Using the Physical Ducky

By Zach Riback

The Digispark microcontroller we are using

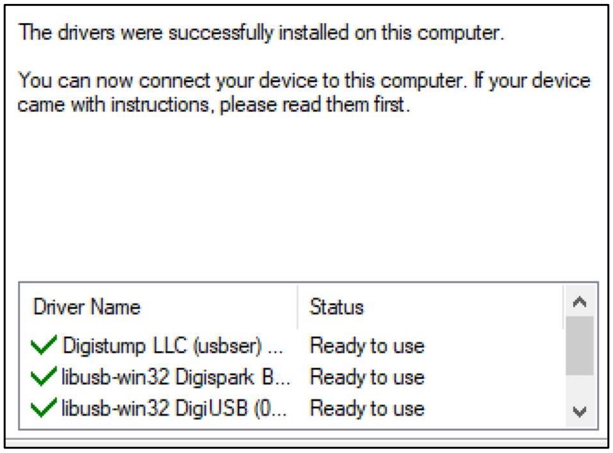
Connecting the Digispark Microcontroller

The first step in turning the microcontroller from the picture above into a proper rubber ducky device is connecting it to our computer. It simply plugs right into a USB port of course but connecting it so we can program it takes a little bit more effort. Firstly, we need to download the bootloader drivers from the following link:

https://github.com/digistump/DigistumpArduino/releases

These drivers are needed to use Arduino to program the ducky. Note that they do not need to be installed for the ducky to execute its code on a hypothetical victim's computer. They are only needed for the purpose of programming.

Run the DPinst64.exe executable to install the drivers (or just DPinst.exe for 32-bit machines):



The drivers were successfully installed on this computer.

You can now connect your device to this computer. If your device came with instructions, please read them first.

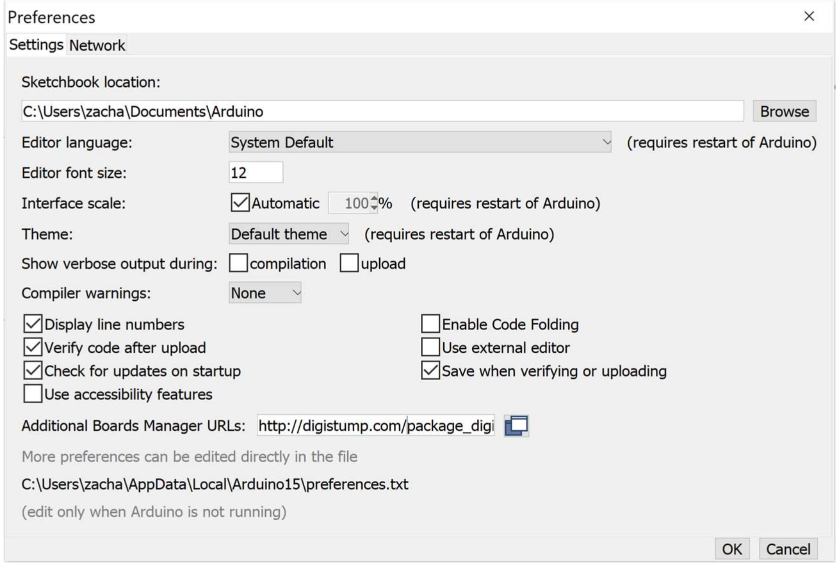
Driver Name	Status
✓ Digistump LLC (usbser) ...	Ready to use
✓ libusb-win32 Digispark B...	Ready to use
✓ libusb-win32 DigiUSB (0...	Ready to use

This should be the only thing that needs to be done to get the ducky acquainted with the computer. Now we can go on with setting up Arduino to work with the ducky.

Setting up Arduino

To configure Arduino, there are number of settings that need to be changed. Firstly, we need to add in the board manager so Arduino will be able to communicate with our digispark. To do this, under preferences in the Arduino application, add the following URL under 'Additional Board Manager URLs"

http://digistump.com/package_digistump_index.json



Preferences ×

Settings Network

Sketchbook location:

C:\Users\zacha\Documents\Arduino Browse

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

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
☑ Verify code after upload
☑ Check for updates on startup
□ Use accessibility features
□ Enable Code Folding
□ Use external editor
☑ Save when verifying or uploading

Additional Boards Manager URLs: http://digistump.com/package_digi

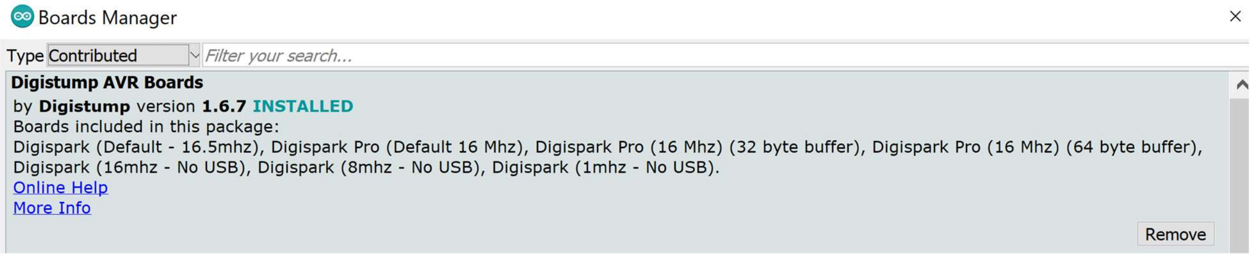
More preferences can be edited directly in the file

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

(edit only when Arduino is not running)

OK Cancel

Then, go to tools, Board, Board Manager, change the option at the top to 'contributed', then select 'Digistump AVR Boards' and click install.



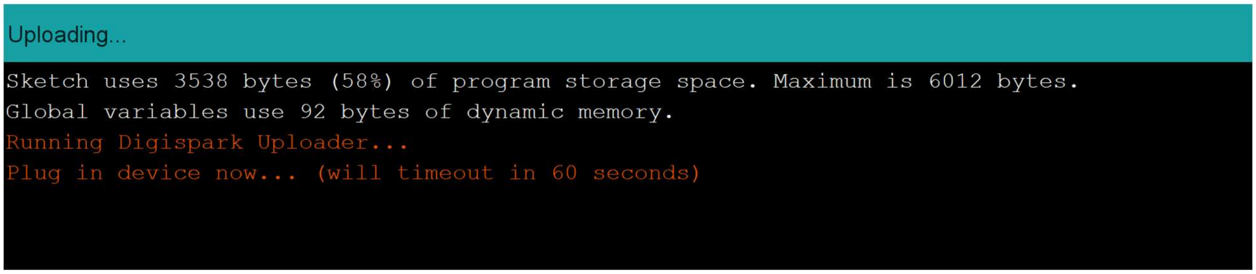
Boards Manager ×

Type Contributed Filter your search...

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

Remove

Now the Digispark microcontroller that we have is able to have Arduino sketches downloaded onto it. Once our sketch is done, we click upload, plug the ducky it, and once it connects the script we wrote will be automatically downloaded onto the ducky. If the ducky already has code on it, it will wait about five seconds before executing its script to wait to see if it is trying to have scripts put onto it.



```
Sketch uses 3538 bytes (58%) of program storage space. Maximum is 6012 bytes.
Global variables use 92 bytes of dynamic memory.
Running Digispark Uploader...
Plug in device now... (will timeout in 60 seconds)
```

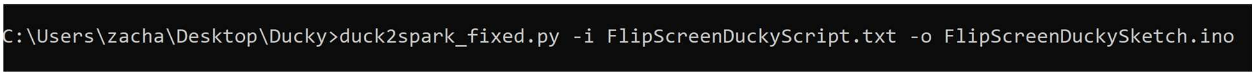
Screenshot of the uploading process

Converting Ducky Scripts

To get our ducky scripts to work with Arduino and on our physical ducky, they first need to be converted into Arduino sketches. Arduino natively, of course, does not know how to interpret the ducky language, but a python program called duck2spark.py from the following GitHub repository is able to convert the ducky script into code for the Arduino application.

https://github.com/mame82/duck2spark

The python program is run using the command line, and it is very easy to use. the '-i' option is used for the ducky script text input file and the '-o' option is used for the output file, which should have the .ino file extension for Arduino sketches. In the below screenshot it is shown how this program is used.



```
C:\Users\zacha\Desktop\Ducky>duck2spark_fixed.py -i FlipScreenDuckyScript.txt -o FlipScreenDuckySketch.ino
```

Use of the duck2spark.py file. Note the 'fixed' marker on the file used in this screenshot, as the original had a small but unpatched issue. The small fix used here is located at the end of this document.

Conclusion

Overall, the process for programming our ducky is quite simple. First, a script is written in the ducky script language in a basic text file. Then, a python program is used to convert script into an Arduino sketch. Finally, assuming the ducky has been set up to be programmed on the given computer, it is as simple as using the Arduino application to download the program onto the ducky. Now, the ducky will automatically run that given code when ever it is plugged into any computer.

Resources

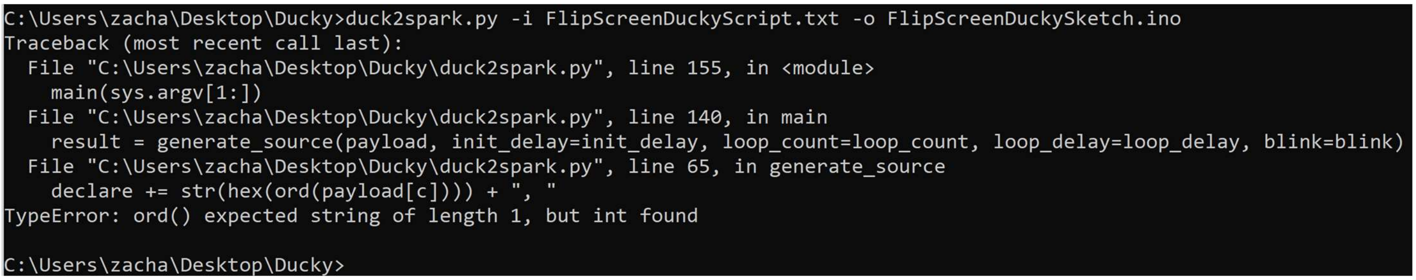
https://medium.com/hackernoon/low-cost-usb-rubber-ducky-pen-test-tool-for-3-using-digispark-and-duck2spark-5d59afc1910

https://github.com/mame82/duck2spark

https://www.youtube.com/watch?v=A3cB9BDE6XM&ab_channel=NullByte

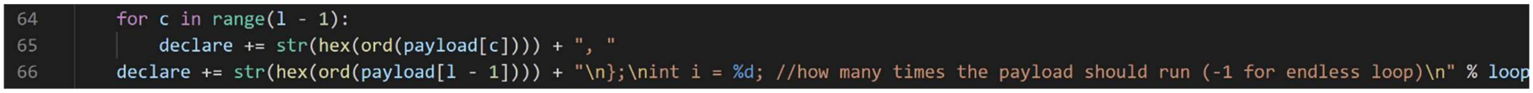
Duck2Spark.py Fix

The original duck2spark.py file from the GitHub repository gives the following error when used:



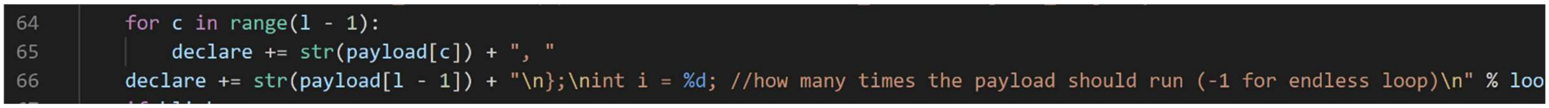
```
C:\Users\zacha\Desktop\Ducky>duck2spark.py -i FlipScreenDuckyScript.txt -o FlipScreenDuckySketch.ino
Traceback (most recent call last):
  File "C:\Users\zacha\Desktop\Ducky\duck2spark.py", line 155, in <module>
    main(sys.argv[1:])
  File "C:\Users\zacha\Desktop\Ducky\duck2spark.py", line 140, in main
    result = generate_source(payload, init_delay=init_delay, loop_count=loop_count, loop_delay=loop_delay, blink=blink)
  File "C:\Users\zacha\Desktop\Ducky\duck2spark.py", line 65, in generate_source
    declare += str(hex(ord(payload[c]))) + ", "
TypeError: ord() expected string of length 1, but int found

C:\Users\zacha\Desktop\Ducky>
```

To fix this issue, these lines:



```
64      for c in range(l - 1):
65          declare += str(hex(ord(payload[c]))) + ", "
66      declare += str(hex(ord(payload[l - 1]))) + "\n};\nint i = %d; //how many times the payload should run (-1 for endless loop)\n" % loop
```

should be changed to this:



```
64      for c in range(l - 1):
65          declare += str(payload[c]) + ", "
66      declare += str(payload[l - 1]) + "\n};\nint i = %d; //how many times the payload should run (-1 for endless loop)\n" % loo
```