Internet of Things Final Project

A. Joseph W Saval B. Ryan Easter
Department of Electrical and Computer Engineering, Kettering University, Flint, MI, USA

1 Project Purpose

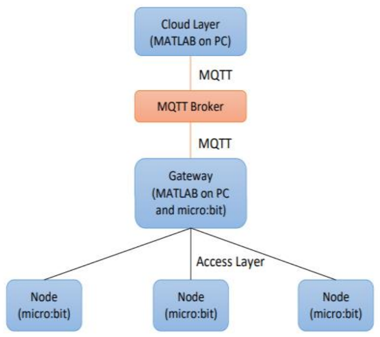
This project aimed to demonstrate the authors' ability to design and implement an Internet of Things network using skills demonstrated and learned from previous lab projects.

1.1 Project Goals

The goals of this project are to create a implementation of an IoT system that will demonstrate a mist, fog, and cloud layer. Along with this a custom protocol must also be designed to communicate between the mist and fog layers, which we call the Access Layer Protocol.

2 Project Design

The structure of each layer will have a specific purpose and design.



As shown above the Cloud layer will be designed using MATLAB, we will then use a MQTT Broker (for this project Hivemq is used) to connect and send data to and from the Cloud and Gateway layers. The Gateway will be split between taking the information from the nodes using micro:bit code and then sending that information across the COM port into the MATLAB Gateway where it can be broken down and sent into the correct MQTT Broker topics. The Access layer is the code that links the micro:bit Nodes to the Gateways.

3 Access Layer Protocol

The access layer was implemented using concepts and design choices copied from a variety of protocols studied in this class. The access protocol meets all requirements as specified on pages two and three of the project instructions.

The access layer protocol operates on group 10 of the micro:bit radio.

3.1 Connection Protocol

Connections are established using a simplified TCP 3-Way Handshake. Only

A device may request a connection with a gateway by broadcasting a message over the micro:bit radio called a SYN message. This device must resend requests on a two second interval until a valid SYNACK is received.

If a gateway reads a valid SYN and determines that it should establish a connection, the gateway must reply with a valid SYNACK on a two second interval until a valid ACK is received.

Once the device receives a valid SYNACK, it must reply with a valid ACK. The device must then wait two and a half seconds without a new SNYACK being received. If a valid SYNACK is read within this time frame the device must listen for another time interval. Once the interval has expired, the device can be considered a valid node and standard communication may commence.

Table 1: 3-Way Handshake Messages

Message	Prefix	Payload
SYN	EASTSAVA	GATE
SYNACK	EASTSAVA	[node ID]
ACK	EASTSAVA	[node ID]

3.1.1 Node ID Generation

The gateway must store a variable indicating the ID of the next node to connect. Only three nodes may be connected at once.

3.2 Standard Communications

All messages must follow the following format:

Table 2: Message Structure

Length (bytes)	Field	Data Type
8	Prefix	Char Array/String
1	Source Micro:Bit ID	Int
1	Destination Micro:Bit ID	Int
1	Message Type	Char
0-3	Payload	Int/IntArray

- Prefix: Must be "SAVAEAST" to indicate that messages come from a validated micro:bit and that it is a standard message, not a 3-Way Handshake.
- Source Micro:Bit ID: ID of the sending Micro:Bit.
- Destination Micro:Bit ID: ID of the intended Micro:Bit destination.
- Message Type: Indicates the data type of the message. Message reliability is determined by the message type. Three message types are supported: 'A', 'L', and 'C' type messages.
- Payload: Message data. Format varies based on data type.
-

3.2.1 Message Type

- A: Acknowledgement from the client to the server specifically for L type messages. Provides semi-reliability for L type messages.
- L: Message from the gateway that requests a node to display a letter on the LED display.
- C: Message from a node providing accelerometer data.

3.2.2 Payload Formats

- A: A 1 byte hex value of the letter the node must display.
- L: A 1 byte character
- C: A three element array of two byte accelerometer values. Values are added by fifteen hundred in order to keep values positive during transit.

4 Node Behavior

Once a node has powered on it will immediately broadcast a SYN message and attempt to connect to a gateway using the connection protocol described in section 3.1.

Once a node has successfully connected to a gateway it will periodically read and send accelerometer values. Values are sent using the C type message format.

Nodes will also display any requested letter onto the LED array. To achieve this, nodes must continuously listen for valid L type messages. Only once a valid message is read will a letter be displayed.

Once a letter has been displayed nodes will reply with an A type message so that the gateway can stop sending messages and enforce semi-reliability.

5 Gateway Behavior

The gateway is split into a micro:bit and computer that communicates over USB. It acts as a medium between the mist and cloud layers by repackaging and forwarding messages between the two.

5.1 Micro:Bit Gateway

The micro:bit gateway communicates with nodes over the access layer and transfers data through the COM link using UART to the computer gateway. Once a gateway has powered on, it will continuously listen for messages from nodes and the computer gateway. The micro:bit gateway establishes node connections in such a way that other messages from verified nodes may be transmitted.

5.1.1 Micro:Bit Gateway Forwarding

If a connection is established with a node, an N type message is passed up to the computer gateway to notify the upper layers that a new node has connected. If a C type message is received from a node, the accelerometer data is forwarded to the computer gateway using a C type message. If a L type message is received from the computer gateway then the requested letter to display is forwarded to the specified node using a corresponding L type message in the access layer. A type messages from nodes are passed back up the stack through the computer gateway.

5.2 Computer Gateway

Commented [JS1]: This needs more info about MQTT comms.

The Computer Gateway transfers messages between the cloud layer using MQTT and the micro:bit gateway using the gateway communication protocol.

The Computer Gateway publishes accelerometer readings from L type messages in the gateway protocol to the cloud using MQTT. It also publishes A type messages to the cloud to acknowledge that a letter has been displayed on the specified node. It transfers letter display requests using C type gateway messages.

5.3 Gateway Protocol

The gateway protocol defines communication between the micro:bit and computer gateways. Because it utilizes a wired connection it is very lightweight and doesn't require any message validation.

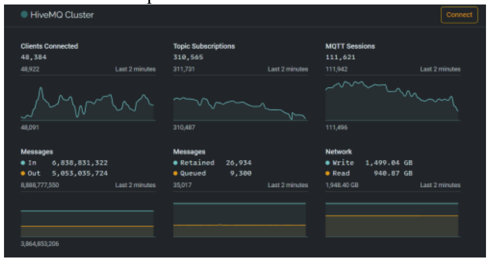
Table 3: Gateway Protocol

Message Type	Format	Description
N	N:[Node ID]	Transferers a notification that a new node has connected up the stack.
C	C:[Source Node ID]:[X Tilt]:[Y Tilt]:[Z Tilt]	Transfers accelerometer readings up the stack.
L	L[destination Node ID][Display Letter]	Transfers a letter display request down the stack
A	A:[Source Node ID]:[Letter Displayed]	Transfers a confirmation that a letter has been displayed up the stack.

6 MQTT Design

6.1 MQTT Broker

The MQTT Broker used was a free online MQTT Broker server that used a TCP connection in order for our Gateway and Cloud to be able to communicate. This MQTT Broker was created by Hivemq as a public MQTT Broker service with multiple other clients that connect to it.



6.2 MQTT Clients

One of the MQTT Clients is the MATLAB Gateway, which does the majority of the publishing to the specified topics we have created in the Hivemq MQTT Broker. The other MQTT Client is MATLAB Cloud, which does the majority of the subscribing to the specified topics we have created in the Hivemq MQTT Broker.

6.3 MQTT Specified Topics

Each MQTT topic that was set up on Hivemq was specified based on the data that the Gateway needed to send to the Cloud. For every node's accelerometer data, each node would have its own topic to publish data to (SAVAEASTN1, SAVAEASTN2, and SAVAEASTN3). There was also a topic created so the Gateway would be able to send an acknowledgement to the Cloud that a node has been connected or that a node has received the requested letter (SAVAEASTack). The final topic created was SAVAEASTLetterReq, so that the cloud could transmit a command through the MQTT Broker and have a specific node show a letter on its LED display.

7 Cloud Design

7.1 Subscriptions

Once the cloud layer receives published data from the MQTT Broker through its subscriptions, it will then use a callback function that reads the topic name and then performs a specified function depending on the topic the data was received from. If the data was received from one of the Node topics (SAVAEASTN1, N2, or N3) it will read the accelerometer data and then plot the X Y and Z points on a single graph for each node. If the data received from the ACK topic (SAVAEASTack) it will write that acknowledgment to the command window whether it was a confirmation of what node has been connected or an acknowledge that the requested letter has been received and displayed on that specific node's LED display.

7.2 Graphing

As mentioned in the overview, for the callback function that reads the topic as either SAVAEASTN1, SAVAEASTN2, or SAVAEASTN3 that function will then graph those accelerometer points on the proper Node's graph. The graph window is set up to receive up to 20 sets of accelerometer values, and then once it receives it's 21st value and beyond, the function graph_n# (# representing either 1, 2, or 3 based on which node the data was read from) will plot the new data while getting rid of the oldest set of data. This allows the graph to only stay update with the 20 most current readings.

7.3 Sending a Letter

When the callback function is activated for the subscription of topic 'SAVAEASTLetterReq', the function activates the response of publishing a letter back to the

MQTT Broker to be continued to be sent down to the node so that it can be displayed on the LED display.