Alex Shah
EN.695.741.81.SP25 Information Assurance Analysis
Mod 4 Assignment
February 18, 2025

1.

Connectivity: OBD2

An OBD2 connection uses a 16 pin connector most often in automotive applications to read vehicle diagnostics and interact with the onboard computer. Commonly the connector uses the CAN protocol (ISO 15765-4) though there are other ISO and SAE standards used with different pin layouts. The CAN protocol has a baud rate of 250k or 500k to communicate over 2 CAN pins at a peak of 1 Mbps. The protocol is used in emissions testing and engine tuning, and for reading error codes in diagnostics.

OBD2 is usually a wired connection though an adapter like an OBD2 reader can use Bluetooth or Wifi to send data wirelessly over 2.4 Ghz to a device like a phone with an app to read codes and data in real time, or dedicated wired/wireless receivers. This makes OBD2 a short range communication connector, typically within a few feet of the vehicle.

There may be security concerns with an unsecured wireless OBD2 reader attached to a vehicle, allowing private vehicle data to be captured which can be used in a replay attack, or the connection could be used to inject commands to the onboard computer.

Link Protocol: Zigbee

Zigbee operates with the wireless standard IEEE 802.15.4 to create a PAN (personal area network) as a mesh network where nodes are deployed to pass messages and extend the reach of the network. Zigbee is a stack from the physical, networking, and application layers as an end to end solution such as for home automation devices like smart devices and sensors. Typically the Zigbee stack isn’t intended to put the devices on the Internet, but allows local communication organized through Zigbee gateway like devices called coordinators.

End devices are usually low powered like IoT devices that communicate over ISM (Industrial, Scientific, Medicine) frequencies like 2.4 Ghz for 250 Kbps data rate or some sub Ghz bands depending on the location, such as 915 Mhz for 40 Kbps in the US. Some of the channels available to Zigbee do not line up with common Wifi 2.4 Ghz channels, ensuring that there are some bands with less interference for Zigbee to conduct its low power transmissions.

The low power <100 mW transmissions travel similar to Wifi distances on 2.4 Ghz, up to 100 feet, though some clear line of sight transmission could cover more distance and sub Ghz could carry signal half a mile. End devices can hop nodes to reach their destination in the mesh network to cover any distance the mesh can reach. For example lights systems, industrial sensors, or roaming trackers can report status or data to the network. These transmissions can be secured with AES-128 encryption to prevent unauthorized reads when the data hops from node to node. Using Zigbee securely requires a well implemented key management system as well as security patching on the end devices and nodes.

Transport: 6LoWPAN

IPv6 over Low Power Wireless Personal Area Network is a link protocol that uses IPv6 packets over a low power wireless network to create a PAN (personal area network). 6LoWPAN, like the Zigbee stack can also use the IEEE 802.15.4 wireless standard, but 6LoWPAN uses standard gateways to adapt IPv6 connectivity for low powered or resource constrained devices. This enables these devices and sensors to connect to the Internet, where the use of IPv6 is intended to help addressing concerns for the large number of devices that are being added to the Internet through IoT.

6LoWPAN can operate through different wireless protocols like IEEE 802.15.4 like Zigbee, or Bluetooth, Wifi, or sub Ghz since it is a protocol on top of an existing physical networking. This means that the distances are similar to Zigbee, where the 2.4 Ghz bands can carry about 100 feet, bluetooth and other protocols could carry signal up to half that distance, or sub Ghz could carry about half a mile. It also supports mesh networking for messages to hop between nodes, and uses strategies like compression and fragmentation to make it more efficient for low power devices to process packets.

Since 6LoWPAN enables Internet connectivity on resource limited devices, processing encryption can be intensive and higher security methods and protocols could be constrained. Typically 6LoWPAN is used on similar devices to Zigbee like environmental monitors, industrial applications, smart agriculture products, and home automation.

Session/Communication: MQTT

Message Querying Telemetry Transport is a publish/subscribe messaging protocol that operates on top of an existing physical/network stack over TCP/IP to coordinate messages between clients via a centralized broker. MQTT is often used on high latency or otherwise unreliable networks to try and send small payloads efficiently. For example MQTT might be deployed in an agriculture setting with a weak existing network for sensors to publish data that a dashboard would subscribe to. The minimal overhead and pub/sub protocol helps ensure that the limited devices and network are still able to relay and receive messages whereas a something like Zigbee experiencing intermittent dropouts could cause lost messages.

Since MQTT can be used on a variety of networks, including Wifi, cellular, and satellite connections. The bands and distances vary with the implementation but the typical size of payloads are bytes to kilobytes in size typically using intermittent or low data rate connections. This low overhead and small payload are useful to ensure message delivery, but create problems for encryption which may increase the resource usage on small low powered devices or create additional overhead.

While MQTT is well suited for small payloads, large feeds like a video or file transmission would be divided into chunks that would increase overhead and add latency. Transport for video would require low latency and is usually done over UDP which does not require confirmation that packets get delivered. So these types of uses go against the benefits that MQTT can provide to the right payloads and would probably overwhelm the network or brokers. An MQTT solution might be to initiate a video feed over a different protocol if a motion sensor is triggered, for example, where the events and metadata are reported over the Message protocol.

Data Aggregation/Processing: Apache Kafka

Kafka is a distributed event streaming platform used for high throughput data applications. Compared to MQTT which has low overhead and can be used on low quality networks, Kafka is meant for continuous large volumes of data to be processed in parallel. It is also a pub/sub model like MQTT but differs in the scale of data being processed at any given time.

While Kafka could be deployed in a limited data situation like MQTT, it would provide the most benefit to cloud environments, or for on site data processing. And unlike MQTT, Kafka uses a distributed log of events enabling persistence of the events and records. While payloads in MQTT might be bytes to kilobytes, Kafka could handle messages several megabytes large. Kafka could work in an agriculture setting with several distributed sensors and process them effectively, but a more realistic use would be lots of messages created continuously like a social media platform where messages are brokered between producers and client feeds.

Kafka can handle slightly larger payloads and more secure methods of encryption since limiting overhead is less of a resource concern on the brokers and higher powered devices that might be generating large amounts of messages. However lower powered devices may not be capable of the high throughput Kafka is intended for. MQTT is still more efficient for small scale deployments and battery powered devices. Millions of messages per second could be passing through Kafka, making it a large scale solution.

2.

Comparing Bluetooth which I am familiar with, and Zigbee which I am less familiar with, there are some similarities as well as unique security threats to both wireless protocols. Bluetooth is a very common short range wireless connection, making threats against Bluetooth devices a larger threat target compared to Zigbee which is less common and usually deployed as a PAN.

There are common avenues for Bluetooth attacks like default or easy to guess pairing PINs, and low levels of encryption. Since these devices are often used in public, sniffing can be performed passively on a large amount of devices at once. There are attacks that leverage vulnerabilities in the Bluetooth stack like Blueborne that could allow remote code execution without needing to pair with the device. In public, attackers could scan and exploit multiple devices at once without physical access or user interaction.

There are also spoofing or impersonation attacks that attempt to use previously paired devices to gain access to a device. Bluetooth devices rely on stored keys which attackers can exploit or use a downgrade attack to circumvent. At the MAC address level, only Bluetooth low energy and recent versions of Bluetooth use address randomization. So impersonating devices by their hardware identifier for use in spoofing or man in the middle attacks are possible on some versions of Bluetooth devices still in use like phones, laptops, and peripherals. Some Bluetooth devices are used in access or security systems like smart locks and keyless entry systems where this type of vulnerability could be dangerous.

Zigbee is a wireless mesh protocol that could also be vulnerable to similar default or hardcoded encryption key vulnerabilities. For example preshared keys could be hardcoded or set to defaults in a Zigbee deployment. There are also replay attacks where an attacker could capture and retransmit

legitimate commands to manipulate devices for example, replaying the unlock command to a smart lock.

Zigbee itself didn't until recently have a way to detect illegitimate retransmitted packets or rogue access to the protocol and devices. For example Zigbee devices would trust known nodes and this trust could be exploited if a device was compromised, potentially getting keys or reading and replaying messages to anyone on the network. While there is effort to reduce some of these common security vulnerabilities, there still isn't a way to centrally monitor transmissions going on between nodes in the network. This leaves messaging security like issuing and following commands securely to be implemented on the devices that are sending and receiving the commands. If Zigbee allows an unlock command to be transmitted again later, it would be up to the lock maker to use something like rolling codes or other means of securing against a replay attack.

These IoT protocols have some similarities to attacks that happen to IT/OT infrastructure such as spoofing and man in the middle attacks. And these wireless protocols as well as IT/OT/ICS networks can use weak encryption and poor credentials that make them vulnerable to unauthorized access. However, the ability to detect and secure against them differs. A large amount of attacks in these categories or events that indicate compromise can be detected or prevented by using host based IDS and network monitoring security software that a traditional network and devices in an IT/OT/ICS deployment would use.

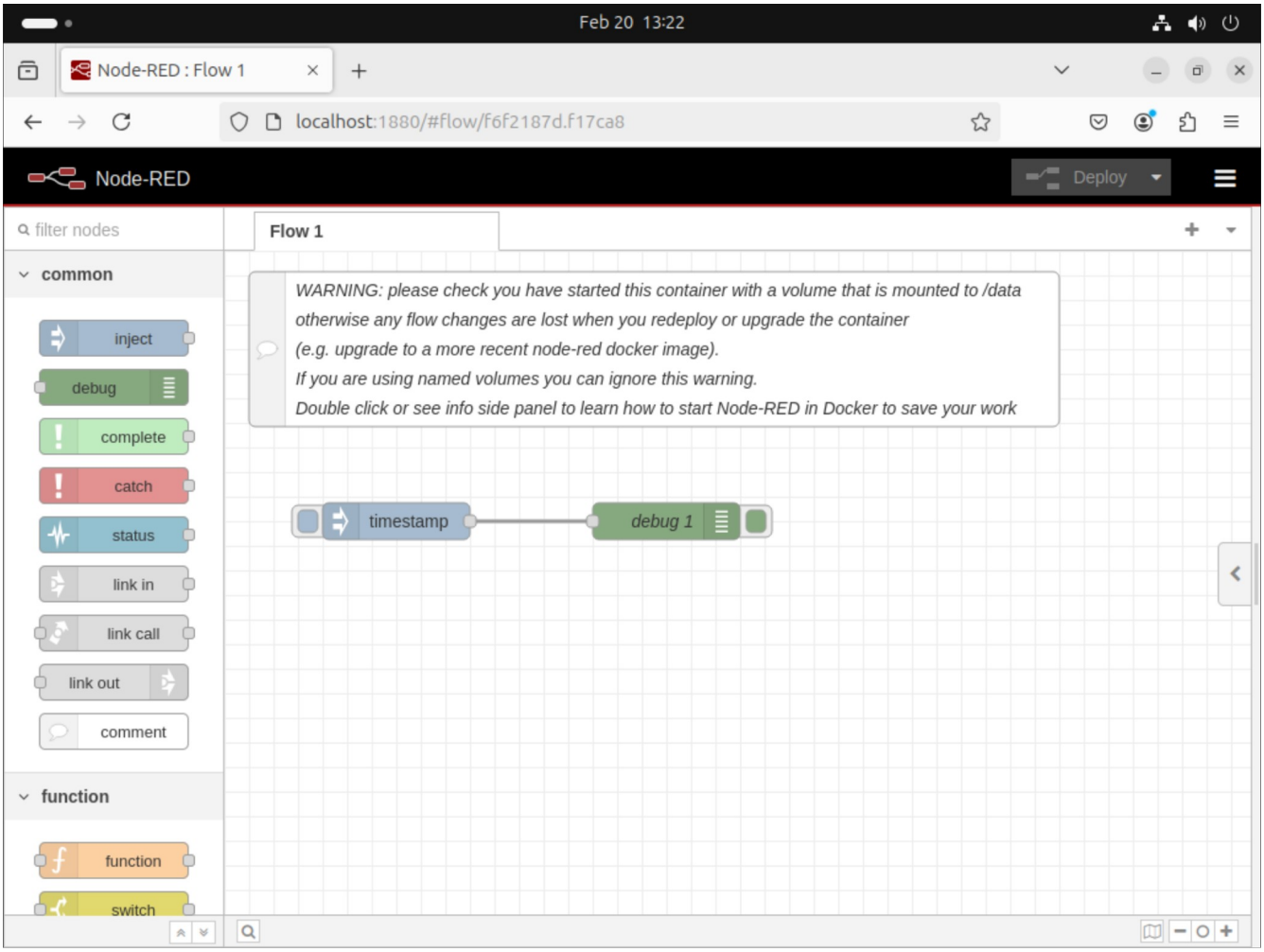
Low powered IoT devices are mostly incapable of running host based IDS software to detect real time intrusion or even to use more secure encryption methods. This makes detection of security events more difficult and potentially allows brute force attacks. Comparatively, traditional systems would be capable of host and network based intrusion detection and prevention in real time as well as leveraging up to date security protocols that limit brute force attempts.

The distributed mesh layout to Zigbee, or point to point connection like Bluetooth, include no central security monitoring like a traditional system might use. For example firewalls and a network IDS are commonly deployed in traditional networks. Without these, an attacker could compromise a Zigbee node and move laterally in the system without detection, or compromise a Bluetooth device. And IoT devices are often deployed and left in the field where someone could gain physical access to them. Having direct access to the hardware could allow easy access to compromising the node and other devices on the network. Comparatively, a legacy IT/OT/ICS deployment would be in a controlled environment with limited access to direct physical threats since server hardware and industrial systems are usually behind lock and key. The common weaknesses of IoT wireless protocols like Bluetooth and Zigbee create security risks unlike traditional networks partially due to their low power and deployment environments which are inherent factors in using IoT in the field.

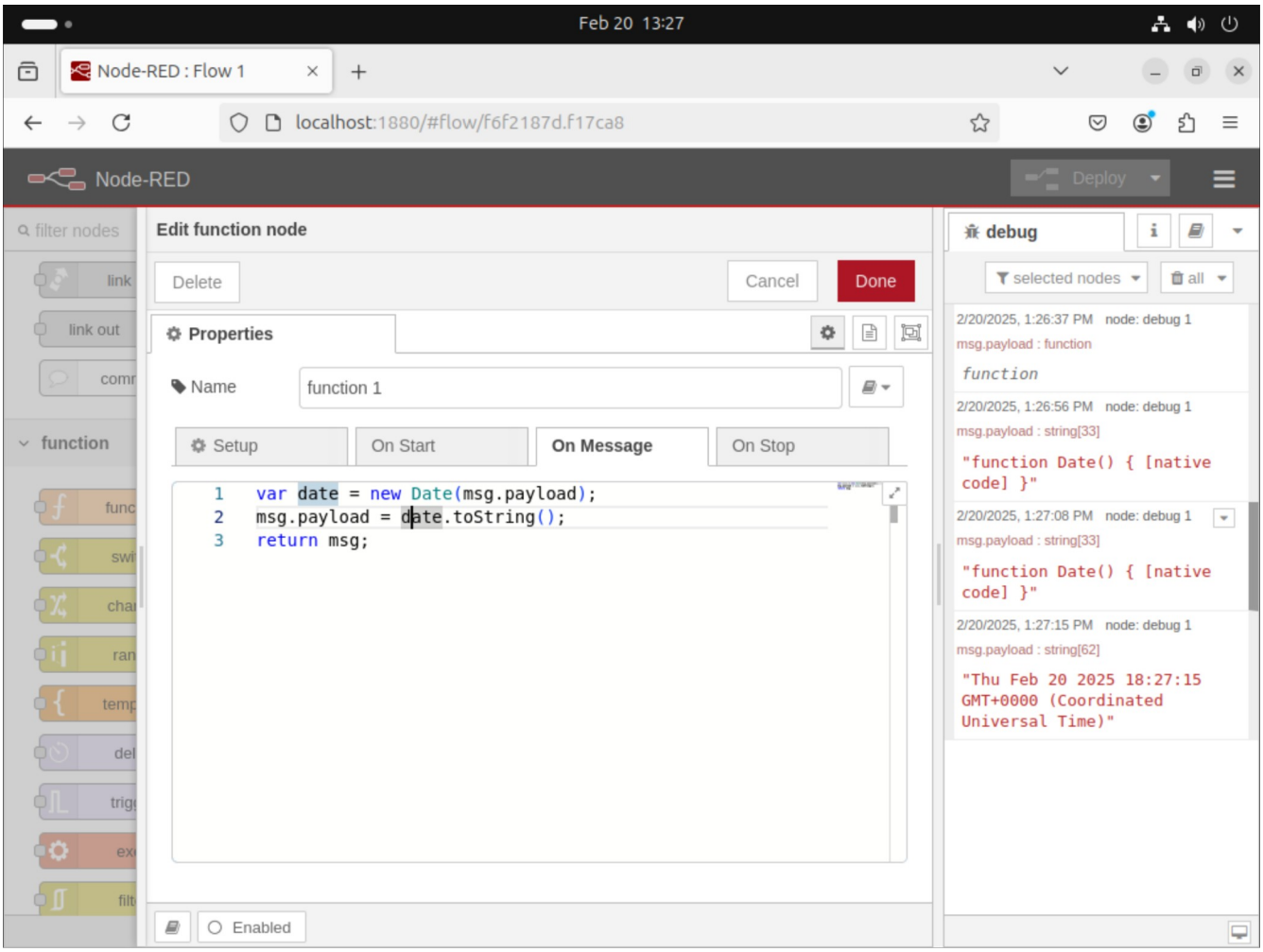
3.

Node Red First Flow:

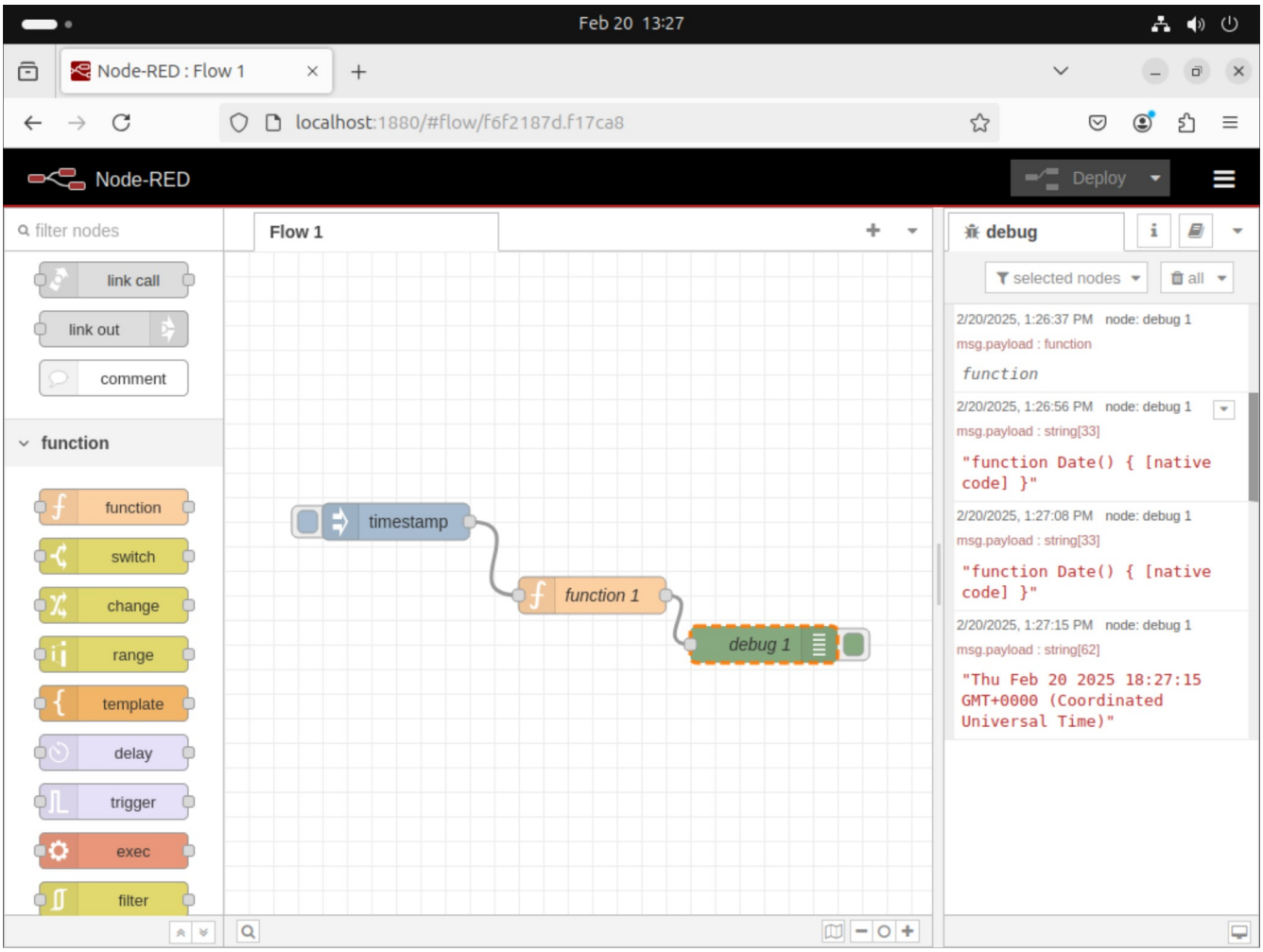
I connected the inject and debug components



And inserted a function in the middle:

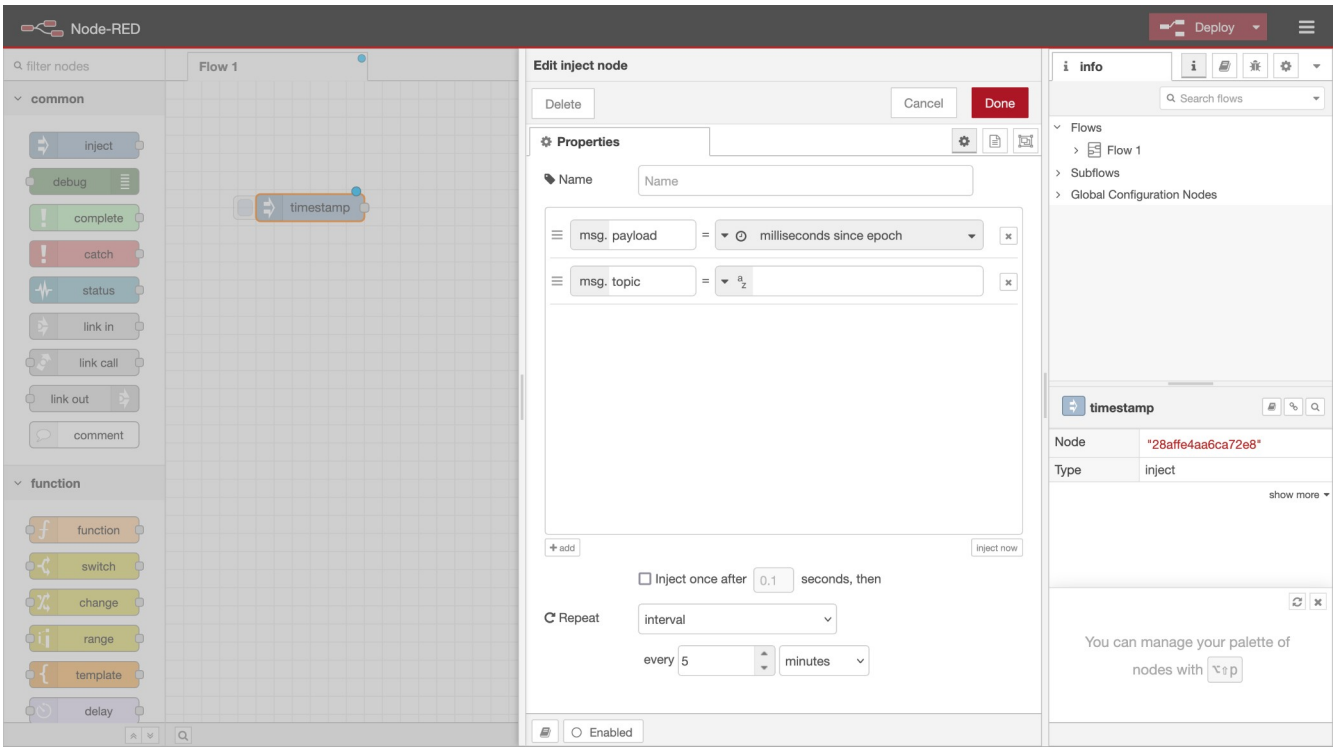


This function showed the timestamp in the debug output:

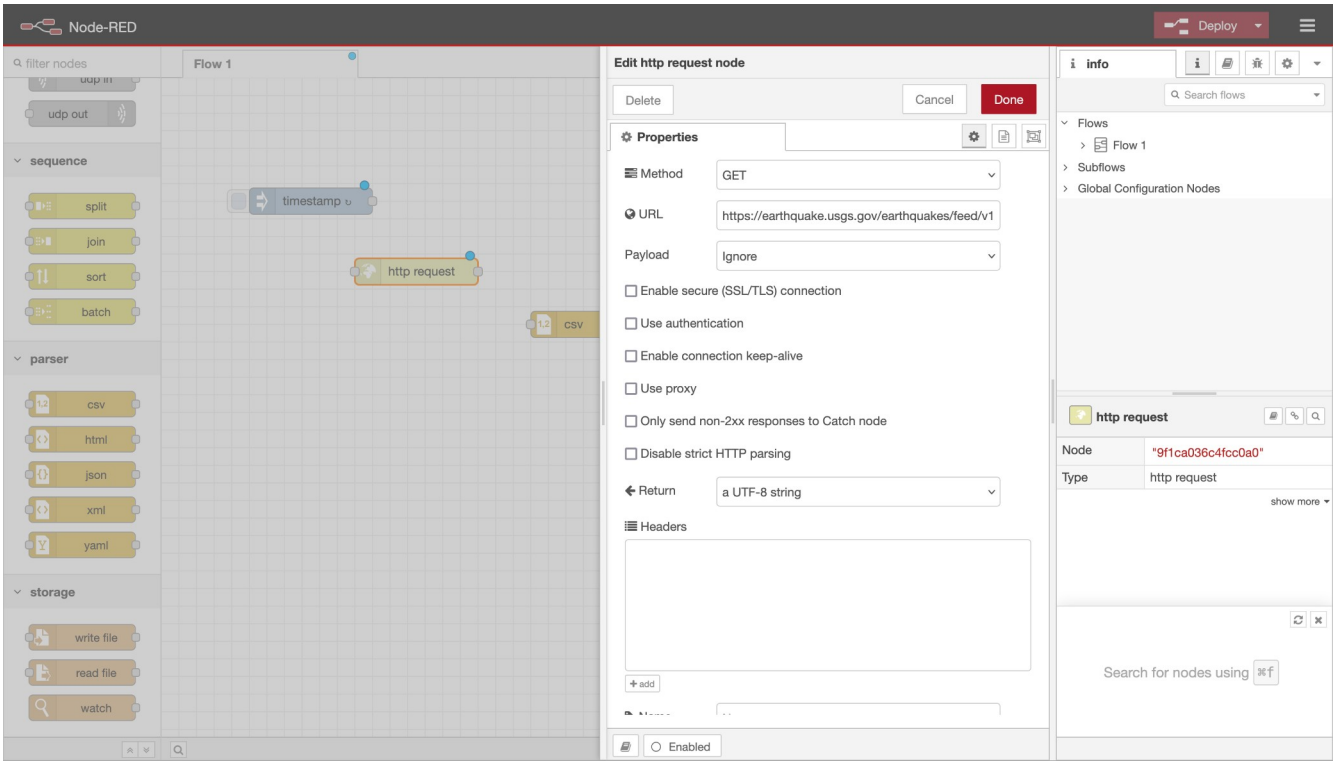


Second flow:

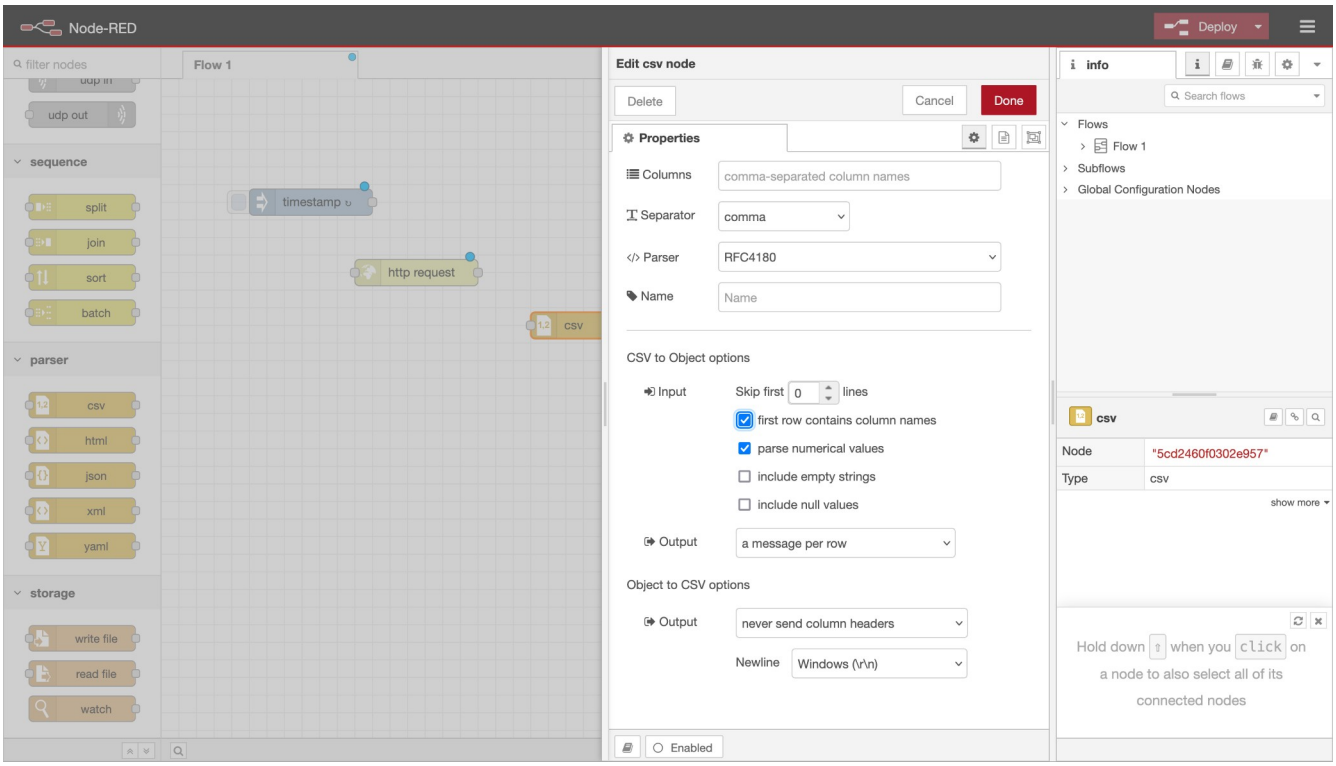
I created an inject node that triggered every 5 minutes



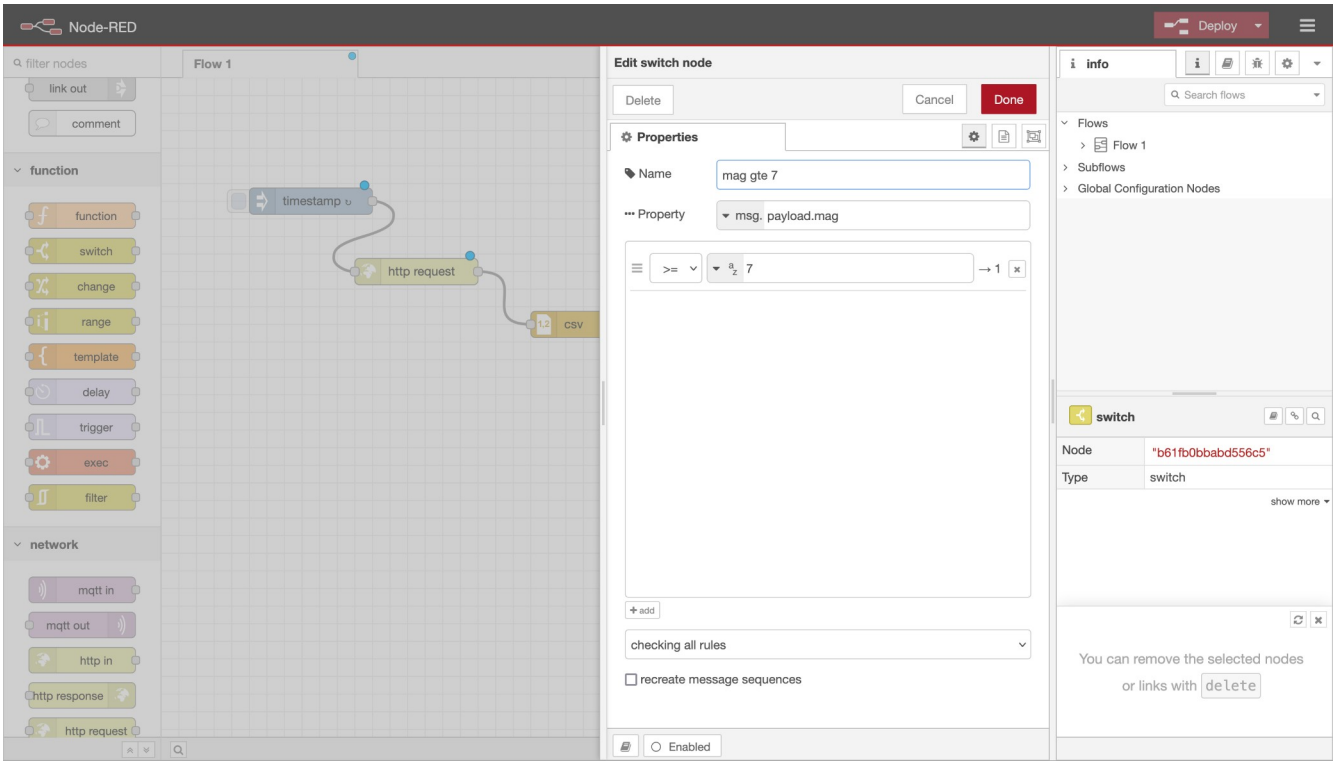
And used an HTTP request node which downloaded a CSV from a URL



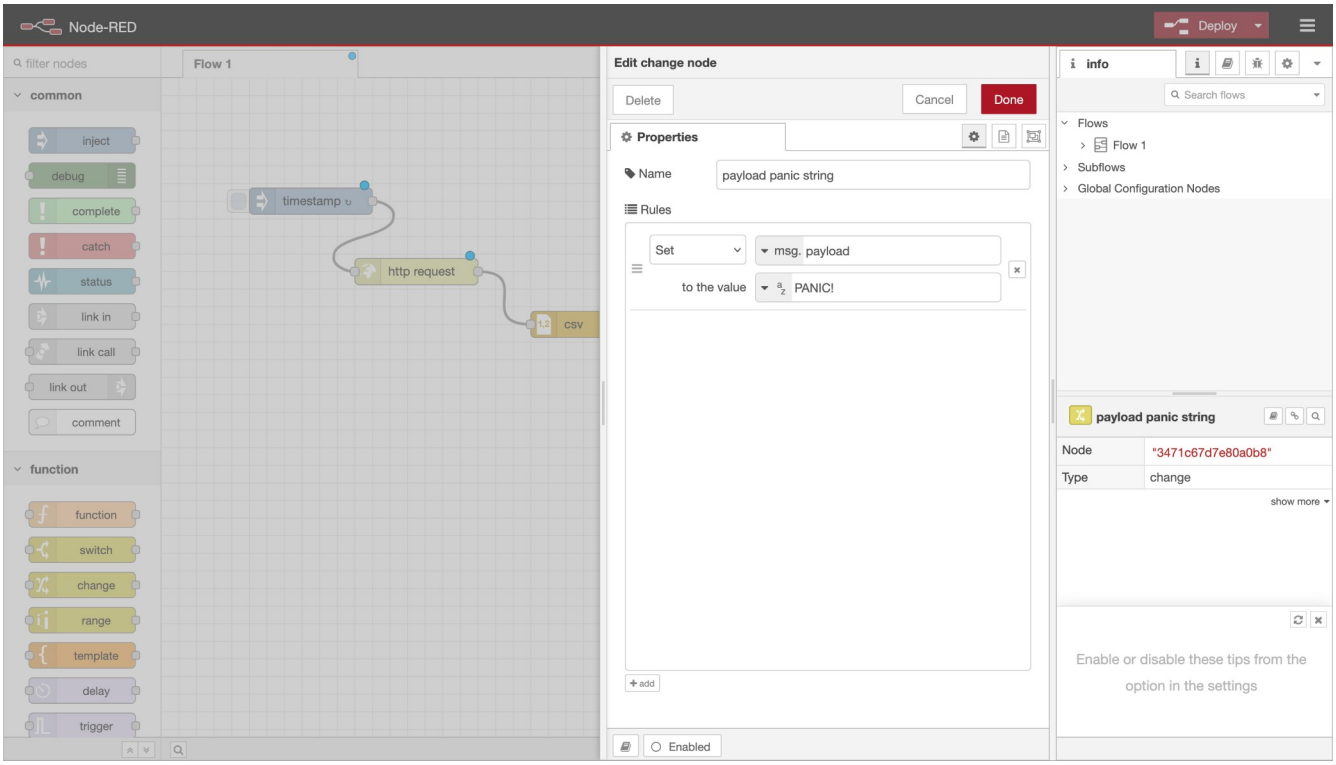
which was then processed by a csv node



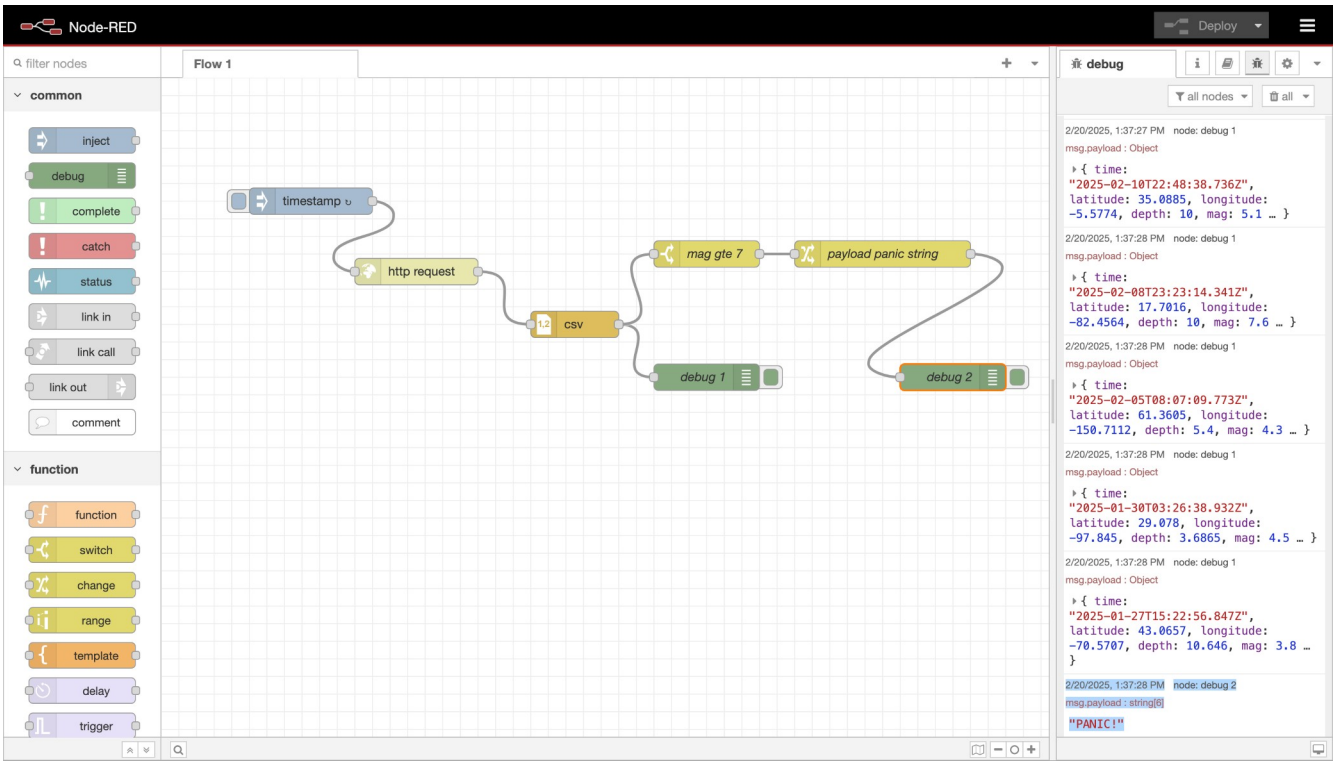
I then added a switch node to define when the magnitude of the earthquake CSV was greater than 7



This leads into a change node which changes the message in the debug output to show “PANIC!” when the earthquake magnitude is greater than 7.



The flow layout has a branching layout to the debug nodes which shows in the output. the debug output reports normally for small earth quakes and shows the “PANIC!” message when the earthquake is large.



4.

One idea is to create a smart weather clock that can tell me when to bring an umbrella or jacket based on the weather and certain criteria. For example an ingest node can trigger every half hour to check the forecast using an http request node to get data from an API like Open Weather. Then there could be a function or simply switch/change nodes to read in certain parameters from today’s upcoming forecast values like precipitation chance or temperature to trigger output. For example, if precipitation is greater than a 60% chance, there could be a message passed to “bring an umbrella” or if the low temperature during the day is less than 50 F, the message could be passed to “bring a coat”. These messages could be sent to a device like a raspberry pi attached an an eink display which would show the weather as well as contain a section for the message to be passed, which could otherwise contain some sort of quip. This setup could be orchestrated via message passing with something like MQTT for the raspberry pi to be set up as a subscriber to the produced content the node-red container produces. Or it could be a webserver in between where the Node-red container uses an http post to change values on the local webserver that the smart clock queries in a loop to display up to date information as soon as the node flow changes the values and the clock refreshes. For real time updates the MQTT approach could be more efficient, allowing the low powered clock to only refresh when there’s a new message to to display, saving power on the eink display. However, the webserver approach could be easier to

deploy and expand on later such as increasing the amount of useful information or allowing a back and forth between the client and the webserver, and webserver and node-red container.

Sources

https://cdn.standards.iteh.ai/samples/67245/caab5ce7ee9b41f8a2f59d60c812c0f7/ISO-15765-4-2016.pdf

https://docs.silabs.com/zigbee/latest/zigbee-security-overview/

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2-upd1.pdf

https://kafka.apache.org/20/documentation.html#gettingStarted

https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

https://www.ti.com/lit/wp/swry013/swry013.pdf