Code Review

Code: Joris Elfferich
Reviewer: Tim Leijssen

Algemeen
Ik zou minstens twee witregels tussen functies en classes doen, dat maakt het voor mij iets overzichtelijker.

CODE_FINAL.ino:20-29
```
bool dataSent = false;
int mydata;
```
Zo te zien worden deze niet gebruikt (dataSent wordt wel op 3 plekken op false gezet, maar niet gelezen).
```
uint8_t downlinkData[64];
uint8_t prevDownlink[64];
uint8_t storedData[64];
```
Deze worden alleen binnen de onWrite() functie gebruikt, dus kunnen waarschijnlijk in lokale variabelen veranderd worden.
```
int totalMessages;
```
Ook deze kan lokaal in onWrite().

CODE_FINAL.ino:197-207
```
state = node.beginOTAA(...);
...
while(state != 0 && attempts < maxAttempts) {
    state = node.beginOTAA(...);
    USBSerial.println(state);
    delay(500);
    attempts++;
}
```
Ik zou dit doen met een enkele while-true loop met een paar breaks erin. Dit vind ik iets simpeler. Ik zie ook dat mocht je boven je maxAttempts uitkomen, dat genegeerd wordt.
```
while (true) {
    state = node.beginOTAA(...);
    if (state == 0) { break; }
    attempts++;
    if (attempts >= maxAttempts) { /* Handle error */ }
    delay(500);
}
```

CODE_FINAL.ino:84
```
size_t datalength = sizeof(downlinkData) / sizeof(downlinkSize);
```
Klopt dit? downlinkSize lijkt op de lengte van je downlinkData die word ingevuld een paar regels eerder, maar hier pak je de size van deze variabele in bytes (die is size_t dus zal 2 of 4 zijn).
Moet dit misschien zijn:
```
size_t datalength = sizeof(downlinkData) / sizeof(downlinkData[0]);
```

CODE_FINAL.ino:86-87

```
    byteArrayToUnsignedCharArray(downlinkData, totalUnsigned, datalength);
    String totlen = String((char*)totalUnsigned);
```

Is het kopieren van de array uberhaupt nodig? Meestal zijn char, unsigned char en uint8_t hetzelfde voor ASCII karakters, dus casten tussen pointers naar deze types moet geen probleem zijn.

```
    String totlen = String((char*)downlinkData);
```

CODE_FINAL.ino:130-133

```
    unsigned char unsignedCharArray[arrayLength];
    byteArrayToUnsignedCharArray(storedData, unsignedCharArray, arrayLength);
    USBSerial.println((char*)unsignedCharArray);
    String blockchain = String((char*)unsignedCharArray);
```

Ook hier, kan dit niet gewoon?

```
    USBSerial.println((char*)storedData);
    String blockchain = String((char*)storedData);
```

CODE_FINAL.ino:80,111

```
    while(attempts < maxAttempts) {
```

In beide gevallen wordt attempts alleen binnen de loop gebruikt, dus dit kan gewoon een standaard for-loop worden met de variabele attempts alleen zichtbaar binnen de loop:

```
    for (int attempt = 0; attempts < maxAttempts; attempts++) {
```

CODE_FINAL.ino:121-123

```
    char updatedMessage[10];
    itoa(num, updatedMessage, 10);
    uplinkMessage = updatedMessage;
```

uplinkMessage wordt buiten dit blok gedeclareerd en gebruikt, terwijl de scope van updatedMessage alleen binnen dit blok is. Dus zodra het blok eindigt op r138 wijst uplinkMessage naar ongeldig geheugenruimte dat misschien overschreven kan worden door andere variabelen.
Het zal misschien goed gaan in dit geval, maar eigenlijk is het undefined behavior.
Om dit op te lossen kan je de char updatedMessage[10] declaratie bij de uplinkMessage declaratie zetten, zodat ze dezelfde scope hebben. Misschien dat je van uplinkMessage gewoon een char[10] kan maken, dan heb je updatedMessage niet meer nodig.

CODE_FINAL.ino:139

```
    } else {
        int state = node.downlink(downlinkData, &downlinkSize);
    }
```

"state" maskeert hier een variabele met dezelfde naam (gedeclareerd op r112), ik weet niet of het de bedoeling is om die (r112) state aan te passen. Zoals het nu is wordt deze nieuwe state-variabele in ieder geval niet gebruikt omdat het blok meteen de volgende regel eindigt.