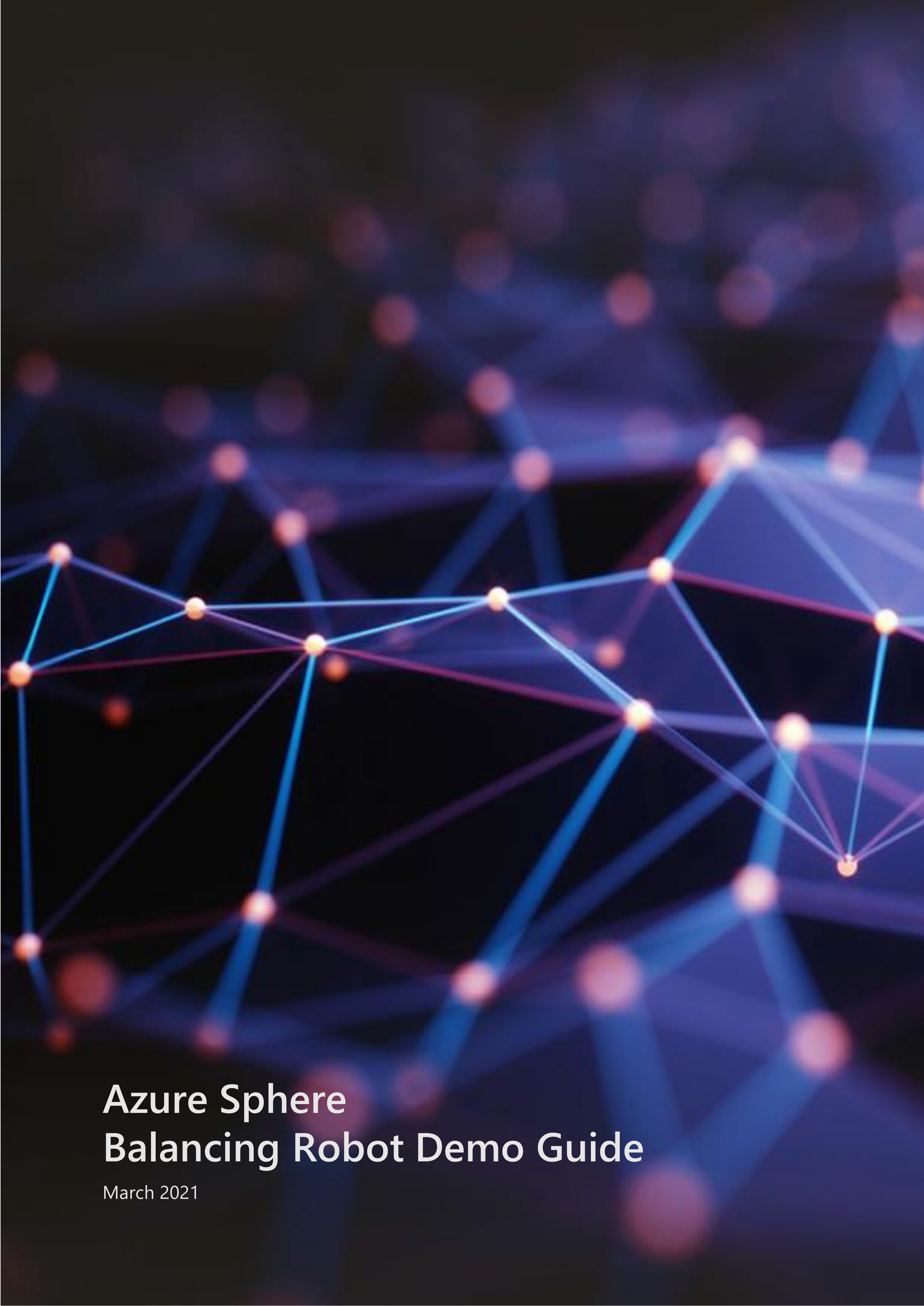


Azure Sphere Balancing Robot Demo Guide

March 2021

Contents

Welcome to the Balancing Robot Demo Guide

Purpose & Scope

This document describes how to use the Balancing Robot proof-of-concept device to demonstrate features of Azure Sphere including:

- *How to set up the demo*
- *How to demo deferred over-the-air update*
- *How to demo telemetry and control through Azure IoT Central*

Introduction and Setup

To use the robot you need the [Azure Sphere SDK]() (select the appropriate link at the bottom of the page for Windows or Linux – note that you only need the SDK, you won't need the Visual Studio or Visual Studio Code extensions).

You will also need to have a login account for Azure IoT Central, and an Azure Sphere tenant.

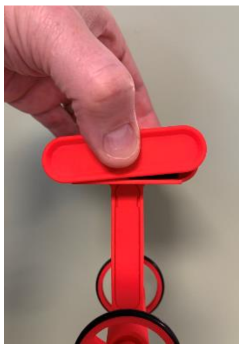
Instructions for setting up the Azure IoT Central application and Azure Sphere tenant can be found in the accompanying document "Balancing_Robot_Demo_Setup_Guide".

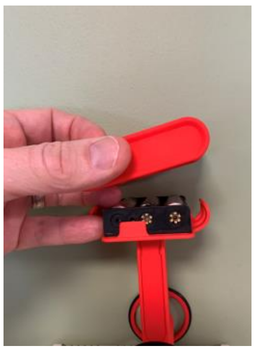
Note: You should complete the Demo Setup Guide instructions before continuing with the demo guide.

Robot Setup

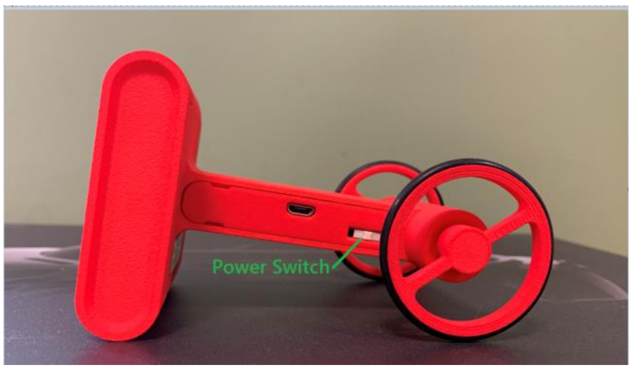
The robot needs 3x AA batteries to power the robot (the USB connector does not power the robot).

To open the top of the robot and access the battery compartment gently pull up on one side then pull straight up. It will take a bit of force to pull the top off, but don't worry it will be fine.





IMPORTANT: Make sure you lay it flat with the display facing up when you turn on the power switch. It needs to be in this position to calibrate the accelerometer, gyro and magnetometer. Once the hourglass icon has disappeared you can stand the robot up.



Robot features

- Uses Azure Sphere + Azure RTOS enables developers to build secure Real-Time applications
- Uses Azure Sphere deferred update/update awareness features
- Shows Azure IoT Central support for Telemetry and Remote Control

UI icons explained

🔋	Battery indication, 50%
📶	Wi-Fi connected
⬢	IoTC connected
🕓	Deferred update
🔄	Update pending
A	Running app A
B	Running app B

Robot Hardware/Software

The robot is based on the following hardware components: Avnet MT3620 Module, Invensense ICM20948 IMU (Accelerometer, Gyroscope, Magnetometer), TB6612FNG Motor controller, SSD1306 OLED Display (32x128 pixels), and VL53L1X Time of Flight (distance) sensors.

The robot uses the MT3620 ARM Cortex-A core (High Level) application to connect to the internet, send telemetry to Azure IoT Central, and to send/receive inter-core messages to one of the MT3620 ARM Cortex-M cores (Real-Time capable application).

One of the MT3620 Cortex-M cores is running Azure RTOS, the real-time application has three threads:

- The main thread is responsible for reading sensors, doing 'sensor fusion' to compute Yaw, Pitch, Roll, running two PID control algorithms (one to evaluate the current 'roll angle' and adjust the motors to achieve balance, and the second to evaluate current speed and adjust the motors to bring the robot to a halt).
- The Time of Flight sensor code is running on the second (lower priority) thread
- Inter-core communications are handled by the third thread (also a low priority thread)

Azure Sphere/Azure RTOS real-time system.

Wikipedia: A system is said to be real-time if the total correctness of an operation depends not only upon its logical correctness, but also upon the time in which it is performed.

The Azure Sphere OS which runs on the ARM Cortex-A core is based on the Linux Kernel – Linux is a general purpose OS and does not support real-time applications. The MT3620 MCU supports an ARM Cortex-A processor, and two real-time capable ARM Cortex-M processors – The robot runs Azure RTOS on one of the ARM Cortex-M processors, this enables building a real-time system.

A two wheeled robot is inherently unstable and would fall over if the underlying hardware/software didn't keep the robot balanced. In simple terms if the robot starts to fall in one direction the robot needs to compensate by applying an appropriate amount of power to the motors to move the robot in the opposite direction (similar to balancing a stick on the palm of your hand).

Usage

To use the robot you simply need to turn it on, lay the robot on its back, with the display facing up, wait for the hourglass icon to disappear, and then stand the robot upright. The robot will balance and make small adjustments to its position to maintain the upright position.

You should be able to gently push the robot from the front/back and notice that the robot will compensate and continue to maintain balance.

Deferred Update/update awareness

Applications that run on an embedded system may require updates, this might be due to a bug in the code, or an update to functionality (perhaps determined by analysis of device telemetry data).

Azure Sphere supports updating the OS, and High Level and Real-Time capable applications. An overview of application deployment can be found here.

In Azure Sphere, high Level applications can register for update notifications and can either accept the update, or postpone (defer) the update. The robot's High Level application has registered for update notifications and will allow these updates to be applied if the robot is in a 'safe' position (laying down) – As an embedded systems developer you should consider what a 'safe' condition is for your device – for example, a washing machine may be in the middle of a spin-cycle when an update notification is received by the High Level application, the application may want to defer the update until after the wash cycle is completed, or to a point in time where the washing machine isn't typically used.

The robot includes a 32x128 pixel OLED display which provides information about the system state of the robot, this includes: Battery level, Wi-Fi connectivity, Azure IoT Central connectivity, and an icon that shows whether 'Application A' or 'Application B' are executing, or an icon that shows update status.

Deferred update

To use deferred update you will need the following:

- Azure Sphere SDK for Windows/Linux (see prerequisites)
- The device ID of the robot
- Determine whether the robot is currently running 'Application A' or 'Application B'

Understanding Deferred Update

A high-level application can temporarily defer updates to the Azure Sphere OS and to application images to prevent the update from interrupting critical processing. An Azure Sphere device in a kitchen appliance, for example, could defer updates during use.

More information on Deferred Updates can be found here.

There is also a Channel 9 IoT Show that discusses and demonstrates (code and demo) deferred update – here.

Obtaining the Azure Sphere Device ID

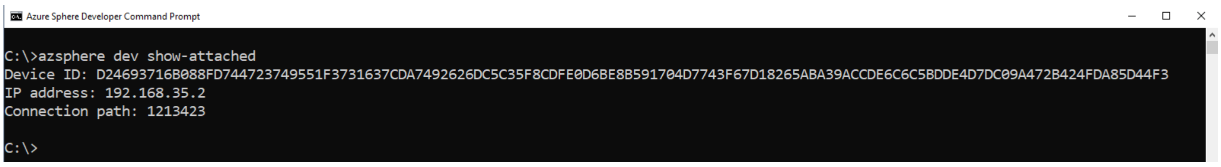
Open a command prompt.

Connect your robot to your PC using the provided USB cable, ensure that the robot has 3x AA batteries installed, and is turned on.

At the Command Prompt type:

```
azsphere device show-attached
```

This should display output like the following:



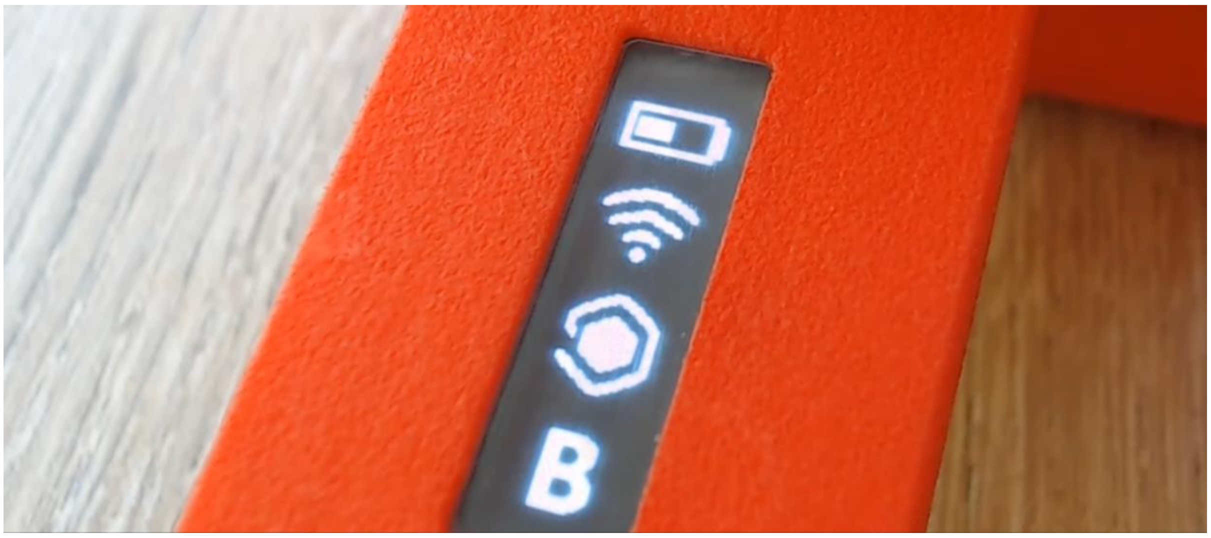
```
C:\>azsphere dev show-attached
Device ID: D24693716B088FD744723749551F3731637CDA7492626DC5C35F8CDFE0D6BE8B591704D7743F67D18265ABA39ACCDE6C6C5BDDE4D7DC09A472B424FDA85D44F3
IP address: 192.168.35.2
Connection path: 1213423

C:\>
```

Copy the device id to the clipboard.

Determine whether the robot is running ‘Application A’ or ‘Application B’

To determine whether the robot is running ‘Application A’ or ‘Application B’ ensure that the robot has 3x AA batteries installed, and turn on the robot – the robot will initially display an hourglass icon – once this has cleared you should see four icons (Battery level, Wi-Fi state, IoT Central state, and ‘A’ or ‘B’ – this indicates which application you are running.



Triggering an application update

To trigger an update you will need to login to an Azure Sphere Tenant.

Type:

```
azsphere login
```

Robot Running 'Application A'

If your robot is currently running 'Application A' then issue the following command:

Type:

```
azsphere device update --device-group "Robot/AppB" --device <your device ID>
```

Robot Running 'Application B'

If your robot is currently running 'Application B' then issue the following command:

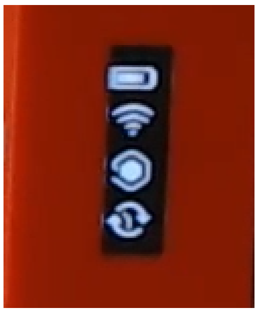
Type:

```
azsphere device update --device-group "Robot/AppA" --device <your device ID>
```

Note: Without interacting with the robot, the update could take up to 24 hours to be applied. The quickest way to trigger the update is to turn the robot off – lay the robot down – and turn the robot back on again.

Once the hourglass has been replaced by the power, Wi-Fi, Azure IoT Central, and Application icon you should then *immediately* stand the robot up, doing this ASAP so it is standing when the OS realizes there's an App update.

Once an update notification is received the robot will determine whether it's safe to apply the update – if the robot is laying down then it's safe to apply the update, but if the robot is standing up then it's not safe to apply the update (the robot would fall over if the Real-Time capable application is updated while trying to balance the robot). In this case the robot display will show a 'deferred update' icon (the lowest icon in the image below), the robot will 'defer' the update for one minute – the update will continue to be deferred until the robot is laying down, and therefore in a safe position to accept the update.

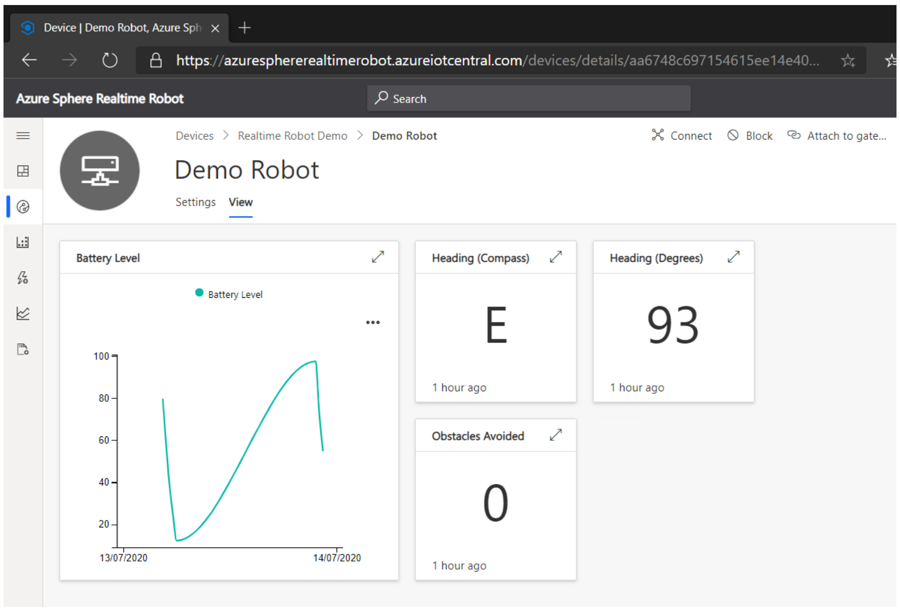


Azure IoT Central Telemetry and Remote Control (Heading)

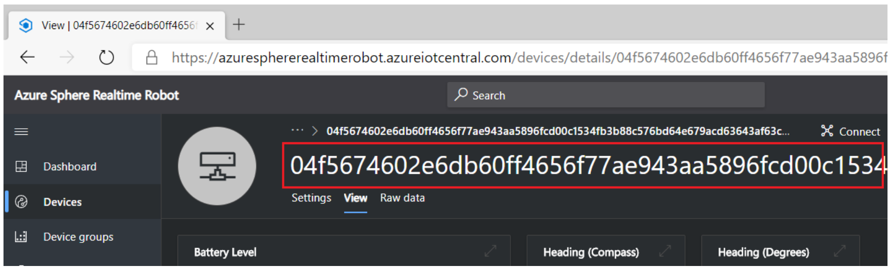
The robot demo uses [Azure IoT Central]() to display device telemetry and for remote control.

Open your Azure IoT Central application in your browser.

Azure IoT Central has a navigation menu on the left side of the page - Select the third icon down (devices), and then select your device from the list of Robots – you will see a telemetry page similar to the image below.



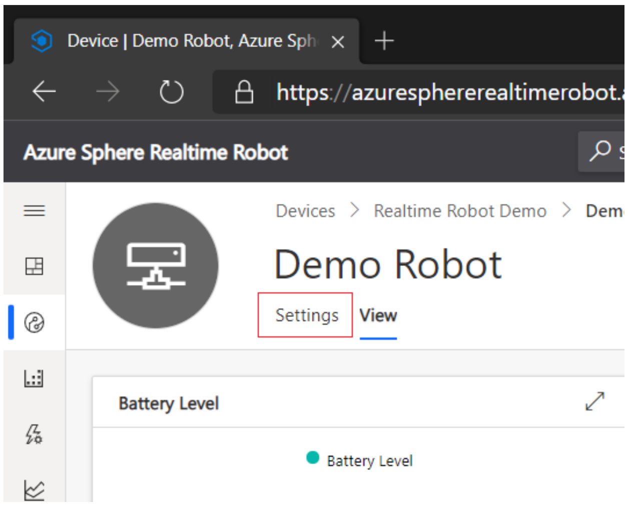
Note that the IoT Central telemetry page may show the current device name as the Device ID that you captured earlier (like the image below).


View | 04f5674602e6db60ff4656
https://azuresphererealtimerobot.azureiotcentral.com/devices/details/04f5674602e6db60ff4656f77ae943aa5896f
Azure Sphere Realtime Robot
Search
Dashboard
Devices
Device groups
04f5674602e6db60ff4656f77ae943aa5896fcd00c1534fb3b88c576bd64e679acd63643af63c...
Connect
04f5674602e6db60ff4656f77ae943aa5896fcd00c1534
Settings
View
Raw data
Battery Level
Heading (Compass)
Heading (Degrees)

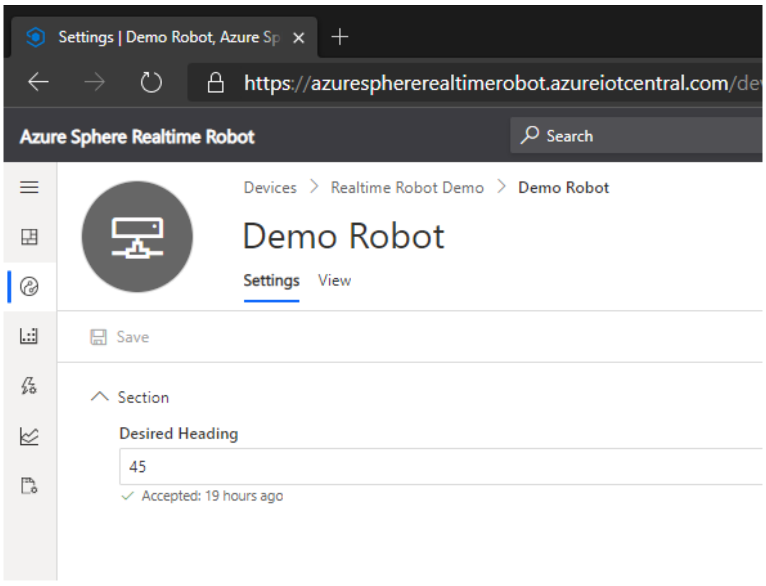
You can change the display name by clicking on the Device ID and giving the robot a more appropriate demo name, for example 'Robot1' – be aware that the device name you choose will be visible to other uses of the Azure IoT Central application.

The telemetry page displays information about current battery level, the current compass direction both in text and degrees, and also the number of obstacles avoided (number of obstacles detected by the Time of Flight sensors).

Azure IoT Central can also be used to request changes to a devices behaviour, select 'Settings' (highlighted in the image below).


Device | Demo Robot, Azure Sph
https://azurespherereltimerobot.
Azure Sphere Realtime Robot
Devices > Realtime Robot Demo > Dem
Demo Robot
Settings
View
Battery Level
Battery Level

The Settings page enables you to select a new heading for the robot – simply enter a new heading (0-360) in the Desired Heading Text Box, and then click "Save", after a few seconds you will see your robot rotate to point in the desired direction.


Settings | Demo Robot, Azure Sp
https://azurespherereltimerobot.azureiotcentral.com/de
Azure Sphere Realtime Robot
Search
Devices > Realtime Robot Demo > Demo Robot
Demo Robot
Settings
View
Save
Section

Desired Heading
45
✓ Accepted: 19 hours ago

Connecting the Robot to Wi-Fi

Note: Wi-Fi isn't needed to show real-time capabilities, but is needed for deferred updates, Azure IoT Central telemetry and remote control.

Connect your robot to your PC using the provided USB cable, ensure that the robot has 3x AA batteries installed, and is turned on.

Follow the instructions here to get connected to Wi-Fi.

Safety

The robot incorporates two laser-based distance sensors – for the demo devices which the Azure Sphere team built, we undertook certification to ensure compliance with BS EN 60825-1:2014.

You should determine the appropriate safety/compliance testing needed for the devices you build.