

Payload TTN formatos

Descrição sobre a decodificação do pacote e significado de cada campo. O pacote terá tamanho variável a depender da porta em que é recebido, sendo cada byte formado por dois algarismos em hexadecimais (0 – F).

1. Uplink *Keep Alive*: Porta 1

OF F5 FA 30 00

- **Bateria = Byte 1 e 2:** - os dois primeiros bytes (primeiro e segundo) são referentes a bateria. O valor dos 4 bytes em hexadecimal forma um valor numérico, chamado *batEncoded*. Para encontrar o valor da tensão em volts, é realizado uma decodificação como abaixo:
 - $Battery = \left(\frac{batEncoded \times 3}{4096} \right) \times 1,22 + \text{compensação de erro } (0,1)$
 - $0Xff5 = 4085d.$
 - $\left(\frac{4085 \times 3}{4096} \right) \times 1,22 = 3,66 + 0,1 = 3,66$
 - Corresponde a *bateria* da figura 1.
- **Temperatura = Byte 3 e 4** – O terceiro e o quarto byte são usados para formar o valor da temperatura em graus Celsius. Usando os 2 bytes forma-se um valor numérico em complemento de 2 (verifica-se o primeiro bit desse valor e então decide se o valor é negativo ou não), com o nome de *valor_temp*. Para encontrar o valor em celsius segue a fórmula abaixo:
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = valor_{temp} - 65536 (0x10000).$
 - $Temperatura_{Celsius} = \frac{auxiliar}{256} + 25,00.$
 - Se primeiro bit for 0:
 - $Temperatura_{Celsius} = \frac{valor_{temp}}{256} + 25,00.$
 - $0Xfa30 = 64048. Auxiliar = 64048 - 65536 = -1488$
 - $\frac{-1488}{256} + 25 = 25 - 5,8125 = 19,1875$
 - Corresponde ao *Temperatura* da figura 1.
- **Flag = Byte 5:** O último byte contém as flags (alertas) de algumas situações que podem ocorrer no dispositivo. Essas flags são representadas pelos bits desse byte, nesse caso temos um total de 6 flags, deixando 3 bits sem significado. O byte então será trabalhado na forma *b7 b6 b5 b4 b3 b2 b1 b0*, em que b indica byte e o número sua posição. Cada bit (7 a 0) indicará algum tipo de ocorrência de flag.
 - *Bit 7* = Interrupção de luminosidade.
 - *Bit 6* = Interrupção de movimento.
 - *Bit 5* = Alerta de bateria.
 - *Bit 4* = Sem uso.
 - *Bit 3* = Sem uso.
 - *Bit 2* = Alerta de erro de leitura em algum dos sensores.

- *Bit 1 = Sem uso.*
- *Bit 0 = Sem uso.*



Figura 1 - Uplink na porta 1 - Padrão keep alive

2. Uplink estado de alerta: Porta 2

80 FF 00 F9 00 3B 3F 97

- **Flag = Byte 1:** O primeiro byte contém as flags (alertas) de algumas situações que podem ocorrer no dispositivo. Essas flags são representadas pelos bits desse byte, nesse caso temos um total de 6 flags, deixando 3 bits sem significado. O byte então será trabalhado na forma *b7 b6 b5 b4 b3 b2 b1 b0*, em que *b* indica byte e o número sua posição. Cada bit (7 a 0) indicará algum tipo de ocorrência de flag.
 - *Bit 7 = Interrupção de luminosidade.*
 - *Bit 6 = Interrupção de movimento.*
 - *Bit 5 = Alerta de bateria.*
 - *Bit 4 = Sem uso.*
 - *Bit 3 = Sem uso.*
 - *Bit 2 = Alerta de erro de leitura em algum dos sensores.*
 - *Bit 1 = Sem uso.*
- **Luminosidade = Byte 2** – O segundo byte é usado para formar o valor da temperatura. Neste caso, basta formar o valor numérico usando o byte, o valor formado está na unidade (0 até 255) *lux*.
 - Se 0xFF indicará que está acima de ou igual a 255 lux.
 - Corresponde ao *Luminosity* da figura 2.
 - Logo, 0xFF = 255 lux (Acima de 255).
- **Movimento de aceleração eixo X = Byte 3 e 4** – Os bits três e quatro formam o valor de aceleração do eixo X. Mais uma vez, com esse valor é em complemento de dois, o valor numérico será chamado *mov_acc_X*.

- Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = mov_acc_X - 65536 (0x10000).$
 - $AccX_{ms^2} = \frac{auxiliar}{100} \times 0,061$
- Se primeiro bit for 0:
 - $AccX_{ms^2} = \frac{mov_acc_X}{100} \times 0,061.$
- $0xf9 = \frac{249}{100} \times 0,061 = 0,15189$
- Corresponde ao *MovimAccelerationX* da figura 2.
- **Movimento de aceleração eixo Y = Byte 5 e 6** – Os bits cinco e seis formam o valor de aceleração do eixo Y. Similar ao **Movimento de aceleração eixo X**.
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = mov_acc_Y - 65536 (0x10000).$
 - $AccY_{ms^2} = \frac{auxiliar}{100} \times 0,061$
 - Se primeiro bit for 0:
 - $AccY_{ms^2} = \frac{mov_acc_Y}{100} \times 0,061.$
 - $0x3b = \frac{59}{100} \times 0,061 = 0,03599$
 - Corresponde ao *MovimAccelerationY* da figura 2.
- **Movimento de aceleração eixo Z = Byte 7 e 8** – Os bits nove e dez formam o valor de aceleração do eixo Z. Similar ao **Movimento de aceleração eixo X**.
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = mov_acc_Z - 65536 (0x10000).$
 - $AccZ_{ms^2} = \frac{auxiliar}{100} \times 0,061$
 - Se primeiro bit for 0:
 - $AccZ_{ms^2} = \frac{mov_acc_Z}{100} \times 0,061.$
 - $0x3F97 = \frac{16297}{100} \times 0,061 = 9,94117$
 - Corresponde ao *MovimAccelerationZ* da figura 2.



Figura 2 - Uplink na porta 2 - Padrão estado alerta

3. Uplink do estado atual: Porta 3 (ativado via downlink)

| | | | | |
|------------|----|---|--|--|
| ▲ 10:04:02 | 41 | 3 | dev id: placa_antena_porco | payload: 0F FF 00 D2 03 A7 03 66 3F 26 |
| ▼ 10:04:02 | | 3 | dev id: placa_antena_porco | payload: 00 11 |

Figura 3 - Downlink na porta 3 como o valor 00 11 (estado atual)

0F FF 00 D2 03 A7 03 66 3F 26

- **Bateria = Byte 1 e 2:** - os dois primeiros bytes (primeiro e segundo) são referentes a bateria. O valor dos 4 bytes em hexadecimal forma um valor numérico, chamado *batEncoded*. Para encontrar o valor da tensão em volts, é realizado uma decodificação como abaixo:
 - $Battery = \left(\frac{batEncoded \times 3}{4096} \right) \times 1,22 + \text{compensação de erro (0,1)}$
 - $\left(\frac{4095 \times 3}{4096} \right) \times 1,22 = 3,66 + 0,1 = 3,67$
 - Corresponde a *bateria* da figura 4.
- **Temperatura = Byte 3** – O terceiro byte é usado para formar o valor da temperatura em graus Celsius. Usando o byte forma-se um valor numérico em complemento de 2 (verifica-se o primeiro bit desse valor e então decide se o valor é negativo ou não), com o nome de *valor_temp*. Para encontrar o valor em celsius segue a fórmula abaixo:
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = valor_{temp} - 256 (0x100).$
 - $Temperatura_{Celsius} = \frac{auxiliar}{256} + 25,00.$
 - Se primeiro bit for 0:
 - $Temperatura_{Celsius} = \frac{valor_{temp}}{256} + 25,00.$
 - $0x00 = 00 = \frac{00}{256} + 25 = 25 + 0,0 = 25,00$
 - Corresponde ao *Temperatura* da figura 4.
- **Luminosidade = Byte 4** – O quarto byte é usado para formar o valor da temperatura. Neste caso, basta formar o valor numérico usando o byte, o valor formado está na unidade (0 até 255) *lux*.
 - Se 0xFF indicará que está acima de ou igual a 255 lux.
 - Corresponde ao *Luminosity* da figura 2.
 - Logo, $0xd2 = 210 \text{ lux}.$
- **Movimento de aceleração eixo X = Byte 5 e 6** – Os bits cinco e seis formam o valor de aceleração do eixo X. Mais uma vez, com esse valor é em complemento de dois, o valor numérico será chamado *mov_acc_X*.
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = mov_acc_X - 65536 (0x10000).$
 - $AccX_{ms^2} = \frac{auxiliar}{100} \times 0,061$

- Se primeiro bit for 0:
 - $AccX_{ms^2} = \frac{mov_acc_X}{100} \times 0,061.$
- $0x3A7 = \frac{935}{100} \times 0,061 = 0,57035$
- Corresponde ao *MovimAccelerationX* da figura 4.
- **Movimento de aceleração eixo Y = Byte 7 e 8** – Os bits sete e oito formam o valor de aceleração do eixo Y. Similar ao **Movimento de aceleração eixo X**.
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = mov_acc_Y - 65536 (0x10000).$
 - $AccY_{ms^2} = \frac{auxiliar}{100} \times 0,061$
 - Se primeiro bit for 0:
 - $AccY_{ms^2} = \frac{mov_acc_Y}{100} \times 0,061.$
 - $0x366 = \frac{870}{100} \times 0,061 = 0,5307$
 - Corresponde ao *MovimAccelerationY* da figura 4.
- **Movimento de aceleração eixo Z = Byte 9 e 10** – Os bits nove e dez formam o valor de aceleração do eixo Z. Similar ao **Movimento de aceleração eixo X**.
 - Se primeiro bit for 1 (ou seja, é negativo):
 - $auxiliar = mov_acc_Z - 65536 (0x10000).$
 - $AccZ_{ms^2} = \frac{auxiliar}{100} \times 0,061$
 - Se primeiro bit for 0:
 - $AccZ_{ms^2} = \frac{mov_acc_Z}{100} \times 0,061.$
 - $0x3F26 = \frac{16166}{100} \times 0,061 = 9,86126$
 - Corresponde ao *MovimAccelerationZ* da figura 4.
-

10:04:02 41 3 dev id: [placa_antena_porco](#) payload: 0F FF 00 D2 03 A7 03 66 3F 26

Uplink

Payload

0F FF 00 D2 03 A7 03 66 3F 26 

Fields

```
{
  "Battery": "3.67 V",
  "Luminosity": "210.00 lx",
  "MovimAccelerationX": "0.57 m/s²",
  "MovimAccelerationY": "0.53 m/s²",
  "MovimAccelerationZ": "9.86 m/s²",
  "Temperatura": "25.0 °C"
}
```

Figura 4 - Uplink na porta 3 - Estado do dispositivo

5. Uplink das versões de HW e SW: Porta 5 (ativado via downlink)

| | | | | |
|------------|----|---|--|----------------------|
| ▲ 10:04:05 | 42 | 5 | dev id: placa antena porco | payload: 00 10 01 00 |
| ▼ 10:04:04 | | 5 | dev id: placa antena porco | payload: 00 33 |

Figura 5 - Downlink na porta 5 como o valor 00 33 (informações de dispositivo)

00 10 01 00

- **Versão de HW = Byte 1 e 2** – Os dois bytes serão usados da seguinte forma cada nibble (valor em hexa) corresponderá a um valor de versão. [nibble1]. [nibble2]. [nibble3]. [nibble4], podendo ir da versão 0.0.0.0 até 15.15.15.15.
 - Corresponde a “versãoHW” na figura 6.
 - **0x0010** = 0.0.1.0 = 1.0
- **Versão de SW = Byte 3 e 4** – Os dois bytes serão usados da seguinte forma cada nibble (valor em hexa) corresponderá a um valor de versão. [nibble1]. [nibble2]. [nibble3]. [nibble4], podendo ir da versão 0.0.0.0 até 15.15.15.15.
 - Corresponde a “versãoSW” na figura 6.
 - **0x0100** = 0.1.0.0 = 1.0.0

▲ 10:04:05 42 5 dev id: [placa antena porco](#) payload: 00 10 01 00

Uplink

Payload

00 10 01 00

Fields

```
{
  "versaoHW": "Versão: 1.0",
  "versaoSW": "Versão: 1.0.0"
}
```

Figura 6 - Uplink na porta 5 - Versão de Hardware e firmware

6. Uplink das informações de limiares e tempos: Porta 6 (ativado via downlink)

| | | | | | | |
|----------|---|---|--|-------------------------------------|---------------------------------|--------------|
| 08:29:11 | 3 | 6 | dev id: placa_antena_fio | payload: 00 60 05 01 31 40 F3 D2 BC | Battery_threshold: "3.20 Volts" | Keep_Alive |
| 08:28:54 | | 6 | dev id: placa_antena_fio | payload: 00 77 | | |
| 08:29:08 | 2 | 2 | dev id: placa_antena_fio | payload: 80 FF 00 68 00 58 40 18 | Luminosity: "Acima de 255 lx" | MovimAcceler |
| 08:28:17 | | 6 | scheduled dev id: placa_antena_fio | payload: 00 77 | | |

Figura 7 - Downlink na porta 6 como o valor 00 77 (informações de timers e limiares)

00 60 05 01 31 40 F3 D2 BC

- **Keep Alive = Primeiro byte e primeira metade do segundo byte** – Corresponde um valor em decimal que será multiplicado por 30 segundos para dar o resultado atual do tempo de keep alive.
 - **0x006**. Corresponde $30s * 6 = 3$ minutos.
 - Corresponde ao *Keep_Alive_Timer* da figura 8.
- **Warn Period = primeira metade do segundo byte e o terceiro byte** - Corresponde um valor em decimal que será multiplicado por 5 segundos para dar o resultado atual do tempo de envio de mensagem dentro do estado de alerta.
 - **0x005**. Corresponde $5s * 5 = 25$ segundos.
 - Corresponde ao *Warn_Period* da figura 8.
- **Warn Tx = quarto byte e primeira metade do quinto byte** - Corresponde um valor em decimal que será multiplicado por 5 segundos para dar o resultado atual do tempo do estado de alerta.
 - **0x013**. Corresponde $5s * 0x13 = 5s * 19 = 95$ segundos = 1 minuto e 35 segundos.
 - Corresponde ao *Warn_Tx_Timeout* da figura 8.
- **Limiar de bateria = segunda metade do quinto byte e sexto byte** – Corresponde ao limiar da bateria, valor corresponde ao valor da bateria em centivolts. Logo, para transformar para volts é necessário dividir esse valor por 100.
 - **0x140** = $320/100 = 3,2$ V.
 - Corresponde ao *Battery_threshold* da figura 8.
- **Limiar de movimento** – Correspondente aos limiares de movimento, a saber, (a) limiar de detecção de mudança de ângulo, (b) limiar de detecção de queda livre e (c) tempo de duração para detecção de queda livre. O byte e meio de limiares informa acerca desses limites da seguinte forma:
 - (hexadecimal)XX X = **aa****bb** **b0****cc** **cccc** (binário)

(a) Os dois primeiros bits informam o limiar de detecção de mudança de ângulo da seguinte forma:

- (binário) **aa** = Y (decimal)

- 0 → limiar de 80º
- 1 → limiar de 70º
- 2 → limiar de 60º
- 3 → limiar de 50º

(b) Os três bits seguintes informam o limiar de detecção de queda livre da seguinte forma:

| bbb (binário) | Limite | bbb (binário) | Limite |
|---------------|--------|---------------|--------|
| 000 | 156 mg | 100 | 344 mg |
| 001 | 219 mg | 101 | 406 mg |
| 010 | 250 mg | 110 | 469 mg |
| 011 | 312 mg | 111 | 500 mg |

(c) Os seis últimos bits informam o tempo de duração para detecção de queda livre da seguinte forma:

- (binário) **cc cccc** = Y (decimal),
- Tempo para detecção de queda livre = Y*ODR_Time
- Onde ODR_Time está configurado em 0,625 segundos.
- O tempo para detecção de queda livre vai de 0 segundos com o código 00 até 39,375 segundos com o código 3F.

- **F3D** = **1111 0011 1101** -> 50º; 469 mg; 38,125 seg;
- Corresponde ao *Mov_threshold* da figura 8.

- **Limiar de luminosidade = meio byte do sétimo byte e oitavo byte** - Corresponde ao limiar da luminosidade, valor corresponde ao valor limite em valores unitários de lux. Logo, para saber qual o limiar basta ver qual o decimal correspondente do hexa.

- **0x2BC** - 1000 lux.
- Corresponde ao *Lux_threshold* da figura 8



Figura 8 - Uplink na porta 6 - Limiares e timers configurados

9. Uplink das informações de erro no envio de downlink: Porta 9 (ativado via erro no envio de downlink)

00 10

- **Flag de erro de downlink:** Este byte indica se houve algum erro no envio de downlink, ou seja, seu envio está condicionado a existência de falha na hora do downlink. Além indicar que houve erro é especificado o erro, sendo assim, cada um de seus bits representará um erro.
 - **Bit 7 = Warn Period deve ser menor que Keep Alive.** (+ 0x80)
 - **Bit 6 = Limiar de duração de queda livre não deve ultrapassar 0x40.** (+ 0x40)
 - **Bit 5 = Limiar de queda livre não deve ultrapassar 8.** (+ 0x20)
 - **Bit 4 = Limiar de angulo não deve ultrapassar 4.** (+ 0x10)
 - **Bit 3 = Comando inexistente ou porta errada.** (+ 0x08)
 - **Bit 2 = KeepAlive deve ser maior que WarnTx. (Sem uso)** (+ 0x04)
 - **Bit 1 = Warn Period deve ser maior que warnTx.** (+ 0x02)
 - **Bit 0 = KeepAlive deve ser maior que Warn Period.** (+ 0x01)

| | | | |
|------------|----|-----------|--|
| ▲ 11:57:49 | 38 | 9 | payload: 08 00 downlink_failed: " Comando inexistente ou porta errada " |
| ▼ 11:57:28 | 21 | | payload: 00 33 |
| ▲ 11:57:46 | 37 | 1 | payload: 0F F6 FA 3A 00 Battery: "3.66 V" Temperatura: "19.23 °C" statusBateria: "OK" statusLuminosid: |
| ▼ 11:56:11 | 21 | scheduled | payload: 00 33 |
| ▲ 11:53:36 | 36 | 1 | payload: 0F F5 FA 3E 00 Battery: "3.66 V" Temperatura: "19.24 °C" statusBateria: "OK" statusLuminosid: |

Figura 9 - Simulação de erro no downlink

▲ 11:57:49 38 9 payload: 08 00 downlink_failed: " Comando inexistente ou porta errada |"

Uplink

Payload

08 00

Fields

```
{
  "downlink_failed": " Comando inexistente ou porta errada |"
}
```

Figura 10 - Uplink na porta 9 - Erros de downlink