```bash
#!/bin/bash
#
#                                    __      __  __
#                                   /  |    /  |/  |
#     ______    ______          __ _██ |_   ██/ ██ |
#    /      \  /      \  ______/  |/  ██   |   /  |██ |
#   /██████  |/██████  |/      |██ |██████/    ██ |██ |
#   ██ |  ██ |██ |  ██ |██████/ ██ |  ██ | __  ██ |██ |
#   ██ |__██ |██ \__██ |        ██ \__██ |/  | ██ |██ |
#   ██    ██/ ██    ██/         ██    ██/ ██  ██/ ██ |██ |
#   ███████/   ██████/           ██████/   ████/  ██/ ██/
#   ██ |
#   ██ |
#   ██/                              https://po-util.com
#
# Particle Offline Utility: A handy script for installing and using the Particle
# Toolchain on Ubuntu-based distros and OSX. This script downloads and installs:
# dfu-util, nodejs, gcc-arm-embedded, particle-cli, and the Particle Firmware
# source code.
# Read more at https://github.com/nrobinson2000/po-util
# ACII Art Generated from: http://www.patorjk.com/software/taag
#
# Helpful Tables: (*Please* update these lists if you make any modifications!)
#
    #---SUBCOMMAND------LINE#-#       #--GITHUB CONTRIBUTORS-#
    #-  help            177  -#       #- @nrobinson2000     -#
    #-  install         278  -#       #- @mrmowgli          -#
    #-  init            389  -#       #- @GeertWille        -#
    #-  serial          417  -#       #-                    -#
    #-  dfu-open        431  -#       #-                    -#
    #-  dfu-close       438  -#       #-                    -#
    #-  update          445  -#       #-                    -#
    #-  dfu             493  -#       #-                    -#
    #-  upgrade / patch 504  -#       #-                    -#
    #-  clean           524  -#       #-                    -#
    #-  ota             539  -#       #-                    -#
    #-  build           550  -#       #-                    -#
    #-  debug-build     558  -#       #-                    -#
    #-  flash           566  -#       #-                    -#
    #------------------------#       #----------------------#

# Helper functions
function pause(){
  read -rp "$*"
}

blue_echo() {
  echo "$(tput setaf 6)$(tput bold)$MESSAGE$(tput sgr0)"
}

green_echo() {
  echo "$(tput setaf 2)$(tput bold)$MESSAGE$(tput sgr0)"
}

red_echo() {
  echo "$(tput setaf 1)$(tput bold)$MESSAGE$(tput sgr0)"
}

choose_directory()
{
  if [ "$3" != "" ];
  then
    if [ -d "$3" ];
    then
      FIRMWAREDIR="$3"

      if [ -d "$CWD/$FIRMWAREDIR/firmware" ];  # If firmwaredir is not found relative to CWD, u
se absolute path instead.
      then
        FIRMWAREDIR="$CWD/$FIRMWAREDIR/firmware"
      else
        if [ -d "$FIRMWAREDIR/firmware" ]; # Use absolute path / firmware
        then
          FIRMWAREDIR="$FIRMWAREDIR/firmware"
          echo "Found firmwaredir" > /dev/null # Continue
        fi
        if [ -d "$FIRMWAREDIR" ]; # Use absolute path
        then
          echo "Found firmwaredir" > /dev/null # Continue
        fi
      fi #  CLOSE: if [ -d "$CWD/$FIRMWAREDIR/firmware" ]

    # Remove '/' from end of string
    case "$FIRMWAREDIR" in
      */)
       #"has slash"
       FIRMWAREDIR="${FIRMWAREDIR%?}"
       ;;
     *)
       echo "doesn't have a slash" > /dev/null
       ;;
    esac

    if [ "$3" == "." ];
    then
      FIRMWAREDIR="$CWD"
    fi
    else # of if [ -d "$3" ];

      MESSAGE="Firmware directory not found.
Please run \"po init\" to setup this repository or choose a valid directory." ; red_echo ; exi
t
    fi # CLOSE: if [ -d "$3" ];

  else # of if [ "$3" != "" ];

    if [ -d firmware ];
      then
        FIRMWAREDIR="$CWD/firmware"
    else
        MESSAGE="Firmware directory not found.
Please run \"po init\" to setup this repository or cd to a valid directory." ; red_echo ; exit
    fi
  fi
}

function find_bin() #Like choose_directory but for .bin files
{
  if [ "$1" != "" ];
  then
    case "$1" in
      */)
       #"has slash"
       FIRMWAREBIN="${1%?}"
       ;;
     *)
       echo "doesn't have a slash" > /dev/null
     FIRMWAREBIN="$1"
       ;;
    esac
      if [ -f "$CWD/$FIRMWAREBIN/bin/firmware.bin" ]; # If .bin file is not found relative to C
WD, use absolute path instead.
      then
      FIRMWAREBIN="$CWD/$FIRMWAREBIN/bin/firmware.bin"
  else
    if [ -f "$CWD/bin/firmware.bin" ];
    then
      FIRMWAREBIN="$CWD/bin/firmware.bin"
    else
      if [ -f "$CWD/firmware.bin" ];
      then
        FIRMWAREBIN="$CWD/firmware.bin"
      else
        if [ -f "$1" ];
        then
          FIRMWAREBIN="$1"
        else
          if [ -f "$CWD/$1" ];
          then
            FIRMWAREBIN="$CWD/$1"
          else
            MESSAGE="Firmware not found." ; red_echo
          exit
          fi
        fi
      fi
    fi
  fi
else
  FIRMWAREBIN="$CWD/bin/firmware.bin"
fi
echo "$FIRMWAREBIN"
}

build_message() {
  cd "$FIRMWAREDIR"/.. || exit
  BINARYDIR="$(pwd)/bin"
  MESSAGE="Binary saved to $BINARYDIR/firmware.bin" ; green_echo
  exit
}

dfu_open()
{
  stty "$STTYF" "$MODEM" "$DFUBAUDRATE"
}

# End of helper functions



if [ "$1" == "" ]; # Print help
then
MESSAGE="
                                    __      __  __
                                   /  |    /  |/  |
      ______    ______          __ _██ |_   ██/ ██ |
     /      \  /      \  ______/  |/  ██   |   /  |██ |
    /██████  |/██████  |/      |██ |██████/    ██ |██ |
    ██ |  ██ |██ |  ██ |██████/ ██ |  ██ | __  ██ |██ |
    ██ |__██ |██ \__██ |        ██ \__██ |/  | ██ |██ |
    ██    ██/ ██    ██/         ██    ██/ ██  ██/ ██ |██ |
    ███████/   ██████/           ██████/   ████/  ██/ ██/
    ██ |
    ██ |
    ██/                              http://po-util.com

"
blue_echo
echo "Copyright (GPL) 2016  Nathan Robinson

Usage: po DEVICE_TYPE COMMAND DEVICE_NAME
       po DFU_COMMAND
       po install [full_install_path]

Commands:
  install      Download all of the tools needed for development.
               Requires sudo. You can also re-install with this command.
               You can optionally install to an alternate location by
               specifying [full_install_path].
               Ex.:
                   po install ~/particle

               By default, Firmware is installed in ~/github.

  build        Compile code in \"firmware\" subdirectory
  flash        Compile code and flash to device using dfu-util

               NOTE: You can supply another argument to \"build\" and \"flash\"
               to specify which firmware directory to compile.
               Ex.:
                   po photon flash photon-firmware/

  clean        Refresh all code (Run after switching device or directory)
  init         Initialize a new po-util project
  update       Update Particle firmware, particle-cli and po-util
  upgrade      Upgrade system firmware on device
  ota          Upload code Over The Air using particle-cli
  serial       Monitor a device's serial output (Close with CRTL-A +D)

DFU Commands:
  dfu         Quickly flash pre-compiled code
  dfu-open    Put device into DFU mode
  dfu-close   Get device out of DFU mode
" && exit
fi

# Configuration file is created at "~/.po"
SETTINGS=~/.po
BASE_FIRMWARE=~/github # These
BRANCH="latest"        # can
BINDIR=~/bin           # be
DFUBAUDRATE=19200      # changed in the "~/.po" file.

CWD="$(pwd)" # Global Current Working Directory variable

# Mac OSX uses lowercase f for stty command
if [ "$(uname -s)" == "Darwin" ];
  then
    OS="Darwin"
    STTYF="-f"
    MODEM="$(ls -1 /dev/cu.* | grep -vi bluetooth | tail -1)"
  else
    OS="Linux"
    STTYF="-F"
    MODEM="$(ls -1 /dev/* | grep "ttyACM" | tail -1)"

    #THIS COULD BE IMPROVED!
    GCC_ARM_VER=gcc-arm-none-eabi-4_9-2015q3 # Updated to 4.9
    export GCC_ARM_PATH=$BINDIR/gcc-arm-embedded/$GCC_ARM_VER/bin/
    export PATH=$GCC_ARM_PATH:$PATH
fi

# Check if we have a saved settings file.  If not, create it.
if [ ! -f $SETTINGS ]
then
  echo BASE_FIRMWARE="$BASE_FIRMWARE" >> $SETTINGS
  echo BRANCH="latest" >> $SETTINGS
  echo PARTICLE_DEVELOP="1" >> $SETTINGS
  echo BINDIR="$BINDIR" >> $SETTINGS
  echo DFUBAUDRATE="$DFUBAUDRATE" >> $SETTINGS

  if [ $OS == "Linux" ];
    then
      echo export GCC_ARM_PATH=$GCC_ARM_PATH >> $SETTINGS
  fi
fi

# Import our overrides from the ~/.po file.
source "$SETTINGS"

# GCC path for linux make utility
if [ $GCC_ARM_PATH ]; then GCC_MAKE=GCC_ARM_PATH=$GCC_ARM_PATH ; fi

if [ "$1" == "install" ]; # Install
then

  if [ "$CWD" != "$HOME" ];
  then
    cp po-util.sh ~/po-util.sh #Replace ~/po-util.sh with one in current directory.
  fi

  if [ -f ~/.bash_profile ]; #Create .bash_profile
  then
    MESSAGE=".bash_profile present." ; green_echo
  else
    MESSAGE="No .bash_profile present. Installing.." ; red_echo
    echo "
    if [ -f ~/.bashrc ]; then
        . ~/.bashrc
    fi" >> ~/.bash_profile
  fi

  if [ -f ~/.bashrc ];  #Add po alias to .bashrc
  then
    MESSAGE=".bashrc present." ; green_echo
    if grep "po-util.sh" ~/.bashrc ;
    then
      MESSAGE="po alias already in place." ; green_echo
    else
      MESSAGE="no po alias.  Installing..." ; red_echo
      echo 'alias po="~/po-util.sh"' >> ~/.bashrc
      echo 'alias p="particle"' >> ~/.bashrc  #Also add 'p' alias for 'particle'
    fi
  else
    MESSAGE="No .bashrc present.  Installing..." ; red_echo
    echo 'alias po="~/po-util.sh"' >> ~/.bashrc
  fi

  # Check to see if we need to override the install directory.
  if [ "$2" ] && [ "$2" != $BASE_FIRMWARE ]
  then
    BASE_FIRMWARE="$2"
    echo BASE_FIRMWARE="$BASE_FIRMWARE" >  $SETTINGS
  fi

  [ -d "$BASE_FIRMWARE" ] || mkdir -p "$BASE_FIRMWARE"  # If BASE_FIRMWARE does not exist, crea
te it

  # clone firmware repository
  cd "$BASE_FIRMWARE" || exit
  MESSAGE="Installing Particle firmware from Github..." ; blue_echo
  git clone https://github.com/spark/firmware.git

  if [ "$OS" == "Linux" ]; # Linux installation steps
  then
    cd "$BASE_FIRMWARE" || exit
    # Install dependencies
    MESSAGE="Installing ARM toolchain and dependencies locally in $BINDIR/gcc-arm-embedded/...
" ; blue_echo
    mkdir -p $BINDIR/gcc-arm-embedded && cd "$_" || exit
    wget https://launchpad.net/gcc-arm-embedded/4.9/4.9-2015-q3-update/+download/gcc-arm-none-
eabi-4_9-2015q3-20150921-linux.tar.bz2 #Update to v4.9
    tar xjf gcc-arm-none-eabi-*-linux.tar.bz2

    curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
    sudo apt-get install -y nodejs python-software-properties python g++ make build-essential
libusb-1.0-0-dev libarchive-zip-perl screen

    # Install dfu-util
    MESSAGE="Installing dfu-util (requires sudo)..." ; blue_echo
    cd "$BASE_FIRMWARE" || exit
    git clone git://git.code.sf.net/p/dfu-util/dfu-util
    cd dfu-util || exit
    git pull
    ./autogen.sh
    ./configure
    make
    sudo make install
    cd ..

    # Install particle-cli
    MESSAGE="Installing particle-cli..." ; blue_echo
    sudo npm install -g node-pre-gyp npm particle-cli

    # Install udev rules file
    MESSAGE="Installing udev rule (requires sudo) ..." ; blue_echo
    curl -fsSLO https://raw.githubusercontent.com/nrobinson2000/po-util/master/60-po-util.rule
s
    sudo mv 60-po-util.rules /etc/udev/rules.d/60-po-util.rules

  fi # CLOSE: "$OS" == "Linux"

  if [ "$OS" == "Darwin" ]; # Mac installation steps
  then
    # Install Homebrew
    MESSAGE="Installing Brew..." ; blue_echo
    /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/i
nstall)"
    brew tap PX4/homebrew-px4
    brew update

    # Install ARM toolchain
    MESSAGE="Installing ARM toolchain..." ; blue_echo
    brew install gcc-arm-none-eabi-49 dfu-util

    # Install Nodejs version 6.2.2
    MESSAGE="Installing nodejs..." ; blue_echo
    curl -fsSLO https://nodejs.org/dist/v6.2.2/node-v6.2.2.pkg
    sudo installer -pkg node-*.pkg -target /
    rm -rf node-*.pkg
    # Install particle-cli
    MESSAGE="Installing particle-cli..." ; blue_echo
    sudo npm install -g node-pre-gyp npm serialport particle-cli
  fi # CLOSE: "$OS" == "Darwin"

  cd "$CWD" && MESSAGE="Sucessfully Installed!" ; green_echo && exit
fi

# Create our project files
if [ "$1" == "init" ];
then
  if [ -d firmware ];
  then
    MESSAGE="Directory is already Initialized!" ; green_echo
    exit
  fi

  mkdir firmware/
  echo "#include \"application.h\"

  void setup() // Put setup code here to run once
  {

  }

  void loop() // Put code here to loop forever
  {

  }" > firmware/main.cpp
  cp *.cpp firmware/
  cp *.h firmware/
  ls firmware/ | grep -v "particle.include" | cat > firmware/particle.include
  MESSAGE="Copied c++ files into firmware directory.  Setup complete." ; green_echo
  exit
fi

# Open serial monitor for device
if [ "$1" == "serial" ];
then
  if [ "$MODEM" == "" ]; # Don't run screen if device is not connected
  then
    MESSAGE="No device connected!" red_echo ; exit
  else
    screen -S particle "$MODEM"
    screen -S particle -X quit && exit || MESSAGE="If \"po serial\" is putting device into DFU
 mode, power off device, removing battery for Electron, and run \"po serial\" several times.
This bug will hopefully be fixed in a later release." && blue_echo
  fi
  exit
fi

# Put device into DFU mode
if [ "$1" == "dfu-open" ];
then
  dfu_open "$@"
  exit
fi

# Get device out of DFU mode
if [ "$1" == "dfu-close" ];
then
  dfu-util -d 2b04:D006 -a 0 -i 0 -s 0x080A0000:leave -D /dev/null
  exit
fi
```

```
# Update po-util
if [ "$1" == "update" ];
then
  MESSAGE="Updating firmware..." ; blue_echo
  cd "$BASE_FIRMWARE"/firmware || exit
  git checkout $BRANCH
  git pull
  MESSAGE="Updating particle-cli..." ; blue_echo
  sudo npm update -g particle-cli
  MESSAGE="Updating po-util.." ; blue_echo
  # curl -fsS https://raw.githubusercontent.com/nrobinson2000/po-util/po-util.com/update | bas
h ##nrobinson2000: People don't like piping curl to bash
  rm ~/po-util.sh
  curl -fsSLo ~/po-util.sh https://raw.githubusercontent.com/nrobinson2000/po-util/master/po-u
til.sh
  chmod +x ~/po-util.sh
  exit
fi

# Make sure we are using photon, P1, or electron
if [ "$1" == "photon" ] || [ "$1" == "P1" ] || [ "$1" == "electron" ];
then
  MESSAGE="$1 selected." ; blue_echo
else
  MESSAGE="Please choose \"photon\", \"P1\" or \"electron\", or choose a proper command." ; re
d_echo ; exit
fi

cd "$BASE_FIRMWARE"/firmware || exit

if [ "$1" == "photon" ];
then
  git checkout $BRANCH > /dev/null
  DFU_ADDRESS1="2b04:D006"
  DFU_ADDRESS2="0x080A0000"
fi

if [ "$1" == "P1" ];
then
  git checkout $BRANCH > /dev/null
  DFU_ADDRESS1="2b04:D008"
  DFU_ADDRESS2="0x080A0000"
fi

if [ "$1" == "electron" ];
then
  git checkout $BRANCH > /dev/null
  DFU_ADDRESS1="2b04:d00a"
  DFU_ADDRESS2="0x08080000"
fi

# Flash already compiled binary
if [ "$2" == "dfu" ];
then
  dfu_open "$@"
  sleep 1
  find_bin "$3"
  echo "$FIRMWAREBIN"
  dfu-util -d "$DFU_ADDRESS1" -a 0 -i 0 -s "$DFU_ADDRESS2":leave -D "$FIRMWAREBIN" || ( MESSAGE
="Device not found." ; red_echo )
  exit
fi

#Upgrade our firmware on device
if [ "$2" == "upgrade" ] || [ "$2" == "patch" ];
```

```
then
  pause "Connect your device and put into DFU mode. Press [ENTER] to continue..."
  cd "$CWD" || exit
  sed "2s/.*/START_DFU_FLASHER_SERIAL_SPEED=$DFUBAUDRATE/" "$BASE_FIRMWARE/"firmware/build/mod
ule-defaults.mk > temp.particle
  rm -f "$BASE_FIRMWARE"/firmware/build/module-defaults.mk
  mv temp.particle "$BASE_FIRMWARE"/firmware/build/module-defaults.mk

  cd "$BASE_FIRMWARE/firmware/modules/$1/system-part1" || exit
  make clean all PLATFORM="$1" "$GCC_MAKE" program-dfu

  cd "$BASE_FIRMWARE/firmware/modules/$1/system-part2" || exit
  make clean all PLATFORM="$1" $GCC_MAKE program-dfu
  cd "$BASE_FIRMWARE/firmware" && git stash || exit
  sleep 1
  dfu-util -d $DFU_ADDRESS1 -a 0 -i 0 -s $DFU_ADDRESS2:leave -D /dev/null
  exit
fi

# Clean firmware directory
if [ "$2" == "clean" ];
then
    cd "$CWD" || exit
    choose_directory "$@"
    cd "$BASE_FIRMWARE"/firmware || exit
    make clean
    cd "$CWD" || exit
    if [ "$FIRMWAREDIR/../bin" != "$HOME/bin" ];
    then
      rm -rf "$FIRMWAREDIR/../bin"
    fi
  exit
fi

# Flash binary over the air
if [ "$2" == "ota" ];
then
  if [ "$3" == "" ];
  then
    MESSAGE="Please specify which device to flash ota." ; red_echo ; exit
  fi
  find_bin "$4"
  particle flash "$3" "$FIRMWAREBIN" || ( MESSAGE="Try using \"particle flash\" if you are havi
ng issues." ; red_echo )
  exit
fi

if [ "$2" == "build" ];
then
  cd "$CWD" || exit
    choose_directory "$@"
    make all -s -C "$BASE_FIRMWARE/"firmware APPDIR="$FIRMWAREDIR" TARGET_DIR="$FIRMWAREDIR/../
bin" PLATFORM="$1" $GCC_MAKE  || exit
    build_message "$@"
fi

if [ "$2" == "debug-build" ];
then
    cd "$CWD" || exit
    choose_directory "$@"
    make all -s -C "$BASE_FIRMWARE/"firmware APPDIR="$FIRMWAREDIR" TARGET_DIR="$FIRMWAREDIR/../
bin" PLATFORM="$1" DEBUG_BUILD="y" $GCC_MAKE  || exit
    build_message "$@"
fi
```

```
if [ "$2" == "flash" ];
then
    cd "$CWD" || exit
    choose_directory "$@"
    dfu_open "$@"
    make all -s -C "$BASE_FIRMWARE/"firmware APPDIR="$FIRMWAREDIR" TARGET_DIR="$FIRMWAREDIR/../
bin" PLATFORM="$1"  $GCC_MAKE  || exit
    dfu-util -d "$DFU_ADDRESS1" -a 0 -i 0 -s "$DFU_ADDRESS2":leave -D "$FIRMWAREDIR/../bin/firm
ware.bin"
    exit
fi

# If an improper command is chosen:
MESSAGE="Please choose a proper command." ; red_echo
```