**LEGO nyáktervező tanfolyam**

A tanfolyam során egy STM32 mikrokontrollert tartalmazó fejlesztőkártyát fogunk megtervezni, majd a későbbi alkalmakon legyártani és programozni. Az előző félév tapasztalatai alapján nagyon sok idő elmegy az áramkör működésének magyarázásával, ezért most összeszedtem ide a lényeges (?) tudnivalókat. A tanfolyamon már csak rajzolni fogunk!

**Önnek is joga van tudni: mibű készül a Nucleo?**

Úgy vettem észre, hogy a körben néhányan az ARM/STM/Nucleo fogalmakkal nincsenek teljesen tisztában (nem piszkálódásból mondom; azért vagyunk itt, hogy tanuljunk), ezért szeretném ezt itt az elején tisztázni.

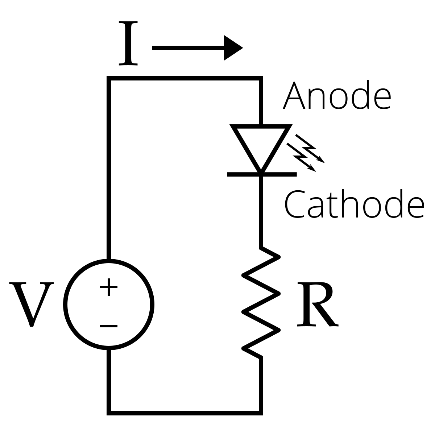
Szóval az ARM egy processzor-architektúra, aminek a terveit a tervezők eladják, hogy a pl. mikrokontroller tervezéssel fogalkozó emberkék integrálhassák az eszközeikbe. Ilyen emberkék dolgoznak az STMicroelectronics-nál is, akik az STM32-es mikrokontrollereket (is) gyártják. Az STM32-be tehát belekerül egy valamilyen ARM processzormag, valamint mellé építenek még egy csomó perifériát (timer, uart, i2c, stb.), és ezt becsomagolják egy IC-be.

Ahhoz, hogy ezt értelmesen használhassuk, ahhoz szükség van tápellátásra, csatlakozókra, meg esetleg még egy csomó más dologra is. A Nucleo nevű fejlesztőpaneleket is az ST gyártja, és ezeken épp egy ilyen STM32 van, tápellátás, csatlakozók, programozó, meg pár apróság. Most mi is egy ilyen fejlesztőpanelt fogunk csinálni, ami valószínűleg nem lesz egy világbajnok áramkör, de mi azért nagyon büszkék leszünk rá.

**Mi a cél?**

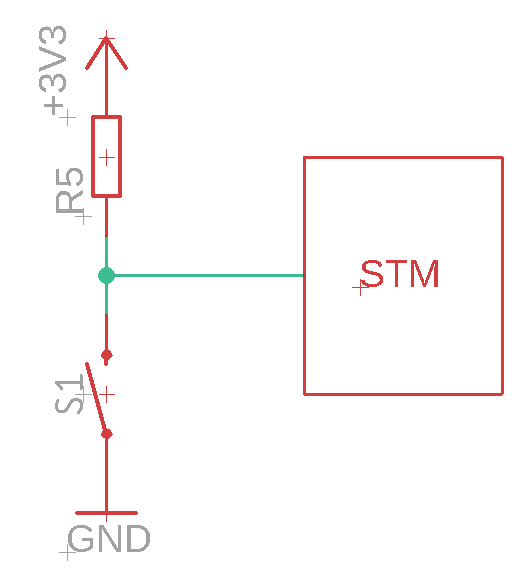
Egy kis arduino-shieldekkel (többnyire) kompatibilis egyszerű fejlesztőpanel. Ehhez az STM32F030K6T6-ot választottuk, mivel minden szükséges perifériát (5 db timer, UART, SPI, I2C, ADC) tartalmaz, valamint ez volt a legolcsóbb.

**LED-ek**

Ha a LED-ekre simán rákapcsoljuk a tápfeszültséget, akkor általában azt tapasztalhatjuk, hogy előbb-utóbb elhaláloznak. Ez azért van, mert -ahogy a nevük is monjda- ezek diódák, melyeknek tudjuk, hogy exponenciális az áram-feszültség karakterisztikája, tehát nem kell túl nagy feszültséget rájuk adni, hogy nagyon nagy áram folyjon keresztül rajtuk. Az áramot úgy tudjuk korlátozni, hogy a LED-del sorba kapcsolunk egy ún. előtét ellenállást. Ennek a méretét úgy kell megválasztani, hogy az ellenálláson a (tápfeszültség – LED nyitófeszültség) feszültség essen, azaz R = (Vcc-Vf)/I, ahol I a LED adatlapjából kilesett névleges áram. Ha nem szeretnénk teljes fényerővel világítani, akkor persze ennél kisebb I-vel is számolhatunk, mivel (szintén az exponenciális karakterisztika miatt) a diódán eső feszültség nem fog jelentősen változni.

**Gombok**

A digitális áramkörökben általában egy vezetéken a tápfeszültség jelenti az 1, a GND (0V) pedig a 0 logikai értéket. Emiatt egy nyomógombot egyszerűen berakhatunk a táp/föld és az adott vezeték közé, hogy megnyomáskor 1/0 értéket adjon. Mivel az egyszerű nyomógombok elengedett állapotban sehova sem kapcsolják a vezetéket, a mikrokontroller lába lebegni fog, azaz fogalmunk sincs, hogy milyen feszültség lesz rajta, mivel egy csomó random behatástól függhet. Ezt elkerülhetjük azzal, hogy egy ellenállást teszünk a láb és a föld/táp közé. Ekkor ha a gombot elengedjük, a láb az ellenálláson keresztül egy definit értékre kapcsolódik, megnyomott állapotban pedig az ellenálláson egy pici áram fog folyni, de ez nem probléma.

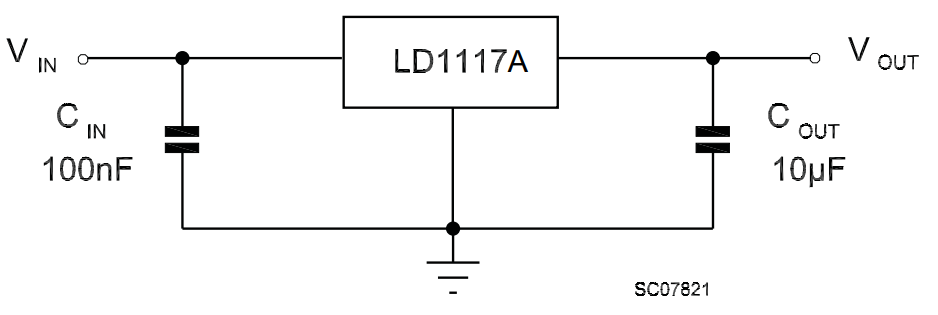


**Tápegység**

Az mikrokontrollerünk működéséhez 3.3V-os tápfeszültség szükséges. Azért, hogy nagyobb feszültéggel is táplálhassuk az áramkört (5V USB-ből, 9V elem, stb.), egy lineáris feszültségszabályzót fogunk használni. A legtöbb ilyennek 3 lába van: V\_IN, V\_OUT, GND. Egy fix kimeneti feszültégű típust használunk, ami azt jelenti, hogy „bármit” kapcsolok a bemenetre, a kimenet 3.3V lesz. A feszültségkülönbséget az IC elfűti, kb. (V\_IN-V\_OUT) \* I veszteség fog keletkezni. Mivel az áramkörünk általában nem fog sok áramot felvenni, ez nem lesz számottevő.

I

I

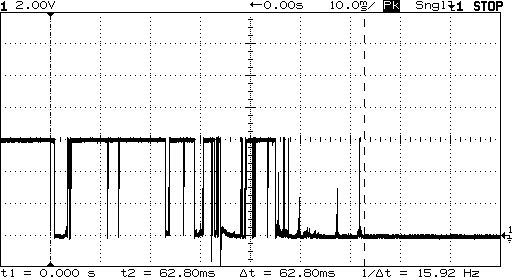


A ki- és bemeneti kondenzátorokra azért van szükség, mert ha elkezdjük hirtelen rángatni az áramot a kimeneten, akkor a stabilizátor nem tud elég gyorsan reagálni, ezért addig az áramot a feltöltött kondenzátorokban tárolt energiából pótoljuk. Ezen kívül ez az IC igazából egy visszacsatolt szabályozási hurok, aminek a stabilitását biztosítják a kondenzátorok.

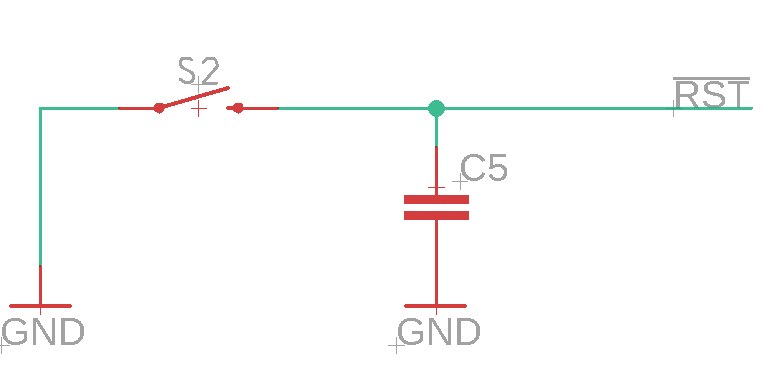
**STM32**

**Reset**

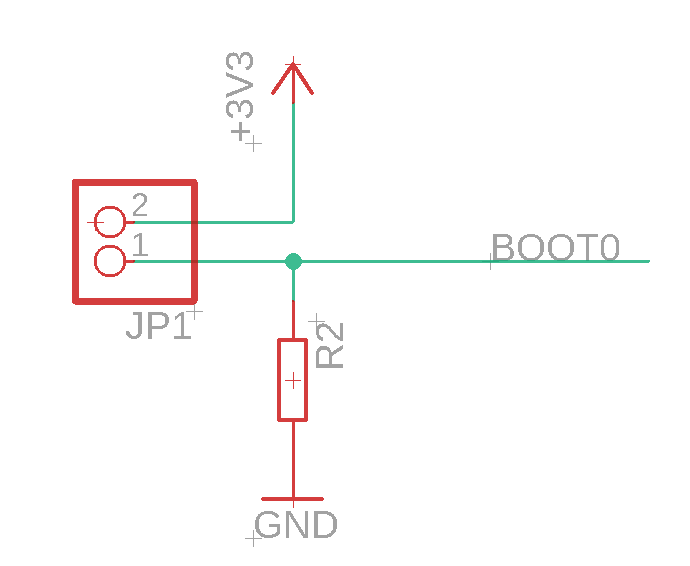
A mikrokontrollernek van egy negált reset lába (NRST), amire ha 0-t kapcsolunk, majd tápfeszültséget, akkor újraindul. Ez a láb belülről egy ellenállással a tápra kapcsolódik, tehát a gombnál leírtak értelmében nekünk csak egy gombot kell tennünk az NRST és a GND közé. A nyomógombok kontaktusai viszont nem érnek össze és válnak el tökéletesen, hanem ilyen csúnyaságokat művelnek:



Ez egy programból beolvasott gomb állapot esetén még szoftveresen megoldható, viszont a reset jel tényleg stabil kell hogy legyen. Ezért egy kondenzátort teszünk az NRST lábhoz, hogy a hirtelen feszültségugrásokat elnyelje.



**BOOT0 láb**

Minden STM32-ben van beépített bootloader, azaz nem csak speciális programozóval lehet a programot a memóriájába írni, hanem egyszerű kommunikációs interfészeken keresztül is. Ha a reset jel megszűnésekor a BOOT0 láb magas értékű, akkor a DFU (Device Firmware Updater) kezd el futni, ha pedig alacsony, akkor az általunk beleírt program. DFU módban például egyszerű UART-on is beleküldhetjük a programot az eszközbe, ami nagyon hasznos tud lenni. Ehhez egy kis jumpert teszünk a panelre, amivel be tudjuk kapcsolni a DFU módot. A működés tökéletesen ugyanaz, mint a nyomógomb esetében.

**Programozás**

Az említett DFU módon kívül programozhatjuk a kontrollert SWD interfészen keresztül is, ami debugolást is támogat. Ez egy JTAG-szerű dolog, a részleteit most még én sem ismerem. A lényeg, hogy a programozóval össze kell kötni az SWDIO (adat), SWCLK (órajel), és az NRST lábakat, és akkor minden nagyon jó lesz. Ehhez lerakunk egy kis csatlakozót, hogy könnyen össze lehessen kábelezni.

**Órajel**

Mivel a mikrokontroller is csak egy szinkron sorrendi hálózat, a működéshez szüksége van valamilyen órajelre. Az STM-ekben van beépített oszcillátor, viszont az nem túl pontos. Emiatt tartalmaznak egy beépített Pierce-oszcillátort is, mely lehetővé teszi, hogy egy precízen gyártott kvarc kristály rezgéseiből állítsuk elő az órajelet. Ha érdekelnek a részletek, akkor [itt](http://www.crystek.com/documents/appnotes/pierce-gateintroduction.pdf) egy cikk róla. Most elég annyit tudni, hogy az OSC\_IN és az OSC\_OUT láb közé egy ilyen struktúrát kell építenünk.

