April 15, 2025

AERTS, BRANDON B.J.J.

Fontys hogeschool ict

Rachelsmolen 1, R10

Communicatie Protocol Drone Project

Explanation and assignment of protocol for drone communication

Below is the second version of the declaration of the communication protocol. The protocol shown here is designed to be used in unison with Haming code but that has not been added to this document for clarity reasons, just know that all further commands are also encoded with 8 | 16 or 16 | 32 hamming depending on the datatype.

These messages consist of 2 parts a Header and the Data Rows. The header can further be divided into 3 different elements, all letting the receiver know what type of command to execute:

**The address:** The address is a unique number that shows who the sender of a message is, this is used to make sure that the LoRa module does not pick up its own message and to ensure that the messages are able to be encrypted securely.

**Command:** The command contains an integer corresponding to the command that you wish to send via the communication protocol. This tells the controller what methods to execute.

**Data Length:** This shows the length of the coming message in bytes. This number includes any increases that come with hamming code parity bits and thus are not accurate with the number of bytes in the DataType column. However, this will make sure the receiver knows how long to listen before doing anything else.

Lastly there is the **Data**, this consists of arguments and gives additional information with the message this is number value and the meaning of this value depends on the Header before it.

Every argument has its own hamming encoding and might use 8 | 16 or 16 | 32 hamming depending on the datatype.

For instance, the Ack message consists of a 16 bit *Adress* which tells the receiver who is the sender, followed by the 8 bit *Command* 0x01 and finally the 8 bit *Length* 0x00. If the Adress of the sender would be 0x17a4 for instance, sending an acknowledge message would look something like this, 0x17a40100, the sender then knows not to pick up the message as it starts with 0x17a4 and the receiver knows not to listen for anymore data after the header has been received as the header ends with 0x00.

## Control Message

**Message Ack:**

*#define ACK 0x0100*

We want to be able to acknowledge when a command has been received correctly and hasn’t encountered an error in transmission, for this we have the simple Ack command.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x01 |
| Data Length | UINT8 | 0x00 |

**Message Nack:**

*#define NACK 0x0200*

When a message does arrive but has been corrupted in such a way that it cannot be fixed or a command is unable to be executed, we want to be able to communicate back that this command could not be issued.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x02 |
| Data Length | UINT8 | 0x00 |

**Message Status:**

*#define STATUS 0x0302*

This command tells the controller what the status of the drone is with added context, it will include the State of the issue like “Low Battery” or “Messages Unreadable”.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x03 |
| Data Length | UINT8 | 0x02 |
| Data | Argument0 | UINT8 | <State> |

## Toggle Message

**Start**

*#define Start 0x0400*

This message tells the drone to start taking in other commands, until this message is received, even if the drone is on, the drone should not try to take flight or listen to any commands aside from toggle commands.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x04 |
| Data Length | UINT8 | 0x00 |

**Stop**

*#define Stop 0x0500*

This message tells the drone to turn off, If the drone senses it is in the air it will first try to land before shutting down.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x05 |
| Data Length | UINT8 | 0x00 |

## Movement Message

**Acceleration Pitch**

*#define APitch 0x0602*

This command tells the drone spin in the pitch direction, this either makes the front fans spin faster then the rear fans when the value is positive or makes the rear fans spin faster then the front fans if the value is negative.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x06 |
| Data Length | UINT8 | 0x02 |
| Data | Argument0 | INT8 | <Speed> |

**Acceleration Yaw**

*#define AYaw 0x0702*

This command tells the drone to spin in the yaw direction, this either makes fans 1 and 3 spin faster and keeps 2 and 4 at a steady pace when this value is positive, or makes fans 2 and 4 spin faster and keeps 1 and 3 steady when the value is negative.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x07 |
| Data Length | UINT8 | 0x02 |
| Data | Argument0 | INT8 | <Speed> |

**Acceleration Roll**

*#define ARoll 0x0802*

This command tells the drone to spin in the roll direction, this makes the fans on the left side of the drone spin faster, and fans on the right side of the drone spin slower. A negative value does the opposite.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x08 |
| Data Length | UINT8 | 0x02 |
| Data | Argument0 | INT8 | <Speed> |

**Acceleration Throttle**

*#define AThrottle 0x0902*

This command tells the drone how fast to move the blades, A positive value makes all the blades move faster and a negative value slows them down.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x09 |
| Data Length | UINT8 | 0x02 |
| Data | Argument0 | INT8 | <Speed> |

## Telemetry Message

**Telemetry Speed**

*#define TSpeed 0x0a10*

This command tells the computer how fast its moving away from the controller and how fast it is moving in general.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x0a |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | FL32 | <Relative Speed> |
| Argument1 | FL32 | <Velocity> |

**Telemetry Acceleration / Pressure**

*#define TAP 0x0b0c*

This command tells the computer how fast the drone is accelerating and what air pressure, in BAR, the drone is currently feeling to calculate the rough height of the drone.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x0b |
| Data Length | UINT8 | 0x0c |
| Data | Argument0 | UINT16 | <Acceleration> |
| Argument1 | FL32 | <Pressure> |

**Telemetry RPM**

*#define TRPM 0x0c10*

This command tells the computer how fast each of the fans are spinning.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x0c |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | UINT16 | <Fan1 RPM> |
| Argument1 | UINT16 | <Fan2 RPM> |
| Argument2 | UINT16 | <Fan3 RPM> |
| Argument3 | UINT16 | <Fan4 RPM> |

**Telemetry Rotor COM**

*#define TCOM 0x0d10*

This command tells the computer what each of the fans is currently doing. Speeding up, slowing down etc.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x0d |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | UINT16 | <Fan1 COM> |
| Argument1 | UINT16 | <Fan2 COM> |
| Argument2 | UINT16 | <Fan3 COM> |
| Argument3 | UINT16 | <Fan4 COM> |

**Telemetry Error**

*#define TError 0x0e04*

If there is an error state that is not critical, this is where it would be sent to the computer.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x0e |
| Data Length | UINT8 | 0x04 |
| Data | Argument0 | UINT16 | <Error> |

**Telemetry Power Status**

*#define TPower 0x0f08*

This sends the battery percentage to the computer so you know when the drone is almost out of power and need to go retrieve it.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x0f |
| Data Length | UINT8 | 0x08 |
| Data | Argument0 | UINT16 | <Battery Voltage> |
| Argument1 | UINT16 | <Circuit Current> |

**Telemetry Current Pitch**

*#define TPitch 0x1008*

This command tells the computer what the current Pitch of the drone is.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x10 |
| Data Length | UINT8 | 0x08 |
| Data | Argument0 | FL32 | <Current Pitch> |

**Telemetry Current Roll**

*#define TRoll 0x1108*

This command tells the computer what the current Roll of the drone is.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x11 |
| Data Length | UINT8 | 0x08 |
| Data | Argument0 | FL32 | <Current Roll> |

**Telemetry Current Yaw**

*#define TYaw 0x1208*

This command tells the computer what the current Yaw of the drone is.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x12 |
| Data Length | UINT8 | 0x08 |
| Data | Argument0 | FL32 | <Current Yaw> |

**Telemetry SOC Load**

*#define TSOC 0x1304*

This command tells the computer how much of the processing power of the ESP32 chip is being used at the moment.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x13 |
| Data Length | UINT8 | 0x04 |
| Data | Argument0 | FL32 | <Current SOC load> |

**Telemetry EMF 1 & 2**

*#define TEMF12 0x1410*

This sends the Electromagnetic Force generated by the rotors 1 and 2 to the computer, this force is denoted in Voltage.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x14 |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | FL32 | <Fan1 EMF> |
| Argument1 | FL32 | <Fan2 EMF> |

**Telemetry EMF 3 & 4**

*#define TEMF34 0x1510*

This sends the Electromagnetic Force generated by the rotors 3 and 4 to the computer, this force is denoted in Voltage.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x15 |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | FL32 | <Fan3 EMF> |
| Argument1 | FL32 | <Fan4 EMF> |

**Telemetry IMotor 1 & 2**

*#define TIMotor12 0x1610*

This sends the current flowing through motor 1 and 2 to the computer.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x16 |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | FL32 | <Fan1 Current> |
| Argument1 | FL32 | <Fan2 Current> |

**Telemetry IMotor 3 & 4**

*#define TIMotor34 0x1710*

This sends the current flowing through motor 3 and 4 to the computer.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Field | Datatype | Data |
| Header | Adress | UINT16 | <Adress> |
| Command | UINT8 | 0x17 |
| Data Length | UINT8 | 0x10 |
| Data | Argument0 | FL32 | <Fan3 Current> |
| Argument1 | FL32 | <Fan4 Current> |