2017

Microsoft IoT Workshop Labs Guide

March 25, 2017

SHAW CHYN CHIA
@snakechia
Microsoft MVP : Windows Development
MCT

Introduction

The goal of this labs exercise is to familiarize you with some of the components and technologies associated with Internet of Thing (IoT). Along the way, you will experience deploying code, streaming sensor data to Microsoft Azure, aggregating data with Stream Analytics and reporting with Microsoft Power BI.

Hardware Needed:

- Development Computer
- Raspberry Pi 2 or 3 (highly recommended Raspberry Pi 3)
- Sense HAT for Raspberry Pi

Software Required:

- Computer with Microsoft Windows 10 Anniversary Update (update 1607 or build 14393.xxx)

- Microsoft Visual Studio 2015 Community Edition with Update 3 or above.
  "**Universal Windows App Development Tool**" MUST be selected.

- Windows 10 IoT Core Dashboard
  http://go.microsoft.com/fwlink/?LinkID=708576

- Device Explorer
  https://github.com/Azure/azure-iot-sdks/releases (Scroll down for SetupDeviceExplorer.msi)

- Windows IoT Remote Client
  (Windows Store Apps)

Others

- Microsoft Azure Account
  https://azure.microsoft.com/

- Microsoft Power BI access
  http://www.powerbi.com

Table of Contents

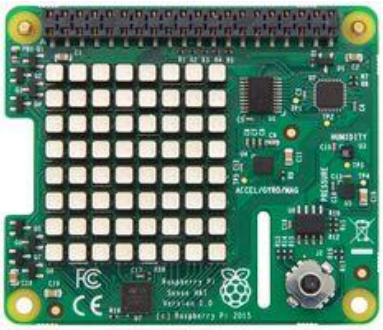
Mini Weather Station with Sense HAT and Raspberry Pi 2/3

Sense HAT is attached on top of the Raspberry Pi via the 40 GPIO pins. The Sense HAT has several integrated circuit based sensors, such as: Gyroscope, Accelerometer, Magnetometer, Barometer, Temperature, Humidity sensors, it also equipped with 8x8 LED matrix display and a small 5 button joystick.

In this Lab, we are going to read the Temperature, Humidity sensors values, display on the 8x8 LED matrix as well as output to screen if the device is connected to a screen.

Components needed:

- Sense HAT



What we going to achieve in this Lab:

1. Build a Sensor Model for our Mini Weather Station.
2. Read values from Sense HAT and put into our sensor model.
3. Display sensor data on the device screen output.
4. Display sensor data on Sense HAT 8x8 LED matrix.

Building the Weather Station

Before we can build our Weather Station, some ground work need to be done. Let's install RPi.SenseHat and Json.NET library from nuget

1. Json.Net

```
PM> Install-Package Newtonsoft.Json
```

2. RPi.SenseHat

```
PM> Install-Package Emmellsoft.IoT.RPi.SenseHat
```

3. Azure.Devices.Client

```
PM> Install-Package Microsoft.Azure.Devices.Client
```

Construct Sensor Data Model

As we know that, Sense HAT for Raspberry Pi is going to return us various of sensors data, our goal is just to have Temperature, Pressure and Humidity data, hence our data model will build around it.

```
public class SensorData
{
    public double temperature { get; set; }
    public double pressure { get; set; }
    public double humidity { get; set; }
    public DateTime createdAt { get; set; }
}
```

Build our own StringFormatConverter

The current version of XAML for UWP lack of **StringFormatConverter**, we need to write our own code for it. This is pretty much simple. The future release of UWP will have built in **StringFormatConverter**. The StringFormatConverter code as follow:

```
public class StringFormatConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter,
            string language)
    {
        if (value == null)
            return null;

        if (parameter == null)
            return value;

        return string.Format((string)parameter, value);
    }

    public object ConvertBack(object value, Type targetType, object parameter,
            string language)
    {
        throw new NotImplementedException();
    }
}
```

Building the UI

1. Open MainPage.xaml.

2. Insert the following code before
   `<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">`

```
<Page.Resources>
    <local:StringFormatConverter x:Key="StringFormatConverter" />
</Page.Resources>
```

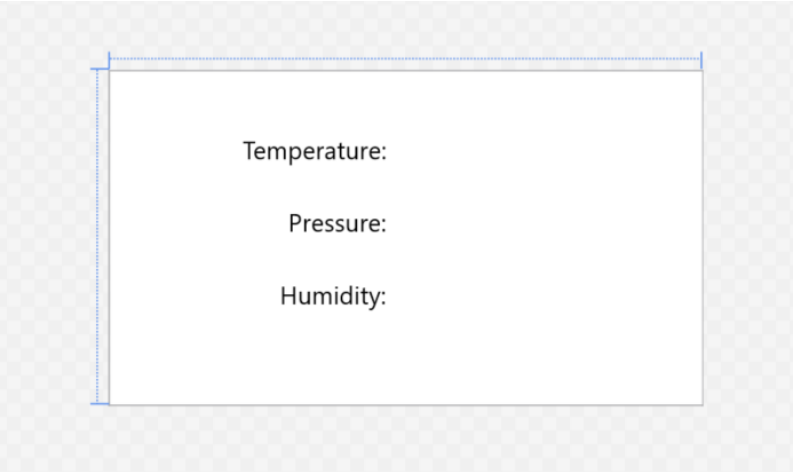
3. Replace
   `<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">` with
   `<Grid Name="dataGrid" Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">`

4. Insert the following code between

```
<StackPanel VerticalAlignment="Center">
    <TextBlock Text="Mini Weather Station" FontSize="24" HorizontalAlignment="Center"
               Margin="24" />
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <TextBlock Text="Temperature:" HorizontalAlignment="Right" Margin="12" />
        <TextBlock Text="{Binding temperature,
                Converter={StaticResource StringFormatConverter},
                ConverterParameter='{}{0:0.00 C}'}" Grid.Column="1"
                Margin="12" FontWeight="Thin" />
    </Grid>

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <TextBlock Text="Pressure:" HorizontalAlignment="Right" Margin="12" />
        <TextBlock Text="{Binding pressure,
                Converter={StaticResource StringFormatConverter},
                ConverterParameter='{}{0:0.000 Pa}'}" Grid.Column="1"
                Margin="12" FontWeight="Thin" />
    </Grid>

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>

        <TextBlock Text="Humidity:" HorizontalAlignment="Right" Margin="12" />
        <TextBlock Text="{Binding humidity,
            Converter={StaticResource StringFormatConverter},
            ConverterParameter='{}{0:0}'}" Grid.Column="1" Margin="12"
            FontWeight="Thin" />
    </Grid>
```

```
    <TextBlock Text="{Binding createdAt,
        Converter={StaticResource StringFormatConverter},
        ConverterParameter='{}{0:HH:mm:ss}'}" HorizontalAlignment="Center"
        Margin="24,24,24,6" />
    <TextBlock Text="{Binding createdAt,
        Converter={StaticResource StringFormatConverter},
        ConverterParameter='{}{0:MMM dd, yyyy}'}" HorizontalAlignment="Center"
        Margin="24,6,24,24" />
</StackPanel>
```

This is how the UI look like in Visual Studio



Writing the Code in MainPage.xaml.cs

1. Open MainPage.xaml.cs

2. Define the following objects.

```
using Emmellsoft.IoT.Rpi.SenseHat;

ISenseHat senseHat;
SensorData sensordata = new SensorData();
DispatcherTimer timer;

DeviceClient deviceClient;
string deviceName = "<Your device name>";
string deviceconnectionstring = "<Your device connection string>";
```

3. Put the following code into `public MainPage()`

```
timer = new DispatcherTimer();
timer.Interval = TimeSpan.FromSeconds(1);
timer.Tick += Timer_Tick;

deviceClient = DeviceClient.CreateFromConnectionString(deviceconnectionstring);
```

4. To initialize Sense HAT, we have to use “Try Catch method” to prevent the app crash if there is no Sense HAT installed. Insert the following code

```
protected override async void OnNavigatedTo(NavigationEventArgs e)
{
    try
    {
        senseHat = await SenseHatFactory.GetSenseHat();
        timer.Start();
    }
    catch
    {
        return;
    }
}
```

5. Define the timer Tick event.

```
private void Timer_Tick(object sender, object e)
{
    getsensordata();
}
```

6. Read various sensors data and display it to device screen output.

```
private void getsensordata()
{
    senseHat.Sensors.HumiditySensor.Update();
    senseHat.Sensors.PressureSensor.Update();

    sensordata.temperature = (double)senseHat.Sensors.Temperature;
    sensordata.pressure = (double)senseHat.Sensors.Pressure;
    sensordata.humidity = (double)senseHat.Sensors.Humidity;
    sensordata.createdAt = DateTime.UtcNow;

    // this simple app we do not implement NotifyPropertyChanged
    // hence we manually refreshed the databinding.
    dataGrid.DataContext = null;
    dataGrid.DataContext = sensordata;
}
```

7. You may deploy the application to your Raspberry Pi 2/3 and use **Windows IoT Remote Client** to see the output screen.

8. To be able to display the information on the 8x8 LED matrix on Sense HAT, we need to define the following:

```
using Emmellsoft.IoT.Rpi.SenseHat.Fonts.SingleColor;

ISenseHatDisplay display;
TinyFont tinyFont = new TinyFont();
int count = 0;
```

9. The following function is to display temperature and humidity value from Sense HAT sensors, with 5 second interval.

```
private void displayDataOn8x8()
{
    double value = sensordata.temperature;
    Color color = Colors.Blue;

    switch (count)
    {
        case 1:
            value = double.Parse(senseHat.Sensors.Humidity.ToString());
            color = Colors.Red;
            break;

        default:
            value = sensordata.temperature;
            color = Colors.Blue;
            break;
    }

    display = senseHat.Display;
    display.Clear();
    tinyFont.Write(display, ((int)Math.Round(value)).ToString(), color);
    display.Update();

    count++;

    // Reset counter to 0 so that the LED matrix can show temperature value
    if (count > 1)
    {
        count = 0;
    }
}
```

10. Call `displayDataOn8x8()` function from `getsensordata()` by adding `displayDataOn8x8()` to the last line of `getsensordata()` function.

11. Create a function to send the data to Azure IoT Hub.

```
private async Task sendDataToAzureIoTHub(string message)
{
    try
    {
        var msg = new Message(Encoding.UTF8.GetBytes(message));
        await deviceClient.SendEventAsync(msg);
    }
    catch
    { }
}
```

12. Finally, we need to convert the **sensordata** to a `string`, so that it can be send to Azure IoT Hub. Insert the following code into end of the `getsensordata()` section. It will send the data to cloud every 5 seconds.

```
sensordata.deviceId = deviceName;
string message = JsonConvert.SerializeObject(sensordata);
sendDataToAzureIoTHub(message);
```

13. Next, we are going to add the codes that enable the device to receive data from Azure IoT Hub. To achieve this task, we are going to add the following function:

```
public async Task ReceiveDataFromAzureIoTHub()
{
    try
    {
        Message receivedMessage;
        string messageData;

        while (true)
        {
            receivedMessage = await deviceClient.ReceiveAsync();
            if (receivedMessage != null)
            {
                messageData = Encoding.ASCII.GetString(receivedMessage.GetBytes());
                await deviceClient.CompleteAsync(receivedMessage);

                if (messageData.Length > 2)
                    messageData = messageData.SubString(0, 2);

                timer.Stop();

                display.Clear();
                tinyFont.Write(display, messageData, Colors.Green);
                display.Update();

                timer.Start();
            }
        }
    }
    catch
    { }
}
```

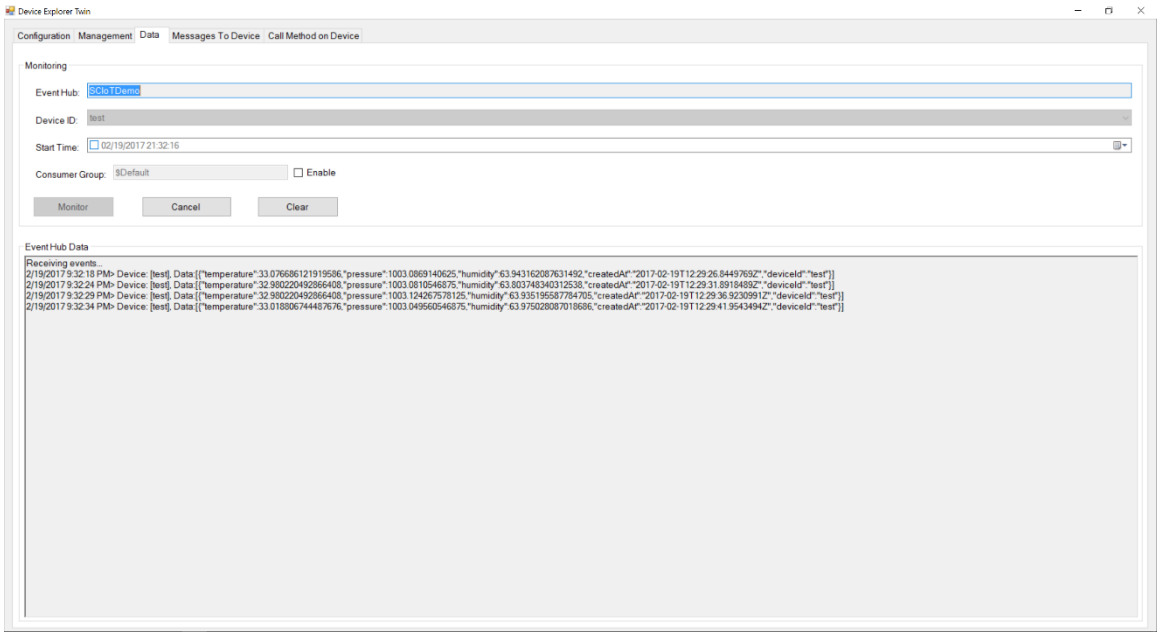
14. The timer will stop for the 8x8 LED matrix to display incoming message, after that it will be restarted. As for the result, the incoming message will be display for about 5 second before it being updated by temperature and humidity info.

15. Due to the TinyFont, it only display HEX number and it can't display the message longer than 2 characters, as for the result, we only take the first 2 characters for display.

16. Deploy this application to Raspberry Pi3 device.

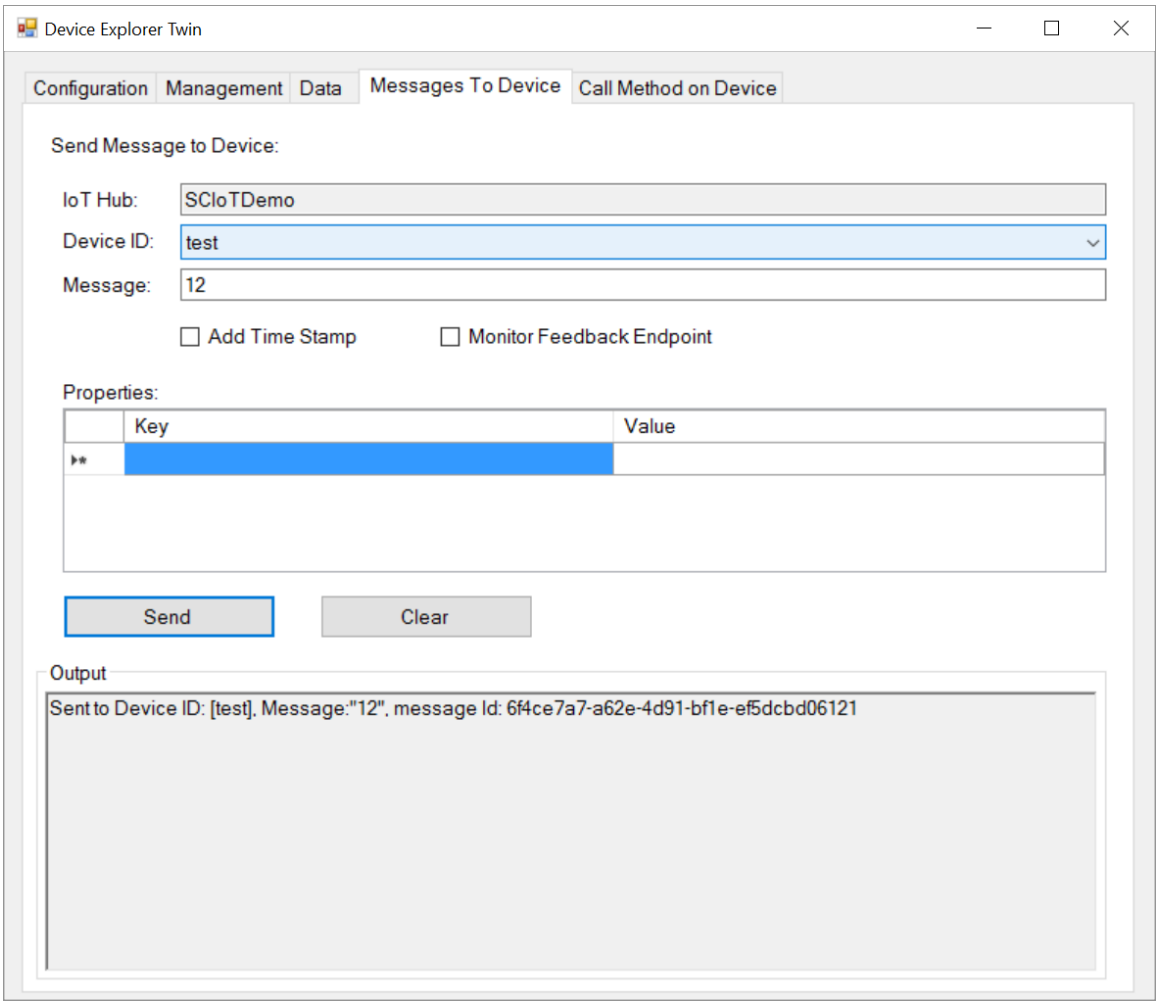
17. Press the Windows key and type "**Device Explorer**" and run the app.
    Select the "**Data**" tab, follow by choose your device name at "**Device ID**" section and click "**Monitor**" button.

18. You should be able to see the data flowing in.



19. You may now re-deploy the application to your Raspberry Pi 2/3 and you should see the LED matrix will start display value and switch every 5 seconds.

20. Navigate to “Message To Device” tab in “Device Explorer”.
    Enter the your message in the Message Box, and press “Send”. Once the message been sent, you will receive the message ID.



Congratulation! You just make your weather station an IoT device. :)

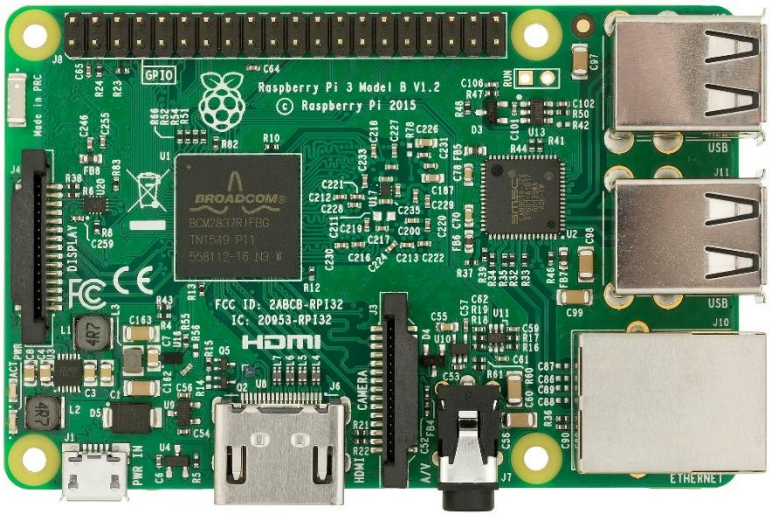
Note:

You may continue to explore other sensors available in Sense HAT by typing “*sensehat.sensor.*” you will notice there are more sensors available.

Appendix

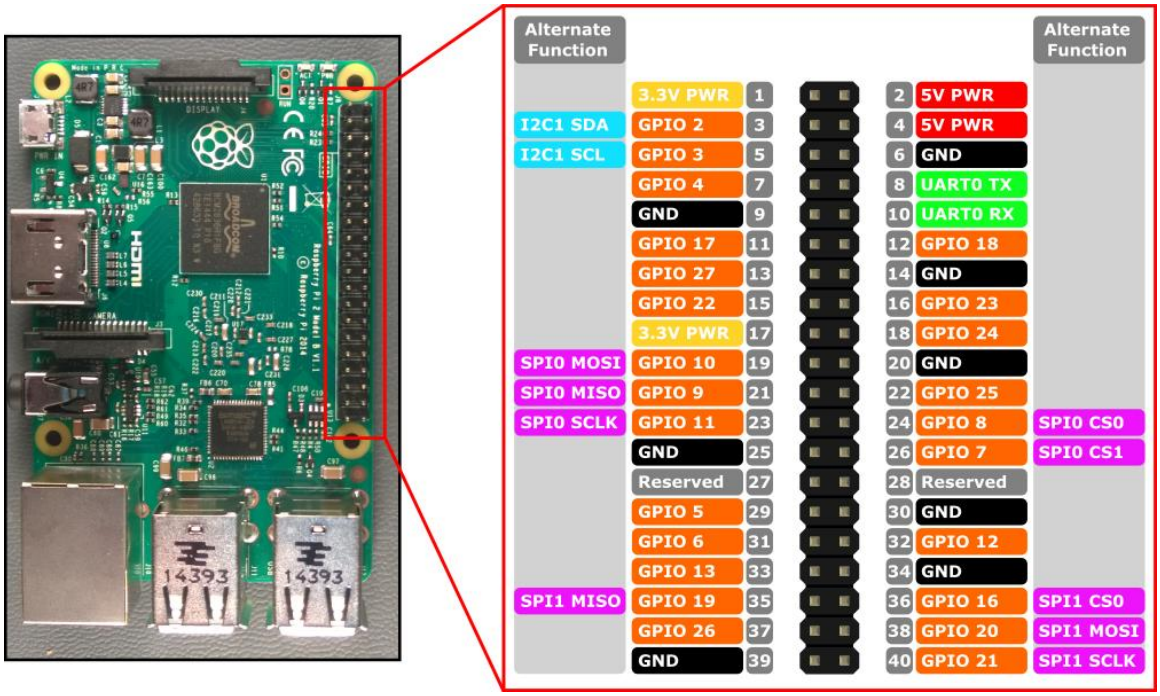
Raspberry Pi 2/3 Specification

	Model B Gen 2	Model B Gen 3
Release date	February 2015	February 2016
Architecture	ARM v7-A (32-bit)	ARM v80A (64-bit/32-bit)
SoC	Broadcom BCM2836	Broadcom BCM2837
CPU	900 MHz 32-bit quad-core ARM Cortex-A7	1.2 GHz 64-bit quad-core ARM Cortex-A53
GPU	Broadcom VideoCore IV @ 250 MHz	
Memory	1 GB (shared with GPU)	
USB 2.0 ports	4 (via the on-board 5-port USB hub	
Video Input	15-pin MIPI camera interface (CSI) connector, used with the Raspberry Pi camera or Raspberry Pi NoIR camera	
Video Output	HDMI (rev 1.3), composite video (3.5 mm TRRS jack), MIPI display interface (DSI) for raw LCD panels	
Audio Input	via I^2S	
Audio Output	Analog via 3.5 mm phone jack; digital via HDMI	
On-board Storage	MicroSDHC slot	MicroSDHC slot, USB Boot Mode
Onboard Network	10/100 Mbit/s Ethernet (8P8C) USB adapter on the USB hub	10/100 Mbit/s Ethernet, 802.11n wireless, Bluetooth 4.1
Low Level Peripherals	17× GPIO plus the same specific functions, and HAT ID bus	
Power Ratings	800 mA (4.0 W)	
Power Source	5 V via MicroUSB or GPIO header	
Size	85.60 mm × 56.5 mm (3.370 in × 2.224 in)	
Weight	45 g (1.6 oz)	
Console	Micro-USB cable or a serial cable with optional GPIO power connector	



Raspberry Pi 3

Raspberry Pi 2/3 GPIO Layout



SenseHat Fact Sheet

The Raspberry Pi Sense HAT is attached on top of the Raspberry Pi via the 40 GPIO pins (which provide the data and power interface) to create an 'Astro Pi'. The Sense HAT has several integrated circuit based sensors that you can use for many different types of experiments, applications, and even games.

Technical Specification

- Gyroscope – angular rate sensor: ±245/500/2000dps
- Accelerometer - Linear acceleration sensor: ±2/4/8/16 g
- Magnetometer - Magnetic Sensor: ±4/8/12/16 gauss
- Barometer: 260 – 1260 hPa absolute range (accuracy depends on the temperature and pressure, ±0.1 hPa under normal conditions)
- Temperature sensor (Temperature accurate to ± 2 °C in the 0-65 °C range)
- Relative Humidity sensor (accurate to ±4.5% in the 20-80%rH range, accurate to ± 0.5 °C in 1540 °C range)
- 8x8 LED matrix display
- Small 5 button joystick

Others

Useful Network Commands

From PowerShell

- netsh wlan show profile
- netsh wlan add profile *Wi-Fi-ProfileName.xml*
- netsh wlan export profile key=clear
- netsh wlan delete profile *ProfileName*
- netsh wlan connect name=*ProfileName*
- netsh wlan show interfaces
- netsh interface ipv4 set dns “Wi-Fi” static <IP address>
- netsh interface ipv4 set address “Wi-Fi” static <IP address> <space> <subnet> <space> <gateway address>

Social

#windows10 #windows10iot #windows10iotcore #azure #azureiothub #iot #raspberrypi

Disclaimer

All care has been taken to ensure the accuracy of this document. No liability accepted.

Copyright