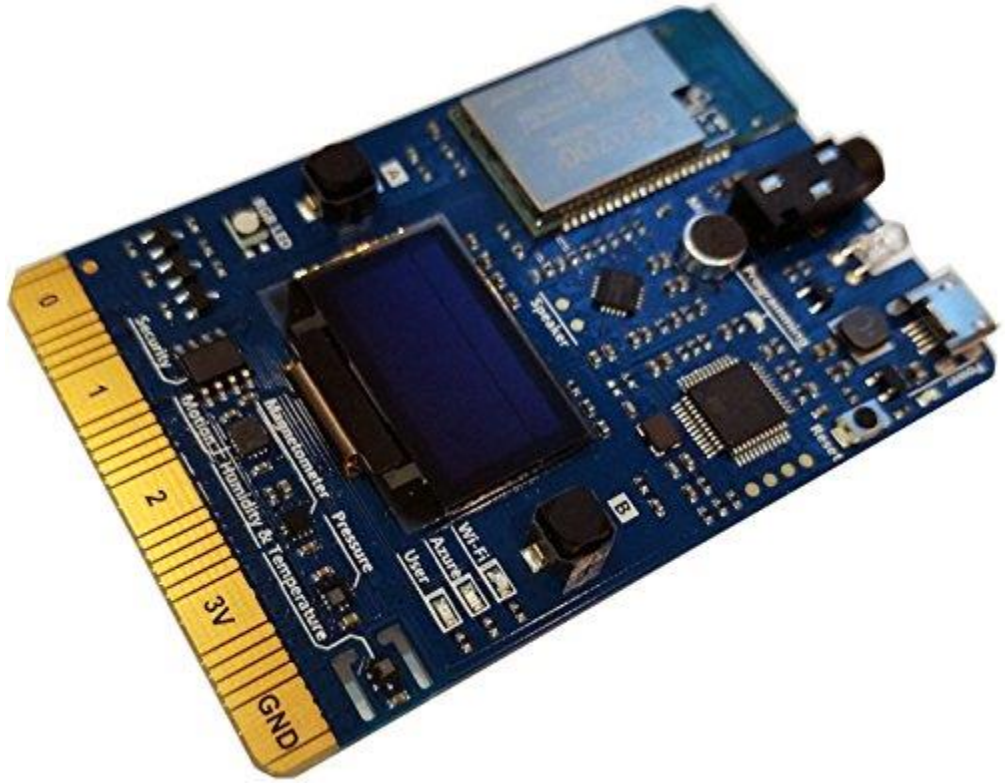
Monitoring IoT appliances in real time

Mark Allibone

Mobile Lead

Rey Automation

@mallibone

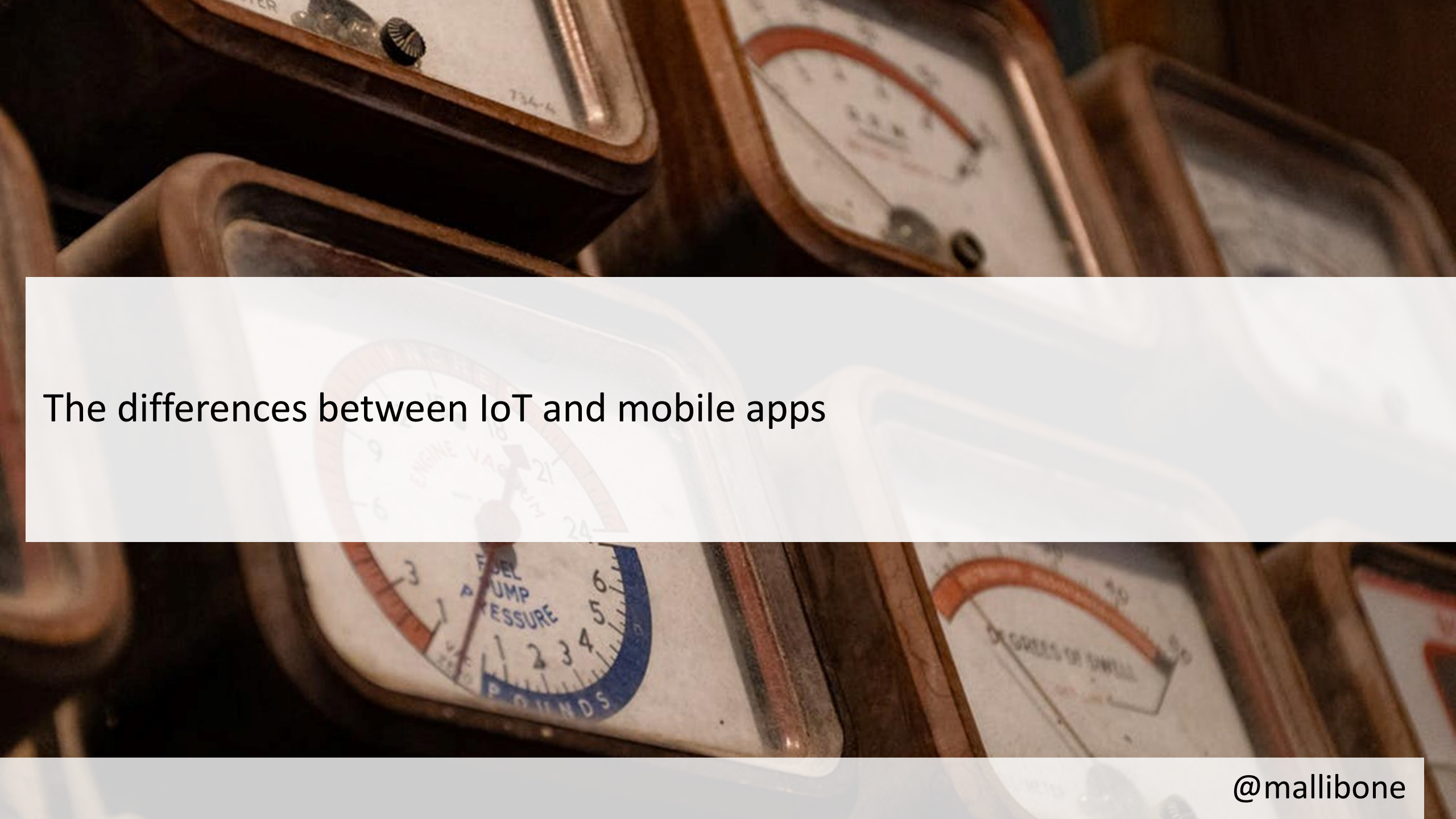


totalgadgetsite.com

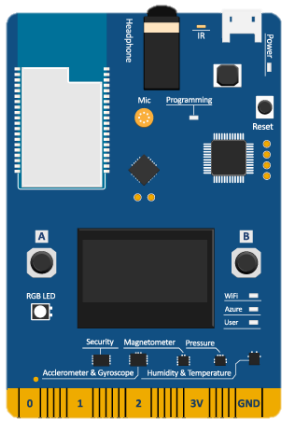
@mallibone

mscloud.be

@mallibone



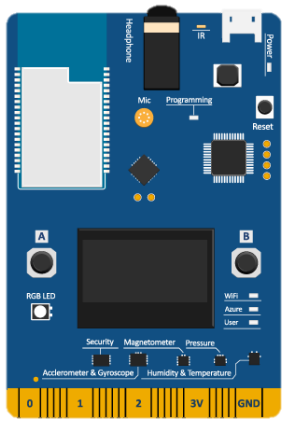
The differences between IoT and mobile apps


Headphone
IR
Power
Mic
Programming
Reset
A
B
RGB LED
WiFi
Azure
User
Security
Magnetometer
Pressure
Accelerometer & Gyroscope
Humidity & Temperature
0
1
2
3V
GND

?

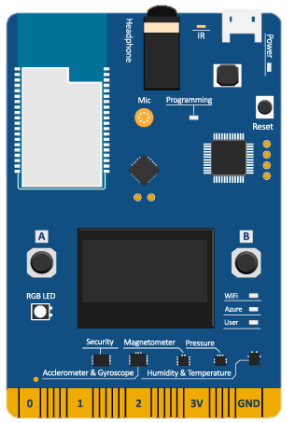
@mallibone

Bluetooth



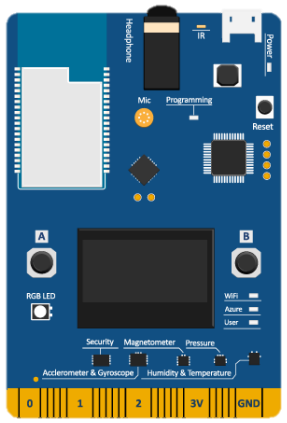
@mallibone

WiFi



@mallibone

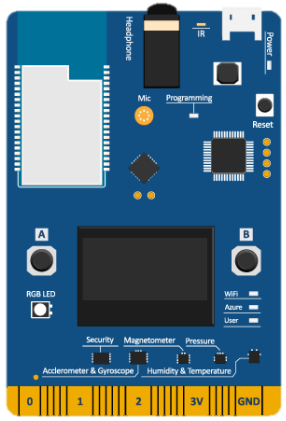
Backend



Probably something involving Azure

@mallibone

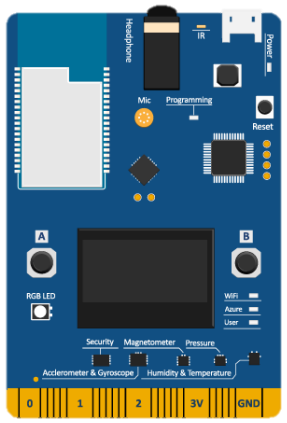

“To the cloud!”
@mallibone



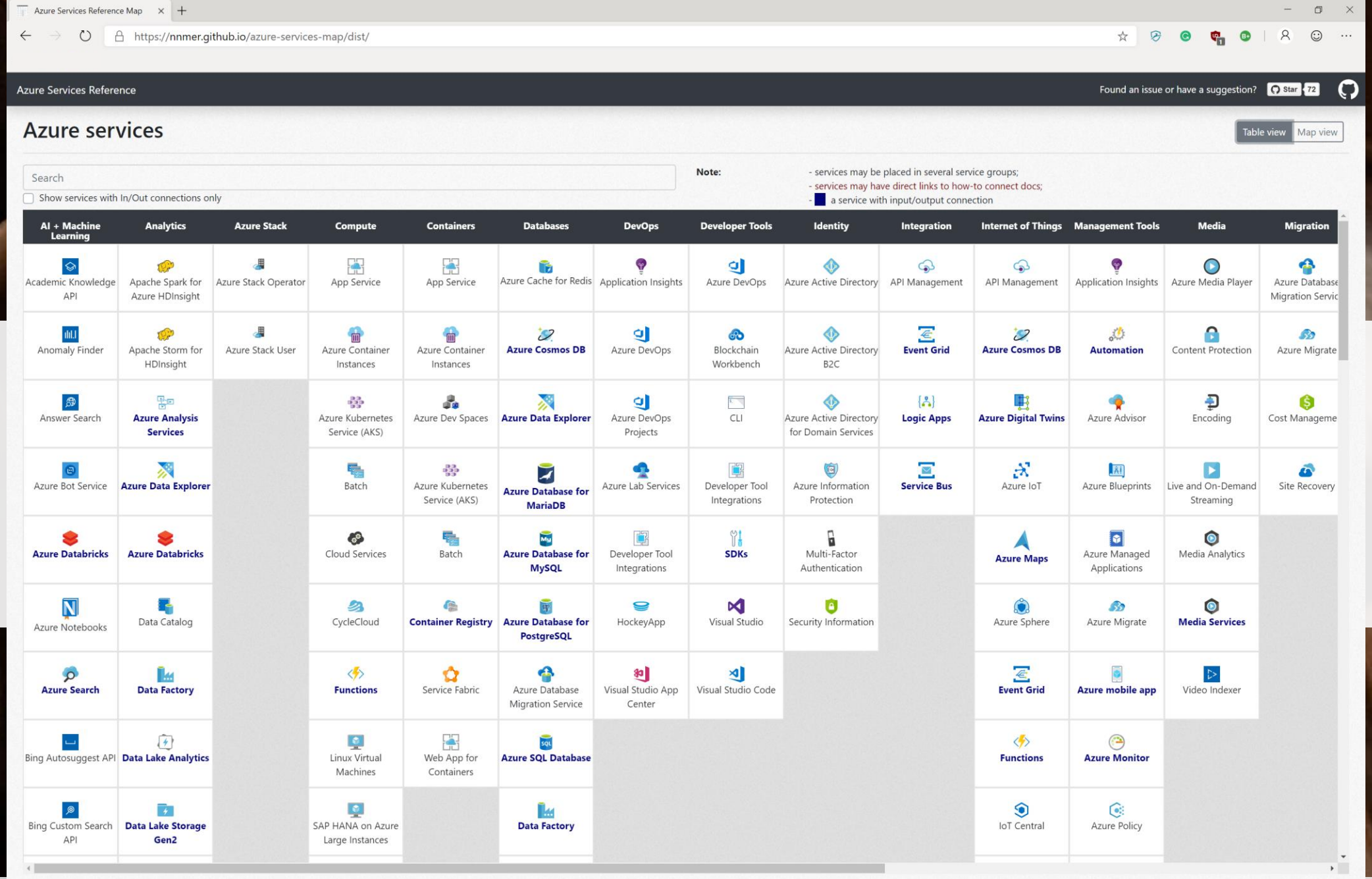
Azure

@mallibone

Where the magic happens



@mallibone



https://nnmer.github.io/azure-services-map/dist/

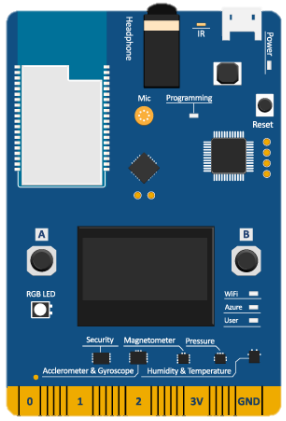
@mallibone

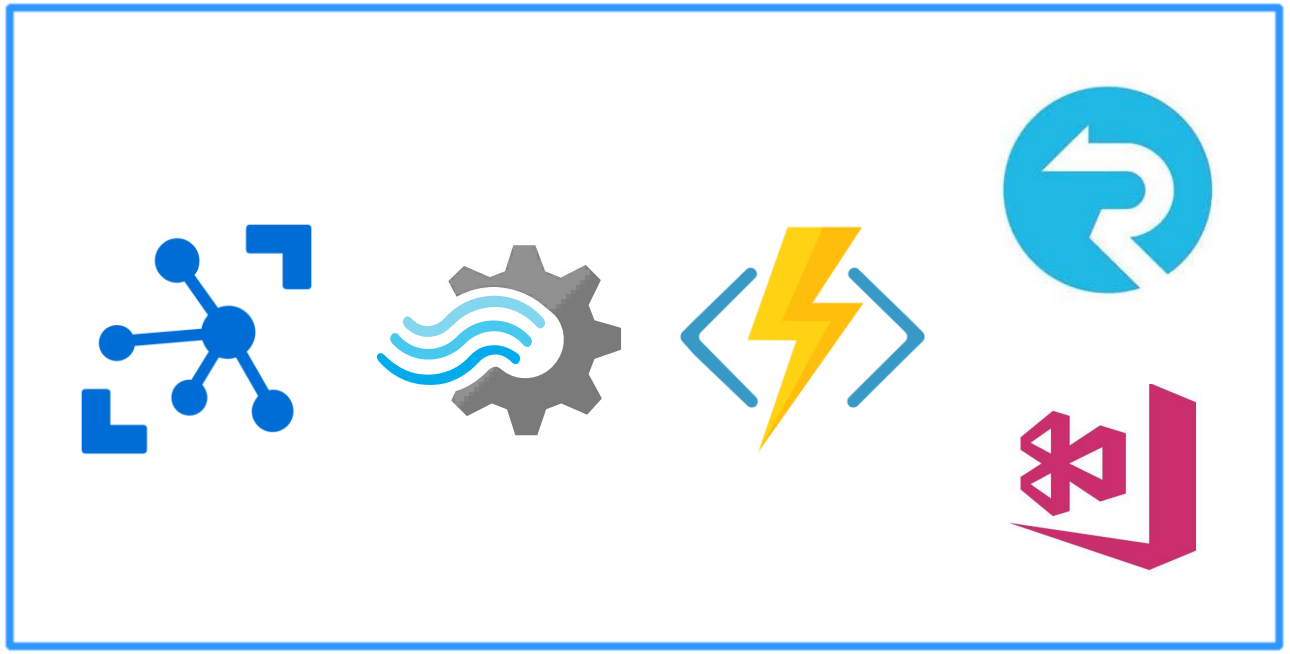
https://nnmer.github.io/azure-services-map/dist/

@mallibone

Azure Resources 101

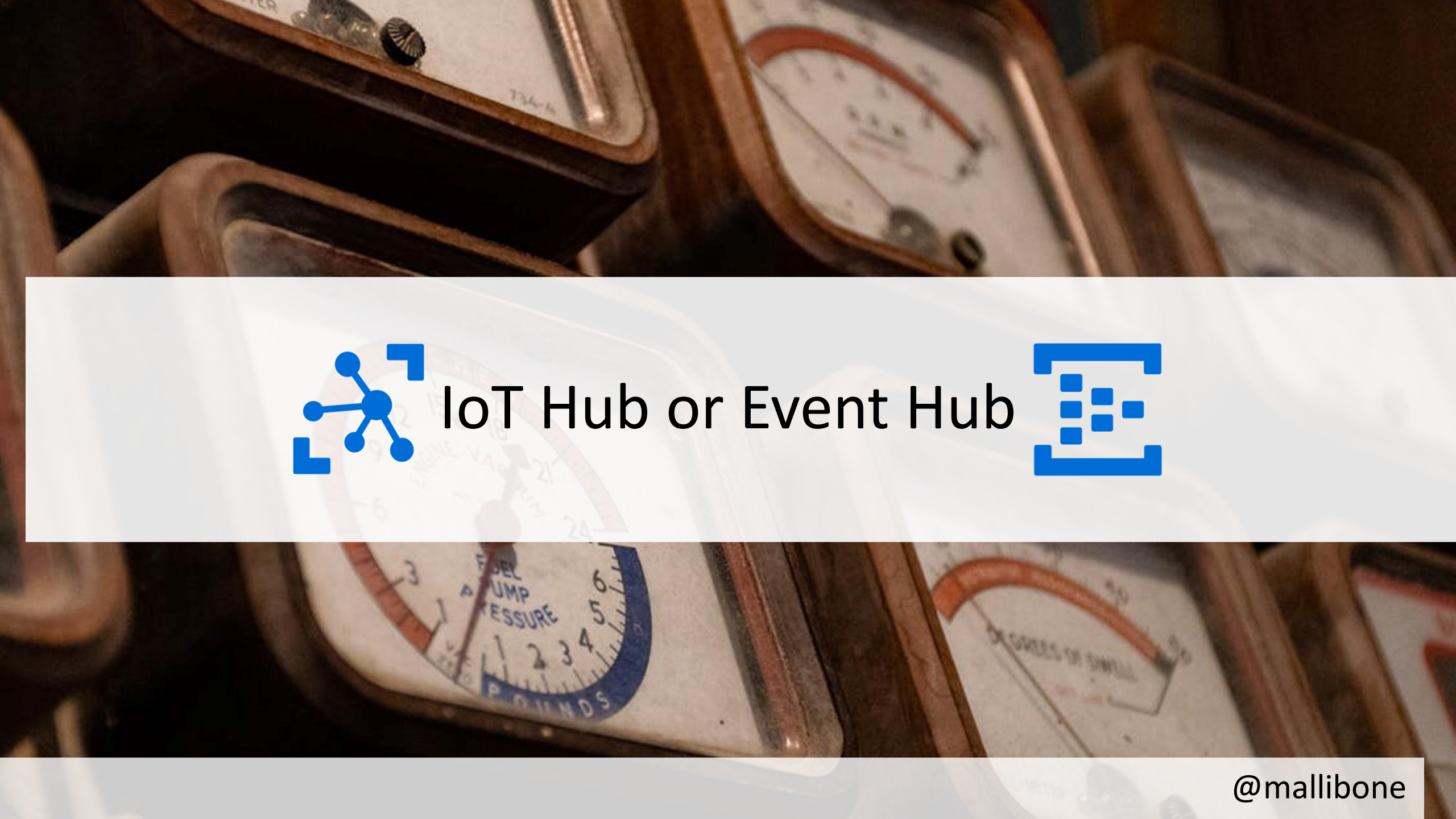
- Group your resources
- Create services in the same region/datacenter
- Check Pricing / Limitations





Azure Cloud

@mallibone


IoT Hub or Event Hub
@mallibone

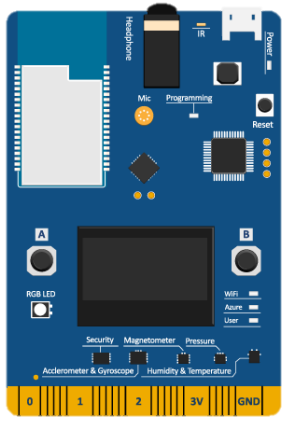
IoT Hub or Event Hub?

IoT Hub	**Event Hub**
Communication from Device to Cloud and Cloud to Device	**Communication** from Client to Cloud
Millions of **Concurrent Connections** (AMQP)	5000 Concurrent Connections (AMQP)
Integrated **Device Management**	No **Device Management**

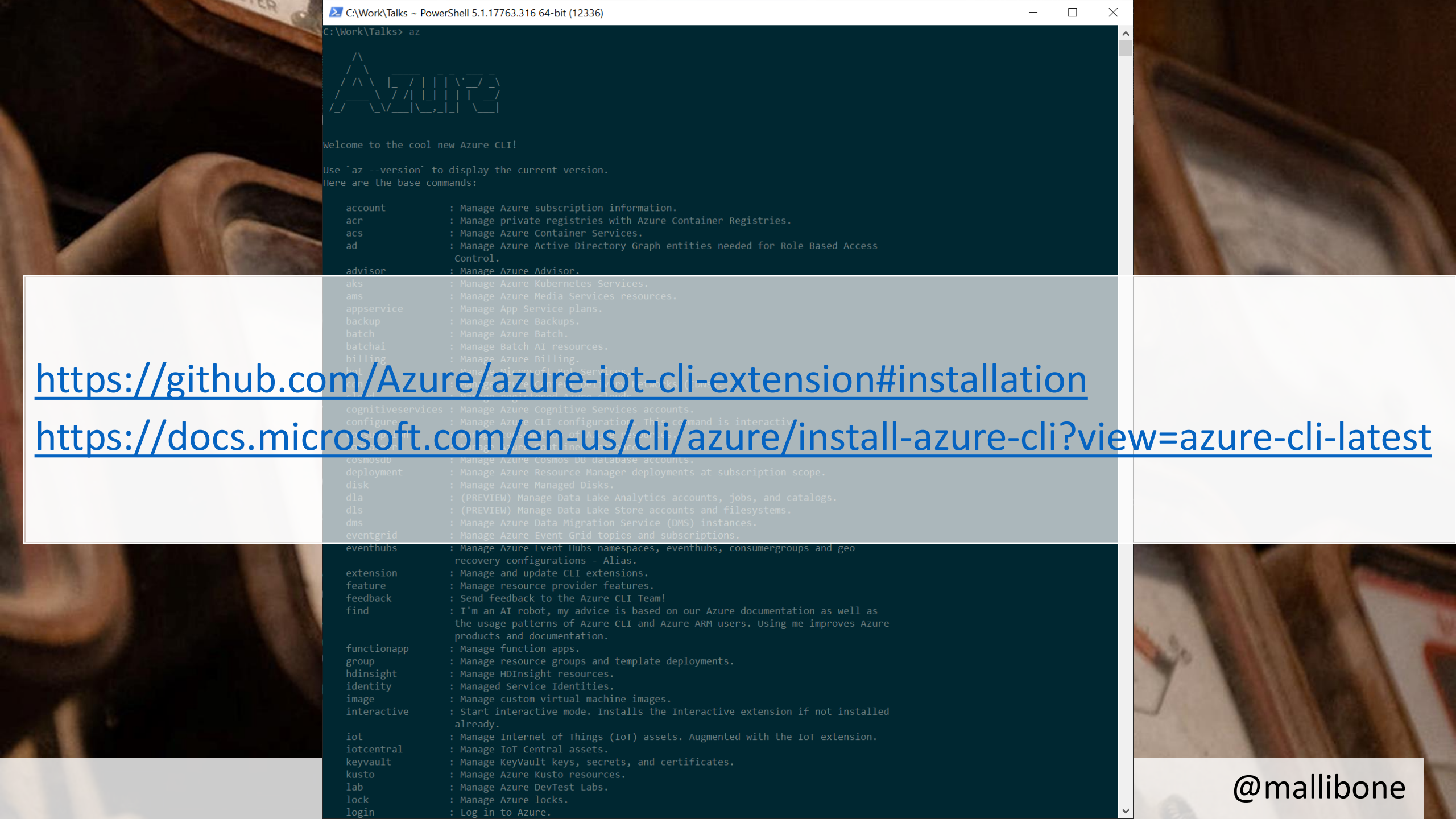
Credits to Thomas C. Huber

@mallibone

Azure Cloud



@mallibone



https://github.com/Azure/azure-iot-cli-extension#installation

https://docs.microsoft.com/en-us/cli/azure/install-azure-cli?view=azure-cli-latest

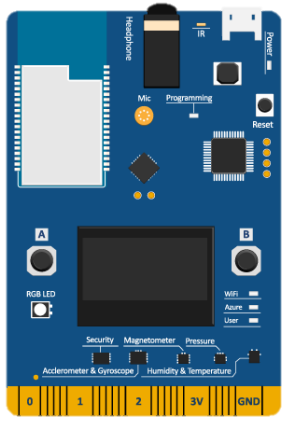
@mallibone

Azure Cloud

DEMO

@mallibone

Azure Cloud



@mallibone

IoT DevKit to cloud -- Connect I

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-arduino-iot-devkit-az3166-get-started

NoserEngineering

Microsoft Azure | Contact Sales: 1-800-867-1389 | Search | Portal

Overview | Solutions | Products | Documentation | Pricing | Training | Marketplace | Partners | Support | Blog | More | Free account

Azure / Internet of Things / IoT Hub

Feedback | Edit | Share | Theme | Sign in

Learn Azure at your own pace | See training modules

Filter by title

- IoT Hub Documentation
- Overview
- Quickstarts
- Tutorials
- Concepts
- How-to guides
  - Develop
  - Manage
  - Use real devices
    - Use an online simulator
    - Use a physical device
      - Raspberry Pi with Node.js
      - Raspberry Pi with C
      - MXChip IoT DevKit with Arduino
      - Adafruit Feather HUZZAH ESP8266 with Arduino
    - Use MXChip IoT DevKit
    - Extended IoT scenarios
  - Troubleshoot
- Reference
- Related
- Resources

Download PDF

Connect IoT DevKit AZ3166 to Azure IoT Hub

04/17/2019 • 12 minutes to read • Contributors all

IoT DevKit AZ3166 with VS Code

You can use the MXChip IoT DevKit to develop and prototype Internet of Things (IoT) solutions that take advantage of Microsoft Azure services. It includes an Arduino-compatible board with rich peripherals and sensors, an open-source board package, and a rich sample gallery.

What you learn

- How to create an IoT hub and register a device for the MXChip IoT DevKit.
- How to connect the IoT DevKit to Wi-Fi and configure the IoT Hub connection string.
- How to send the DevKit sensor telemetry data to your IoT hub.
- How to prepare the development environment and develop application for the IoT DevKit.

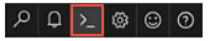
Don't have a DevKit yet? Try the DevKit simulator or purchase a DevKit.

What you need

- A MXChip IoT DevKit board with a Micro-USB cable. Get it now.
- A computer running Windows 10, macOS 10.10+ or Ubuntu 18.04+.
- An active Azure subscription. Activate a free 30-day trial Microsoft Azure account.

Open Azure Cloud Shell

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Select **Copy** to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

Select **Try It** in the upper-right corner of a code block.	
Open Cloud Shell in your browser.	
Select the **Cloud Shell** button on the menu in the upper-right corner of the Azure portal.	

In this article

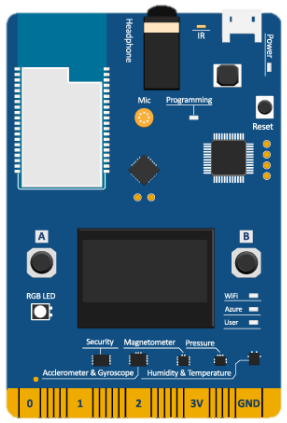
- What you learn
- What you need
- Open Azure Cloud Shell
- Prepare your hardware
- Quickstart: Send telemetry from DevKit to an IoT Hub
- Prepare the development environment
- Build your first project
- Test the project
- Review the code
- Problems and feedback
- Next steps

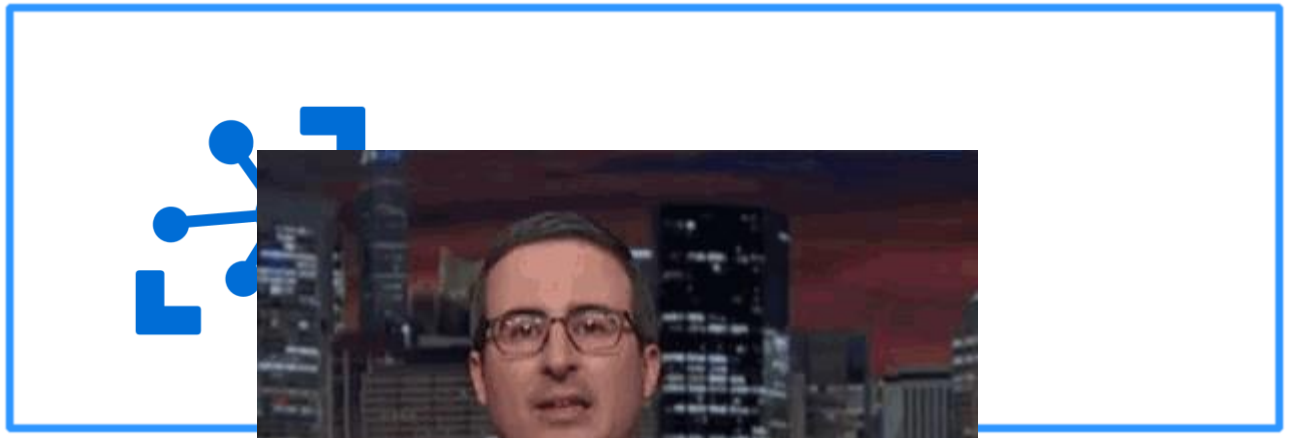
Is this page helpful? Yes No

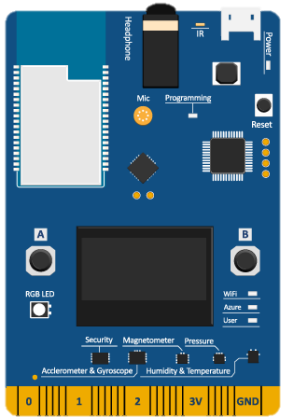
https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-arduino-iot-devkit-az3166-get-started

@mallibone

Azure Cloud

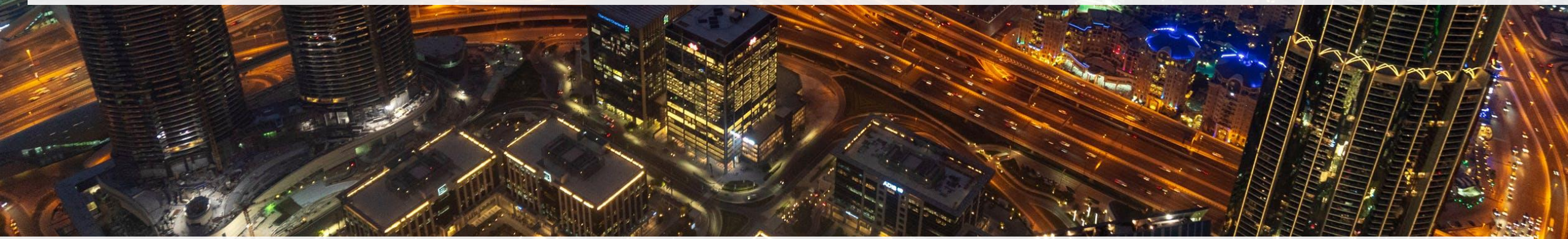

Headphone
IR
Power
Mic
Programming
Reset
A
B
RGB LED
WiFi
Azure
User
Security
Magnetometer
Pressure
Acclerometer & Gyroscope
Humidity & Temperature
0
1
2
3V
GND



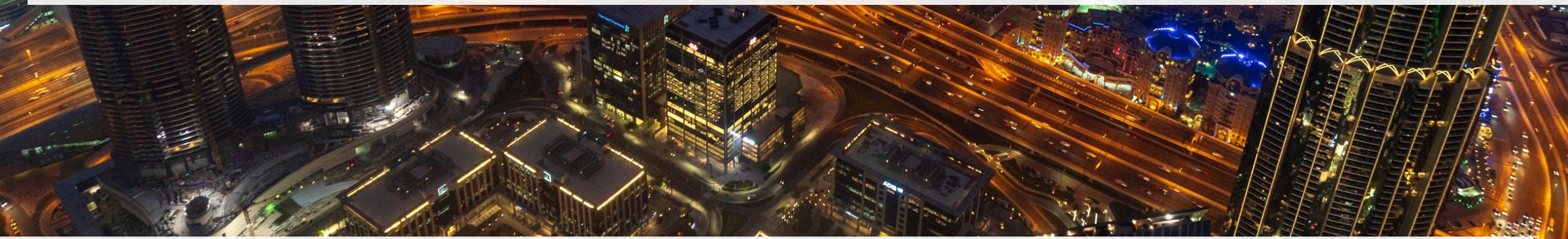

Headphone
IR
Power
Mic
Programming
Reset
A
B
RGB LED
WiFi
Azure
User
Security
Magnetometer
Pressure
Acclerometer & Gyroscope
Humidity & Temperature
0
1
2
3V
GND

@mallibone

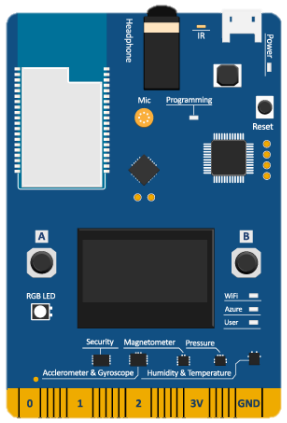
Getting the data to the Xamarin App

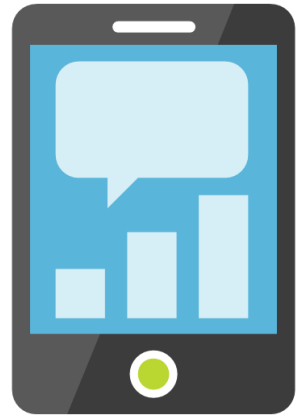


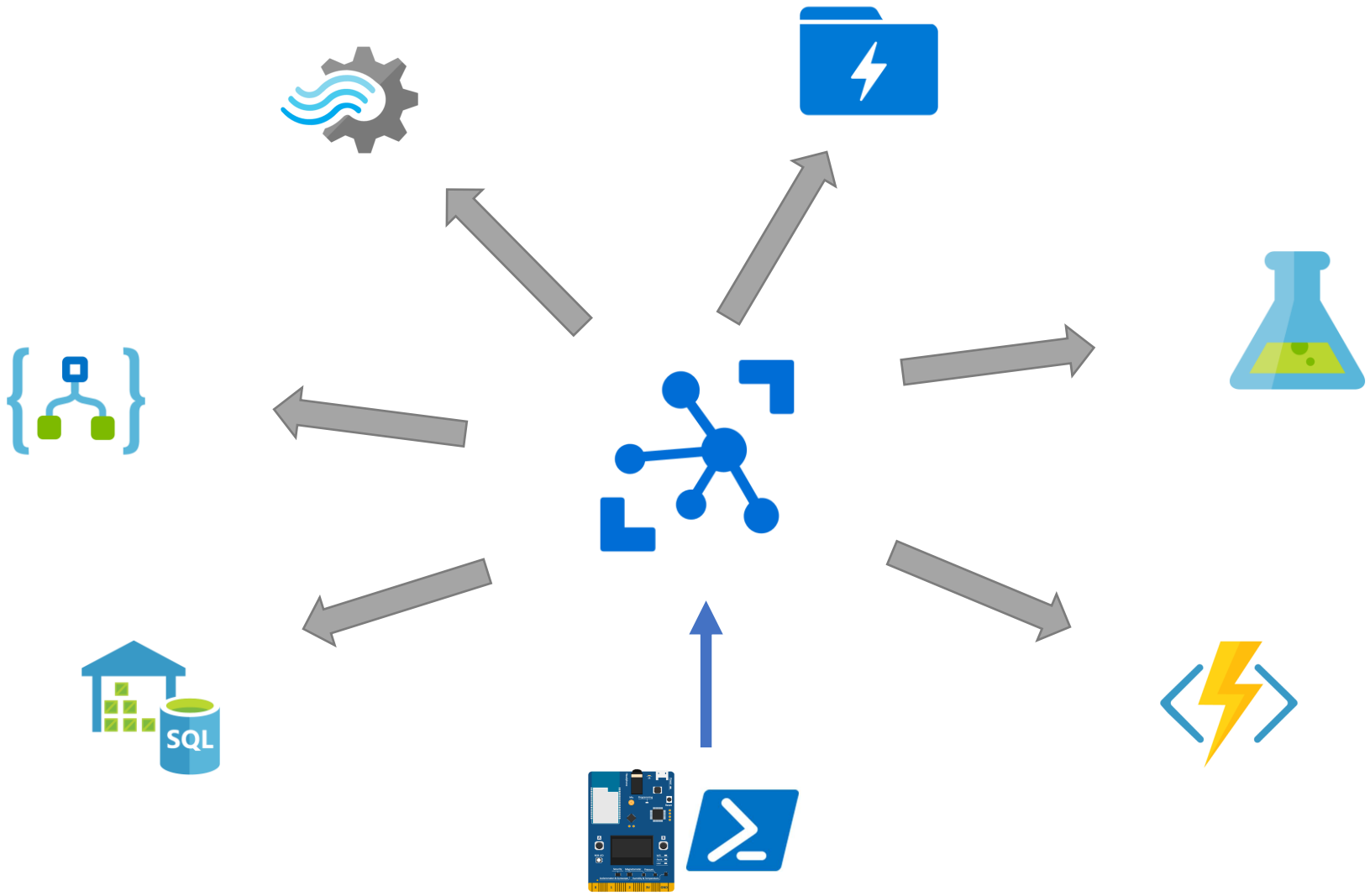
Poll vs Push



Azure Cloud

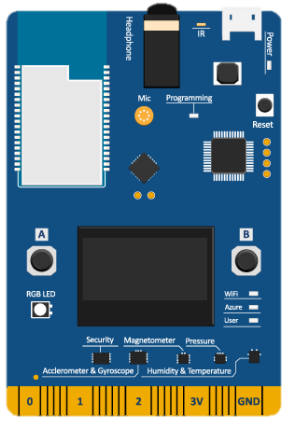

Headphone
IR
Power
Mic
Programming
Reset
A
B
RGB LED
WiFi
Azure
User
Security
Magnetometer
Pressure
Accelerometer & Gyroscope
Humidity & Temperature
0
1
2
3V
GND



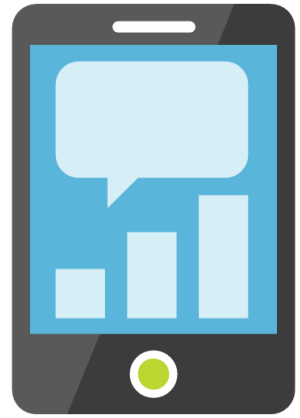


@mallibone

Processing IoT Hub data



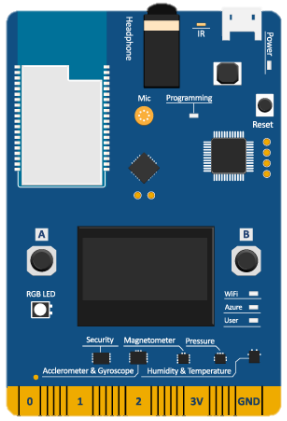
Azure Cloud



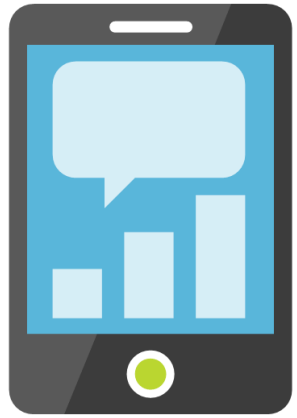
@mallibone

Azure Functions

@mallibone

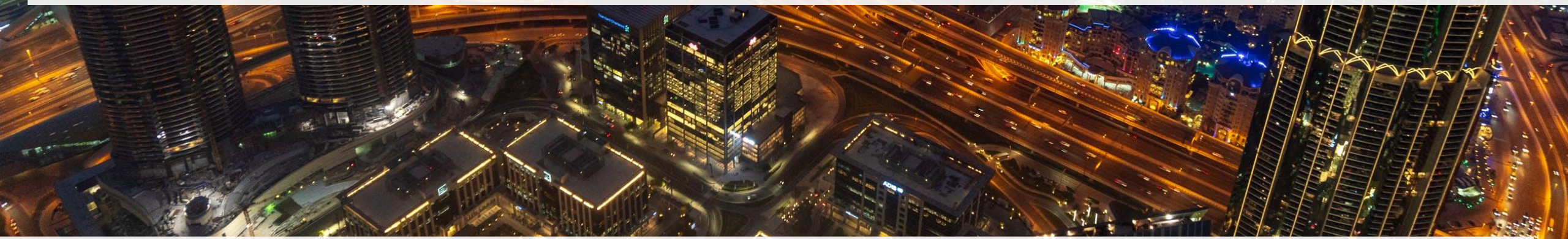


Azure Cloud

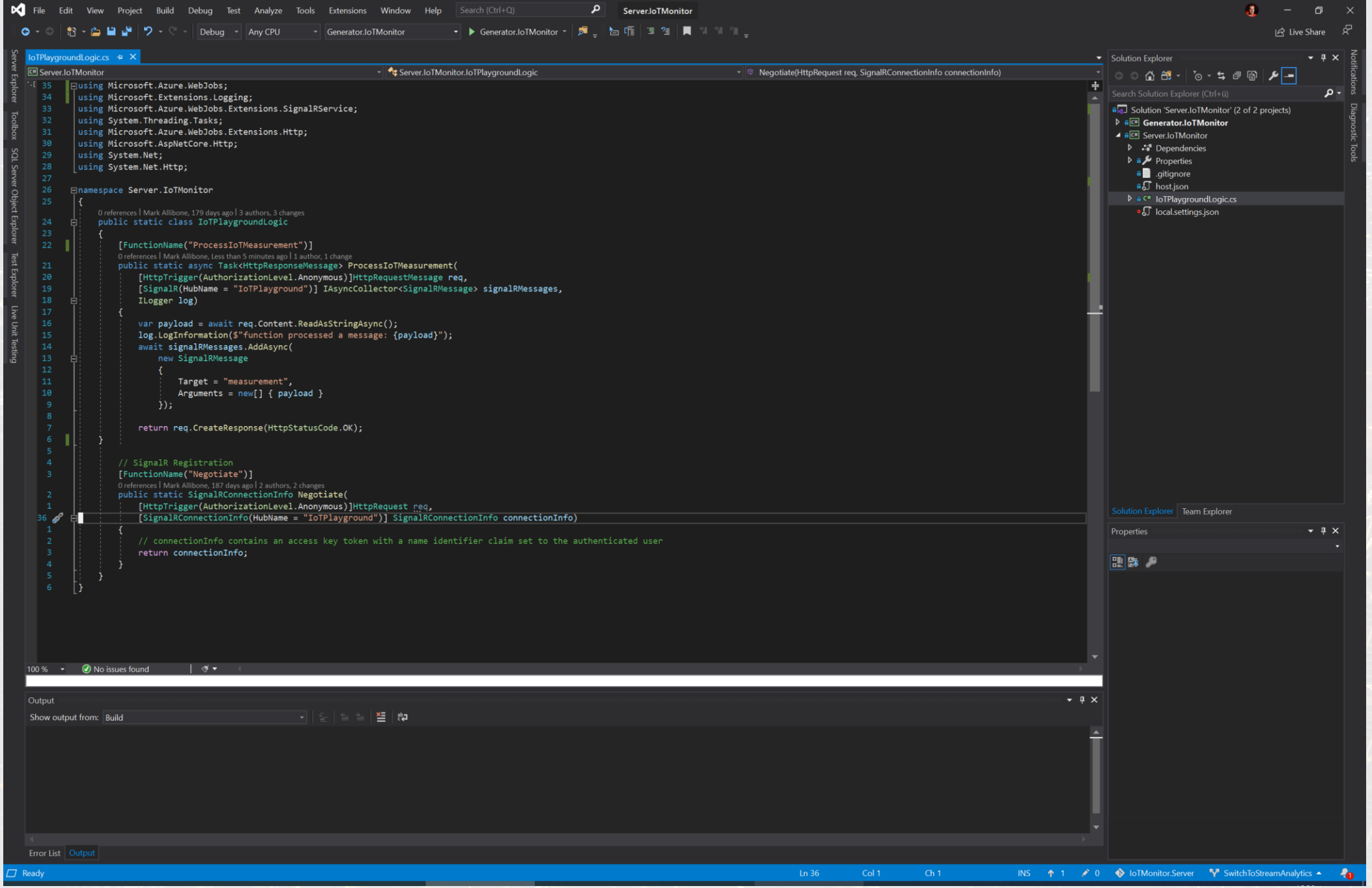


@mallibone

SignalR



@mallibone


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Server.IoTMonitor
Debug
Any CPU
Generator.IoTMonitor
Live Share
IoTPlaygroundLogic.cs
Server.IoTMonitor
Server.IoTMonitor.IoTPlaygroundLogic
Negotiate(HttpRequest req, SignalRConnectionInfo connectionInfo)
using Microsoft.Azure.WebJobs;
using Microsoft.Extensions.Logging;
using Microsoft.Azure.WebJobs.Extensions.SignalRService;
using System.Threading.Tasks;
using Microsoft.Azure.WebJobs.Extensions.Http;
using Microsoft.AspNetCore.Http;
using System.Net;
using System.Net.Http;
namespace Server.IoTMonitor
{
0 references | Mark Allibone, 179 days ago | 3 authors, 3 changes
public static class IoTPlaygroundLogic
{
[FunctionName("ProcessIoTMeasurement")]
0 references | Mark Allibone, Less than 5 minutes ago | 1 author, 1 change
public static async Task<HttpResponseMessage> ProcessIoTMeasurement(
[HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequestMessage req,
[SignalR(HubName = "IoTPlayground")] IAsyncCollector<SignalRMessage> signalRMessages,
ILogger log)
{
var payload = await req.Content.ReadAsStringAsync();
log.LogInformation($"function processed a message: {payload}");
await signalRMessages.AddAsync(
new SignalRMessage
{
Target = "measurement",
Arguments = new[] { payload }
});
return req.CreateResponse(HttpStatusCode.OK);
}
// SignalR Registration
[FunctionName("Negotiate")]
0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
public static SignalRConnectionInfo Negotiate(
[HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
[SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
{
// connectionInfo contains an access key token with a name identifier claim set to the authenticated user
return connectionInfo;
}
}
}
100 %
No issues found
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'Server.IoTMonitor' (2 of 2 projects)
Generator.IoTMonitor
Server.IoTMonitor
Dependencies
Properties
.gitignore
host.json
IoTPlaygroundLogic.cs
local.settings.json
Solution Explorer
Team Explorer
Properties
Output
Show output from: Build
Error List
Output
Ready
Ln 36
Col 1
Ch 1
INS
IoTMonitor.Server
SwitchToStreamAnalytics

@mallibone

```
        var payload = await req.Content.ReadAsStringAsync();
        log.LogInformation($"function processed a message: {payload}");
        await signalRMessages.AddAsync(
            new SignalRMessage
            {
                Target = "measurement",
                Arguments = new[] { payload }
            });

        return req.CreateResponse(HttpStatusCode.OK);
    }

    // SignalR Registration
    [FunctionName("Negotiate")]
    0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
    public static SignalRConnectionInfo Negotiate(
        [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
        [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
    {
        // connectionInfo contains an access key token with a name identifier claim set to the authenticated user
        return connectionInfo;
    }
}
```

```
        var payload = await req.Content.ReadAsStringAsync();
        log.LogInformation($"function processed a message: {payload}");
        await signalRMessages.AddAsync(
            new SignalRMessage
            {
                Target = "measurement",
                Arguments = new[] { payload }
            });

        return req.CreateResponse(HttpStatusCode.OK);
    }

    // SignalR Registration
    [FunctionName("Negotiate")]
    0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
    public static SignalRConnectionInfo Negotiate(
        [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
        [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
    {
        // connectionInfo contains an access key token with a name identifier claim set to the authenticated user
        return connectionInfo;
    }
}
```

```
            var payload = await req.Content.ReadAsStringAsync();
            log.LogInformation($"function processed a message: {payload}");
            await signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "measurement",
                    Arguments = new[] { payload }
                });

            return req.CreateResponse(HttpStatusCode.OK);
        }

        // SignalR Registration
        [FunctionName("Negotiate")]
        0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
        public static SignalRConnectionInfo Negotiate(
            [HttpTrigger(AuthorizationLevel.Anonymous)] HttpRequest req,
            [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
        {
            // connectionInfo contains an access key token with a name identifier claim set to the authenticated user
            return connectionInfo;
        }
    }
}
```

```
            var payload = await req.Content.ReadAsStringAsync();
            log.LogInformation($"function processed a message: {payload}");
            await signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "measurement",
                    Arguments = new[] { payload }
                });

            return req.CreateResponse(HttpStatusCode.OK);
        }

        // SignalR Registration
        [FunctionName("Negotiate")]
        0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
        public static SignalRConnectionInfo Negotiate(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
            [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
        {
            // connectionInfo contains an access key token with a name identifier claim set to the authenticated user
            return connectionInfo;
        }
    }
}
```

```
            var payload = await req.Content.ReadAsStringAsync();
            log.LogInformation($"function processed a message: {payload}");
            await signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "measurement",
                    Arguments = new[] { payload }
                });

            return req.CreateResponse(HttpStatusCode.OK);
        }

        // SignalR Registration
        [FunctionName("Negotiate")]
        0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
        public static SignalRConnectionInfo Negotiate(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
            [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
        {
            // connectionInfo contains an access key token with a name identifier claim set to the authenticated user
            return connectionInfo;
        }
    }
}
```

```
        var payload = await req.Content.ReadAsStringAsync();
        log.LogInformation($"function processed a message: {payload}");
        await signalRMessages.AddAsync(
            new SignalRMessage
            {
                Target = "measurement",
                Arguments = new[] { payload }
            });

        return req.CreateResponse(HttpStatusCode.OK);
    }

    // SignalR Registration
    [FunctionName("Negotiate")]
    0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
    public static SignalRConnectionInfo Negotiate(
        [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
        [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
    {
        // connectionInfo contains an access key token with a name identifier claim set to the authenticated user
        return connectionInfo;
    }
}
```

```
namespace Server.IoTMonitor
{
    0 references | Mark Allibone, 179 days ago | 3 authors, 3 changes
    public static class IoTPlaygroundLogic
    {
        [FunctionName("ProcessIoTMeasurement")]
        0 references | Mark Allibone, Less than 5 minutes ago | 1 author, 1 change
        public static async Task<HttpResponseMessage> ProcessIoTMeasurement(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequestMessage req,
            [SignalR(HubName = "IoTPlayground")] IAsyncCollector<SignalRMessage> signalRMessages,
            ILogger log)
        {
            var payload = await req.Content.ReadAsStringAsync();
            log.LogInformation($"function processed a message: {payload}");
            await signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "measurement",
                    Arguments = new[] { payload }
                });

            return req.CreateResponse(HttpStatusCode.OK);
        }

        // SignalR Registration
        [FunctionName("Negotiate")]
        0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
        public static SignalRConnectionInfo Negotiate(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
            [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
        {
```

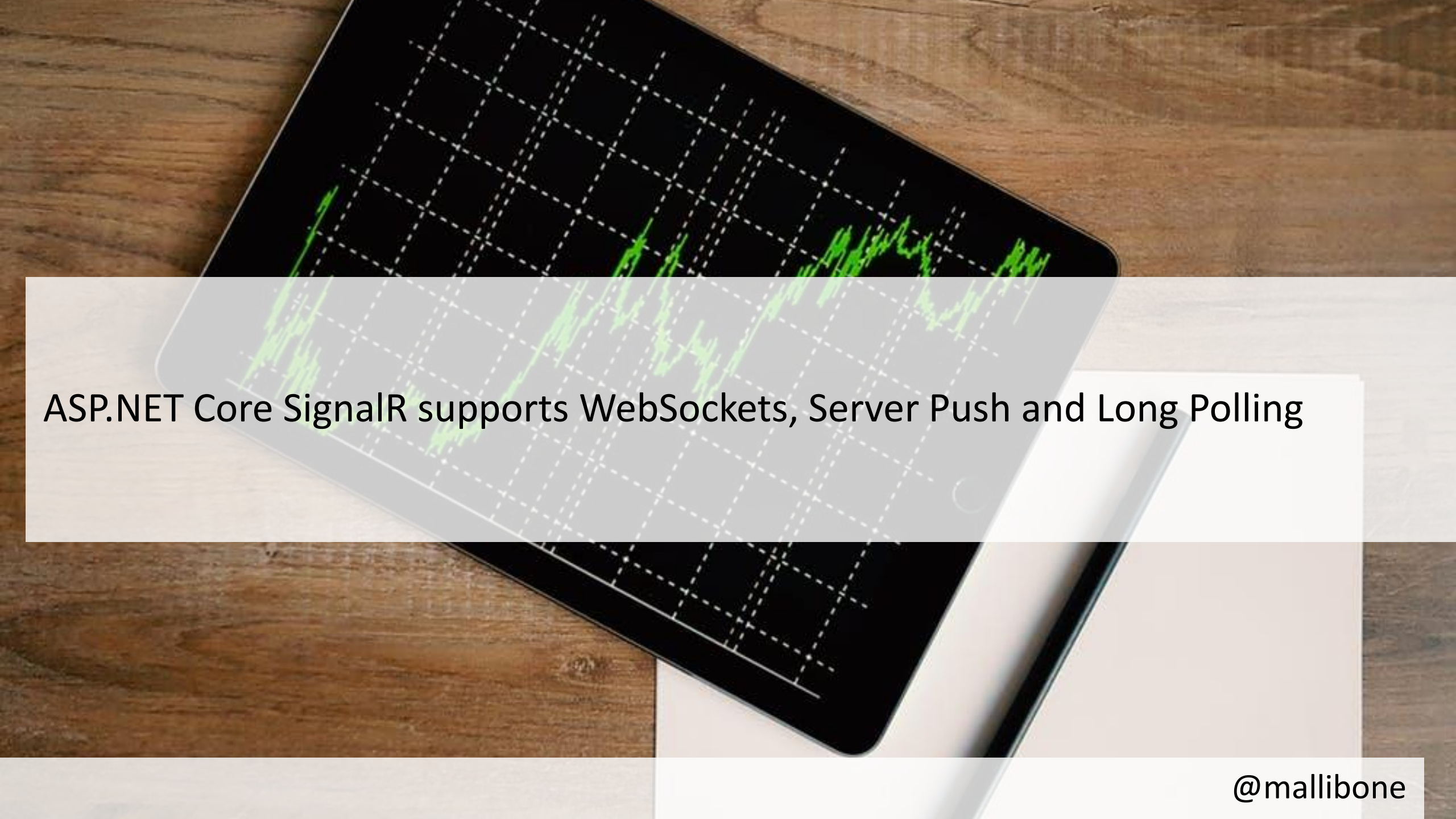
```
namespace Server.IoTMonitor
{
    0 references | Mark Allibone, 179 days ago | 3 authors, 3 changes
    public static class IoTPlaygroundLogic
    {
        [FunctionName("ProcessIoTMeasurement")]
        0 references | Mark Allibone, Less than 5 minutes ago | 1 author, 1 change
        public static async Task<HttpResponseMessage> ProcessIoTMeasurement(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequestMessage req,
            [SignalR(HubName = "IoTPlayground")] IAsyncCollector<SignalRMessage> signalRMessages,
            ILogger log)
        {
            var payload = await req.Content.ReadAsStringAsync();
            log.LogInformation($"Function processed a message: {payload}");
            await signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "measurement",
                    Arguments = new[] { payload }
                });

            return req.CreateResponse(HttpStatusCode.OK);
        }

        // SignalR Registration
        [FunctionName("Negotiate")]
        0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
        public static SignalRConnectionInfo Negotiate(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
            [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
        {
```

```
namespace Server.IoTMonitor
{
    0 references | Mark Allibone, 179 days ago | 3 authors, 3 changes
    public static class IoTPlaygroundLogic
    {
        [FunctionName("ProcessIoTMeasurement")]
        0 references | Mark Allibone, Less than 5 minutes ago | 1 author, 1 change
        public static async Task<HttpResponseMessage> ProcessIoTMeasurement(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequestMessage req,
            [SignalR(HubName = "IoTPlayground")] IAsyncCollector<SignalRMessage> signalRMessages,
            ILogger log)
        {
            var payload = await req.Content.ReadAsStringAsync();
            log.LogInformation($"function processed a message: {payload}");

            await signalRMessages.AddAsync(
                new SignalRMessage
                {
                    Target = "measurement",
                    Arguments = new[] { payload }
                });

            return req.CreateResponse(HttpStatusCode.OK);
        }

        // SignalR Registration
        [FunctionName("Negotiate")]
        0 references | Mark Allibone, 187 days ago | 2 authors, 2 changes
        public static SignalRConnectionInfo Negotiate(
            [HttpTrigger(AuthorizationLevel.Anonymous)]HttpRequest req,
            [SignalRConnectionInfo(HubName = "IoTPlayground")] SignalRConnectionInfo connectionInfo)
        {
```


FOREX
Instant Execution
-0.01
-0.1
0.27
+0.01
+0.1
Stop Loss
Take Profit
Deviation
1.27165
1.27165
Sell

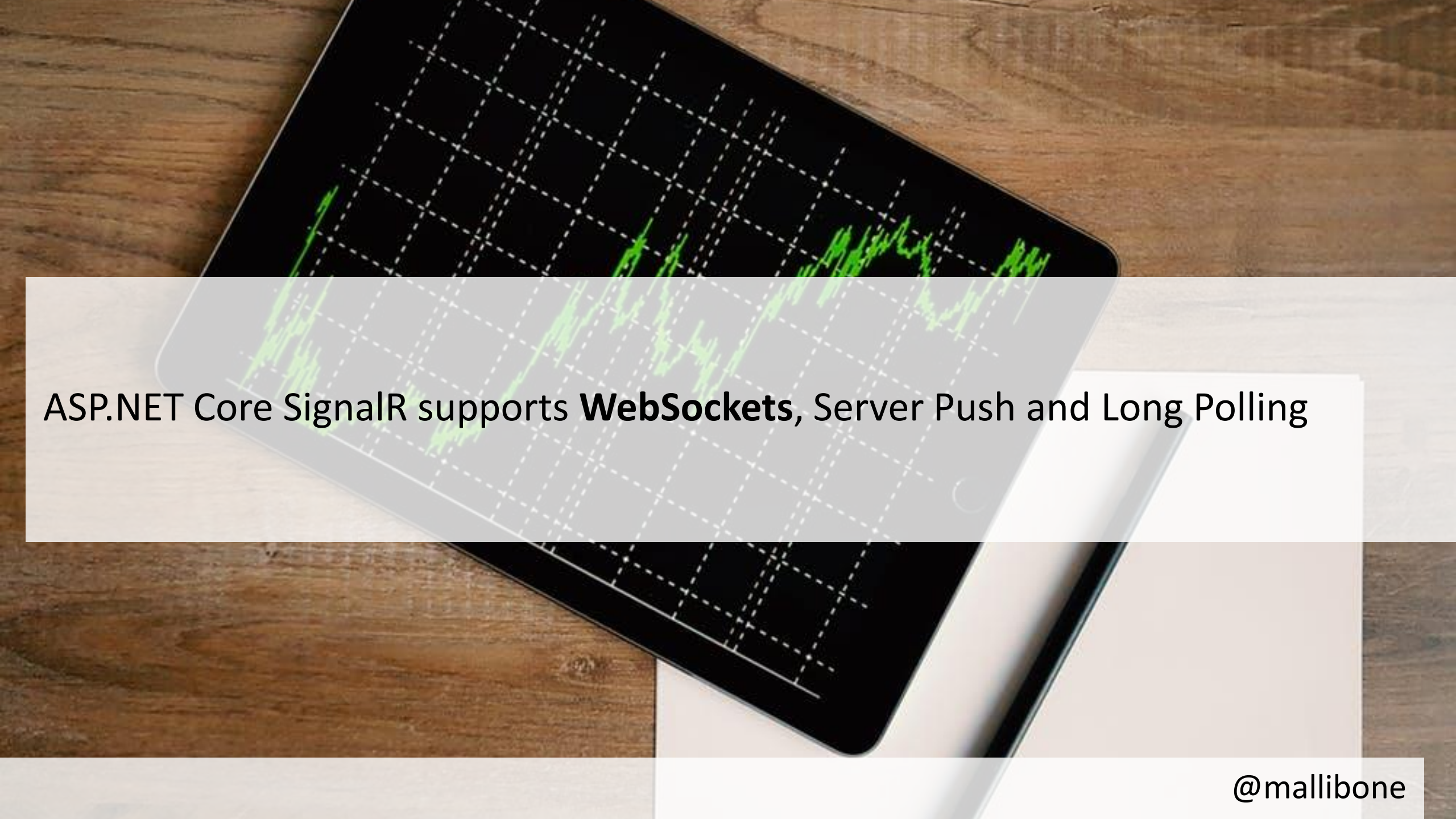
SignalR Service is ASP.Net SignalR Core “carefree”

@mallibone



ASP.NET Core SignalR supports WebSockets, Server Push and Long Polling

@mallibone



ASP.NET Core SignalR supports **WebSockets**, Server Push and Long Polling

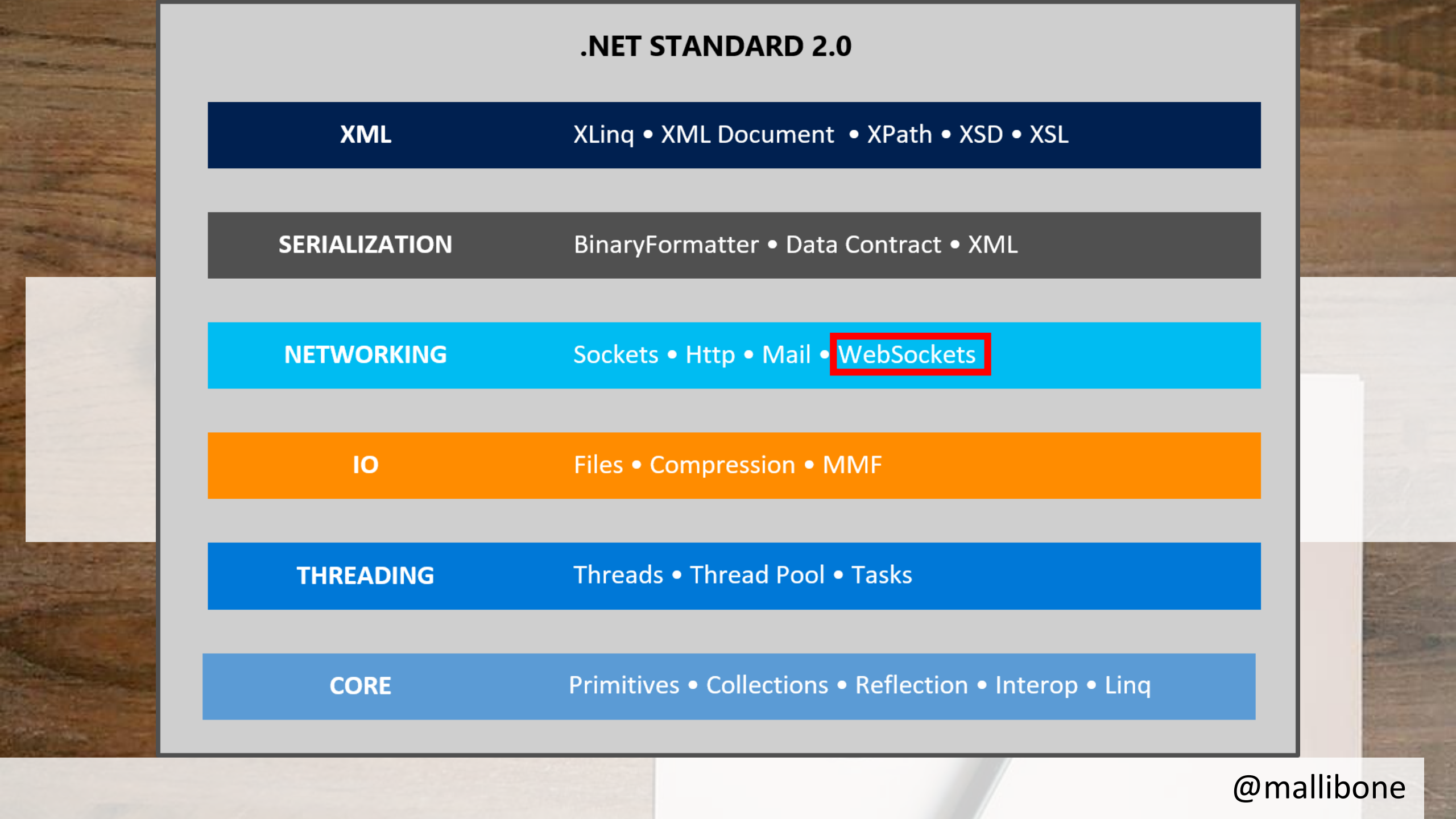
@mallibone

```
10      {
11          const string backendUrl = "https://iotdemofunction.azurewebsites.net/api/";
12          //const string backendUrl = "http://localhost:7071/api/";
13
14          // Connect to SignalR Service
15          var connect = new HubConnectionBuilder()
16              .WithUrl(backendUrl)
17              .Build();
18
19          // Attach Message handler
20          connect.On<string>("measurement", (messageString) =>
21          {
22              var message = JsonConvert.DeserializeObject<IEnumerable<Measurement>>(messageString);
23              System.Console.WriteLine("Message Received: " + messageString);
24          });
25
26          // Start Monitoring
27          await connect.StartAsync();
28
29          // User input handling
30
31          System.Console.WriteLine("...");
32          System.Console.ReadLine();
33          await connect.DisposeAsync();
34      }
35  }
```

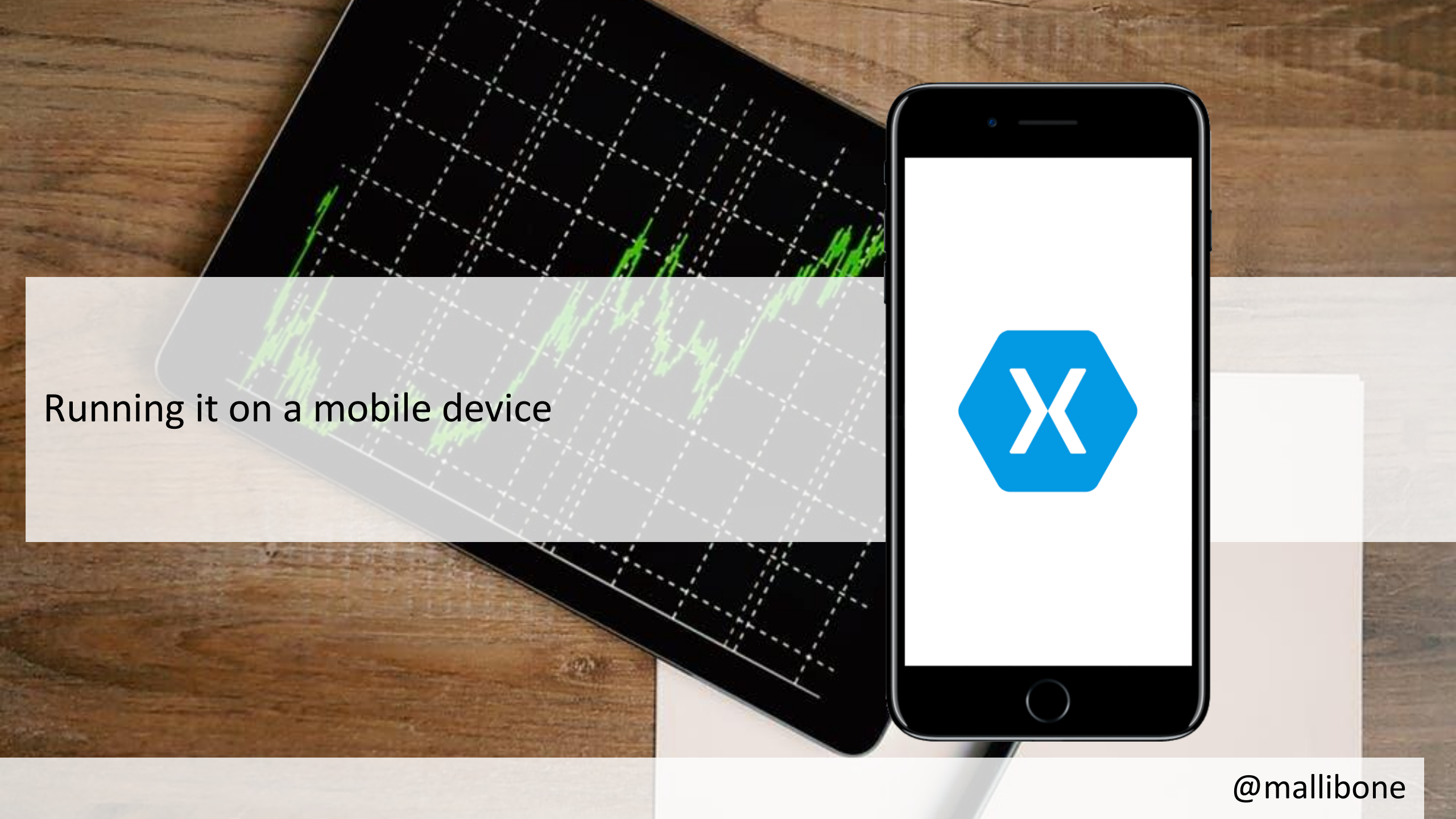
```
10      {
11          const string backendUrl = "https://iotdemofunction.azurewebsites.net/api/";
12          //const string backendUrl = "http://localhost:7071/api/";
13
14          // Connect to SignalR Service
15          var connect = new HubConnectionBuilder()
16              .WithUrl(backendUrl)
17              .Build();
18
19          // Attach Message handler
20          connect.On<string>("measurement", (messageString) =>
21          {
22              var message = JsonConvert.DeserializeObject<IEnumerable<Measurement>>(messageString);
23              System.Console.WriteLine("Message Received: " + messageString);
24          });
25
26          // Start Monitoring
27          await connect.StartAsync();
28
29          // User input handling
30
31          System.Console.WriteLine("...");
32          System.Console.ReadLine();
33          await connect.DisposeAsync();
34      }
35  }
```

```
10      {
11          const string backendUrl = "https://iotdemofunction.azurewebsites.net/api/";
12          //const string backendUrl = "http://localhost:7071/api/";
13
14          // Connect to SignalR Service
15          var connect = new HubConnectionBuilder()
16              .WithUrl(backendUrl)
17              .Build();
18
19          // Attach Message handler
20          connect.On<string>("measurement", (messageString) =>
21          {
22              var message = JsonConvert.DeserializeObject<IEnumerable<Measurement>>(messageString);
23              System.Console.WriteLine("Message Received: " + messageString);
24          });
25
26          // Start Monitoring
27          await connect.StartAsync();
28
29          // User input handling
30
31          System.Console.WriteLine("...");
32          System.Console.ReadLine();
33          await connect.DisposeAsync();
34      }
35  }
```

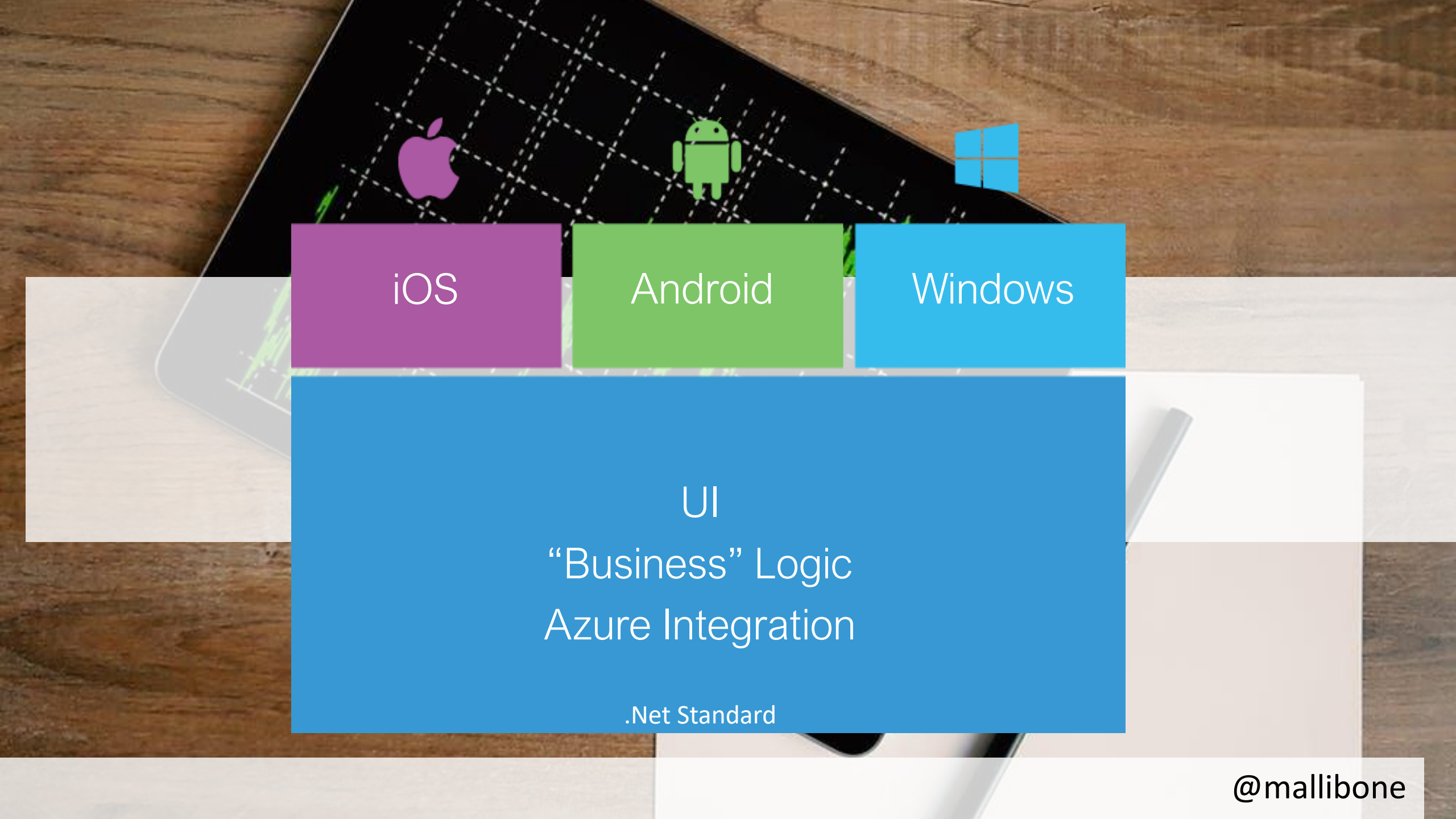
```
10      {
11          const string backendUrl = "https://iotdemofunction.azurewebsites.net/api/";
12          //const string backendUrl = "http://localhost:7071/api/";
13
14          // Connect to SignalR Service
15          var connect = new HubConnectionBuilder()
16              .WithUrl(backendUrl)
17              .Build();
18
19          // Attach Message handler
20          connect.On<string>("measurement", (messageString) =>
21          {
22              var message = JsonConvert.DeserializeObject<IEnumerable<Measurement>>(messageString);
23              System.Console.WriteLine("Message Received: " + messageString);
24          });
25
26          // Start Monitoring
27          await connect.StartAsync();
28
29          // User input handling
30
31          System.Console.WriteLine("...");
32          System.Console.ReadLine();
33          await connect.DisposeAsync();
34      }
35  }
```

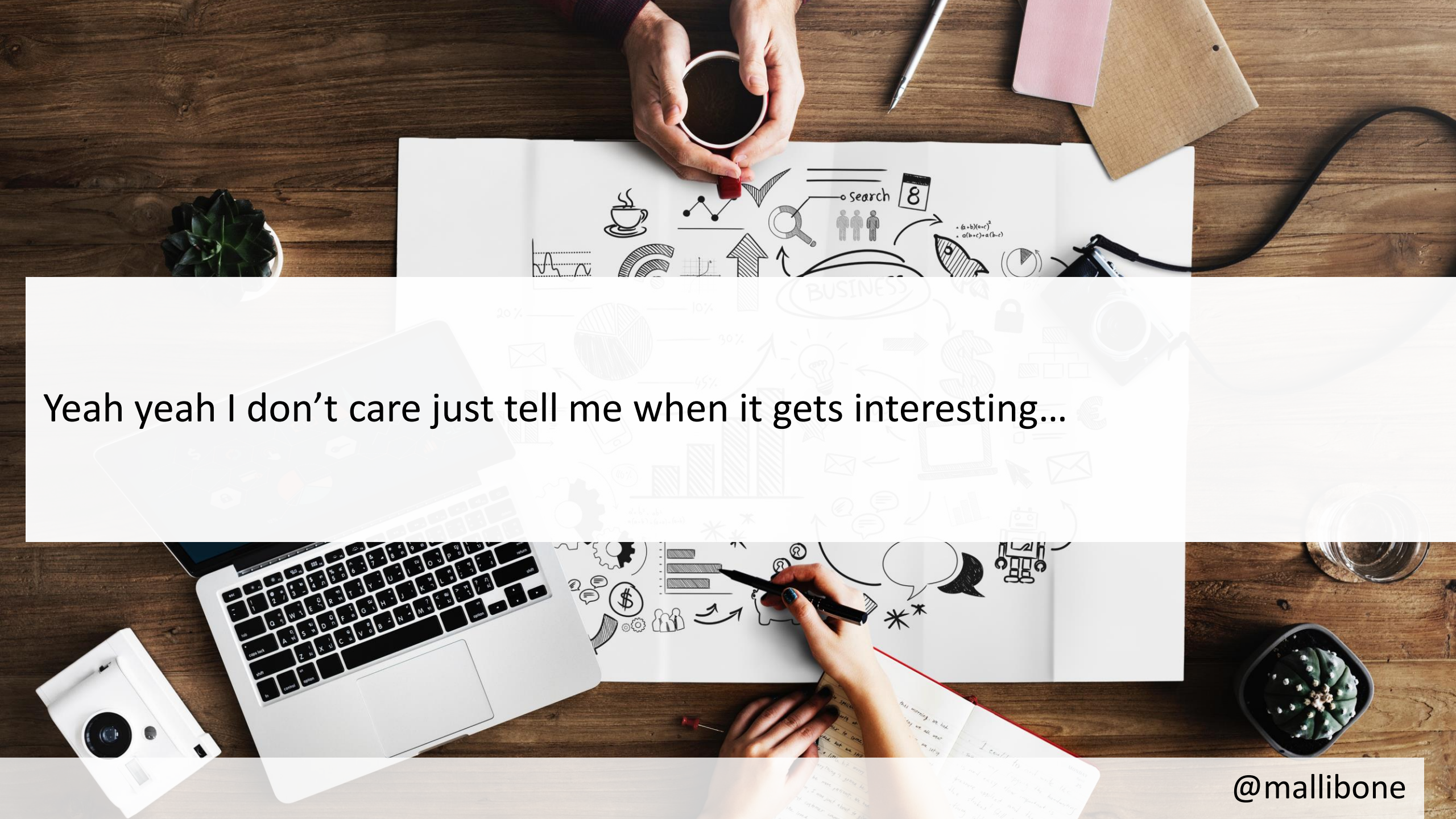

.NET STANDARD 2.0
XML
XLinq • XML Document • XPath • XSD • XSL
SERIALIZATION
BinaryFormatter • Data Contract • XML
NETWORKING
Sockets • Http • Mail • WebSockets
IO
Files • Compression • MMF
THREADING
Threads • Thread Pool • Tasks
CORE
Primitives • Collections • Reflection • Interop • Linq
@mallibone


Cross Platform ♡ .Net Standard
@mallibone



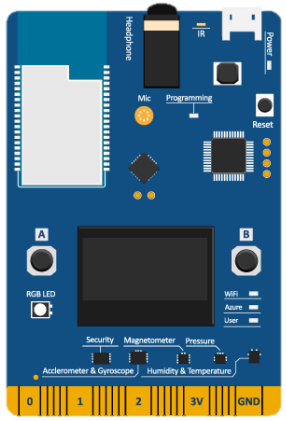
Running it on a mobile device

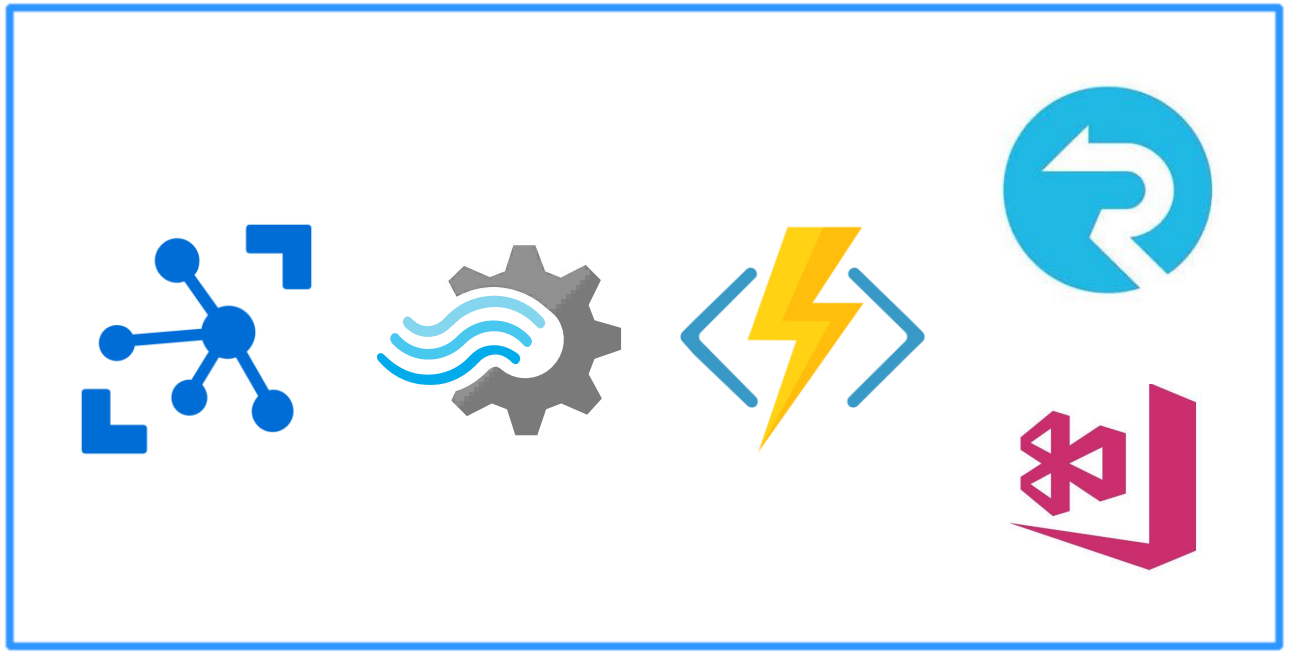

iOS
Android
Windows
UI
“Business” Logic
Azure Integration
.Net Standard



Yeah yeah I don’t care just tell me when it gets interesting…

@mallibone



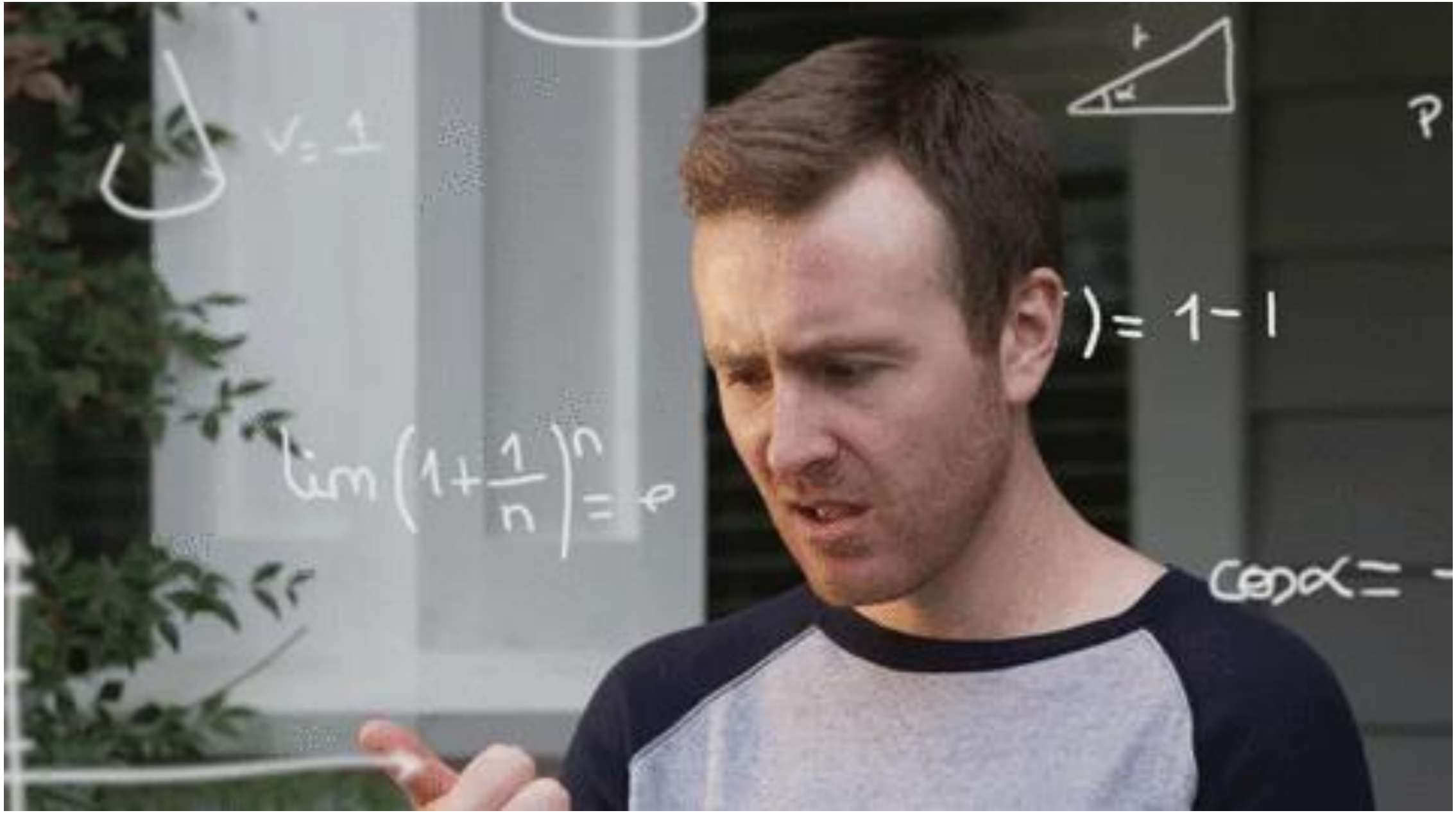


Azure Cloud

@mallibone

But how do we detect something interesting?

Using machine learning



@mallibone

```
WITH AnomalyDetectionStep AS
(
    SELECT
        EVENTENQUEUEDUTCTIME AS time,
        CAST(temperature AS float) AS temp,
        AnomalyDetection_SpikeAndDip(CAST(temperature AS float), 95, 120, 'spikesanddips')
            OVER(LIMIT DURATION(second, 120)) AS SpikeAndDipScores
    FROM IoTHub
)
SELECT
    time,
    temp,
    CAST(GetRecordPropertyValue(SpikeAndDipScores, 'Score') AS float) AS
    SpikeAndDipScore,
    CAST(GetRecordPropertyValue(SpikeAndDipScores, 'IsAnomaly') AS bigint) AS
    IsSpikeAndDipAnomaly
INTO AnomalyOutut
FROM AnomalyDetectionStep

SELECT
    *
INTO
    DataStream
FROM
    IoTHub

```

Input preview | Test results

Showing 100 rows from 'anomalyoutut'. Download results

time	temp	spikeanddipscore	isspikeanddipanomaly
"2019-10-25T05:33:44.7720000Z"	20.1	0.39797473087212243	0
"2019-10-25T05:33:43.6770000Z"	20.1	0.39918301431657754	0
"2019-10-25T05:33:42.4270000Z"	20.8	0.46537282782537737	0
"2019-10-25T05:33:41.3790000Z"	20.8	0.46537282782537737	0
"2019-10-25T05:33:40.3310000Z"	20.2	0.41932561007473657	0
"2019-10-25T05:33:39.2690000Z"	20.8	0.46440524496941693	0
"2019-10-25T05:33:38.2060000Z"	20.4	0.4586440721747156	0
"2019-10-25T05:33:37.1580000Z"	31.4	0.015477066118496863	1
"2019-10-25T05:33:35.9390000Z"	20.1	0.4068078323426809	0
"2019-10-25T05:33:34.3920000Z"	20.2	0.4248894379197309	0

```
WITH AnomalyDetectionStep AS
(
    SELECT
        EVENTENQUEUEDUTCTIME AS time,
        CAST(temperature AS float) AS temp,
        AnomalyDetection_SpikeAndDip(CAST(temperature AS float), 95, 120, 'spikesanddips')
            OVER(LIMIT DURATION(second, 120)) AS SpikeAndDipScores
    FROM IoTHub
)
SELECT
    time,
    temp,
    CAST(GetRecordPropertyValue(SpikeAndDipScores, 'Score') AS float) AS
    SpikeAndDipScore,
    CAST(GetRecordPropertyValue(SpikeAndDipScores, 'IsAnomaly') AS bigint) AS
    IsSpikeAndDipAnomaly
INTO AnomalyOutut
FROM AnomalyDetectionStep

SELECT
    *
INTO
    DataStream
FROM
    IoTHub
```

@mallibone

```
WITH AnomalyDetectionStep AS
(
    SELECT
        EVENTENQUEUEDUTCTIME AS time,
        CAST(temperature AS float) AS temp,
        AnomalyDetection_SpikeAndDip(CAST(temperature AS float), 95, 120, 'spikesanddips')
            OVER(LIMIT DURATION(second, 120)) AS SpikeAndDipScores
    FROM IoTHub
)
SELECT
    time,
    temp,
    CAST(GetRecordPropertyValue(SpikeAndDipScores, 'Score') AS float) AS
    SpikeAndDipScore,
    CAST(GetRecordPropertyValue(SpikeAndDipScores, 'IsAnomaly') AS bigint) AS
    IsSpikeAndDipAnomaly
INTO AnomalyOutut
FROM AnomalyDetectionStep

SELECT
    *
INTO
    DataStream
FROM
    IoTHub

```

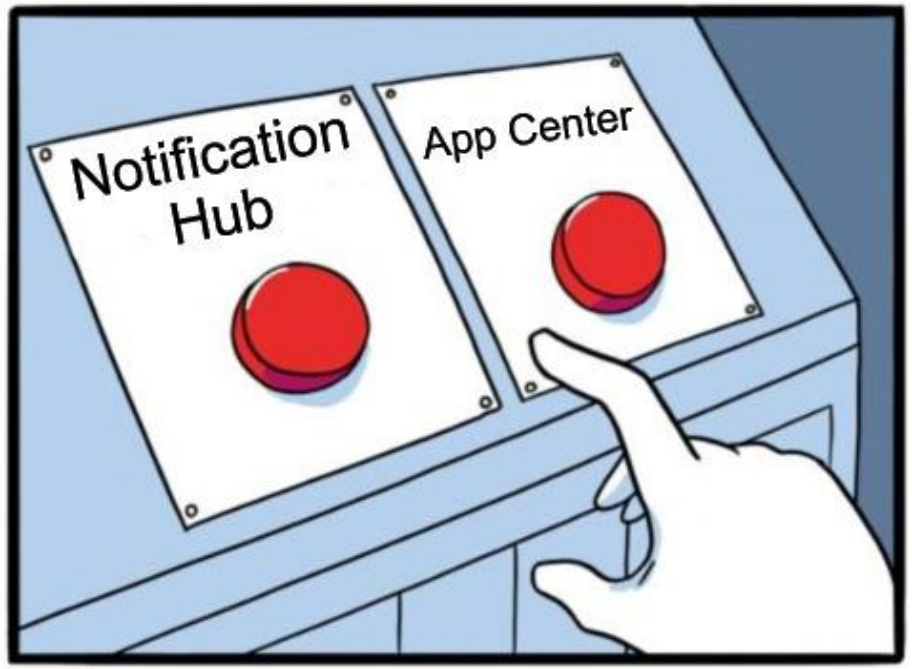
```
24  FROM
25      IoTHub
26
```

Input preview | Test results

Showing 100 rows from 'anomalyoutut'.

time	temp	spikeanddipscore	isspikeanddipanomaly
"2019-10-25T05:33:44.7720000Z"	20.1	0.39797473087212243	0
"2019-10-25T05:33:43.6770000Z"	20.1	0.39918301431657754	0
"2019-10-25T05:33:42.4270000Z"	20.8	0.46537282782537737	0
"2019-10-25T05:33:41.3790000Z"	20.8	0.46537282782537737	0
"2019-10-25T05:33:40.3310000Z"	20.2	0.41932561007473657	0
"2019-10-25T05:33:39.2690000Z"	20.8	0.46440524496941693	0
"2019-10-25T05:33:38.2060000Z"	20.4	0.4586440721747156	0
"2019-10-25T05:33:37.1580000Z"	31.4	0.015477066118496863	1
"2019-10-25T05:33:35.9390000Z"	20.1	0.4068078323426809	0
"2019-10-25T05:33:34.3920000Z"	20.2	0.4248894379197309	0

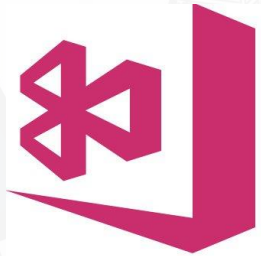
@mallibone



imgflip.com

@mallibone

Microsoft App Center



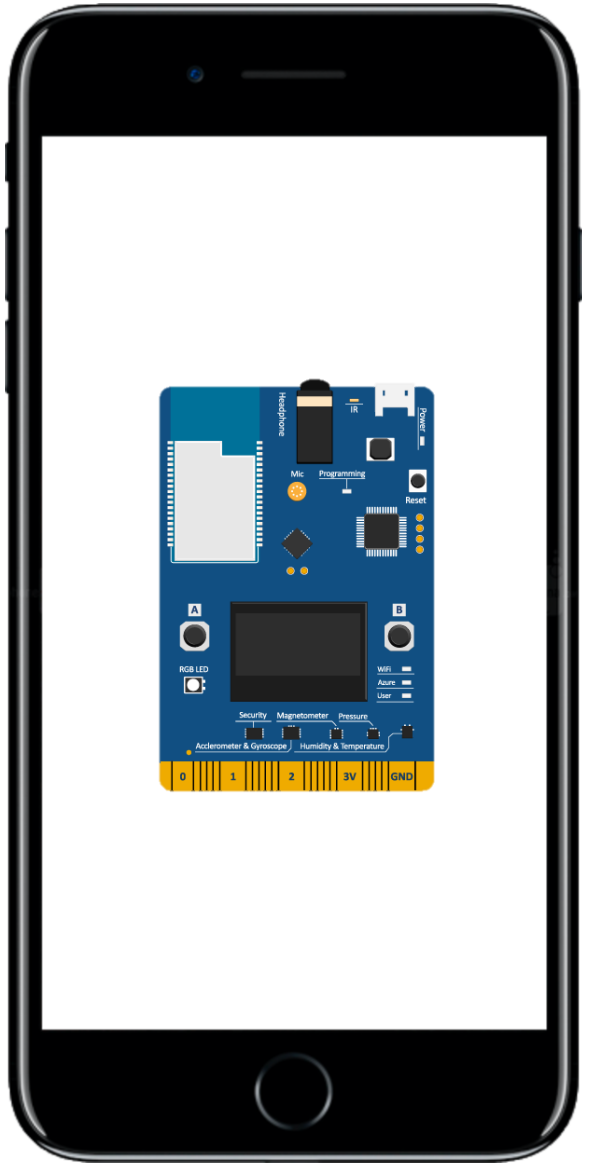
@mallibone

Demo starting at 11:10 😉

@mallibone

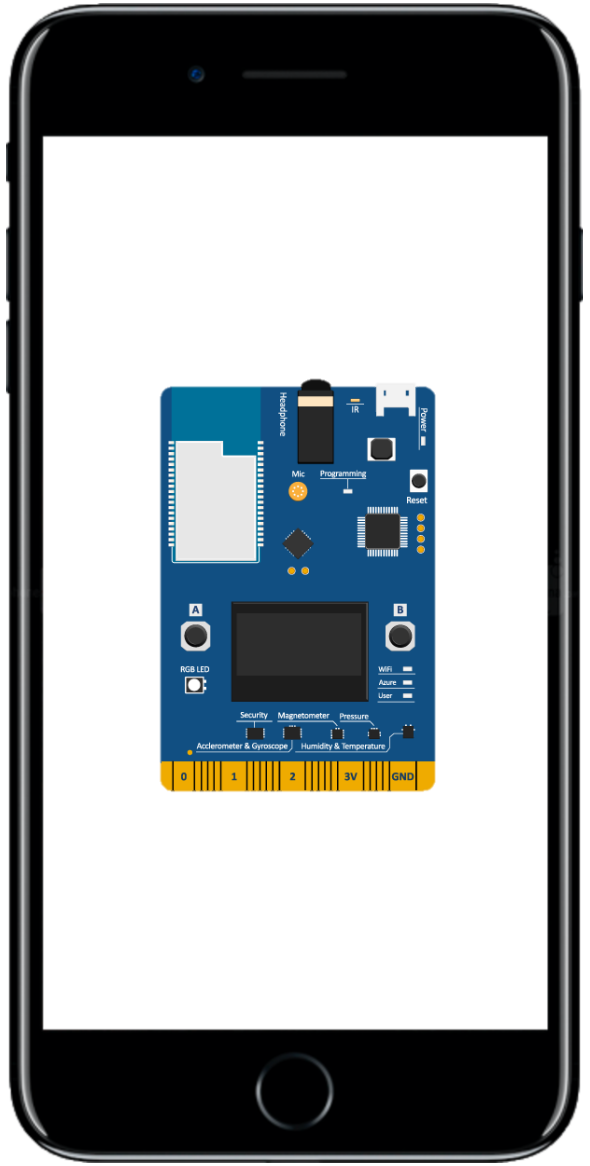
REY AUTOMATION

Takeaways



Takeaways

- Mobile is a great IoT monitoring/visualization client
- Live streaming to clients made easy with SignalR
- Implement IoT solutions that can scale with Azure and it’s services like IoT Hub, Stream Analytics, App Center, …



REY AUTOMATION

Thank you for your time!

Mark Allibone

Mobile Lead, Rey Automation AG

https://mallibone.com

https://bit.ly/2p2GZnm

@mallibone

FEATURED COMMUNITY BLOG

Microsoft® Most Valuable Professional

Microsoft CERTIFIED Solution Developer

Xamarin Certified Mobile Developer