Bishop's University
Division of Natural Sciences & Mathematics

Weather Awareness App on Raspberry Pi 3 with Win 10 IoT Core

A project submitted for
CS501 Internet of Things

by

Tianye Zhao (002254003)

Professor

Rami Yared

November 2018

Table of Contents

1. Overview

As people who live in certain area are always concern about local climate. Here introduces a device which is operated on Raspberry Pi 3 Model B with Sense HAT. It connects up to the internet to get a forecast, and then presents the forecast to user. It also monitors the temperature in local area, and uploads the information to the cloud. The whole device is hyper-local weather reporting to be able to help forecast.

2. Design

As most of the people have cellphones, they prefer to get weather information in apps. But it is easily to find that the information in apps may not be so accurate and in time. For example, for those who live near Taklamakan desert in Xinjiang, China, the temperature may change rapidly during short period. What's more, some cellphones may not function properly in extreme weather condition, such as powering off automatically when it becomes cold. Thus, it should be better to have a small and simple IoT device which can help report and feedback local weather in time. It would be a great help to prepare people for climate change and gather accurate information for record.

This device is operated on Raspberry Pi 3 Model B with Sense HAT which includes 8×8 RGB LED matrix, a five-button joystick, and a temperature sensor. Windows 10 IoT Core system is running on the motherboard with micro SD. It connects up to the internet to get the forecast from an open weather API, and then it presents the responded forecast to user for agreeing or disagreeing with it. Thereafter, it also monitors what's actually happening in local area, collects local temperature information and then uploads the data to the cloud in Azure. So the whole device is hyper-local weather reporting to be able to help do better weather forecasting.

2.1 *Required Resources*

Hardware	Raspberry Pi 3 Model B[CITATION Ras \l 1033][CITATION Win \l 1033] with micro USB cable (CDN$50) Sense HAT[CITATION Sen \l 1033] (CDN$50) Sandisk micro SD 16GB[CITATION Har \l 1033] (CDN$15) HDMI cable (borrow from friend) Monitor, mouse and keyboard (university library)
System	Windows 10 IoT Core[CITATION Win1 \l 1033]
Cloud	Azure IoT Hub[CITATION Azu \l 1033]
Software	Windows 10 IoT Core Dashboard[CITATION Win2 \l 1033] Visual Studio Community 2017 Device Explorer[CITATION Mic \l 1033]
API	OpenWeatherMap [CITATION Ope \l 1033]

2.2 *Planned work*

Activity	Description
Gather required resources	Purchase Raspberry Pi, Sense HAT etc. Download Windows 10 IoT Core required tools
Set up device	Install the IoT Core Dashboard and Windows 10 IoT Configure Raspberry Pi hardware Configure Sense HAT
Build the device	IoT app design Implement UX for testing Implement sensor access Implement data acquisition Implement state display
Connect to Cloud	Connect to Azure IoT Hub Send data to the cloud

3. Windows 10 IoT Core

Windows 10 IoT comprises two main editions. There are the IoT Core edition and the IoT Enterprise edition[CITATION Win1 \l 1033]. Windows 10 IoT Core is the smallest, simplest configuration that can be used to build a simple IoT device.

3.1 *Why Windows 10 IoT Core*

It runs on both x86 and ARM, various different versions of device can be built on different platforms and it can run with same code across both systems. It also contains an almost full implementation of the Windows Universal Windows Platform API set, basic API set is included in Windows 10 IoT Core. Certain aspects of it can be managed over the web using Windows Azure. It provides full functions on XAML, existing codes can be complied easily with a display to monitor the process. Either compute libraries or some user interfaces can be complied conditionally for desktop version or IoT device version.

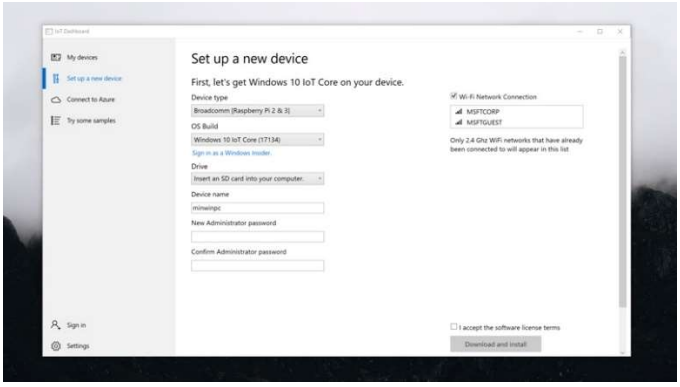
3.2 *Requirement*

Windows 10 IoT Core is the lightest version of Windows 10 can be used for IoT. It requires 256MB of RAM without a display, and 512MB of RAM with display. It requires a minimum of 2GB of storage, and can run either x86 or ARM.

4. Hardware

4.1 *Raspberry Pi 3*

There are many devices that Windows 10 IoT can run on[CITATION Win \l 1033]. This project choose Raspberry Pi 3, which is an ARM processor. The Windows 10 IoT Core system can be flashed on to the blank micro SD card[CITATION Har \l 1033] through Windows 10 IoT Dashboard[CITATION Win2 \l 1033].

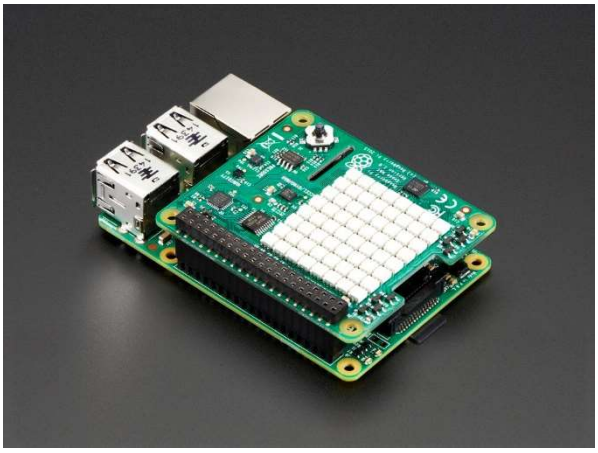


Source: https://docs.microsoft.com/en-us/windows/iot-core/media/iotdashboard/dashboard-800x450.jpg

Figure 1 Windows 10 IoT Dashboard

4.2 Sense HAT

Sensors are things that can detect change in the external environment - temperature, humidity, switches, light, orientation, motion, etc. Display can provide visual feedback. Sense HAT[CITATION Sen \l 1033] can be plugged directly into Raspberry Pi and it has a great collection of sensors and effectors already built in, which is an ideal board for this project. It contains the sensors - Orientation (yaw, pitch & roll) via an accelerometer, 3D gyroscope, and magnetometer, Pressure, Humidity, Temperature. The 8x8 LED Matrix enables display the data from the various sensors. The joystick can also be used to enable a human user to interact with the programs running on the Raspberry Pi Sense HAT. With emmellsoft open-source library[CITATION RPi \l 1033], all of its sensors and effectors can be manipulated.



Source: https://cdn-shop.adafruit.com/970x728/2738-01.jpg

Figure 2 Raspberry Pi 3 with Sense HAT

5. Installation of Windows 10 IoT Core

5.1 Enable administrator account

In order to be able to flash Windows 10 IoT Core to the SD cards (latest version is 17763 which is released in October 2018, there are also two previous realeases 17134 and 16299 available[CITATION Pre \l 1033] on the website), administrator account should be logged in. To enable hidden super-administrator account using Command Prompt:

a. Open command prompt as administrator.
b. Run the following command to activate administrator user: *net user administrator /active:yes*.

c. To set a password for administrator, use the following command: *net user administrator **.

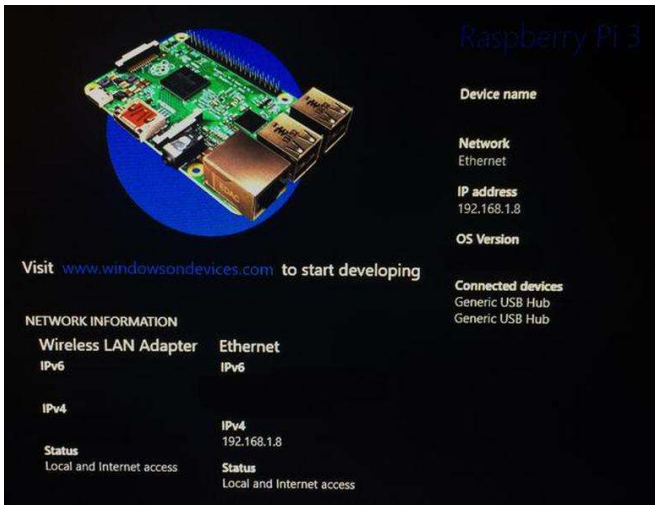
5.2 Install and Configure Raspberry Pi

Follow the tutorial video provided by Microsoft[CITATION Get \l 1033] for installation.

Run PowerShell command to add Raspberry Pi device to the trusted hosts file:

set-Item Wsman:\localhost\client\TrustHosts –value {IP address}

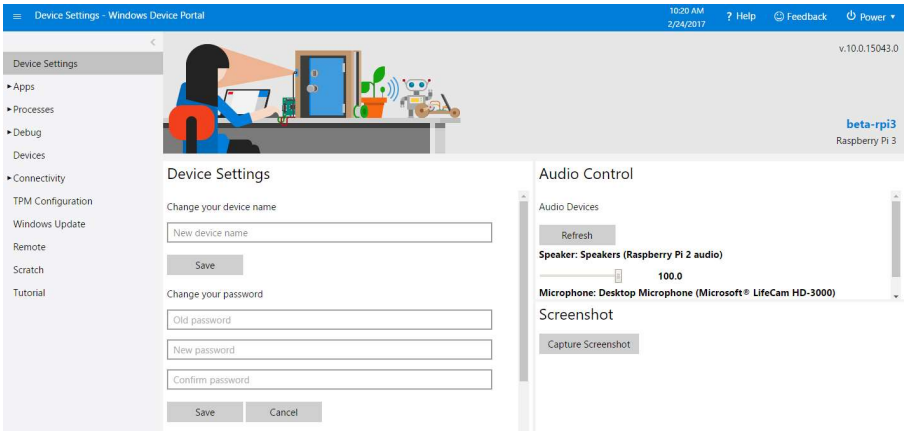
IP address can be got on screen once device connected to display.



Source: https://tr3.cbsistatic.com/hub/i/2018/07/06/d8957dd3-2696-4383-84cc-01229a8b08e2/step9.jpg

Figure 3 Screenshot of Windows 10 IoT Core on Raspberry Pi

To access the IoT device through Device Portal. Make sure the device connects the same network with laptop. Type *{IP address}:8080* in browser and log in the administrator account created during burning the SD card.



Source: https://docs.microsoft.com/en-us/windows/iot-core/media/deviceportal/deviceportal.png

Figure 4 Screenshot of Windows Device Portal

6. Software and API

6.1 *Visual Studio 2017*

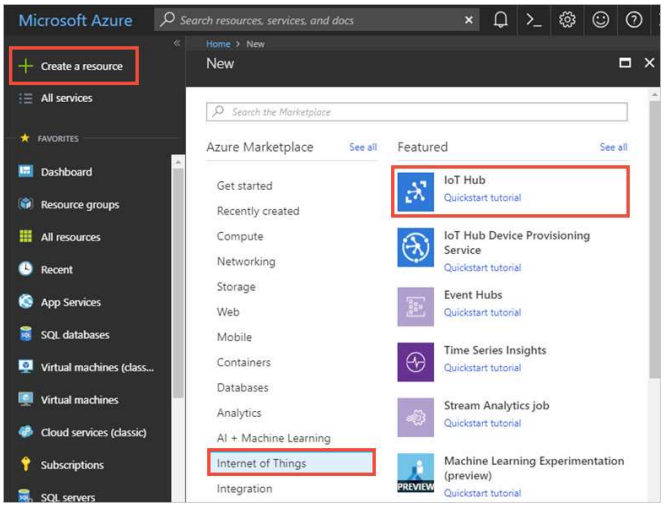
Install Windows IoT Core Project Templates for VS2017[CITATION VS \l 1033] in Marketplace and install. Choose an ARM processor and Remote Machine (Type device *{IP address}* in Manual Configuration). Install Connected Service for Azure IoT Hub[CITATION Con \l 1033]. Add connected service in References.

6.2 *OpenWeatherMap API*

In API document[CITATION Ope \l 1033], 3-hours forecast is used for this project.

6.3 *Azure*

Set up IoT hub on Azure[CITATION Qui1 \l 1033], connect device to the hub.

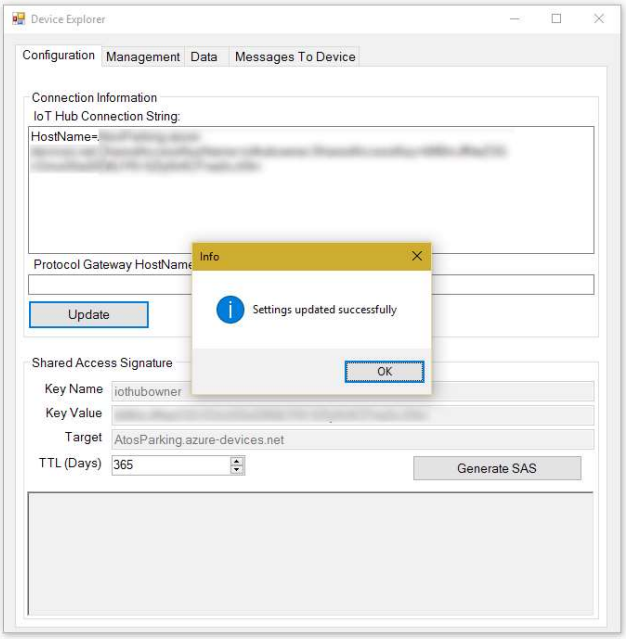


Source: https://docs.microsoft.com/en-us/azure/includes/media/iot-hub-create-hub/create-iot-hub1.png

Figure 5 IoT Hub in Azure

6.4 *Device Explorer*

Comply Device Explorer[CITATION Mic \l 1033] to monitor data communication on Azure. Copy IoT Hub HostName connection link to Device Explorer to gain access.



Source: https://sandervandevelde.files.wordpress.com/2016/02/08-iot-hub-extension.png

Figure 6 Connect IoT Hub through Device Explorer

7. Weather Awareness App Development

7.1 Overview

Weather Awareness App is a product that's designed to provide human-assisted weather data. Users can vote on whether agreeing or disagreeing with the next 3-hours forecast. It monitors the current set of conditions to determine whether or not the vote is accurate and upload the results to cloud.

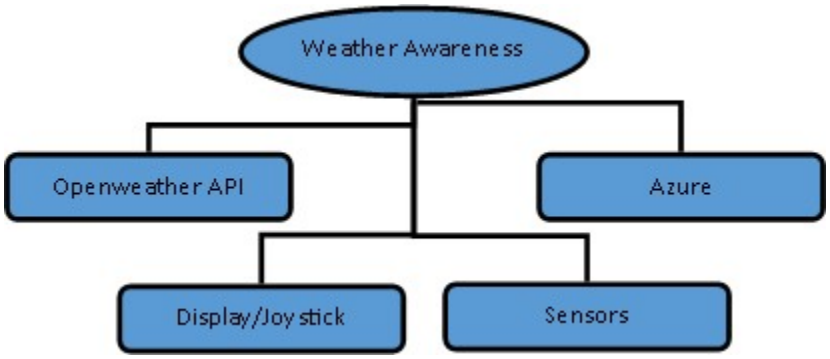


Figure 7 Communications in the App

7.2 User Experience

With text plus colors on the 8x8 LED display. The user votes yes or no using the joystick first on the temperature and the forecast. Over the course of the day and the night,

ambient temperature with visual feedback displays using OpenWeatherMap[CITATION Ope \l 1033] API service.

7.3 Design Flow

With only a joystick and a display, it is easier to move the app through states. When the app starts up, it makes a light show first to let user know the product is running, then get the forecast with a yellow screen (a yellow screen means a request is sent off and a purple screen means a response is received. Then it asks the user if agreeing with the forecasted temperature which is depended on day or night (through comparing sunrise and sunset time, it shows high temperature on day and low temperature at night), and user can vote yes (green screen) or no (red screen). Next it moves on to ask the user whether agree with the forecast, it twinkles all LEDs again which indicates that user has made both options and keep a log of what happened. Finally it continues monitor the ambient temperature every 5 seconds, shows the result on display and keeps record in log file.

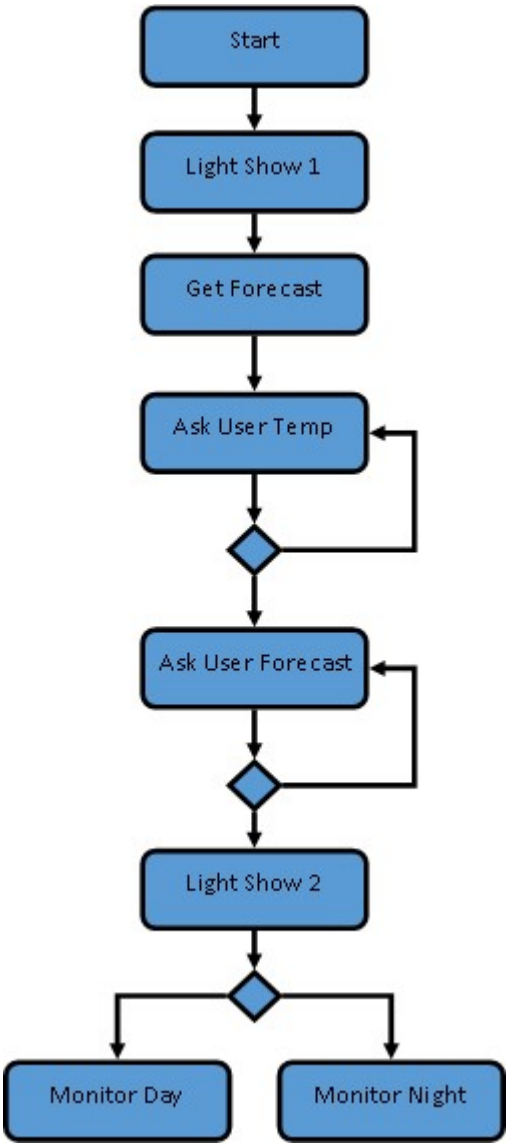


Figure 8 States in Weather Awareness App

7.4 UX Implementation

a. Create Universal Windows template with desired IoT version[CITATION VS \l 1033] .(even though only latest target version 17134 is supported, it works for 17763)

b. Modify *MainPage.xaml* file, create controls to display title and log.

c. Add open source IO library *IO Recipes*[CITATION IOR \l 1033] to project, create log file (File path on the device - *User files\LocalAppData\{AppName}\LocalState\log.txt*, which can be checked through Device Portal) and write entries to it.

7.5 *Sensor Access Implementation*

Import *RPi.SenseHat* and *RTMULibCS* projects (which includes the interfaces for the display, joystick and sensors) from *RPi.SenseHat* library[CITATION RPi \l 1033]. Implement run loops and timer callback to make sure only one task is handled at same time. Gain access to Sense HAT during initialization stage and set enumerators to monitor button states on Sense HAT.

7.6 *Data Acquisition with OpenWeatherMap API*

API call for weather data:

http://api.openweathermap.org/data/2.5/forecast?id=6146143&units=metric&mode=xml&cnt=1&APPID={APIKey}

It returns weather data (in Celsius degree) of Sherbrooke[CITATION Lis \l 1033] in XML format for next 3 hours[CITATION 5da \l 1033] as following (an API key should be registered first):


```
▼<weatherdata>
  ▼<location>
     <name>Sherbrooke</name>
     <type/>
     <country>CA</country>
     <timezone/>
     <location altitude="0" latitude="45.4001" longitude="-71.8991" geobase="geonames" geobaseid="6146143"/>
   </location>
   <credit/>
  ▼<meta>
     <lastupdate/>
     <calctime>0.0049</calctime>
     <nextupdate/>
   </meta>
   <sun rise="2018-11-02T11:29:27" set="2018-11-02T21:32:10"/>
  ▼<forecast>
    ▼<time from="2018-11-02T21:00:00" to="2018-11-03T00:00:00">
       <symbol number="500" name="light rain" var="10n"/>
       <precipitation unit="3h" value="1.58" type="rain"/>
       <windDirection deg="69.0027" code="ENE" name="East-northeast"/>
       <windSpeed mps="1.51" name=""/>
       <temperature unit="celsius" value="7.81" min="7.77" max="7.81"/>
       <pressure unit="hPa" value="990.45"/>
       <humidity value="99" unit="%"/>
       <clouds value="overcast clouds" all="92" unit="%"/>
     </time>
   </forecast>
</weatherdata>
```


Figure 9 Response from OpenWeatherMap API call

From above, low temperature, high temperature, sunrise, sunset and weather condition[CITATION Wea \l 1033] are obtained.

7.7 *State Machine Implementation*

Set two timers:

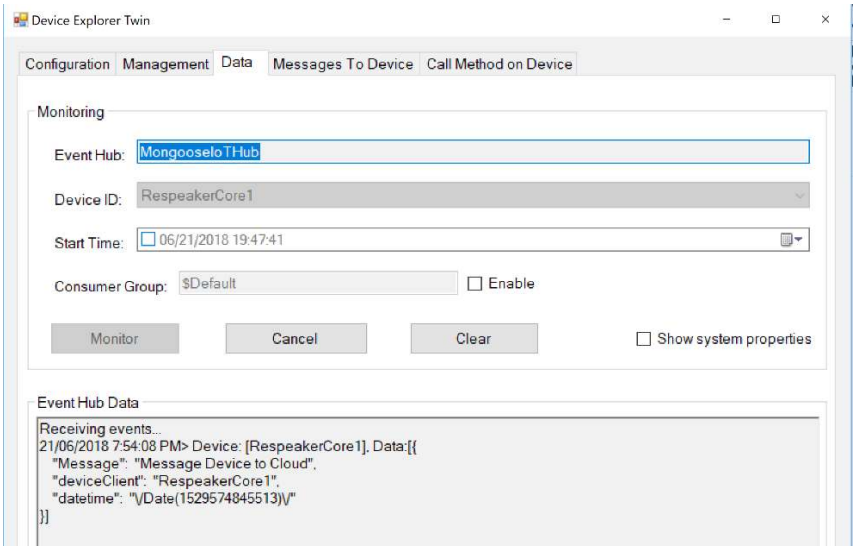
timer 1 (15ms)	timer 2 (4s)
Check joystick to call state-specific method	Acquire update temperature

Implement states mentioned previously in design flow:

States	Description
START_UP	Move on to the first light show
LIGHT_SHOW_1	Pixelated random color display Move on to get forecast
GET_FORECAST	Get weather forecast data from API Move on to ask temperature
ASK_TEMP	Show the temperature to user Ask user make decision Move on to ask forecast
ASK_FORECAST	Show the forecast to user Ask user make decision Move on to light show 2
LIGHT_SHOW_2	Pixelated random color display Move on to get monitor ambient temp
MONITOR_DAY	Show high temperature if daytime Move to monitor night if nighttime Move to light show 2 after certain runs
MONITOR_NIGHT	Show low temperature if nighttime Move to monitor day if daytime Move to light show 2 after certain runs
WEATHER_ERROR	Display error (??) if forecast not get

7.8 Connect to Azure

Provision the Raspberry Pi 3 to mate the device twin created in cloud through IoT Dashboard (make sure software TPM installed on device). Modify *AzureIoTHub.cs*[CITATION Get1 \l 1033] file to achieve sending and getting message from Azure. Monitor data transmit in Device Explorer.



Source: https://i0.wp.com/blog.kloud.com.au/wp-content/uploads/2018/06/Device-Explorer.png?resize=768%2C488&ssl=1

Figure 10 Monitor data through Device Explorer

References

[1] "Raspberry Pi 3 Model B," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[2] "Windows 10 IoT Core Development Devices," [Online]. Available: https://docs.microsoft.com/en-us/windows/iot-core/tutorials/quickstarter/prototypeboards#windows-10-iot-core-development-devices.

[3] "Sense HAT," [Online]. Available: https://www.raspberrypi.org/products/sense-hat/.

[4] "Hardware compatibility list," [Online]. Available: https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/hardwarecompatlist#storage-media.

[5] "Windows 10 IoT Core," [Online]. Available: https://developer.microsoft.com/en-us/windows/iot.

[6] "Azure IoT Hub," [Online]. Available: https://azure.microsoft.com/en-ca/services/iot-hub/.

[7] "Windows 10 IoT Core Dashboard," [Online]. Available: https://developer.microsoft.com/en-us/windows/iot/Downloads.

[8] "Microsoft Azure IoT SDK for .NET," [Online]. Available: https://github.com/Azure/azure-iot-sdk-csharp.

[9] "OpenWeatherMap API," [Online]. Available: https://openweathermap.org/api.

[10] "RPi.SenseHat," [Online]. Available: https://github.com/emmellsoft/RPi.SenseHat.

[11] "Previous Windows 10 IoT Core releases," [Online]. Available: https://docs.microsoft.com/en-us/windows/iot-core/downloads#previous-windows-10-iot-core-releases.

[12] "Getting Started - Windows 10 IoT Core + Raspberry Pi 3," [Online]. Available: https://www.youtube.com/watch?v=JPRUbGIyODY.

[13] "Windows IoT Core Project Templates for VS 2017," [Online]. Available: https://marketplace.visualstudio.com/items?itemName=MicrosoftIoT.WindowsIoTCoreProjectTemplatesforVS15.

[14] "Connected Service for Azure IoT Hub," [Online]. Available: https://marketplace.visualstudio.com/items?

itemName=MicrosoftIoT.ConnectedServiceforAzureIoTHub.

[15] "Quickstart: Control a device connected to an IoT hub (.NET)," [Online]. Available: https://docs.microsoft.com/en-us/azure/iot-hub/quickstart-control-device-dotnet.

[16] "IORecipes," [Online]. Available: https://github.com/DreamTimerZ/IORecipes.

[17] "List of city ID," [Online]. Available: http://bulk.openweathermap.org/sample/.

[18] "5 day weather forecast," [Online]. Available: https://openweathermap.org/forecast5.

[19] "Weather Conditions," [Online]. Available: https://openweathermap.org/weather-conditions.

[20] "Get started with Azure IoT Hub and Visual Studio connected services (C#)," [Online]. Available: https://github.com/Azure/azure-iot-hub-vs-cs/wiki/C%23-Usage.