Sensor Communication Protocol Specification

December 26, 2024

1 Overview

This document describes the communication protocol used for transmitting data between the master board and the graphical user interface (GUI) in the metal detector project. The protocol is designed to be robust, extendable, and efficient while ensuring all sensor data from up to 8 sensors is transmitted simultaneously in one message.

2 Message Structure

A message consists of the following fields:

Field	Size (Bytes)	Description
Header	2	Fixed value: 0xAA 0x55
Message Type	1	Specifies the type of message (e.g., data update, command).
Payload Length	1	Number of bytes in the payload.
Payload	Variable	Data specific to the message type (e.g., sensor values).
Checksum	1	XOR of all bytes except the header.

2.1 Field Details

- **Header**: A 2-byte fixed value (0xAA 0x55) marking the start of a message.
- **Message Type**: Defines the purpose of the message. For example:
  - 0x01: Sensor data update
  - 0x02: Command to master board
  - 0x03: Acknowledgment
- **Payload Length**: Indicates the number of bytes in the payload.
- **Payload**: Varies based on the message type. For sensor updates, it contains the values of up to 8 sensors.
- **Checksum**: XOR of all bytes from Message Type to the end of the Payload.

3 Sensor Data Update

The master board sends sensor readings in a single message. The structure of the payload is as follows:

Offset	Size (Bytes)	Description
0	2	Sensor 1 value (16 bits)
2	2	Sensor 2 value (16 bits)
4	2	Sensor 3 value (16 bits)
⋮	⋮	⋮
14	2	Sensor 8 value (16 bits)

Example payload (hexadecimal):

```
12 34 56 78 9A BC DE F0 AB CD EF 01 23 45 67 89
```

3.1 Example Message

Full message in hexadecimal (for 8 sensor readings):

```
AA 55 01 10 12 34 56 78 9A BC DE F0 AB CD EF 01 23 45 67 89 XX
```

Here, `XX` is the checksum.

4 Checksum Calculation

The checksum is calculated by performing an XOR operation on all bytes from the Message Type to the end of the Payload.

Example Python implementation:

```python
def calculate_checksum(message):
    checksum = 0
    for byte in message[2:]:  # Skip header
        checksum ^= byte
    return checksum
```

5 Extensibility

This protocol can be extended for:

- Additional sensors by increasing the Payload Length field and adding more sensor values to the payload.
- New message types (e.g., configuration commands, error reports).
- Variable payload formats based on Message Type.

6 Physical Considerations

- Use reliable transmission media to prevent data loss.
- Consider adding acknowledgments for critical commands.

7 Conclusion

This protocol provides an efficient and robust way to communicate sensor data and commands between the master board and GUI. By consolidating sensor values into a single message, it ensures data consistency and optimizes transmission.