Microsoft

Develop your UWP App on Win10 IoT Core

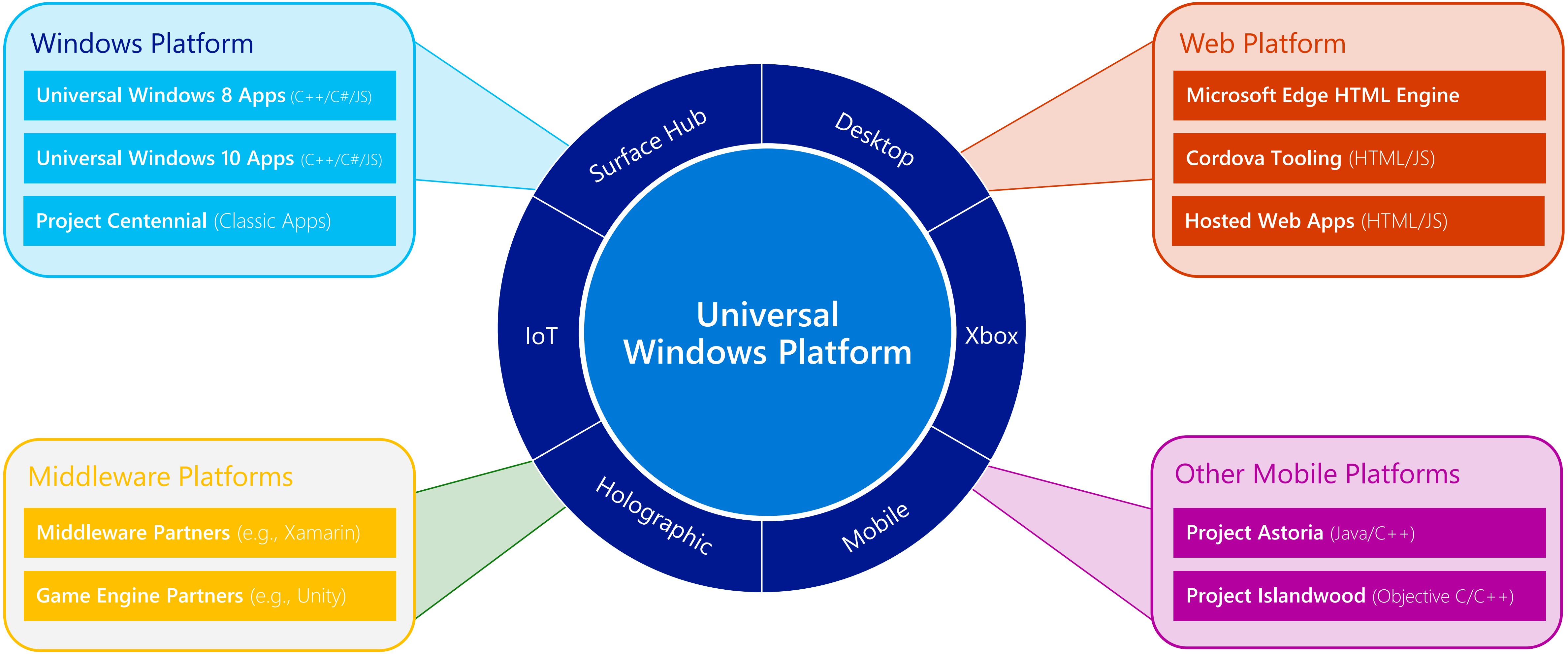
Hancom MDS Inc.

Agenda

- Windows 10 UWP App Overview
- Adaptive UWP App
- UWP Device Apps Development

Windows 10 UWP App Overview

UWP becomes the one platform for developers. Learn one set of core APIs for all devices.



Universal Windows Platform

Surface Hub

Desktop

Xbox

Mobile

Holographic

IoT

Windows Platform

- **Universal Windows 8 Apps** (C++/C#/JS)
- **Universal Windows 10 Apps** (C++/C#/JS)
- **Project Centennial** (Classic Apps)

Web Platform

- **Microsoft Edge HTML Engine**
- **Cordova Tooling** (HTML/JS)
- **Hosted Web Apps** (HTML/JS)

Middleware Platforms

- **Middleware Partners** (e.g., Xamarin)
- **Game Engine Partners** (e.g., Unity)

Other Mobile Platforms

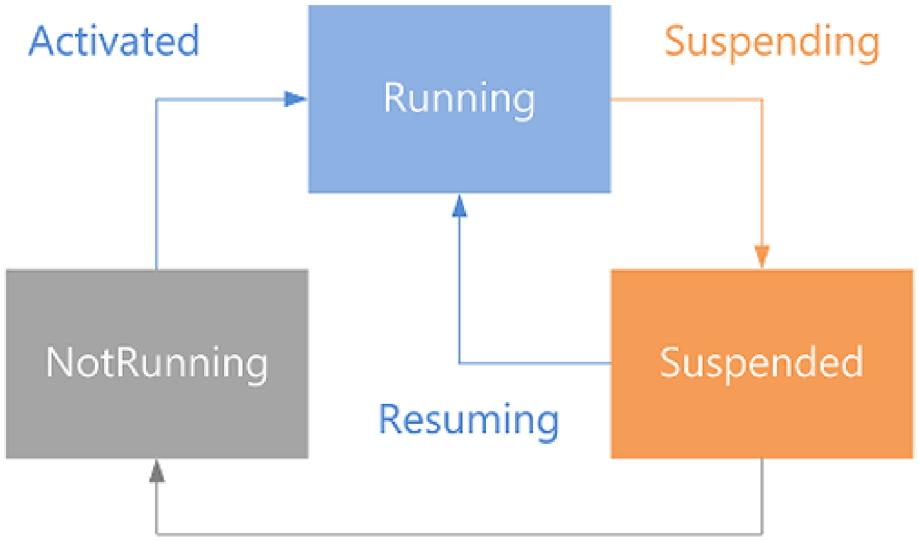
- **Project Astoria** (Java/C++)
- **Project Islandwood** (Objective C/C++)

Application Lifecycle

How Windows interacts with your app

- Suspend and resume
- Background execution
- Resource management
- System triggers and notifications

Application Lifetime



Apps can be in 1 of 3 states

- Not Running
- Running
- Suspended

Application receive events when transitioning between states

- Except: Suspended->NotRunning

Extended Execution

Continue a session when not in the foreground

- Location Tracking
- Save critical data
- You just want more time

Background Execution

Background Execution

- Apps provide real-time content even when suspended

Draw users into your app

Delight them with features

Trigger based Background Tasks

- Apps subscribe to triggers they are interested in
  Only run *when* trigger is fired

- Exmaple

  - Push notification
  - Geofencing
  - BLE device
  - Timer
  - Sensors

Adaptive UWP App

Multiple Adaptive Dimensions

Version Adaptive

- App runs on a base OS version but can use up-level APIs

Device family adaptive

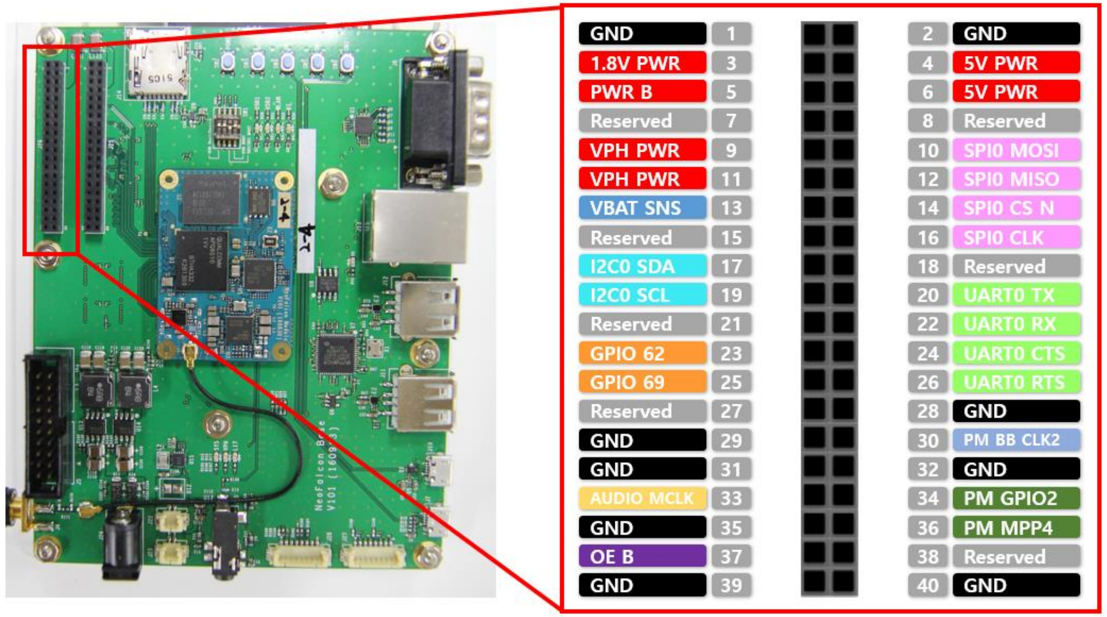
- App uses device family-specific APIs when running on such a device

Form factor adaptive aka responsive layout

- App provides user-interface tailored to one or more specific form factors

UWP IoT Device Apps Development

Electrical Engineering for SW Engineers



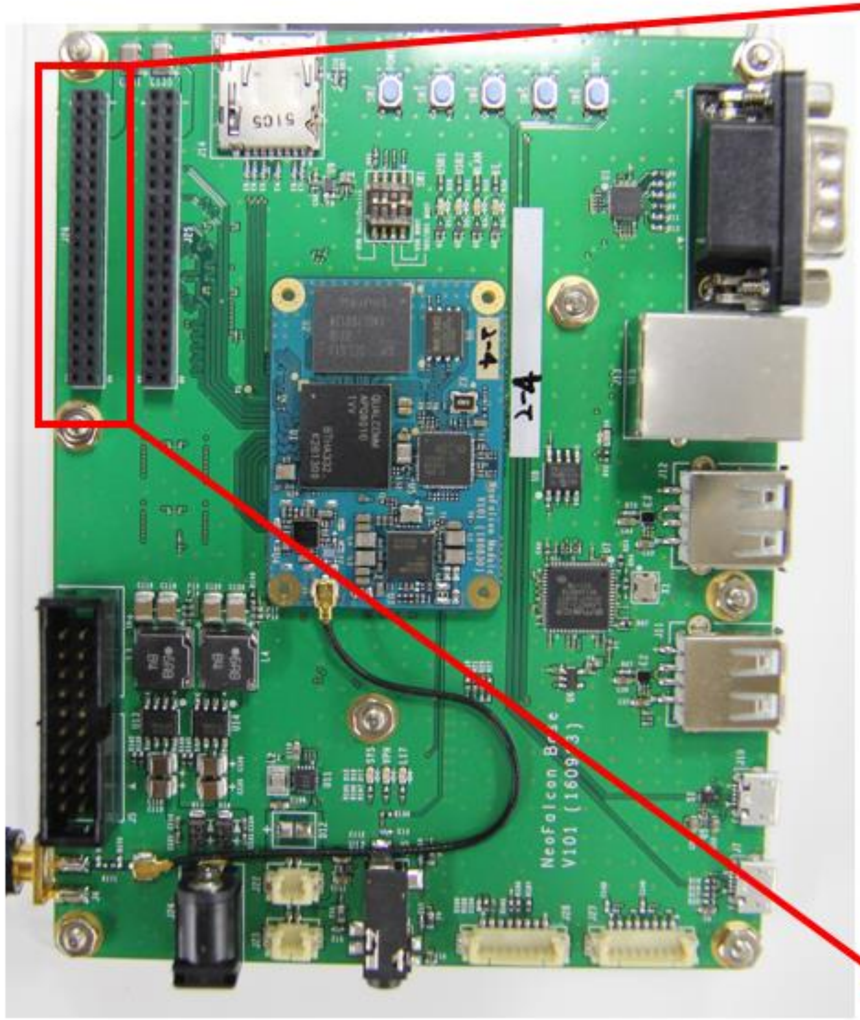
GND	1	2	GND
1.8V PWR	3	4	5V PWR
PWR B	5	6	5V PWR
Reserved	7	8	Reserved
VPH PWR	9	10	SPI0 MOSI
VPH PWR	11	12	SPI0 MISO
VBAT SNS	13	14	SPI0 CS N
Reserved	15	16	SPI0 CLK
I2C0 SDA	17	18	Reserved
I2C0 SCL	19	20	UART0 TX
Reserved	21	22	UART0 RX
GPIO 62	23	24	UART0 CTS
GPIO 69	25	26	UART0 RTS
Reserved	27	28	GND
GND	29	30	PM BB CLK2
GND	31	32	GND
AUDIO MCLK	33	34	PM GPIO2
GND	35	36	PM MPP4
OE B	37	38	Reserved
GND	39	40	GND

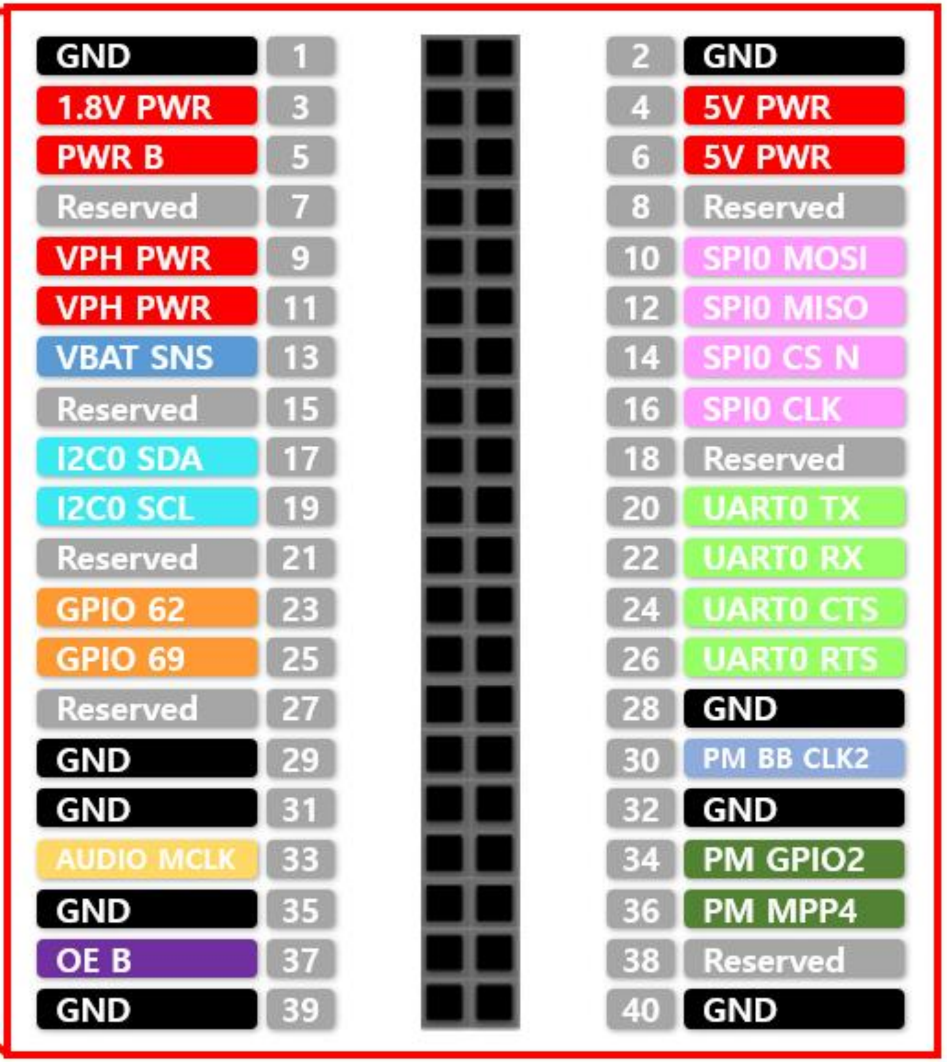
PCIe ➡ SPI: Higher speed, fewer available

PCI ➡ I2C: Lower speed, more available

RS-232 ➡ GPIO: DIY communication

NeoFalcon 410





GND	1	2	GND
1.8V PWR	3	4	5V PWR
PWR B	5	6	5V PWR
Reserved	7	8	Reserved
VPH PWR	9	10	SPI0 MOSI
VPH PWR	11	12	SPI0 MISO
VBAT SNS	13	14	SPI0 CS N
Reserved	15	16	SPI0 CLK
I2C0 SDA	17	18	Reserved
I2C0 SCL	19	20	UART0 TX
Reserved	21	22	UART0 RX
GPIO 62	23	24	UART0 CTS
GPIO 69	25	26	UART0 RTS
Reserved	27	28	GND
GND	29	30	PM BB CLK2
GND	31	32	GND
AUDIO MCLK	33	34	PM GPIO2
GND	35	36	PM MPP4
OE B	37	38	Reserved
GND	39	40	GND

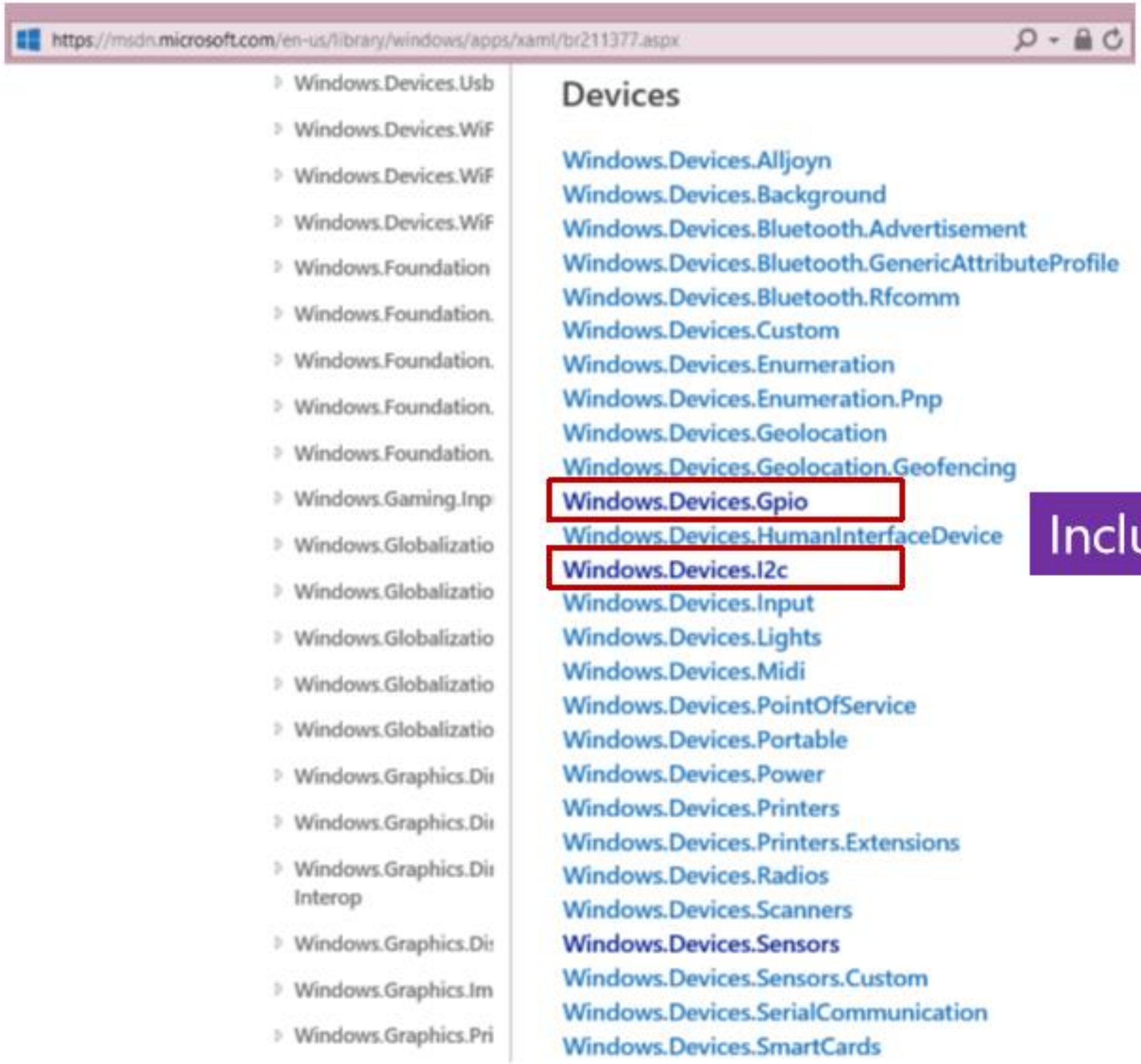
Your complete Maker toolkit

Windows 10

Visual Studio

+

Windows.Devices Namespace



https://msdn.microsoft.com/en-us/library/windows/apps/xaml/br211377.aspx

- Windows.Devices.Usb
- Windows.Devices.WiF
- Windows.Devices.WiF
- Windows.Devices.WiF
- Windows.Foundation
- Windows.Foundation.
- Windows.Foundation.
- Windows.Foundation.
- Windows.Foundation.
- Windows.Gaming.Inp
- Windows.Globalizatio
- Windows.Globalizatio
- Windows.Globalizatio
- Windows.Globalizatio
- Windows.Globalizatio
- Windows.Graphics.Di
- Windows.Graphics.Di
- Windows.Graphics.Di Interop
- Windows.Graphics.Di
- Windows.Graphics.Im
- Windows.Graphics.Pri

Devices

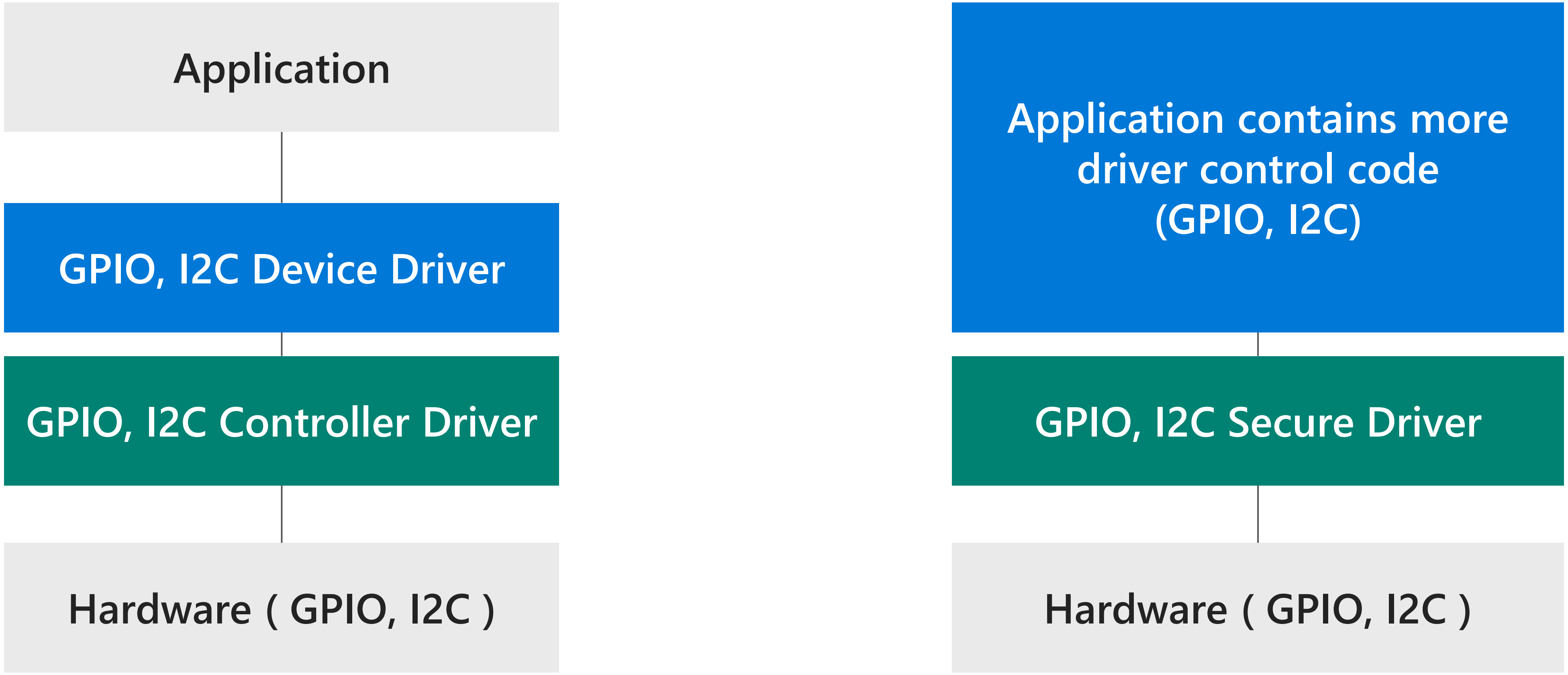
Windows.Devices.Alljoyn
Windows.Devices.Background
Windows.Devices.Bluetooth.Advertisement
Windows.Devices.Bluetooth.GenericAttributeProfile
Windows.Devices.Bluetooth.Rfcomm
Windows.Devices.Custom
Windows.Devices.Enumeration
Windows.Devices.Enumeration.Pnp
Windows.Devices.Geolocation
Windows.Devices.Geolocation.Geofencing
Windows.Devices.Gpio
Windows.Devices.HumanInterfaceDevice
Windows.Devices.I2c
Windows.Devices.Input
Windows.Devices.Lights
Windows.Devices.Midi
Windows.Devices.PointOfService
Windows.Devices.Portable
Windows.Devices.Power
Windows.Devices.Printers
Windows.Devices.Printers.Extensions
Windows.Devices.Radios
Windows.Devices.Scanners
Windows.Devices.Sensors
Windows.Devices.Sensors.Custom
Windows.Devices.SerialCommunication
Windows.Devices.SmartCards

Includes APIs to direct access buses

Accessing Buses Directly

- Windows.Devices.I2c
  - Contains types that you can use to communicate with peripheral devices connected through a inter-integrated circuit (I^2C) bus from an application.
- Windows.Devices.Gpio
  - Contains types for using general-purpose I/O (GPIO) pins in user mode.

UWP Access to Custom Hardware


Application

GPIO, I2C Device Driver

GPIO, I2C Controller Driver

Hardware (GPIO, I2C)

Application contains more driver control code (GPIO, I2C)

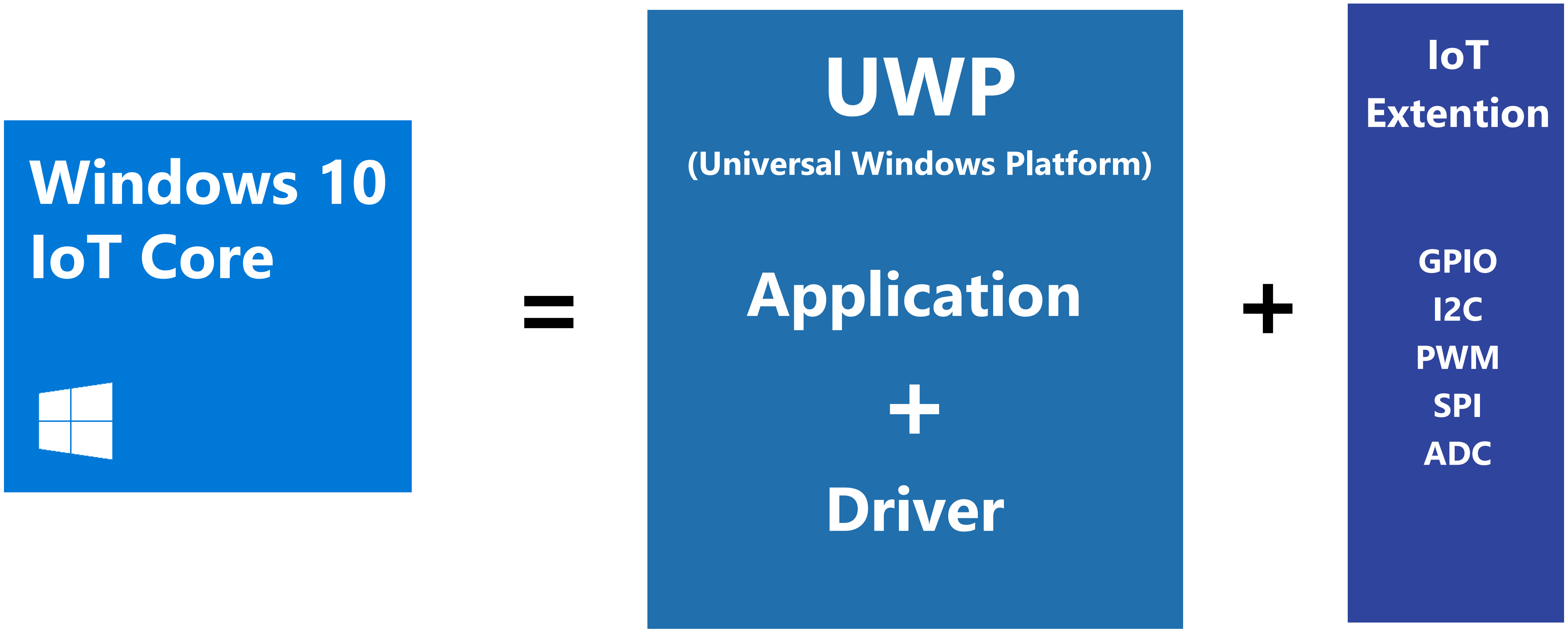
GPIO, I2C Secure Driver

Hardware (GPIO, I2C)

<Windows 10>

<Windows 10 IoT Core>

Windows 10 IoT Core



Windows 10 IoT Core

=

UWP

(Universal Windows Platform)

Application

+

Driver

+

IoT Extention

GPIO

I2C

PWM

SPI

ADC

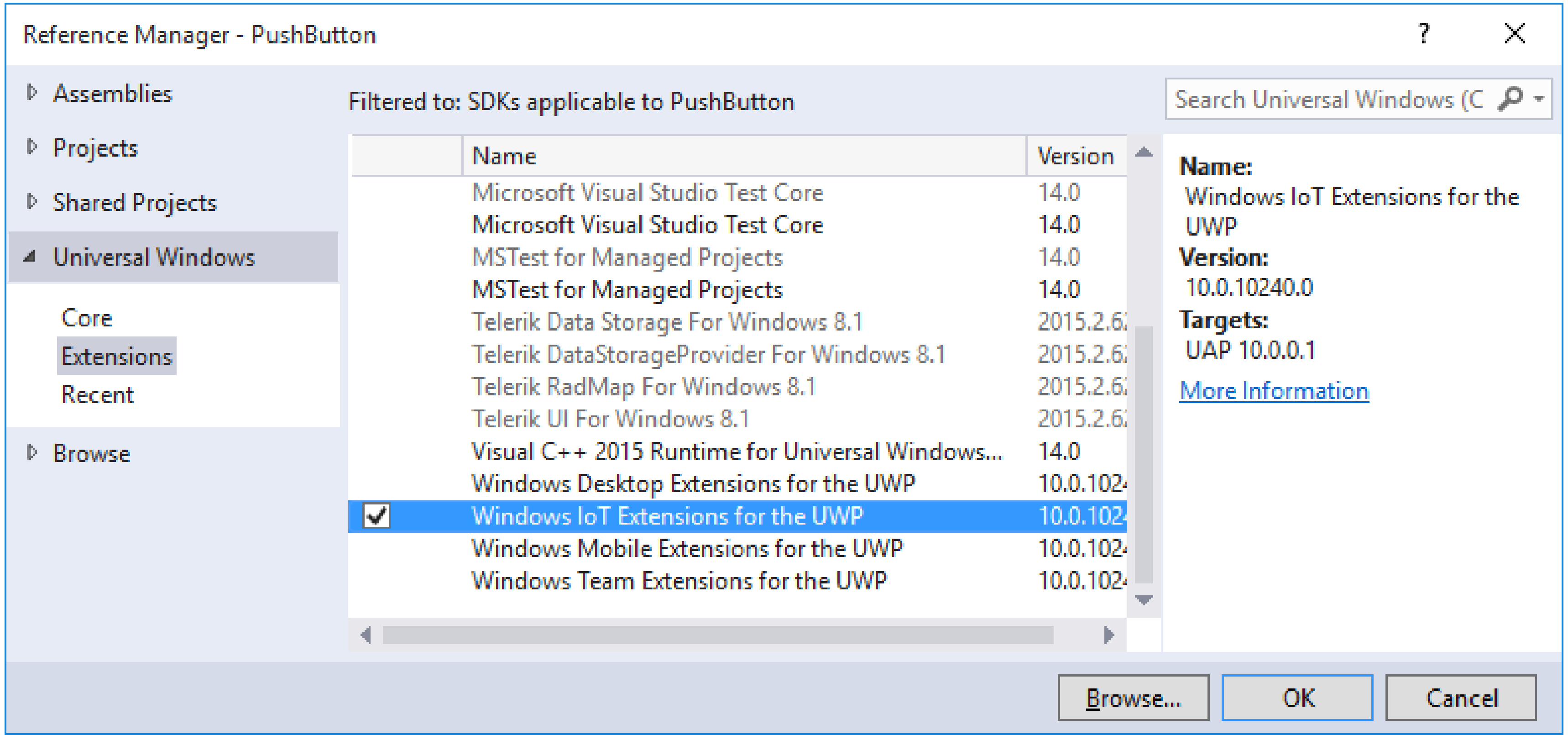
IoT Extentions for the UWP

Windows.devices.gpio

Windows.devices.i2c

Windows.devices.SPI

…

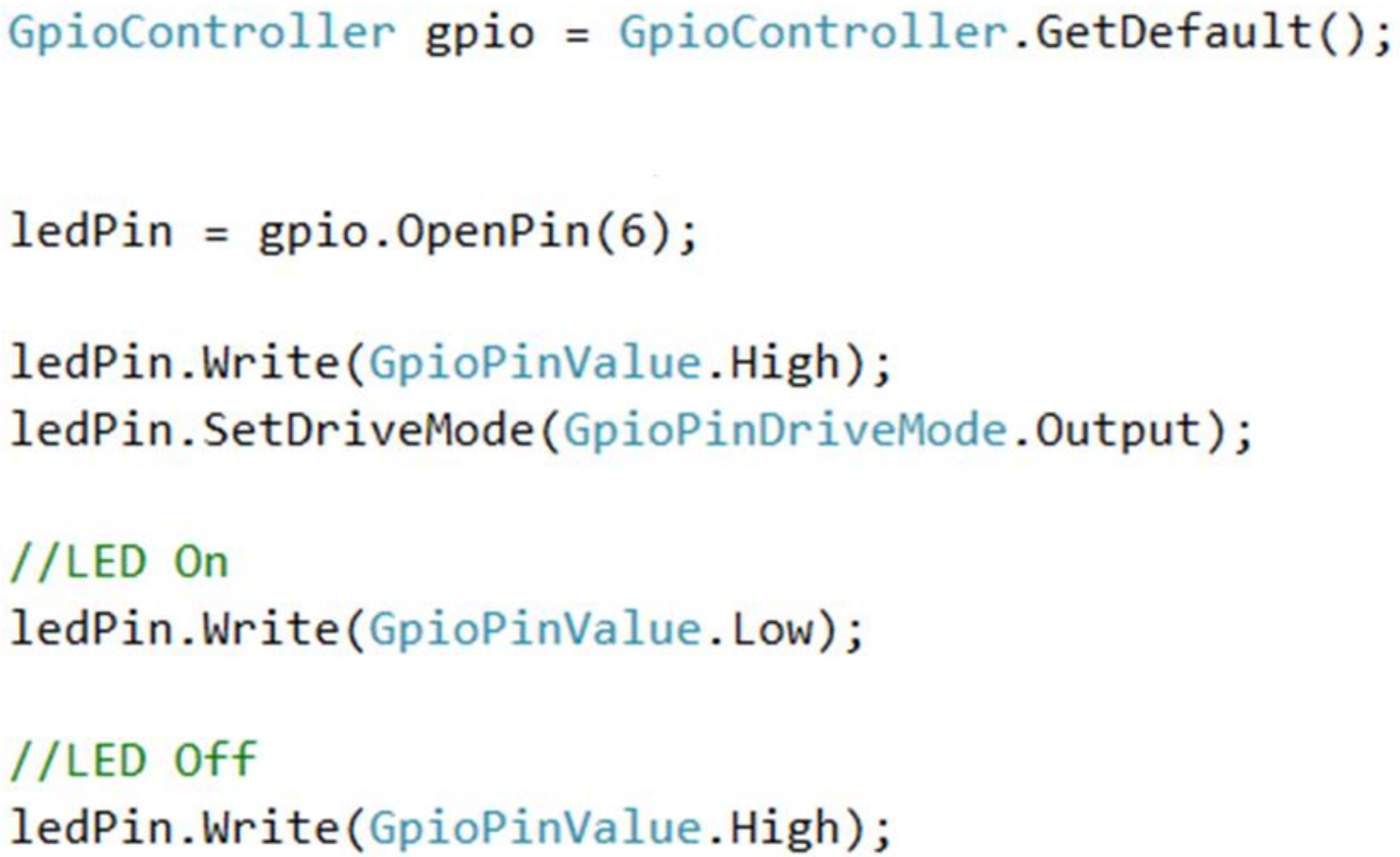


Filtered to: SDKs applicable to PushButton

	Name	Version
	Microsoft Visual Studio Test Core	14.0
	Microsoft Visual Studio Test Core	14.0
	MSTest for Managed Projects	14.0
	MSTest for Managed Projects	14.0
	Telerik Data Storage For Windows 8.1	2015.2.6
	Telerik DataStorageProvider For Windows 8.1	2015.2.6
	Telerik RadMap For Windows 8.1	2015.2.6
	Telerik UI For Windows 8.1	2015.2.6
	Visual C++ 2015 Runtime for Universal Windows...	14.0
	Windows Desktop Extensions for the UWP	10.0.102
☑	Windows IoT Extensions for the UWP	10.0.102
	Windows Mobile Extensions for the UWP	10.0.102
	Windows Team Extensions for the UWP	10.0.102

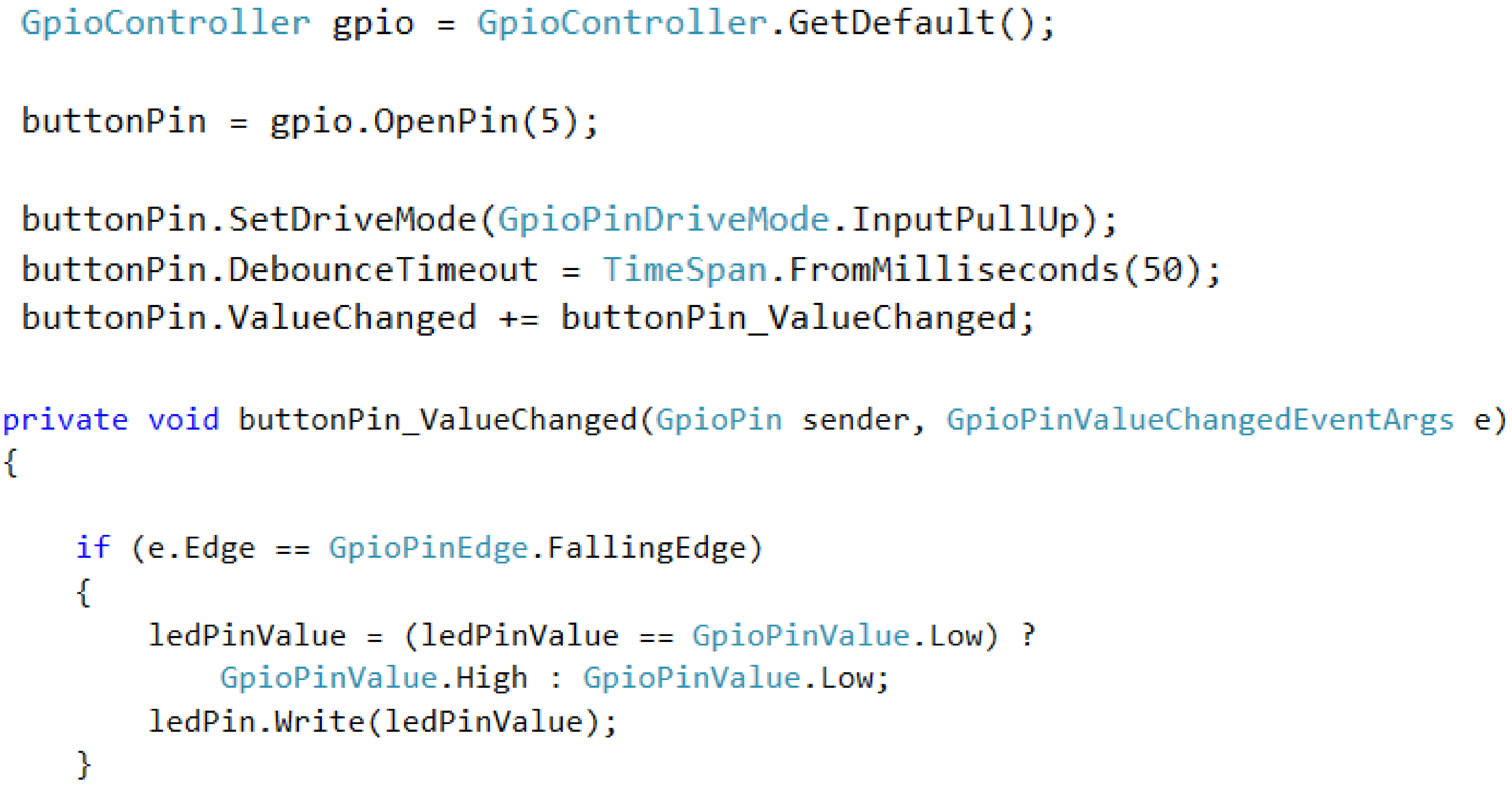
Name:
Windows IoT Extensions for the UWP
Version:
10.0.10240.0
Targets:
UAP 10.0.0.1
More Information

Windows.Devices.Gpio Output configuration



```
GpioController gpio = GpioController.GetDefault();


ledPin = gpio.OpenPin(6);

ledPin.Write(GpioPinValue.High);
ledPin.SetDriveMode(GpioPinDriveMode.Output);

//LED On
ledPin.Write(GpioPinValue.Low);

//LED Off
ledPin.Write(GpioPinValue.High);
```

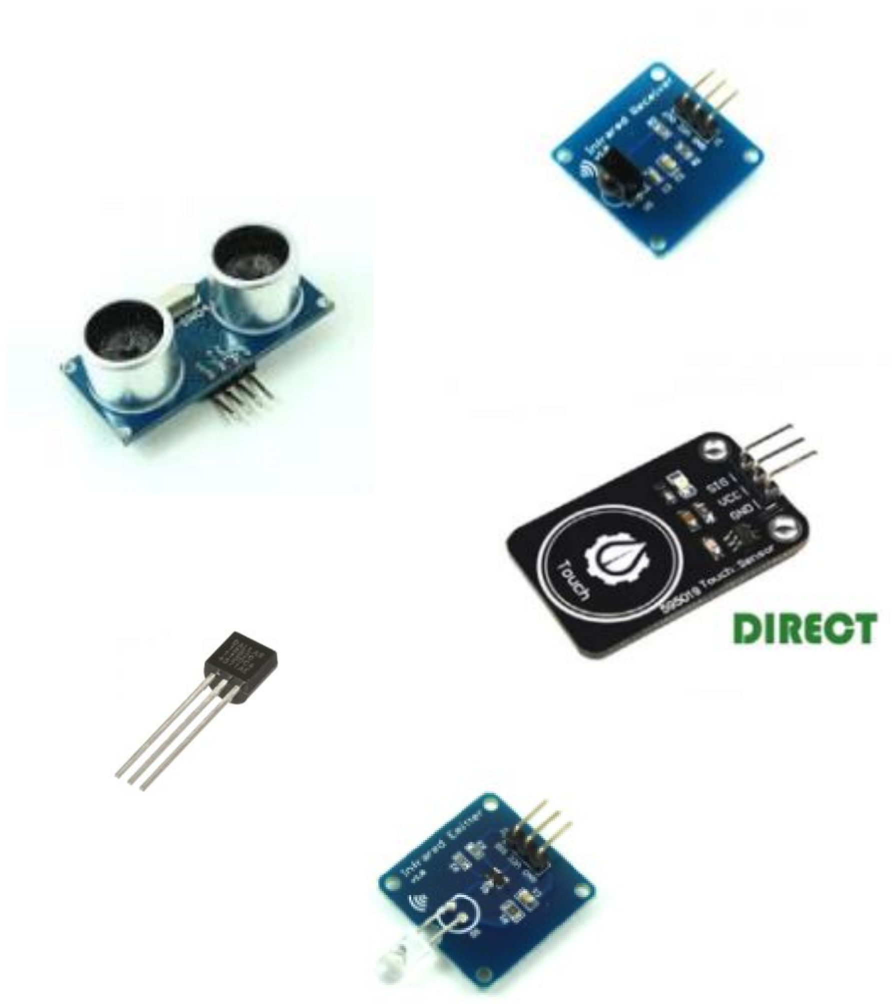
Windows.Devices.Gpio Input configuration



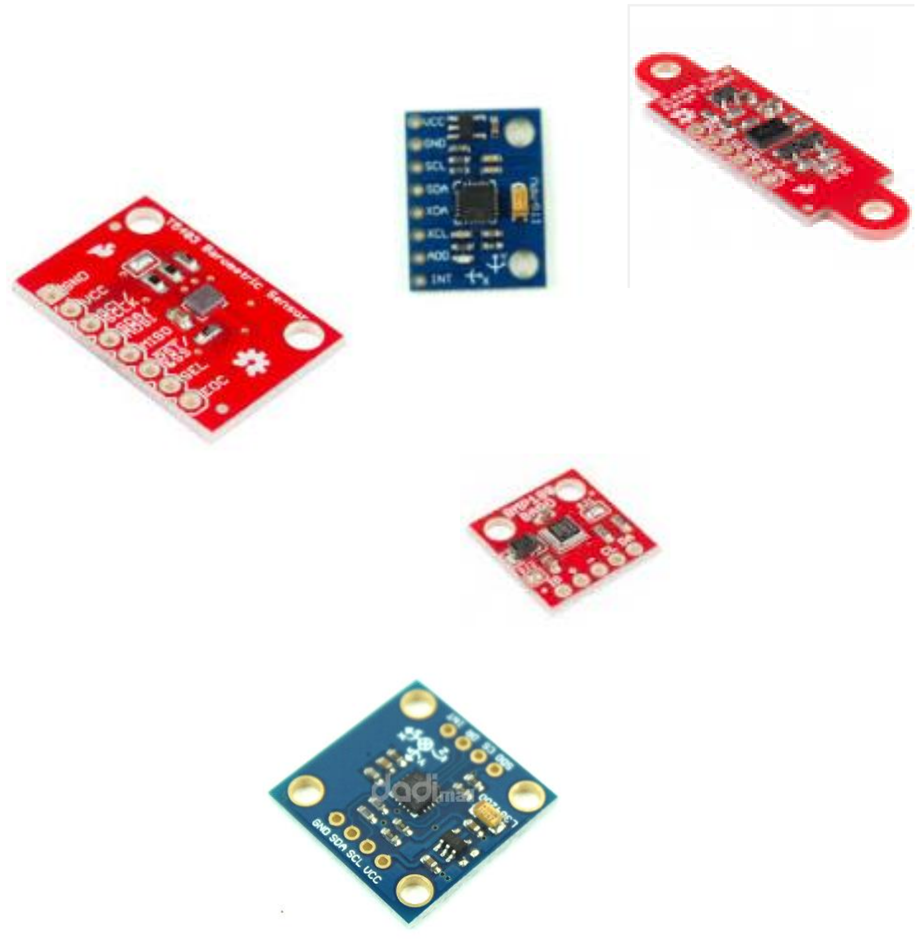
```
 GpioController gpio = GpioController.GetDefault();

 buttonPin = gpio.OpenPin(5);

 buttonPin.SetDriveMode(GpioPinDriveMode.InputPullUp);
 buttonPin.DebounceTimeout = TimeSpan.FromMilliseconds(50);
 buttonPin.ValueChanged += buttonPin_ValueChanged;

private void buttonPin_ValueChanged(GpioPin sender, GpioPinValueChangedEventArgs e)
{

    if (e.Edge == GpioPinEdge.FallingEdge)
    {
        ledPinValue = (ledPinValue == GpioPinValue.Low) ?
            GpioPinValue.High : GpioPinValue.Low;
        ledPin.Write(ledPinValue);
    }
```

HOL 2-1 OS GPIO Control

GPIO vs I2C Sensor

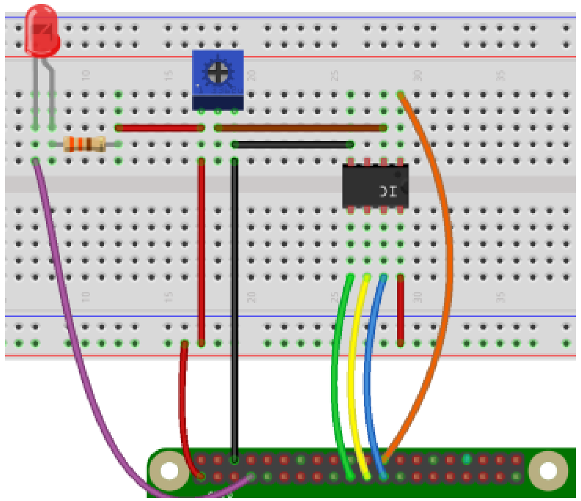


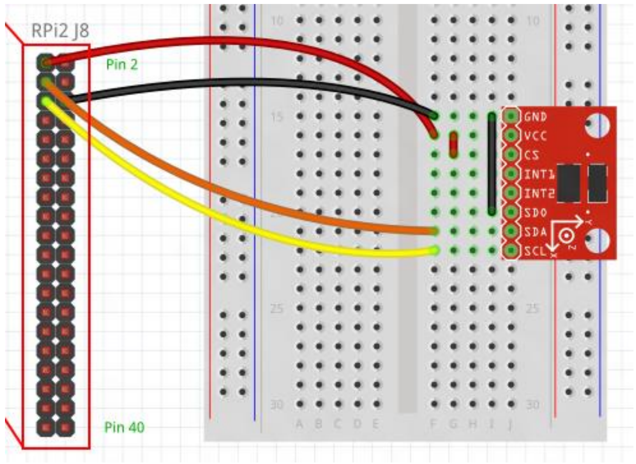
VS



The advantage of I2C/SPI Digital Sensor

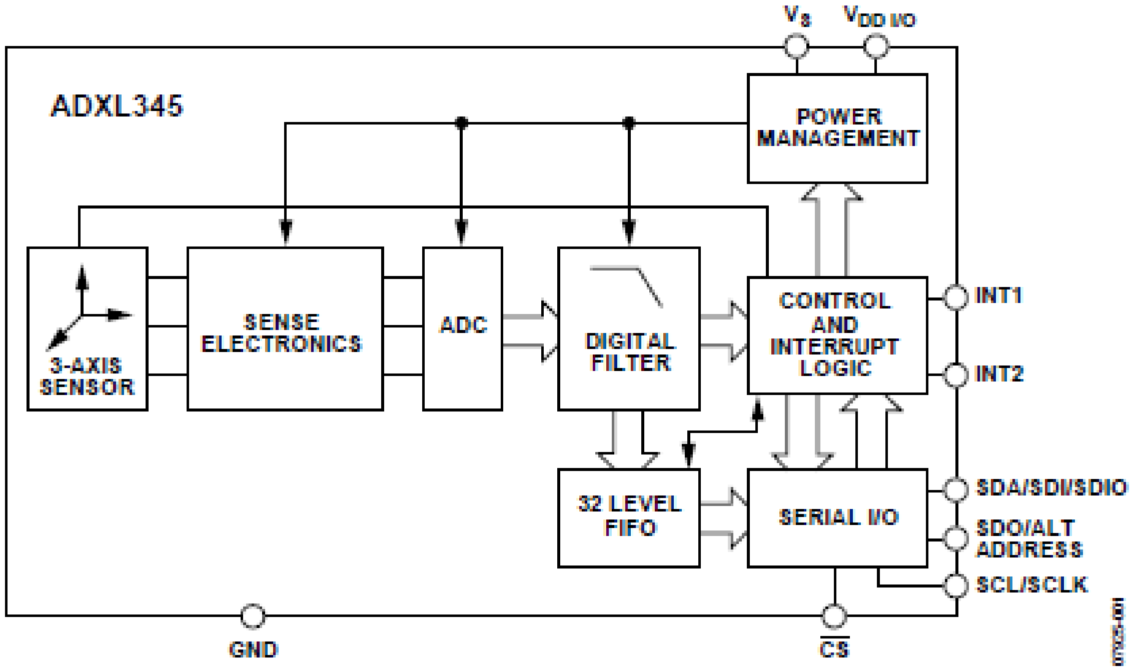
- Contains Sensor and ADC
- Data buffer, Register, I2C interface
- Effectively immune to noise when compared to an analog signal (Noise Free)
- All-In-One

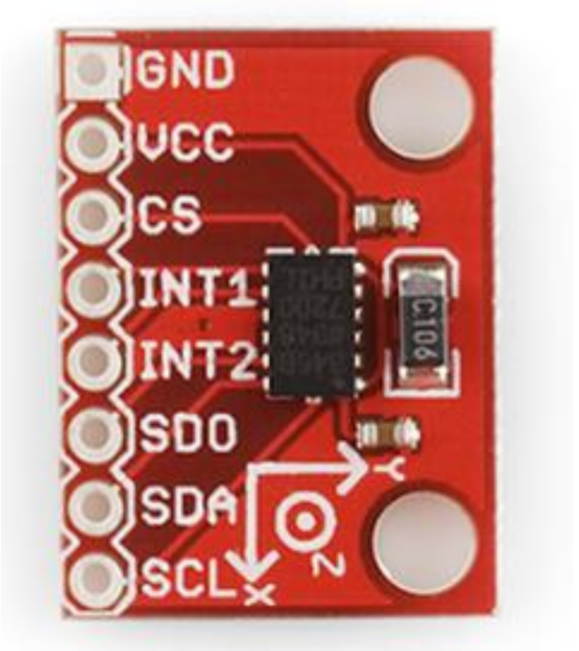




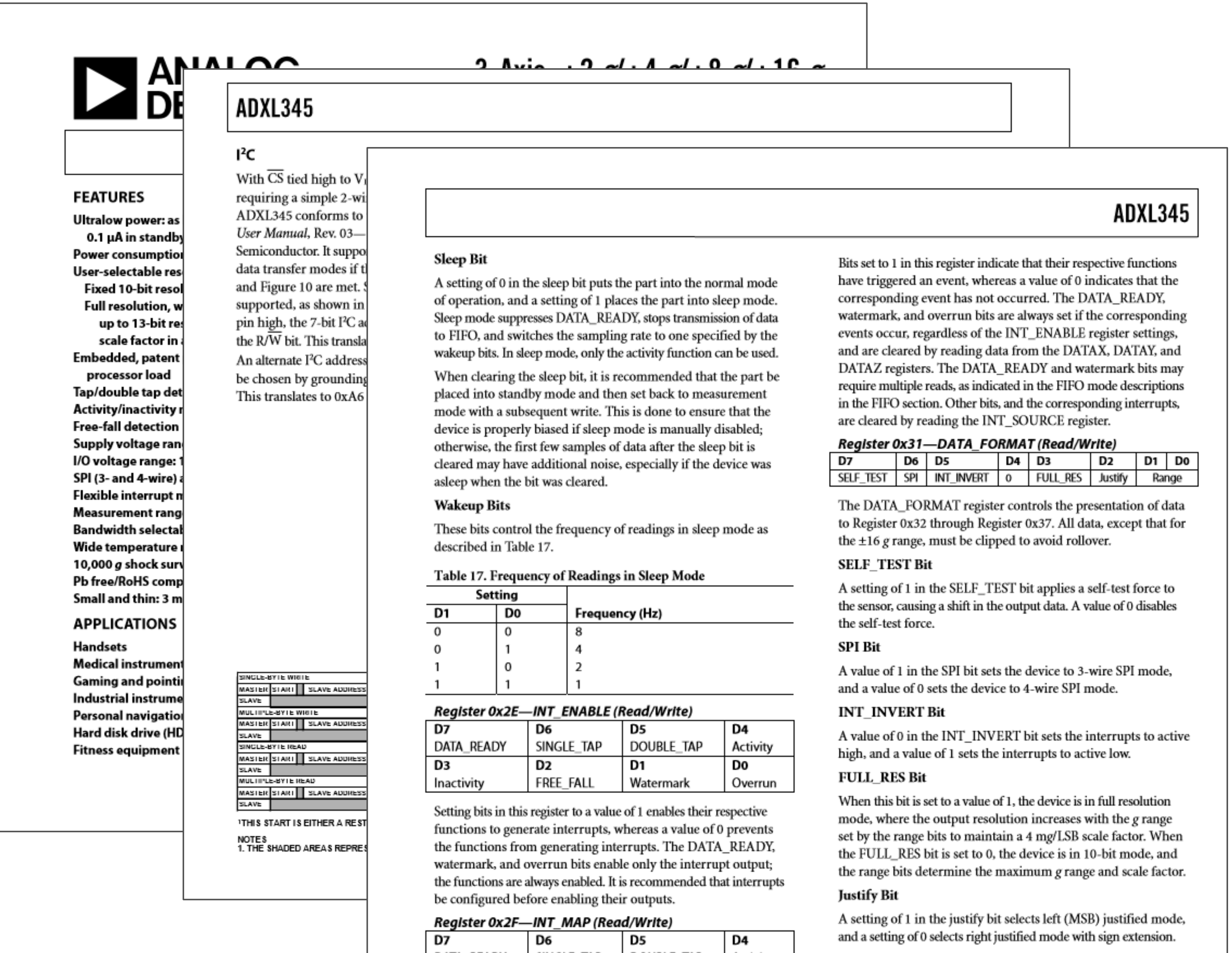
Digital Accelerometer ADXL345

- Manufacturer : ANALOG DEVICES
- Mobile, Medical, Game, Industry device, HDD protection
- Each X, Y, Z axis output data is 16 (2x8) bit
- Accessible through either a SPI or I2C , 400 KHz Fast mode
- Free-fall detection





Everything is in the datasheet



ANALOG DEVICES

3-Axis, ±2 g/±4 g/±8 g/±16 g

FEATURES

Ultralow power: as
0.1 µA in standby
Power consumption
User-selectable res
Fixed 10-bit resol
Full resolution, w
up to 13-bit res
scale factor in
Embedded, patent
processor load
Tap/double tap det
Activity/inactivity
Free-fall detection
Supply voltage ran
I/O voltage range: 1
SPI (3- and 4-wire)
Flexible interrupt m
Measurement rang
Bandwidth selectab
Wide temperature
10,000 g shock surv
Pb free/RoHS comp
Small and thin: 3 m

APPLICATIONS

Handsets
Medical instrument
Gaming and pointi
Industrial instrume
Personal navigatio
Hard disk drive (HD
Fitness equipment

ADXL345

I²C

With $\overline{CS}$ tied high to V
requiring a simple 2-wi
ADXL345 conforms to
User Manual, Rev. 03—
Semiconductor. It suppo
data transfer modes if t
and Figure 10 are met.
supported, as shown in
pin high, the 7-bit I²C a
the R/$\overline{W}$ bit. This transla
An alternate I²C address
be chosen by grounding
This translates to 0xA6

SINGLE-BYTE WRITE
MASTER START SLAVE ADDRESS
SLAVE
MULTIPLE-BYTE WRITE
MASTER START SLAVE ADDRESS
SLAVE
SINGLE-BYTE READ
MASTER START SLAVE ADDRESS
SLAVE
MULTIPLE-BYTE READ
MASTER START SLAVE ADDRESS
SLAVE

¹THIS START IS EITHER A REST

NOTES
1. THE SHADED AREAS REPRES

ADXL345

Sleep Bit

A setting of 0 in the sleep bit puts the part into the normal mode of operation, and a setting of 1 places the part into sleep mode. Sleep mode suppresses DATA_READY, stops transmission of data to FIFO, and switches the sampling rate to one specified by the wakeup bits. In sleep mode, only the activity function can be used.

When clearing the sleep bit, it is recommended that the part be placed into standby mode and then set back to measurement mode with a subsequent write. This is done to ensure that the device is properly biased if sleep mode is manually disabled; otherwise, the first few samples of data after the sleep bit is cleared may have additional noise, especially if the device was asleep when the bit was cleared.

Wakeup Bits

These bits control the frequency of readings in sleep mode as described in Table 17.

Table 17. Frequency of Readings in Sleep Mode

Setting		
D1	D0	Frequency (Hz)
0	0	8
0	1	4
1	0	2
1	1	1

Register 0x2E—INT_ENABLE (Read/Write)

D7 DATA_READY	D6 SINGLE_TAP	D5 DOUBLE_TAP	D4 Activity
D3 Inactivity	D2 FREE_FALL	D1 Watermark	D0 Overrun

Setting bits in this register to a value of 1 enables their respective functions to generate interrupts, whereas a value of 0 prevents the functions from generating interrupts. The DATA_READY, watermark, and overrun bits enable only the interrupt output; the functions are always enabled. It is recommended that interrupts be configured before enabling their outputs.

Register 0x2F—INT_MAP (Read/Write)

D7	D6	D5	D4

Bits set to 1 in this register indicate that their respective functions have triggered an event, whereas a value of 0 indicates that the corresponding event has not occurred. The DATA_READY, watermark, and overrun bits are always set if the corresponding events occur, regardless of the INT_ENABLE register settings, and are cleared by reading data from the DATAX, DATAY, and DATAZ registers. The DATA_READY and watermark bits may require multiple reads, as indicated in the FIFO mode descriptions in the FIFO section. Other bits, and the corresponding interrupts, are cleared by reading the INT_SOURCE register.

Register 0x31—DATA_FORMAT (Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0
SELF_TEST	SPI	INT_INVERT	0	FULL_RES	Justify	Range	

The DATA_FORMAT register controls the presentation of data to Register 0x32 through Register 0x37. All data, except that for the ±16 g range, must be clipped to avoid rollover.

SELF_TEST Bit

A setting of 1 in the SELF_TEST bit applies a self-test force to the sensor, causing a shift in the output data. A value of 0 disables the self-test force.

SPI Bit

A value of 1 in the SPI bit sets the device to 3-wire SPI mode, and a value of 0 sets the device to 4-wire SPI mode.

INT_INVERT Bit

A value of 0 in the INT_INVERT bit sets the interrupts to active high, and a value of 1 sets the interrupts to active low.

FULL_RES Bit

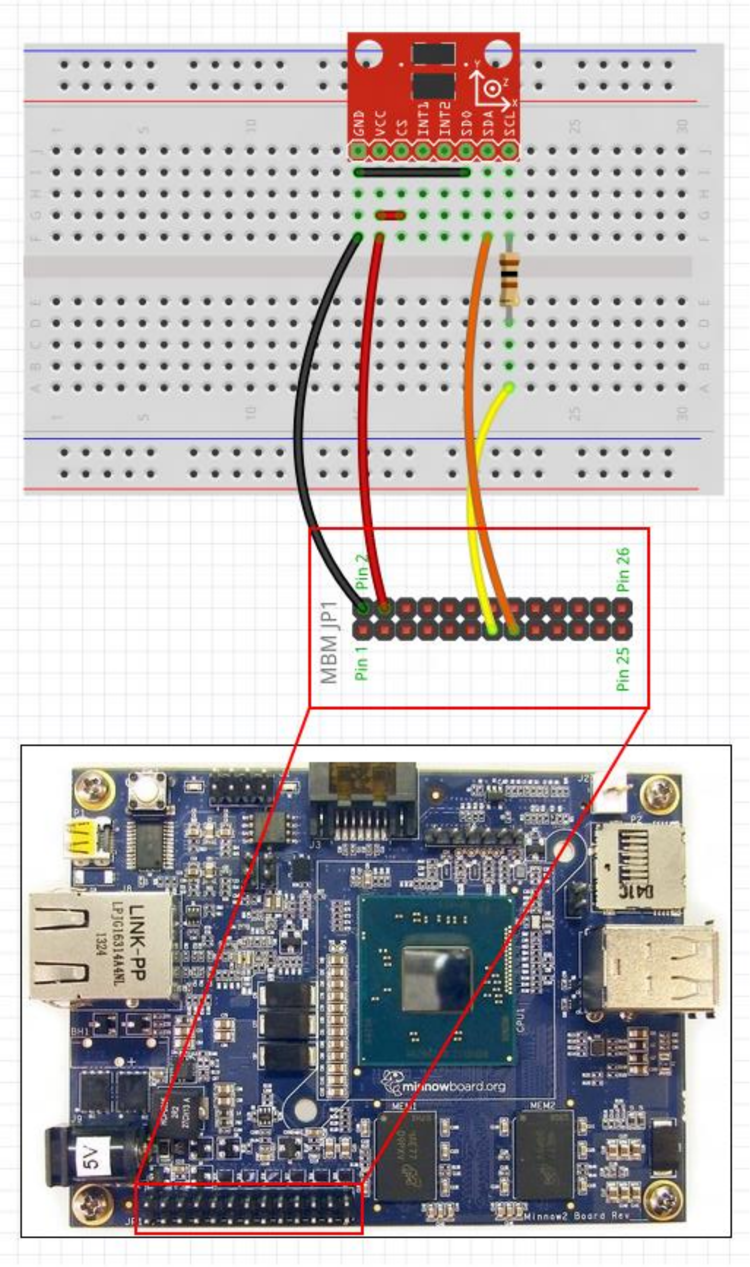
When this bit is set to a value of 1, the device is in full resolution mode, where the output resolution increases with the g range set by the range bits to maintain a 4 mg/LSB scale factor. When the FULL_RES bit is set to 0, the device is in 10-bit mode, and the range bits determine the maximum g range and scale factor.

Justify Bit

A setting of 1 in the justify bit selects left (MSB) justified mode, and a setting of 0 selects right justified mode with sign extension.

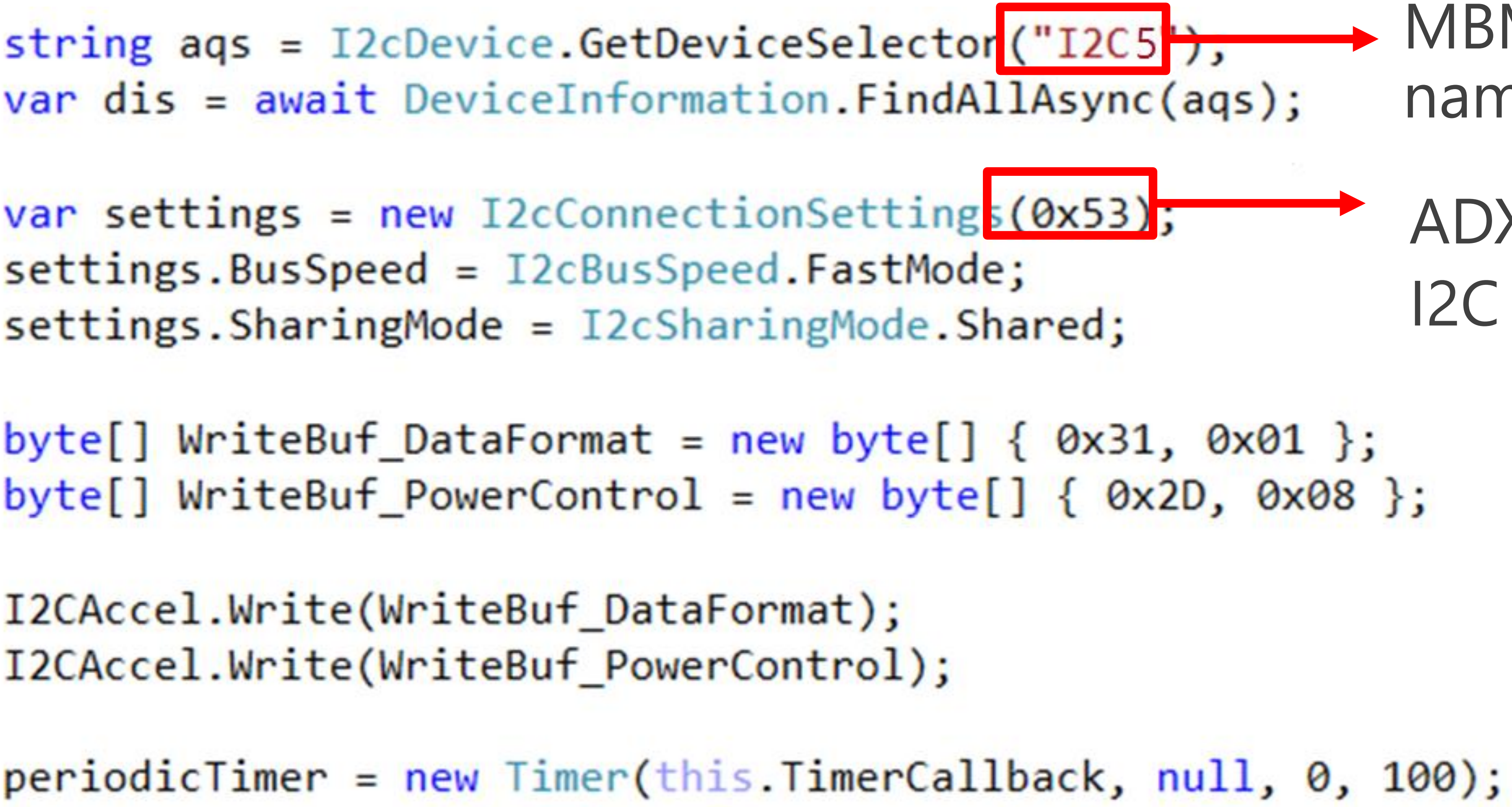
Connect sensor device to MBM

- ADXL345 3V3 → MBM 3.3v(Pin #04)
- ADXL345 CS → MBM 3.3v(Pin #04)
- ADXL345 GND → MBM GND(Pin #02)
- ADXL345 SDA → MBM SDA(Pin #15)
- ADXL345 SCL → MBM SCL(Pin #13)



- SDA, SCL (Pin #13, #15) is defined as "I2C5" in ACPI.
- ADXL345's Slave address is "0x53"

Using Windows.Devices.I2c namespace



```
string aqs = I2cDevice.GetDeviceSelector("I2C5");
var dis = await DeviceInformation.FindAllAsync(aqs);

var settings = new I2cConnectionSettings(0x53);
settings.BusSpeed = I2cBusSpeed.FastMode;
settings.SharingMode = I2cSharingMode.Shared;

byte[] WriteBuf_DataFormat = new byte[] { 0x31, 0x01 };
byte[] WriteBuf_PowerControl = new byte[] { 0x2D, 0x08 };

I2CAccel.Write(WriteBuf_DataFormat);
I2CAccel.Write(WriteBuf_PowerControl);

periodicTimer = new Timer(this.TimerCallback, null, 0, 100);
```

MBM’s I2C interface name

ADXL345
I2C Slave Address

Write Data to ADXL345

s	Slave Address	Wr	A	Register Address	A	Data	A	Stop

I2

bf)

REGISTER MAP

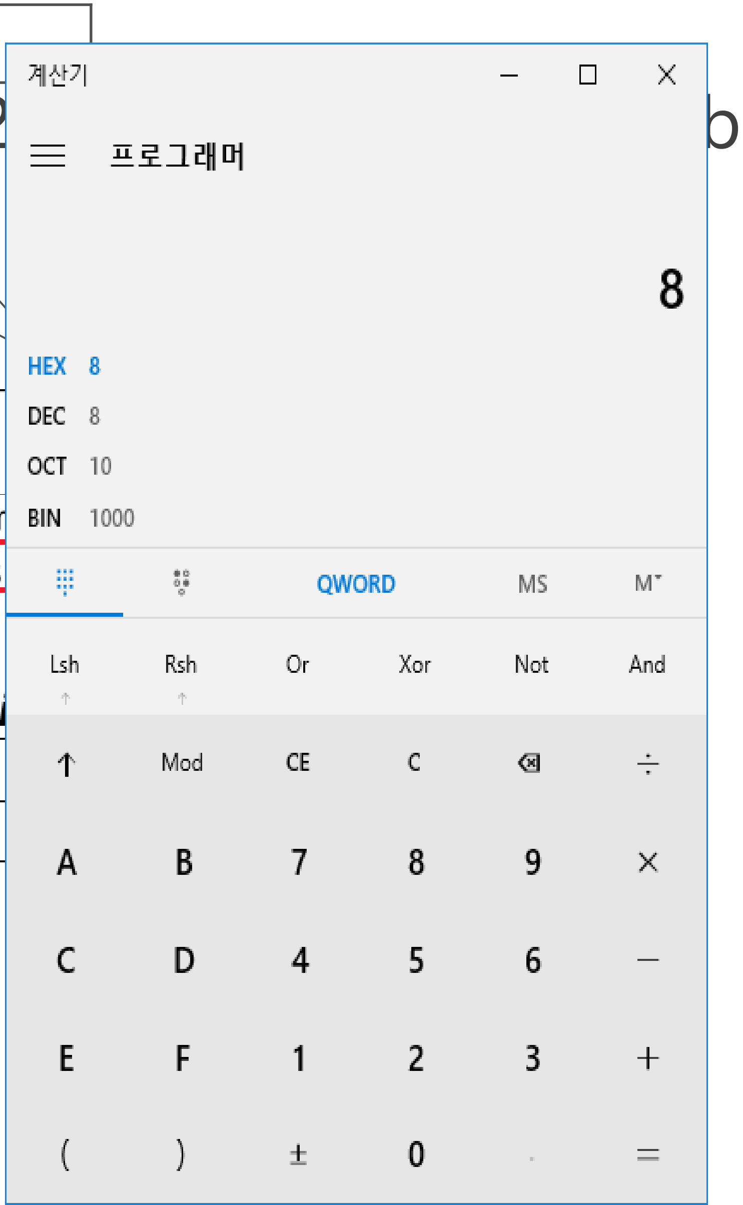
Table 16. Register Map

Address		Name	Type	Reset Value	Description
Hex	Dec				
0x2C	44	BW_RATE	R/W̄	00001010	Data rate and power m
0x2D	45	POWER_CTL	R/W̄	00000000	Power-saving features

Register 0x2D—POWER_CTL (Read/Wri

D7	D6	D5	D4	D3
0	0	Link	AUTO_SLEEP	Measure

```
byte[] WriteBuf_PowerControl = new byte[] { 0x2D,
I2CAccel.Write(WriteBuf_PowerControl);
```



계산기

프로그래머

8

HEX 8

DEC 8

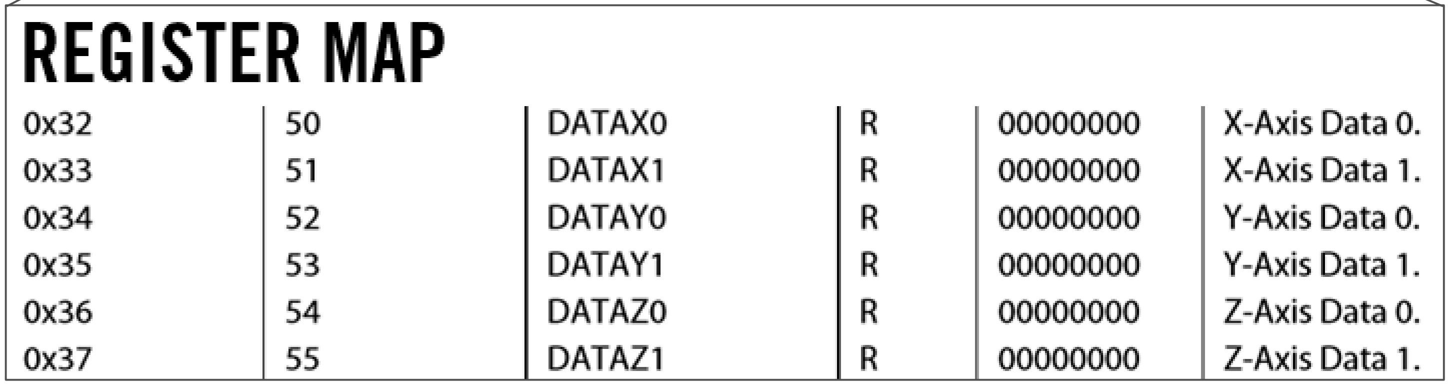
OCT 10

BIN 1000

Read Data from ADXL345

s	Slave Address	Wr	A	Register Address	A	sr	Slave Address	Rd	A	Data	A	Data	A

```
I2cDevice.WriteRead( byte[] wbf, byte[] rbf )
```



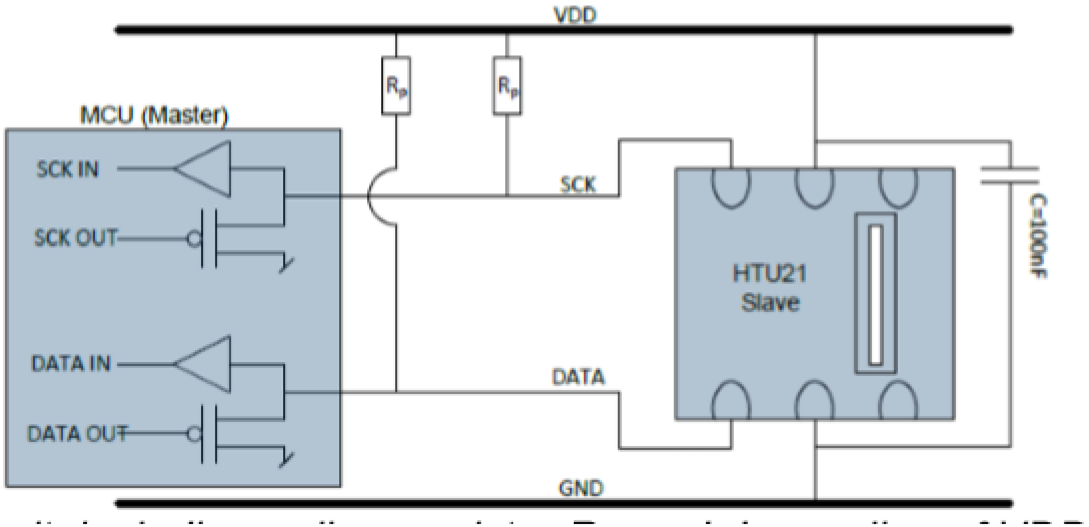
REGISTER MAP

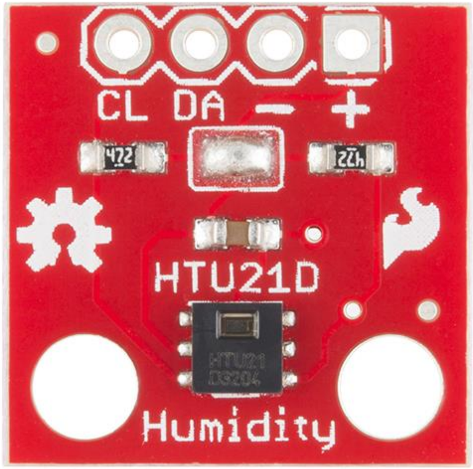
0x32	50	DATAX0	R	00000000	X-Axis Data 0.
0x33	51	DATAX1	R	00000000	X-Axis Data 1.
0x34	52	DATAY0	R	00000000	Y-Axis Data 0.
0x35	53	DATAY1	R	00000000	Y-Axis Data 1.
0x36	54	DATAZ0	R	00000000	Z-Axis Data 0.
0x37	55	DATAZ1	R	00000000	Z-Axis Data 1.

```
byte[] RegAddrBuf = new byte[] { 0x32 };
byte[] ReadBuf = new byte[6];

I2CAccel.WriteRead(RegAddrBuf, ReadBuf);
```

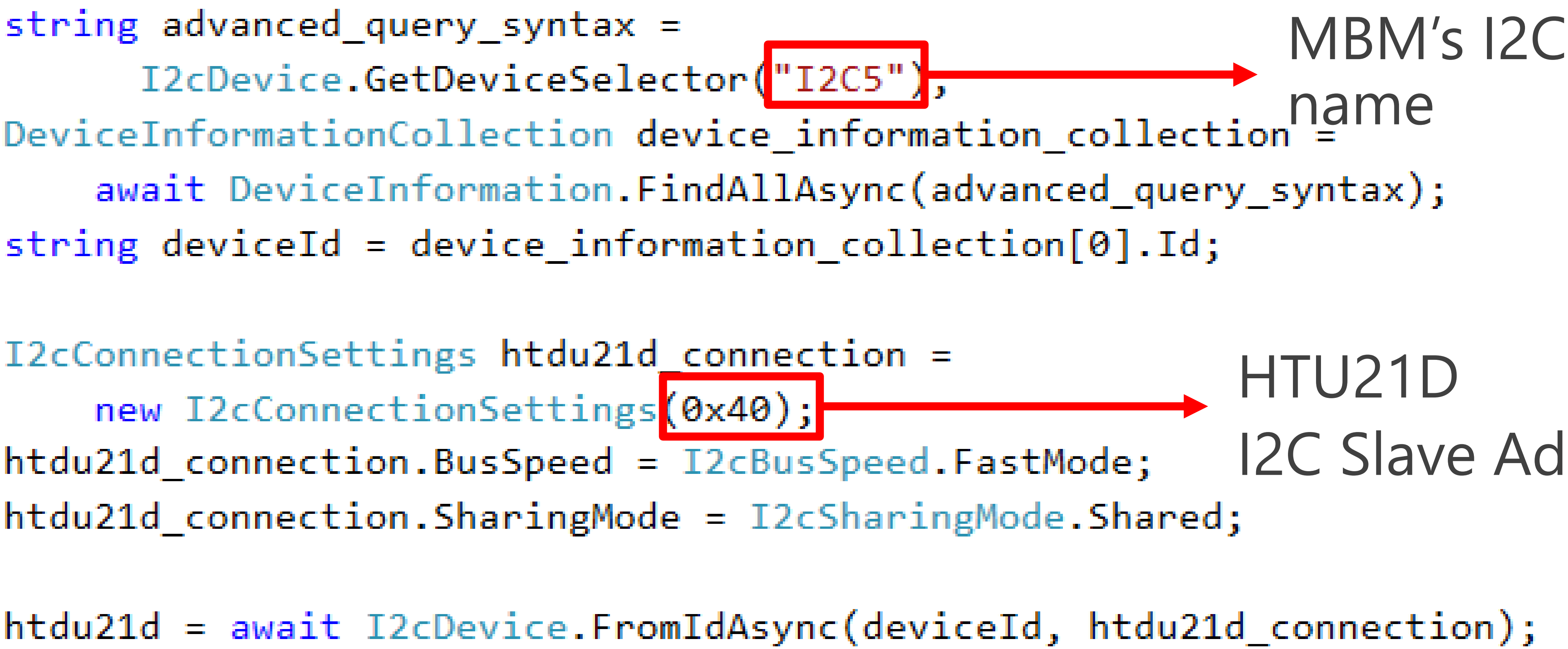
Humidity, Temp sensor HTU21D

- Manufacturer : Measurement
- Automotive, Home Appliance, Medical, Printer.. Etc
- Output data MSB, LSB, Checksum 3x 8 bit
- Accessible through either a I2C , 400 KHz Fast mode
- Hold Master, No hold Mater communication sequence





Using Windows.Devices.I2c namespace



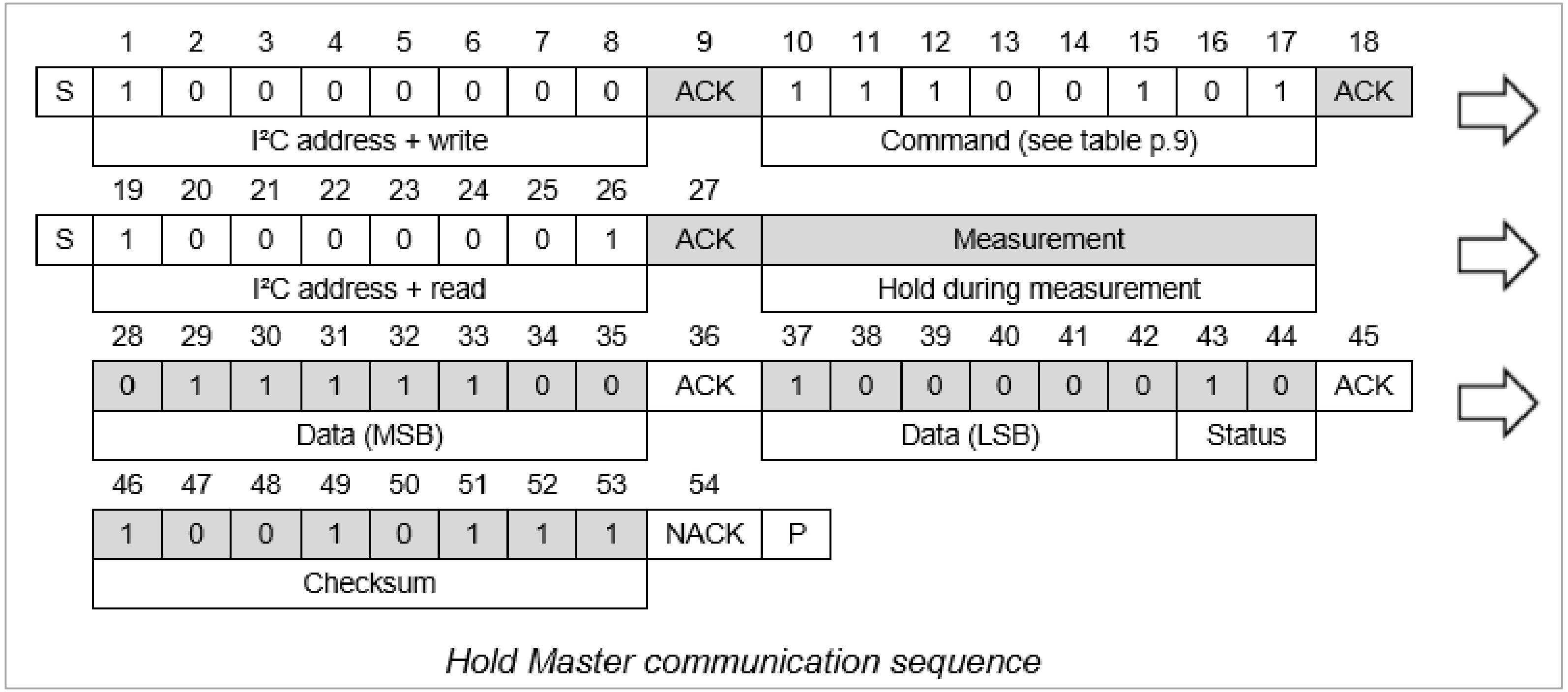
```
string advanced_query_syntax =
      I2cDevice.GetDeviceSelector("I2C5");
DeviceInformationCollection device_information_collection =
    await DeviceInformation.FindAllAsync(advanced_query_syntax);
string deviceId = device_information_collection[0].Id;

I2cConnectionSettings htdu21d_connection =
    new I2cConnectionSettings(0x40);
htdu21d_connection.BusSpeed = I2cBusSpeed.FastMode;
htdu21d_connection.SharingMode = I2cSharingMode.Shared;

htdu21d = await I2cDevice.FromIdAsync(deviceId, htdu21d_connection);
```

MBM’s I2C interface name

HTU21D

I2C Slave Address

Read Temp Data from HTU21D

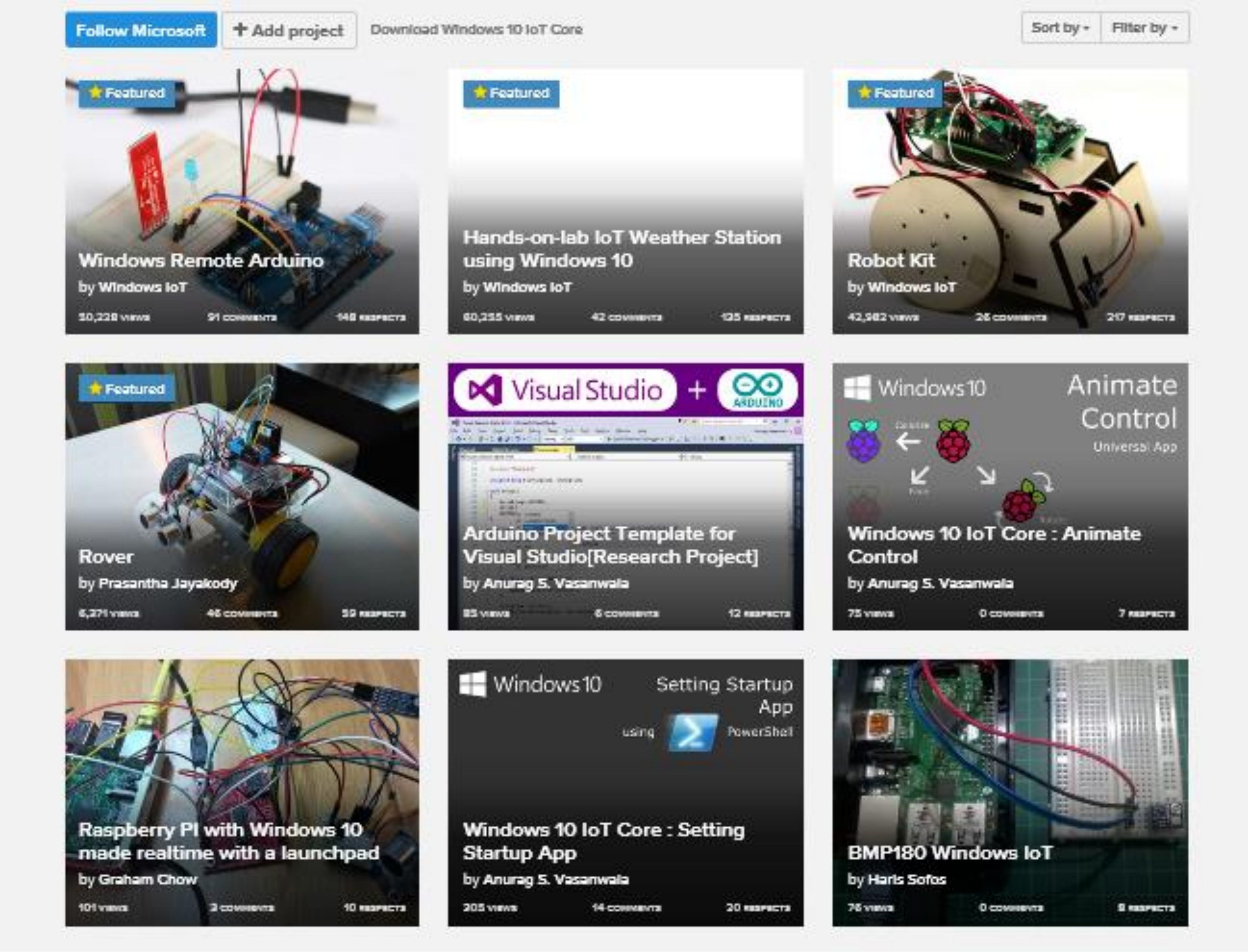


	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
S	1	0	0	0	0	0	0	0	ACK	1	1	1	0	0	1	0	1	ACK
	I²C address + write									Command (see table p.9)								
	19	20	21	22	23	24	25	26	27									
S	1	0	0	0	0	0	0	1	ACK	Measurement								
	I²C address + read									Hold during measurement								
	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
	0	1	1	1	1	1	0	0	ACK	1	0	0	0	0	0	1	0	ACK
	Data (MSB)									Data (LSB)						Status		
	46	47	48	49	50	51	52	53	54									
	1	0	0	1	0	1	1	1	NACK	P								
	Checksum																	

Hold Master communication sequence

```
byte[] RegAddrBuf = new byte[] { 0xE3 };  //Humidity address 0xE5
byte[] ReadBuf = new byte[3];

I2CAccel.WriteRead(RegAddrBuf, ReadBuf);
```
]

To do… WindowsOnDevices.com



Windows Remote Arduino
by Windows IoT
50,228 VIEWS 91 COMMENTS 148 RESPECTS

Hands-on-lab IoT Weather Station using Windows 10
by Windows IoT
60,255 VIEWS 42 COMMENTS 125 RESPECTS

Robot Kit
by Windows IoT
42,982 VIEWS 26 COMMENTS 217 RESPECTS

Rover
by Prasantha Jayakody
6,271 VIEWS 46 COMMENTS 59 RESPECTS

Arduino Project Template for Visual Studio[Research Project]
by Anurag S. Vasanwala
85 VIEWS 6 COMMENTS 12 RESPECTS

Windows 10 IoT Core : Animate Control
by Anurag S. Vasanwala
75 VIEWS 0 COMMENTS 7 RESPECTS

Raspberry PI with Windows 10 made realtime with a launchpad
by Graham Chow
101 VIEWS 2 COMMENTS 10 RESPECTS

Windows 10 IoT Core : Setting Startup App
by Anurag S. Vasanwala
205 VIEWS 14 COMMENTS 20 RESPECTS

BMP180 Windows IoT
by Haris Sofos
76 VIEWS 0 COMMENTS 8 RESPECTS

HOL 2-2 I2C Control

Enhanced UWP

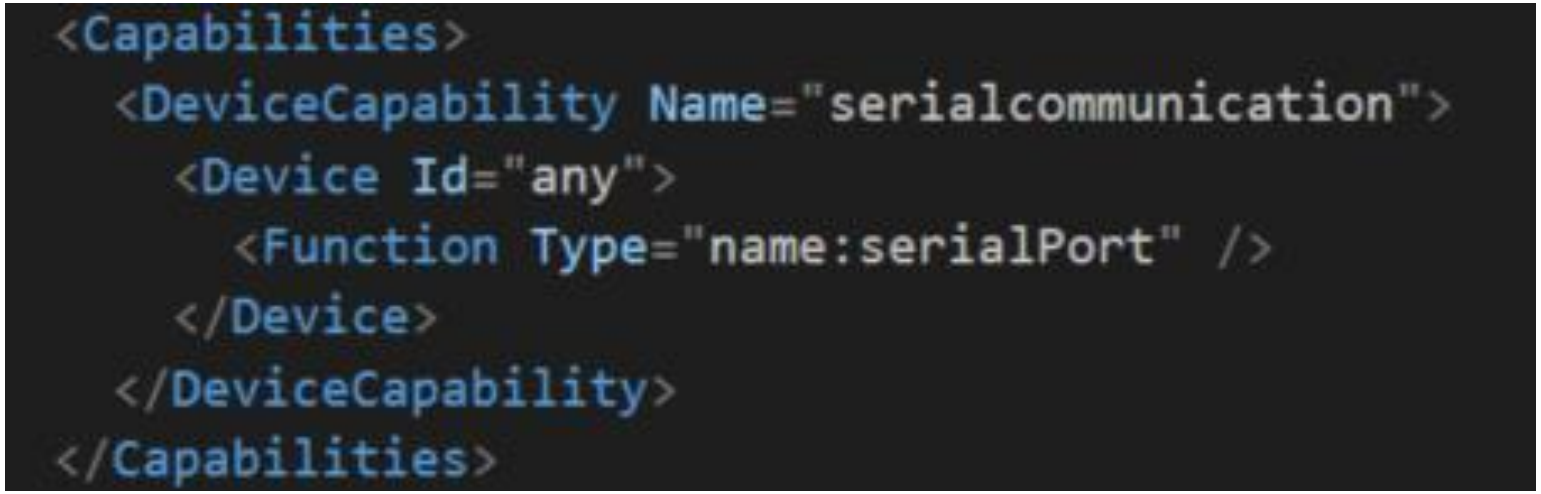
- Serial UART
- UWP Launcher
- Embedded function
- Legacy code
- Open Source Library
- Xamarin

Serial UART

UWP can support Serial UART

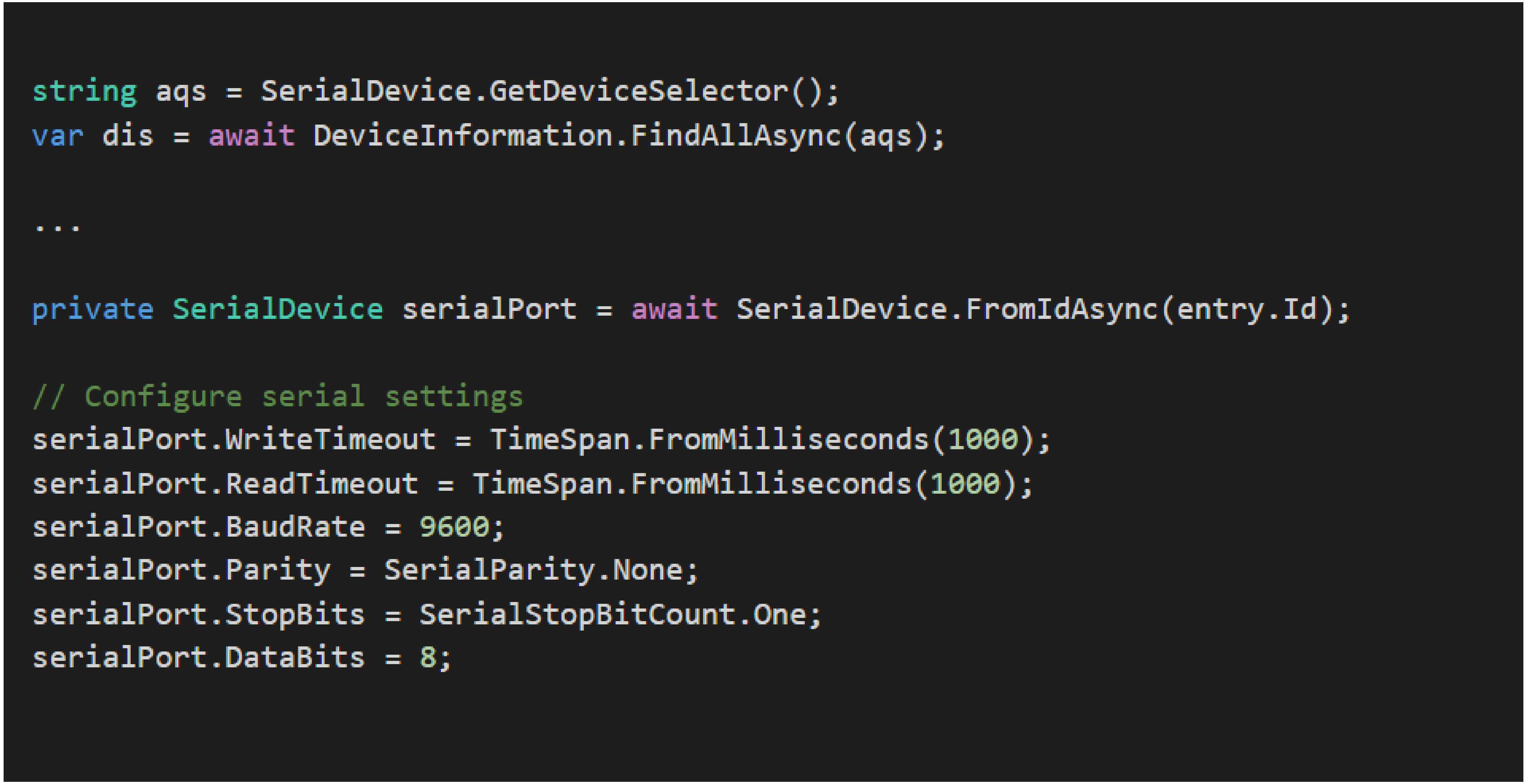
Windows.Devices.SerialCommunication

Serial communication define in Manifest



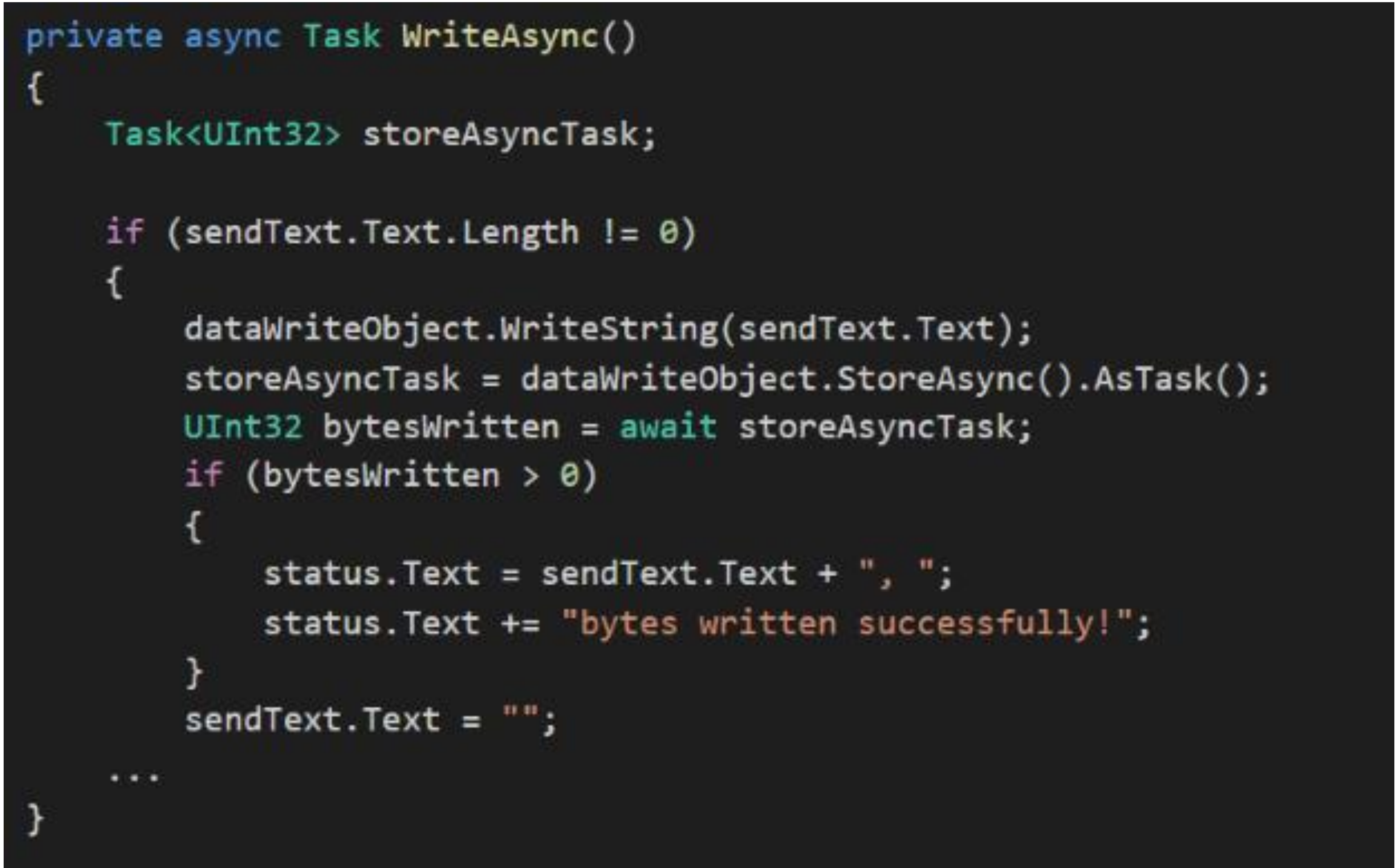
```
<Capabilities>
  <DeviceCapability Name="serialcommunication">
    <Device Id="any">
      <Function Type="name:serialPort" />
    </Device>
  </DeviceCapability>
</Capabilities>
```

Connect to selected serial device



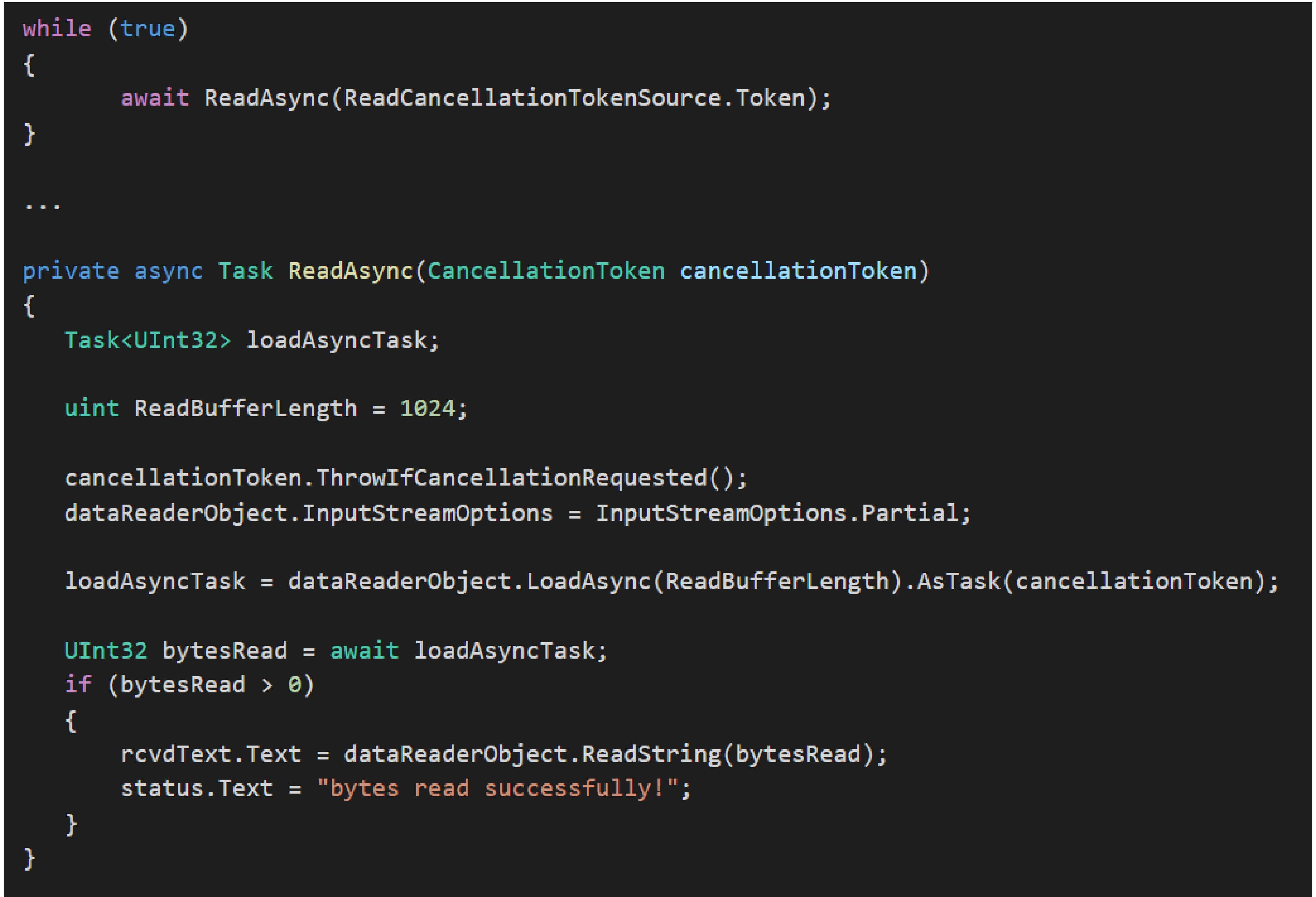
```
string aqs = SerialDevice.GetDeviceSelector();
var dis = await DeviceInformation.FindAllAsync(aqs);

...

private SerialDevice serialPort = await SerialDevice.FromIdAsync(entry.Id);

// Configure serial settings
serialPort.WriteTimeout = TimeSpan.FromMilliseconds(1000);
serialPort.ReadTimeout = TimeSpan.FromMilliseconds(1000);
serialPort.BaudRate = 9600;
serialPort.Parity = SerialParity.None;
serialPort.StopBits = SerialStopBitCount.One;
serialPort.DataBits = 8;
```

Write Data



```
private async Task WriteAsync()
{
    Task<UInt32> storeAsyncTask;

    if (sendText.Text.Length != 0)
    {
        dataWriteObject.WriteString(sendText.Text);
        storeAsyncTask = dataWriteObject.StoreAsync().AsTask();
        UInt32 bytesWritten = await storeAsyncTask;
        if (bytesWritten > 0)
        {
            status.Text = sendText.Text + ", ";
            status.Text += "bytes written successfully!";
        }
        sendText.Text = "";
    ...
}
```

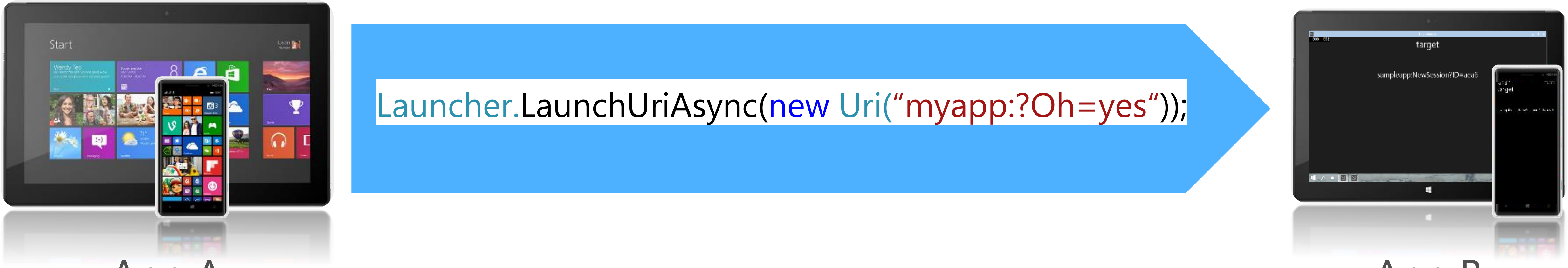
Read Data



```
while (true)
{
       await ReadAsync(ReadCancellationTokenSource.Token);
}

...

private async Task ReadAsync(CancellationToken cancellationToken)
{
   Task<UInt32> loadAsyncTask;

   uint ReadBufferLength = 1024;

   cancellationToken.ThrowIfCancellationRequested();
   dataReaderObject.InputStreamOptions = InputStreamOptions.Partial;

   loadAsyncTask = dataReaderObject.LoadAsync(ReadBufferLength).AsTask(cancellationToken);

   UInt32 bytesRead = await loadAsyncTask;
   if (bytesRead > 0)
   {
       rcvdText.Text = dataReaderObject.ReadString(bytesRead);
       status.Text = "bytes read successfully!";
   }
}
```

Launching UWP

LaunchUri



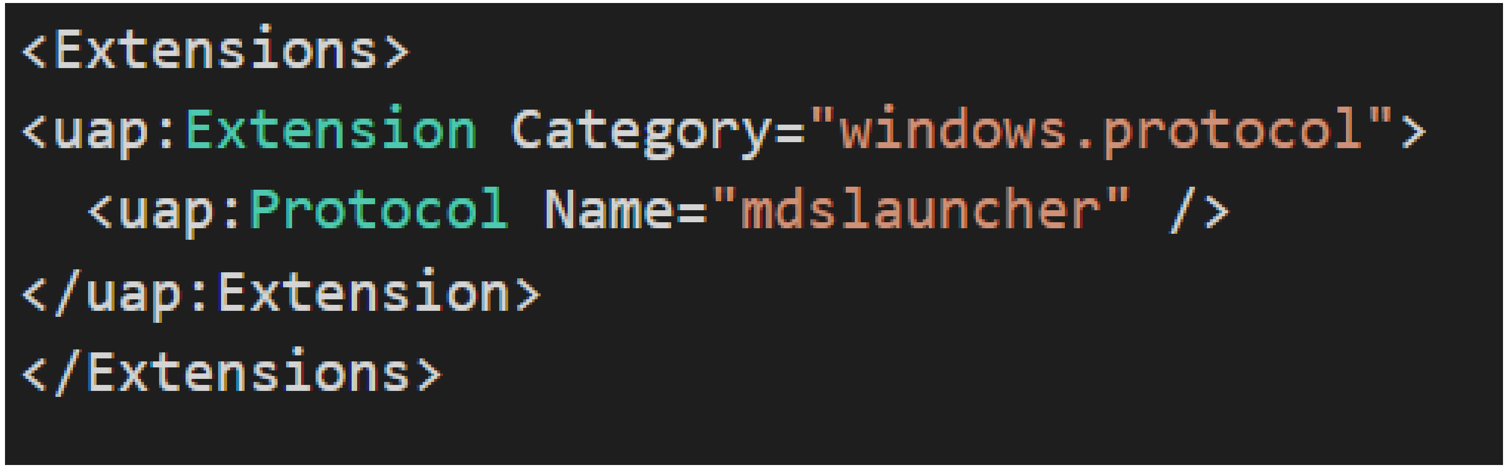
App A

```
Launcher.LaunchUriAsync(new Uri(“myapp:?Oh=yes“));
```

App B

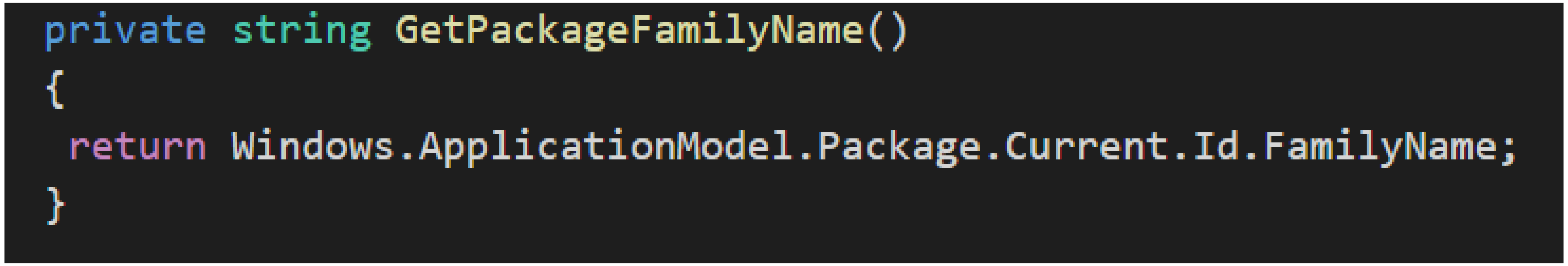
Launching UWP

Protocol defined in Manifest



```
<Extensions>
<uap:Extension Category="windows.protocol">
  <uap:Protocol Name="mdslauncher" />
</uap:Extension>
</Extensions>
```

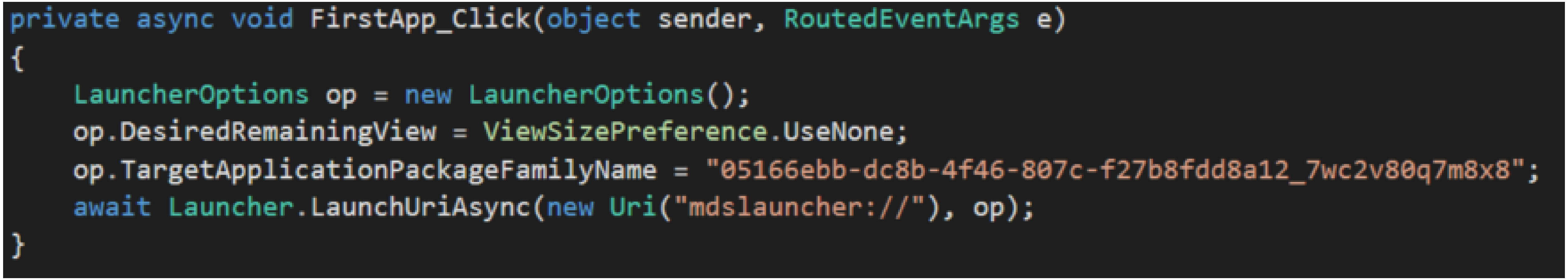
App's Package FamilyName



```
private string GetPackageFamilyName()
{
 return Windows.ApplicationModel.Package.Current.Id.FamilyName;
}
```

Launching UWP

LaunchUriAsync Usage



```
private async void FirstApp_Click(object sender, RoutedEventArgs e)
{
    LauncherOptions op = new LauncherOptions();
    op.DesiredRemainingView = ViewSizePreference.UseNone;
    op.TargetApplicationPackageFamilyName = "05166ebb-dc8b-4f46-807c-f27b8fdd8a12_7wc2v80q7m8x8";
    await Launcher.LaunchUriAsync(new Uri("mdslauncher://"), op);
}
```

Embedded Mode

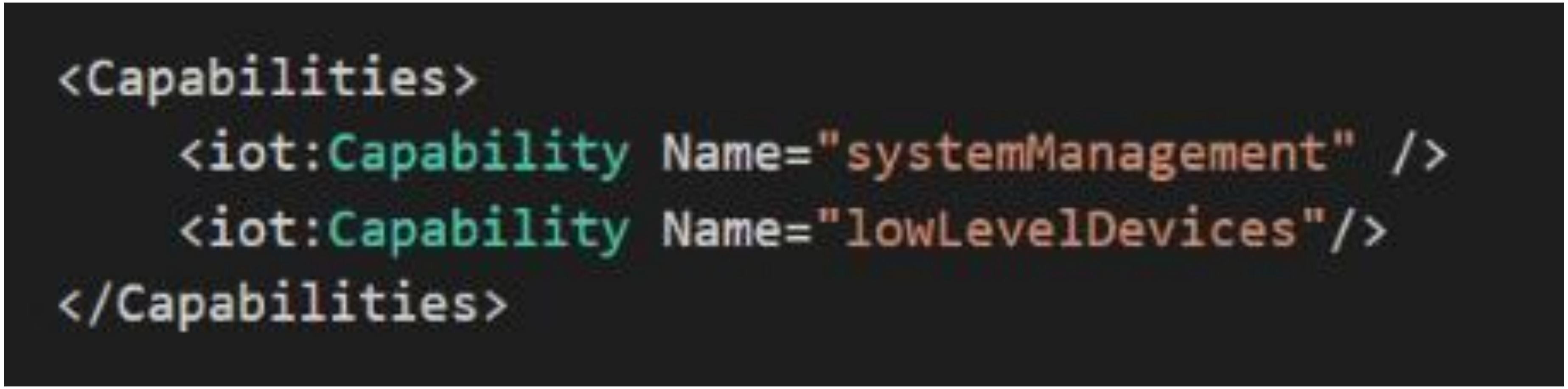
Embedded Mode enables…

- Background Applications
- LowLevelDevice Capability
- SystemManagement Capability
  - Windows.System.ProcessLauncher
  - Windows.System.TimeZoneSettings
  - Windows.System.ShutdownManager
  - Windows.Globalization.Language.TrySetInputMethodLanguageTag
  - AllJoyn loopback

External Process Launcher

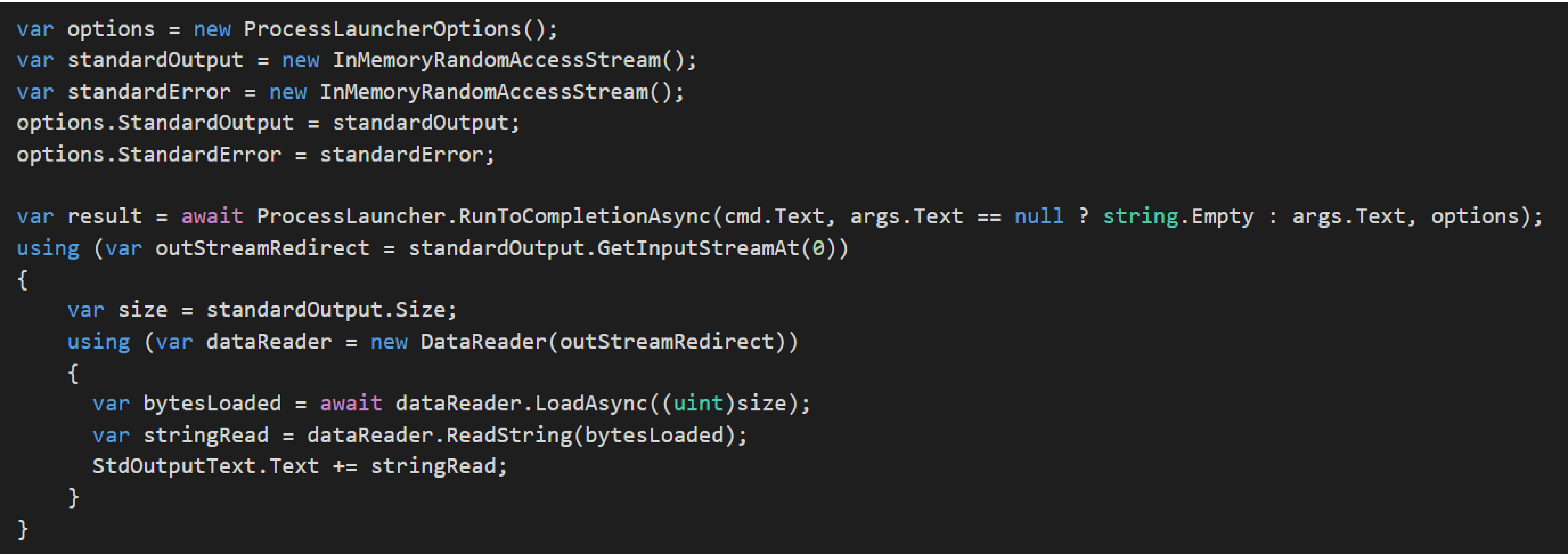
IoT capability enabled in Manifest

Windows.System.ProcessLauncher

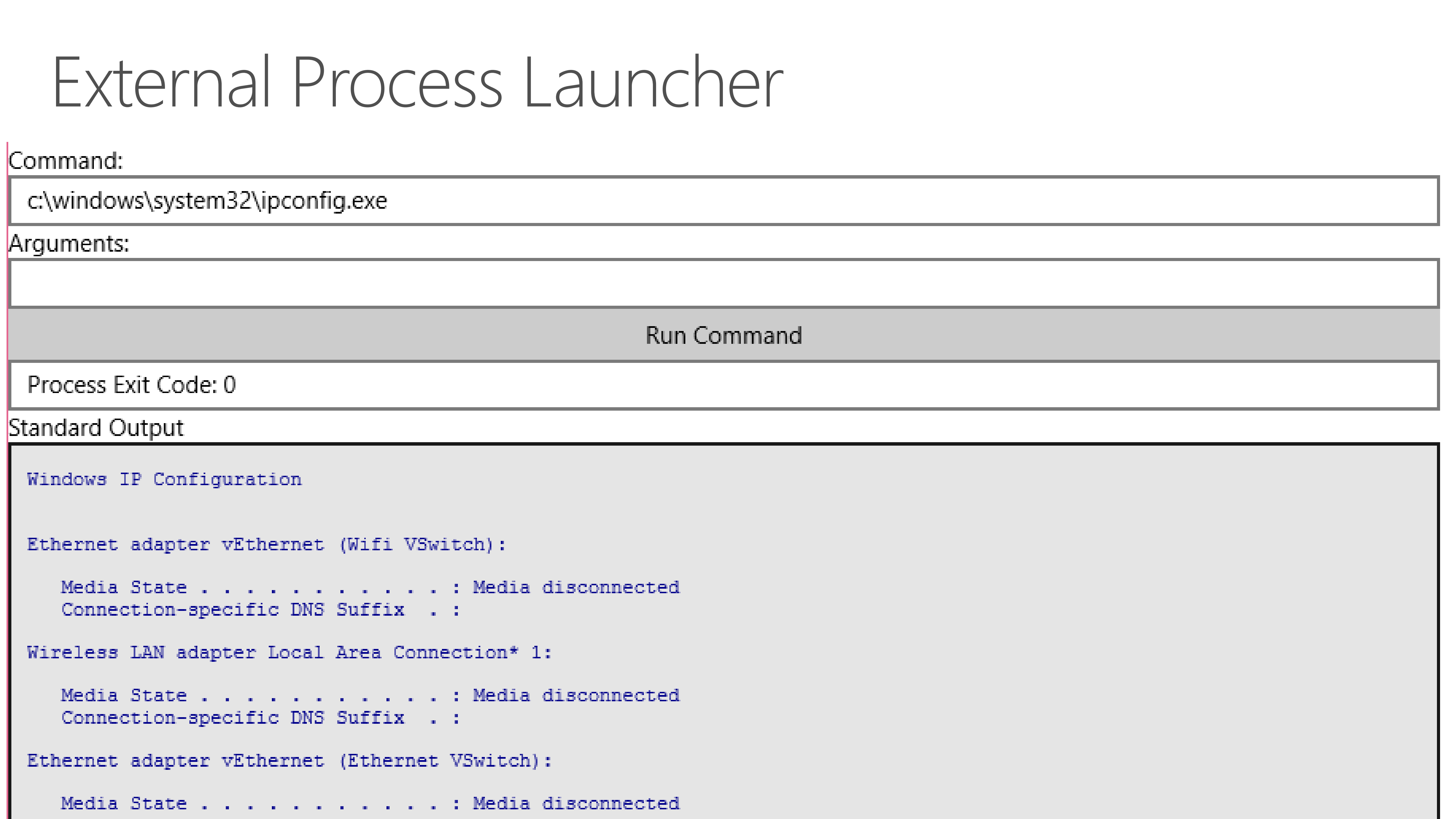


```
<Capabilities>
    <iot:Capability Name="systemManagement" />
    <iot:Capability Name="lowLevelDevices"/>
</Capabilities>
```

External Process Launcher



```
var options = new ProcessLauncherOptions();
var standardOutput = new InMemoryRandomAccessStream();
var standardError = new InMemoryRandomAccessStream();
options.StandardOutput = standardOutput;
options.StandardError = standardError;

var result = await ProcessLauncher.RunToCompletionAsync(cmd.Text, args.Text == null ? string.Empty : args.Text, options);
using (var outStreamRedirect = standardOutput.GetInputStreamAt(0))
{
    var size = standardOutput.Size;
    using (var dataReader = new DataReader(outStreamRedirect))
    {
      var bytesLoaded = await dataReader.LoadAsync((uint)size);
      var stringRead = dataReader.ReadString(bytesLoaded);
      StdOutputText.Text += stringRead;
    }
}
```



External Process Launcher

Command:

c:\windows\system32\ipconfig.exe

Arguments:

Process Exit Code: 0

Standard Output

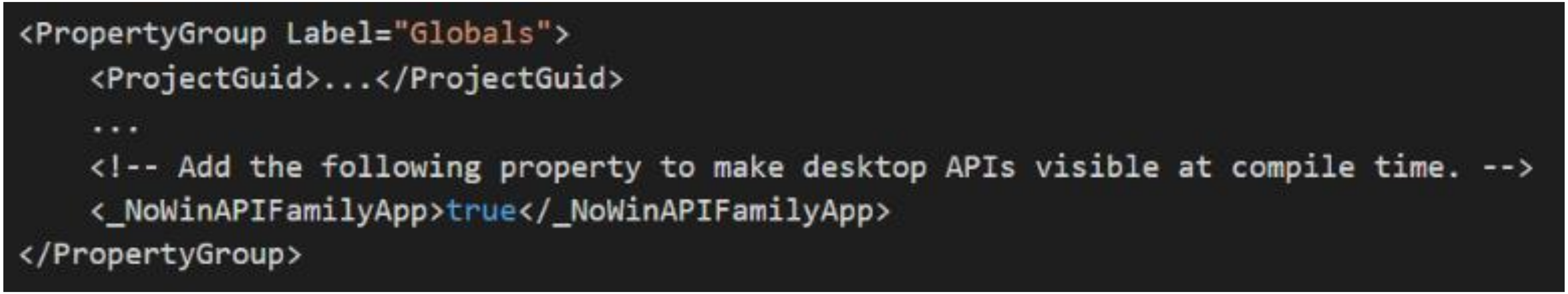
```

Windows IP Configuration


Ethernet adapter vEthernet (Wifi VSwitch):

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Local Area Connection* 1:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Ethernet adapter vEthernet (Ethernet VSwitch):

   Media State . . . . . . . . . . . : Media disconnected
```

CreateFile/DeviceIoControl

OneCoreUap.lib

Provides subset of Win32 APIs

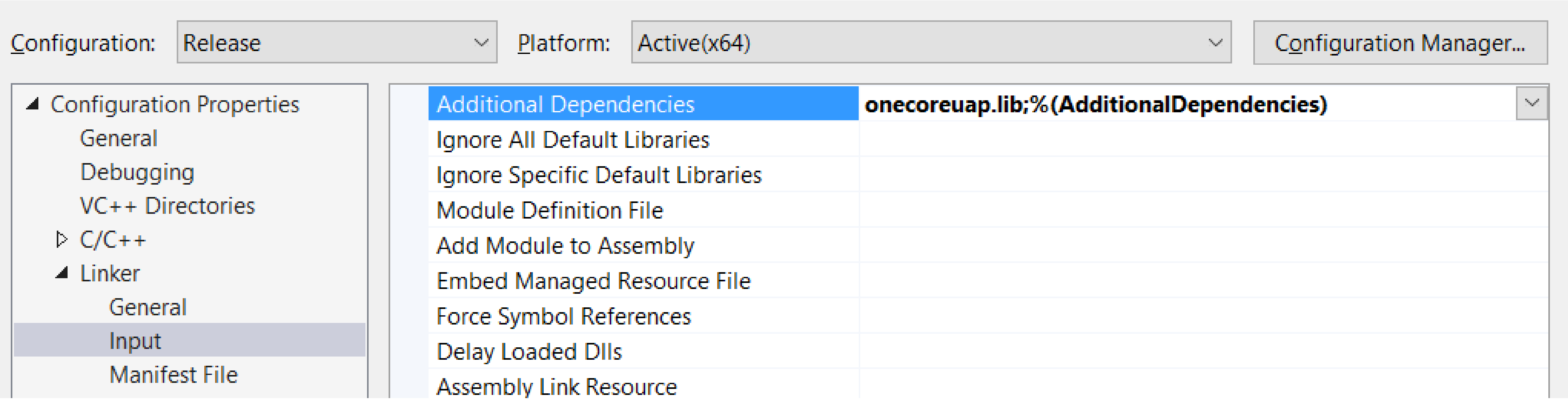
Add <_NoWinAPIFamilyAPP> in .vcxproj file



```
<PropertyGroup Label="Globals">
    <ProjectGuid>...</ProjectGuid>
    ...
    <!-- Add the following property to make desktop APIs visible at compile time. -->
    <_NoWinAPIFamilyApp>true</_NoWinAPIFamilyApp>
</PropertyGroup>
```

CreateFile/DeviceIoControl

Link onecoreuap library and friends

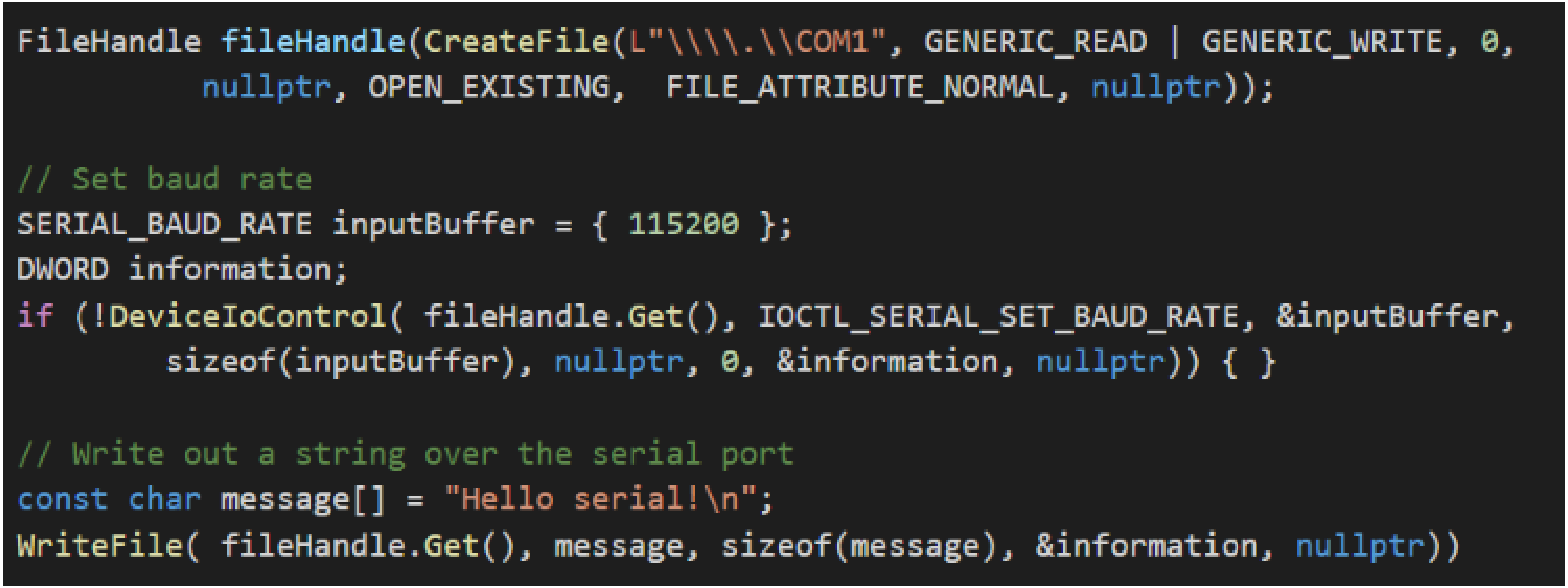


Configuration: Release

Platform: Active(x64)

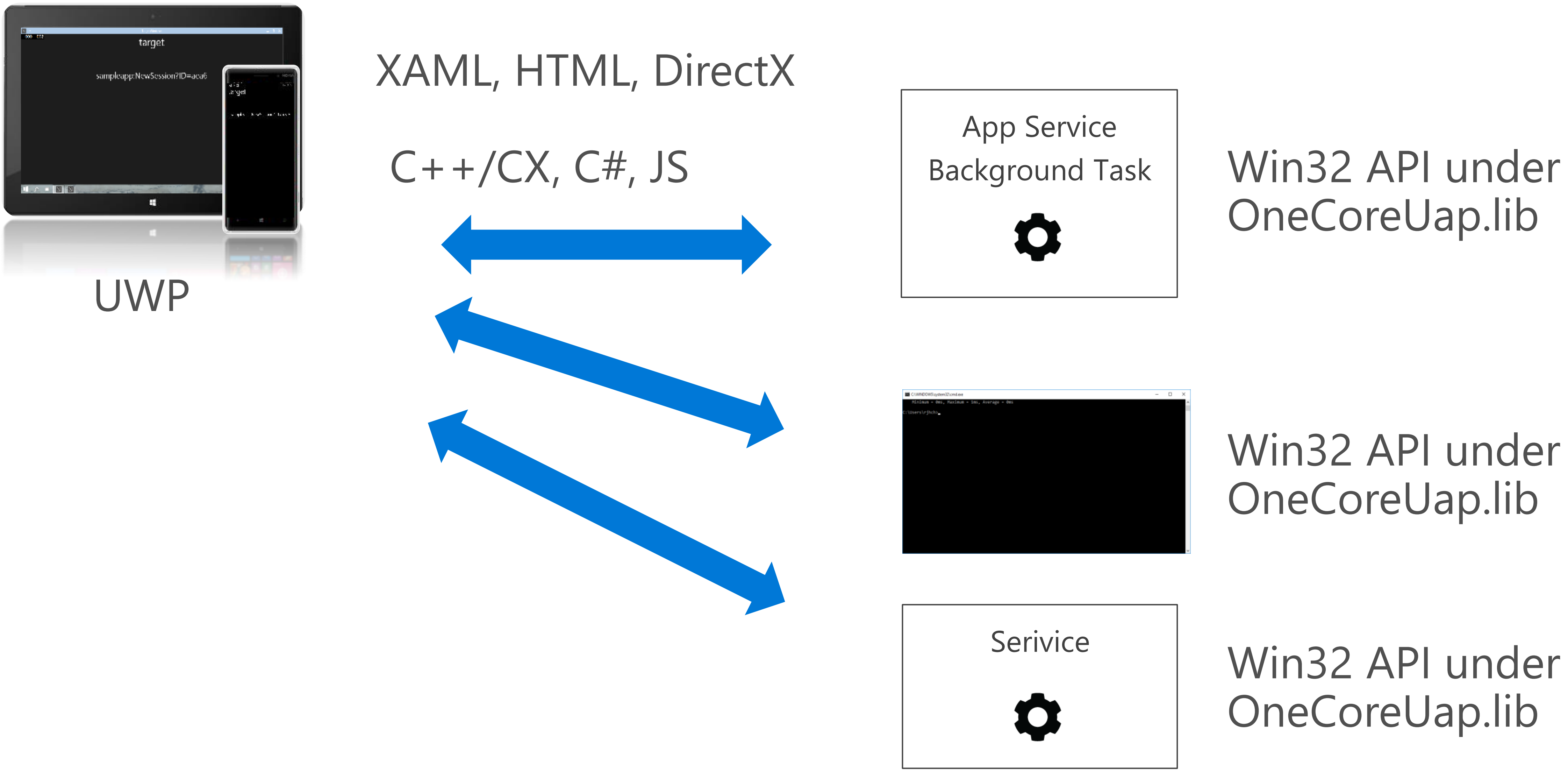
Additional Dependencies: onecoreuap.lib;%(AdditionalDependencies)

CreateFile/DeviceIoControl



```
FileHandle fileHandle(CreateFile(L"\\\\.\\COM1", GENERIC_READ | GENERIC_WRITE, 0,
          nullptr, OPEN_EXISTING,  FILE_ATTRIBUTE_NORMAL, nullptr));

// Set baud rate
SERIAL_BAUD_RATE inputBuffer = { 115200 };
DWORD information;
if (!DeviceIoControl( fileHandle.Get(), IOCTL_SERIAL_SET_BAUD_RATE, &inputBuffer,
        sizeof(inputBuffer), nullptr, 0, &information, nullptr)) { }

// Write out a string over the serial port
const char message[] = "Hello serial!\n";
WriteFile( fileHandle.Get(), message, sizeof(message), &information, nullptr))
```

Using existing source in UWP



UWP

XAML, HTML, DirectX

C++/CX, C#, JS

App Service
Background Task

Win32 API under
OneCoreUap.lib

C:\WINDOWS\system32\cmd.exe

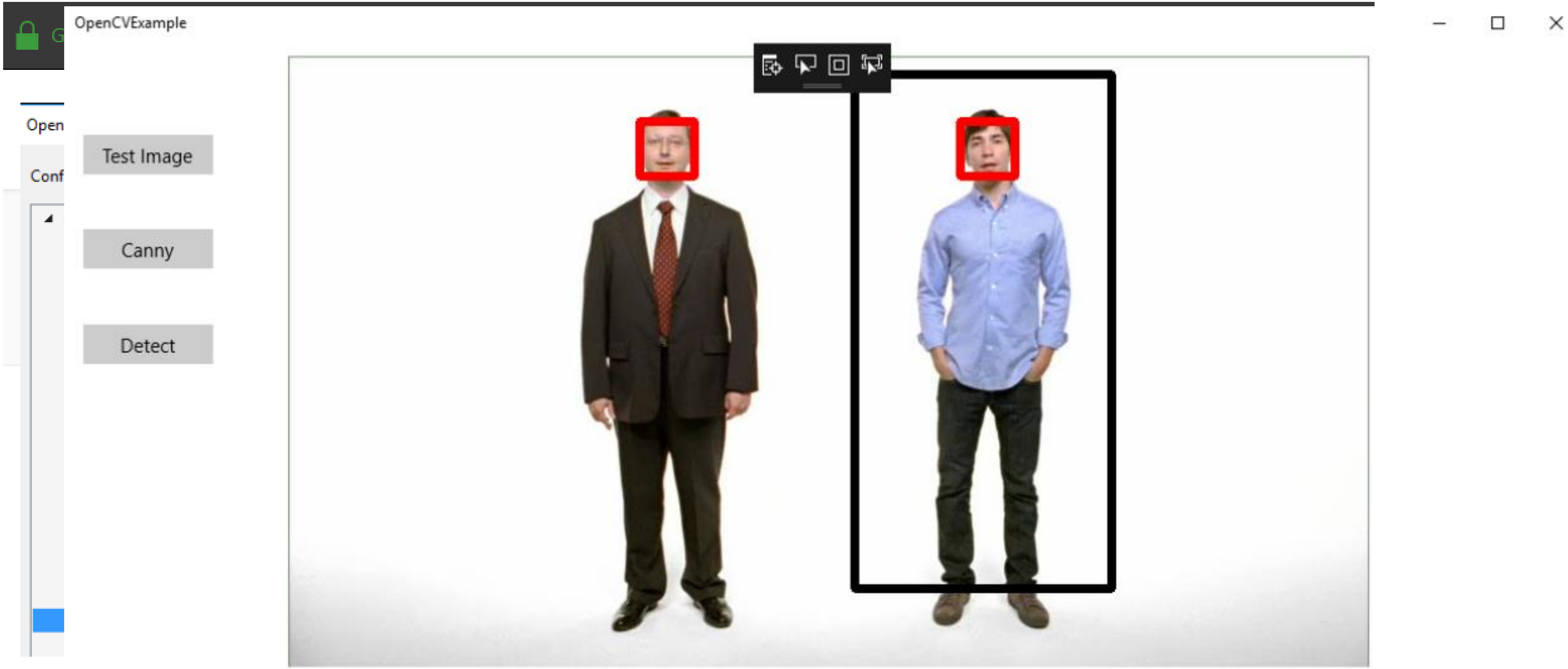
```
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

C:\Users\rjhch>
```

Win32 API under
OneCoreUap.lib

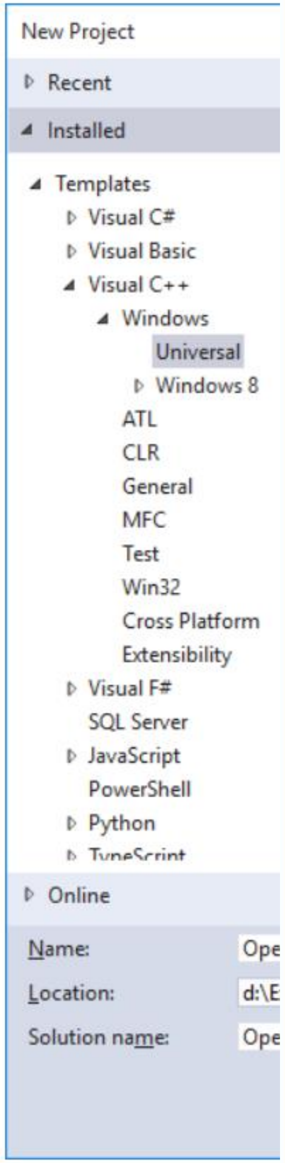
Serivice

Win32 API under
OneCoreUap.lib

OpenCV



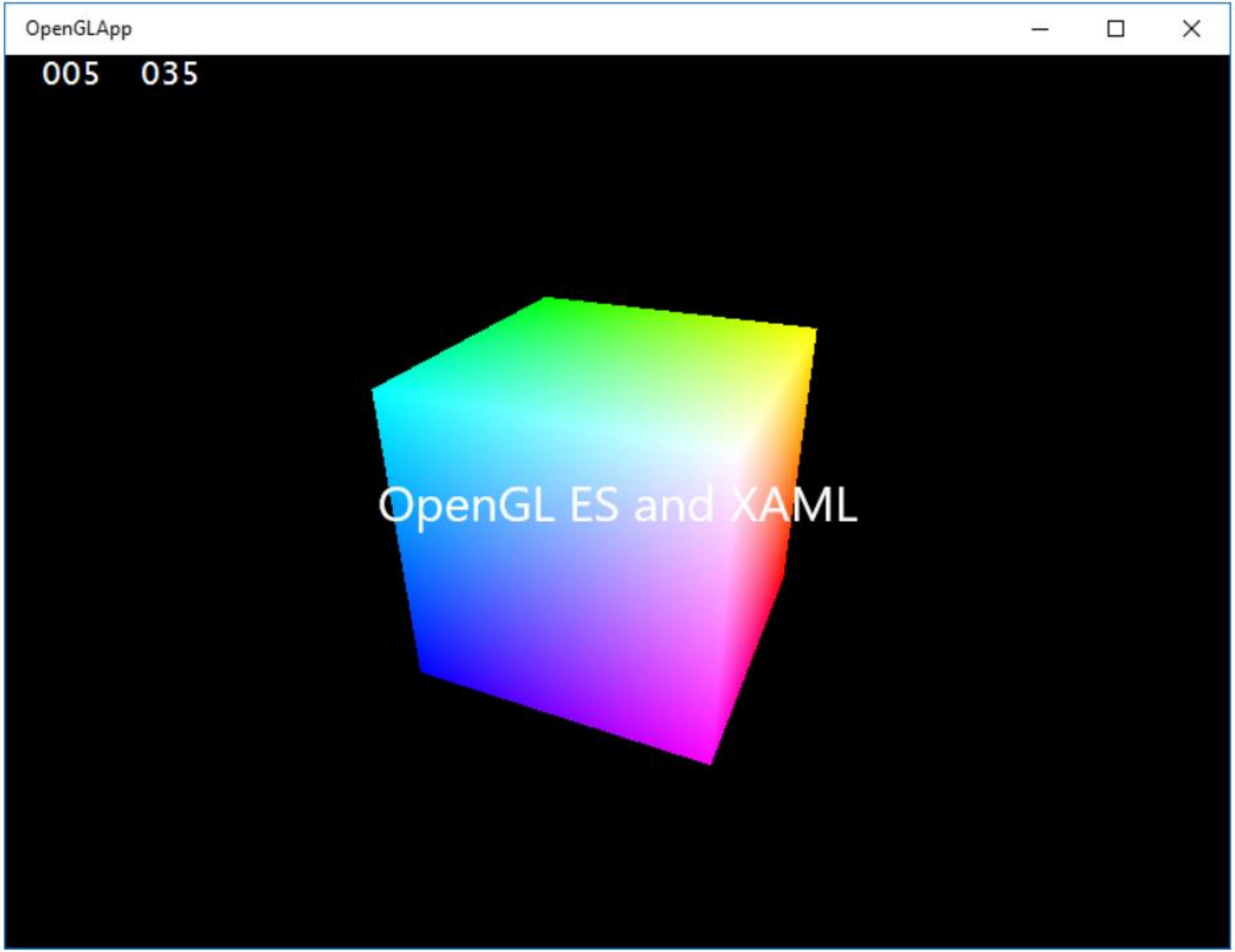
OpenGL



Name: Ope

Location: d:\E

Solution name: Ope

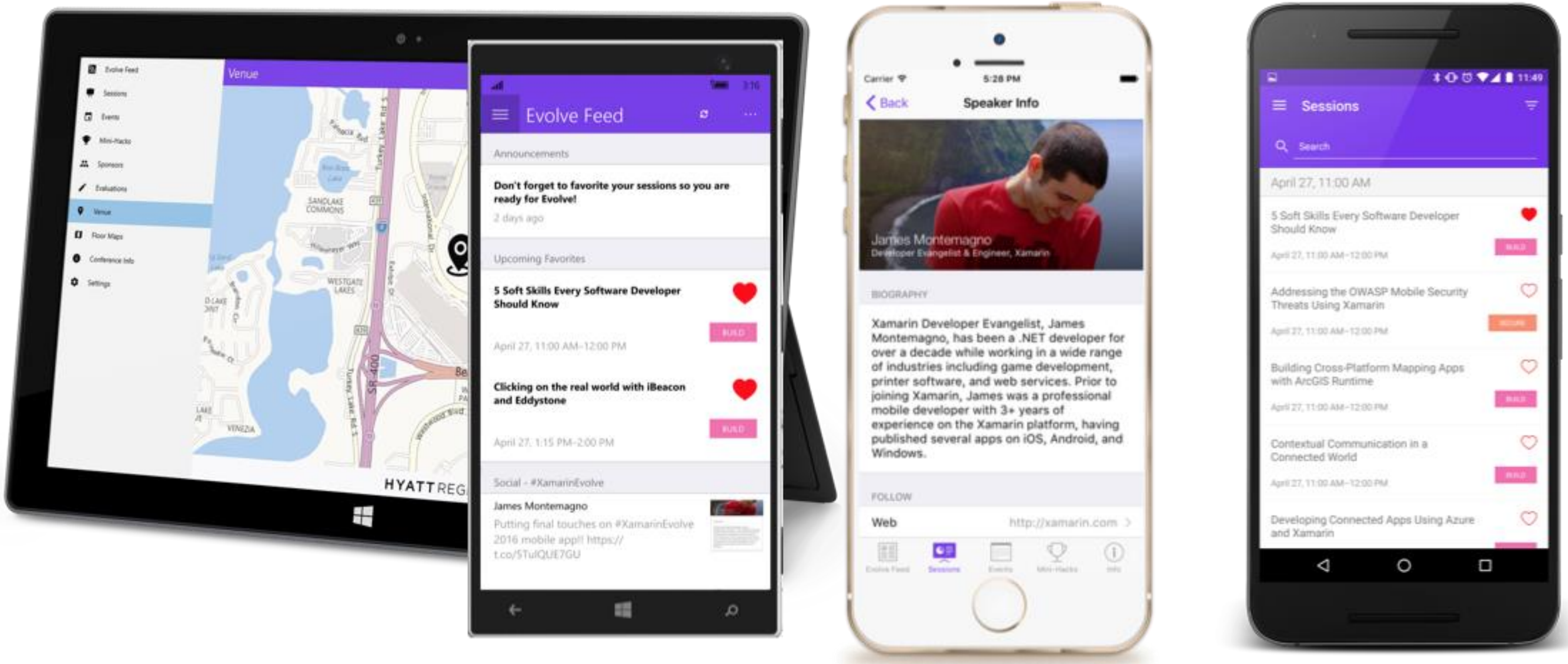


```
005   035
```

OpenGL ES and XAML

Xamarin for Visual Studio

Build Native Android, iOS & Window Apps in C#



Evolve Feed

Announcements

Don't forget to favorite your sessions so you are ready for Evolve!

2 days ago

Upcoming Favorites

5 Soft Skills Every Software Developer Should Know

BUILD

April 27, 11:00 AM–12:00 PM

Clicking on the real world with iBeacon and Eddystone

BUILD

April 27, 1:15 PM–2:00 PM

Social - #XamarinEvolve

James Montemagno

Putting final touches on #XamarinEvolve 2016 mobile app!! https://t.co/5TuIQUE7GU

Speaker Info

James Montemagno
Developer Evangelist & Engineer, Xamarin

BIOGRAPHY

Xamarin Developer Evangelist, James Montemagno, has been a .NET developer for over a decade while working in a wide range of industries including game development, printer software, and web services. Prior to joining Xamarin, James was a professional mobile developer with 3+ years of experience on the Xamarin platform, having published several apps on iOS, Android, and Windows.

FOLLOW

Web http://xamarin.com

Sessions

April 27, 11:00 AM

5 Soft Skills Every Software Developer Should Know

April 27, 11:00 AM–12:00 PM

Addressing the OWASP Mobile Security Threats Using Xamarin

April 27, 11:00 AM–12:00 PM

Building Cross-Platform Mapping Apps with ArcGIS Runtime

April 27, 11:00 AM–12:00 PM

Contextual Communication in a Connected World

April 27, 11:00 AM–12:00 PM

Developing Connected Apps Using Azure and Xamarin