



# **Project Report** Weather Awareness

CS501 Internet of Things

Tianye Zhao

# Introduction

## Overview

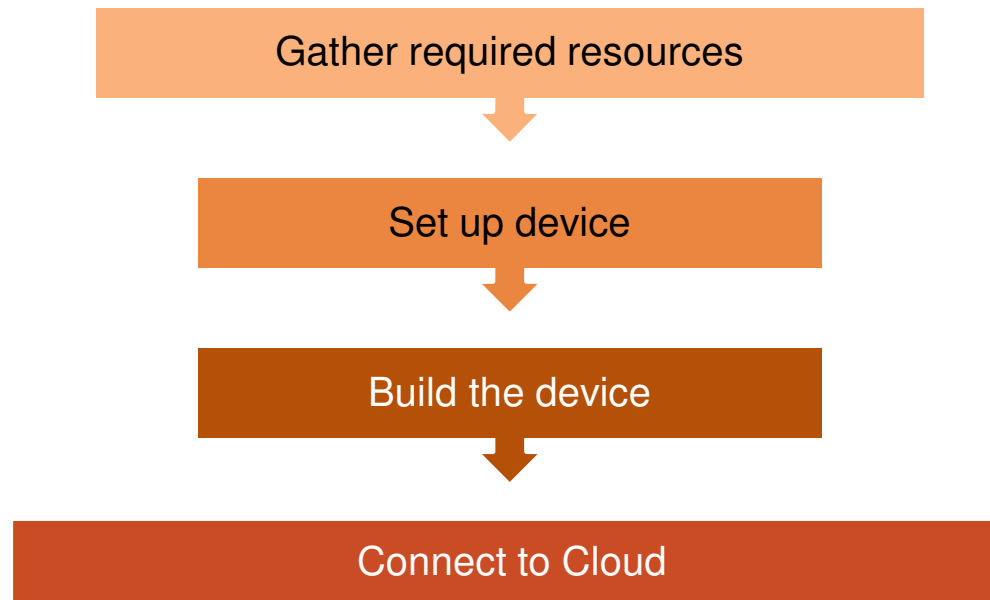
As people who live in certain area are always concern about local climate. Here introduces a device which is operated on Raspberry Pi 3 Model B with Sense HAT. It connects up to the internet to get a forecast, and then presents the forecast to user. It also monitors the temperature in local area, and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

# Introduction

## Approach

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.

# Plan



# Plan

## 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

# Plan

## 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

# Required

## Hardware

- Raspberry Pi 3 Model B: CDN\$ 45.75
- Sense HAT: CDN\$ 48.95
- SanDisk Ultra Micro SDHC 16GB: CDN\$ 13.73
- Power Supply: CDN\$ 14.95

## Service

- OpenWeatherMap
- Azure

## Software

- IoT Dashboard
- Device Explorer

# Windows 10 IoT

## Windows 10 IoT

- IoT Core
- IoT Enterprise (Create commercial device)

## Why Windows 10 IoT?

- Runs on both **x86 and ARM**.
- Contains an almost full implementation of the **Windows UWP** (Universal Windows Platform) API set, i.e. XAML.
- Contains **security and management** tools.
- Can use **existing code** (C#/C++)
- Can use **web technology** (Node.js)



# IoT Core

## Windows 10 IoT Core

- Simple configuration to build a small IoT device

## Minimum Requirement for Windows 10 IoT Core

- 256MB RAM (no display)
- 512MB RAM (with display)
- 2GB storage
- x86 or ARM

# Hardware

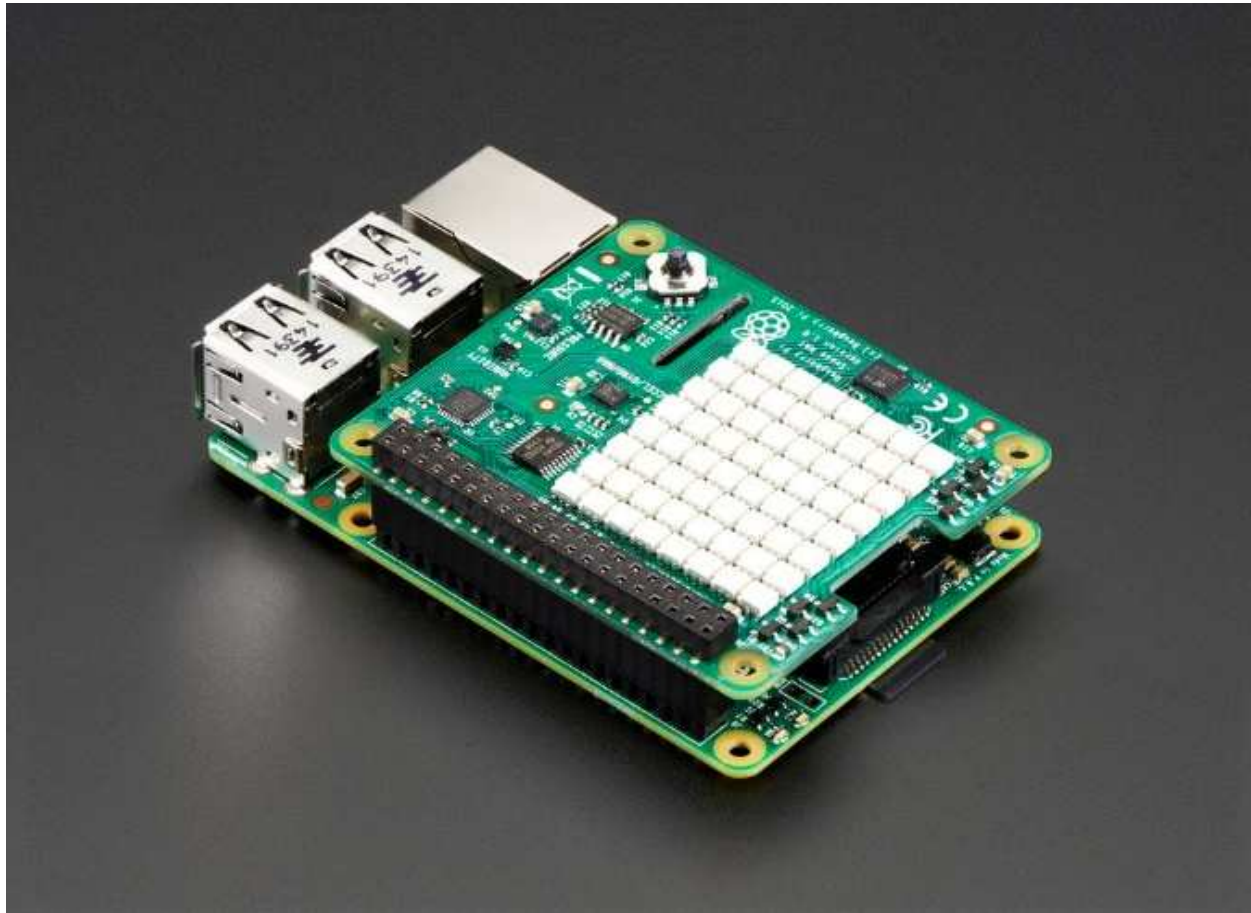
## Raspberry Pi 3 Model B (ARM)

- 3B+ is in technical preview

## Sensors and Effectors

- Breadboard
- “Hat” board

# Pi w/ Hat



# Installation

## Download and Install (Tutorial)

- IoT Dashboard
- Enable administrator account: use command line *net user Administrator /active:yes* and *net user Administrator \** to activate Administrator account and set up password
- Install to SD card
- Enable Developer Mode in Windows 10: Update & Security > For developers

## Configure Raspberry Pi

- Add Raspberry Pi to trusted hosts file
- Access device through Device Portal

# IoT Core Desktop



Raspberry Pi 3

**Device name**

**Network**  
Ethernet

**IP address**  
192.168.1.8

**OS Version**

Visit [www.windowsondevices.com](http://www.windowsondevices.com) to start developing

**Connected devices**  
Generic USB Hub  
Generic USB Hub

**NETWORK INFORMATION**

Wireless LAN Adapter	Ethernet
<b>IPv6</b>	<b>IPv6</b>
<b>IPv4</b>	<b>IPv4</b>
<b>Status</b> Local and Internet access	<b>Status</b> Local and Internet access

# Weather Data

## API Call

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

## XML

```

<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="3n" 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>

```

# App Dev

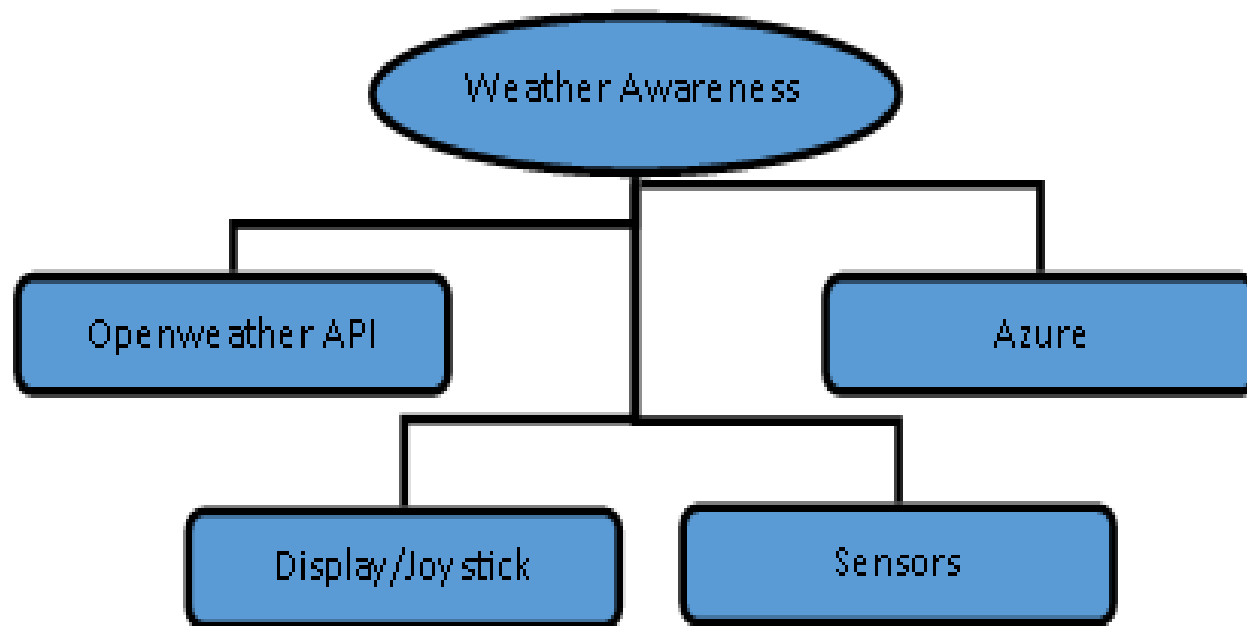
## Overview

This product is 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.

# App Dev

## Overview

- Communications in application





# App Dev

## User Experience / UI

- 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 API service.



# App Dev

## UWP Template

- Specific IoT Core version should be selected

## MainPage.xaml

- Create textblock for logging

## IO

- Write entries to log file and display on screen

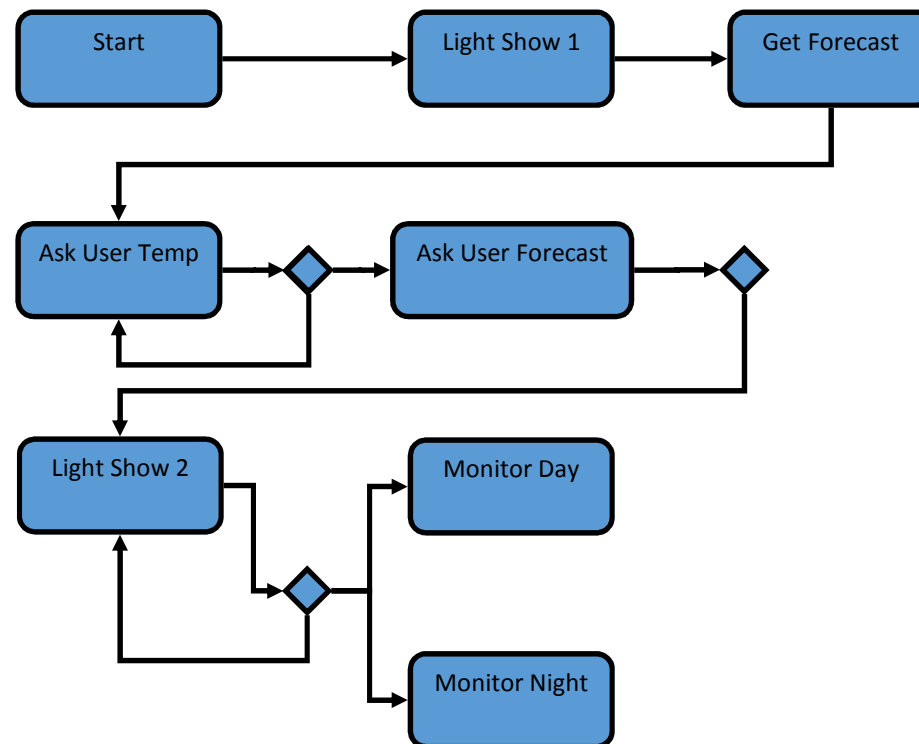
# App Dev

## 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 (**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 (**Digits, Symbols**) 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 (**Light show**) again which indicates that user has made both options and keep a log of what happened. Finally it continues monitor the ambient temperature (**Digits**) every 4 seconds, shows the result on display and keeps record in log file.

# App Dev

## Flow



# App Dev

## State Machine Implementation

- Timers

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

# App Dev

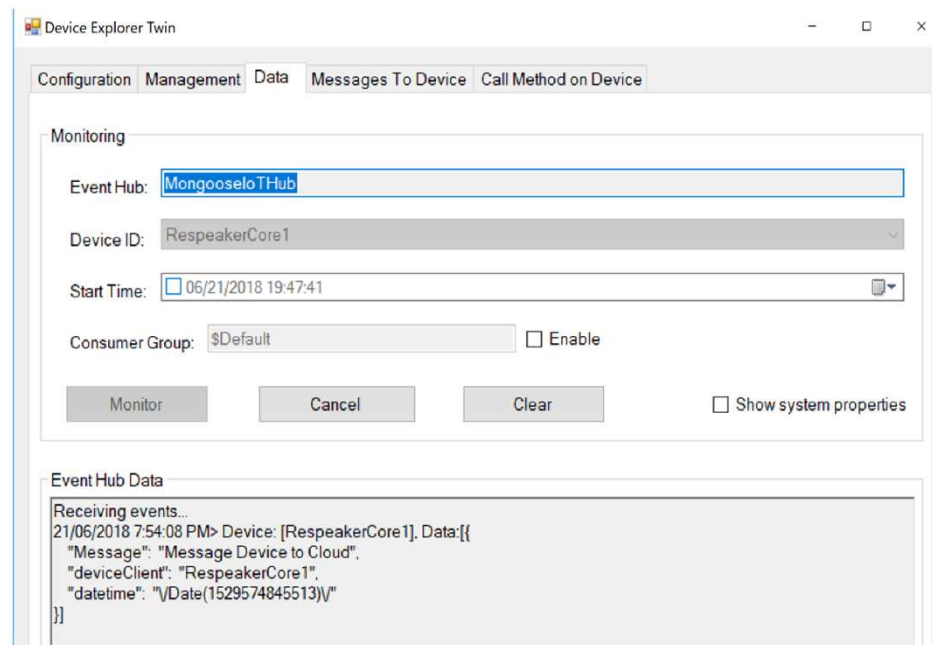
## State Machine Implementation

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

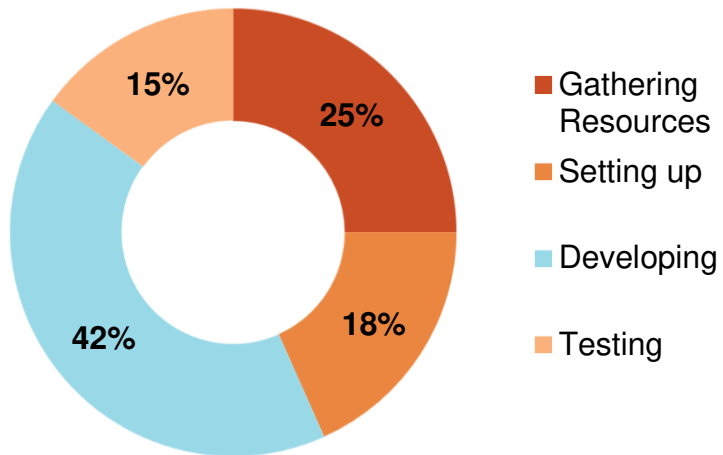
# Cloud

## Connect to Azure

- Provision the Raspberry Pi 3 to mate the device twin created in cloud. Modify server connect file to achieve sending and getting message from Azure. Monitor data transmit in Device Explorer.



# Schedule



## Approximate Time Cost

- Mid of Sep. to Start of Nov.
- Graph shown is estimation.
  - There is a newly IoT Core version released in Oct. Fortunately, most of old code can apply to latest version.



# Conclusion

Special thanks to Bassam and Anirban who provide good suggestions and necessary support to this project.

## Resources Incorporation

- There are limited related materials (comparing with other system) online. It hard to fix some bugs during development.
- Most of the ideas are borrowed from existing materials and code. It takes time to modify and combine them.
- Thanks to all the contributors who provide the resources.

## Unexpected Problems

- Win 10 IoT Core system is still under development, and doesn't provide full features to Raspberry Pi. It is important to read the release notes of each version beforehand.
- According to above mentioned, it is killing to take long time to settle tiny problem which can be easily solved by other platform.

# Further Work

## Design

Gain access to test other sensors and interact with user

Improve connection to cloud server (just simple test now)

Migrate current system to widely used platform (linux+AWS)

## User Experience


Improve current work flow and UI (speed up)

Ask user to input data through Joystick

Implement remote desktop monitoring

## Concerns

Find a proper method to compensate the higher ambient temperature detected by sensor



**Thanks!**  
**Q & A**

