This readme will attempt to clarify how TCP payloads are constructed on the MeshWorks platform for use with the 2lemetry cloud (using MQTT). The actual data will be in JSON format which will then be transferred using the MQTT protocol over a TCP connection. You should be familiar with JSON and MQTT before modifying this script.

celPy.ApplicationName = **"MeshWorks"** ***This will be used to create the network. All devices on the network need to use the same application name***

celPy.DeviceName = **"Gateway" *This is the user friendly device name for identification on the network***

celPy.IsSleepyDevice = False

# Button definition

buttonPoint = [**"button"**, **"PA3"**, **"digital"**, **"buttonF"**, 1]

buttonValues = [**"discrete"**, 2, **"up"**, **"down"**]

# Data & control point lists

celPy.DataCollectionPoints = [buttonPoint]

celPy.DataCollectionValues = [buttonValues]

celPy.ControlPoints = []

celPy.ControlValues = []

# Button function definition

def buttonF():

value = readDigital()

if (value == 1):

***If the gateway is not connected to the cloud…***

if (bConnected == 0):

tcpPayload = x**"1065" *’10’ is the connect command and ‘65’ is the number of bytes (in hex) left in the payload***

tcpPayload = (tcpPayload + x**"0006"**) ***The length (in hex) of the following string. It will always be 6 in this case. See the next note…***

tcpPayload = (tcpPayload + **"MQIsdp"**)***This is always ‘MQIsdp for the connect command***

tcpPayload = (tcpPayload + x**"03c200b4"**)

tcpPayload = (tcpPayload + x**"000f"**) ***The length (in hex) of the following string.***

tcpPayload = (tcpPayload + **"MidwayGateway01"**)

tcpPayload = (tcpPayload + x**"0024*") The length (in hex) of the following string.***

tcpPayload = (tcpPayload + username) ***This uses the ‘username’ variable that we defined in main()***

tcpPayload = (tcpPayload + x**"0020"**) ***The length (in hex) of the following string.***

tcpPayload = (tcpPayload + md5secret***) This uses the ‘md5secret’ variable that we defined in main()***

print(**"tcp send [%s]"**, tcpPayload)

tcp.send(**"q.m2m.io"**, 1883, tcpPayload)

def cpCallbackTcpReceived(type, ip, srcPort, dstPort, data):

print(**"RX TCP from %s"**, ip)

print(**" src port %d"**, srcPort)

print(**" dst port %d"**, dstPort)

bConnected = 1

def cpCallbackDataPointMessageReceived(deviceName, datapointName, discreteValueString, rangeValue):

if (bConnected == 1):

if (datapointName == **"button"**):

print(**"Button change received from %s"**, deviceName)

tcpPayload = x**"323e001f"** ***0x3e is the important byte in this example. It indicates how many bytes follow in the payload. In other examples, you will see how this changes as the length of the payload changes.***

tcpPayload = (tcpPayload + project)

tcpPayload = (tcpPayload + **"/simulators/sim1"**)

tcpPayload = (tcpPayload + x**"0001"**)

The rest of the code builds a JSON payload to look like: “{"Sensor 1":{"button":"0"}}”

tcpPayload = (tcpPayload + x**"7b22"**) ***Hex bytes for ASCII*** ***{“***

tcpPayload = (tcpPayload + deviceName) ***Sensor X***

tcpPayload = (tcpPayload + x**"223a7b22"**) ***Hex bytes for ASCII*** ***“:{“***

tcpPayload = (tcpPayload + **"button"**)

tcpPayload = (tcpPayload + x**"223a22"**) ***Hex bytes for ASCII*** ***“:”***

tcpPayload = (tcpPayload + rangeValue) ***0 or 1***

tcpPayload = (tcpPayload + x**"227d7d"**) ***Hex bytes for ASCII*** ***“}}***

print(**"tcp send [%s]"**, tcpPayload)

tcp.send(**"q.m2m.io"**, 1883, tcpPayload)

if (datapointName == "reedSw"):

print("Reed switch report received from %s", deviceName)

tcpPayload = x"3243001f" ***We use 0x43 (67) here instead of 0x32 (62) in the button example because “reed switch” is 5 bytes longer than “button”***

tcpPayload = (tcpPayload + project)

tcpPayload = (tcpPayload + "/simulators/sim1")

tcpPayload = (tcpPayload + x"0001")

The rest of the code builds a JSON payload to look like: “{"Sensor 1":{"reed switch":"0"}}”

tcpPayload = (tcpPayload + x"7b22")

tcpPayload = (tcpPayload + deviceName)

tcpPayload = (tcpPayload + x"223a7b22")

tcpPayload = (tcpPayload + "reed switch")

tcpPayload = (tcpPayload + x"223a22")

tcpPayload = (tcpPayload + rangeValue)

tcpPayload = (tcpPayload + x"227d7d")

print("tcp send [%s]", tcpPayload)

tcp.send("q.m2m.io", 1883, tcpPayload)

if (datapointName == "tempSensor"):

print("Temperature report received from %s", deviceName)

tcpPayload = x"3244001f" ***We use 0x44 (68) here instead of 0x32 (62) in the button example because “temperature” is 5 bytes longer than “button” and we also need a second digit in rangeValue for a total increase of 6 bytes***

tcpPayload = (tcpPayload + project)

tcpPayload = (tcpPayload + "/simulators/sim1")

tcpPayload = (tcpPayload + x"0001")

The rest of the code builds a JSON payload to look like: “{"Sensor 1":{"temperature":"74"}}”

tcpPayload = (tcpPayload + x"7b22")

tcpPayload = (tcpPayload + deviceName)

tcpPayload = (tcpPayload + x"223a7b22")

tcpPayload = (tcpPayload + "temperature")

tcpPayload = (tcpPayload + x"223a22")

tcpPayload = (tcpPayload + rangeValue) ***2-digit temperature***

tcpPayload = (tcpPayload + x"227d7d")

print("tcp send [%s]", tcpPayload)

tcp.send("q.m2m.io", 1883, tcpPayload)

if (datapointName == "humiditySensor"):

print("Humidity report received from %s", deviceName)

tcpPayload = x"3240001f" ***We use 0x40 (64) here instead of 0x32 (62) in the button example because “humidty” is 1 byte longer than “button” and we also need a second digit in rangeValue for a total increase of 2 bytes***

tcpPayload = (tcpPayload + project)

tcpPayload = (tcpPayload + "/simulators/sim1")

tcpPayload = (tcpPayload + x"0001")

The rest of the code builds a JSON payload to look like: “{"Sensor 1":{"humidity":"22"}}”

tcpPayload = (tcpPayload + x"7b22")

tcpPayload = (tcpPayload + deviceName)

tcpPayload = (tcpPayload + x"223a7b22")

tcpPayload = (tcpPayload + "humidty")

tcpPayload = (tcpPayload + x"223a22")

tcpPayload = (tcpPayload + rangeValue) ***2-digit humidity***

tcpPayload = (tcpPayload + x"227d7d")

print("tcp send [%s]", tcpPayload)

tcp.send("q.m2m.io", 1883, tcpPayload)

celPy.addTickFunction(ping2lemetry, 200)

def ping2lemetry():

if(bConnected == 1):

***# send a ping every 20 seconds to keep session alive when not publishing***

tcpPayload = x**"c000"**

tcp.send(**"q.m2m.io"**, 1883, tcpPayload)

def main():

***# Connect to 2lemetry***

bConnected = 0

***The following three variables will be unique to your project and can be found on your Credentials page under your project***

project = **"AaXYLSrpLSTEO4ED"**

username = **"abcd1234-ab12-cd34-1234-abcde1234567"**

md5secret = **"64bc245216728cc22cc7d8bf981560ca"**

tcpPayload = x**"1065"**

tcpPayload = (tcpPayload + x**"0006"**)

tcpPayload = (tcpPayload + **"MQIsdp"**)

tcpPayload = (tcpPayload + x**"03c200b4"**)

tcpPayload = (tcpPayload + x**"000f"**) ***The length (in hex) of the following string.***

tcpPayload = (tcpPayload + **"MidwayGateway01"**) ***15 bytes (0x0f) as indicated above.***

tcpPayload = (tcpPayload + x**"0024"**) ***The length (in hex) of the following string.***

tcpPayload = (tcpPayload + username***) This uses the ‘username’ variable that we defined in main()***

tcpPayload = (tcpPayload + x**"0020"**) ***The length (in hex) of the following string.***

tcpPayload = (tcpPayload + md5secret) ***This uses the ‘md5secret’ variable that we defined in main()***

print(**"tcp send [%s]"**, tcpPayload)

tcp.send(**"q.m2m.io"**, 1883, tcpPayload)