PROGRAMMING SWITCH BOARD

Welcome

If you are here its because you are a curious person, willing to improve or customize what doesn't fit you, and that is GREAT!

Open source is what makes this possible. In any other magic trick, if you don't like it, you don't have much choice than using it as is. But with open source now you can change just a tiny part, or the whole working of this trick. Its up to you, and you have all the tools available and explained so you can do it by your own.

Isn't this amazing? I hope that slowly, but steadily you will fell in love with open source ❤

Warning

The steps explained in this document could be dangerous for your Switch Board if you don't know what you are doing.

Any procedure explained in this document is not covered by the warranty.

Not following the steps correctly could cause your Switch Board not to work again. Even following exactly the same steps, could make your Switch Board not work again.

Please understand this are the steps we use to program the Switch Boards, but may not work correctly for you. You may need to adapt the information provided.

Basics are not covered in this document. Programming, microcontrollers, Arduino, protoboard wiring, etc. are supposed to be known. If you don't know these basics you shouldn't be trying to reprogram your Switch Board.

Al information is provided "as is", with no guarantee that will work, and no support is provided.

If you are willing to proceed it is strongly advised that you fully understand what you are doing, and it's preferred to first make all the test on an Arduino board, and after debugging the changes then switch to the Switch Board.

Tools

We will need some software and hardware tools:

Software

- Arduino IDE (1.8.13 or newer)
  https://www.arduino.cc/en/software
- ATTinyCore (1.5.2 or newer)
  https://github.com/SpenceKonde/ATTinyCore
- AVRDUDE (included in the same directory as this document)
  https://www.nongnu.org/avrdude/
- Switch Board source code
  https://gitlab.com/ideaalab/switch-board

Hardware

- High voltage fuse eraser
  https://www.electronics-lab.com/recover-bricked-attiny-using-arduino-as-high-voltage-programmer/
- Arduino as ISP
  https://www.arduino.cc/en/Tutorial/BuiltInExamples/ArduinoISP
- Alligator clips to connect these tools to Switch Board

Overview

Inside the Switch Board there is an Atmel ATTiny84A microcontroller. This is placed bellow the battery holder.

This are the steps needed to program the new firmware on your Switch Board

1. Program and debug on an Arduino board
2. Compile your code for Switch Board and generate the HEX file
3. Reset the fuses of your Switch Board
4. Upload the new HEX to your Switch Board
5. Reprogram the fuses to your Switch Board
6. Done!

Before we begin

Go to you file explorer and create a folder named “sb” in the main drive of your operating system. It should look like:

```
C:\sb
```

This document is inside a “Programming” folder. Please move everything inside this folder, including this document to the “sb” folder you just created.

This will make easier to work in the command line later.

Fuses

Fuses are a part of the microcontroller where it says how to behave.

In the ATTiny84A used on Switch Board we have to disable RESET pin to prevent accidental resets caused by static discharges. So, after uploading a new firmware to the board we disable RESET pin in the fuses.

Problem is that when RESET pin is disabled, is no longer possible to program the boards with “low voltage” (5v). And our programmer (Arduino as ISP) can only program with low voltage.

In order for the microcontroller to allow low voltage programming we have to change the fuses back to default. Then we can upload the new program, and after that burn (write) the fuses again to disable RESET pin.

Program and debug on an Arduino board

First step is to get familiar with the source code. Open the Arduino IDE and see how it works, and why it is programed that way.

Then upload the code with no modifications to an Arduino board and see if you can make the Switch Board firmware run on your Arduino. To do this:

1. Wire your Arduino board to 4 switches and 4 LEDs (with resistor!). Use the pins you like for this.
2. On the Arduino IDE, in the “a_pinout” tab comment the second line (SWITCH_BOARD), and uncomment the fourth (CUSTOM).
3. Now you have to define the pins used on your Arduino board. Just make them match to the ones you chose. You will find how to define these pins between lines 95 and 130.
4. You can uncomment the DEBUG line to see the status of every step in the firmware execution on the serial monitor. It is very useful!
5. Connect the board to the computer.
6. Choose the appropriate board in the “Tools” menu from the IDE and select the correct port.
7. Press the compile button to check if there is no error.
8. Press the upload button.

If everything was fine, you should now have the unmodified Switch Board firmware uploaded to your Arduino board! Check that everything works as expected before proceeding to the next step.

After this you can start to make changes to the code.

Make small changes incrementally, so if at any point the firmware stops working, you can undo a few lines of code and not a whole routine!

Once you have all the modifications working as you like make some tests to check that every other part of the code still works as expected. Like for example, check that if you change a part of an effect, then the next effect behaves also as expected.

Generate HEX file

After you have your code working on an Arduino, it is time to move it to you Switch Board.

Remember to switch back the board to “SWITCH_BOARD” in the “a_pinout” tab.

Now you will need to get the HEX file so you can upload it to the board.

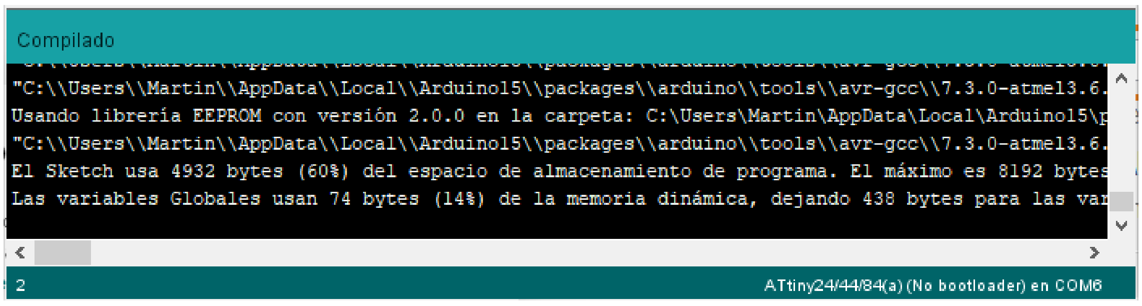
First you need to be sure you have ATTinyCore installed in the Arduino IDE Board Manager (https://support.arduino.cc/hc/en-us/articles/360016119519-How-to-add-boards-in-the-board-manager).

Use these options:

Board:	ATtiny24/44/84(a) (no bootloader)
Chip:	ATtiny84(a)
Clock:	1Mhz (internal)
Pin mapping:	Clockwise (like damellis core)
LTO:	Enabled
tinyNeopixel port:	Port A
millis()/micros:	Enabled
Save EEPROM:	EEPROM Retained
B.O.D. Level:	B.O.D. Disabled

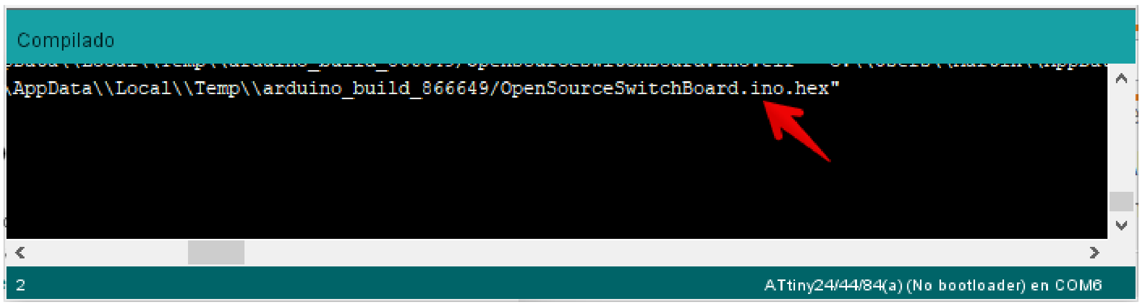
In order to find the HEX file, we need to enable the verbose output on compilation. This option is found in File>Settings from Arduino IDE.

Now press the compile button and if everything goes correctly the HEX file will be placed in a temp folder. This folder is written in the command line output:



```
"C:\\Users\\Martin\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.
Usando librería EEPROM con versión 2.0.0 en la carpeta: C:\Users\Martin\AppData\Local\Arduino15\p
"C:\\Users\\Martin\\AppData\\Local\\Arduino15\\packages\\arduino\\tools\\avr-gcc\\7.3.0-atmel3.6.
El Sketch usa 4932 bytes (60%) del espacio de almacenamiento de programa. El máximo es 8192 bytes
Las variables Globales usan 74 bytes (14%) de la memoria dinámica, dejando 438 bytes para las var
```

You may have to move the scroll bars to find it:



```
AppData\\Local\\Temp\\arduino_build_866649/OpenSourceSwitchBoard.ino.hex"
```

Go to that directory and grab the HEX file and place it in the “C:\sb” folder you created.

Connecting hardware to your Switch Board

The Arduino as ISP and the HV fuse eraser will need to interface with the Switch Board.

To do this you will need to use some alligator clips to connect to the microcontroller. Check the schematic attached in the PCB folder for further details.

The RESET pin of the microcontroller is “hidden”. If you watch the battery holder closely, you will notice a tiny hole between positive and negative metal tabs. Just beneath the hole is a copper pad connecting to the RESET pin. You can make contact with it using a metal leg from an LED or any other thin and rigid metal pin that fit in the hole. Press it tightly while uploading the firmware.

Connections when programming and when resetting fuses are not the same!

Switch Board	**Attiny84A (QFN20)**	**Arduino as ISP**	**HV fuse eraser**
Positive	VCC (PIN 9)	Positive 5v	Positive 5v
Negative	GND (Pin 8)	Negative 0v	Negative 0v
Bulb 1	PA4 (PIN 1)	SCL	SDO
Bulb 2	PA5 (PIN 20)	MISO	SII
Bulb 3	PA6 (PIN 16)	MOSI	SDI
Bulb 4	PB0 (PIN 11)	NOT USED	SCI
Reset (bellow battery holder)	PB3 (PIN 13)	RESET	RESET
Switch 1	PA3 (PIN 2)	NOT USED	NOT USED
Switch 2	PA2 (PIN 3)	NOT USED	NOT USED
Switch 3	PA1 (PIN 4)	NOT USED	NOT USED
Switch 4	PA0 (PIN 5)	NOT USED	NOT USED

Reset the fuses

Switch Board has some fuses that prevents serial programming.

Use any high voltage fuse eraser to reset the fuses from the microcontroller. Follow the instructions on how to do this on the website given.

There are many websites explaining how to do this. Its often called “unbrick”, because when the microcontroller does not allow programming, it’s called “bricked”. So, search for attiny “high voltage programmer”, “fuse reset”, “unbrick” and similar information. Any of this will do the same.

The circuit provided will basically be an Arduino and a high voltage source (12v) that is controlled with a transistor. The Arduino must be programmed with a code that will communicate with the Attiny84A of our Switch Board and reset the fuses back where you will be able to program at 5v.

A tip: instead of trying to generate 12v from the 5v from your Arduino, a step-up inverter or a 12v power supply, you can just use a small 23A battery. This are 12v batteries used in remote controls. They have very little current available, but for this purpose is enough. This is what we use.

Upload firmware to Switch Board

You have everything ready: the HEX file is in “C:\sb\OpenSourceSwitchBoard.hex” and your board has the fuses that allow serial programing, now you are ready to upload the firmware to the board.

1. Connect the Arduino as ISP to your Switch Board.
2. Open the command line (press the Windows button and type “cmd” and then ENTER).
3. Run the following instruction in the CMD
   ```
   C:\sb\avrdude -C"C:\sb\avrdude.conf" -v -pattiny84 -carduino -PCOM6 -b19200 -
   Uflash:w:"C:\sb\OpenSourceSwitchBoard.hex":i -Uefuse:w:0xff:m -Ulfuse:w:0x62:m -
   Uhfuse:w:0x57:m
   ```

- Change the blue part of the code to the port where you have your Arduino as ISP.
- Change the red part of the code to any other firmware you may want:
  - v1.1.1.hex is the latest firmware to date
  - fossat.hex is a modified firmware proposed by Sebastien Fossat with a variation in Effect 7, phase 2.
- The green part set the fuses back. If you don’t want to make your board “unprogrammable” again just remove this. It’s useful to test several firmware’s without needing to reset the fuses each time. Once you have your final firmware remember to program the board using those fuses.

Share

If you made a significant change to the firmware, you could share it with all other Switch Board users. Send the new source code with a detailed explanation of the mod, and if possible, a video performing this for better understanding.

Open source is sharing 😊