Connecting a FLIR thermal camera to Microsoft Azure using ICONICS IoTWorX

Contents

Copyright and Confidentiality

By accessing and using the installation instructions (the “instructions”) you acknowledge and agree, on your behalf and on behalf of the person, entity or other organization on whose behalf you are accessing the instructions, that neither Microsoft, ICONICS, FLIR Systems, nor any of its service providers, including, without limitation, any system integrator or independent software vendor: (1) makes any representations or warranties of any kind, either express, implied, statuary or otherwise with respect to the instructions, including the accuracy, completeness or usefulness thereof; and (2) shall be liable for damages of any kind, under any legal theory, arising out of or in connection with your election to follow or use, or inability to follow or use, the instructions, including any direct, indirect, incidental, special, punitive or consequential damages, or for loss of use, loss of profits, loss of data, loss of business, or loss of privacy or security, even if foreseeable, arising out of or in connection with your election to follow or use, or inability to follow or use, the instructions. You further acknowledge and agree that your use of the instructions, whether directly or indirectly, is at your own risk and that you expressly assume all risk in connection with your use of the instructions. If you do not agree to the foregoing, you may not access or use the instructions.

Authors

- Spyros Sakellariadis, Applied Innovation/Enterprise Commercial Business, Microsoft Corporation
- Zhi Wei Li, Director of Innovation & Incubation Solutions, ICONICS
- Patrik Simion, Global Solutions Engineer, FLIR Systems AB

1 Introduction

Thermal cameras have many uses, in as diverse industries as manufacturing, healthcare, and energy. There is plenty of literature and guidance on how and when to use the various models that are available. The majority of these cameras are deployed in such a way as to display the data on a console adjacent to, or nearby, the camera that is watched by a human to observe conditions or anomalies. Take for example the image on the home page of FLIR Systems, a manufacturer of thermal cameras:

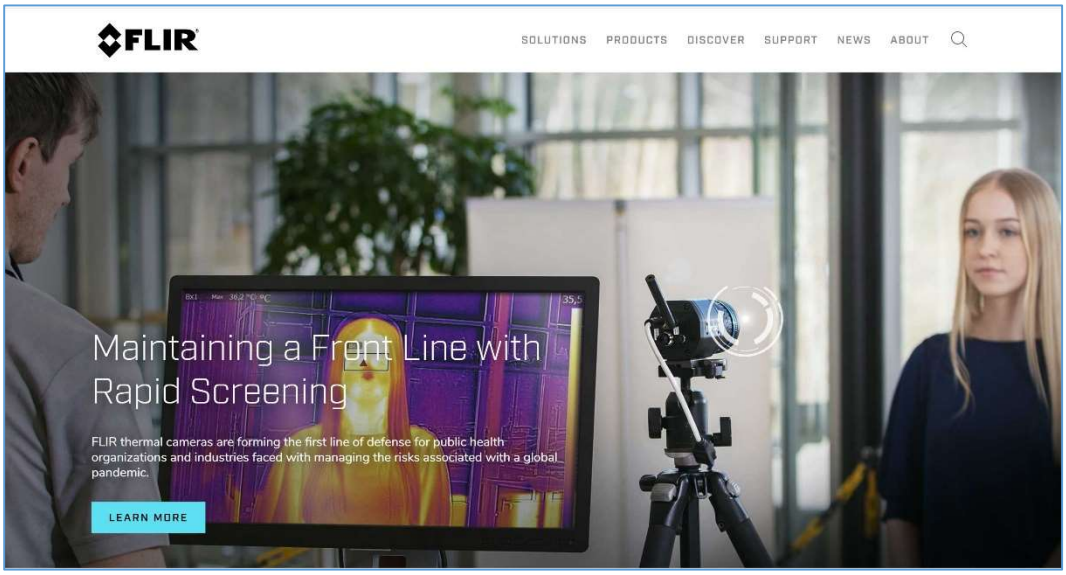

Figure 1 FLIR Systems home page

These scenarios require local personnel to observe the output and take an action. Another group of scenarios involves monitoring the output of these cameras remotely, and specifically monitoring the digital output rather than visual heatmaps. And with the rapid adoption of cloud computing, these scenarios can be made more valuable by integration with powerful cloud-based applications and artificial intelligence/machine learning. However, there is not a lot of literature or guidance on how to connect thermal cameras to cloud-based systems. This document seeks to fill this gap, or at least be a start in that.

In this context, this documentation describes one example of how to connect a thermal camera to a cloud-based monitoring system, which might be a standalone application or part of a larger Smart Building solution. The example here involves monitoring temperature readings and alerts from a thermal camera from installed in a remote location. The addresses of the FLIR registers, the Azure and ICONICS configurations are those that were correct at the time of writing of this document and may not be the same for any future versions of those products. It also assumes that the relevant features and licenses have been obtained.

2 Infrastructure

2.1 Hardware

In the setup described in this paper, the camera being monitored is a FLIR A320 Tempscreen camera. The camera supports external communication over Ethernet, and in the setup described here the A320 communicates with a Dell Edge Gateway 5000 and with a Microsoft Surface Laptop over a local Cat6 network. The physical configuration is shown in Figure 2:

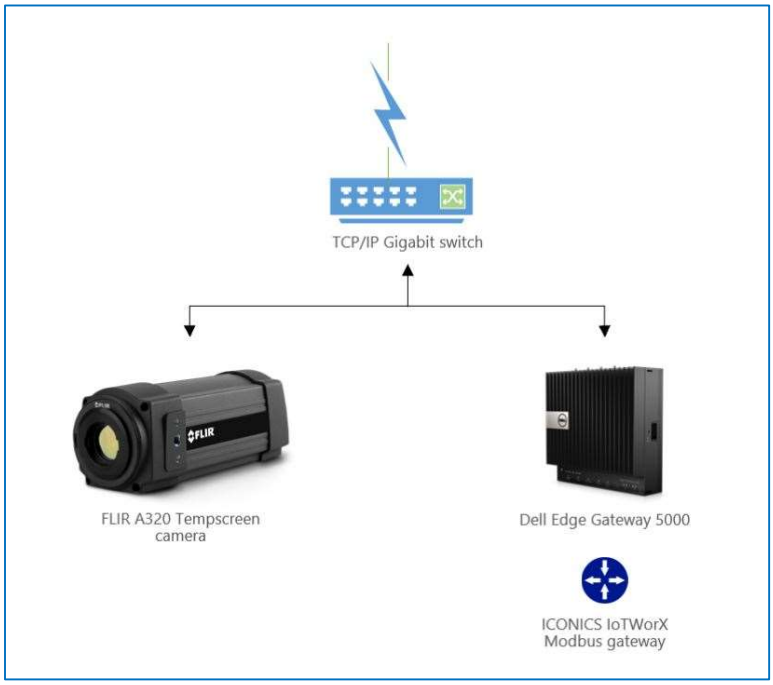

Figure 2 Production hardware configuration

Both the camera and the Dell computer are connected to a local network switch, and both systems are configured to get dynamic IP addresses from a local DHCP server. The switch itself is connected through a firewall to the Internet, with only outbound ports open.

While this is the hardware configuration that we are testing, the actual hardware configuration that we have in place for doing the testing is shown in Figure 3:

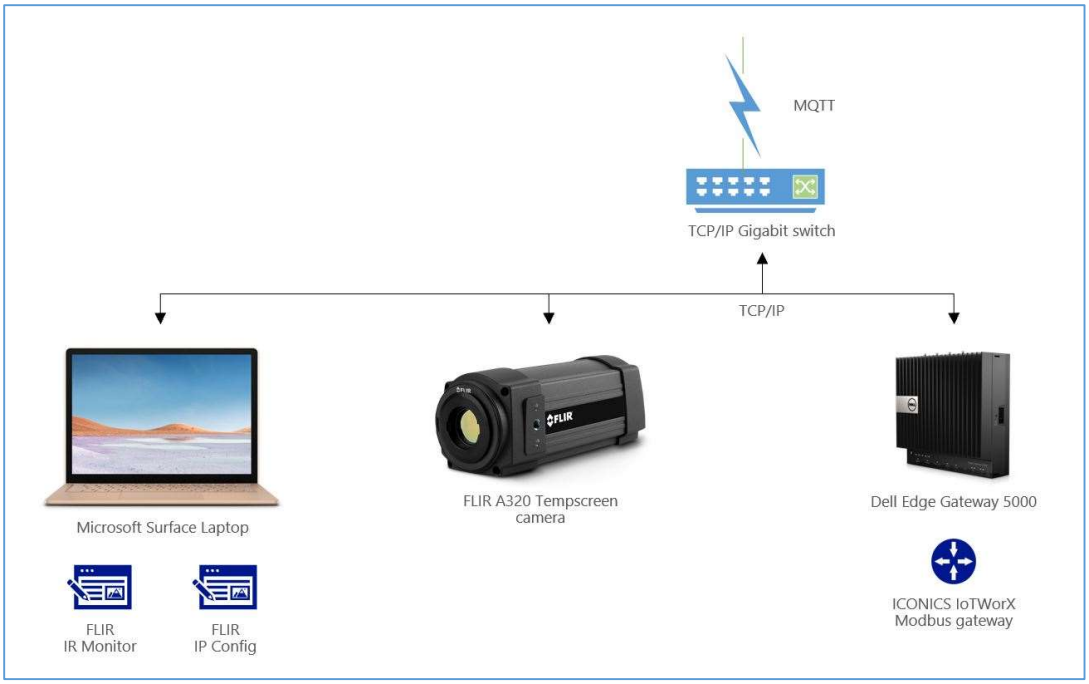

Figure 3 Testing hardware configuration

The laptop has the FLIR IR Monitor application installed, for the purpose of configuring the scanning areas, alarm thresholds, and checking that the values received in Azure are correct. It also has the FLIR IP Config utility (not to be confused with the Windows IPCONFIG tool), for the purpose of configuring the network setup of the camera. Both IR Config and IR Monitor are available at this public link: https://flir.custhelp.com/app/answers/detail/a_id/1137

2.2 On-premises Software

In this configuration we are using the ICONICS IoTWorX Modbus gateway to read from the A320. Specifically, we will configure IoTWorX to perform the following functions:

1. Connecting to the A320
2. Requesting values of certain objects on the A320 every 30 seconds
3. Reformatting the data into a prescribed format
4. Transmitting that data to Azure IoT Hub

2.3 Cloud services

After the data arrives in Azure IoT Hub, an Azure Stream Analytics job reads the incoming data stream and writes it into a blob, which we can review to see the data received. The data flow is shown in Figure 4:

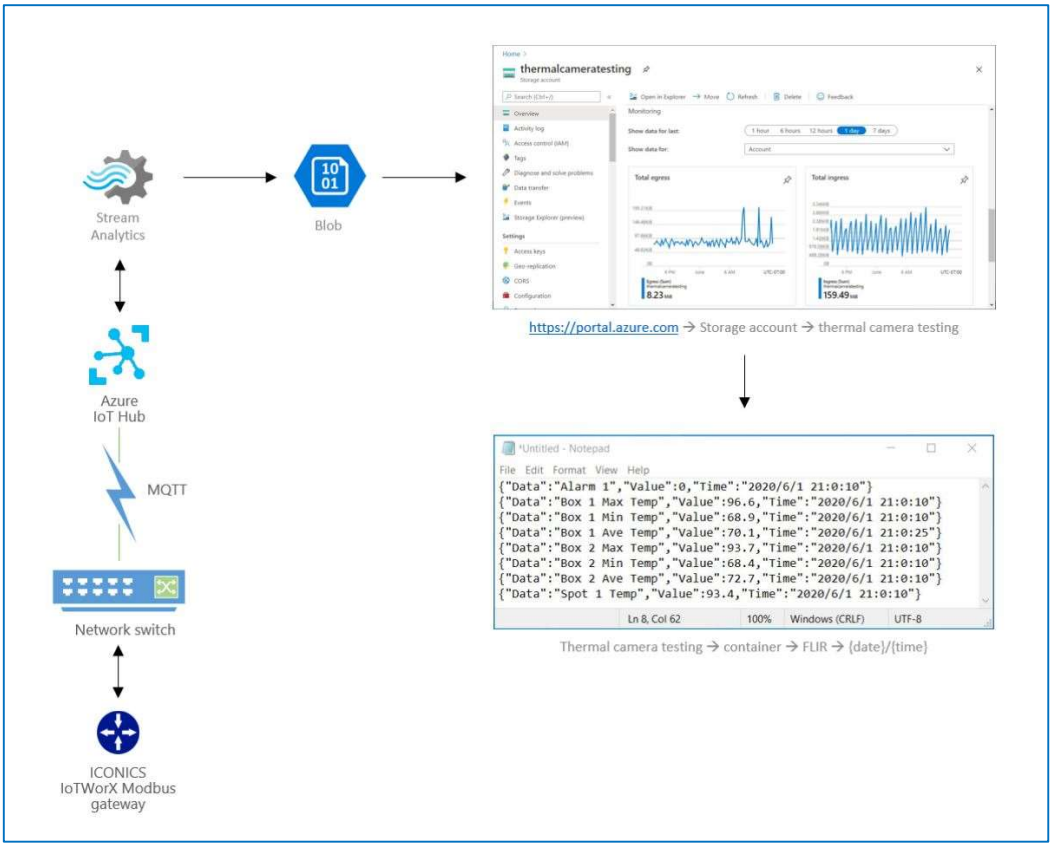

Figure 4 Software components

The purpose of this paper is to show only how to configure IoTWorX to read the data from the camera. In a production scenario you would likely use an application in Azure that takes an action based upon the data, for example a Stream Analytics job that issues alerts based upon various criteria.

The following sections contain a description of how to configure the IoTWorX gateway and the Azure components to monitor the A320.

3 Configuring Azure IoT Hub

Full guidance on using Azure is beyond the scope of this document, which assumes you have a basic knowledge of navigating and using the Azure portal to create and configure services. It also assumes you have the proper licenses and access to Azure and ICONICS IoTWorX. Follow the instructions below to prepare the environment for receiving data from the camera.

3.1 Create a new IoT device in Azure IoT Hub to receive the data

1. Login to the Azure portal, and navigate to the Home Page:

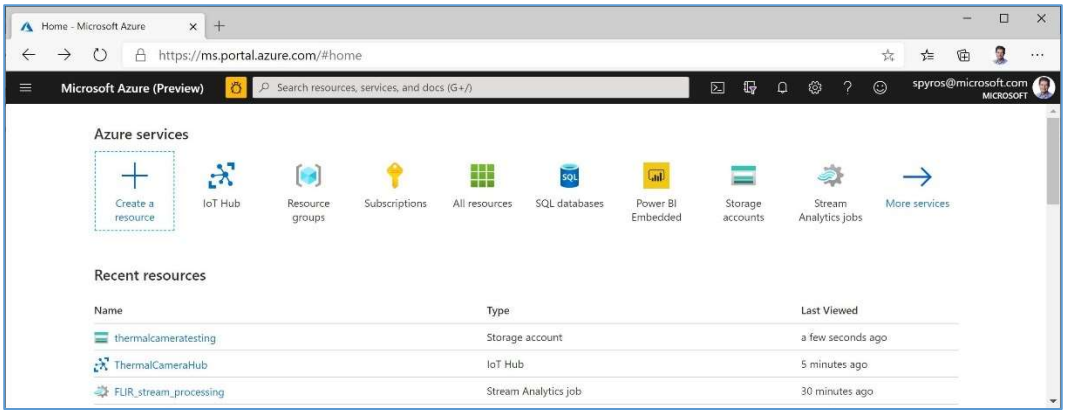

Figure 5 Azure Portal Home Page

2. Click on the Plus icon to create a new resource
3. Select Internet of Things from the list of options in the column on the left
4. Click on IoT Hub on the right to create a new IoT Hub that will receive published data. In the setup used in this document, we used the name 'ThermalCameraHub' (Follow this guide to create a new IoT Hub: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-create-through-portal)
5. Open the configuration blade of the IoT Hub.
6. Click on IoT devices from the list of options on the left.
7. In the devices blade, select 'Add' at the top to create a new device.
8. Give the device an ID. In the setup used in this document, we used the ID 'FLIR_A320_lab'
9. Keep all other options as default.
10. Click on Save to create the new device.
11. Once the gateway device is created, re-open the IoT devices list.
12. Select FLIR_A320_lab.
13. In the FLIR_A320_lab page, look for 'Primary Connection String'.
14. To the right of that field is a button to copy the connection string - click on the copy button to copy the connection string to the clipboard. A connection string will look something like the following. Note both the name of the IoT Hub (ThermalCameraHub) and the device ID (FLIR_A320_Lab) are included in the string:

    HostName=ThermalCameraHub.azure-devices.net;DeviceId=FLIR_A320_lab;SharedAccessKey=******=

15. Store the connection string somewhere temporarily until it is needed, which will be in Section 6.3 below.

3.2 Create a new storage account to hold the data

1. Login to the Azure portal and navigate to the Home Page.
2. Click on the Plus icon to create a new resource
3. Select Get Started from the list of options in the column on the left
4. Click on Storage Account from the list of options on the right.
5. Create a new storage account that will receive published data. In the setup used in this document, we used the name 'thermalcameratesting'

4 Configuring the A320

There are several steps needed to configure a virgin A320 to run in the intended configuration. These are as follows.

4.1 Specify that the A320 should get its IP address from a DHCP server

Out of the box, the A320 we received came with an Ethernet cable and instructions to connect it directly to the Ethernet port on a laptop. The camera will either have a static IP address (e.g. 192.168.1.120) or be set to get an address from a DHCP server.

To see if you need to change the settings in the camera, consult the documentation that came with the camera for the default configuration. If it is set for DHCP, you need to do nothing. If it is set for a static IP, you need to change it.

The easiest way to change the networking settings of the camera is to launch the FLIR IP Config utility, downloaded and installed per the note in Section 2.1 above. When you launch the utility, it scans the local network for FLIR Cameras on the network, and when found you can configure the IP settings – setting it to DHCP or Static by manually entering the IP. The screenshot below shows IP Config having discovered the camera in the current deployment:

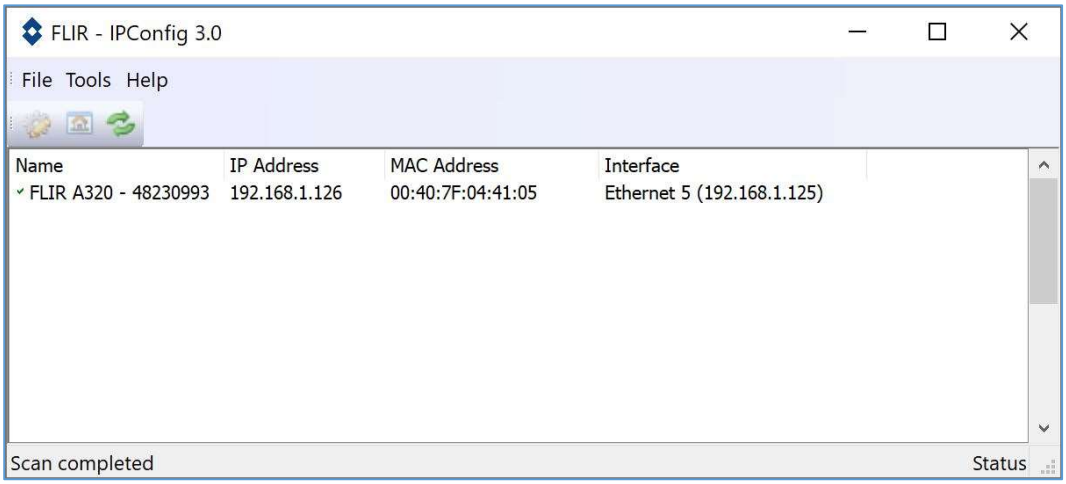

Figure 6 FLIR IP Config showing results of scanning 192.168.1.x network

In our case, the camera was found at 192.168.1.126. Right click on the camera listed and select Modify to bring up a window allowing you to see the current settings, and set it to DCHP:

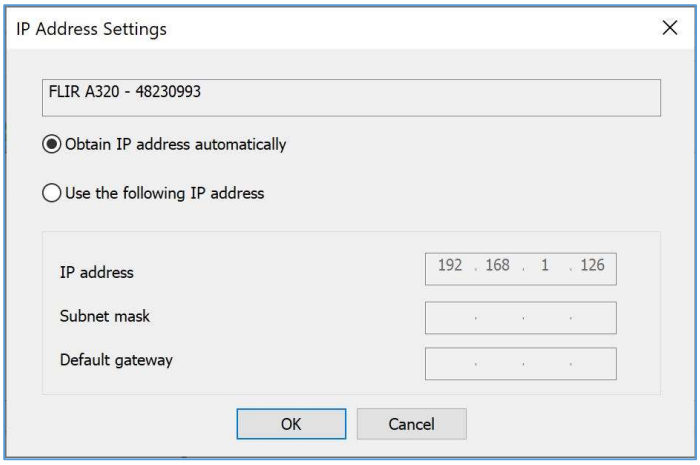

Figure 7 IP Address Settings

If, for some reason, FLIR IP Config fails to find the camera on the network, you can modify the settings using Telnet from a computer on the same subnet. Once you have connected using Telnet, follow the steps below to change the camera settings. Specifically, you are writing to a register in the camera, analogous to making a change in the Windows registry.

1. Connect from your laptop to the camera (assuming it is set at the factory to 192.168.1.20):

   C:\>Telnet 192.168.1.20

   Assuming the connection is successful, you should see:

   Welcome to the Windows CE Telnet Service on <factory set camera name>

   FLIR Command Line Interpreter
   Version 0.2.03 running on WinCE 6.0

2. Check the camera registry for the DHCP setting. The registry settings are detailed in the FLIR document Basic ICD FLIR IR Camera – PC, section 4.2.1 on Network resources. You read the registry settings with the command 'rls' and set them with the command 'rset'. Cutting to the chase, type the following to check the current setting:

   \>rls .services.net.interface.LAN90001.DHCP

   You want it to return

   DHCP true

   Assuming that it returned DHCP false, you need to change it to true.

3. To change the DHCP setting, type:

   \>rset .services.net.interface.LAN90001.DHCP true

This should return you to the \> command prompt. To check that it worked, rerun the rls command above and verify that it returned 'true'.

4. You can check the new network information with the following commands:

```
\>rls .services.net.interface.LAN90001.address
\>rls .services.net.interface.LAN90001.netmask
\>rls .services.net.interface.LAN90001.gateway
```

4.2 Load the Modbus software stack

The FLIR A320 supports the Modbus protocol, but is the networking stack is not necessarily loaded by default. You can use Telnet to check and to load it if necessary.

1. After loading Telnet and connecting to the camera using the IP address identified above, use the following command to determine whether the Modbus stack is loaded:

```
\>ps
```

This should return a list similar to the following:

```
Process NK.EXE         (68 threads), id 0x00400002, loaded at 0x88190000
Process udevice.exe     ( 1 threads), id 0x01560002, loaded at 0x00010000
Process udevice.exe     ( 3 threads), id 0x01640002, loaded at 0x00010000
Process udevice.exe     ( 1 threads), id 0x00CC0006, loaded at 0x00010000
Process udevice.exe     ( 1 threads), id 0x03DE0002, loaded at 0x00010000
Process servicesd.exe   (30 threads), id 0x03F60006, loaded at 0x00010000
Process udevice.exe     ( 1 threads), id 0x05E00002, loaded at 0x00010000
Process cmd.exe         ( 1 threads), id 0x05FE0002, loaded at 0x00010000
Process appcore.exe     (36 threads), id 0x046A0006, loaded at 0x00010000
Process AppServices.exe (13 threads), id 0x04750062, loaded at 0x00010000
Process Resmon.exe      (13 threads), id 0x049B008E, loaded at 0x00010000
Process MediaServer.exe (19 threads), id 0x06100002, loaded at 0x00010000
Process Gui.exe         (11 threads), id 0x06360002, loaded at 0x00010000
Process dighandler.exe  ( 7 threads), id 0x06530002, loaded at 0x00010000
Process Watchdog.exe    ( 7 threads), id 0x06640002, loaded at 0x00010000
Process nexuscontrol.exe ( 5 threads), id 0x067D0002, loaded at 0x00010000
Process CMD.EXE         ( 1 threads), id 0x0994001E, loaded at 0x00010000
Process eips.exe        ( 8 threads), id 0x088D0022, loaded at 0x00010000
Process CMD.EXE         ( 1 threads), id 0x0925324E, loaded at 0x00010000
Process ps.exe          ( 1 threads), id 0x08FA178E, loaded at 0x00010000
```

The **eips.exe** process is the Modbus stack. If it is included in the output from the ps command, Modbus is loaded.

2. If eips.exe is not listed, type

```
start eips
```

Repeat the ps command above to make sure that eips.exe has loaded. One thing to note is that you may need to reload eips.exe if the camera is powered cycled.

4.3 Define the areas and alarms to be monitored

As described in the Introduction earlier, the intent of this document is not to provide guidance on how to use the camera. It is only to provide guidance on how to get data from the camera to the cloud. However, to do this we do need to identify which registers in the camera should be read. For the purposes of this exercise, we will read eight registers:

Register name
Alarm 1
Spot 1 Temperature
Box 1 Minimum Temperature
Box 1 Maximum Temperature
Box 1 Average Temperature
Box 2 Minimum Temperature
Box 2 Maximum Temperature
Box 2 Average Temperature

Table 1 A320 registers of interest

The default configuration of the A320 includes Spot 1 coordinates, and Box 1 coordinates and size. For this task we will use the FLIR IR Monitor tool, downloaded and installed per the note in Section 2.1 above.

Once you have launched the IR Monitor, and selected the camera (Camera → Connect), select the Analysis tab on the right:

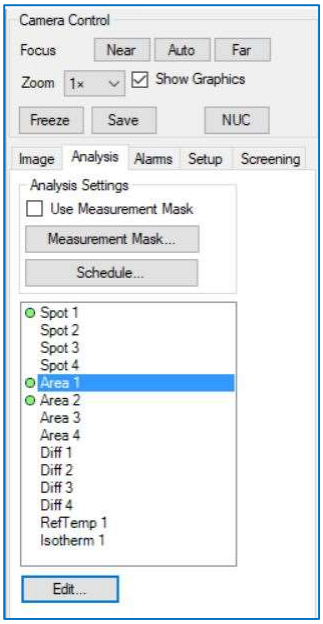

Figure 8 IR Monitor Analysis Tab

In the Analysis tab, click Edit and set each of the three areas – Spot 1, Area 1, and Area 2 – as active, and edit the parameters. For this document, we chose:

Box 1: x=60, y = 80, Width = 106, Height = 80, Show Max/Min = Both, and Show Area checked
Box 1: x=170, y = 80, Width = 106, Height = 80, Show Max/Min = Both, and Show Area checked
Spot 1: x=150, y = 120, and Show Spotmeter checked

4.4 Identify the registers to be read

Finally, we need to know the register addresses for the eight variables we want to monitor. The addresses for all the registers in the camera can be found in the FLIR Systems Ethernet/IP and Modbus TCP Object Models, Mapping 3 – Read Assembly Values (pp 69-71). Specifically the register address for the objects above is as follows:

Register name	Register address
Alarm 1	402003
Spot 1 Temperature	402019 – 402020
Box 1 Minimum Temperature	402021 – 402022
Box 1 Maximum Temperature	402023 – 402024
Box 1 Average Temperature	402025 – 402026
Box 2 Minimum Temperature	402033 – 402034
Box 2 Maximum Temperature	402035 – 402036
Box 2 Average Temperature	402037 – 402038

Table 2 Register addresses

The documentation states specifically that “the Temperature values are mapped as a floating point value with the most significant word stored in the first register and the least significant

word store in the second register." It is important to note also that we must swap of words inside dwords in the device settings although the FLIR spec does not explicitly state this on page 69. This will become clearer in Section 5.3 below.

5 Configuring IoTWorX to access the A320

Installing ICONICS IoTWorX to read from devices is covered in detail in a document published on the ICONICS website here: https://iconics.com/Documents/WhitePapers/Using-IoTWorX-as-a-Gateway (below referred to as *Using IoTWorX*). That document specifically talks about BACnet, but the steps are essentially the same for Modbus. Read through the instructions in that document for using ICONICS Workbench, then proceed as below.

5.1 Specify a channel to communicate with the camera

This section addresses how to set up IoTWorX based upon the camera settings discussed in Sections 4.1 and 4.2, above. Follow these steps:

1. Open Workbench Desktop on the GENESIS64 machine/gateway that will be publishing data.
2. In the Project Explorer, expand your project, then Data Connectivity, then Modbus.
3. Right click on Channels.
4. Click on Add Channel
5. In the [New Channel] tab
    - Give the Channel a name. In the setup we used for this document, we chose "192.168.1.126 TCP 502", where 192.168.1.126 is the IP address of the camera.
6. In the Channel Type section
    - Check the box "Enable"
    - For Channel Type, pick Modbus TCP/IP Ethernet.
7. In the Channel TCP/IP Settings
    - For IP addr. Or hostname, enter the IP address of the camera. In our setup, we put 192.168.1.126
    - Leave the TCP port as 502, the TCP sockets as 1, and Re-open sockets as checked.
8. In the Communication Trends section
    - Enter 10 milliseconds for the Send/Recv delay
9. Click Apply

When complete, the ICONICS Workbench should look like this

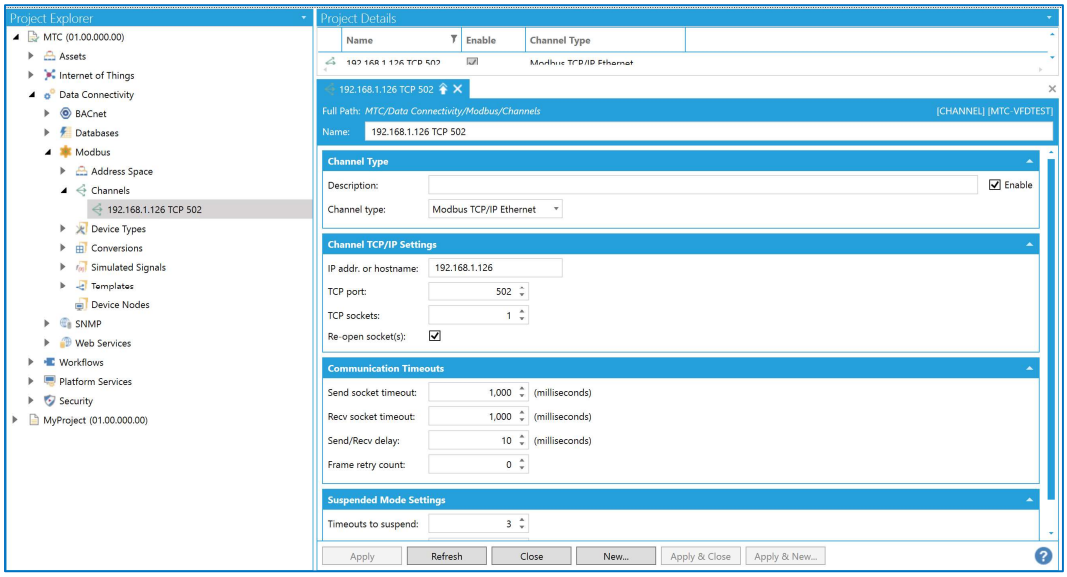

Figure 9 Modbus Channels

5.2 Specify the camera device type

This section addresses how to set up IoTWorX based upon the camera settings discussed in Section 4.3 above. Follow these steps:

1. In the Project Explorer, expand your project, then Data Connectivity, then Modbus, then Device Types
2. Right click on Default Device Type and select Edit.
3. In the Swap Options section, leave all the fields the same, except
    - Enable “Swap words inside dwords”. This is critical, and due to the point made near the end of Section 4.3, above. Not doing this will lead to meaningless data in Azure.
4. Click Apply

When finished, the ICONICS Workbench should look like this:

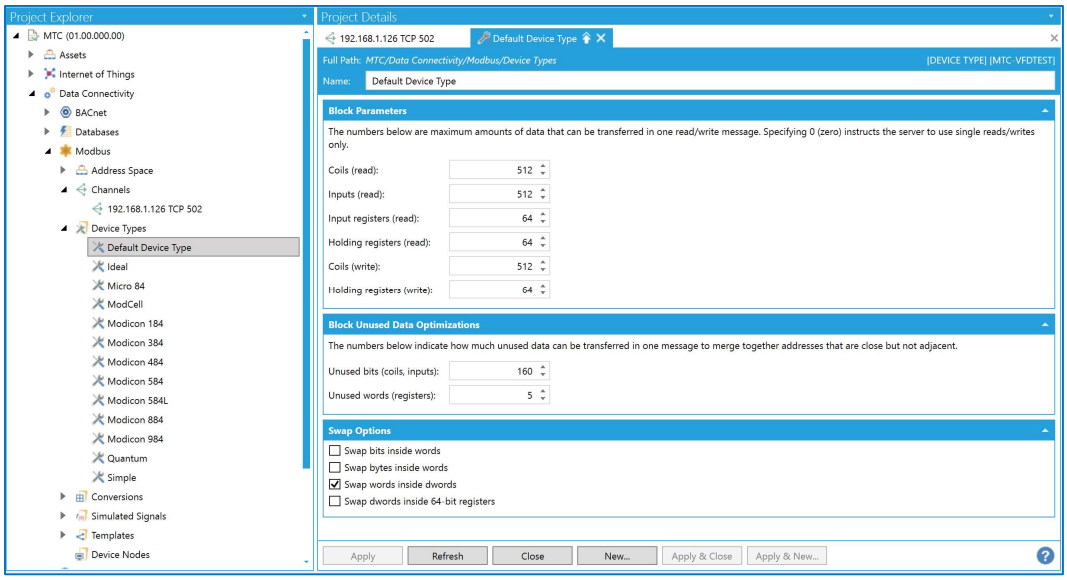

Figure 10 Modbus Default Device Type

5.3 Add the camera as a device

Follow these steps:

1. In the Project Explorer, expand your project, then Data Connectivity, then Modbus.
2. Right click on Address Space.
3. Click on Add Device.
4. In the [NewDevice] tab
    - Add a name in the [Name] field. We used “FLIR A320”.
5. In the Device Properties section
    - Check the Enable checkbox
    - In the Channel field, select the Channel created in 5.1 above
    - In the Device Type field, select “Default Device Type”
    - For the Unit Id, choose “1”
    - For Base Scan Rate, put 1,000 milliseconds.
6. In the Other Settings section, ensure that only the following boxes are checked:
    - Use zero based bit access addressing
    - Disable tags at invalid addresses
    - Use single coil write (Modbus Fnc 05)
    - Use single register write (Modbus Fnc 06)
7. Click Apply

When finished, the ICONICS Workbench should look like this:

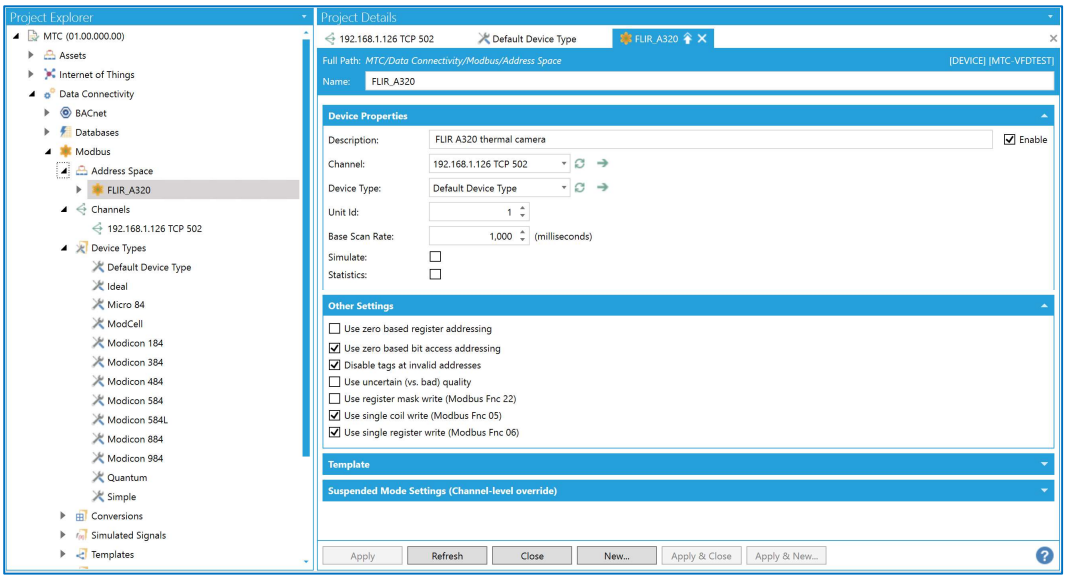

Figure 11 Modbus Device

5.4 Add the camera registers

Follow these steps to enter the first object:

1. In the Project Explorer, expand your project, then Data Connectivity, then Modbus, then Address Space
2. Right click on the device, here the FLIR A320.
3. Click on Add Data Item.
4. In the [NewDataItem] tab
    - Enter “Alarm 1” for the Name
5. In the Basic Properties section
    - Enter “Alarm 1” for the Description
    - Check the Enable checkbox
    - In the Location Type, select Holding Register
    - In the Starting Address, enter 2003, the last 4 digits for Alarm 1 in Table 2 Register addresses
    - In the Modbus Datatype, select Int(16 bits)
6. Click Apply & Close

To enter the next seven objects follow steps 1-6 again, but

- Use the Register Name in Table 2 Register addresses for the Name and Description
- In the Starting Address, use the last 4 digits of the Register Address.
- In the Modbus Datatype, select **Float (32 bits)**

When finished, the ICONICS Workbench should look like this:

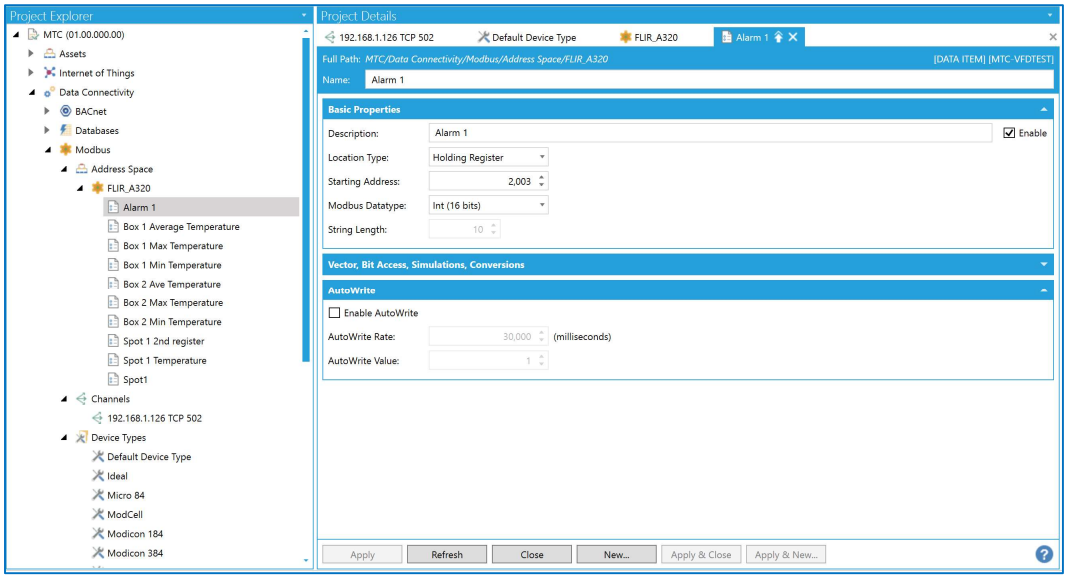

Figure 12 FLIR A320 with registers

6 Configuring IoTWorX to access Azure

Configuring IoTWorX to format and send data to Microsoft Azure is the same regardless of what protocols are used to communicate to downstream devices. This is covered in detail in https://iconics.com/Documents/WhitePapers/Using-IoTWorX-as-a-Gateway. The process consists of four steps

1. Create one or more lists of data points you want to send to the cloud (Publish Lists)
2. Specify one or more ways the messages should be formatted (Custom Encoders)
3. Specify one or more URLS of cloud endpoints (Publisher Connections)

Read through the instructions in the document for using ICONICS Workbench, then proceed as below.

6.1 Create a publish list

Follow these steps to create the publish list:

1. In the Project Explorer, expand your project, then Internet of Things
2. Right click on Publish Lists and click on Add Publish List.
3. Click on Add Data Item.
4. In the [NewPublishList] tab
    - Enter “FLIR_PubList” for the Name
5. In the General Tab, Rate Settings section
    - Check the Only send refresh messages checkbox
    - Select 15 seconds for the Refresh rate
    - Select 5 seconds for the Refresh Hold Off Time
    - Select 50 for the Dynamic Tags Published
    - Select Collection Group 1 for the Default Collection Group
6. In the Published Points tab
    - In the blue section header bar click on “Click here to add multiple tags”
    - In the Data Browser windows
        i. Select My Computer, then Data Connectivity, then Modbus, then FLIR_A320
        ii. Hold down the Control key and select all the data points identified in Section 5.4, above
        iii. Click OK
    - Click Apply

When finished, the ICONICS Workbench should look like this:

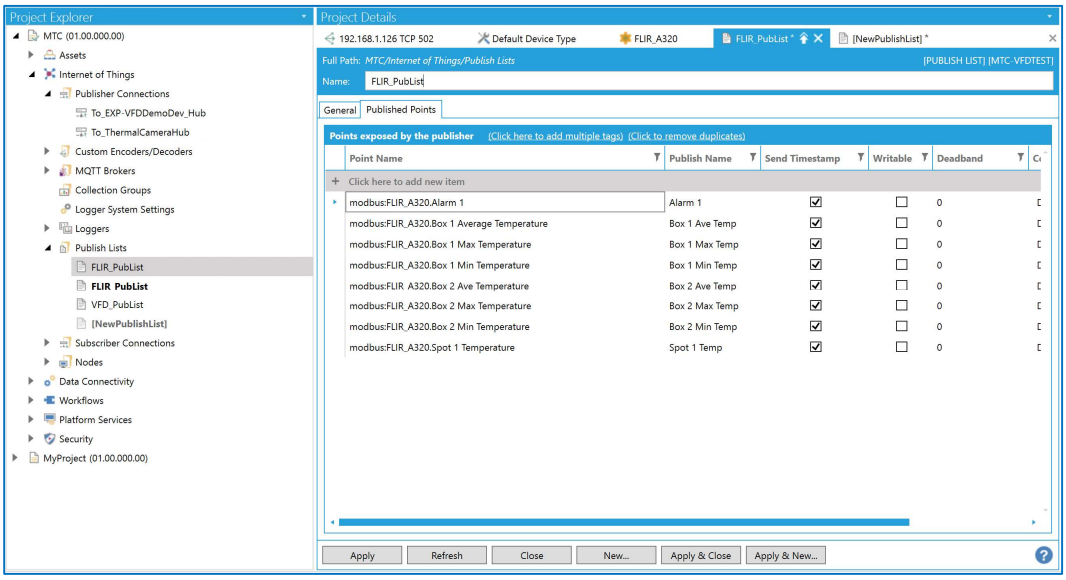

Figure 13 FLIR_PubList Published Points

It is useful to enter a Publish Name manually in the Publish Name column, as that will make it easier to parse the data in Azure.

6.2 Create a custom encoder

Follow these steps to create the encoder:

1. In the Project Explorer, expand your project, then Internet of Things
2. Right click on Custom Encoders/Decoders and click on Add Encoder/Decoder.
3. In the [NewEncoder/Decoder] tab
    - Enter “FLIR_Encoder” for the Name
4. In the General Settings Tab
    - For the Plugin, select CustomJson
    - For the Message Type, select “One value for each message”
    - For the Value format, enter

```
{
 "gwy":"%DEVICENAME%",
  "name": "%PUBLISHNAME%",
  "value": "%VALUE%",
  "timestamp":"%NOWUTC.TEXT%",
  "status": "%STATUS.GOOD%"
}
```

    - Make sure the checkbox labelled Use different format for writing is not checked
    - Click Apply

When finished, the ICONICS Workbench should look like this:

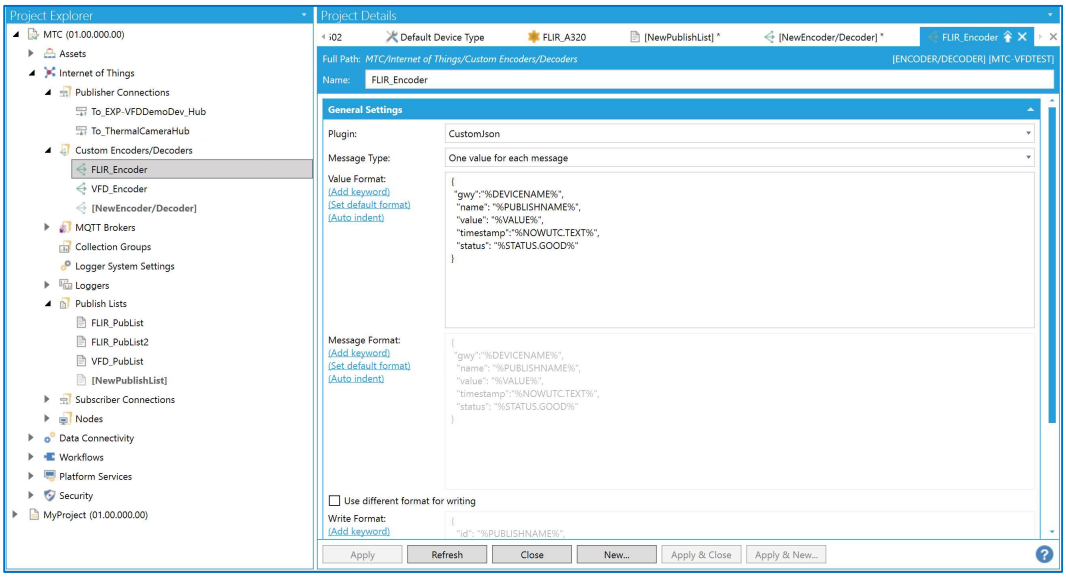

Figure 14 Custom encoder

6.3 Create a publisher connection

Follow these steps to create the publisher connection:

1. In the Project Explorer, expand your project, then Internet of Things
2. Right click on Publisher Connections and click on Add Publisher Connection.
3. In the [NewPublisherConnection] tab
    - Enter “To_ThermalCameraHub” for the Name. Note that for convenience, we used the name of the IoT Hub in the Publisher Connection name.
4. In the General Settings Tab
    - Select “The connection is enabled”. Without this, nothing is sent to the cloud.
    - Make sure that “Enable compatibility with ICONICS clients” is not checked
    - For the Connection Type, select Azure IoT Hub
    - For the Encoder, select the encoder created above, FLIR_Encoder
    - For the Heartbeat rate, enter 20 Seconds
5. In the IoT Hub Settings
    - Paste in the Primary Connection String for the IoT Hub to which you want to send the data. This is the string you copied in Section 3.1, above.
    - For the Protocol, select Automatic
    - For the Max Message Size, select 250,000 bytes
6. Click Apply

When finished, the ICONICS Workbench should look like this:

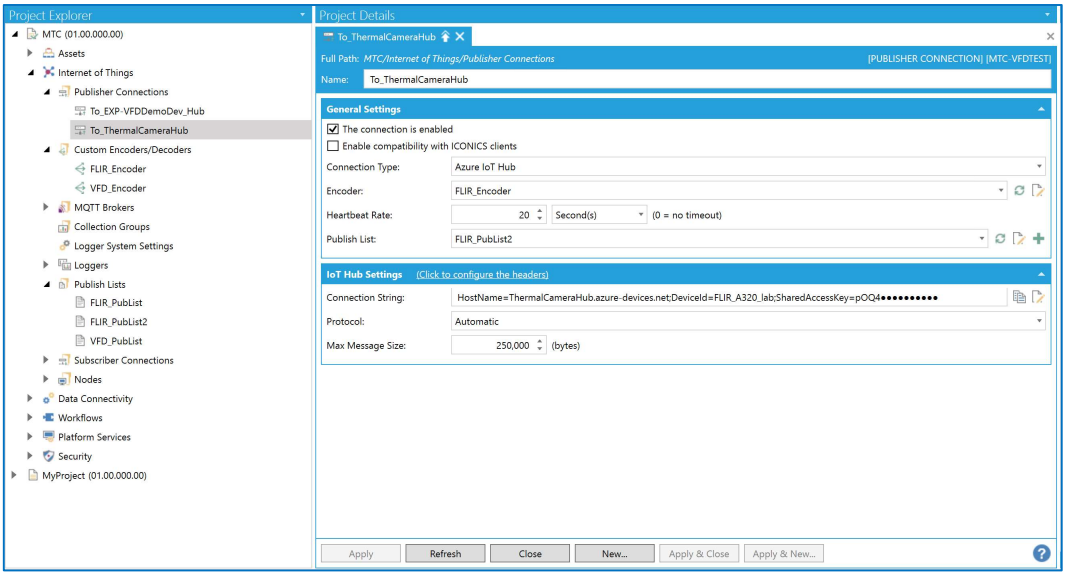

Figure 15 Publisher Connection

After you create and save the Publisher Connection, click the button in the top menu bar to start or restart the Publisher Service. At this point, data should start flowing to Azure IoT Hub, which you can confirm by viewing the data being received at the hub. Follow the instructions in *Using IoTWorX,* Section 4, Viewing the data in Azure IoT Hub, to do this.

7 Storing and reviewing the data

At this point

- The FLIR camera has scanned an object and saved the data in its internal registers
- ICONICS IoTWorX running on a networked computer has read the data from the FLIR camera registers, formatted it, and sent it to Azure IoT Hub

We now need to process the data. In a normal production scenario you might apply some logic (such as Spot 1 Temperature is greater than 101 degrees Fahrenheit) and take an action (such as alert a technician). For the purposes of this document we will simply read the data coming in to Azure IoT Hub and store it in a location where we can download and examine it. To do this, we will use Azure Stream Analytics and Azure Storage.

7.1 Create an Azure Stream Analytics job to send data to blob storage

1. Login to the Azure portal and navigate to the Home Page.
2. Click on the Plus icon to create a new resource
3. Select Internet of Things from the list of options in the column on the left
4. Click on Stream Analytics job from the list of options on the right
5. Give the job a name. In the setup used in this document, we called it 'FLIR_stream_processing'
6. In the left pane, under Job topology, select Inputs
7. Click 'Add a stream input' called StreamInput and select the Azure IoT Hub created earlier, ThermalCameraHub.
8. In the left pane, also under Job topology, select Outputs
9. Click Add and create a blob output called BlobOutput, and select the storage account created earlier, thermalcameratesting.
10. Create a new storage container called flir and add a Path pattern of {date}/{time}. This will create a folder structure under the flir root that will store individual blobs hourly.
11. In the left pane, also under Job topology, select Query
12. Enter a query using standard SQL-like syntax. To start you could use one as simple as SELECT * INTO BlobOutput FROM Streaminput. However, to make the output easier to understand we created the following query, which you should paste into the query editor:

```
SELECT
    name AS Data,
    CASE
        WHEN value >273 THEN ROUND(32 + (value-273.15)*9/5,2)
        ELSE value
    END AS Value,
    CONCAT(
        cast(DATEPART(year,timestamp) as nvarchar(max)),'/',
        cast(DATEPART(month,timestamp) as nvarchar(max)),'/',
        cast(DATEPART(day,timestamp) as nvarchar(max)),' ',
        cast(DATEPART(hour,timestamp) as nvarchar(max)),':',
        cast(DATEPART(minute,timestamp) as nvarchar(max)),':',
        cast(DATEPART(second,timestamp) as nvarchar(max))
    ) as Time
INTO
    BlobOutput
FROM
    Streaminput
```

13. Save your work, then in the left pane select Overview and click the Start button in the right pane to start the job.

7.2 Opening the blob

As data starts coming in from the camera the Stream Analytics job will write it to the Storage folders created earlier. To download and examine the data follow these steps.

1. Login to the Azure portal and navigate to the Home Page.
2. Under Recent resources, click on the storage account thermalcameratesting
3. In the thermalcameratesting window, click on the word Containers part way down the page on the right
4. In the Containers window, click on the folder root, flir
5. Under flir, click on the year, then the month, then the day, then the hour. This is the folder structure created in the previous section, line 10. You will see blobs created there.
6. Click on a blob
7. In the right blade, click on Download
8. In Windows, open the File Explorer and navigate to your downloads folder. The blob will have been downloaded there.
9. Double-click on the blob in the downloads folder to open it up in your default JSON viewer (Notepad, unless you have changed it). The blob will open and look something like this:

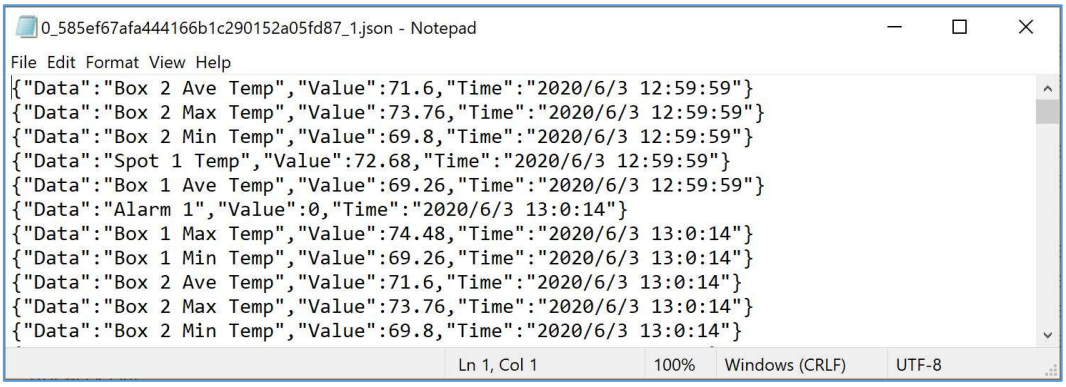

Figure 16 Captured camera data

There is one record for each register you selected in the IoTWorX Publish List, formatted however you specified in the IoTWorX encoder, and then reformatted however you specified in the Stream Analytics query.

8 Next steps

If you followed the steps in this document, you have a working example of how to get eight values every 15 seconds from the thermal camera and push them to block storage in the Azure cloud. Some next steps to turn this into a more comprehensive application might be as follows:

1. Increase the number of registers being polled, adding alarms for example
2. Change the Stream Analytics job to push to a SQL database rather than a blob, to make historical analyses easier
3. Add a Power BI dashboard to show trends and current values
4. Add an alerting mechanism to generate emails or SMS messages
5. Integrate with a workforce management system like Dynamics 365 Field Service to automatically generate tickets and dispatch personnel to investigate if certain criteria met
6. Utilize the ICONICS GENESIS64 suite in a cloud VM to organize, visualize, historize, alarm and analyze data received from IoTWorX

9 Conclusion

As stated in the Introduction, the purpose of this document is not to be a tutorial on how to use a thermal camera, or to suggest its appropriateness for any specific purpose. It is solely to show how to get data from a specific thermal camera into the Azure cloud for processing. Hopefully, the steps outlined are sufficient to get you started.