25" FontSize="18" />  
 </StackLayout>  
 </ViewCell>  
 </DataTemplate>  
 </ListView.ItemTemplate>  
 </ListView>  
 </StackLayout>  
 </ContentPage.Content>  
</ContentPage>  
  
// Файл SelectionPage.xaml.cs  
using System;  
using System.Collections.Generic;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public partial class SelectionPage : ContentPage  
 {  
  
 public SelectionPage()  
 {  
 InitializeComponent();  
  
 RemotesList.ItemSelected += OnRemoteSelected;  
  
  
 }  
 protected override async void OnAppearing()  
 {  
 base.OnAppearing();  
 if (BindingContext == null)  
 {  
 Title = "All Remotes";  
 RemotesList.ItemsSource = await App.Database.GetRemotesAsync();  
 }  
 else  
 {  
 Title = BindingContext as string;  
 RemotesList.ItemsSource = await App.Database.GetRemotesWithCategoryAsync(BindingContext as string);  
 }  
  
 }  
  
  
 protected async void AddNewClicked(object sender, EventArgs e)  
 {  
 Remote newRemote = new Remote(true)  
 {  
 Name = "New Remote",  
 Category = BindingContext == null ? "" : BindingContext as string  
 };  
  
 await App.Database.SaveRemoteAsync(newRemote);  
 List<Remote> remotes = await App.Database.GetRemotesAsync();  
 await Navigation.PushAsync(new TabbedRemote { BindingContext = remotes[remotes.Count - 1] } );  
 }  
  
  
 protected async void OnRemoteSelected(object sender, SelectedItemChangedEventArgs e)  
 {  
  
 await Navigation.PushAsync(new TabbedRemote { BindingContext = e.SelectedItem as Remote } );  
  
 }  
  
  
 }  
}  
// Файл TabbedRemote.cs

using System;  
  
using Xamarin.Forms;  
  
namespace IRemote  
{  
 public class TabbedRemote : TabbedPage  
 {  
 public TabbedRemote()  
 {  
  
 }  
 protected override void OnAppearing()  
 {  
 base.OnAppearing();  
 Title = (BindingContext as Remote).Name;  
 Children.Clear();  
 Children.Add(new RemotePage { BindingContext = this.BindingContext, IStart = 0, IFinish = 19, Title = "Main" } );  
 Children.Add(new RemotePage { BindingContext = this.BindingContext, IStart = 20, IFinish = 43, Title = "Additonal" } );  
 Children.Add(new RemotePage { BindingContext = this.BindingContext, IStart = 44, IFinish = 59, Title = "Numbers" } );  
 }  
 }  
}  
// Файл DefaultRemote.cs

using System;  
namespace IRemote  
{  
 public static class DefaultRemote  
 {  
 public static string[] DefaultButtonNames  
 {  
 get  
 {  
 return new string[60]  
 {  
 "?", "Source", "Mute","\u23FB",  
 "CH+", "CH-", "VOL+", "VOL-",  
 "GUIDE", "MENU", "\u25B2", "\u232B",  
 "SMART", "\u25C0", "\u23CE", "\u25B6",  
 "EXIT", "FRAME", "\u25BC", "HOME",  
  
 "", "", "", "", // row with RGYB buttons  
 "C1", "C2", "C3", "C4",  
 "C5", "C6", "C7", "C8",  
 "|\u25C0\u25C0", "\u25C0\u25C0", "\u25B6\u25B6", "\u25B6\u25B6|",  
 "\u25CF", "\u25B6", "||", "\u25A0",  
 "BASS+", "BASS-", "TRE+", "TRE-",  
  
 "1", "2", "3", "C9",  
 "4", "5", "6", "C10",  
 "7", "8", "9", "C11",  
 "C12", "0", "C13", "C14"  
 };  
 }  
 }  
 }  
 public partial class Remote  
 {  
 public Remote()  
 {  
  
 }  
 public Remote(bool IsNew)  
 {  
 if (IsNew)  
 {  
 Buttons = new System.Collections.Generic.List<RemoteButton>();  
 for (int i = 0; i < 60; i++)  
 {  
 Buttons.Add(new RemoteButton(i, DefaultRemote.DefaultButtonNames[i]));  
 }  
 ID = 0;  
 Name = "New Remote";  
 }  
 }  
  
 public static int[] CustomKeys  
 {  
 get  
 {  
 return new int[] { 24, 25, 26, 27, 28, 29, 30, 31, 47, 51, 55, 56, 58, 59 };  
 }  
 }  
 }  
}

// Файл IRSignal.cs

using System;  
namespace IRemote  
{  
 public class IRSignal  
 {  
 public IRSignal() : this(0, 32, 3, 0) { }  
  
 public IRSignal(uint code, int type) : this(code, 32, type, 0) { }  
  
 public IRSignal(uint code, int len, int type) : this(code, len, type, 0) { }  
  
 public IRSignal(uint code, int len, int type, int adress)  
 {  
 Code = code;  
 Length = len;  
 ArduinoIRemoteType = type;  
 Adress = adress;  
 }  
  
 public uint Code { get; set; }  
  
 public int Length { get; set; }  
  
 public int Adress { get; set; }  
  
 public int ArduinoIRemoteType { get; set; }  
  
 public override string ToString()  
 {  
 return $"{Length.ToString("D3")}@{Code.ToString("D10")}" +  
 $"@{ArduinoIRemoteType.ToString("D2")}@{Adress.ToString("D5")}";  
 }  
 public bool Ok { get; set; }  
 }  
}  
// Файл Remote.cs

using System;  
using System.Collections.Generic;  
namespace IRemote  
{  
 public partial class Remote  
 {  
  
 public string Name { get; set; }  
 public string Category { get; set; }  
 public int ID { get; set; }  
 public List<RemoteButton> Buttons;  
  
 }  
}  
// Файл RemoteButton.cs

using System;  
namespace IRemote  
{  
 /// <summary>  
 /// Remote button class  
 /// </summary>  
 public class RemoteButton  
 {  
 /// <summary>  
 /// Initializes a new instance of the <see cref="T:IRemote.RemoteButton"/> class.  
 /// </summary>  
 public RemoteButton() : this(0, "Unnamed", new IRSignal()) { }  
 /// <summary>  
 /// Initializes a new instance of the <see cref="T:IRemote.RemoteButton"/> class.  
 /// </summary>  
 /// <param name="id">Identifier</param>  
 public RemoteButton(int id) : this(id, "Unnamed", new IRSignal()) { }  
 /// <summary>  
 /// Initializes a new instance of the <see cref="T:IRemote.RemoteButton"/> class.  
 /// </summary>  
 /// <param name="id">Identifier.</param>  
 /// <param name="code">Code.</param>  
 /// <param name="text">Text.</param>  
 public RemoteButton(int id, string text) : this(id, text, new IRSignal()) { }  
  
 public RemoteButton(int id, string text, IRSignal signal)  
 {  
  
 Text = text;  
 ID = id;  
 Signal = signal;  
 }  
  
 /// <summary>  
 /// Gets or sets the text of button  
 /// </summary>  
 /// <value>The text</value>  
 public string Text { get; set; }  
 /// <summary>  
 /// Gets or sets the identifier of button  
 /// </summary>  
 /// <value>The identifier.</value>  
 public int ID { get; set; }  
 public IRSignal Signal { get; set; }  
  
  
 }  
}

// Файл RemoteToSave.cs

using System;  
using SQLite;  
  
  
namespace IRemote  
{  
  
 /// <summary>  
 /// Remote to save in SQLite db  
 /// </summary>  
  
 public class RemoteToSave  
 {  
 [PrimaryKey, AutoIncrement]  
 /// <summary>  
 /// Gets or sets the identifier of Remote  
 /// </summary>  
 /// <value>The identifier.</value>  
 public int ID { get; set; }  
 /// <summary>  
 /// Gets or sets the Category of Remote  
 /// </summary>  
 /// <value>The Category</value>  
 public string Category { get; set; }  
 /// <summary>  
 /// Gets or sets the string with JSON Relation of Remote object  
 /// </summary>  
 /// <value>The keys.</value>  
 public string JSONRemote { get; set; }  
  
 }  
}

// Файл App.xaml

<?xml version="1.0" encoding="utf-8"?>  
<Application xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="IRemote.App">  
 <Application.Resources>  
 <!-- Application resource dictionary -->  
 </Application.Resources>  
</Application>  
  
// Файл App.xaml.cs  
using Xamarin.Forms;  
  
namespace IRemote  
{  
  
 public partial class App : Application  
 {  
 static IBlueConnection \_con;  
  
 static RemotesDatabase \_database;  
  
 static IMakeToast \_toastMaker;  
 public App()  
 {  
 InitializeComponent();  
 //Задаем ресурсы с цветами приложения, чтобы потом не путать где какой цвет  
 Resources = new ResourceDictionary();  
 Resources.Add("primaryDarkColor", Color.FromHex("01579B"));  
 Resources.Add("primaryColor", Color.FromHex("03A9F4"));  
 Resources.Add("backgroundColor", Color.FromHex("ECEFF1"));  
  
 //  
 //Создаем страницу навигации из главной страницы и указываем ей необходимые цвета  
  
 var startfrom = new MainPage();  
  
  
  
  
 // передаем управление на главную страницу  
 MainPage = startfrom;  
  
 }  
  
 public static IBlueConnection BlueCon  
 {  
 get  
 {  
 if (\_con == null)  
 {  
 \_con = DependencyService.Get<IBlueConnection>();  
 }  
 return \_con;  
 }  
 }  
 public static IMakeToast ToastMaker  
 {  
 get  
 {  
 if (\_toastMaker == null)  
 {  
 \_toastMaker = DependencyService.Get<IMakeToast>();  
 }  
 return \_toastMaker;  
 }  
 }  
  
 public static RemotesDatabase Database  
 {  
 get  
 {  
 if (\_database == null)  
 {  
 \_database = new RemotesDatabase(DependencyService.Get<IDbFileProvider>().GetLocalFilePath("RemotesDatabase.db3"));  
 }  
 return \_database;  
 }  
  
 }  
  
  
 protected override void OnStart()  
 {  
 // Handle when your app starts  
 }  
  
 protected override void OnSleep()  
 {  
 // Handle when your app sleeps  
 }  
  
 protected override void OnResume()  
 {  
 // Handle when your app resumes  
 }  
 }  
}  
  
//C# код, проект IRemote.Droid

// Файл BlueConnection.cs

using System;  
using System.Threading.Tasks;  
using IRemote.Droid;  
using Xamarin.Forms;  
using System.Collections.Generic;  
using Android.Bluetooth;  
using IRemote;  
using System.Threading;  
  
[assembly: Dependency(typeof(BlueConnection))]  
namespace IRemote.Droid  
{  
  
 public class BlueConnection : IBlueConnection  
 {  
 private BluetoothAdapter adapter;  
 private MakeToast toster;  
  
 private BluetoothDevice device;  
 private List<BluetoothDevice> bondedDevices;  
  
 public void SendIR(IRSignal signal)  
 {  
 try  
 {  
 char[] charsToSend = ("S" + signal.ToString()).ToCharArray();  
 byte[] toSend = new byte[charsToSend.Length];  
 for (int i = 0; i < charsToSend.Length; i++)  
 {  
 toSend[i] = Convert.ToByte(charsToSend[i]);  
 }  
 socket.OutputStream.Write(toSend, 0, toSend.Length);  
 }  
 catch (Exception)  
 {  
 toster.ShowMessage("Failed to send, check your connction", true);  
 }  
 }  
  
 public bool AnyBluetooth  
 {  
 get;  
 }  
 private BluetoothSocket socket;  
  
 public BlueConnection()  
 {  
 toster = new MakeToast();  
 adapter = BluetoothAdapter.DefaultAdapter;  
 if (adapter == null)  
 {  
 toster.ShowMessage("You haven't bluetooth on your device", true);  
 AnyBluetooth = false;  
 }  
 else  
 {  
 AnyBluetooth = true;  
 }  
  
 }  
  
 public List<string> BoundedDevicesNames  
 {  
 get  
 {  
 List<string> devNames = new List<string>();  
 bondedDevices = new List<BluetoothDevice>(adapter.BondedDevices);  
  
 foreach (BluetoothDevice dev in bondedDevices)  
 {  
 devNames.Add(dev.Name);  
 }  
 return devNames;  
 }  
 }  
 public bool IsBluetoothOn  
 {  
 get { return adapter.IsEnabled; }  
 }  
 public int DeviceToWorkSetByNumber  
 {  
 set { device = bondedDevices[value]; }  
 }  
 async public Task<bool> ConnectToSelectedDevice()  
 {  
 bool result = await Task.Run(() =>  
 {  
 try  
 {  
 if (socket != null)  
 {  
 socket.Close();  
 socket = null;  
 }  
 adapter.StartDiscovery();  
 device.SetPairingConfirmation(false);  
 device.SetPairingConfirmation(true);  
 device.CreateBond();  
 adapter.CancelDiscovery();  
 socket = device.CreateRfcommSocketToServiceRecord(Java.Util.UUID.FromString(  
 "00001101-0000-1000-8000-00805f9b34fb"));  
 socket.Connect();  
  
 }  
 catch (Exception) { return false; }  
 return true;  
 } );  
 return result;  
 }  
 public void Disconnect()  
 {  
 if (socket != null)  
 {  
 socket.Close();  
 socket = null;  
 }  
 }  
 public bool IsConnected  
 {  
 get  
 {  
 if (socket != null && socket.IsConnected) { return true; }  
 return false;  
 }  
  
 }  
 public string ConnectedDeviceName  
 {  
 get  
 {  
 if (socket != null && socket.IsConnected) return $"Connected to {device.Name} ";  
 else return "Not connected";  
 }  
 }  
 public async Task<IRSignal> ReciveIR()  
 {  
  
 IRSignal output = new IRSignal();  
 output.Ok = false;  
  
  
 try  
 {  
 socket.OutputStream.WriteByte(Convert.ToByte('R'));  
 byte[] buffer = new byte[23];  
 socket.InputStream.Flush();  
  
 int readen = 0;  
 while (readen < 23)  
 {  
 await socket.InputStream.ReadAsync(buffer, readen, 1);  
 readen++;  
  
 }  
 char[] charedBuffer = new char[buffer.Length];  
 for (int i = 0; i < buffer.Length; i++)  
 {  
 charedBuffer[i] = (char)buffer[i];  
 }  
 string recived = new string(charedBuffer);  
 if (recived[0] == 'T')  
 {  
 (new MakeToast()).ShowMessage("Timeout", true);  
 }  
 string[] splited = recived.Split('@');  
 output.Length = int.Parse(splited[0]);  
 output.Code = uint.Parse(splited[1]);  
 output.ArduinoIRemoteType = int.Parse(splited[2]);  
 output.Adress = int.Parse(splited[3]);  
 output.Ok = true;  
 }  
 catch (Exception a)  
 {  
  
 }  
 socket.InputStream.Flush();  
 return output;  
 }  
 public void CalancellReciving()  
 {  
 if (AnyBluetooth && IsBluetoothOn && IsConnected)  
 {  
 try  
 {  
 byte[] toSend = new byte[] { (byte)'C' };  
 socket.OutputStream.Write(toSend, 0, 1);  
 }  
 catch  
 {  
 toster.ShowMessage("Please, restart the app", true);  
 }  
 }  
 }  
 }  
}  
// Файл DbFileProvider.cs

using System;  
using System.IO;  
using Xamarin.Forms;  
using IRemote;  
using IRemote.Droid;  
  
[assembly: Dependency(typeof(DbFileProvider))]  
namespace IRemote.Droid  
{  
  
 public class DbFileProvider : IDbFileProvider  
 {  
 public string GetLocalFilePath(string filename)  
 {  
 string path = Environment.GetFolderPath(Environment.SpecialFolder.Personal);  
 return Path.Combine(path, filename);  
 }  
 }  
}  
// Файл MainActivity.cs

using System;  
  
using Android.App;  
using Android.Content;  
using Android.Content.PM;  
using Android.Runtime;  
using Android.Views;  
using Android.Widget;  
using Android.OS;  
  
namespace IRemote.Droid  
{  
 [Activity(Label = "IRemote.Droid", Icon = "@drawable/icon", Theme = "@style/MyTheme", MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize | ConfigChanges.Orientation)]  
 public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity  
 {  
 protected override void OnCreate(Bundle bundle)  
 {  
 TabLayoutResource = Resource.Layout.Tabbar;  
 ToolbarResource = Resource.Layout.Toolbar;  
  
 base.OnCreate(bundle);  
  
 global::Xamarin.Forms.Forms.Init(this, bundle);  
 \_mainContext = this;  
  
  
 LoadApplication(new App());  
  
 }  
 /// <summary>  
 /// private field to save Android context for my Activity  
 /// </summary>  
 static Context \_mainContext;  
 /// <summary>  
 /// Returns context to native methods which needs it  
 /// </summary>  
 /// <value>The main context</value>  
 public static Context MainContext { get { return \_mainContext; } }  
  
 }  
}

// Файл MakeToast.cs

using System;  
using Xamarin.Forms;  
using Android.App;  
using Android.Widget;  
using IRemote.Droid;  
[assembly: Dependency(typeof(MakeToast))]  
namespace IRemote.Droid  
{  
  
 public class MakeToast : IMakeToast  
 {  
  
 public void ShowMessage(string Message, bool IsLong)  
 {  
 ToastLength len = IsLong ? ToastLength.Long : ToastLength.Short;  
 Toast toShow = Toast.MakeText(MainActivity.MainContext, Message, len);  
 toShow.Show();  
 }  
 }  
}

//C++ код, проект CW.ino (код для микроконтроллера)

// Файл CW.ino

#include <IRremoteInt.h>

#include <IRremote.h>

#include "Trigger.h"

TriggerClass trigger;

int RecivePin = 11;

IRrecv reciver(RecivePin);

IRsend sender;

decode\_results result;

String zerrofill(uint32\_t number, int len);

void setup()

{

Serial.begin(9600);

reciver.enableIRIn();

}

void loop()

{

if (Serial.available())

{

byte com = Serial.read();

if (com == 'S' || com == 's')

{

int len = Serial.parseInt();

if (Serial.read() == '@') {

uint32\_t code = Serial.parseInt();

if (Serial.read() == '@')

{

int type = Serial.parseInt();

unsigned int buff[] = { (unsigned)code };

sender.sendRaw(buff, 32, 38000);

Serial.read();

uint16\_t adress = Serial.parseInt();

switch (type)

{

case UNKNOWN:

break;

case AIWA\_RC\_T501:

sender.sendAiwaRCT501(code);

break;

case DENON:

sender.sendDenon(code, len);

break;

case DISH:

sender.sendDISH(code, len);

break;

case JVC:

sender.sendJVC(code, len, false);

break;

case LEGO\_PF:

sender.sendLegoPowerFunctions(code);

break;

case LG:

sender.sendLG(code, len);

break;

case NEC:

sender.sendNEC(code, len);

break;

case PANASONIC:

sender.sendPanasonic(adress, code);

break;

case RC5:

sender.sendRC5(code, len);

break;

case RC6:

sender.sendRC6(code, len);

break;

case SAMSUNG:

sender.sendSAMSUNG(code, len);

break;

case SHARP:

sender.sendSharp(adress, code);

break;

case SONY:

sender.sendSony(code, len);

break;

case WHYNTER:

sender.sendWhynter(code, len);

break;

default:

sender.sendNEC(code, 32);

break;

}

}

}

}

if (com == 'C' || com == 'c')

{

trigger.setDisabled();

delay(30);

Serial.print("EEE@EEEEEEEEEE@EE@EEEEE");

}

if (com == 'R' || com == 'r')

{

trigger.setEnabled();

reciver.resume();

}

}

if(trigger.isEnabled() && (bool)reciver.decode(&result))

{

Serial.print(zerrofill(result.bits, 3));

Serial.print('@');

Serial.print(zerrofill(result.value, 10));

Serial.print('@');

Serial.print(zerrofill(result.decode\_type, 2));

Serial.print('@');

Serial.print(zerrofill(result.address, 5));

trigger.setDisabled();

reciver.resume();

}

trigger.check();

}

String zerrofill(uint32\_t number, int len)

{

String res = String(number);

while (res.length() < len)

{

res = "0" + res;

}

return(res);

}

// Файл Trigger.cpp

//

//

//

#include "Trigger.h"

void TriggerClass::init()

{

}

void TriggerClass::check()

{

if(\_enabled && millis() - \_setAt > 60000)

{

Serial.println("TTT@TTTTTTTTTT@TT@TTTTT");

\_enabled = false;

}

}

void TriggerClass::setEnabled()

{

if(\_enabled)

{

\_enabled = false;

Serial.println("TTT@TTTTTTTTTT@TT@TTTTT");

return;

}

\_enabled = true;

\_setAt = millis();

}

void TriggerClass::setDisabled()

{

\_enabled = false;

}

bool TriggerClass::isEnabled()

{

return \_enabled;

}

TriggerClass Trigger;

// Файл Trigger.h

// Trigger.h

#ifndef \_TRIGGER\_h

#define \_TRIGGER\_h

#if defined(ARDUINO) && ARDUINO >= 100

#include "arduino.h"

#else

#include "WProgram.h"

#endif

class TriggerClass

{

protected:

uint32\_t \_setAt;

bool \_enabled;

public:

void init();

void check();

void setEnabled();

void setDisabled();

bool isEnabled();

};

extern TriggerClass Trigger;

#endif