

Raspberry Pi Wireless inventors Kit – Azure extension.

Summary.

This lab uses a Raspberry Pi 2 as an IoT gateway device and a XinoRF as the sensor prototyping developer board. Using the provided Azure WIK application on the RPI2 enables the analog and digital pins of the XinoRF to be read and manipulated wirelessly. By connecting the XinoRF to components on the provide bread board different sensors and actuators can be built which can be driven from the RPI2 Azure WIK application and the readings can be automatically pushed to an Azure Event Hub, where the data can be processed using Azure Stream Analytics. Use of a simple custom event processor enables the received event hub messages to be monitored in real time on a PC. This monitor can provide the basis for a Signalr command architecture to send actions down to the RPI2 and on to the XinoRF from the processed sensor readings.

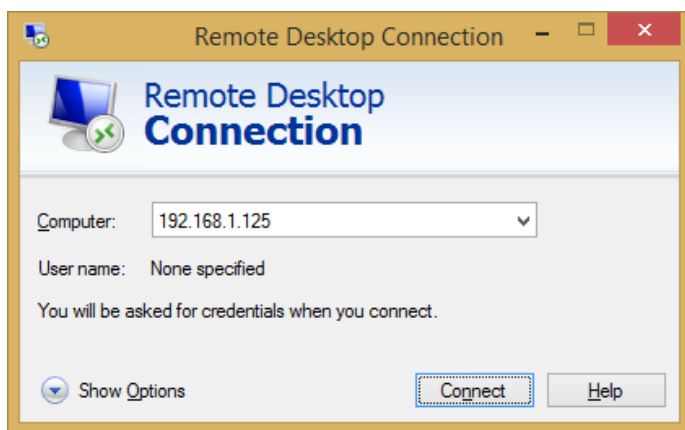
In this hands on lab you will build a series of sensor experiments on the XinoRF, test them via the RPI2 Azure WIK application and then publish the sensor readings live to an Azure Event Hub to be processed by Azure Stream Analytics. Effectively creating an end to end IoT implementation.

You should have completed the separate Event Hub and Azure Stream Analytics hands on labs prior to this lab.

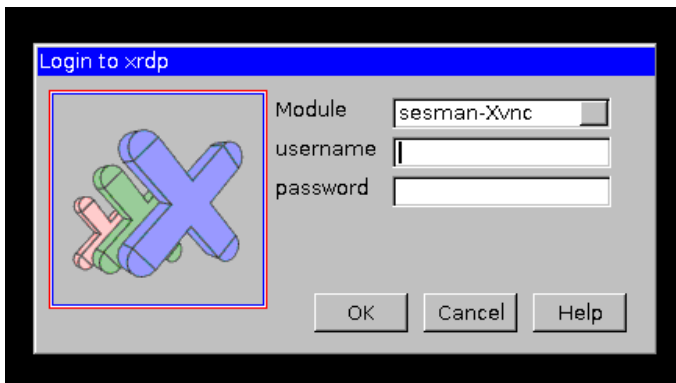
1. Setting up your Raspberry Pi and Xino RF sensor

The Raspberry Pi (RPI) has been pre-configured for the Wifi network and has the allocated IP address documented on it. You will need to use a PC with an RDP Remote Desktop Client to connect to the RPI. Plug the RPI in to the mains using the provided RPI power adapter. It will take a few seconds to boot.

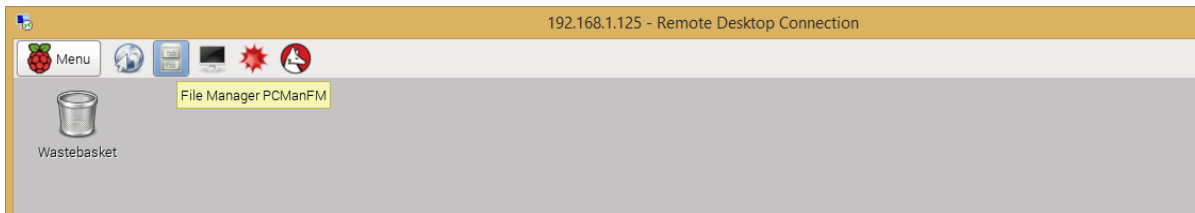
Using a Remote Desktop Client connect to the RPI. PC dialog requires the RPI IP Address. Ignore the security warns for the connection.



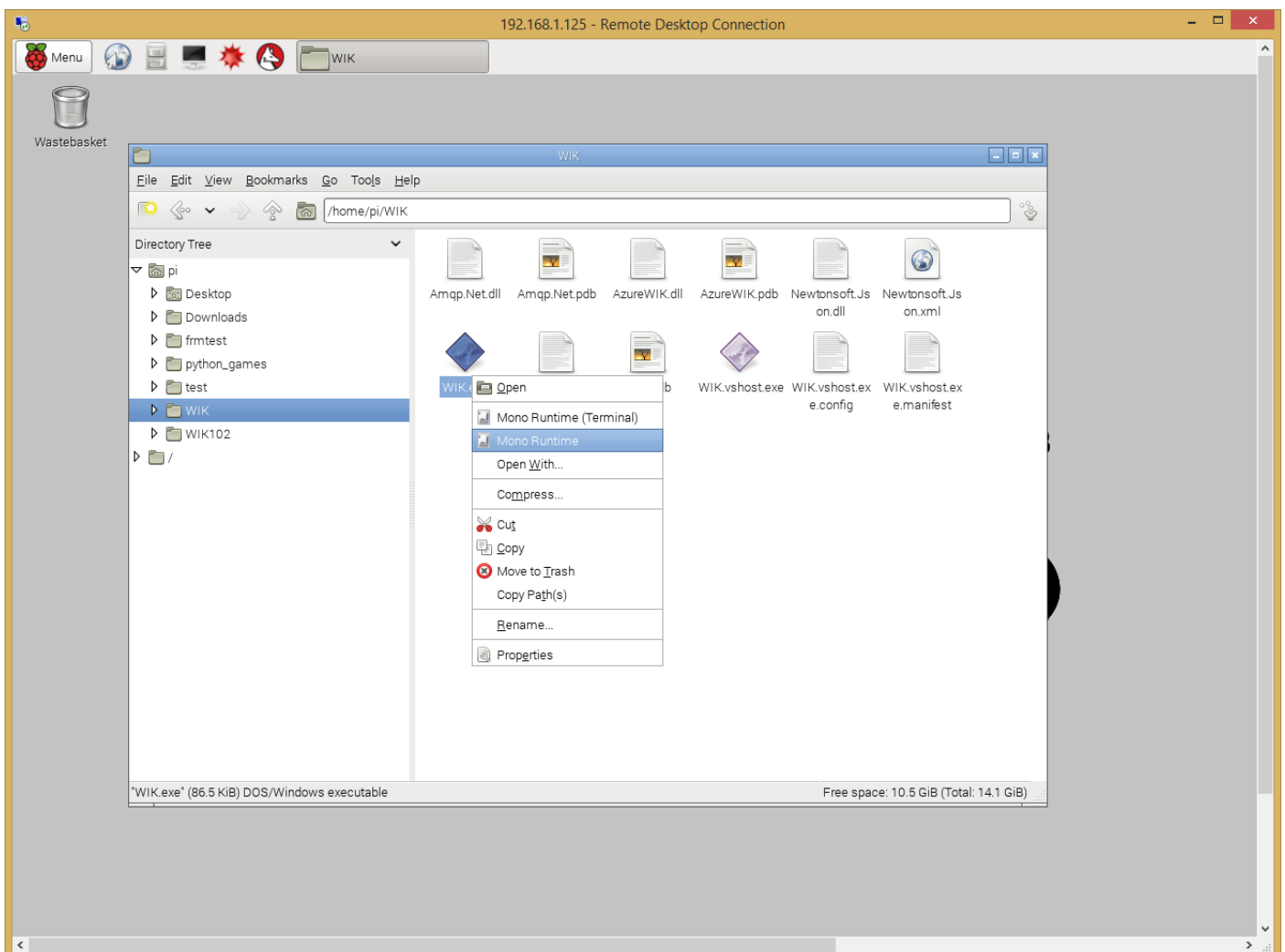
Once connected to the RPI a login screen will be presented. The user account details are the standard RPI ones, username = pi, password = raspberry.



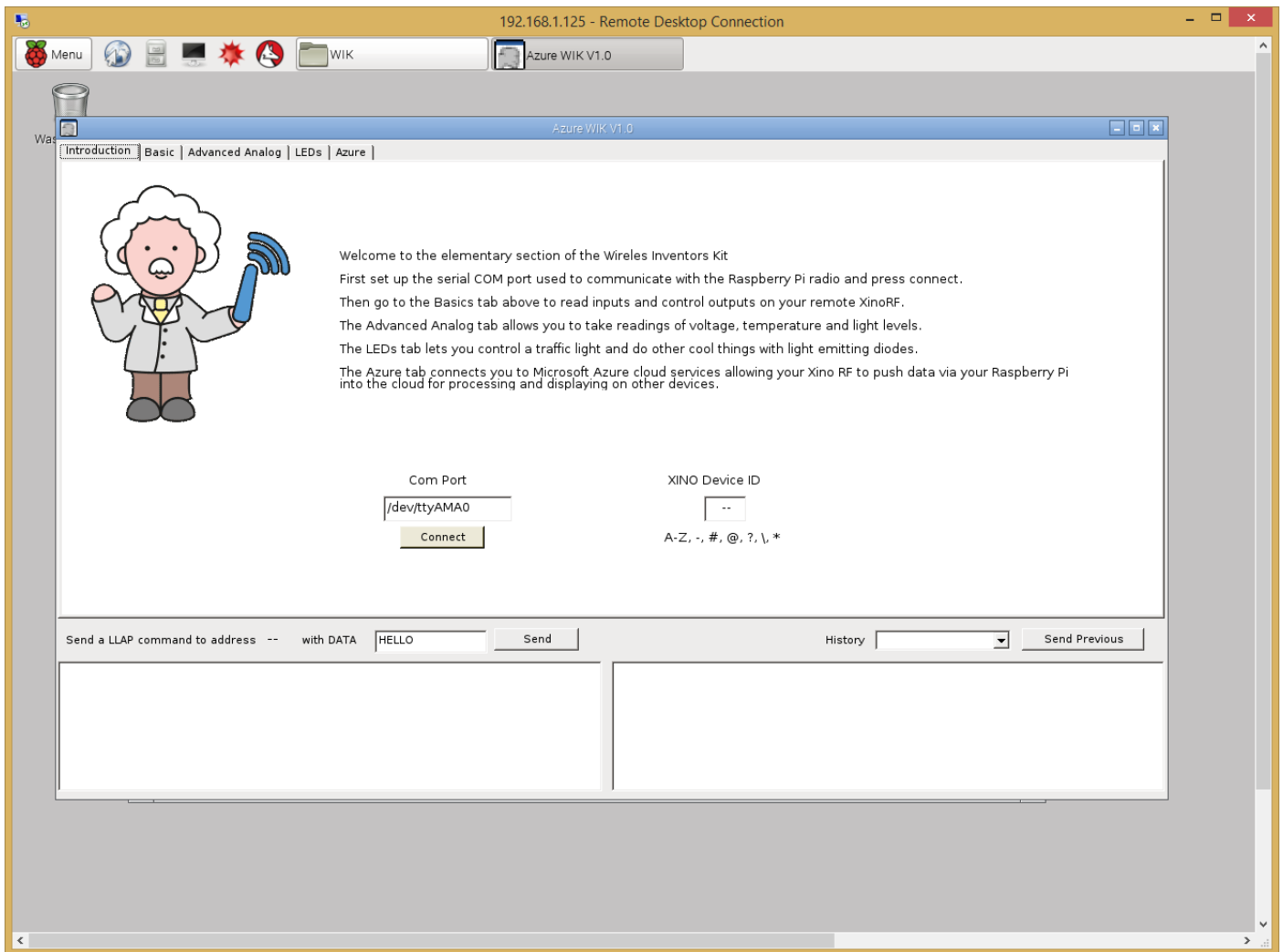
Once connected you will see a desktop environment.



On the taskbar select the filing cabinet to open the File Manager. Navigate to the WIK directory. Right click on the Wik.exe and select Mono Runtime. This will start the Azure WIK application.



The Azure WIK application will start as follows:



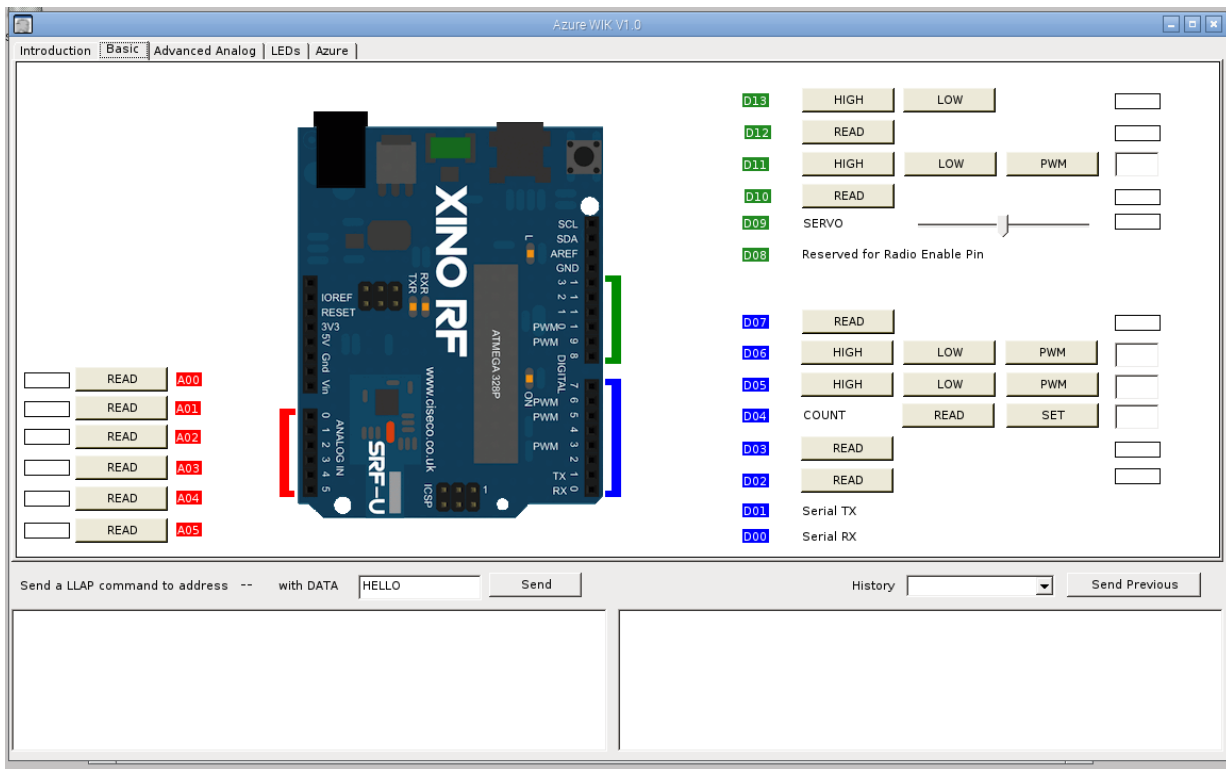
Before proceeding power up your XinoRF by connecting the provided USB cable to a suitable USB port on your PC. No drivers need to be installed, the XinoRF just requires power.

On the XinoRF will be a small sticker documenting the device's two character unique ID.

Enter this unique ID into the XINO device ID field in the Azure WIK application. The Com Port required for the RPI is defaulted so you can now press connect.

After a few seconds you can test your set up by pressing the Send button which will send a wireless message from the RPI to the XinoRF saying Hello. The XinoRF will echo a response to you which is shown in the bottom left hand corner. It may take a moment for the communications to be established.

You can now switch to the BASIC tab.



On the basic tab you can see a schematic of the XinoRF and application buttons that either read or set the analog (red) or digital (green and blue) pins.

Pin D13 is connected to the LED 'L' on the XinoRF. By pressing HIGH and LOW buttons for D13 in the application you can turn this LED on and off. The application is sending a LLAP message across the 868mhz radio link established between the RPI and the XinoRF. The RPI is fitted with an expansion card radio card to enable this link.

You are now ready to build experiments with the XinoRF and provided bread board and components. A wide range of experiments are at <http://www.openmicros.org/index.php/articles/81-xrf-projects/296-raspberry-pi-wireless-inventors-kit>

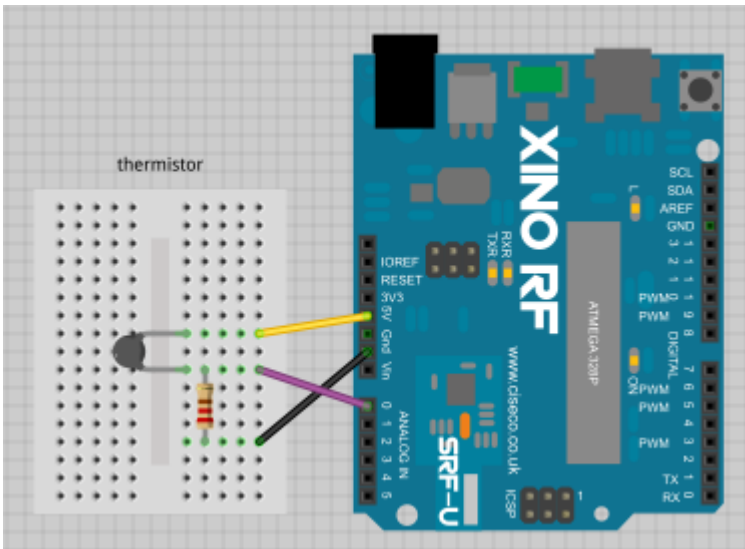
You can experiment as you like with any of the documented activities to get to grips with the XinoRF.

To proceed with this hands on lab you need to build a sensor to capture data to send to Azure. You can build either a temperature sensor or a light sensor. The circuit is the same for both, just the LDR or the thermistor needs to be fitted. So you can easily swap between both.

Building the temperature sensor.

Requirements:

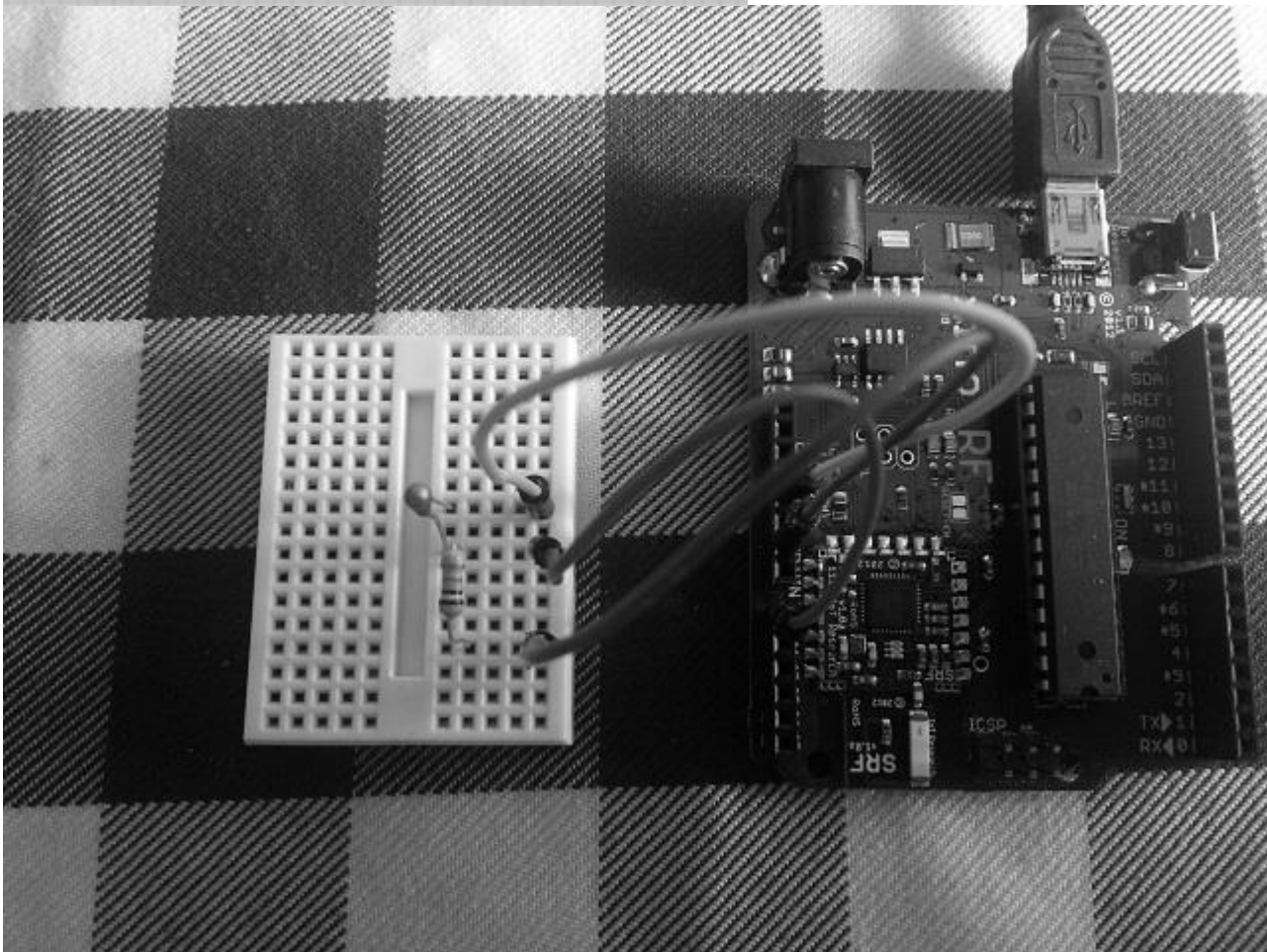
- 1 x XinoRF
- 1 x 10K resistor
- 1 x Thermistor
- 1 x Breadboard
- 3 x Jumper wire



Assemble the components as illustrated in the photographs.

Yellow lead is from 5V
Black lead is from GND
Purple lead is from A0

Gently push the thermistor and the resistor into the bread board as shown. Join the jump leads to finish the circuit.



Switch the Azure WIK application to the Advanced Analog tab. Press the READ button and observe the readings.

Azure WIK V1.0

Introduction | Basic | Advanced Analog | LEDs | Azure

Via this interface we take advanced readings of voltage, temperature and light levels experienced on sensors on the remote XinoRF. The XinoRF gives out a RawADC number that is between 0 and 1023, as seen on the basics tab, this is then converted into more useful information using the formulae given below.

Read

Volts: 2.32V
 Correction Factor: 1
 Temperature: 21.75°C
 Percentage: 46.3%

This is the raw ADC value converted to Volts
 $\text{Volts} = (\text{RawADC} / 1023 * 5.0V) * \text{Correction Factor}$

Temperature is calculated using the following formula
 $\text{RTemp} = (1023.0 / \text{RawADC} - 1) * 10000$
 $\text{Kelvin} = \text{RTemp} * \text{BVAL} / (\text{BVAL} + \text{RTemp} * (\log(\text{Rtherm} / \text{RNOM})))$
 $\text{Temperature} = \text{Kelvin} - 273.15$

The light reading from the LDR is presented as a percentage
 $\text{Percentage} = \text{RawADC} / 1023 * 100$

Send a LLAP command to address XX with DATA HELLO Send

History Send Previous

Received LLAP to TM with DATA: TMPA19.91
 Received LLAP to AW with DATA: PIRTRIG
 Received LLAP to BZ with DATA: PIRTRIG
 Received LLAP to EH with DATA: TMPA37.26
 Received LLAP to CB with DATA: PIRTRIG
 Received LLAP to AT with DATA: PIRTRIG
 Received LLAP to AT with DATA: PIRTRIG
 Received LLAP to BG with DATA: PIRTRIG
 Sending LLAP to XX with DATA: AD0READ
 Received LLAP to XX with DATA: A00+474

The application calculates a voltage, temperature and percentage light reading from the same A0 analogue input pin using the documented formula.

By pressing the thermistor between your finger and thumb you will increase the temperature. Press the READ button several times while doing this to see the temperature increase, and then without your finger and thumb to see the temperature decline.

If you'd like to do the same with for light levels you can carefully swap the thermistor for the LDR and measure light levels.

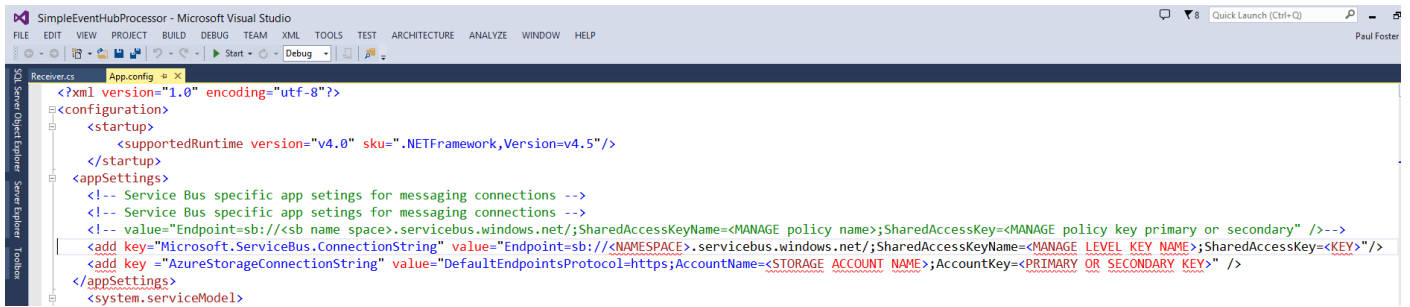
Now we have a temperature/light sensor gathering data we need to publish this data to Azure so we can do further processing on it.

2. Publishing sensor data to Azure Event Hub

Requirements:

1. Azure subscription. A free trial or your own Azure account is required.
2. Event Hub. Create an event hub as documented in the Event Hub HOL. Note the namespace, event hub name, SEND policy name, the SEND policy key and then the MANAGE endpoint URI.
3. Add a custom consumer group to the event hub called 'custom'.
4. An Azure Storage Account. Create one in the Azure Management Portal or use an existing one. Make sure you place it in the same region as your Event Hub.
5. Download the SimpleEventHubProcessor project from the OneDrive share to your Windows PC.

First your event hub details need to be provided to the SimpleEventHubProcessor. With the project open in Visual Studio 2013 (or higher) open the app.config file.



```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5"/>
  </startup>
  <appSettings>
    <!-- Service Bus specific app settings for messaging connections -->
    <!-- Service Bus specific app settings for messaging connections -->
    <add key="Microsoft.ServiceBus.ConnectionString" value="Endpoint=sb://<NAMESPACE>.servicebus.windows.net/;SharedAccessKeyName=<MANAGE policy name>;SharedAccessKey=<MANAGE policy key primary or secondary>" />
    <add key="AzureStorageConnectionString" value="DefaultEndpointsProtocol=https;AccountName=<STORAGE ACCOUNT NAME>;AccountKey=<PRIMARY OR SECONDARY KEY>" />
  </appSettings>
  <system.serviceModel>

```

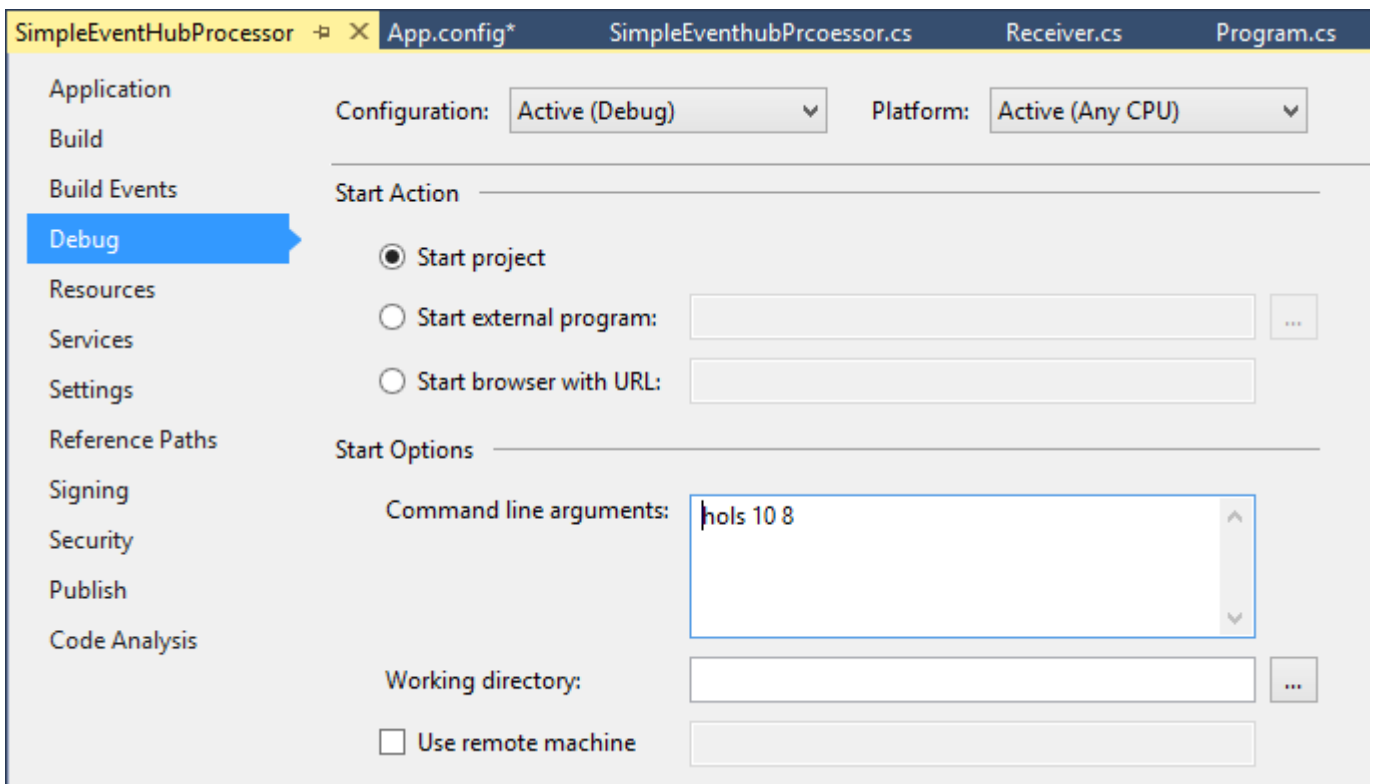
The EndPoint URI has the following format and can be accessed from the event hub dashboard by viewing the connection string.

Endpoint=sb://<sbnamespace>.servicebus.windows.net/;SharedAccessKeyName=MANAGE;SharedAccessKey=<KEY>

The Azure Storage Connection String has the following format, use the storage account details created as part of the lab requirements (4). You can look the details up in the Azure management portal under Storage Accounts -> Manage Access Keys.

<add key="AzureStorageConnectionString" value="DefaultEndpointsProtocol=https;AccountName=<STORAGE ACCOUNT NAME>;AccountKey=<PRIMARY OR SECONDARY KEY>" />

Next name your event hub in the project properties -> Debug command line arguments by replacing hols, keeping 10 8 at the end as below:



Executing this app with F5 will show a console app which connects to and listens for your event hub messages.

Now let's publish some messages!

On the RPI switch to the Azure tab in the Azure WIK app.

Introduction | Basic | Advanced Analog | LEDs | **Azure**

This tab enables you to extend your IoT experiments to the cloud using Microsoft Azure.

You need a Microsoft Azure subscription, you can try it for free for a month, just click to [sign up to the Azure free trial](#)

Once you have an Azure subscription you can push your LLAP messages to the cloud for processing and generate actions that can be returned to your Xino RF.

Send your LLAP messages to an Azure Event Hub.

Complete the event hub details opposite and press Publish to Event Hub.

Your XinoRF LLAP output will be sent to the hub ready for processing.

On Advanced Analog tab press Read to cause LLAP messages, for example.

Service bus name space

Event Hub

Policy name

Key

Data message sent to event hub:

```
subject as string
time as datetime
from as string
payload as string
pin as string
value as string
volt as float
tmp as float
ldr as float
```

Send a LLAP command to address XX with DATA

History

Received LLAP to FH with DATA: TMPA19.14
Received LLAP to TM with DATA: TMPA20.14
Received LLAP to AW with DATA: PIRTRIG
Received LLAP to BG with DATA: PIRTRIG
Received LLAP to BH with DATA: PIRTRIG
Received LLAP to BZ with DATA: PIRTRIG
Received LLAP to AA with DATA: TMPA41.47
Received LLAP to AW with DATA: PIRTRIG
Received LLAP to BG with DATA: PIRTRIG
Received LLAP to TM with DATA: TMPA20.09

Complete the fields on this tab as specified:

Service bus name space, just the unique name not the full Azure URI.

Event Hub, the case sensitive name of your event hub.

Policy name, the case sensitive name of the SEND policy. Typically SEND.

Key, the primary or secondary key for the SEND policy.

Once completed press the Publish to Event Hub button.

The application now uses the AMQP.NET Lite component to publish a data set from the XinoRF state to the Azure Event Hub.

The message contains:

Subject	string – name of message type. LLAP
Time	datetime – creation of data on RPI, so when it was received from the XinoRF
From	string – the device name that sent it
Payload	string – the full LLAP payload message. Format is Pin id, value all concatenated e.g. A00+545
Pin	string - the pin id extracted from the LLAP payload
Value	string – the value extracted from the LLAP payload
Volt	float – calculated voltage
Tmp	float- calculated temperature
Ldr	float – calculated light level percentage.

The volt, tmp and ldr fields are all created from the value of the A00 pin on the XinoRF. The meaningful value of these three will depend on the sensor you have connected to this pin.

By pressing the READ button on the Advanced Analog tab you will cause data READ request to be sent to the XinoRF, which will return with the A00 reading. Once received, the Azure WIK app uses AMQP to send the message to the specified Event Hub.

The Simple Event Processor console window should now show the received messages which have been sent in a binary JSON form to the event hub and received by the simple event processor, which has unpacked the binary data to show the raw JSON.

3. Processing the sensor data with Azure Stream Analytics

Now we have data being sent from the wireless sensor (XinoRF) via the IoT Gateway (RPI2) to the event ingestor (Azure Event Hub), it can now be processed to create some output.

A simple way to do this is to use the Azure Stream Analytic service (ASA). Note, ASA currently uses the default consumer group of the event hub. This is way the SimpleEventProcessor application has to use a secondary consumer group called custom.

To process the data start by creating a new ASA task in the Azure management portal as documented in the ASA hands on lab.

The Input is the Event Hub being named in the Azure WIK step 2. The data format is JSON.

The output should be blob storage for this first job. This allows data to be captured and used repeatedly for ASA query development.

The query. Initially, the query should be a simple

```
Select * from <input alias>
```

Start this ASA job and then send data to it by pressing the READ button on the Advanced Analog tab on the Azure WIK application. The SimpleEventProcessor, if still running, will show the data being received.

After several messages have been sent, open the named output blob storage location in your favourite Azure storage explorer tool (I use Azure Management Studio) and open the JSON file created. Saving this locally to your PC harddrive allows you to use it in future ASA query development.

Once you have confirmed data is flowing from your sensor to ASA, you can enhance your ASA query. Either by modifying the existing select * query or by creating a new ASA job. Leaving the existing select * query will ensure your test data accumulates in blob storage.

To enable ASA message filtering you can use the Create Table statement in the query:



query

```

1 create table hols (
2   subject nvarchar(max),
3   time DateTime,
4   [from] nvarchar(max),
5   payload nvarchar(max),
6   pin nvarchar(max),
7   value nvarchar(max),
8   volt float,
9   tmp float,
10  ldr float)
11
12 select [from], Max(tmp), System.Timestamp as WindowEnd from hols TIMESTAMP by time
13 group by [from], TumblingWindow(minute,1);

```

Missing some language constructs? [Let us know!](#)

Test

Rerun

Here, create table doesn't create a table but rather defines the data schema of the expected message. Messages that do not fit this schema are rejected by ASA.

The table alias is the same as the input alias which is referred to in the select statement from clause.

This select statement groups the data by the device that sent it, and calculates the maximum temperature reached over a 1 minute time window. The Window aggregates data received within one minute, then moves to a new minute etc. The WindowEnd file records the UTC time of the minute aggregated. The time series for the query is set to the time field of the message which is the creation data of the message on the IoT Gateway (RPi2).

More complex multi-step queries can be created to do further processing.