SMART HOME SELBST GEBAUT

Mach schlau das Haus

Der Artikel beschreibt eine Beispielanwendung für die Gebäudeautomatisierung mit Raspberry Pi und Windows 10 IoT Core.

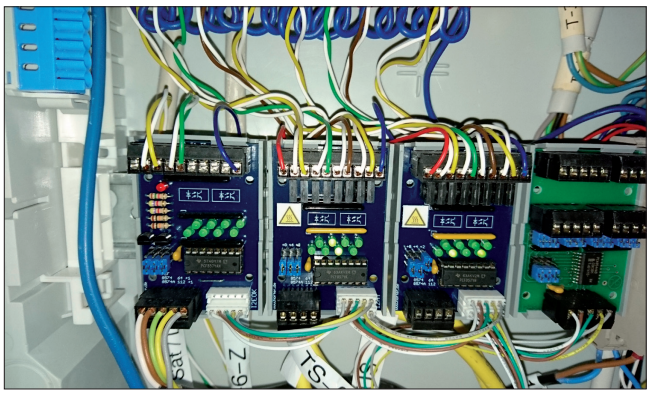
Die Schlagworte Internet of Things (IoT) oder Industrie 4.0 haben sich mittlerweile auch in der Umgangssprache etabliert. Selbst Politiker treiben diese Digitalisierung voran. Auch bei den Großen der IT-Branche ist das Thema inzwischen angekommen. Unter diesen bemüht sich Microsoft um entsprechende Marktanteile.

Sie als Entwickler profitieren von vielen coolen und kostenlosen Werkzeugen, um sich mit den IoT-Technologien zu beschäftigen.

Dieser Artikel zeigt anhand eines Beispiels für eine Smart-Home-Steuerung den Umgang mit Raspberry Pi, Windows 10 IoT Core und I²C-Bus inklusive Hardware sowie der Möglichkeit der Integration in die Cloud. Die Ansätze und Technologien lassen sich dabei auf professionelle IoT-Lösungen übertragen, sodass dieser Artikel sowohl den Hobbyentwickler als auch den IT-Profi ansprechen soll.

Technischer Schnelleinstieg zu IoT

Grundlegend besteht eine IoT-Lösung aus einer Vielzahl von Geräten, welche Daten erfassen und übertragen. Die Geräte sind vielfältig hinsichtlich der Funktionalität sowie der Größe und der Komplexität. Beispiele sind einfache Temperatursensoren mit Kommunikationsschnittstellen bis hin zu komplexen Maschinen, welche Prozessdaten in die Cloud transferieren. Zum Teil gibt es auch sogenannte Broker, welche die lokale Kommunikation zu den Geräten übernehmen und ausgewählte Daten weiterreichen. Die Verbindung in die Cloud erfolgt dabei beispielsweise über Standardprotokolle wie HTTP, zum Beispiel in Form von REST-Endpoints oder über IoT-Protokolle wie MQTT.

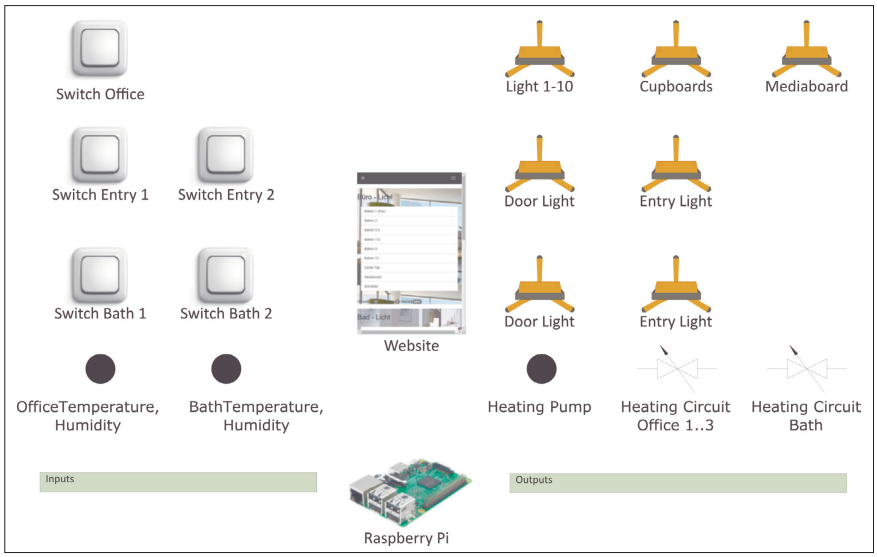


Verdrahtung der I²C–Module (Bild 2)

Grundlegend lassen sich damit Überwachungs- und Steuereinrichtungen losgelöst von Standortgrenzen umsetzen. Aufgrund des technischen Fortschritts und der Minifizierung der Geräte können diese fast unsichtbar im Alltag eingesetzt werden.

Auch eine instabile Anbindung an das Internet ist kein Problem. Die Anzahl möglicher Anwendungen ist unbegrenzt. Häufig findet man IoT aktuell in den Bereichen der Gebäudeautomatisierung, zum Beispiel Fernsteuerung der Heizung, in der Industrie (zum Beispiel Erfassen von Prozessdaten) oder auch im Sport, etwa als Fitness Tracker.

Ein häufiges Ziel von IoT-Lösungen besteht darin, ein bestehendes System wie eine Fertigungsanlage oder ein Gebäude oder komplexe Prozesse auf der Grundlage der gewonnenen Daten besser zu verstehen und letztlich zu optimieren. Das Ziel kann sein, die Ausbeute bei einem Fertigungsprozess zu erhöhen oder den Energiebedarf in einem Gebäude zu reduzieren.



Viele Elemente, ein System: Aufbau der Hardware mit Sensoren und Aktoren (Bild 1)

Beispielprojekt Smart Home

Als Einstieg in die Welt der Gebäudeautomatisierung wurde ein Praxispro-

```
pi@raspberrypi: ~
0xff
pi@raspberrypi:~ $ i2cget -y 1 0x38
0xff
pi@raspberrypi:~ $ i2cget -y 1 0x20
0xff
pi@raspberrypi:~ $ i2cget -y 1 0x21
0xf7
pi@raspberrypi:~ $ i2cset -y 1 0x21 0xF6
pi@raspberrypi:~ $ i2cget -y 1 0x21
0xf6
pi@raspberrypi:~ $ i2cset -y 1 0x21 0xF7
pi@raspberrypi:~ $ i2cget -y 1 0x21
0xf7
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: 20 21 -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- 38 -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
pi@raspberrypi:~ $
```

Befehle, die zum Testen mit i2cdetect ausgeführt wurden (Bild 3)

jekt erstellt. Dabei steuert ein Raspberry Pi Lichtquellen und Ventile einer Fußbodenheizung in einem Büro. Die Steuerung der Lichtquellen erfolgt mittels Standardtastern aus dem Elektroinstallationszubehör. Es kommen keine speziellen Taster für einen Bus wie den KNX-Bus zum Einsatz.

Weiterhin gibt es eine Website für das mobile Endgerät, um die Lichtquellen an- und auszuschalten sowie Parameter wie etwa Temperaturgrenzen für die Steuerung der Fußbodenheizung zu konfigurieren. Der grundlegende Aufbau ist in Bild 1 gezeigt.

Das Herz der Steuerung bildet der Raspberry Pi 3. Dieser Ein-Platinen-Computer ist sehr performant und bietet neben einer ARM-Dual-Core-CPU, einem HDMI-Anschluss, Ethernet, WLAN, Bluetooth, 24 nutzbaren GPIO-Pins (General Purpose Input Output) auch einen SPI- und einen I²C-Bus und damit eine optimale Basis. Microsoft unterstützt dieses Board mit dem Betriebssystem Windows 10 IoT Core und stellt damit für .NET-Entwickler eine gute Grundlage für Anwendungen bereit.

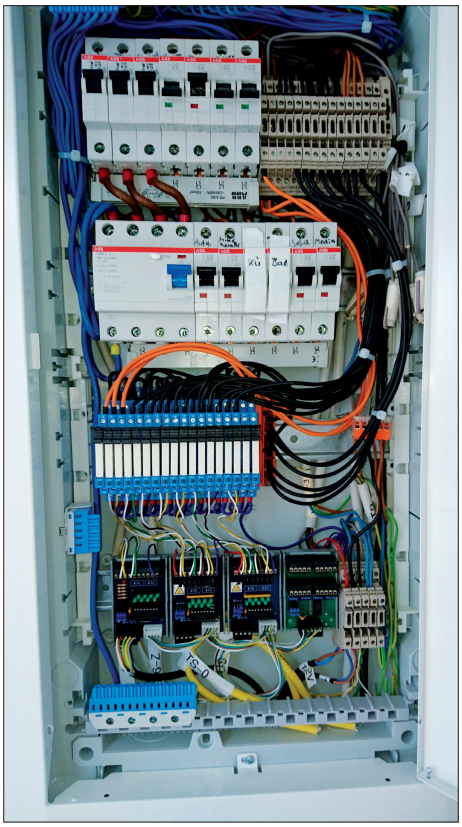
Für den Raspberry Pi ist eine Micro-SD-Karte mit 8 GB oder mehr erforderlich. Die Installation des kostenlos verfügbaren Betriebssystems Windows 10 IoT Core auf der SD-Karte erfolgt mit dem Tool „IoT Dashboard“. Eine Anleitung inklusive Downloads findet sich unter [1]. Interessant ist die Hardware-Kompatibilitätsliste von Microsoft, welche beispielweise die unterstützten SD-Karten beschreibt [2].

Für den Anschluss der erforderlichen Aktoren – hier die Lichtquellen und die Ventile – sowie der Sensoren (Taster, Temperatursensoren) gibt es verschiedene Lösungsvarianten. Es ist möglich, die digitalen Ein-/Ausgangspins (GPIOs) zu nutzen. Diese werden mit dem Pegel von 3,3 V betrieben. Damit benötigt man eine Schutzschaltung, um den Raspberry bei höheren Spannungen nicht zu zerstören. Diese können durch Induktivitäten oder bei Fehlern auftreten.

Der Nachteil dieser Umsetzung ist aber, dass man sehr viel Eigenleistung in die Schaltungsentwicklung investieren muss. Weiterhin ist die Anzahl der verfügbaren GPIOs ziemlich begrenzt.

Eine bessere Möglichkeit bietet der I²C-Bus. Eine Einführung zum I²C-Bus findet sich in [3]. Für ihn gibt es eine Vielzahl von I²C-Hardware zu kaufen. Die Firma Horter & Kalb [4] beispielsweise bietet digitale und analoge Ein- und Ausgangsmodule und I²C-Repeater als Bausatz oder als fertige Module an.

Der Anschluss des I²C-Busses erfolgt über das I²C-Repeater-Modul von Horter & Kalb, welches die Pegel von 3,3 V auf den I²C-Standard-Pegel von 5 V anhebt. Damit kann man Standard-I²C-Module direkt anschließen. Das I²C-Repeater-Modul steckt man dabei einfach auf die GPIO-Leiste des Raspberry Pi und stellt den I²C-Bus für die Peripherie physikalisch zur Verfügung. Die Verkabelung der weiteren Module erfolgt durch fünf Drähte vom Repeater zum ersten Modul und von dort zum nächsten (Bild 2).



I²C-Module, Finder-Relais und Sicherungen (Bild 4)

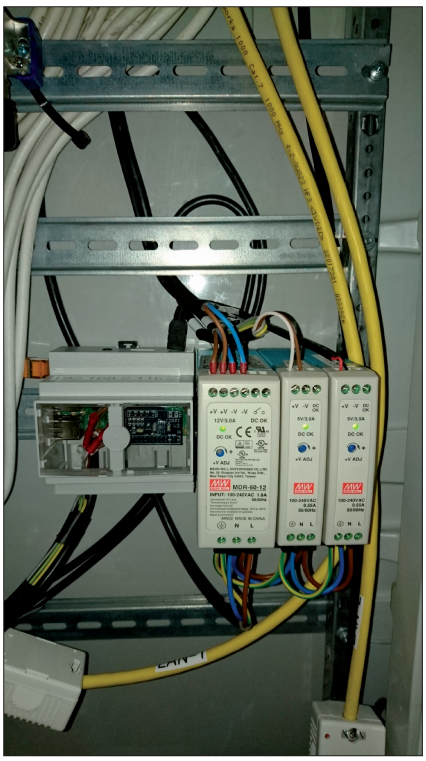
Ein 5-V-Netzteil versorgt die Taster sowie den I²C-Bus. Diese Taster schließt man direkt am digitalen Eingangsmodul (8 Bit) an. Die Aktoren der Beispielanwendung werden mit 230-V-Wechselspannung versorgt. Im Beispiel steuert der Raspberry einen Ausgang am I²C-Ausgangsmodul an. Die Ausgangsmodule besitzen eine invertierte Logik, das heißt, logisch 1 wird durch 0 V und eine logische 0 wird mit 5 V signalisiert.

Der Ausgang wird durch 12 V versorgt und gibt diese dann auf 12-V-Finder-Relais (schmale Bauform) weiter. Diese Relais wiederum schalten dann die 230-V-Wechselspannung des Aktors. Weiterhin bietet das Ausgangsmodul eine galvanische Trennung zwischen Steuereinheit und Verbraucher mittels Optokopplern, sodass der Raspberry vor Überspannung geschützt ist. Die Temperaturmessung erfolgt mit dem I²C-Temperatursensor Bosch BME 280. Im Beispiel gibt es zwei Stück – für jeden Raum einen. ▶

Da beim I²C-Bus jeder Teilnehmer über eine eindeutige ID verfügen muss, wird im Beispiel ein Multiplexer eingesetzt. Andernfalls müsste die ID des Temperaturfühlers angepasst werden.

Aktuell gibt es für Windows 10 IoT Core kein nützliches I²C-Testtool. Abhilfe schafft an dieser Stelle das Tool i2c-detect für Linux. Dazu wird der Raspberry mit Linux gestartet. Über das Tool putty verbindet man sich mit dem Raspberry und kann nun die grundlegende Funktion des I²C-Bus testen beziehungsweise die I²C-Teilnehmer-IDs ermitteln. Für das Standard-Login ist der Benutzername *pi* und das Passwort *raspberry*. Bild 3 zeigt die ausgeführten Befehle mit i2cdetect. [5] gibt weiterführende Informationen dazu.

Die Arbeiten im Schaltschrank wurden von einem regionalen Elektroinstallationsfachbetrieb ausgeführt. Bild 4 zeigt den Schaltschrank mit Sicherungen und FI-Schalter, Relais und I²C-Modulen. In Bild 5 sehen Sie die Netzteile sowie den Raspberry Pi mit Hutschienen-Gehäuse.



Raspberry und Netzteile (Bild 5)

Softwareentwicklung

Der Raspberry Pi mit Windows 10 IoT Core wird im lokalen Netzwerk verfügbar gemacht. Danach kann die Entwicklung unter Visual Studio mit Vorlage für *Universal Apps* beginnen. Im Beispiel wurde noch Visual Studio 2015 genutzt. Das Vorgehen sowie die Toolkette sind für Visual Studio 2017 allerdings identisch.

In der Beispielanwendung wurden drei grundlegende Anforderungen umgesetzt:

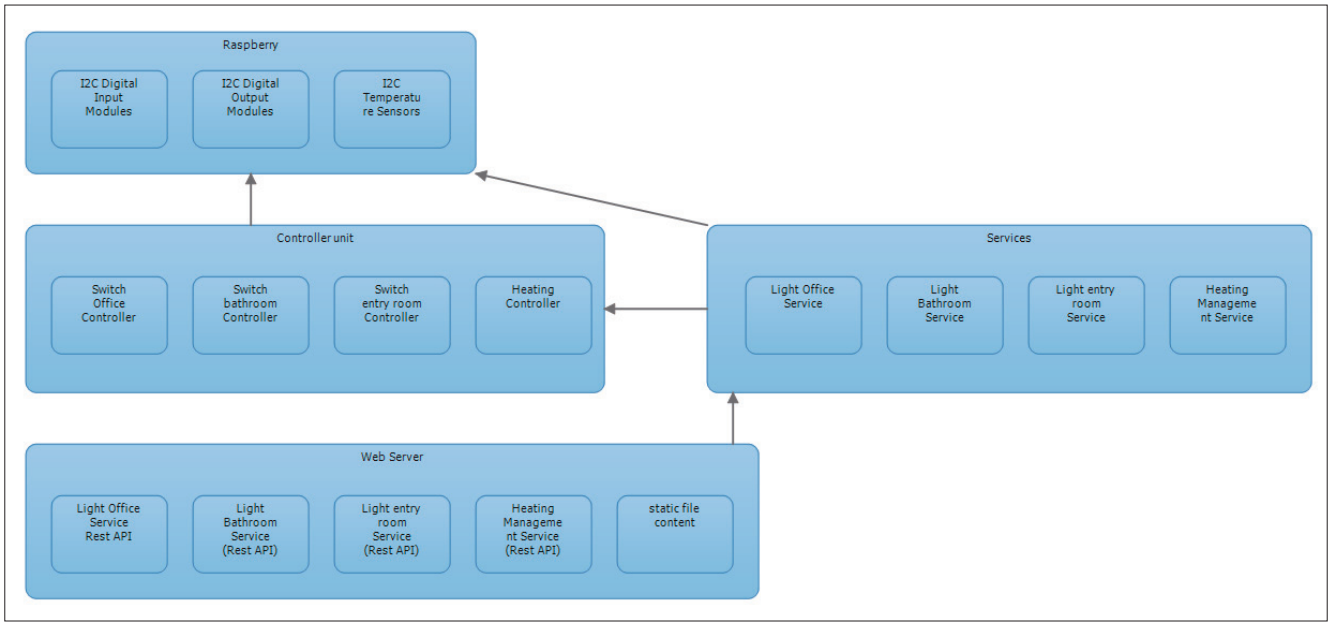
1. Eine Website für den Zugriff via mobile Device
2. Steuerungslogik für Lichtquellen und Ventile
3. Ansteuerung der I²C-Module

Bild 6 zeigt die vereinfachte Gesamtarchitektur der Software.

Eine SPA (Single Page Application) dient als UI für das Smart Home. Dabei stellt das Backend REST-Endpunkte bereit, welche im Frontend mit JavaScript angesprochen werden. Der statische Inhalt sorgt mit Bootstrap für die Anpassung an verschiedenste Bildschirmauflösungen und mit Knockout.js wird das MVVM-Pattern umgesetzt. Diesen Bereich der Gesamtlösung würde der Autor heute eher mit Angular 4 umsetzen. Bild 7 zeigt die Website im Browser.

Windows 10 IoT Core bietet keine IIS (Internet Information Services), wie man sie von anderen Windows-Betriebssystemen gewohnt ist. Deshalb bleibt an dieser Stelle nur die Eigenentwicklung eines Webservers oder die Hoffnung auf ein Community-Projekt. Zum Glück hat Tom Kuijsten sein Webserver-Projekt auf GitHub veröffentlicht und gut dokumentiert [6]. Damit kann man in fast gewohnter Manier REST-Services erstellen beziehungsweise statischen Inhalt ausliefern. Listing 1 zeigt einen WebAPI-REST-Controller.

Das Projekt ist über NuGet veröffentlicht und wird einfach im eigenen Projekt referenziert. Ein möglicher REST-Control-



Vereinfachte Gesamtarchitektur (Bild 6)

ler ist in Listing 1 aufgeführt. Funktional entsteht im Beispiel nur ein Wrapper für das WebAPI (Projekt: *SmartApp.WebPortal*).

Zusätzlich enthält das Projekt auch den Inhalt für die Webseite in Form von HTML und JavaScript. Der Webserver wird als einzelner Backgroundtask auf dem Raspberry Pi gestartet. Danach horcht er schon auf Anfragen.

Im Beispiel ist das Starten des Webservers im Projekt *SmartApp.Host.Webserver* umgesetzt. Listing 2 zeigt die Initialisierung des Webservers.

Steuerungslogik für Lichtquellen und Ventile

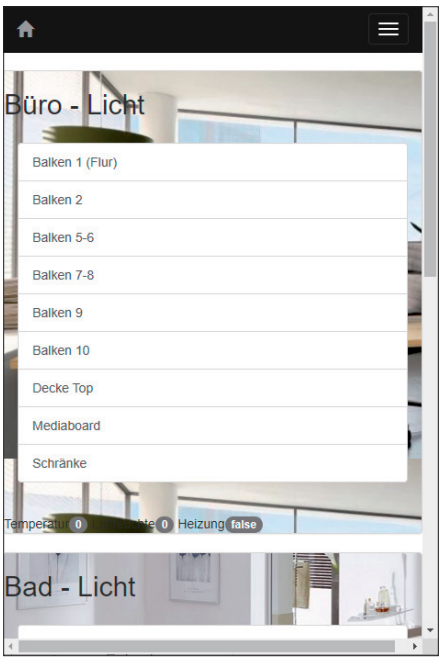
Die Projektmappe enthält ein weiteres Projekt vom Typ Klassenbibliothek: Im Projekt *SmartApp.Services* ist die eigentliche fachliche Logik implementiert, also das, was passieren soll, wenn ein Taster gedrückt wird oder die untere Temperaturgrenze für die Fußbodenheizung erreicht ist. Diese Logik wird ebenfalls von der Website konsumiert.

Im folgenden Listing ist ein Beispiel für ein Polling der Eingänge aufgeführt, um auf die Ereignisse von Tastern zu reagieren. Die Lichtquelle *EntryMainLight* schaltet demnach auf Tastendruck der Taster *SwitchEntry* oder *EntryMainLight*. Es findet sich keine Referenz auf die reale Peripherie, sodass hier Unit-Tests platziert werden können.

```
while (!_stoppController)
{
  // entry lights
  if (switchEntryChecker.StateChanged(this._lightsEntry
    .SwitchEntry) | switchMediaboard1Checker
    .StateChanged(this._lightsOffice.SwitchMediaboard1))
  {
    if (this._lightsEntry.SwitchEntry |
      this._lightsOffice.SwitchMediaboard1)
    {
      this._lightsEntry.EntryMainLight =
        !this._lightsEntry.EntryMainLight;
```

Listing 1: REST-Controller

```
[UriFormat("/lightoffice/balklight1/{state}")]
public IPutResponse SwitchBalkLight1(bool state)
{
  _lightOfficeService.SwitchBalkLight1(state);
  return new PutResponse(PutResponse
    .ResponseStatus.OK);
}
```



Website zur Steuerung des Smart Home (Bild 7)

```
    }
  }
}
```

Zugriff auf die Hardware vom Raspberry Pi

Dazu wird die „Windows IoT Extension for the UWP“ benötigt, welche die Funktionen durch ein leicht verständliches API bereitstellt. Das Beispielprojekt enthält eine weitere Klassenbibliothek, welche den Zugriff auf den I²C-Bus des Raspberry umsetzt. Ohne Detailwissen zum I²C-Bus erfolgt nun der Zugriff auf entsprechende Module. Listing 3 zeigt die Initialisierung eines I²C-Teilnehmers.

Der Beispielcode enthält für jeden Typ eines I²C-Moduls eine Schnittstelle mit entsprechender Implementierung. Die Klasse *DigitalOutputModule* zeigt die Implementierung für ein I²C-8-Bit-Ausgangsmodul.

Die Implementierung bleibt damit relativ übersichtlich. Das Wichtigste sind an dieser Stelle die richtigen I²C-Geräte-IDs, welche zuvor mit dem bereits beschriebenen Linux-Tool ermittelt wurden.

Bereitstellung als Backgroundtask

Wie schon bei der Umsetzung der Website erwähnt, wird ein Backgroundtask immer als separates Projekt (Typ *Universal App*) bereitgestellt. In der Beispielanwendung wurde neben dem Backgroundtask für die Website auch ein Background- ▶

Listing 2: Start des Webservers

```
// register handlers for webserver
var restRouteHandler = new RestRouteHandler();
restRouteHandler.RegisterController<
  HeatingManagementController>();
// register all the other custom
// rest controller here

var configuration = new HttpServerConfiguration()
  .ListenOnPort(80)
  .RegisterRoute("api/v1", restRouteHandler)
  .RegisterRoute(new StaticFileRouteHandler(
    @"SmartApp.WebPortal\Web\"))
  .EnableCors();
  // allow cors requests on all origins

_httpServer = new HttpServer(configuration);

await _httpServer.StartServerAsync();
```

task zur Überwachung der Taster (*SmartApp.Host.SwitchController*) sowie der Raumtemperatur inklusive Steuerung der Ventile (*SmartApp.Host.HeatingController*) umgesetzt. Listing 4 zeigt die Startmethode für einen Backgroundtask. Ein projektinterner Helper (sehr einfache Dependency Injection Factory) hilft beim Zugriff auf die Serviceinstanz für die Heizungssteuerung.

Ein Package-Manifest beschreibt eine UWP-App. Damit wissen das System sowie der Benutzer, welche Ressourcen die UWP nutzen muss. In der Beispielanwendung werden dem Projekt *SmartApp.Host.Webserver* beispielsweise die Fähigkeiten Internet (Client), Internet (Client & Server), Private Networks (Client & Server) zugeordnet.

Mit dem üblichen Startbefehl über die Taste [F5] in Visual Studio erfolgt das Deployment sowie der Start der Backgroundtask-Projekte auf dem Raspberry. Häufig treten Fehler auf, weil die Option *Remote Machine* nicht richtig konfiguriert wurde. Dabei ist der Hostname oder alternativ die IP-Adresse des Raspberry einzutragen. Die Debugging-Features von Visual Studio funktionieren ohne Probleme.

Weitere Deployment-Optionen sind über das Kontextmenü *Bereitstellen* oder über PowerShell-Skripte erreichbar. Zusätzlich kann eine App über das Device-Portal hochgeladen werden. Nach dem Deployment des Projekts *SmartApp.Host.Webserver* ist die Website über den URL *http:///[Raspberry-IP]/views/index.html* erreichbar.

Der Raspberry mit Windows 10 IoT Core bietet eine Konfigurationswebsite – das sogenannte Device-Portal. Im Standard wird dazu eine Website nach Anmeldung als Administrator über *http:///[Raspberry-IP]:8080* zur Verfügung gestellt. Dort ist es möglich, Apps hochzuladen – ohne Einsatz von Visual Studio oder PowerShell – oder Live-Daten der CPU-Auslastung zu sehen und eine ganz Menge mehr.

Anbindung eines IoT-Devices an die Microsoft Azure Cloud

Die Vorstellung des Beispielprojekts ist damit abgeschlossen. Eine Nutzung von Cloud Services (wie zum Beispiel eine Sprachsteuerung) wurde im Beispielprojekt aus Zeitgründen noch nicht umgesetzt, soll aber noch folgen.

In professionellen IoT-Lösungen bietet aber genau die Cloud enormes Potenzial. Häufig werden dabei Daten von einem Azure Table Storage konsumiert, der Blob Storage (für Dateien) genutzt oder Geräteinformationen über Twins bereitgestellt.

Damit kann man beispielweise lokale Prozessdaten einer Maschine oder eines Gebäudes in die Cloud stellen. Das funktioniert mithilfe des Azure IoT Hub von Microsoft. Sehen Sie dazu auch [7].

Listing 3: I²C-Geräte-Initialisierung

```
_deviceId = deviceId;

// get i2c bus devices
string i2CDeviceSelector =
  I2cDevice.GetDeviceSelector(BusId);
IReadOnlyList<DeviceInformation> i2CDevices =
  await DeviceInformation.FindAllAsync(
    i2CDeviceSelector);
if (i2CDevices.Count == 0)
{
  return false;
}

var settings = new I2cConnectionSettings(_deviceId)
{
  BusSpeed = I2cBusSpeed.StandardMode,
  SharingMode = I2cSharingMode.Shared
};

_device = await I2cDevice.FromIdAsync(
  i2CDevices[0].Id, settings);
```

Listing 4: Starten des Backgroundtask

```
public void Run(IBackgroundTaskInstance taskInstance)
{

  _deferral = taskInstance.GetDeferral();
  // -> don't release deferral,
  // otherwise app will stop

  taskInstance.Canceled += BackgroundTaskCanceled;

  if (_heatingController != null)
  {
    // clean up existing stuff
    _heatingController.StopControlHeating();
    while (!_heatingController.IsStopped)
    {
      // nothing to do
    }
  }
  else
  {
    _heatingController = DependencyFactory
      .GetInstance<IHeatingRoomService>();
  }

  _heatingController.StartControlHeating();
}
```

Für C# und weitere Sprachen (zum Beispiel Node.js) existieren gute Helper-Libaries. Alternativ gibt es ein REST-API. Im Kontext von C# und Universal Apps wird dazu das NuGet-Paket *Windows.Azure.Storage* für den Table- und Blob Storage bereitgestellt.

Für die Kommunikation über den Azure IoT Hub existiert das NuGet-Paket *Microsoft.Azure.Devices.Client*. Damit ist die Anbindung des Raspberry an die Azure-Cloud relativ schnell umgesetzt.

Fazit

Der Artikel hat anhand einer Beispiellösung einer Smart-Home-Steuerung die Möglichkeiten des Raspberry Pi mit Windows 10 IoT und I²C-Modulen aufgezeigt. Vielleicht findet sich der ein oder andere Häuslebauer hier wieder, der zukünftig sein eigenes Smart Home in DIY-Manier zum vergleichsweise kleinen Preis erstellt.

Der Technologie-Stack erlaubt die Umsetzung von professionellen IoT-Lösungen. Dazu braucht der Entwickler .NET-Kenntnisse und Erfahrungen im Webbereich sowie Grundkenntnisse der Elektrotechnik.

Unter diesen Voraussetzungen ist der Einarbeitungsaufwand gering. Es gibt aber auch professionelle Hardwarelösungen, zum Beispiel von der Janz Tec AG auf Basis des Raspberry Pi 3 mit nützlicher Außenbeschaltung und vorinstalliertem Windows 10 IoT [8]. ■

[1] Windows 10 IoT Core, www.dotnetpro.de/SL1804WindowsIoT1
[2] Hardware-Kompatibilitätsliste für Windows 10 IoT, www.dotnetpro.de/SL1804WindowsIoT2
[3] I²C-Bus-Erklärung, www.dotnetpro.de/SL1804WindowsIoT3
[4] Horter & Kalb, I²C-Module und Raspberry-Zubehör, www.dotnetpro.de/SL1804WindowsIoT4
[5] i2C detect für Linux, www.dotnetpro.de/SL1804WindowsIoT5
[6] Webserver für Windows 10 IoT Core, www.dotnetpro.de/SL1804WindowsIoT6
[7] Entwicklerhandbuch IoT Hub, www.dotnetpro.de/SL1804WindowsIoT7,
[8] Janz Tec AG, www.dotnetpro.de/SL1804WindowsIoT8

Ronald Creutz
arbeitet seit über zehn Jahren mit Microsoft-Technologien als Entwickler, Architekt und Berater in Projekten im Umfeld von Industrie, Automatisierung und IoT. Derzeit absolviert er ein Fernstudium der technischen Informatik.
info@ronald-creutz.de

dnpCode A1804WindowsIoT