Transactions of the VŠB – Technical University of Ostrava, Mechanical Series
No. 1, 2017, vol. LXIII
article No. 2022

Marek BABIUCH*

WINDOWS 10 IOT APPLICATION FOR DATA ACQUISITION FROM SENSORS

APLIKACE NA PLATFORMĚ WINDOWS 10 IOT PRO ZÍSKÁVÁNÍ DAT ZE SENSORŮ

Abstract

The paper describes the creation of Windows IoT applications in Windows 10, specifically on mobile device for collecting and displaying data obtained from sensors connected to the Arduino development board. This article describes all the steps necessary to create an application in C #, development tools, hardware devices, and libraries in a compact summary which forms the algorithm of the implementation of Windows Universal Application on so-called IoT devices from different manufacturers.

Abstrakt

Příspěvek popisuje tvorbu aplikace v prostředí Windows 10, konkrétně na mobilním zařízení pro sběr a zobrazení dat získaných ze sensorů připojených na vývojovou desku Arduino. Příspěvek popisuje všechny nezbytné kroky k vytvoření aplikace v jazyce C#, vývojové prostředky, hw zařízení a knihovny v uceleném souhrnu, který tvoří postup implementace Windows Universal Apps na tzv. IoT zařízení.

Keywords

Arduino, application, Bluetooth, C#, USB, UWP, Visual Studio.NET, Windows10 IoT

1 USED PLATFORMS, TOOLS, HW DEVICES AND PROTOCOLS

I will gradually introduce the necessary building elements to create the Windows 10 mobile application to display the data obtained from the sensors connected to the Arduino board. Because it is a UWP application, the first condition is the development environment with installed Universal Windows App Development Tools, which includes SDK tools and device emulator. The following chapter is dedicated to the development of Windows Universal Apps. The Windows Remote Arduino library is available for the development with Uno, Mega and Leonardo boards, and we will use this library to implement the application and install it using the NuGet package. The Firmata protocol will be installed in the development board, allowing our application to access all Arduino ports. Last but not least, it is necessary to ensure communication between the development board and the deployed application on mobile device. This is done using the Bluetooth module, whose functionality we should ensure in our application, it means discover of device, connection setup and device configuration. In the final chapter, we will introduce the implementation of the application in C #, which shows the values of the sensors on the screen of mobile device connected the development board via Bluetooth Connection. Sensors will use the analogue and digital pins of developed board.

* doc., Ing., Ph.D., VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 17. listopadu 15, Ostrava - Poruba, 708 33, Czech Republic, tel. (+420) 59 732 4173, e-mail: marek.babiuch@vsb.cz

2 WINDOWS UNIVERSAL APPS

The described application is focused on Windows 10 IoT application, but has been developed as a UWP application. Universal platform defines one system kernel for all types of devices running under Windows 10 operating system. We can develop applications in a single development environment for these devices from the version of Visual Studio 2015 developed environment, currently version 2017. We can use the so-called SDK - Software Development Kit as an expanding development tool for some devices. UWP can be characterized as an application with a flexible user interface which allows to display the application on any screen of the target device with Windows 10. The layout of the application is proposed in the XAML file using the general elements of the design suite for the layout and navigation of the application.

Fig. 1 Windows Universal Apps Platform

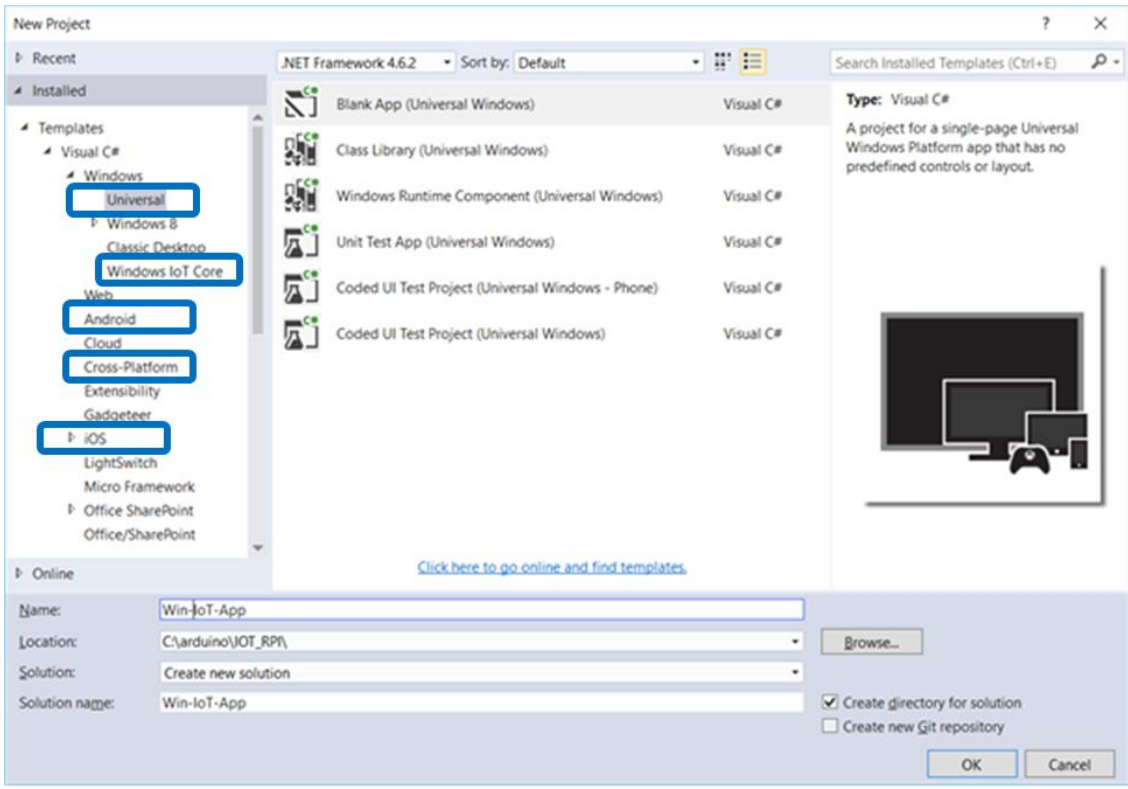

Fig. 2 UWP application template

3 WINDOWS 10 IOT CORE APPLICATION DEVELOPMENT

We develop Windows IoT in the Visual Studio 2017 environment, preferably in C #/C++, Python or node.js languages. Microsoft has recently put a great deal of emphasis on universal application programming across platforms but also in mobile devices we can already develop cross-platform applications for operating systems like android or IOS. In Figure 2, the options are highlighted in the development template. But stay with the Windows IOT application. Here, we can design a supported embedded device as a desktop or web application that supports wireless technology, either as a UWP application with a graphical user interface or as a Windows IOT background service. We deploy the application or service into the remote target device directly in the development environment. We can also set the deployed application on the target device as a startup through the Windows 10 IoT Core Dashboard tool running on a remote laptop. In addition to deploying applications and services, the IoT Core Dashboard Tool also allows us to configure the remote device. In figure 3, we see the Windows 10 IoT application on the Raspberry Pi with the mounted touch screen. The C # application enables the GPIO ports of the device and accesses the measured values of the connected sensors.

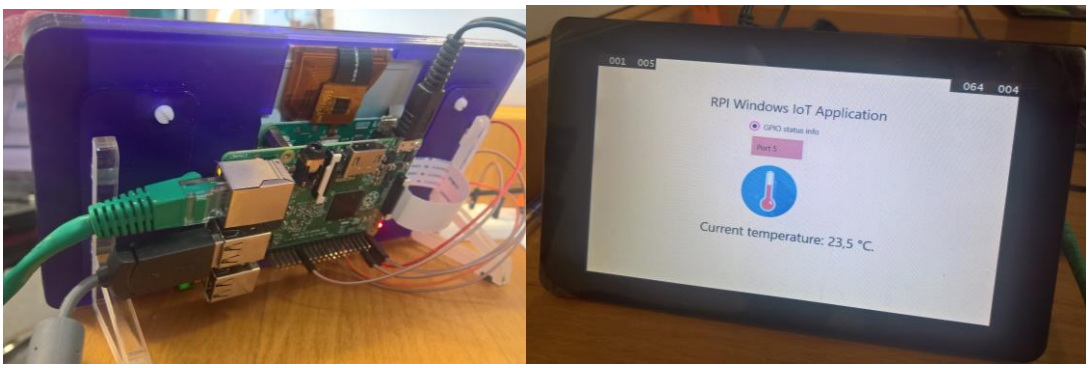

Fig. 3 Windows IoT Core application deployed on Raspberry PI device

4 SUPPORTED HARDWARE MODULES

We can essentially design the Arduino development board as well as Window IoT devices. At the present time Microsoft supports the development of Windows IoT core applications for Raspberry Pi 3, Raspberry Pi 2 with ARM Cortex (1.2GHz or 900MHz) and Minnow Board (Intel x88/x64 1.3GHz) and Dragon Board (ARM Cortex 1.2 GHz). Other supported devices should be SoC (System On Chip), BroadComm, QualComm, and Intel boards. On the Developer Portal page, we can also find a list of supported sensor, communication I / O modules and peripherals that can be used to develop the UWP application on the Windows IoT platform. These include except others Wifi, Bluetooth communication modules, Arduino compatible modules by Adafruit and SparkFun producers as accelerometers, motors, displays, data storage devices, ports expanders and shields for sensors. Supported buses and protocols for UWP and IoT applications are GPIO, SPI, I2C, and UART.

4.1 Windows Remote Arduino Library

The required library for remote communication with the development board is called Windows Remote Arduino. It is an open-source library designed to create a universal Windows application, meaning that it can be deployed on any Windows 10 device or Raspberry Pi2 and higher with Windows IoT. The library is available on the GitHub development portal. The architecture of communication between the deployed application and the Arduino development board is three-layer. The lower physical layer is named Stream, which provides data exchange between the host application and the GPIO terminal device and the connected sensors. The middle layer must detect these information frames and create communications according to Firmata's standardized protocol.

The top layer called Remote Wiring is an interface that abstracts all communication protocols to the RemoteDevice class level, which can be a USB, Bluetooth, Network, or BLE device. The library allows us to access GPIO pins on the development board and enables:

- Configuration of pins as input and output, digital or analog.
- Reading and writing of digital pins using *digitalRead()* and *digitalWrite()* functions.
- Analog Input/Output (PWM) using *analogRead()* and *analogWrite()* functions.
- Event management when the status of the pins is changed.
- Transmit and receive data using the I2C protocol.
- Creating of custom protocol via Firmata.

Therefore, the RemoteDevice class is the API entry point, representing the communication interface that we will use in the guest application and which will make all calls with the Firmata layer below it.

4.2 Firmata Layer and Protocol

It was necessary for the API design to use the appropriate protocol to facilitate communication between Windows 10 and the development board. For this functionality, a widely accepted open source Firmata protocol has been chosen, which has been implemented in many languages, including Arduino Wiring. The Firmata Library for Arduino boards is also included in the Arduino IDE development environment.

The Firmata layer implementation is performed directly from the Firmata repository with minimal changes only with the removing of Arduino hardware dependencies and is packaged with a transparent class of the Windows Runtime Component named *UwpFirmata*.

4.3 Application deployment

Windows 10 unlike previous versions of mobile Windows, allows us to unlock device developer mode without any problems with just a quick configuration in the security settings section - for developers. Then, we can deploy the application directly from the Visual Studio development environment on Windows 10 mobile devices. This is done with the settings on the ARM platform, see figure 4. We need a mobile device connected to the developer PC using a USB cable. After the application has been deployed, we have already created the application among the available installed applications.

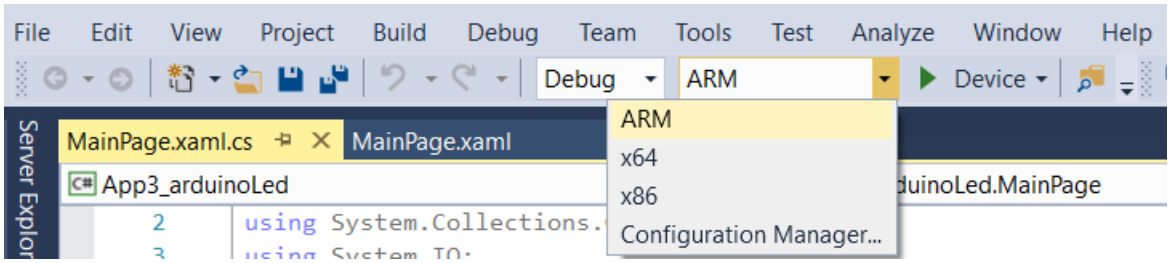

Fig. 4 Deploying and debuging on ARM processor device

5 DESIGNED APPLICATION

The application has been designed to display data obtained from both the analogue and digital GPIO pins of the development board. Communication on the development board side is provided by the SPP-C Bluetooth module shown in Figure 5 on the left. Then we used digital DHT11 sensor, analogue temperature sensor KY013, laser module and RGB diode for data displaying from digital

and analog GPIO pins. So we can measure temperature by analog and also digital sensor, switch the laser module, and show application states by control color lights in our mobile application.

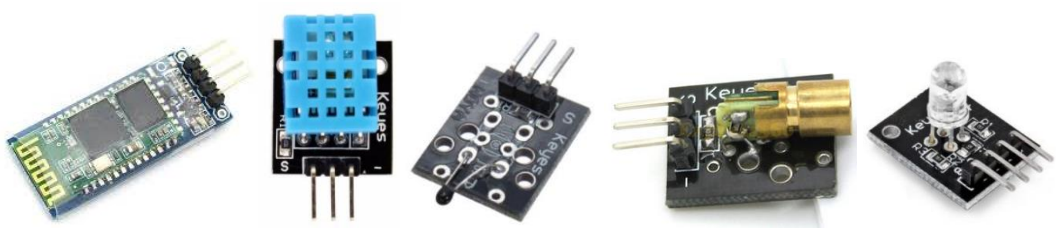

Fig. 5 Application's IoT modules

The look of the demonstration application for communicating the Microsoft Lumia 950 mobile device with the Windows 10 Mobile operating system and the Arduino development board was designed in the development environment in the *MainPage.xaml* file. At first, the appropriate resolution and screen size was selected and the controls, such as buttons, textbooks, panels, descriptions, and images, were transferred from the toolbox to the workplace. These controls are used to initiate communication queries on sensors connected to the GPIO pins of the development board and show the acquired values in text fields.

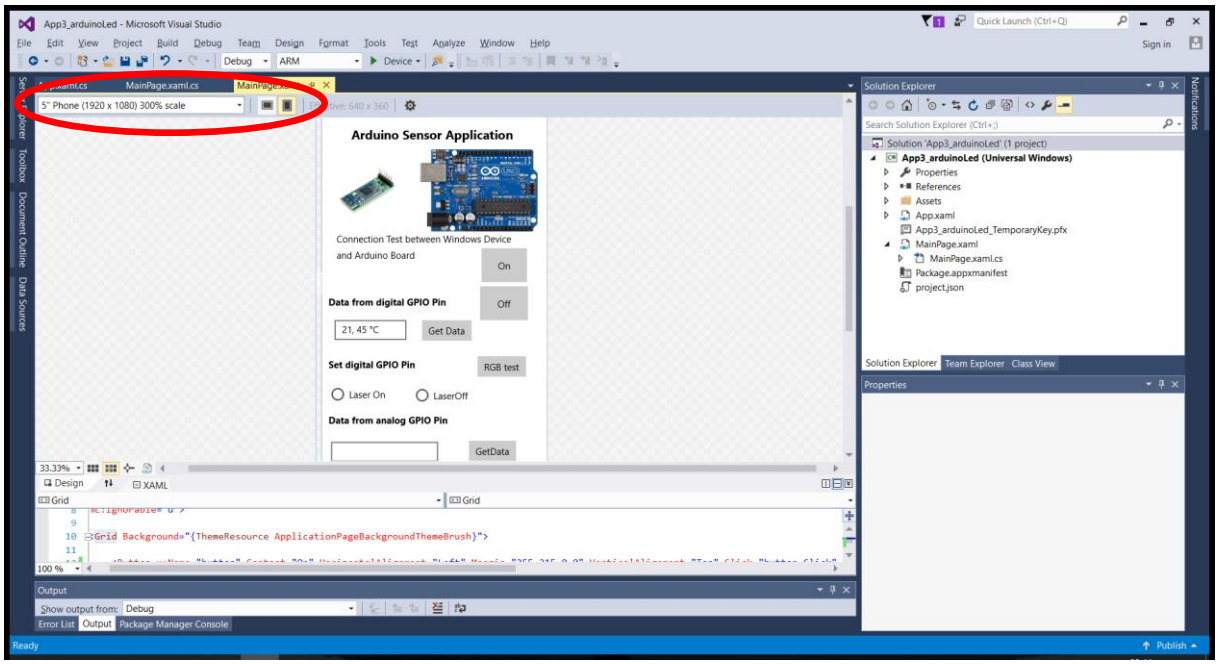

Fig. 6 Application design in MainPage.xaml file

Each Windows project contains a manifest file that must be configured to allow certain permissions, such as connecting an application using a network connection, Bluetooth, or USB. Here is the fragment of XML configuration in the *Capabilities* element.

```
<Capabilities>
   <DeviceCapability Name="bluetooth.rfcomm">
      <Device Id="any"><Function Type="name:serialPort"/></Device>
   </DeviceCapability>
</Capabilities>
```

The main part of the source code is in the *MainPage.xaml.cs* file. First, we create a Bluetooth connection object for our application and transfer it to the RemoteDevice class in the constructor. This can be done by directly registering the device name (the SPP-C Bluetooth device is actually named Port in our case). The available device can also be obtained by asynchronous calling the *static.listAvailableDevicesAsync()* function in the *BluetoothSerial* or *USBSerial* class before creating the object. We also create the *ConnectionEstablished* event, where we then enable the application control elements in the program code. We also need to define communication parameters such as bit rate, data, parity, stop bit, all the options defined by the *SerialConfig* of the *RemoteDevice* object.

```
public sealed partial class MainPage : Page
    {
        IStream connection;
        RemoteDevice arduino;
        public MainPage()
        {
            this.InitializeComponent();
           // connection = new BluetoothSerial("name_of Bluetooth device");
              connection = new BluetoothSerial("Port");
            arduino = new RemoteDevice(connection);
            connection.ConnectionEstablished +=
Connection_ConnectionEstablished;
            connection.begin(9600,SerialConfig.SERIAL_8N1);
        }
```

At now we are directly accessing the GPIO pins of the RemoteDevice object in the application. In the following sample code, we define pin A0 as an analog and process the *AnalogRead()* data.

```
private void button_GetData(object sender, RoutedEventArgs e)
        {
            int value;
            arduino.pinMode("A0", PinMode.ANALOG);
            string sensorPin = "A0";
            value = arduino.analogRead(sensorPin);
            tbData.Text = Thermister(value).ToString();
             }
```

Similarly, we approach to the digital pins of the development board. The following source code fragment shows the operator turning the laser module on and off by using the *RadioButton* control. Basically, it's about sending the High or Low values to the appropriate pin using the *digitalWrite()* function.

```
private void radioButton_Checked_On(object sender, RoutedEventArgs e)
      { arduino.digitalWrite(3, PinState.HIGH); }

private void radioButton_Checked_Off(object sender, RoutedEventArgs e)
      { arduino.digitalWrite(3, PinState.LOW); }
```

6 CONCLUSIONS

The created application demonstrates the current trend of creating UWP applications with an emphasis on IoT devices. For the sample application, the Arduino development board was used to demonstrate application development in a different way than usual in the Arduino IDE. The programming language used is C # in Visual Studio from version 2015 above. Besides this language, we can also use Python, C ++ or Node.js to create UWP standard, background or web based applications., Microsoft has already implemented Windows 10 IoT application's support for Raspberry, DragonBoard, and other boards in Visual Studio environment. In addition to these tasks, we can also develop applications on development boards that support .NET Microframework in this environment. Hot topic at the present time is the application development for small IoT devices as well as application development for various mobile platforms. To this trend, Microsoft has responded by incorporating Xamarin's platform for creating Android mobile applications, as well as launching another .NET Core cross-platform development technology for applications running Linux and MacOS distributions. These technologies will be presented in the future contributions of author.

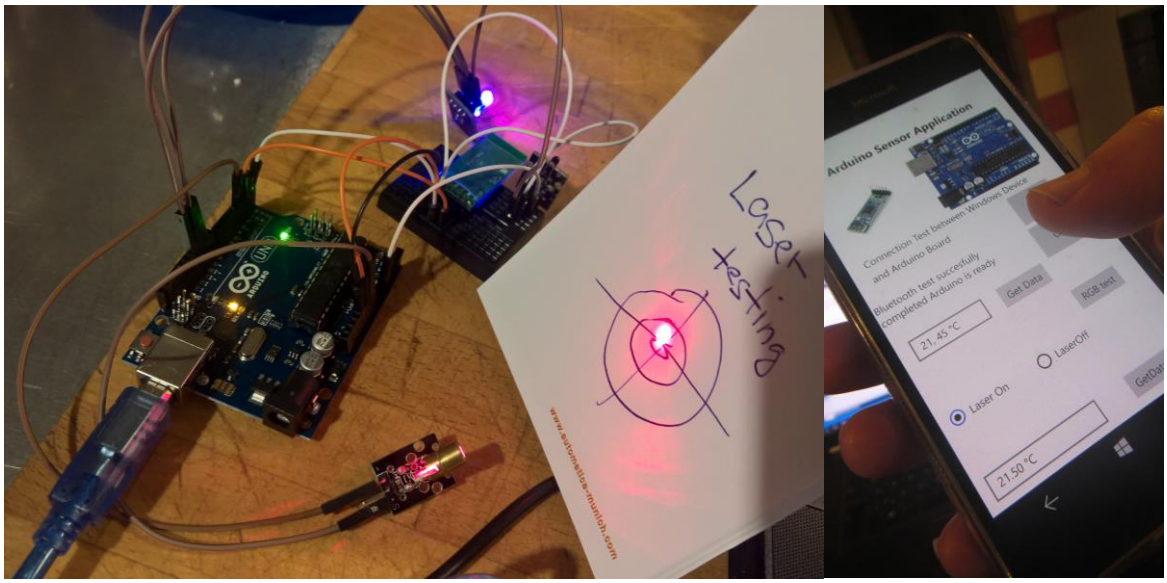

Fig. 7 Developed board and Win10 IoT mobile application

REFERENCES

[1] ARDUINO. *Firmata library*. [online]. 2017 [cit. 2017-09-28]. Available from: https://www.arduino. cc/en/reference/firmata

[2] BABIUCH, M. Vývoj aplikací na platformě .NET Gadgeteer. In *XXIX. Seminar ASR '2015 "Instruments and Control"*. Ostrava: VŠB-TU Ostrava, 24. 4. 2015, pp. 5-13. ISBN: 978-80-248-3744-4.

[3] CZEBE, J., ŠKUTA, J. Usage of single-chip computers for data collection and control. In *Proceedings of 17th International Carpathian Control Conference* ICCC´2016. Tatranská Lomnica, Slovakia, May 29- June 01, 2016, pp. 134-139. ISBN: 978-146738606-7, DOI: 10.1109/CarpathianCC.2016.7501081.

[4] FOJTÍK, D., PODEŠVA, P., GEBAUER J. Storing High Volumes of Data in MS SQL Server Express. In *Proceedings of 16th International Carpathian Control Conference* ICCC´2015. Szilvasvarad, Hungary, May 27-30, 2015, pp. 125-128. ISBN: 978-1-4799-7370-5, DOI: 10.1109/CarpathianCC.2015.7145059.

[5] GITHUB, Inc.(US). *A remote "Arduino Wiring" interface to control an Arduino compatible device from a Windows 10 Universal Windows*. [online]. 2016 [cit. 2017-09-28]. Available from: https://github.com/ms-iot/remote-wiring

[6] PINGUINO-wiki. *SPP Bluetooth Modules*. [online]. 2014 [cit. 2017-09-28]. Available from: //wiki.pinguino.cc/index.php/SPP_Bluetooth_Modules

[7] SLADKÁ, K., BABIUCH, M. Application of control algorithms in.NET Micro Framework Technology. In *Proceedings of 17th International Carpathian Control Conference* ICCC´2016. Tatranská Lomnica, Slovakia, May 29- June 01, 2016, pp. 684-687. ISBN: 978-146738606-7, DOI: 10.1109/CarpathianCC.2016.7501182.

[8] WINDOWS DEV CENTER PORTAL. *Windows 10 IOT Core - The operating system built for the Internet of Things*. [online]. 2017 [cit. 2017-09-28]. Available from: https://developer.microsoft. com/cs-cz/windows/iot

[9] WINDOWS DEV CENTER PORTAL. *Vytvářejte skvělé aplikace pro Windows – Průvodce aplikacemi pro UWP*. [online]. 2017 [cit. 2017-09-28]. Available from: https://developer.microsoft.com/cs-cz/windows/apps