Areeb Hussain

Asylbek Bugybay

Lola Ueda



## 24. April 2022

## 24. April 2022

RUBBER

DUCKY

RUBBER

DUCKY

Detecting “Bad USB” attacks

Detecting “Bad USB” attacks

Contents

[24. April 2022 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031940)

[1.1 24. April 2022 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031941)

[RUBBER 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031942)

[DUCKY 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031943)

[RUBBER 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031944)

[DUCKY 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031945)

[Detecting “Bad USB” attacks 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031946)

[Detecting “Bad USB” attacks 1](file:///C:\Users\Admin\Downloads\RUBBER_DUCKY%20(2).docx#_Toc103031947)

[1 What is Rubber Ducky? 2](#_Toc103031948)

[2 How was Rubber Ducky born? 2](#_Toc103031949)

[3 The usage of Rubber Ducky? 2](#_Toc103031950)

[4 The Rubber Ducky Parts 3](#_Toc103031951)

[5 The Rubber Ducky Costs and alternatives 3](#_Toc103031952)

[5.1 MalDuino 3](#_Toc103031953)

[5.2 WiFi-enabled BadUsb 4](#_Toc103031954)

[5.3 BadUsb cables 4](#_Toc103031955)

[6 The USB Protocol 4](#_Toc103031956)

[6.1 How Does USB Transmit and Receive Data? 4](#_Toc103031957)

[6.2 USB Data Packet Fields 5](#_Toc103031958)

[6.2.1 USB Data Packet 5](#_Toc103031959)

[6.2.2 Types of USB Data Transfers 6](#_Toc103031960)

[7 Counter measurements 7](#_Toc103031961)

[8 Practical Example 8](#_Toc103031962)

[8.1 The process 8](#_Toc103031963)

[8.2 Setting up the lab 8](#_Toc103031964)

[8.2.1 Ducky Encoder 8](#_Toc103031965)

[8.2.2 8.2.2 Ducky Scripts 9](#_Toc103031966)

[8.3 Simple Attack Implementation 9](#_Toc103031967)

[8.4 Simple ATTACK IMPLEMENTATION - 2 12](#_Toc103031968)

[9 References 16](#_Toc103031969)

# What is Rubber Ducky?

The USB **Rubber Ducky** is an Human Interface Device (HID: keyboard, mouse, joystick) that looks similar to an ordinary USB Pen drive. It is used to inject keystroke at superhuman speed into a system and it based on an AMTEL 32bit chip and an SD card. When these automated keyboard entries are carried out the device pretends to be a USB keyboard. The script language that is used to execute the keyboard entries called Ducky Script. It was specifically developed for this tool and stored on a memory card. Once Rubber Ducky is connected to a computer, the prior defined inputs are executed. Not only it cannot be detected by a computer but also by any anti-virus or firewall because it acts as an HID interface. [1]

# How was Rubber Ducky born?

The USB Rubber Ducky was invented by Hak5 founder Darren Kitchen who worked as sysadmin at the time. Apparently, tired of typing the same commands all over again and again. He programmed a development board to emulate the typing for him - and thus the keystroke injection attack was born out of laziness. The miniature bath time friend rubber ducky came to use simply as a case. [3]

# The usage of Rubber Ducky?

The USB Rubber Ducky became the must-have penetration-testing tool (pen-test). This tool allows learning from experiences of many hackers around the world. [3]

The negative association of the term “hacker” with “cybercriminal” gives the impression that all hackers are bad guys but in reality many of them are penetration testers, security enthusiasts, who discover new security vulnerabilities and provide the important information to developers to improve the security. Moreover, the term “hacker” was originally used for people in those professions where engineering and programming involved. By simply, breaking, fixing, and testing, those passionate people pushed the boundaries and advanced new technologies. [4]

Unfortunately, apart from automation, backups and learning, this small but powerful tool opens doors for the people who have chosen the other side and intend to harm. With just a few well-crafted keystrokes the Rubber Ducky runs malicious code unsuspected and could quickly and easily install backdoors into systems, capture credentials, drop malware, exfiltrate documents, and many more. Attackers could leave USB drives at the places they are aiming to attack and patiently wait for users to insert them to their targets, or be physically present at the place where they could use social engineering tactics to gain access to the systems and plug the USB drives into systems themselves. [4] [6]

# The Rubber Ducky Parts

• **MicroSD card:** All the payloads are saved here in this storage device and

when plugged into a victim’s system will do its work (still credentials, drop

malware, exfiltrate documents, etc.).

• **MicroSD-to-USB adapter:** A simple plastic Dongle that is used to mount the

SD card to machine.

• **Mini “keyboard” adapter:** A silicon chip, the main part that sends the

keystrokes to a computer, to insert a microSD card to it. [2]

# The Rubber Ducky Costs and alternatives

The USB Rubber Ducky available on the market and at a modest cost of

approximately 50 euros, which depending on the use we usually plan to make

of it, it is affordable. However, there are a few alternatives at much lower cost.

The first alternative is to use a common USB (there are some constrains, not all

USB drives are suitable). The second involves the usage of a system similar to

“arduino” that equipped with a USB socket. The third solution is to use

“DigiSpark” microcontroller that costs around 7 euros (programmable only

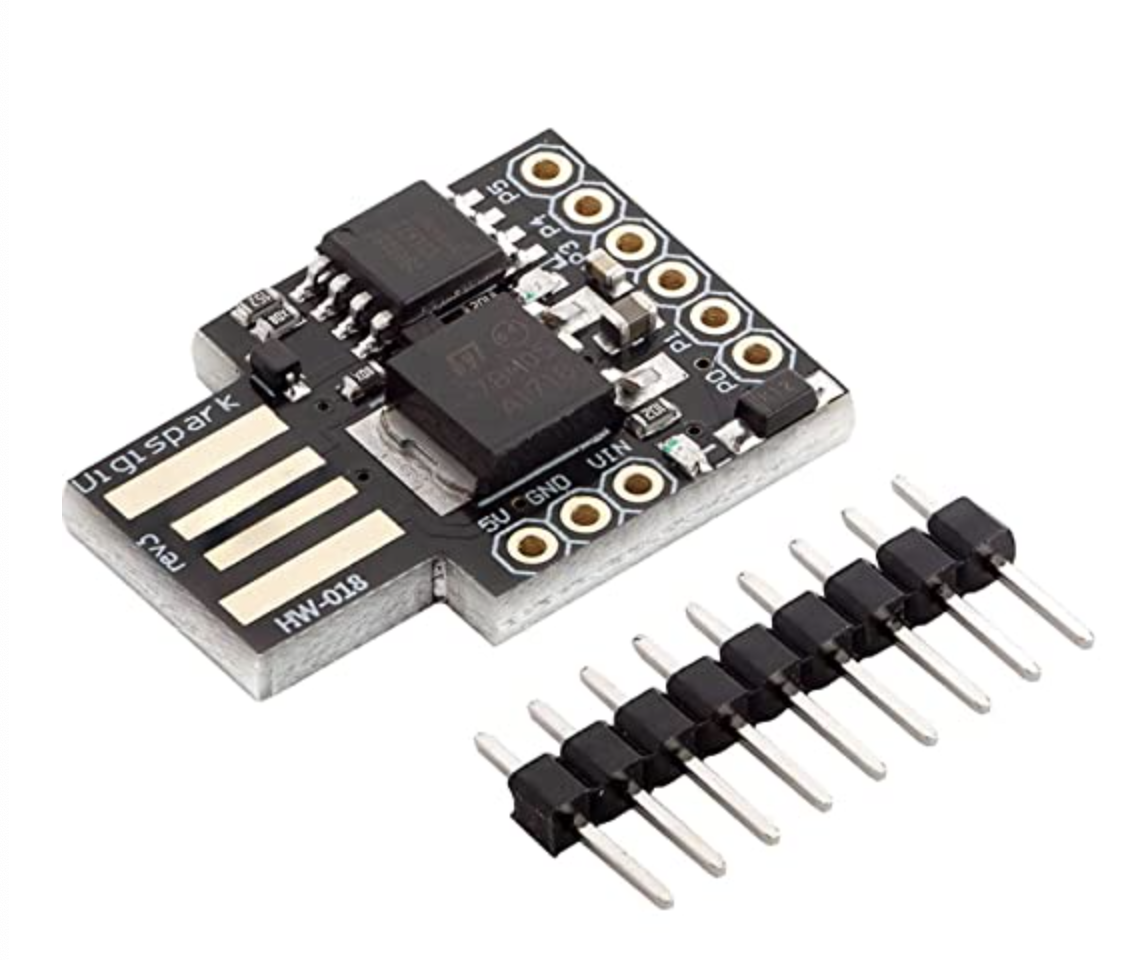
through a USB 3.0 port). [7]

Figure 7. DigiSpark microcontroller

## MalDuino

MalDuino is an open-source Arduino-based BadUSB, similarly it injects malicious payloads into a targeted computer. MalDuino packages several more features compared to regular BadUSB devices by virtue of its onboard computer and grows its popularity. MalDuino devices support Micro SD cards and also have a set of dip switches to allow its user toggle between stored programs within the Micro SD card. [8]

## WiFi-enabled BadUsb

WiFi-enabled BadUSB is specially designed to have WiFi capabilities but similar to MalDuino in that an Arduino board serves as a base for the device. Once inserted into a targeted system, attackers are allowed to introduce malicious payloads into a victim's computer using the WiFi protocol. [8]

WiFi-enabled BadUSB can take different forms depending on the exact purpose and function they serve. The common iterations of this device used today are [8]:

• Wifi-enabled keystroke injection

• WiFi keyloggers

• WiFi deauthers

## BadUsb cables

BadUSB cable also known by names such as USB Ninja and the USB Harpoon, they are generic-looking cables that hide a BadUSB within the internal circuitry. They tend to be more deceptive than any other available variant on the market. A BadUSB cable able to facilitate functions such as charging and data transfer while executing all the malicious activity in the background. [8]

# The USB Protocol

The Universal Serial Bus (USB protocol) was created to uniform cable and connector that could be used across different devices. USB sticks have replaced other mass storage devices such as the floppy disk, the CD-RW because of their higher storage capacity, their easier handling and access speed. The external hard drives are mainly used for backup purposes. [5]

As Rubber Ducky is a USB it is obvious that knowing the mechanism behind a USB and its protocol would be useful.

## How Does USB Transmit and Receive Data?

The USB standard defines how a USB cable or device should operate. There are a variety of mechanisms that must be followed, those include the interaction of various USB devices between themselves (upon enumeration and communication). [9]

USB hosts or also known as master devices initiate all the communication that occurs over the USB bus. A computer or a controller act as a master and only responding to other devices if information is needed. The slave device (peripheral device) is connected to the host device, and is programmed to provide the host device with the information it requires to operate. Those HID devices include USB flash drives, computer mice, keyboards, cameras, and other devices. [9]

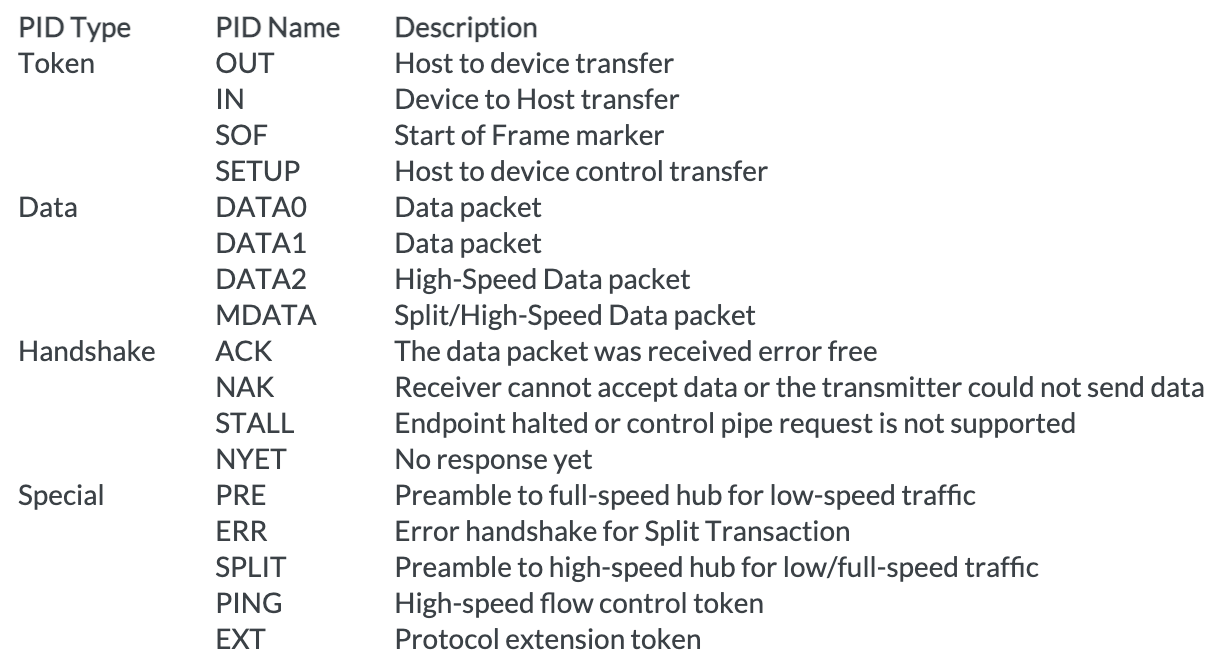
The effective communication over the bus between a host and peripheral devices is an important part. Which leads to questions such as how communication takes places over the USB bus and how is USB data transmitted and received? The theory on how USB data is sent over the bus, the different USB data packet fields and packet types, the types of USB data transfers, explains it in detail. [9]

## USB Data Packet Fields

The USB data packet fields consist of individual bits and include a Sync field, a Packet ID (PID) field, ADDR (Address) field, ENDP (Endpoint) field, CRC (Cyclical Redundancy Check) field, and EOP (End of Packet) field.

• The **SYNC** field is used to synchronise the clocks from both the receiver and transmitter

• The **PID** field provides information on what type of data is being sent. The below table presents the PID Type, the PID Name, and what its purpose is the packet:

Figure 9. USB Packet Types [9]

• The **ADDR** (Address) field includes the address of the device the packet is being

sent to.

• The **ENDP** (Endpoint) field specifies the Endpoint number

• The **CRC** field is used to check the data in the packet for errors

• THE **EOP** field indicates the end of the packet [9]

### USB Data Packet

The fields mentioned above are used to form data packets, they define the various transactions. There are four USB packet types:

**Token Packet**  is initiated by the host and determines if the host will send or receive data. Figure 9.1. Token Packet [9]

**Data Packet**, the transmitter sends the Data and a device can return a NAK (negative-acknowledgment) or Stall packet (the endpoint has halted, or a control pipe does not support a certain request) to indicate if they are not able to respond.

Figure 9.1. Data Packet [9]

**Handshake Packet,** acknowledging data or reporting errors.

Figure 9.3 Handshake Packet [9]

**Start-of-Frame Packet,**the USB bus is split into time segments and schedules the data transfers. After the formation of these packets into frames, they are sent through a USB transaction. The length and frequency of the transaction depends upon the transfer type being used for an endpoint. Figure 9.3 Start-of-Frame Packet [9]

### Types of USB Data Transfers

The communication between a USB host and a USB device is addressed to a specific endpoint on the device. Each device endpoint is a unidirectional specified receiver or transmitter/sender of data from the host. Depending on the bandwidth requirements and the way the data is transferred, each endpoint will be different.

There are four types of USB data transfers:

**Control, Isochronous, Interrupt, and Bulk transfers.** [9]

**Control:** Non-periodic transfers used for device configuration, commands, and status operation.

**Interrupt**: The transaction will definitely occur within a certain time interval. The host will check the device for new data at specified time interval (used by input devices such as mice and keyboards).

**Isochronous**: Devices such as multimedia (audio, video) that need to reserve bandwidth with a high tolerance to errors will periodically and continuously transfer time-sensitive data. No error checks or retransmission of data sent.

**Bulk**: The data is transferred in bulks where the data without errors is a priority over timely arrival. Because general transfers have lower priority, the transactions could be delayed if the bus is busy with other transactions. If an error is detected in the CRC the retransmission take place and error-free data is guaranteed. Often used in file transfers from mass storage devices or outputs from scanners. [9]

# Counter measurements

The following countermeasures could be useful against BadUSB attacks, but could not guarantee secure environment.

**1)** Provide trainings for your team(s)/employees. Never plug in a USB-stick which you do not (fully) trust. Only use USB-sticks that belongs to your company or use a new ones. Never plug in any USB sticks which you found. Never leave your computer unattended and unlocked. Train your employees against (spear) phishing and other social engineering engagements to prevent the attacks.

**2)** Implementation of Admin Rights Management (ARM) is a must. ARM pop-up will appear each time to re-enter user credentials required to start command or PowerShell. Most BadUSB scripts will fail. Avoid granting low-privileged users access to cmd or PowerShell.

**3)** Allow only signed scripts to be run. Configuration is possible on OS-level or in your AV/ATP (Microsoft Defender Advanced Threat Protection) policies.

**4)** Only allow HID devices with trusted ID numbers or block VID value 0x2341 and PID value 0x8037. Note that the blocklist method can be easily circumvented by setting its ID to that of an allow listed keyboard. Leverage Google's USBDLM (a Windows service that gives control over Windows' drive letter assignment for USB drives) and other third party applications.

**5)** Limit physical access to your servers. Disable USB ports in the BIOS/EUFI. BadUSB will still work on machines with removal storage devices disabled since BadUSB registers as an I/O keyboard HID device. Nonetheless, automatic installation of new USB devices could be disabled in your OS.

**6)** Enable OS logging (sysmon) and forward logging to a remote logging server/SIEM/SOC. Increasing your Windows PowerShell event log size to 1GB will help for Window logs to stay not overwritten for a longer time.

**7)** An Intrusion Detection System (IDS) and an Intrusion Prevention System (IPS) might pick up on a reverse shell, but it is too late. [6]

# Practical Example

## The process

Convert into .bin file using Ducky Encoder

Copy .bin and move to the SD card

Plug the Rubber ducky into the targeted machine

Plug SD card into Rubber Ducky

Create a payload in a .txt file

1 2 3 4 5

## Setting up the lab

The Attack requires the following tools:

* Hacking device (e.g. laptop or any other system separated from private system, in this case a Windows system)
* Card reader
* USB Rubber Ducky
* Duckencoder.jar (with preinstalled java on the hacking device)
* Target device (in this case a Windows 10 VM)

The lab environment doesn’t need a lot of preparation as you should do all the work in the field itself. The hacking device, the card reader and the rubber ducky should be ready to use. The hacking device should provide have a preinstalled version of java for Ducky-encoder.

### Ducky Encoder

The USB Rubber Ducky doesn't read the Ducky Script text files natively, rather it expects a binary keystroke injection file. A Duck Encoder is a tool that converts these human readable Ducky Script payload into an Inject.bin file ready for deployment on the ducky. There are several open source, online and cross-platform Duck Encoders available[10].

It can also be simply “installed” by downloading the newest version of the duckencode.jar from hak5darren’s github repository “USB-Rubber-Ducky”: <https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Downloads>

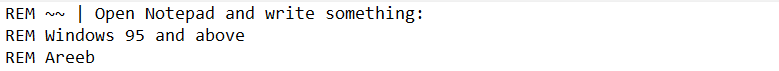
### 8.2.2 Ducky Scripts

Ducky Script is the simple scripting language in which Payloads are written. Ducky scripts can be authored in any standard text editor, such as notepad on Windows, textedit on Mac, Vim, Emacs, Nano, Gedit or Kate on Linux. Ducky Script files must be standard ASCII and cannot contain unicode characters [10].

## Simple Attack Implementation

Since Ducky Script can be written in any standard ASCII text editor, open your favorite - be it gedit, nano, vi, emacs, or even notepad.

We’ll begin our payload with a remark, a comment stating what the payload does, it’s intended target and the author. This won’t be processed by our duck encoder later on, but it will be helpful if we ever share this payload with the community



Our next line should delay for at least one full second. The purpose of this delay is to allow the target computer to enumerate the USB Rubber Ducky as a keyboard and load the generic HID keyboard drivers. On much older machines, consider a slightly longer delay. In my experience no more than three seconds are necessary. This delay is important since the USB Rubber Ducky has the capability of injecting keystrokes as soon as it receives power from the bus, and while USB is capable of receiving the keystroke frames, the operating system may not be ready to process them.



Next we’ll issue our favorite keyboard combination, Windows key + R to bring up the Run dialog



Typically the Run dialog appears near instantly to us humans, however to a USB Rubber Ducky with a clock speed of 60,000 cycles per second, that instant is an eternity. For this reason we’ll need to issue a short delay - perhaps just one tenth of a second



Now with the Run dialog as the active window we’re ready to type our notepad command.

Text

Description automatically generated with medium confidence

The STRING command processes the following characters case sensitive. Meaning STRING C will type a capital letter C. Obviously our keyboards don’t have separate keys for lowercase and capital letters, so our payload actually interprets this as a combination of both the SHIFT key and the letter c - just as you, the human, type. It’s nice to know that the STRING command handles this for you. It does not however end each line of text with a return or enter key, so for that we’ll need to explicitly specify the key.

Graphical user interface, application

Description automatically generated

As before whenever a GUI element changes, we’ll need to wait, albeit briefly, for the window to appear and take focus as the active window. Depending on the speed of the computer and the complexity of the program we’ll want to adjust the delay accordingly. So, I will put a delay of 1 second for this



Finally with notepad open and set as our active window we can finish off our ducky script with these words

A picture containing text

Description automatically generated

At this point our text file should look like the following:

Graphical user interface, text, application

Description automatically generated

Now we need to encode this .txt file via the Ducky encoder. It should look something like this.

Graphical user interface

Description automatically generated with low confidence

https://shop.hak5.org/pages/ducky-encoder

Next, we simply copy the generated .bin and move it to the SD card.

Next we take out the SD card and put it in the rubber duck, a green light indicates that the payload is ready to be executed. If it is red, it means the file is corrupted or the SD is not inserted properly.

Simply plug it in on the target device and Success

## Simple ATTACK IMPLEMENTATION - 2



A screenshot of a video game

Description automatically generated

First line introduces the delay of 1 second, so the rubber ducky can be recognized. After that the Windows d command is pressed to see the Desktop.



Graphical user interface

Description automatically generated

Windows + r command shows you the "RUN" box where you can type commands to either pull up a program or go online.

Text

Description automatically generated

Delay, to synchronize. Write cmd and press enter. Text

Description automatically generatedThis opens Command Prompt.



Delay and write in Command Prompt taskkill command to kill the explorer.exe.



Enter the command.

Now there will be this screen left: Text

Description automatically generated



This command closes the Command Prompt. The user will be left with the only black screen, it will be hard for the normal user to understand what happened.

Shape

Description automatically generated with medium confidence

So there is the full code used:

Graphical user interface, text

Description automatically generated

This keystrokes can be optimized, so that the commands entered faster.

However the speed of the computers are different. So we have to be carefull to not to enter the commands too fast. Otherwise, the commands entered can be incomplete and the sequence of the commands will be broken.

# References

1. URL: <https://de.wikipedia.org/wiki/Rubber_Ducky>

[last accessed: 2022-04-23]

2. URL: <https://www.geeksforgeeks.org/usb-rubber-ducky-penetrationtesting/>

[last accessed: 2022-04-23]

3. URL: <https://shop.hak5.org/products/usb-rubber-ducky-deluxe>

[last accessed: 2022-04-23]

4. URL: <https://blog.teamascend.com/rubber-ducky>

[last accessed: 2022-04-23]

5. URL: <https://de.wikipedia.org/wiki/USB-Massenspeicher#USB-Speicherstick>

[last accessed: 2022-04-23]

6. URL: <https://cryptsus.com/blog/cheap-badusb-rubber-ducky.html>

[last accessed: 2022-04-23]

7. URL: <https://www.secjuice.com/how-to-build-a-low-cost-rubber-ducky/>

[last accessed: 2022-04-23]

Figure 7. DigiSpark microcontroller

8. URL: <https://www.manageengine.com/device-control/badusb.html>

[last accessed: 2022-04-23]

9. URL: <https://www.totalphase.com/blog/2020/07/about-the-usb-protocol-common-usb-bus-errors-and-how-to-troubleshoot-them/>

Figure 9. USB Packet Types

Figure 9.1 Token Packet

Figure 9.2 Data Packet

Figure 9.3 Handshake Packet

Figure 9.4 Start-of-Frame Packet

10. URL: https://docs.hak5.org/usb-rubber-ducky-1/getting-started/the-usb-rubber-duck

[last accessed: 2022-05-08]