

Internet of Things Maker Den Lab Guide

Twitter #makerden #iot #netmf

Document Author: Dave Glover | dglover@microsoft.com | @dglover

Document Reviewer: Andrew Coates | acoat@microsoft.com | @coatsy

Doc Version 2.0

The latest version of this document can be found at <http://aka.ms/makerden-docs>

The latest version on the Maker Den Lab Solution can be found at <http://aka.ms/makerden-source>

Maker Den in a Box for the complete solution including utilities www.github.com/makerden

Disclaimer

All duty and care has taken to ensure the accuracy of this document. However, no liability is accepted.

Copyright

You are free to reuse and modify this document and associated software. Please acknowledge the original document author and reviewer @dglover and @coatsy on Twitter.

CONTENTS

Internet of Things Maker Den Lab Guide.....	1
Introduction.....	3
Time Required	3
Spread the word.....	3
Skills required	3
Prototyping Gear for lab use only	3
Mini Labs	5
Lab 1: Prototyping.....	5
Resetting the Labs	9
Opening the Visual Studio Lab Project	9
Lab 2: Blinky.....	10
Lab 3: Sensing the World	11
Lab 4: Connecting to Azure.....	12
Lab 5: Command and Control.....	13
Lab 6: Lighting up with NeoPixels.....	14
Internet of Things Dashboard.....	15
Appendix	16
Trouble Shooting	16
Software Requirements.....	17
Lab Parts	17
Initial Hardware Setup.....	18
Maker Den Solution Architecture.....	19
Maker Den Solution Implementation.....	19
Sensor Base (Glovebox.microframework project).....	20
Service Manager (Glovebox.microframework project)	20
Configuration Manager (Glovebox.microframework project)	20
Utilities (Glovebox.microframework project).....	20
Command and Control.....	20
Reference Material.....	22
Blogs.....	22
Devices.....	22
Twitter	22
Useful Apps.....	22
References	22

INTRODUCTION

Welcome to the Internet of Things Maker Den lab where you'll get firsthand experience with hardware prototyping and deploying .NET C# code to a Netduino to bring your creation to life.

The goal of the lab is to learn something about wiring circuits (with plenty of guidance), deploying code and streaming your sensor data to Microsoft Azure.

We hope you have fun, get more curious and start dreaming about the 3rd wave of the internet.

GETTING STARTED

If you are setting up your own Maker Den device then refer to the appendix for information on the software setup including Visual Studio, hardware setup, troubleshooting including updating the firmware and setting the MAC Address on your Netduino 2 Plus. All source code is available at www.github.com/makerden.

TIME REQUIRED

The lab will take approximately 15 minutes. You are more than welcome to stay longer and delve a little deeper.

SPREAD THE WORD

Be sure to spread the word about the Internet of Things Maker Den on Twitter. Use hash tags #makerden #iot #netmf

SKILLS REQUIRED

Some dexterity to add a couple of sensors to a breadboard and some experience with Visual Studio will be useful. Coding skills are not essential, as long as you can follow "copy and paste" instructions.

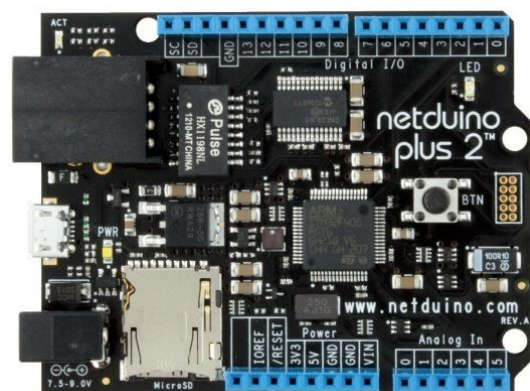
PROTOTYPING GEAR FOR LAB USE ONLY

These labs are based on the Netduino Plus 2, you can find out more about the device at www.netduino.com. The Netduino is essentially an Arduino that runs the .NET Micro Framework and can be programmed in C# or VB.NET.

There is a selection of RGB (Red Green Blue) LEDs, Light Dependent Resistors (to sense light levels), and Temperature sensors.

The .NET Micro Framework and Netduino SDKs have been installed on to the development PC from the Netduino [download](#) page.

When you have completed the lab please remove the RGB LED and two sensors from the breadboard so it is ready for the next person. Do not remove any wires from the breadboard Protoshield.



The Netduino is based on the ARM 32-bit Cortex™-M4 Microcontroller (aka a System on a Chip (SoC)). The device has an Ethernet port plus a Micro SD Card reader and retails for about \$70. They are physically and electronically compatible with the Arduino as well as most Arduino shields. The Netduino is a 3.3v board that is 5v tolerant.



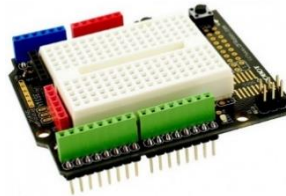
MINI LABS

There are 6 mini labs, all source code is provided and there are detailed diagrams on how to wire up the components. Feel free to explore, delve, be curious, and have fun.

LAB 1: PROTOTYPING

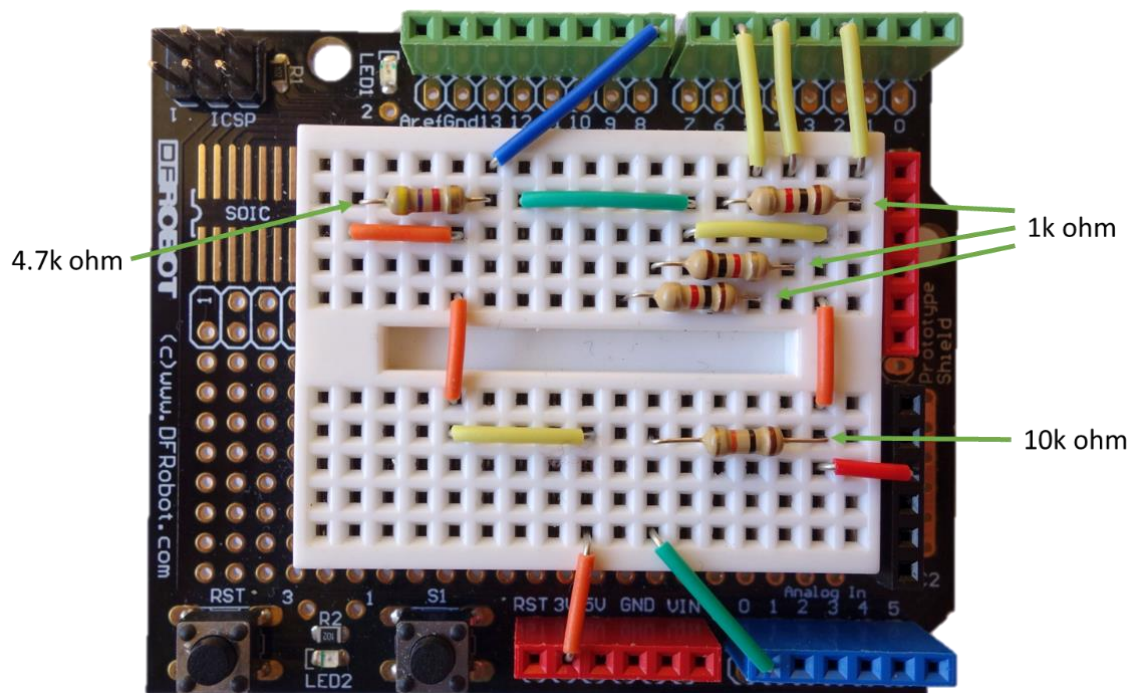
The first lab involves setting up two sensors and one RGB LED. It's important to follow the guide as the temperature sensor and the RGB LED must be inserted into the breadboard the correct way.

WARNING: Please ensure the Netduino is disconnected from the USB cable and powered down before you start adding components.

Temperature Sensor (DS18B20)	Light Dependent Resistor (LDR) aka Photocell	RGB (Red Green Blue) LED	Breadboard Protoshield
A digital temperature sensor that communicates over the 1-Wire protocol.	A Light Dependent Resistor changes its resistance depending on light levels.	Three LEDs in one package. The long pin is ground/common.	A piggy back shield with a breadboard. Very useful for prototyping without a soldering iron.
			

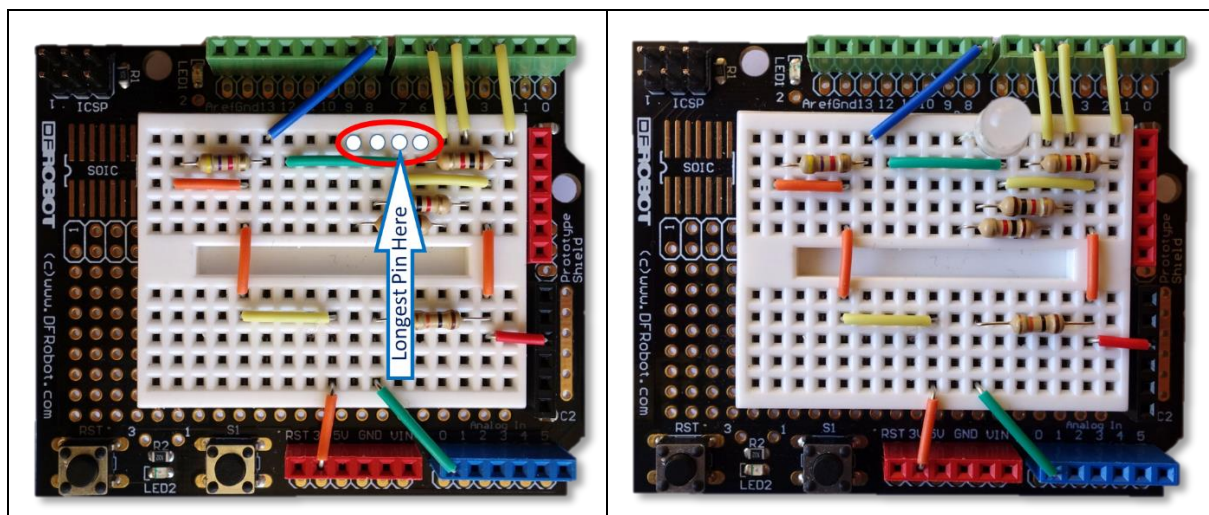
Follow these detailed instructions to set up the sensors on the breadboard Protoshield. Below are before and after images. Be sure your prototype matches the “after” image when you’ve completed the hardware setup.

Your Breadboard Protoshield should look like the image below before you add the sensors and RGB LED¹.



ADDING THE RGB LED

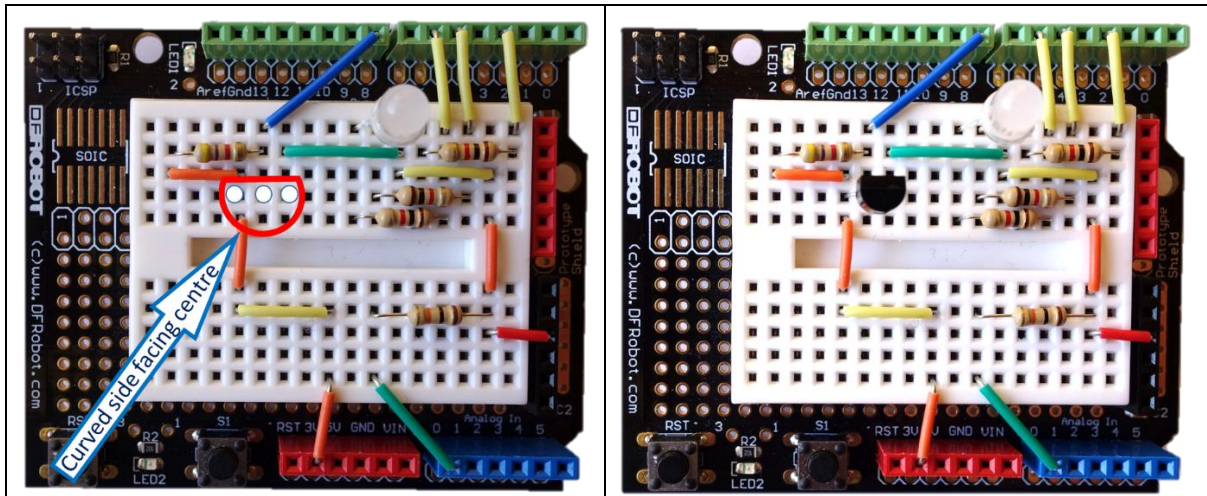
The RGB LED must be inserted in to the breadboard as per the image below. Ensure you insert the LED's longest pin as per the image below.



¹ Breadboards work on the principle that each of the 5 sockets in a column are wired in common, so inserting a component into a socket in the same column as another component, those components will be electrically coupled, as though their wires were joined. This makes it very simple to quickly create and modify electrical circuits without a necessity for soldering, clips or junction boxes.

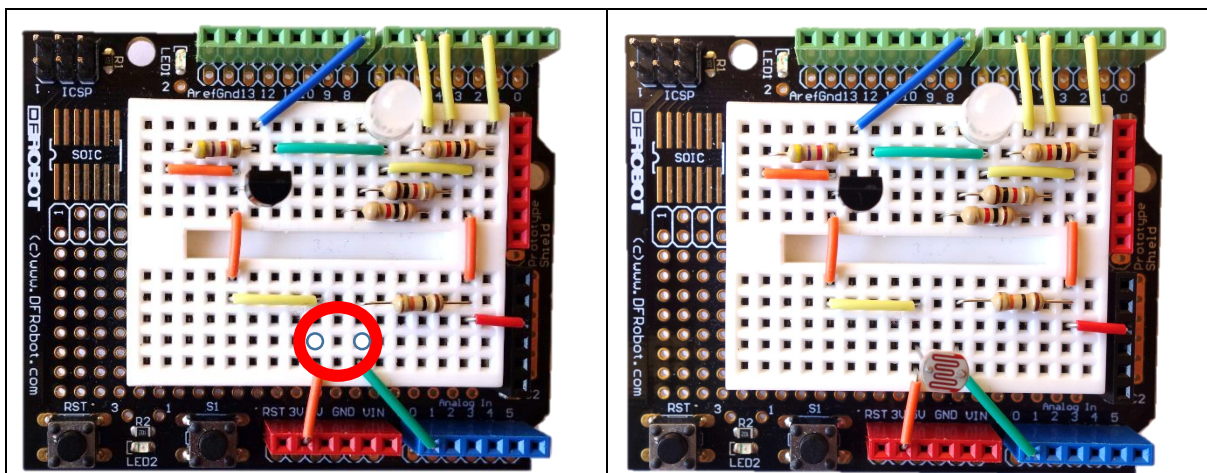
ADDING THE TEMPERATURE SENSOR

The temperature sensor must be inserted in to the breadboard as per the image below.



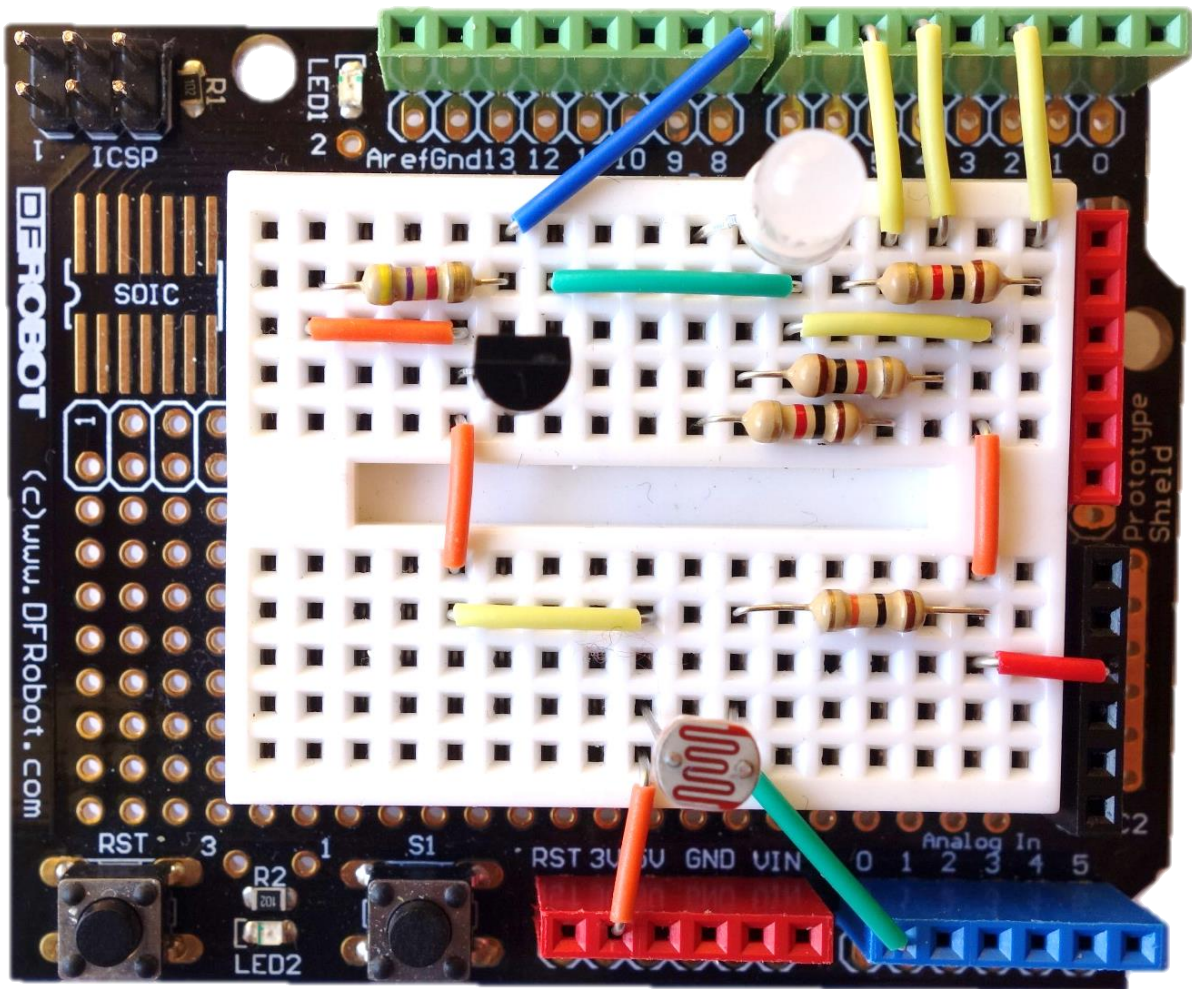
ADDING THE LIGHT DEPENDENT RESISTOR (LDR)

The Light Dependent Resistor can be inserted into the breadboard either way.



Your completed breadboard should look like this.

Ensure the temperature sensor is the correct way around before powering up the device.

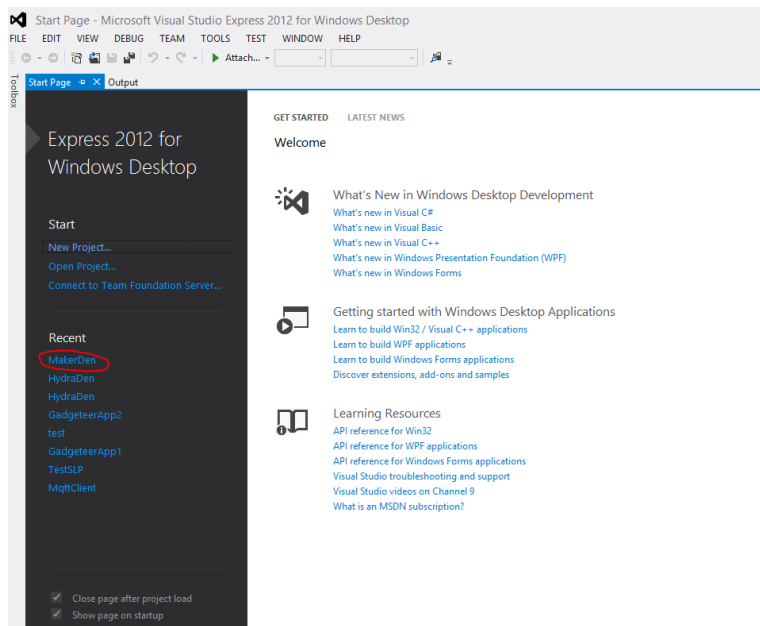


RESETTING THE LABS

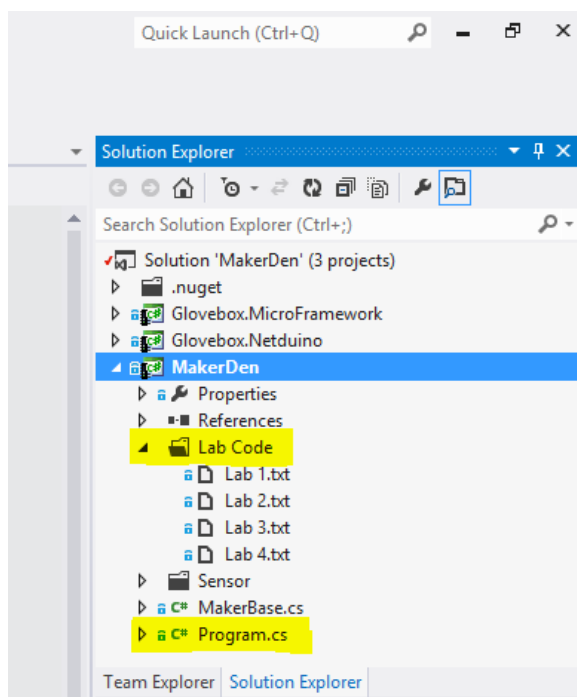
1. Ensure that Visual Studio is closed.
2. On the desktop there is a **ResetLabs.bat** file.
3. Double click to run the script to reset the labs.

OPENING THE VISUAL STUDIO LAB PROJECT

Visual Studio is pinned to the Windows Taskbar. Click it to start, then open the MakerDen solution from the list of projects under Recent on the left hand side of the screen.



The code for all the labs can be found in the **Lab Code** folder. All code will be copy and pasted in to the **Program.cs** file.



LAB 2: BLINKY

Blinking an LED is the “Hello World” of Microcontrollers. It is useful to ensure that everything is setup correctly and that Visual Studio can communicate with the Netduino.

Ensure the Netduino is connected to the PC using the USB cable and then follow these steps.

- 1) Review the code below. On completion of this lab, your code should be the same
- 2) From Visual Studio open the MakerDen solution
- 3) Follow the notes in the Lab 2 (found in the Lab Code folder) and copy and paste the code in to Program.cs
- 4) Press F5 or the click the Start button from the toolbar to run the code on the Netduino
- 5) It will take approx. 30 seconds for the code to deploy to the Netduino
- 6) The red LED will start blinking when the program executes (if it blinks blue then you have put the LED in the wrong way round)
- 7) You can also set breakpoints and step through the code as it executes.

```
using Glovebox.MicroFramework.Sensors;
using Glovebox.Netduino;
using Glovebox.Netduino.Sensors;
using Microsoft.SPOT;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using System.Threading;

namespace MakerDen
{
    public class Program : MakerBaseIoT
    {
        public static void Main()
        {
            // main code marker

            using (rgb = new RgbLed(Pins.GPIO_PIN_D3, Pins.GPIO_PIN_D5, Pins.GPIO_PIN_D6, "rgb01"))
            {
                while (true)
                {
                    rgb.On(RgbLed.Led.Red);
                    Thread.Sleep(500);
                    rgb.Off(RgbLed.Led.Red);
                    Thread.Sleep(500);
                } // End of while loop
            }
        }
    }
}
```

LAB 3: SENSING THE WORLD

This lab reads the current levels from the Temperature and Light Sensors.

Ensure the Netduino is connected to the PC using the USB cable and then follow these steps.

- 1) Review the code below. On completion of this lab, your code should be the same
- 2) If the program is still running from Lab 2, then click the stop button first. You will not be able to edit the Program.cs file while the code executes.
- 3) Follow the notes in the Lab 3 and copy and paste the code in to Program.cs
- 4) Press F5 or the click the Start button from the toolbar to run the code on the Netduino
- 5) The code will deploy to the Netduino
- 6) The red LED should start blinking
- 7) In Visual Studio press Alt+2 to switch to the Output screen. You will see the sensor data displayed in JSON format.

```
using Glovebox.MicroFramework.Sensors;
using Glovebox.Netduino;
using Glovebox.Netduino.Sensors;
using Microsoft.SPOT;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using System.Threading;

namespace MakerDen
{
    public class Program : MakerBaseIoT
    {
        public static void Main()
        {
            // main code marker

            using (SensorTemp temp = new SensorTemp(Pins.GPIO_PIN_D8, 10000, "temp01"))
            using (SensorLight light = new SensorLight(AnalogChannels.ANALOG_PIN_A0, 1000, "light01"))
            using (rgb = new RgbLed(Pins.GPIO_PIN_D3, Pins.GPIO_PIN_D5, Pins.GPIO_PIN_D6, "rgb01"))
            {
                while (true)
                {
                    Debug.Print(light.ToString());
                    Debug.Print(temp.ToString());

                    rgb.On(RgbLed.Led.Red);
                    Thread.Sleep(500);
                    rgb.Off(RgbLed.Led.Red);
                    Thread.Sleep(500);
                } // End of while loop
            }
        }
    }
}
```

LAB 4: CONNECTING TO AZURE

This lab connects the Netduino to a MQTT² Service running on Azure. When you have successfully completed this lab your data will be displayed on the central Internet of Things dashboard.



You can also run the Internet of Things dashboard app from the PC you are using. It is pinned to the taskbar next to Visual Studio.

Ensure the Netduino is connected to the PC using the USB cable and the Ethernet cable is plugged in to the Netduino. Then follow these steps.

- 1) Review the code below. On completion of this lab, your code should be the same
- 2) If the program is still running from Lab 3, then click the stop button first. You will not be able to edit the Program.cs file while the code executes.
- 3) Follow the notes in the Lab 4 and copy and paste the code in to Program.cs
- 4) Replace “**emul**” with something unique such as your initials, use 3 to 5 characters.
- 5) Ensure the network cable is plugged in to the Netduino and connected to the internet.
- 6) Press F5 or the click the Start button from the toolbar to deploy the app to the Netduino
- 7) The RGB LED blinks when a sensor is read (this can take up to 30 seconds to begin)
- 8) Check the central Internet of Things dashboard; your data should be displayed. Try adjusting the light reading by covering the LDR. You’ll see the change reflected on the IoT Dashboard.

```
using Glovebox.MicroFramework.Sensors;
using Glovebox.Netduino;
using Glovebox.Netduino.Sensors;
using Microsoft.SPOT;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using System.Threading;

namespace MakerDen
{
    public class Program : MakerBaseIoT
    {
        public static void Main()
        {
            // main code marker

            //Replace the "emul" which is the name of the device with a unique 3 to 5 character name
            //use your initials or something similar. This code will be visible on the IoT Dashboard
            StartNetworkServices("emul", true);

            using (SensorTemp temp = new SensorTemp(Pins.GPIO_PIN_D8, 10000, "temp01"))
            using (SensorLight light = new SensorLight(AnalogChannels.ANALOG_PIN_A0, 1000, "light01"))
            using (RgbLed rgb = new RgbLed(Pins.GPIO_PIN_D3, Pins.GPIO_PIN_D5, Pins.GPIO_PIN_D6, "rgb01"))
            {
                light.OnBeforeMeasurement += OnBeforeMeasure;
                light.OnAfterMeasurement += OnMeasureCompleted;
                temp.OnBeforeMeasurement += OnBeforeMeasure;
                temp.OnAfterMeasurement += OnMeasureCompleted;
                Thread.Sleep(Timeout.Infinite);
            }
        }
    }
}
```

² Message Queue Telemetry Transport

LAB 5: COMMAND AND CONTROL

See the Command and Control appendix for a brief explanation on how control messages work.

Ensure the Netduino is connected to the PC using the USB cable and the Ethernet cable is plugged in to the Netduino. Then follow these steps.

On completion of this lab the modified StartNetworkServices line of code should be similar to the code snippet below.

- 1) If the program is still running from Lab 4, then click the stop button first. You will not be able to edit the Program.cs file while the code executes.
- 2) In the Program.cs file modify the **StartNetworkServices("emul", true);** line to include a unique device identifier. Use the MAC Address printed on the underside of the board, your email address or something similar. Your code should look similar to **StartNetworkServices("emul", true, "5C-E6-H2-00-6B");**.
- 3) Ensure the network cable is plugged in to the Netduino and there is an active internet connection
- 4) Press F5 or the click the Start button from the toolbar to deploy and run your code on the Netduino.
- 5) Check the central Internet of Things dashboard; your data should be displayed.

```
using Glovebox.MicroFramework.Sensors;
using Glovebox.Netduino;
using Glovebox.Netduino.Sensors;
using Microsoft.SPOT;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using System.Threading;

namespace MakerDen
{
    public class Program : MakerBaseIoT
    {
        public static void Main()
        {
            // main code marker

            //Replace the "emul" which is the name of the device with a unique 3 to 5 character name
            //use your initials or something similar. This code will be visible on the IoT Dashboard
            StartNetworkServices("emul", true, "your device identifier goes here");

            using (SensorTemp temp = new SensorTemp(Pins.GPIO_PIN_D8, 10000, "temp01"))
            using (SensorLight light = new SensorLight(AnalogChannels.ANALOG_PIN_A0, 1000, "light01"))
            using (rgb = new RgbLed(Pins.GPIO_PIN_D3, Pins.GPIO_PIN_D5, Pins.GPIO_PIN_D6, "rgb01"))
            {
                light.OnBeforeMeasurement += OnBeforeMeasure;
                light.OnAfterMeasurement += OnMeasureCompleted;
                temp.OnBeforeMeasurement += OnBeforeMeasure;
                temp.OnAfterMeasurement += OnMeasureCompleted;
                Thread.Sleep(Timeout.Infinite);
            }
        }
    }
}
```

- 6) Next we will issue control messages to the Netduino from the IoT Dashboard application.

LAB 6: LIGHTING UP WITH NEOPIXELS

The lab requires a NeoPixel Ring, a NeoPixel Grid, or a NeoPixel Strip

You need to change the class that Program inherits from. If you are using a NeoPixel Ring, change the base class to be `MakerDenNeoPixelRing`. If you have a NeoPixel Grid, then you need to inherit from `MakerDenNeoPixelGrid`, if a NeoPixel Strip then inherit from `MakerDenNeoPixelStrip`

```
using Glovebox.MicroFramework.Sensors;
using Glovebox.Netduino;
using Glovebox.Netduino.Sensors;
using Microsoft.SPOT;
using SecretLabs.NETMF.Hardware.NetduinoPlus;
using System.Threading;

namespace MakerDen
{
    public class Program : MakerBaseNeoPixelRing
    {
        public static void Main()
        {
            // main code marker

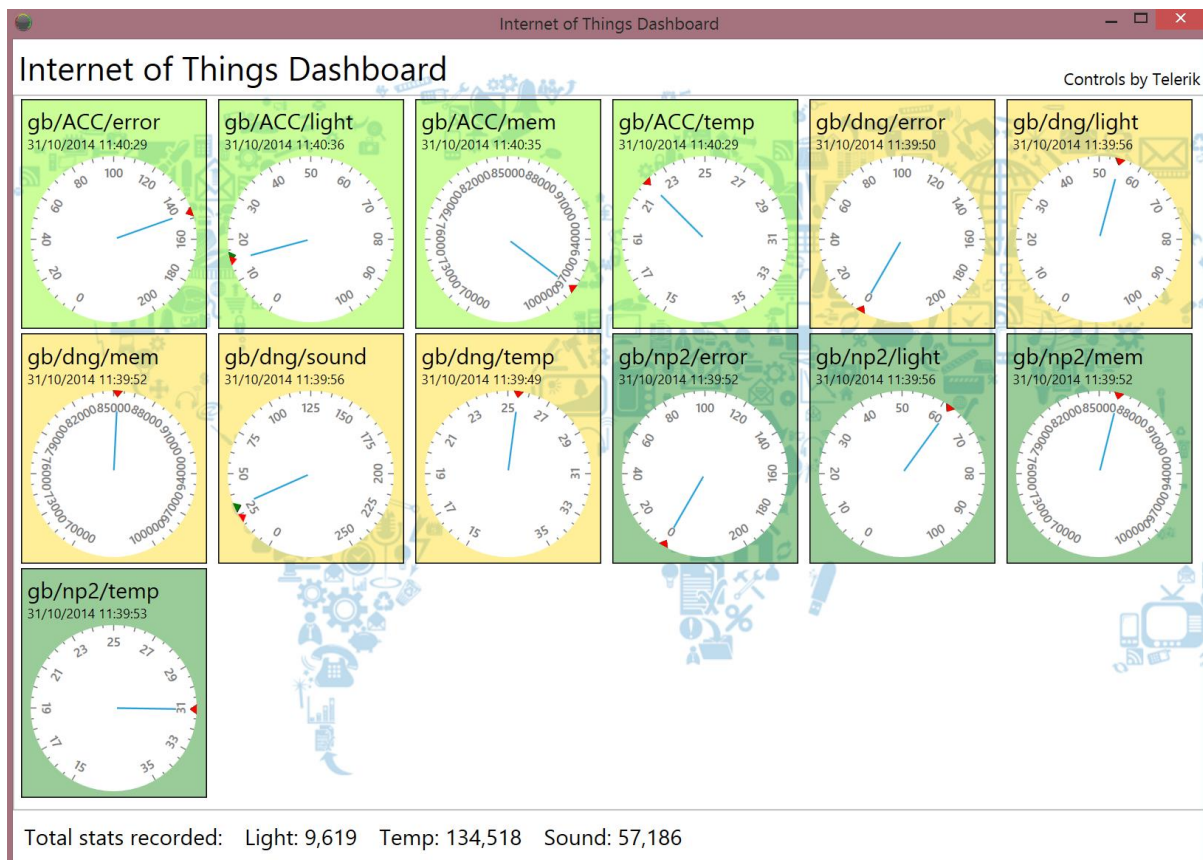
            StartNeoPixel();

            //Replace the "emul" which is the name of the device with a unique 3 to 5 character name
            //use your initials or something similar. This code will be visible on the IoT Dashboard
            StartNetworkServices("emul", true, "your device identifier goes here");

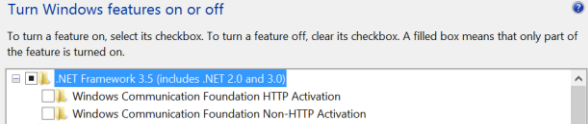
            using (SensorTemp temp = new SensorTemp(Pins.GPIO_PIN_D8, 10000, "temp01"))
            using (SensorLight light = new SensorLight(AnalogChannels.ANALOG_PIN_A0, 1000, "light01"))
            using (rgb = new RgbLed(Pins.GPIO_PIN_D3, Pins.GPIO_PIN_D5, Pins.GPIO_PIN_D6, "rgb01"))
            {
                light.OnBeforeMeasurement += OnBeforeMeasure;
                light.OnAfterMeasurement += OnMeasureCompleted;
                temp.OnBeforeMeasurement += OnBeforeMeasure;
                temp.OnAfterMeasurement += OnMeasureCompleted;
                Thread.Sleep(Timeout.Infinite);
            }
        }
    }
}
```

INTERNET OF THINGS DASHBOARD

You'll find the Internet of Things Dashboard Application pinned to the Windows Taskbar and it will also be running on the central Maker Den screen.



TROUBLE SHOOTING

Problem	Solution
Firmware level does not match	You need to update the firmware on the Netduino. See updating firmware instructions.
Error code 0x80131700 at compile time	<ol style="list-style-type: none"> 1) Ensure the “.NET Framework 3.5 (including .NET 2.0 and 3.0)” Windows feature is installed.  2) Ensure .NET Micro Framework Installed 3) Ensure .NET Framework 3.5 Installed
Netduino network issues. A DHCP IP address does not get allocated or there are network connectivity issues.	<p>Out of the box (or if the Netduino is re-flashed) the MAC address is not configured on the Netduino Plus 2.</p> <p>Using the .NET Micro Framework Deploy Tool (search MFDeploy) select USB from the Device dropdown. From the main menu, select Target -> Configuration -> Network and set the MAC Address.</p>
Problems deploying code to the Netduino. Persistent device connection errors.	<ol style="list-style-type: none"> 1) First call to action is to recycle the power to the device by unplugging and reinserting the USB cable and try again 2) If connection problem persists then <ol style="list-style-type: none"> a. Right mouse click the project you want to deploy and select properties b. Select .NET Micro Framework tab c. From the Transport dropdown select Emulator, then reselect USB. d. Your project should deploy 3) Failing this, then re-flash the firmware. Ensure you reset the MAC address using the .NET Micro Framework Deploy Tool (search MFDeploy).

SOFTWARE REQUIREMENTS

The following software is required (as at Oct 2014).

- 1) Windows 7 or above
- 2) [Visual Studio 2012 Express](#) (or [Visual Studio 2013 with Beta MF SDK](#))
- 3) [.NET Micro Framework SDK v4.3 QFE1](#)
- 4) [Netduino SDK v4.3.1](#)
- 5) [Netduino Plus 2 Firmware](#) (No installation needed. Required for updating/resetting the device firmware)
- 6) Ensure you set the MAC address on the Netduino using the .NET Micro Framework Deploy Tool (search MFDeploy).
- 7) Access to an [MQTT Server](#) such as [Mosquitto](#) which is easy to install locally. Mosquitto host a test MQTT Service at test.mosquitto.org.

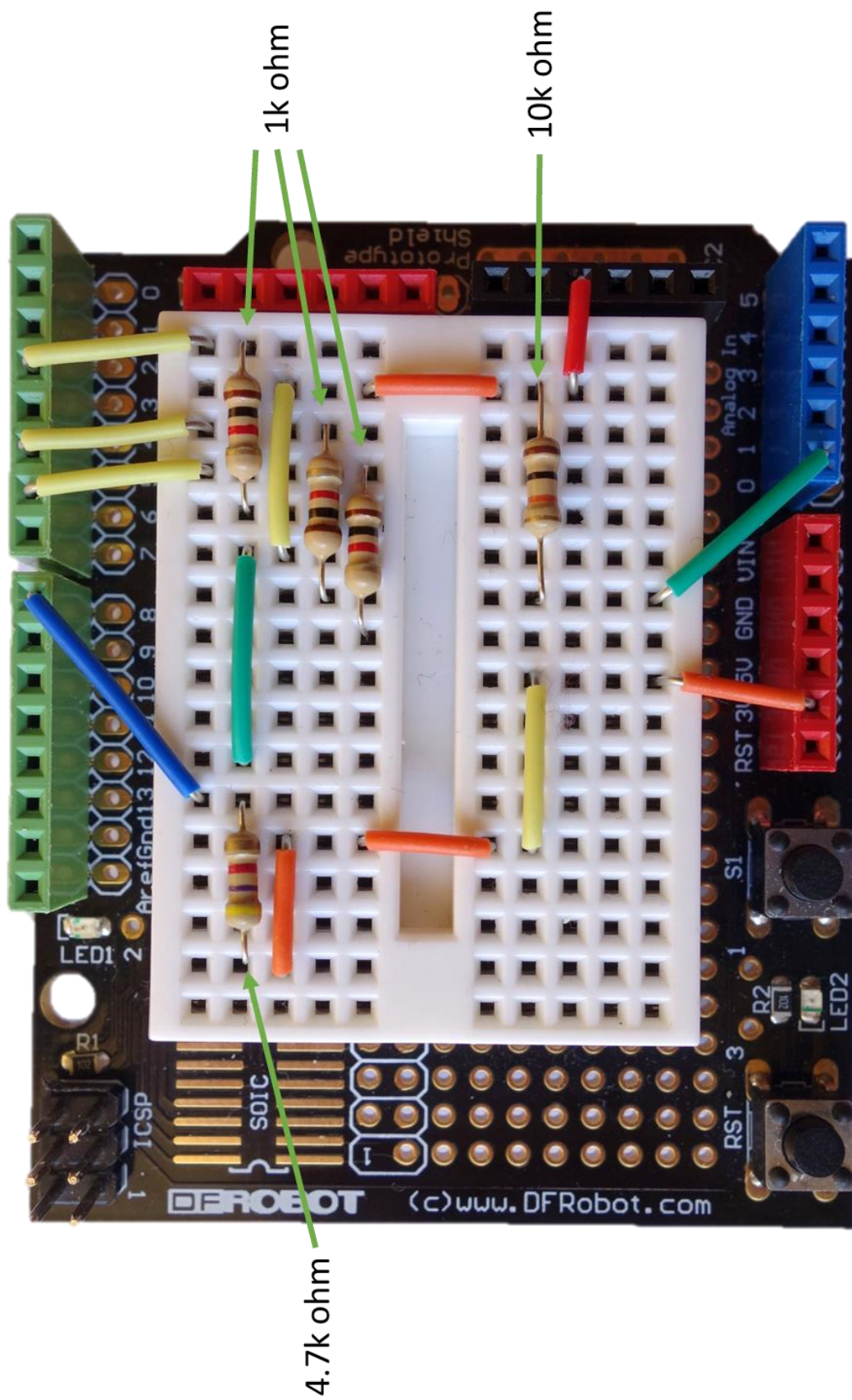
LAB PARTS

The following parts are required for this lab.

In Australia these parts were purchased from [Little Bird Electronics](#). Microsoft Australia has no affiliation with Little Bird Company Pty Ltd. Equivalent parts can easily be purchased elsewhere on the web including [Robot Gear](#), [Australian Robotics](#), [eBay](#), plus others.

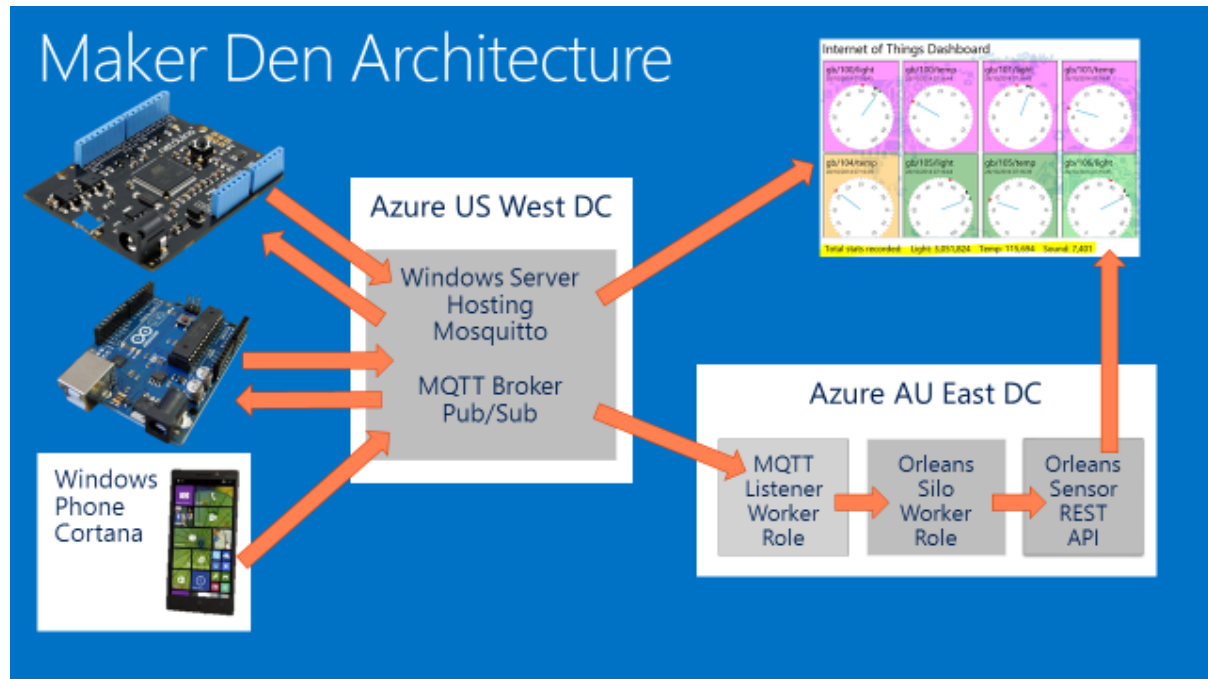
- 1) [Netduino Plus 2](#)
- 2) [Breadboard Protoshield](#)
- 3) [Breadboard Jumper Kit](#)
- 4) [1 x RGB Led](#)
- 5) [1 x Mini Photocell](#)
- 6) [1 x Temperature Sensor \(DS18B20\)](#)
- 7) Recommend a [Resistor Kit](#) or individual resistors
 - a. 3 x 1k ohm resistors
 - b. 1 x 10k ohm resistor
 - c. 1 x 4.7k ohm resistor

INITIAL HARDWARE SETUP



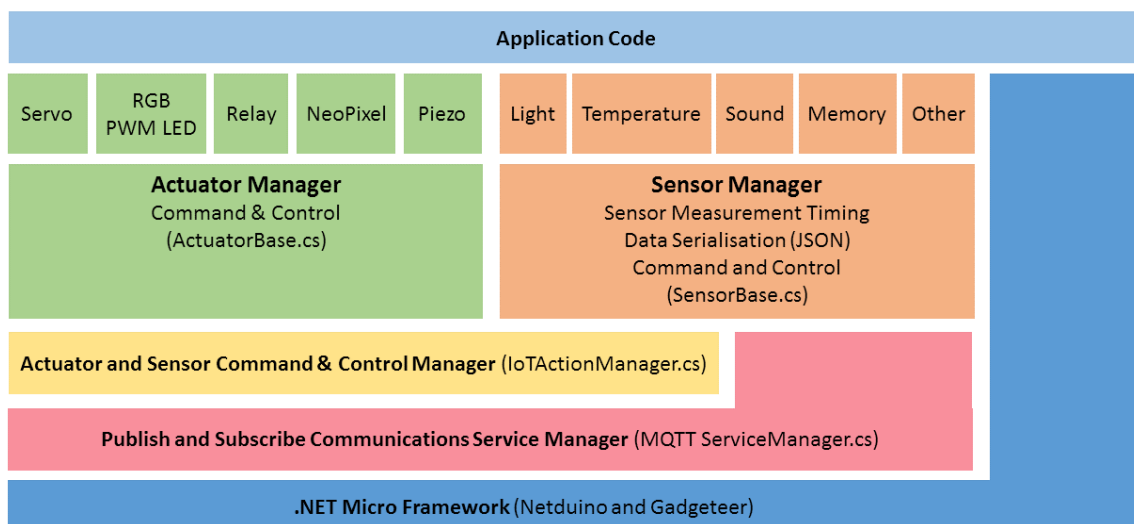
MAKER DEN SOLUTION ARCHITECTURE

The Maker Den architecture is based on the Message Queue Telemetry Transport ([MQTT](#)) service supported by the [M2MQTT](#) Visual Studio NuGet package and the open source [Mosquitto](#) Broker implementation running on a virtual Windows Server hosted on Azure.



MAKER DEN SOLUTION IMPLEMENTATION

Maker Den IoT Framework



SENSOR BASE (GLOVEBOX.MICROFRAMEWORK PROJECT)

The Maker Den IoT Framework provides a pluggable foundation to support sensors, actuators, data serialisation, communications, and command and control.

The heart of the Maker Den solution is the SensorBase class, all sensors inherit from this class. The SensorBase class is responsible for periodic sensor measurement and serialisation of the measurement data to JSON.

The SensorBase class is defined as follows:-

1. The SensorBase constructor requires the sensor sample rate in milliseconds (or -1 if user code controlled sensor sampling), the number of values a sensor will return, and the sensor type. See the Sensor folder for examples of sensor implemenations. SensorMemory and SensorError are the simplist.
2. Two abstract methods and one abstract property that an inheriting class must implement
 - a. Measure(double[]) returns the current sensor value(s). Most sensor return one value, but some sensor will return multiple values, for example a combined temperature and humidity sensor.
 - b. GeoLocation() returns the current geolocation. Geolocation is implementation specific, it could be a long lat, postal code, room number etc
 - c. Current() is the current sensor value. Provides direct access to the current sensor value
3. Two events
 - a. OnBeforeMeasurement allows arbitrary code execution before a sensor measurement, for example blink an LED.
 - b. OnAfterMeasurement allows arbitrary code after a sensor measurement to be executed, for example publish the sensor reading to an Azure service.

SERVICE MANAGER (GLOVEBOX.MICROFRAMEWORK PROJECT)

Data publishing is managed by the ServiceManager class. The Maker Den Service Manager implements MQTT services to publish and receive data using the [M2MQTT](#) NuGet package. The Service Manager could equally implement HTTP REST calls to a web service, or perhaps communications over Bluetooth, whatever makes sense for a particular scenario.

CONFIGURATION MANAGER (GLOVEBOX.MICROFRAMEWORK PROJECT)

Useful configuration details for the solution. For example the URI for the MQTT Broker Service.

UTILITIES (GLOVEBOX.MICROFRAMEWORK PROJECT)

NTP Time Service and Byte Array and String converters.

COMMAND AND CONTROL

The Maker Den IoT Framework supports command and control making it possible to send control messages from Azure to the device to change its state or operation.

The underlying network services implementation for the IoT Framework supports publishing data and subscribing to control messages. A sensor or an actuator is declared with a name that when combined with the device id creates a control channel that messages can be sent to.

A control message is consists of a channel, an action and optional parameters.

MULTICAST CONTROL MESSAGES

Devices can respond to control messages sent to all devices listening on the same base channel.

In these examples the base channel is `gbcmd/all`

1. To stop all light sensors named "light01" on devices listening on the same channel you'd send a control message `gbcmd/all/stop/light01`.
2. To start all light sensor named "light01" you'd send a control message `gbcmd/all/start/light01`.
3. To make an RGB LED glow red you'd send `gbcmd/all/on/rgb01/red`, to turn off you'd send `gbcmd/all/off/rgb01/red`.
4. If you had a piezo speaker attached to the Netduino you could send a message to play a tune using `gbcmd/all/play/piezo01` plus a parameters to describing the tune. For example `{"BeatsPerMinute":"60", "Score":"3qc#,31c"}`. This sequence would be played at 60 beats per minute, the first tone in the 3rd octave, as a quarter beat in c#, the second note in the 3rd octave, 1 beat in duration, in c.

UNICAST CONTROL MESSAGES

You can also send a control message to a sensor or actuator on a specific device.

When you `StartNetworkServices` you can also pass in a parameter to specify a unique identifier for the device, it could be the MAC address of the board, a Guid, a device asset name etc.

Example: `StartNetworkServices("emul", true, "5C-AB-9C-00-A5-92");`

This example uses the Netduino MAC address as the device unique identifier. Be sure to use the MAC address for the board you are using not the one in this example.

To send a control message to this specific Netduino would be as follows:-

- 1) `gbcmd/dev/5C-AB-9C-00-A5-92/stop/light01`
- 2) `gbcmd/dev/5C-AB-9C-00-A5-92/play/piezo01 {"BeatsPerMinute":"60", "Score":"3qc#,31c"}`

REFERENCE MATERIAL

- 1) [Netduino for Beginners](#)
- 2) [Getting Started with Netduino](#)
- 3) [Getting Started with the Internet of Things](#)
- 4) [Beginners Guide to C# and the .NET Micro Framework](#)
- 5) [Getting Started with .NET Gadgeteer](#)

BLOGS

[.NET Micro Framework](#)

[Windows on Devices](#)

[Embedded 101](#)

DEVICES

[Netduino](#) (<http://www.netduino.com>)

[Gadgeteer](#) (<https://www.ghielectronics.com> or <http://tinyclr.com>)

TWITTER

#makerden #iot #netmf

USEFUL APPS

[Resistor Calculator](#) (Windows 8)

[Resistance](#) (Windows 8)

[Resistor Identifier](#) (Windows Phone)

[LED Resistor](#) (Windows Phone)

[Mqtt Assistant](#) (Windows Phone)

[MQTTInspector](#) (iOS)

[Resistor Code Calculator](#) (iOS)

[LEDulator](#) (iOS)

REFERENCES

[How Light Dependent Resistors Work](#)

[1-Wire explained](#)

<http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf>