SHAKE DAT BUUTTI BUILDING INSTRUCTION V1.0 / V1.1

Niko Huttula

26.2.2018

Schematic and required materials

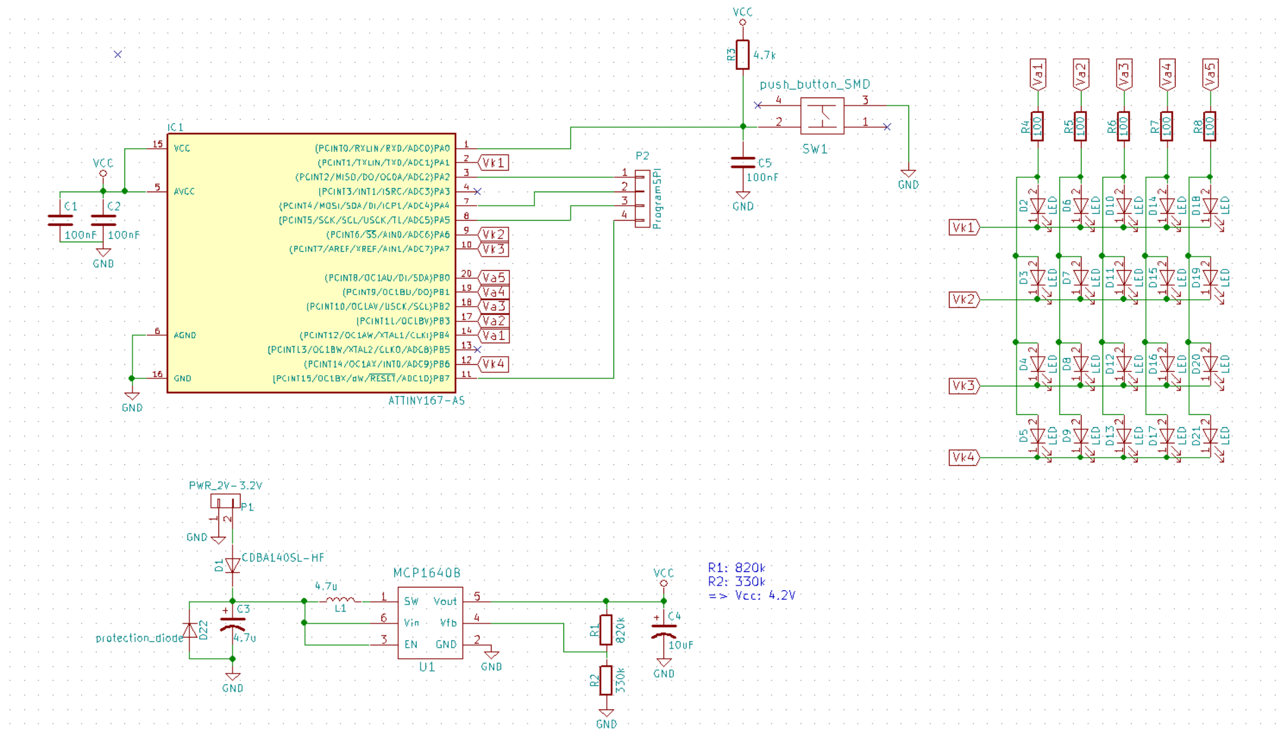

Figure 1. Schematic for SHAKE DAT Buutti LED-mark.

Component	Pcs	Info	Footprint
ATtiny167	1	MCU	SOIC-20 7.5x12.8 mm Pitch 1.27 mm
MCP1640B	1	Boost regulator	SOT23-6
LED Blue	4	Flat top head LEDs	3mm
LED White	16	2.6V Forward voltage	Pitch 2.54mm
Resistors	8	820kΩ, 330kΩ, 4.7kΩ, 5*100Ω	0805
Capacitors	5	10uF, 4.7uF, 3*100nF	1206
Coils	1	4.7uH	1210
Push Button	1	General tactile switch	SMD/Through
CONN	1	1x4 angled pin connector for programming	Pitch 2.54mm
General diode / Schottky	1	Schottky/general diode (optional protection)	DO-214AC
5.5V Zener	1	Overvoltage protection	DO-214AC

Table 1. Bill of materials.

Building instructions for V1.0

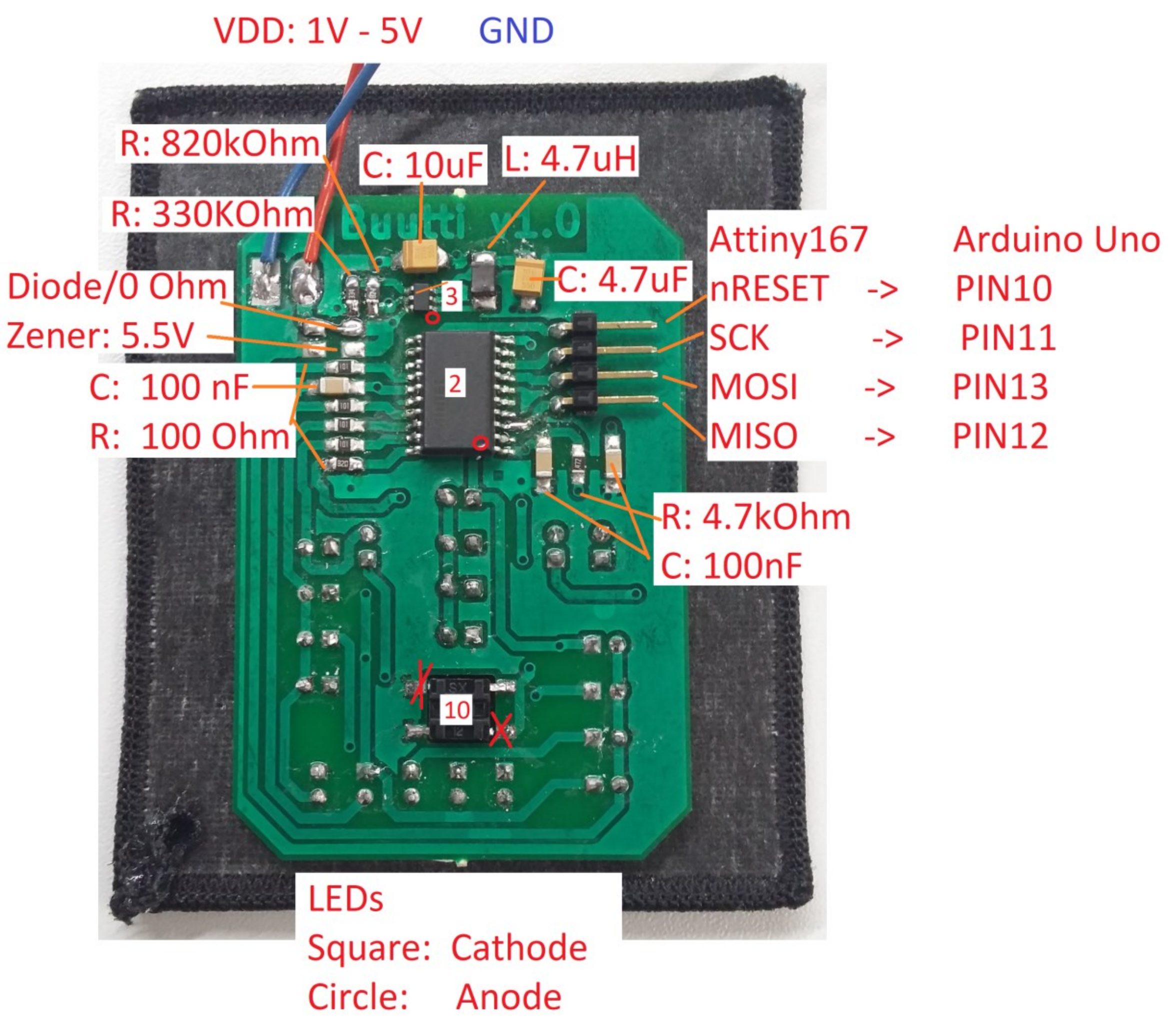

Figure 2. Component placing and fixes for push button

(Button fix is needed only in V1.0)

1. Solder coil, Capacitors and resistors
2. Solder ATtiny167 (notice the mark!)
3. Solder MCP1640B (notice the mark!)
4. Solder angled male pin header
5. Cut a slice from legs of LEDs to be sharp (Keep the other leg longer)
6. Attach all LEDs (Don't solder yet!)
    - Notch at "right" side mean cathode (-> square box).
    - Can be tested using voltage generator because current can flow only from higher potential. => i.e. from anode to cathode (Limit current to minimum or LED might burn!)

7. Solder all LEDs and cut remaining legs
8. Push each LED from other side and reflow soldered area
9. Cut two legs from push button as shown in figure 2. (Only for V1.0)
10. Solder button as close to textile as possible
    - Check that button is still usable
11. Solder two wires for battery
    (1V – 5V input, lower voltage causes higher current consumption)
12. Test with LCR meter that there are no short circuits
    (Or else you might break something when powering up!)
13. When everything is checked up
    - Power up, be sure that red cable is VDD and blue is GND (square box) and that they are wired correctly! (Opposite voltage more likely breaks something!)
    - Measure that VDD at the 10uF capacitor is 4.2V
        - If the voltage is wrong or there are some smoke in the air;
            1. Re check everything and ask for help!
14. Congrats, now you are ready to jump programming part!

Modifications that can be made

You may change 820k and 330k Ohm resistors to differ VDD voltage so that it lowers from 4,2V to 3.3V. In this case current consumption is lower with lower input voltages

- $\boldsymbol{VDD} = (\frac{\mathbf{820k\Omega}}{\mathbf{330k\Omega}} + \mathbf{1}) * \mathbf{1.21}$ (VDD = 4.2V)
- In this case each LED uses approximately current as;

$$\boldsymbol{I_{LED}} = \frac{\boldsymbol{VDD-2.6V}}{\mathbf{100\Omega}} \qquad (I_{LED} = \sim 16mA)$$

- Resistor values can also be changed for more optimal current.
  E.g. 100Ohm -> 70 Ohm (VDD 3.3V and 10mA current)
- Notice that there are some drop out voltage at each pin when current increases

Current over one led should not exceed 20mA!

Compile C and upload to Attiny167

Linux users;

- Install gcc-avr, avr-libc and avrdude
    1. sudo apt-get install gcc-avr avr-libc avrdude
- Add ATtiny167 to avrdude (fix for not finding device)
    1. Copy text from “fix_for_avrdude_missing_attiny167.txt” file
    2. Sudo gedit /etc/avrdude.conf
    3. Scroll bottom of file and paste copied text.
    4. Save
- Wire Attiny167 to your programmer as stated in figure 2.
    1. Arduino Uno used as ISP programmer
- cd ~/Buutti_LEDMerkki/Buutti_src
    1. All the project files and Makefile can be found there
    2. Make fuse
        - Sets correct fuses for ATtiny167, need to be run only once
    3. Make install
        - Compiles and uploads your program to ATtiny167
    4. Make clean
        - Deletes all compiled files, not necessary but nice to have
    5. You can add more libraries by adding the *filename*.o at the end of “OBJS=main.o ctrl_functions.o” line (inside Makefile)
- If the kernel says something about STK500
    1. Check that you have wired RES pin of Arduino Uno to VDD (shorted)
    2. Check that VDD and GND are connected
    3. Check that SPI line has been wired accordingly
    4. Remove USB cable from PC and re attach
- With other compiling problems with gcc-avr and avrdude
    1. Wonder if you have installed all the programs correctly, try Google
- Other errors are more likely caused by bad C program

Be careful when you touch fuses or you might brick your MCU!

REQUIRES original ATMEL programmer or a bit hack!!

Windows users;

- Download and install Atmel Studio
- Create new project with "GCC C Executable Project"
- Name your project
- Set location to "~\Buutti_LEDMerkki\Buutti_src\AtmelStudio"
- Device family ATtiny
  - Find ATtiny167
  - Ok
- Remove existing main.c file from created Atmel Studio project
  - Delete
- Right click Solution Explorer
  - Add >> existing item (Same place where main.c was)
  - Move back to Buutti_src folder
  - Add files four files;
    - confs.h
    - ctrl_functions.c
    - functions.h
    - main.c
- Open new main.c file. The program should be visible
- Build solution (F7 or from top bar)
  - No error should show up
- Add your programmer (Alt+F7 or Project >> Properties)
  - Set "Selected debugger/programmer"
  - Requires original programmer from ATMEL (ATMEL-ICE, AVR Dragon, STK500, etc.)
    - Alternatively you may hack Arduino Uno to work ask ISP programmer by the instructions from http://www.instructables.com/id/Integrate-ArduinoISP-and-Atmel-Studio/
    - Or you could use avrdude externally from terminal with the hex file created from Atmel Studio (debug folder inside project folder)
    - Text editor, avr-gcc, avr-libc and avrdude can also be used as in linux. But the same Makefile will not more likely work

Be careful when you touch fuses or you might brick your MCU!

Programming in general

- Required pin configurations are inside **confs.h** file
  - Address for PORTA/B
  - Index for shifting correct pin HIGH/LOW (Va* and Vk*)
    - E.g. &PORTB |= (1<<Va1)   (Sets LED0 anode to high)
    - E.g. &PORTB &= ~(1<<Vak1)   (Sets LED0 cathode to low)
  - LED mapping
- Actual functions for controlling LEDs are inside **ctrl_functions.c** file
  - Initiation function for default PORT, sleep and interrupt settings
  - Macros to control ports
    - SET_LED_MATRIX(x,y);
    - SHUT_LED_MATRIX(x,y);
  - Default controlling function where decimal numbers are connected to correct led index
    - setPin(…);
  - Safe to use blink functions
    - blinkLED(…);
    - blinkLEDs(…);
  - Controlling functions for LED images
    - lightAll(…);
    - lightWaveAll(…);
    - lightWaveSingle(…);
    - lightWaveBlocks(…);
- Function declarations are inside **functions.h** file
- The actual program is inside **main.c** where functions are being
  - Interrupt vector
    - ISR(PCINT0_vect){…}
  - Main(void){…}

Always shut the previous LED before lighting new one, unless you know what you're doing! ATtiny167 might burn if there are too much current consumption at once.

INSTRUCTION FOR ORDERING NEW PCBs

Version V1.1 has already been designed.

- For ordering more PCBs, plot from Kicad;
    - F.Cu
    - B.Cu
    - F.Mask
    - B.Mask
    - Edge.Cuts
    - Drill file
- Send files to PCB manufacturer and ask for JEP and ~1,4mm PCB thickness