1. **Describe how you would accomplish updating 100+ devices with the same firmware.**
   I believe the best solution for this would be to update them in batches. For example, if you needed to update 1,000 devices, then update them in batches of 50. This way, if something goes wrong mid-update, it would be easier to roll it back and discover which device has a problem.

2. **Imagine the devices respond to messages over a different channel than an HTTP response. For example, imagine the server responds to every valid message to a device with 200, and the device's actual response arrives asynchronously over a websocket. What architecture would you use?**
   I would probably use MQTT over WebSockets as a means of communication. This allows MQTT to be transported over WebSockets which is beneficial because it is bi-directional, ordered, and lossless. Another option would be to create a custom TCP protocol, as WebSocket is an application protocol which makes use of TCP as transport layer. However, I would only suggest this if there were a specific feature required that MQTT over WebSockets did not support. Otherwise, I believe MQTT over WebSockets could accomplish the same task while remaining lightweight and lossless.

3. **In addition to updating the firmware for 100+ devices, imagine each device takes ~30 seconds to respond to each message. Would this change anything?**
   It would change how quickly the process goes. If it would take 30 seconds per response, this would cause a massive delay if you break up the messages so they're under 20 bytes. With the example I did, I sent 73 messages to the server. That means this example would've taken almost 40 minutes to complete. Plus, when you're attempting to compare checksums to make sure it's uploading correctly, there will be a massive delay in the response.

4. **Imagine that in addition to performing firmware updates to devices over a REST endpoint, you also need to communicate with devices over other protocols like MQTT, CoAP, or a custom protocol over TCP. How could your design accommodate this?**
   It would be simple enough to add these protocols. For MQTT, it would be a matter of setting up a broker, and then making sure the device is subscribed to receive and publish messages to and from the right place. CoAP would also be simple as it's based on the REST model. It would just be a matter of posting data through CoAP as well.