Da der automatischen Anpassung auch Grenzen gesetzt sind, steht Ihnen die Möglichkeit zur Verfügung, gezielt gegen definierte Display-Größen zu entwickeln. Dabei werden die verfügbaren Auflösungen mitberücksichtigt (Bild 5.6).

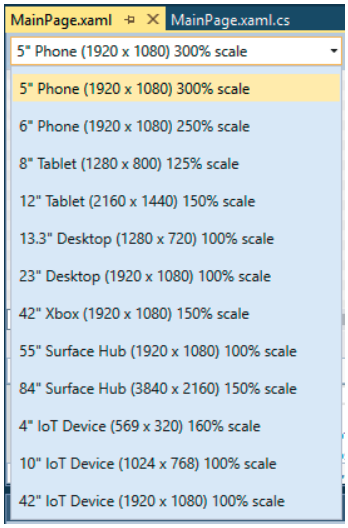

Bild 5.6 Auswahl der gewünschten Auflösung der Zielplattform

5.5 Beispielanwendung: Hello Pi

Unser erstes Beispielprojekt mit dem Namen „Hello Pi“ führt in die Softwareentwicklung für den Raspberry Pi in Kombination mit Windows 10 und Visual Studio 2015 ein.

Das Projekt soll zwei Aufgaben erledigen:

- Ausgabe des Textes „Hello Pi“ über HDMI am Monitor: Hiermit wird der grundlegende Aufbau der Universal Apps gezeigt.
- Blinken einer LED in einem zeitlichen Rhythmus: So erlernen Sie exemplarisch die Ansteuerung der Hardware (in diesem Fall der GPIO-Pins).

Das „Hello Pi“-Beispiel basiert auf dem Einsteiger-Beispiel „Blinky“ von Microsoft[2].

[2] *https://ms-iot.github.io/content/en-US/win10/samples/Blinky.htm*

5.5.1 Benötigte Bauteile

Zur Umsetzung dieses Beispielprojekts benötigen Sie:

- ein Breadboard
- einige Steckbrücken
- eine LED
- einen 220-Ohm-Widerstand

Darüber hinaus sollte der Raspberry über einen Netzwerkzugriff verfügen. Damit Sie das Programm steuern können, sind zusätzlich eine USB-Maus und ein angeschlossener Monitor am HDMI-Ausgang hilfreich.

5.5.2 Hardwareaufbau

Der Hardwareaufbau ist einfach:

1. Verbinden Sie das kürzere Beinchen (Kathode, Minus) der LED mit GPIO 5 (das ist Pin 29).
2. Verbinden Sie das längere Beinchen mit dem Widerstand.
3. Verbinden Sie das andere Beinchen des Widerstands mit einem der 3,3 V-Pins auf dem Raspberry.

HINWEIS: Die Polarität der LED ist wichtig, damit das Beispiel funktioniert. Diese Art der Schaltung nennt man „Low-aktiv“. Dies bedeutet, wenn ein Low-Signal (0) anliegt, wird die LED leuchten.

Die LED sollte zunächst nicht leuchten. Einen schematischen Aufbau des Projekts finden Sie in Bild 5.7.

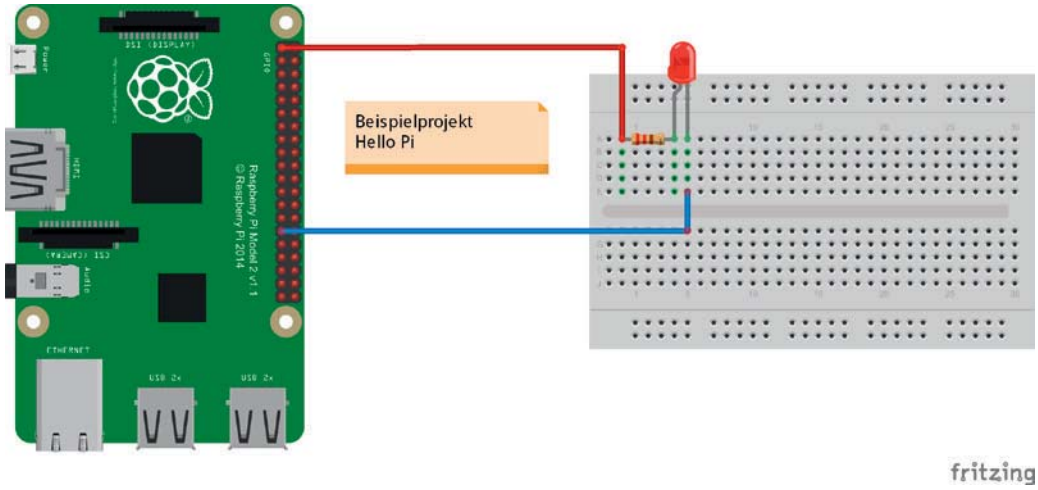

Bild 5.7 Schaltungsaufbau des Beispielprojekts „Hello Pi“

Bild 5.7 zeigt, wie die Schaltung auf einem Breadboard aussehen könnte.

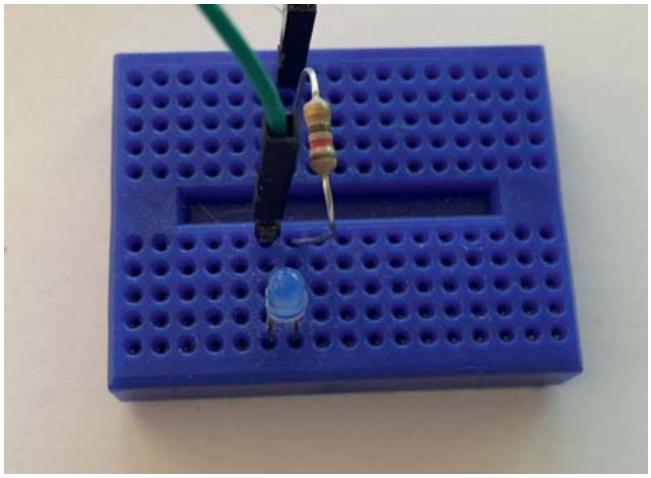

Bild 5.8 Breadboard-Schaltung der LED

5.5.3 Projekt in Visual Studio anlegen

Legen Sie zunächst ein neues Projekt in Visual Studio an. Hierzu wählen wir zunächst den Projekttyp *Blank App (Universal Windows)* aus (Bild 5.9). Achten Sie auf den Namen und das gewünschte Projektverzeichnis. Die Anbindung einer Quellcode-Verwaltung ist selbstverständlich auch bei Universal Apps möglich.

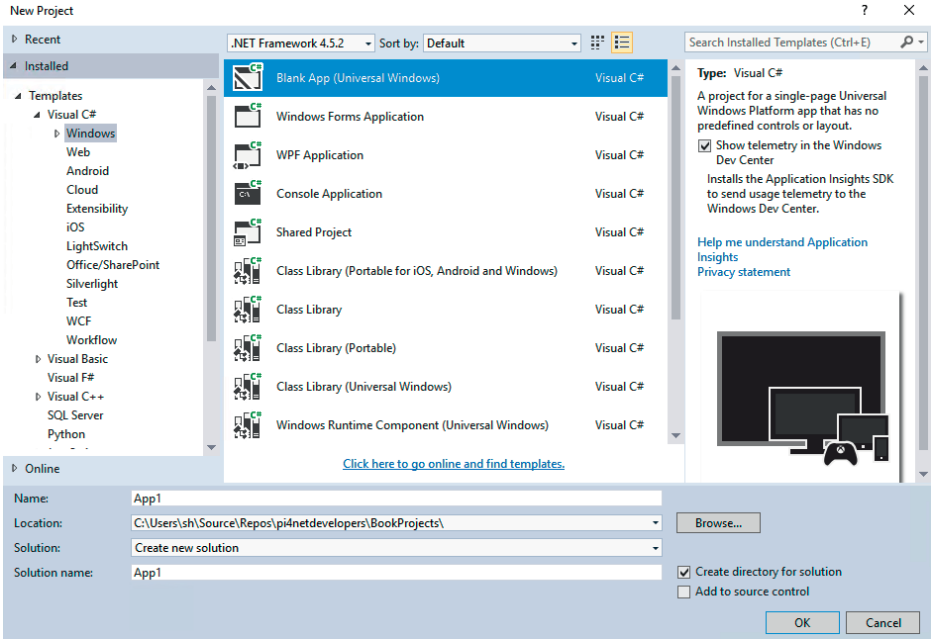

Bild 5.9 Neues Projekt in Visual Studio anlegen

Nachdem das Projekt angelegt wurde, müssen Sie zunächst eine neue Referenz hinzufügen (Bild 5.10) und dort *Windows IoT Extensions for the UWP* auswählen. Sie finden die Referenz unter:

```
Universal Windows->Extensions->Windows IoT Extensions for the UWP
```

Diese Referenz ist notwendig, damit wir die GPIO-Pins vom Code aus ansteuern können.

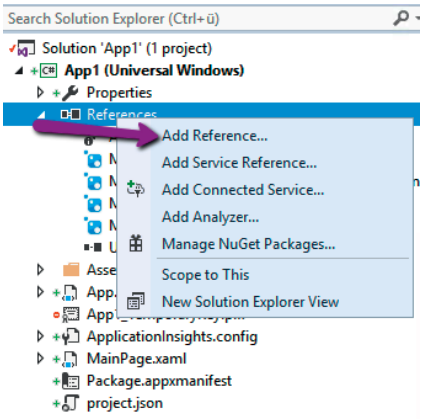

Bild 5.10 Hinzufügen einer neuen Referenz

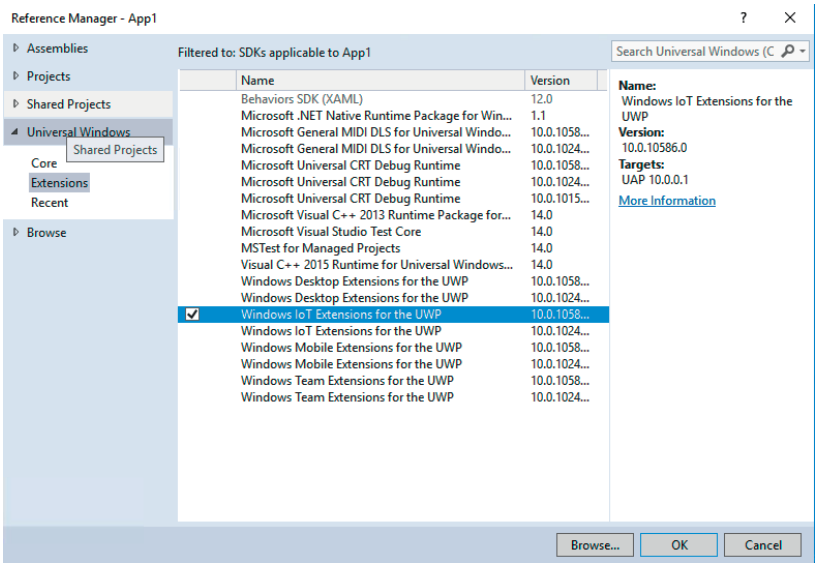

Bild 5.11 Referenz „Windows IoT Extension for the UPW hinzufügen“ (SDK)

HINWEIS: Sofern Ihnen mehrere Varianten der Referenz zur Verfügung stehen, achten Sie darauf, die richtige Version einzubinden. Die richtige Version entspricht der Build-Version von Windows 10, die auf dem Raspberry installiert wurde. Im Zweifel verwenden Sie die neuste verfügbare Version.

5.5.4 Programmaufbau

Auch wenn Sie bereits Erfahrungen in der Softwareentwicklung besitzen, wird das erste Projekt noch Schritt für Schritt erklärt. Bei den späteren Projekten in Kapitel 6 wird dann vor allem auf die spezifischen Besonderheiten eingegangen.

Ein Grundprojekt einer Universal App beinhaltet vier Dateien, die für uns zunächst wichtig sind:

- *App.xaml* und *App.xaml.cs*
- *MainPage.xaml* und *MainPage.xaml.cs*

Es handelt sich dabei um zwei XAML-Seiten mit jeweils der zugehörigen Code-Behind-Datei. Die **App** ist hierbei für die generelle Anwendung verantwortlich; während die **MainPage** das erste Fenster der Anwendung darstellt.

Die Beispiele in diesem Buch kommen fast immer mit einer einzelnen Seite (Page) aus. Es handelt sich somit um Single-Page-Anwendungen. Der größte Teil des Codes wird daher zunächst in der MainPage abgelegt werden.

5.5.4.1 Hinterlegen des Begrüßungstexts und Start-Buttons

Bevor wir uns an die Funktionen der Hardware machen, legen wir den Begrüßungstext an. Wechseln Sie hierzu in die Datei *MainPage.xaml* und ändern den Bereich von `Grid` wie folgt ab:

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
        <TextBox x:Name="HelloPi" Text="Mein erstes Raspberry-Projekt!"
Margin="10" IsReadOnly="True"/>
        <Button x:Name="Start" Content="Start"  Margin="10"
HorizontalAlignment="Center" Click="Start_Click"/>
    </StackPanel>
</Grid>
```

Damit wurde die Anwendung um einen Textbereich und einen Button erweitert. Damit die Anordnung auch optisch ansprechend ist, wurde ein Stackpanel eingebaut. Der Button dient uns später dazu, die Anwendung zu steuern.

Bild 5.12 zeigt, wie die Ausgabe der Anwendung über die HDMI-Schnittstelle am Bildschirm aussehen würde.

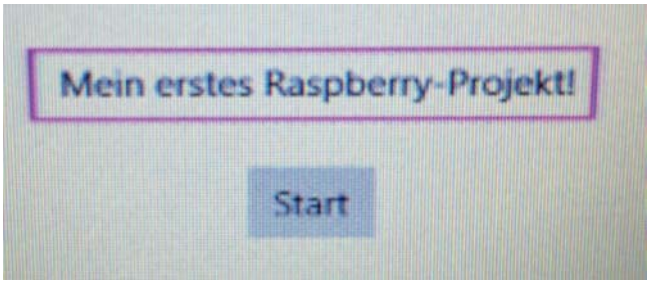

Bild 5.12 Ausgabe der Anwendung über HDMI

TIPP: Sie können das Programm natürlich so umbauen, dass keine Benutzerinteraktion erforderlich ist. In diesem Fall können Sie auch auf den Bildschirm verzichten und das Blinken der LED direkt verfolgen.

5.5.4.2 Timer-Komponente

Damit im Intervall die LED an- und ausgeschaltet werden kann, benötigt man einen Timer. Die Timer sind analog zu Standard-Windows-Projekten zu implementieren.

HINWEIS: Zu beachten ist jedoch, dass die Genauigkeit der Timer nicht sonderlich hoch ist, sie befindet sich in jedem Fall im Millisekunden-Bereich. Logiken, die eine höhere Genauigkeit erfordern, wie z. B. das Initialisieren eines Chips durch Wechselsignale im Nano-Sekundenbereich, können hiermit nicht abgebildet werden.

Wir verwenden in unserem Beispiel den DispatcherTimer[3], der in einem Intervall von 500 Millisekunden das Event `Timer_Tick` feuert.

Wenn das Event gefeuert wird, schalten wir die LED an oder aus. Dieser Code wird in der *MainPage.cs* hinterlegt.

Listing 5.3 Verwendung des Windows-Timers

```
public MainPage()
{
    this.InitializeComponent()
    // Timer initialisieren und starten

    timer = new DispatcherTimer();
    timer.Interval = TimeSpan.FromMilliseconds(500);
    timer.Tick += Timer_Tick;
    timer.Start();

    // . . .
}

private void Timer_Tick(object sender, object e)
{
    // . . .
    // Weitere Logik
}
```

5.5.4.3 GPIO

Im nächsten Schritt benötigen wir die Steuerung der GPIO-Pins, damit wir die LED hardwareseitig an- oder ausschalten können. Hierzu muss zunächst das Using in der *MainPage.cs* eingebunden werden:

```
using Windows.Devices.Gpio;
```

[3] *https://msdn.microsoft.com/de-de/library/system.windows.threading.dispatchertimer(v=vs.110).aspx*

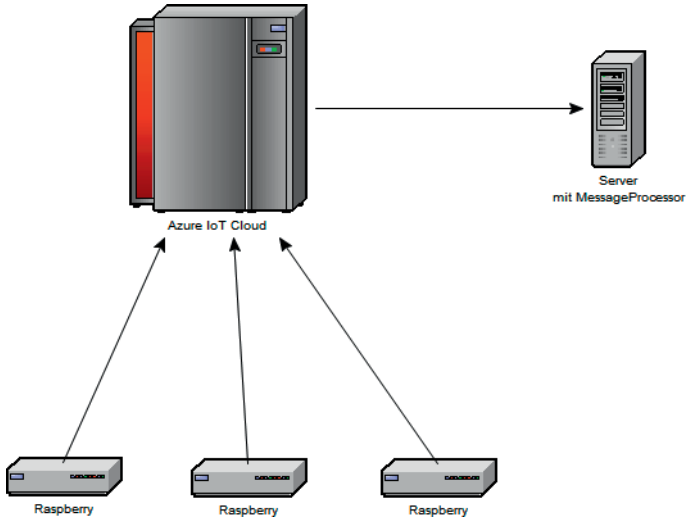

Bild 6.31 Beispielszenario zur Verbindung zwischen Raspberry und Cloud

Die Abrechnung erfolgt in diesem Fall über die Anzahl der versendeten Nachrichten. Microsoft kommt auch hier den Entwicklern entgegen und bietet ein kostenloses Abo an, das für kleinere und mittlere Projekte geeignet ist. Zum Zeitpunkt der Drucklegung war das kostenlose Angebot auf 500 Geräte und 8000 Nachrichten pro Tag limitiert.

Neben den Angeboten zum Nachrichtenaustausch wird Microsoft hier künftig weitere Dienste anbieten, wie z. B. Streaming-Daten oder maschinelles Lernen zur besseren Analyse von Daten. Es ist davon auszugehen, dass das Angebot künftig noch deutlich erweitert wird.

Einsteigerprojekte sind unter folgenden Links bei Microsoft erhältlich und dienen als Vorlage für die genannten Beispiele:

- *https://blogs.windows.com/buildingapps/2015/12/09/windows-iot-core-and-azure-iot-hub-putting-the-i-in-iot*
- *https://azure.microsoft.com/de-de/documentation/articles/iot-hub-csharp-csharp-getstarted*

6.7.1 Benötigte Komponenten

Für dieses Projekt wird benötigt:

- Raspberry Pi mit Windows 10 auf SD-Karte und Netzteil
- Netzwerkverbindung über Kabel oder WLAN
- Ein Microsoft Azure-Konto. Sollten Sie noch kein Konto besitzen, so können Sie ein Testkonto[5] einrichten.

[5] *https://azure.microsoft.com/de-de/pricing/free-trial*

6.7.2 Hardwareaufbau

Im Projekt wird keine weitere Hardware – neben der Grundausstattung – benötigt. Natürlich kann das Projekt hinsichtlich Hardware beliebig erweitert werden, um Daten auszuwerten (z. B. Temperatursensor) oder Aktionen über die Cloud zu steuern (z. B. Lichtschalter wird von einem anderen Sensor betätigt, der die Helligkeit im Raum misst).

6.7.3 Softwareaufbau

Der Softwareaufbau besteht in diesem Fall aus zwei Blöcken. Einerseits muss die grundlegende Cloud-Infrastruktur im Microsoft Azure-Portal eingerichtet werden. Auf der anderen Seite benötigt man zwei Softwareprojekte, eines für die Raspberries, die die Nachrichten versenden. Das zweite Projekt soll auf einem Server laufen, der die Nachrichten zentral empfangen soll.

6.7.4 Einrichtung der Cloud

Zunächst müssen wir bei Microsoft einen Azure IoT Hub erstellen. Mithilfe des Hubs (sinngemäß „zentrale Stelle“) soll eine sichere Kommunikation zwischen den IoT-Geräten oder einem Backend-Dienst ermöglicht werden.

Als Erstes müssen Sie sich im Azure-Portal[6] anmelden und dort einen neuen IoT Hub anlegen. Wechseln Sie dazu zunächst auf *Neu*, dann auf *Internet der Dinge* und dann auf *Azure IoT Hub* (Bild 6.32). Im nächsten Fenster muss das Projekt noch weiter eingerichtet werden (Bild 6.32). Hierzu muss zunächst ein Name vergeben werden. Ist dieser gültig und frei, wird ein grünes Häkchen angezeigt.

Bei *Tarif und Skalierung* können Sie den kostenfreien Tarif *F1* wählen. Dieser genügt in jedem Fall für kleine Projekte.

Als Nächstes muss noch eine Ressourcengruppe erstellt und zugewiesen werden. Die Gruppen werden in der Regel dazu verwendet, Ressourcen einer Anwendung (z. B. Datenbank-Server, IoT Hub, virtueller Server) logisch zu bündeln[7].

Im letzten Schritt muss noch ein Standort zugewiesen werden. Hier empfiehlt es sich, einen Standort zu wählen, der sich geografisch in der Nähe befindet. So können beispielsweise Latenzen niedrig gehalten werden.

[6] *https://portal.azure.com*
[7] *https://azure.microsoft.com/de-de/documentation/articles/resource-group-portal*

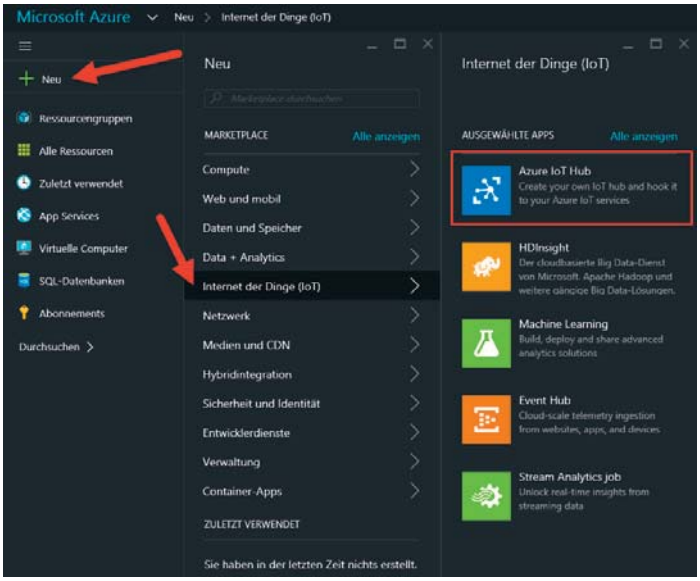

Bild 6.32 Anlegen eines neuen Azure IoT Hub

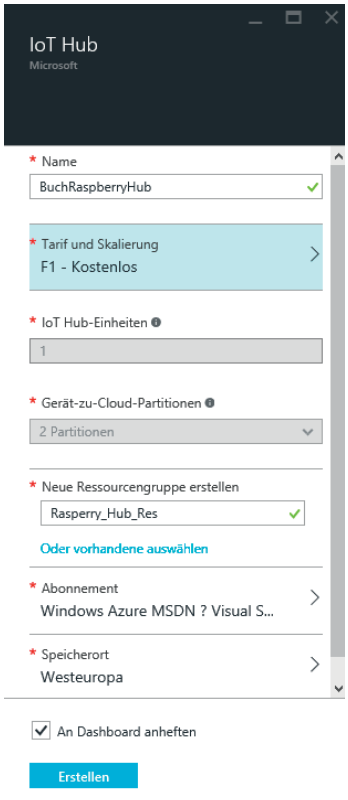

Bild 6.33 Detail-Parameter für den IoT Hub