

# Data transmission protocol for “Pi of the Sky” cameras

J. Uzycki<sup>1a</sup>, G. Kasproicz<sup>b</sup>, M. Mankiewicz<sup>c</sup>, K. Nawrocki<sup>d</sup>, P. Sitek<sup>d</sup>, M. Sokolowski<sup>d</sup>, R. Sulej<sup>e</sup>,  
W. Tlaczala<sup>a</sup>

<sup>a</sup>Faculty of Physics, Warsaw University of Technology;

<sup>b</sup>Institute of Electronic Systems, Warsaw University of Technology;

<sup>c</sup>Center for Theoretical Physics PAS, Warsaw;

<sup>d</sup>Soltan Institute for Nuclear Studies, Warsaw;

<sup>e</sup>Institute of Radioelectronics, Warsaw University of Technology.

## ABSTRACT

The large amount of data collected by the automatic astronomical cameras has to be transferred to the fast computers in a reliable way. The method chosen should ensure data streaming in both directions but in nonsymmetrical way. The Ethernet interface is very good choice because of its popularity and proven performance. However it requires TCP/IP stack implementation in devices like cameras for full compliance with existing network and operating systems. This paper describes NUDP protocol [1], which was made as supplement to standard UDP protocol and can be used as a simple-network protocol. The NUDP does not need TCP protocol implementation and makes it possible to run the Ethernet network with simple devices based on microcontroller and/or FPGA chips. The data transmission idea was created especially for the “Pi of the Sky” project [2, 3].

**Keywords:** Gamma Ray Burst (GRB), astronomy, CCD cameras, TCP/IP protocols stack, Ethernet, IP, UDP, NUDP protocol.

## 1. INTRODUCTION

The “Pi of the Sky” experiment [2, 3] is being prepared by Soltan Institute for Nuclear Studies in collaboration with Center of Theoretical Physics, Warsaw University and Warsaw University of Technology. The main goal of the project is a continuous observation of the sky and a real-time analysis of the collected data in search for optical flashes of cosmological origin. Dedicated CCD cameras, meeting specific requirements, have been designed and are being developed as a part of the project.

In the first stage of the project, communication protocol between cameras and computers based on the USB interface was implemented and successfully tested with the “Pi of the Sky” prototype. However, there were only two cameras for photos acquisition and two computers, each having hardware connection with one camera.

The communication protocol has to provide following features: transferring commands to the camera, setting setup parameters of the camera, reading camera status and transferring data read from CCD sensors. In our implementation the USB interface contains 3 endpoints for: commands from computer (via EP1), data input and output. Commands are from 1 to 4 bytes long. The first byte describes the function, next ones – parameters. For example, string of bytes *0x04 | 0x01 | 0x00 | 0x05* means: first byte – focus motor control command, second byte – number of steps in left direction, third and fourth byte – MSB and LSB value respectively. Full list and description of the command set is available, as a part of the USB and Ethernet driver documentation, on our web-site [4].

Plans for the future for the “Pi of the Sky” experiment assume scalability and improved redundancy of whole system. The resulting new requirements are: parallel work of several cameras (final design consists of two 4×4 camera matrices), parallel image processing, autonomy of cameras from computers. The way cameras are connected to the computing cluster must allow continuous operation of the experiment, even when one or more computers fail.

---

<sup>1</sup> juzycki@fuw.edu.pl

Therefore computers and cameras have to be connected to some kind of switch and have unique addresses, so that another PC would be able to replace the failed one. Unfortunately the USB interface cannot meet these requirements. That was the reason why, apart from the existing USB interface, the Ethernet one was implemented in the design.

## 2. THE ETHERNET AND TCP/IP STACK

The Ethernet interface is fast (1Gbit/s data flow rate possible) and cheap. It ensures individual MAC address for every device. There is good availability of networks devices like switches and routers and the connection and infrastructure costs are low. The technology developed for the mass production market was proven to be reliable. Integrated circuits for the Ethernet hardware implementation are easily accessible on the market, eg. Realtek chipsets.

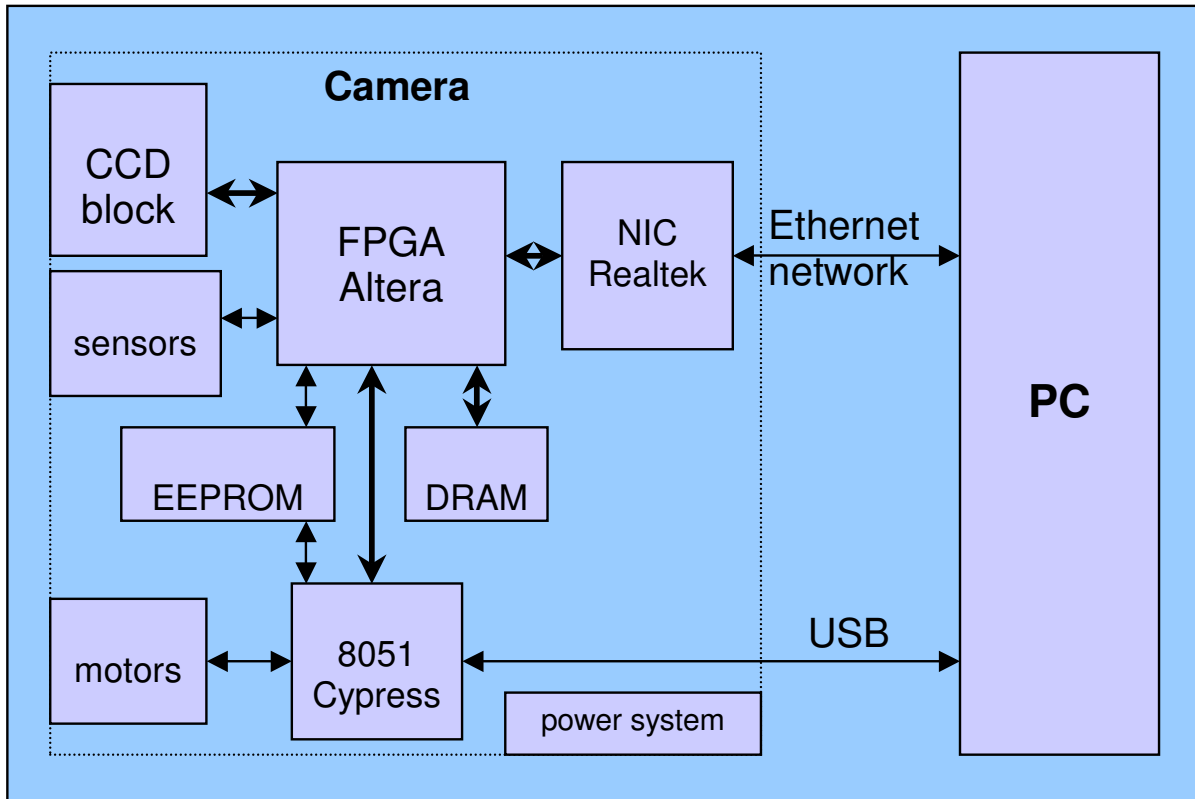


Figure 1. Block diagram of the camera.

The main blocks of the new camera are presented on the diagram in Fig. 1. Altera FPGA and Cypress 8051 microcontroller are the heart of the camera. SDRAM is used to store image data. For the Ethernet communication Realtek's Network Interface Controller (NIC) was selected. It is connected with the FPGA via PCI bus. The Ethernet transmission is controlled by 8051 IC with FPGA support (eg. DMA transfers, NUDP offload). The Ethernet controller automatically calculates not only CRC of ethernet packets but also checksums of IP, TCP and UDP protocols. It significantly offloads CPU. The slow 8051 microcontroller fills only static headers of packets without processing any data.

The Ethernet allows packet transmission (even for non-standard packets) but packet receive acknowledgement mechanism is missing. It has to be provided by a higher level protocol. TCP/IP protocol requires computer's operating system stack usage, therefore the best solution compatible with existing network devices (eg. switches) is IP protocol. Even simple IP protocol implementation allows to transport other protocol implementations like TCP and UDP.

Possible is either full TCP/IP stack implementation or minimized TCP/IP stack implementation with TCP (eg. OpenTCP, uIP, lwIP, Ethernet, etc.) but it is not necessary. Moreover camera's microcontroller is too slow and has not enough RAM memory. Better solution was to give up TCP protocol and to use only UDP as transport layer protocol, which is very simple.

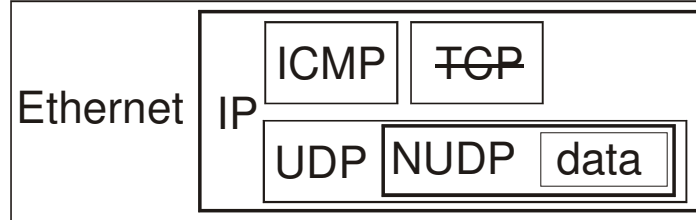


Figure 2. Model of our simple network stack.

Model of the proposed network stack is presented in Fig. 2. Ethernet frame encapsulates IP frame. IP frames service IP addresses and ICMP packets (only ICMP echo is supported in our model for compatibility with *ping* command). Data stream from camera (RAW data – a photo) is divided into 8248 packets 1024 bytes each. Each picture takes 8M bytes as the CCD matrix has resolution  $2k \times 2k$  pixels and each pixel has 2 bytes size. The UDP protocol provides UDP port (one of 64k) dedicated for our application (see NUDP documentation [1]). Optional data checksum is available for reliability improvement. However the choice of UDP also has some disadvantages. UDP datagram does not guarantee the packet delivery. Furthermore a sequence of received packets can be changed accidentally depending on a network traffic. Hence additional protocol encapsulated in the UDP, the NUDP protocol, was created.

### 3. NUDP PROTOCOL

Simple sensor network model is assumed for camera's controller simplification. All advanced functions in the NUDP protocol (e.g. packets retransmission demand, acknowledge mechanism) were moved into Linux driver at a computer controlling the device (camera). Using only camera's received packets acknowledgement, the reliable packet flow control was achieved. A retransmission of lost packets is also possible. Each packet is equipped with unique number, so the driver running at PC is able to detect missing ones. Due to a special structure of NUDP header, compatibility with USB command structure was saved.

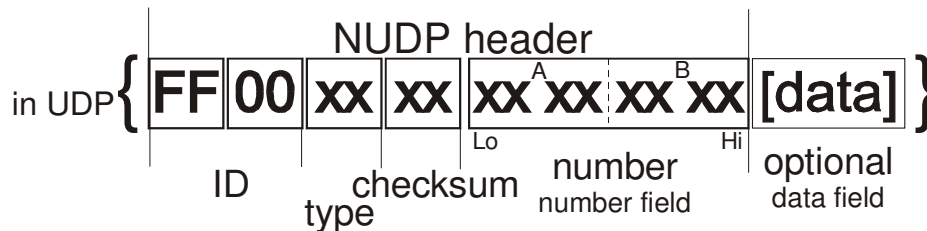


Figure 3. NUDP header structure.

The NUDP header (Fig. 3) always consists of 8 bytes. First two bytes are a simple identification field (ID). The next byte is a type of NUDP frame (Fig. 4). The 4<sup>th</sup> byte is a checksum of NUDP header only. It is an inverted 8-bit sum (checksum as 0) of the first 8 bytes of NUDP frame (header). In another words 8-bit sum of header with calculated checksum should result in 0xFF value. The checksum helps to verify NUDP frame and to give up bad data with the same first two bytes (ID). The last 4 bytes in the header compose multi-function number field. Number field helps to

count RAW packets and it is necessary to identify commands acknowledged by camera, as first 4 bytes of a command are returned in this field.

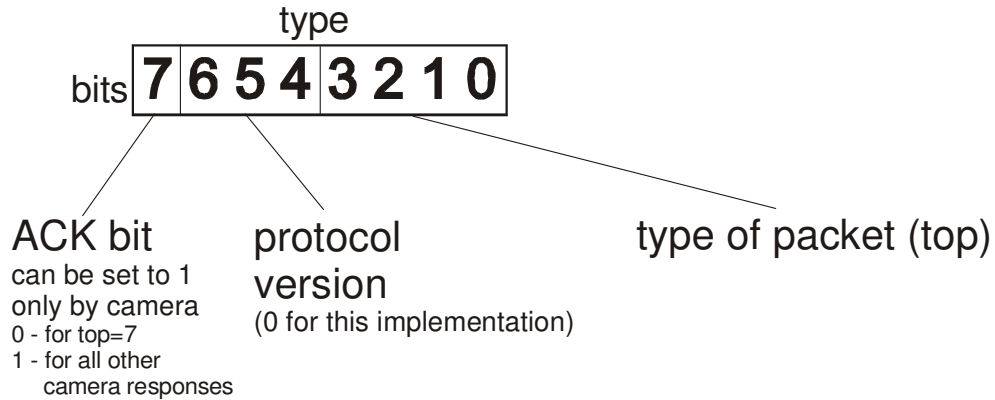


Figure 4. NUDP type field.

The NUDP type field determines the type of a packet (top, see Fig. 5). The field contains also protocol version number and ACK bit. The bit of acknowledgement is a confirmation set by camera that the packet was received and accepted (the whole NUDP header with ACK bit set and new checksum is sent back to the computer). The ACK bit is not set only when TOP value is set to 7, i.e. during dump of RAW picture data from camera's memory. Computer never sends acknowledgements

- 0 – command,  
starts from number field
- 4 – set a 16-bit register (RFU)
- 5 – read a 16-bit register (RFU)
- 6 – retransmission of RAW data packet
- 7 – transmission of RAW data packet,  
never acknowledged and generated by camera only
- Other numbers of type are reserved (RFU)

Figure 5. Types of NUDP packet (top).

Figure 5 specifies some types of NUDP packets. More types are defined in NUDP documentation [1].

<ul style="list-style-type: none"> <li> <b>Parameter setup</b>  (shutter time setting to 10s: 10000ms / 10ms = 0x3E8)  PC: 0xFF 0x00   0x00   0x13   0x02 0x03 0xE8 0x00  Camera: 0xFF 0x00   0x80   0x93   0x02 0x03 0xE8 0x00 </li> </ul>
<ul style="list-style-type: none"> <li> <b>Parameter readout, device recognition etc.</b>  (status and temperatures reading)  PC: 0xFF 0x00   0x00   0xF6   0x0A 0x00 0x00 0x00  Camera: 0xFF 0x00   0x80   0x76   0x0A 0x00 0x00 0x00    0x1E 0x00 0x51 0x4C </li> </ul>
<ul style="list-style-type: none"> <li> <b>Watchdog reset</b> (it should be sent to camera often than 20 seconds – watchdog timeout)  PC: 0xFF 0x00   0x00   0x04   0xFC 0x00 0x00 0x00  Camera: 0xFF 0x00   0x80   0x84   0xFC 0x00 0x00 0x00 </li> </ul>
<ul style="list-style-type: none"> <li> <b>A photo acquisition</b> (start of CCD readout)  PC: 0xFF 0x00   0x00   0xFD   0x03 0x00 0x00 0x00  Camera: 0xFF 0x00   0x80   0x7D   0x03 0x00 0x00 0x00 </li> </ul>
<ul style="list-style-type: none"> <li> <b>Transmission demand</b> (start RAW data dumping)  PC: 0xFF 0x00   0x00   0xF8   0x08 0x00 0x00 0x00  Camera: 0xFF 0x00   0x80   0x78   0x08 0x00 0x00 0x00 </li> </ul>
<ul style="list-style-type: none"> <li> <b>Transmission of RAW data</b>  8248 packets with the following structure (packet number in number field):  Camera: 0xFF 0x00   0x07   0xF7   0x02 0x00 0x00 0x00    &lt;1024 bytes of data&gt; </li> </ul>
<ul style="list-style-type: none"> <li> <b>Retransmission of RAW data packet</b>  PC: 0xFF 0x00   0x06   0xF8   0x02 0x00 0x00 0x00  Camera: 0xFF 0x00   0x86   0x78   0x02 0x00 0x00 0x00    &lt;1024 bytes of data&gt; </li> </ul>

Figure 6. Examples of the most popular types of NUDP frames.

In Fig.6 some examples of commands sent by computer and camera's responses are listed. Please note that camera only responds to host, it never initiates the transfer itself.

## 4. RESULTS

For first tests of the NUDP protocol, dedicated camera simulator NUDPSIM [5] was created. It is based on UDP connections and multithreads. It works under Linux OS and includes NUDP implementation. The simulator and the Linux driver for the camera were written in parallel. This allowed us to speed-up the test phase of the NUDP protocol and camera's driver [6] before the prototype with the final Ethernet implementation was ready. The driver was written without referring to the simulator code, by different team, to prevent duplication of mistakes. A simple Windows application named SockTalk [7] was also created, which allowed low level tests of hardware. Obtained results are more than satisfactory. The NUDP running over 100Mbit Ethernet network infrastructure achieved a data transfer performance very close to the theoretical limit, fulfilling requirements of our application. Moreover, there is an option to run the link at full 1Gbit/s speed. During tests, all packets were usually received under Windows OS, but under Linux OS some packers were lost. A possible reason is that the driver is not implemented as a part or module of the kernel. Therefore some packets could be lost between layers, even when debugging was off.

The performance depends on the kernel UDP timeout for data relaying, driver of the Ethernet card under Linux and on the load of the network (network includes switches). Also the transfer rate depends on the network traffic.

## 5. OUTLOOK

We decided to keep the USB interface in the final design as some commands from the USB were not implemented in the NUDP protocol. For example EEPROM flashing was not included for safety reasons. The EEPROM memory stores the FPGA configuration. The Ethernet controller is behind FPGA (see. Fig. 1), so if something went wrong during transmission, camera could not recover easily and the Ethernet link would be inactive. Therefore the selected solution is to use USB link for safe remote camera's firmware update (both in 8051 and FPGA). Main aims of the presented communication project were already achieved but it still has to be verified in longer running with actual hardware. All software is still under development. In the future we intend to implement the TCP protocol (in one of minimized TCP/IP stacks), RTOS (Real Time OS) and NIOS (Altera soft CPU core) at the FPGA chip. The TCP protocol can be routed much easier than the UDP for long distance connections (no LAN only) and automatically ensures reliability of data streaming.

## ACKNOWLEDGEMENTS

This work was financed by Polish Ministry of Science in 2005-2006 as a research project. It was supported by the grant nr 503 G 1050 0020 006 from the Dean of the Faculty of Physics of the Warsaw University of Technology. We are very grateful to B. Paczynski and G. Pojmanski for encouraging and many practical advices. We would like to thank the staff of the Las Campanas Observatory for their help during the installation of the apparatus.

## REFERENCES

- [1] NUDP protocol homepage, <http://grb.fuw.edu.pl/pi/user/juzycski/> (actual docs | NUDP.pdf)
- [2] M. Cwiok et al., *Search for Optical Counterparts of Gamma Ray Burst*, in *Acta Physica Polonica B*, Vol. 37, No. 3, March 2006, page 919
- [3] M. Biskup et al., *Study of rapidly varying astrophysical objects with the "Pi of the Sky" apparatus*, these proceedings, and "Pi of the Sky" project homepage: <http://grb.fuw.edu.pl>
- [4] Command set of "Pi of the Sky" cameras, Linux driver documentation, <http://grb.fuw.edu.pl/pi0/user/msok/doc/driver/> (only for registered members, contact please)
- [5] NUDPSIM, Ethernet camera simulator – NUDP simulator (works under Linux), <http://grb.fuw.edu.pl/pi/user/juzycski/tests/>
- [6] Driver for camera with the Ethernet (Linux), author: Robert Sulej, <http://www.ire.pw.edu.pl/~rsulej/CEthCamera/>
- [7] SockTalk, application (works under Windows) for UDP / NUDP testings and developing, author: Robert Sulej, <http://www.ire.pw.edu.pl/~rsulej/SockTalk/>