Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Elektronikus Rubik-kocka fejlesztése

Önálló laboratórium jegyzőkönyv

2015/16. II. félév

Pokornyi Balázs Dániel

III. évfolyam, villamosmérnök szakos hallgató
BSc Beágyazott és irányító rendszerek specializáció

Konzulens:

Erdős Csanád
(Méréstechnika és Információs Rendszerek Tanszék)

Tartalomjegyzék

Feladatkiírás és specifikáció

Általános leírás

A cél egy olyan Rubik-kocka fejlesztése, mely a hagyományossal ellentétben nem forog fizikailag, hanem mind a mozgatás, mind a négyzetek színének reprezentálása elektronikusan működik. A kocka 6 db négyzet alakú NYÁK-ból épülne fel, melyeken elhelyezkedő 9-9 db LED által történik az egyes négyzetek aktuális színének megjelenítése, a forgatás pedig szintén az oldallapokon elhelyezkedő érintés érzékelő felületekkel lehetséges. Az eszköz mobilitását akkumulátor biztosítja, mely USB portról újratölthető.

Részletes tulajdonságok

- 6 db négyzet alakú NYÁK-ból felépülő térbeli kocka
- Oldalanként 9 db RGB LED-en történő megjelenítés
- Oldalakon elhelyezkedő kapacitív érintkező felületek segítségével való irányítás
- Akkumulátorról való működés, mely a kockán belül helyezkedik el
- Menthető állás: ki- majd újbóli bekapcsolás esetén folytatható a kirakás, a kikapcsoláskor aktuális állapotból
- Speciális érintés kombinációkkal előhozható utasítások:
    - Visszaállítás kirakott állapotba
    - Automatikus összekeverés
- Minimum 2 óra üzem idő
- USB-ről való tölthetőség kikapcsolt állapotban is
- Kikapcsolt állapotban alacsony fogyasztás
- Lehető legalacsonyabb alkatrész költség

Későbbi továbbfejlesztési lehetőségek

- Vezeték nélküli töltés
- Töltés közbeni animáció (fényeffektek)
- Automatikus kirakás funkció lépésről-lépésre (a kocka magát rakja ki)
- Segítő üzemmód: a következő lépést a kocka mutatja meg

Rendszer terv

A rendszer tervezése során elektronikai szempontból három különböző felépítés előnyeit és hátrányait hasonlítottam össze. A lehetséges megoldások mind címezhető LED-eket használnak, mert oldalanként 9 hagyományos RGB LED multiplexelése bonyolult és helyigényes lenne.

Összehasonlított koncepciók

Egy központi kontrolleres

A kockában, egyetlen központi mikrokontroller van, a többi oldalon pedig cél IC kezeli az érintő felületeket. (6x9 érintő felület nem lenne elvezethető egy oldalra, illetve nagy lenne a bizonytalanság az oldalak közötti érintkezésnél.)

Előnyök:

- Egy kontroller vezérli az egészet: nem tud fellépni a vezérlők között inkonzisztens állapot
- Jobb érintésérzékelés (feladatra készült IC)
- Egy vezérlő program

Hátrányok:

- Drágák a touch vezérlő IC-k (400-500Ft/db olyan, ami tud 9 felületet mintavételezni)
- Az oldallapokon lévő címezhető LED-ek sorba kötése (54 db) bonyolult és a touch IC-k kommunikációs vonalai mellett ezt is át kell vezetni a nyákok között
- Különböző nyák szükséges a master és slave oldalaknak: drágább

Multi masteres

Minden oldallapon külön mikrokontroller van, melyek egy közös kommunikációs buszon keresztül kommunikálnak egymással és egyenértékűen vezérlik a kockát. Minden kontroller tudja a jelenlegi állapotot, és mindenki elvégzi ugyanazt a műveletet.

Előnyök:

- Megoldható mindegyik oldal ugyanolyan nyákkal
- Egy vezérlőprogram, csak paraméterekben különböznek egymástól
- Csak egy fajta kommunikációs buszt kell átvezetni az oldalak között
- Az adott oldalon lévő LED-eket tudja az oldalon lévő kontroller vezérelni

Hátrányok:

- Előfordulhat, hogy valamelyik oldal újraindul vagy meghibásodik: előfordulhat inkonzisztens állapot (kiküszöbölhető, de bonyolultabb program)
- Rosszabb/bonyolultabb érintésérzékelés, mint cél IC-vel

Köztes megoldás

Minden oldalon van mikrokontroller, de azok nem egyenértékűen működnek. Egyetlen MCU vezérli, a többi pedig perifériaként működik, melyek csak az adott oldalon lévő LED-ek meghajtását és az érintő felületek mintavételezéséért felelnek.

Előnyök

- Megoldható ugyanolyan nyákkal minden oldal, de a slave oldalakon olcsóbb kontroller (azonos MCU családból, azonos lábkiosztással) is elég
- Olcsóbb, mint minden oldalba ugyanolyan kontroller
- Csak egy fajta kommunikációs buszt kell átvezetni az oldalak között
- Az adott oldalon lévő LED-eket tudja az oldalon lévő MCU vezérelni
- Nem tud fellépni az állásban inkonzisztens állapot

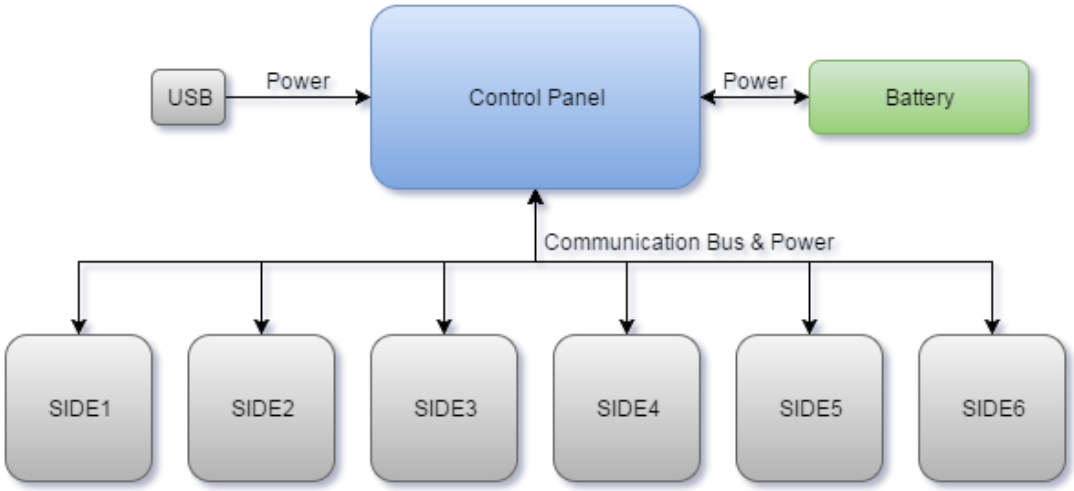
Hátrányok

- Rosszabb/bonyolultabb érintésérzékelés, mint cél IC-vel
- Több féle firmware-t kell írni: master és slave

Végleges rendszerterv

A legtöbb előnye, illetve legolcsóbb megvalósíthatósága miatt köztes megoldást választottam:

A kocka minden oldalán lesz egy mikrokontroller, melyek slave eszközként működnek és emellett lesz egy vezérlőpanel, mely az oldalakkal kommunikálva vezérli a kockát.



Vezérlőpanel

- **Feladatok**
  - Ki-/bekapcsolási feladatok vezérlése
  - Táp menedzsment
    - Akkumulátortöltés
    - Tápfeszültség előállítása az oldalak számára
    - Akkumulátor feszültség (töltöttség) mérése
  - Kommunikáció az oldalakkal
  - Játék/működés vezérlése
- **Megvalósítás**
  - Vezérléshez mikrokontroller alkalmazása
  - 5V előállítása: Step-Up
  - Akkumulátortöltéshez: töltő IC
  - MASTER eszköz

Oldalpanelek

- **Feladatok**
  - Érintésérzékelés (9 db érintőpanel)
  - Színek megjelenítése (9 db RGB led)
  - Kommunikáció a vezérlővel
  - Kocka fizikai stabilitásának biztosítása
- **Megvalósítás**
  - Mikrokontroller alkalmazása, mint cél IC
    - Érintőpanelek kezelése
    - Ledek meghajtása
  - SLAVE eszköz

Kommunikációs busz

A kommunikáció kiválasztásánál az alábbi szempontok játszottak fontos szerepet:

- Kevés vezeték (oldalak közötti átvezetés miatt fontos)
- Külső zajokra érzéketlen
- Egyszerű meghajtás
- Mikrokontrollernek legyen ilyen perifériája
- Elég gyors a szükséges adatok átviteléhez

A fenti szempontokat figyelembe véve az **I^2C** bizonyult alkalmasnak a feladatra, mely az összes kritériumot teljesíti.

Fizikai felépítés, megvalósítás

A kocka minden oldala egy NYÁK lemez, melyeknek kommunikálnia kell egymással, illetve a szerelhetőség miatt szétszedhetőnek is kell lennie.

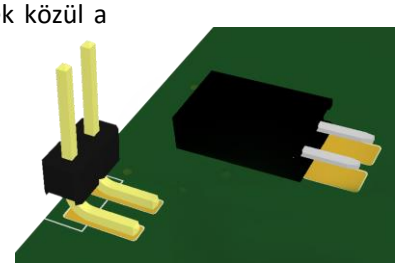
A fizikai felépítés megtervezése során az alábbi szempontokat kellett betartani:

- Szétszedhetőség
- Kommunikáció körbe érjen a kockán, de ne legyen hurok a buszon
- Teljesen egyforma oldalak
  - Olcsóbb gyárthatóság miatt: 1 féléből többet olcsóbb
  - Ne kelljen több fajta NYÁK-ot gyártani

A fenti szempontokat figyelembe véve az alábbi megoldások születtek a fizikai felépítésre:

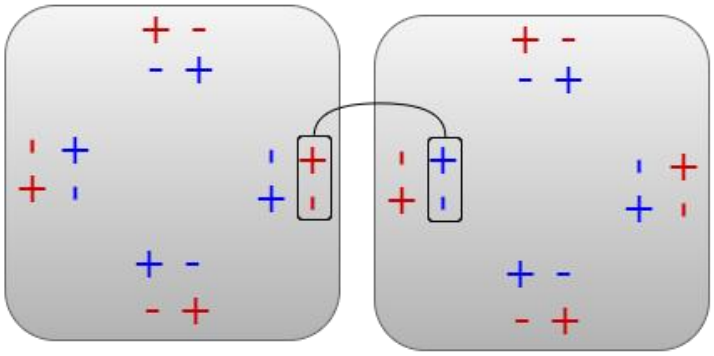
Szétszedhetőség: csatlakozó

Az oldalak csatlakoztatása tüske és hüvelysorpárokkal történik, melyek közül a tüskesor egysoros, merőleges kivitelű, mely a NYÁK felületére, SMD alkatrészként kerülne beforrasztásra. A hüvelysor pedig hagyományos egysoros, mely azonban a nem a NYÁK-ra merőlegesen, hanem a NYÁK felületére fektetve kerülne beforrasztásra. Csatlakozókat nem lehet úgy elhelyezni, hogy pl. a NYÁK 2 oldalán tüske-, a másik 2-n hüvelysor van, mert akkor a kocka nem lesz összetehető (az utolsó oldalt pl. nem lehetne rányomni). Ezért szükség van mindkét fajta (apás és



anyás) csatlakozó minden helyen való elhelyezéséhez, hogy a NYÁK-ok egyformasága megmaradjon. Így lehetőség van minden oldalra a szükséges csatlakozó beforrasztására.

Táp és kommunikációs vonalak elvezetése



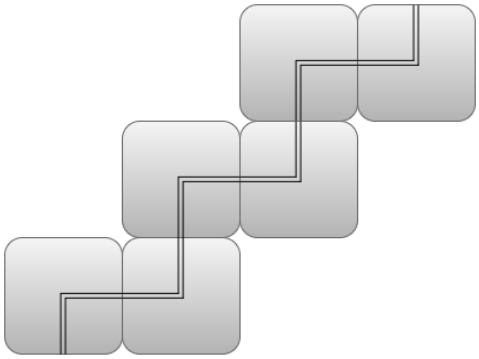
A táp és kommunikációs vonalak átvezetésénél az a probléma merül fel, hogy ha minden csatlakozót egy irányba osztunk ki (az adott helyen lévő tüske- és hüvelysor is ugyanúgy (párhuzamosan) van kiosztva), akkor amikor össze akarnánk az ugyanolyan oldalakból álló kockát rakni, akkor tükrözve találkoznának a pinek. Ez az ábrán is látszik: amennyiben egy meghatározott irányban megjelöljük az oldalakat (kék + és -), akkor 2 ugyanolyan oldal egymás mellé tételekor éppen ellentétesen találkoznak a jelölése.

Erre a problémára 2 megoldást találtam:

- A csatlakozók számát oldalanként megduplázva (legalábbis (n-1)*2) szimmetrikussá tehetjük a csatlakozásokat, így nem jelent gondot a tükrözés
- Amennyiben az apás és anyás csatlakozók pinjeit éppen ellenkező irányban osztjuk ki, akkor az oldalakat egymás mellé téve megfelelően fognak találkozni az oldalak, mivel a minden csatlakozási helyen 1 anyás és 1 apás csatlakozó van (ábrán a kék és piros jelölések)

Ezek közül az **utóbbit választottam**, mivel a NYÁK-on viszonylag kevés hely van.

Kommunikáció körbevezetés hurok nélkül



A kommunikáció körbevezetésénél cél volt, hogy ne legyen hurok a buszon, annak ellenére, hogy a I^2C maximum 400kHz-s sebességét tekintve valószínűleg nem okozott volna gondot.

Ez az ábrán látható módon lett megoldva: amennyiben derékszög alakban, 2 szomszédos oldal között vezetjük el a kommunikációs buszt, akkor össze lehet forgatni úgy a négyzeteket, hogy abból összehajtható legyen egy kocka, illetve hurok se keletkezzen.

Kapcsolás-, NYÁK tervezés

A tervezés rendszerterv utáni lépése a szükséges áramkörök és NYÁK-ok megtervezése volt. Az áramkörök mindegyikét **Altium Designer** tervezőben készítettem el.

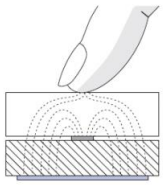
A programban használt alkatrészkönyvtárakat magam készítettem.

Oldalpanel

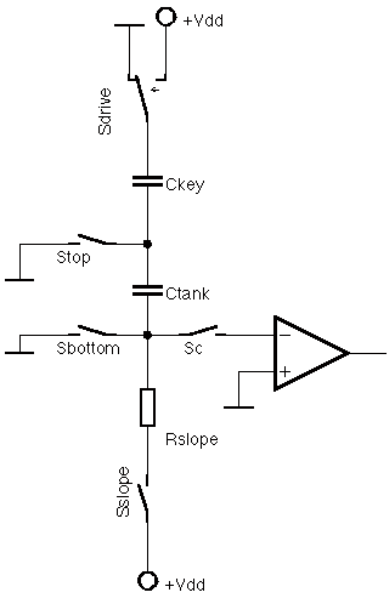
Áramkör elemei és választásuk indoklása:

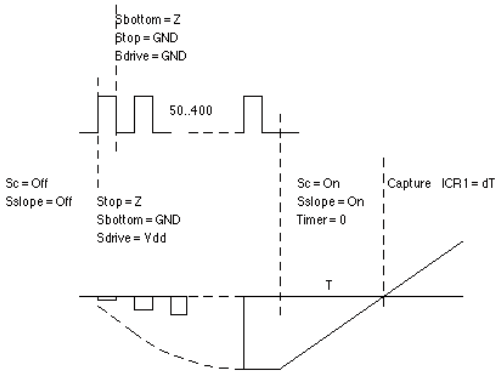
- ATmega88 mikrokontroller
  - Kedvező ár
  - Megfelelő lábszám
  - Rendelkezik a szükséges perifériákkal
  - QMatirx támogatás (touch)
- WS2812B címezhető RGB led
  - Egyszerű meghajtás
  - Nagy felület
  - Hagyományos RGB ledek meghajtására nem lenne elegendő hely (9x3 = 27 led mátrixba kötése viszonylag sok lábat/meghajtót igényel, sok helyet foglal)
- I^2C kommunikáció
- QMatrix érintésérzékelés
- Debug LED
- Programozó port

Érintésérzékelés



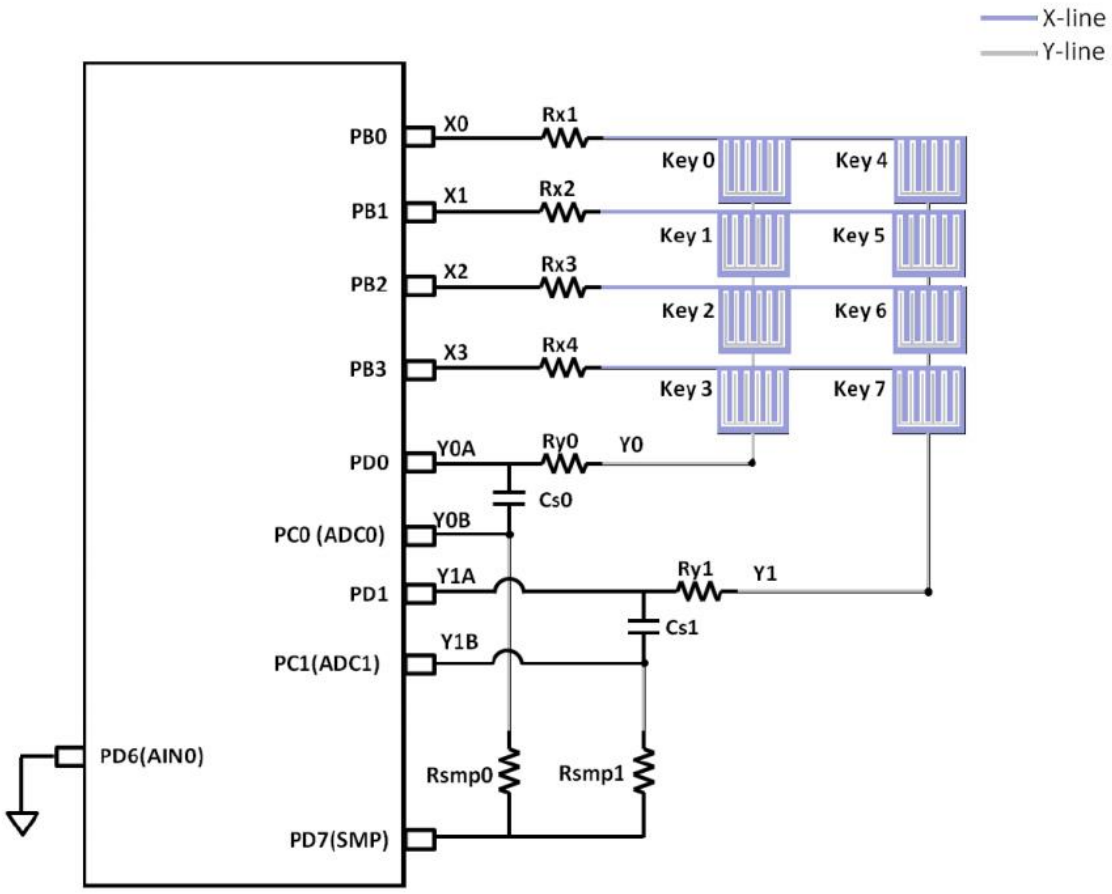
Érintésérzékeléshez az Atmel QMatrix elvét használtam. Ennek előnye, hogy mátrixba rendezhetőek az elhelyezett érintésérzékeny felületek, így a szükséges 9 érintőfelület számár elegendő 10 IO láb a kontrollerről. A módszer lényege, hogy az ábrán látható mintázatot kialakítva a NYÁK-on egy kondenzátort hozunk létre, aminek a kapacitását – az ujjunkat közelítve – befolyásolni tudjuk. Viszont mivel itt pF-os nagyságrendű kapacitásokról van szó, ezért hogy ezt mérni tudjuk, a következő kapcsolást alkalmazhatjuk:





Itt a *Ckey* a NYÁK-on kialakított mintázattal létrehozott kapacitás, a *Ctank* pedig egy, egy nagyságrenddel nagyobb (~nF) segédkapacitás. Az *Sdrive, Stop, Sbottom (= Sc), Sslope* kapcsoló pedig a mikrokontroller 1-1 lábát takarja. A mérés elve, hogy a jobb oldali ábrán látható sorrendben a lábakat a megfelelő potenciálra kapcsolva a *Ckey* kapacitásán keresztül n-szer (n~50..400) töltjük a *Ctank* kapacitást, így mérhető tartományba hozzuk a *Ckey* kapacitását. Következő lépésként meg kell mérnünk a *Ctank*-ban összegyűlt töltést, amit úgy tudunk megtenni, hogy *Stop*-ot zárva hagyva (földre húzva) *Ctank* alsó felén negatív feszültséget mérhetünk. Ezt a pontot *Rslope* kisütő ellenálláson keresztül felhúzzuk tápra, aminek a hatására a komparátor invertáló bementén elkezd nőni a feszültség és amint eléri a 0V-ot átbillen a kimenete 1-ből 0-ba. Ez az idő (amennyi idő alatt a komparátor átbillen/ameddig 1 a kimenet) természetesen a *Ckey* kapacitással arányos, így átvezettük ennek mérését időmérésre, melyet a MCU egyik beépített Timer-évl meg tudunk tenni.

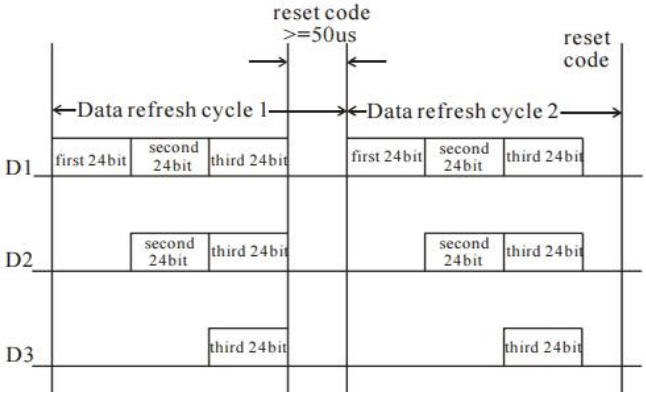
Ez a módszer a következőképpen köthető mátrixba:



A fent látható kapcsolásnál arra szükséges figyelni, hogy az Y0B…YnB vonalakat olyan bemenetre kössük, ami a mikrokontroller belső komparátora invertáló bemenetére kapcsolható. (Ez jelen esetben (ATmega88) a kontroller ADC bemeneteit takarja.)

Címezhető LED-ek

A színek megjelenítésére hagyományos RGB LED-ek helyett WS2812B LED-eket használtam, melynek előnye, hogy felfűzhetőek egy darab soros adatvonalra, így nem igényelnek sok helyet foglaló multiplexelést.



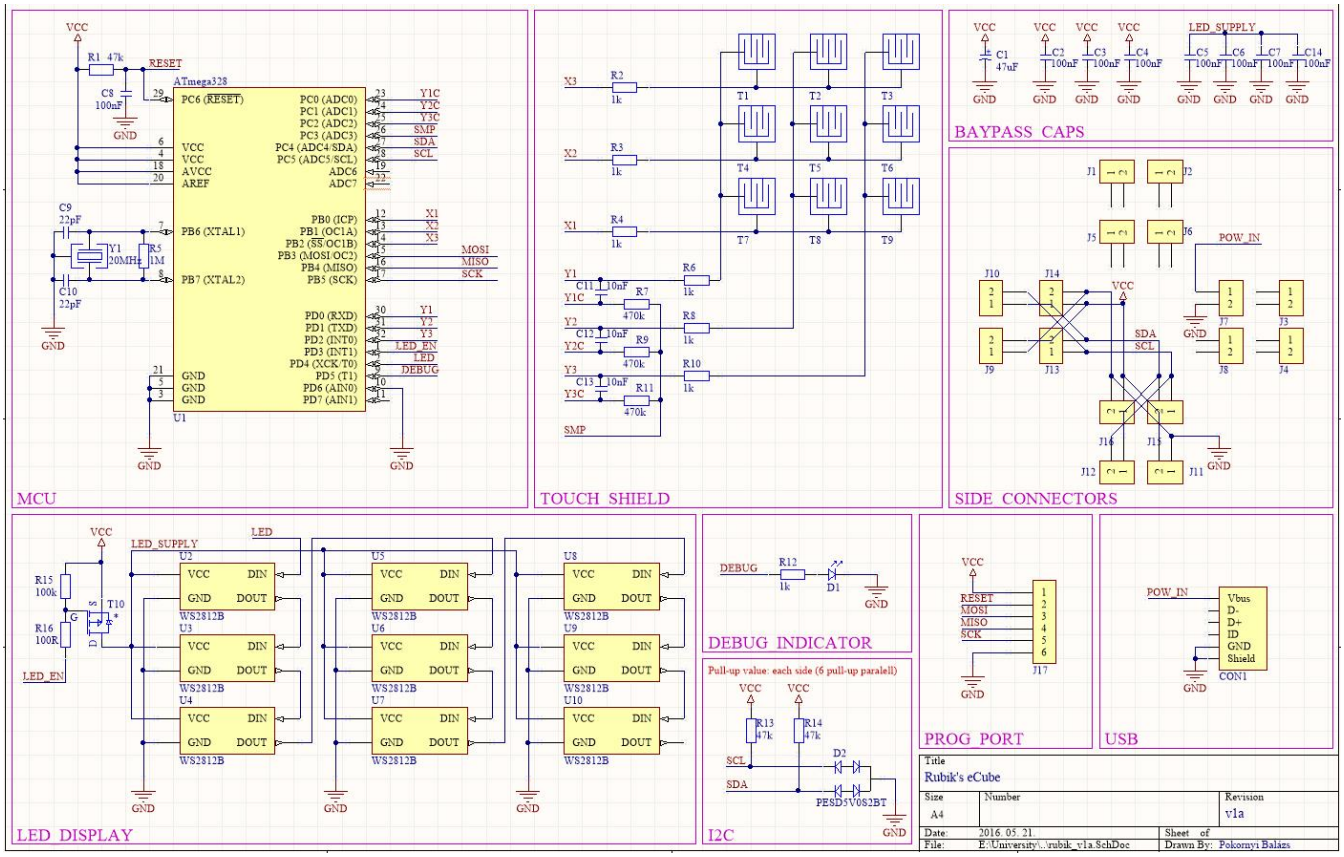
Címzésük a következőképpen történik:
Az ábrán látható módon mindegyik LED-nek egy 24 bites (RGB -> 3x8bit) színkódot kell küldenünk, egy adatfrissítési ciklusban szünet nélkül. A *Dx* vonal az *x.* led előtti adatvonalat mutatja: tehát minden LED leveszi az első 24 bites színkódot, majd a többit tovább küldi, amíg 50us-es holtidő nincs 2 adat közt, aminek a hatására resetelődik az adatküldési ciklus.

Kommunikáció: I²C

A következő előnyei miatt esett az I²C-re a választás:

- 2 vezeték
- Szinkron kommunikációs
- Kétirányú adatküldés (egyszerre egy irány)
- Hosszabb távokra tervezett (akár 50 cm)
- Címzési lehetőség
- Több slave és master csatlakoztatható egy buszra

Kapcsolás



A kapcsolás elemei/moduljai:

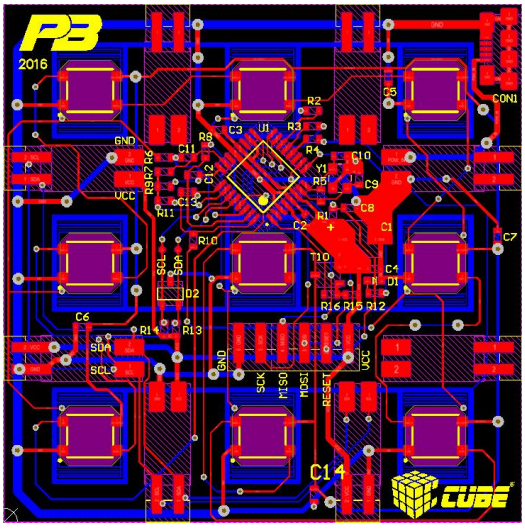
- MCU
  - Tartalmazza a mikrokontrollert
  - Külső 12MHz kvarc és a hozzá tartoró kondenzátorok
  - Reset felhúzó ellenállás és kapacitás a stabil jelhez
- TOUCH_SHIELD
  - QMatrix elven működő érintésérzékelők
- LED_DISPLAY
  - WS2812B típusú címezhető LED-ek sorba kötve
  - LED-ek tápját kapcsoló FET és az ehhez tartozó felhúzó és GATE ellenállás
- BYPASS_CAPS
  - MCU-hoz tartozó tápszűrő kondenzátorok
  - LED-ekhez tartozó tápszűrő kondenzátorok
- I2C
  - I^2C vonal felhúzó ellenállásai (nem feltétlen szükséges a beültetésük, elég a MASTER-nél)
  - I^2C vonal ESD védelme
- DEBUG_INDICATOR
  - MCU egyik pinjére kötött LED debug célokra és állapotjelzésre
- PROG_PORT
  - Programozó csatlakozó
- USB
  - USB csatlakozó (akkumulátortöltéshez)
  - Csak az egyik oldalon kell beültetni
- SIDE_CONNECTORS
  - Oldallapok csatlakoztatására szolgáló csatlakozók
  - Tüske és hüvelysorok
  - A kommunikáció és tápvonalak 2 oldal közötti elvitele, a csatlakozókba szimmetrikusan bekötve (lásd. fizikai felépítés rész)

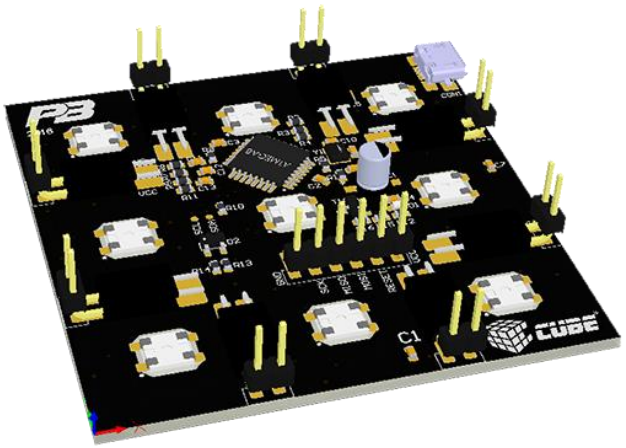
NYÁK

A NYÁK tervezésnél figyelembe vett szempontok:

- QMatrix vonalainak az elhelyezése
  - Minél nagyobb távolság a kommunikációs vonalaktól
  - Ne legyen a közelében semmi, ha kell, akkor is csak merőlegesen keresztezze más huzal (minél kisebb kapacitív csatolás)
- Tápvonalak vastagsága
  - A LED-ek számára szükséges áramtól ne melegedjen a huzal
- Csatlakozók helyzete, pontos elhelyezése
  - Összeilleszthetőek legyenek az oldalak
- LED-ek pontos helyzete: rácsban
- A NYÁK külsején (összeillesztve a kocka belsejéből nézve)
  - LED-ek fejjel lefelé beforrasztva, alattuk, illetve a túloldalon a forrasztásgátló levétele szükséges, hogy át tudjanak világítani

A bal oldali ábrán látható az oldallap elkészült NYÁK terve, a jobb oldali ábrát pedig a NYÁK 3D-s képe látható.





Vezérlőpanel

A vezérlőpanel feladata, a teljes rendszer vezérlése mellett az akku töltése és a tápfeszültség előállítása az oldalak számára.

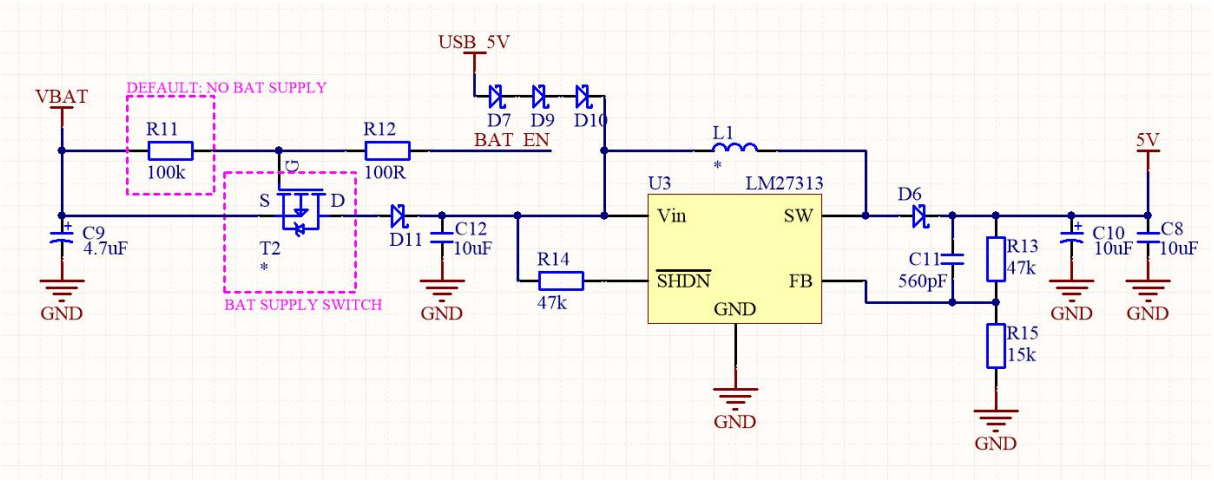
Az áramkör elemei:

- ATmega328 mikrokontroller
  - Rendelkezik a szükséges perifériákkal (I^2C, Timer, ADC
  - 20MHz-es működési frekvencia
  - Megfelelő ár
- Töltő IC: MCP73832
  - Olcsó
  - 500mA töltés (~700mAh akkumulátor számára megfelelő)
  - Egyszerű kezelés
- Akkumulátor kapcsolhatóság
  - Kikapcsoláshoz
- 5V előállítása: Step-up
  - Címezhető LED-ek számára 5V szükséges
  - Stabil érintés érzékeléshez
  - LM27313 (rendelkezésre állás miatt)
- Akkumulátor feszültség mérés
  - Fesz. osztó kapcsolható legyen: ne merítse az akkumulátort
- USB kábel detektálás
- Csatlakoztathatóság valamelyik oldalra

5V előállítása

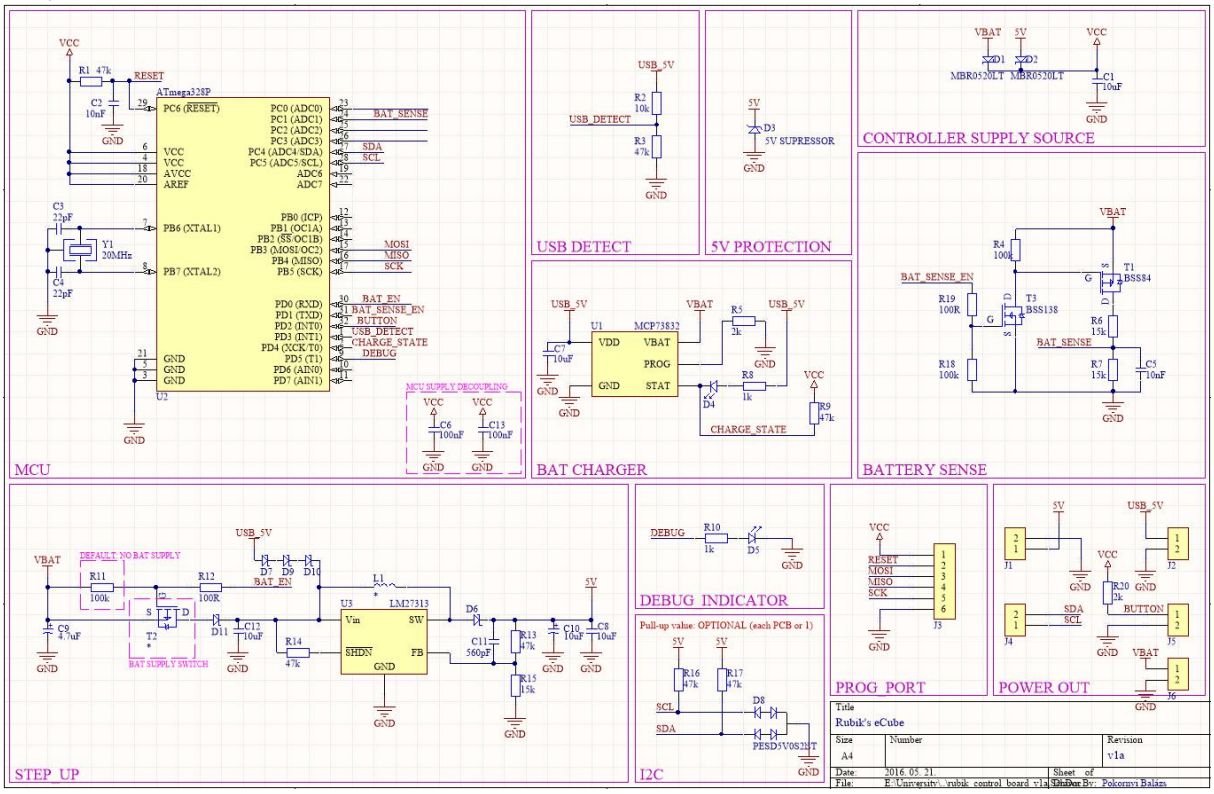
A stabil tápfeszültségre két okból volt szükség: a LED-ek névleges tápfeszültsége és az érintésérzékelés miatt. Az érintésérzékelés miatt azért van szükség stabil tápfeszültségre, mert amennyiben az akkumulátor feszültségéről működtetnénk az áramkört, akkor folyamatosan változna a tápfeszültség, ezzel együtt az érintés érzékelése. Továbbá, azt is meg kell oldani, hogy ugyanannyi legyen a tápfeszültség USB-ről való működés közben és töltés közben is. Mivel miközben töltjük az akkumulátort, nem meríthetjük, így olyankor

az USB vonalat kell használni tápforrásnak, ami azonban nem stabil 5V. Az USB standard szerint a tápvonal 4.40-5.25V közötti tartományba esik. Ezért, hogy közelítőleg állandó 5V legyen az oldallapok számára, az alábbi kapcsolást terveztem:



A VBAT az akkumulátor csatlakozási pontja, melyről egy FET-en keresztül jut a feszültség a Step-up bemenetére. Ezzel a FET-el van lehetősége a vezérlő mikrokontrollerrel lekapcsolni az 5V-ot (a MASTER MCU nem erről az 5V-ról jár). A kapcsoló FET alapból zárva van (R11 felhúzó ellenállás) és a mikrokontroller tudja kinyitni a BAT_EN vonal földre húzásával. (BAT_EN meghajtása elegánsabb lenne egy plusz invertáló fokozattal (N-FET), hogy ne tudjon az előfordulni, hogy az MCU-ban lévő védődiódák kinyissák a FET-et, de működik így is.) Az USB 5V-ja 3db Schottky diódán keresztül van ~4V-ra leejtve, amiből a step-up konverter már elő tudja állítani az 5V-ot. A D11-es diódára azért van szükség, mert különben az USB 5V-ja elkezdené tölteni a FET-ben lévő parazita diódán keresztül az akkumulátort.

Kapcsolás



A kapcsolás elemei/moduljai:

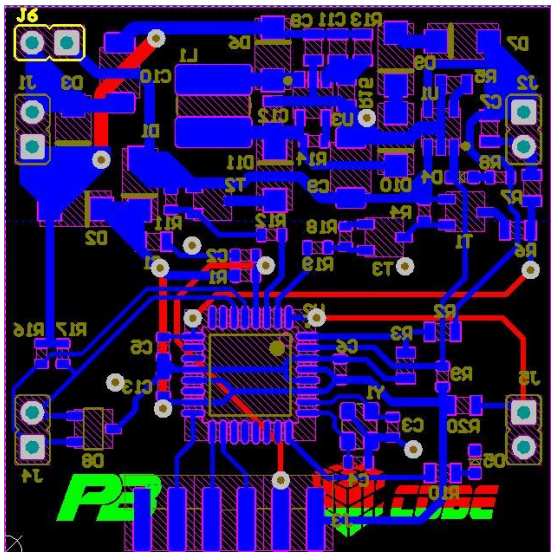
- MCU
  - Tartalmazza a mikrokontrollert
  - Külső 12MHz kvarc és a hozzá tartoró kondenzátorok
  - Reset felhúzó ellenállás és kapacitás a stabil jelhez
  - Szűrő kondenzátzorok a tápra
- USB_DETECT
  - USB detektálásra fesz. osztó: akkumulátorról való működés során ne kapjon túlfeszültséget
- CONTROLLER_SUPPLY_SOURCE
  - Vezérlőpanel áramforrása: akkumulátor és 5V összedióda
  - VBAT azért szükséges, hogy a vezérlő kontrollernek (MASTER) mindig legyen tápja
  - 5V táp azért szükséges, hogy a amennyiben kifelé (SLAVE) oldalak felé van 5V táp, akkor a vezérlő is 5V-ról járjon, a kommunikációnál való feszültségkülönbségek elkerülése végett
  - Schottky: kis feszültségesés miatt (kisebb veszteség)
- BAT_CHARGER
  - Akkumulátor töltő modul
  - MCP73832 töltésvezérlő IC-vel
  - R5 töltőáram beállítás (2k-> 500mA , 750mAh-s akkumulátor miatt)
  - R5 felhúzó ellenállás töltés detektálásához az MCU felől
- BATTERY_SENSE
  - Akkumulátor feszültség mérése
  - Feszültségosztó: belső referenciához képest való méréshez, le kell osztani
  - FET-ek: feszültségosztó kapcsolhatósága, hogy nem terhelje folyamatosan az akkumulátort
  - Előfokozattal kapcsolva
- STEP_UP
  - 5V előállítása
  - LM27313 step-up
  - USB leejtve ~4V-ra, hogy vissza lehessen fix. 5V-ra konvertálni
  - T2: akkumulátor kapcsoláshoz
- DEBUG_INDICATOR
  - MCU egyik pinjére kötött LED debug célokra és állapotjelzésre
- PROG_PORT
  - Programozó csatlakozó
- USB
  - USB csatlakozó (akkumulátortöltéshez)
  - Csak az egyik oldalon kell beültetni
- POWER_OUT
  - Csatlakozók az oldallaphoz való csatlakozáshoz
  - Akkumulátor csatlakozó
  - Táp ki (5V) és bemenet (USB)

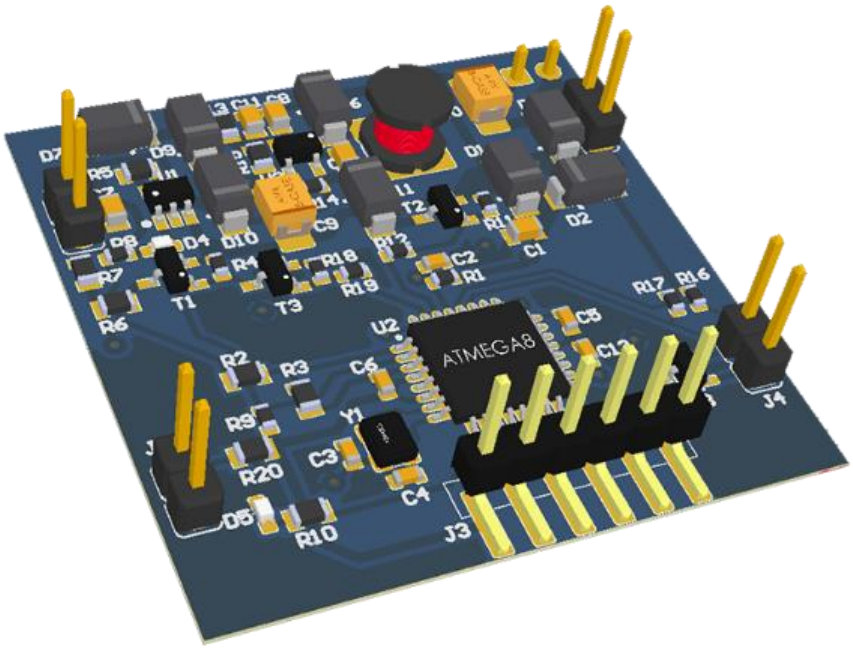
NYÁK

A NYÁK tervezésnél figyelembe vett szempontok:

- Step-Up kialakítása
  - kapcsolóüzemű áramkör (1.6MHz) miatt érzékeny az elhelyezésekre
  - IC, L1, D6 egymáshoz közel helyezése
  - Visszacsatolás minél rövidebb úton
  - Közös földpont a ki, bemeneti kondenzátoroknak
- Szükséges áramoknak megfelelő huzalvastagságok (max 500mA)
- Csatlakozók helye (oldalpanel által megszabott)
- Analóg mért vonalak kialakítása
  - Minél kisebb feszültségesés, zavar
- 40 x 40mm méret

A bal oldali ábrán a NYÁK terv látható (földkitöltések elrejtve), a jobb oldalin a NYÁK 3D-s képe.





Áramkör összerakás, forrasztás, felélesztés

Az elkészült NYÁK forrasztása és felélesztése szintén a saját munkám. A két panel gyártása között vannak különbségek, erre az adott résznél kitérek.

Oldalpanelek

Gyártás

Az oldalpanelek NYÁK-jainak legyártását *dirtypcbs.com* végezte. Azért erre esett a választás, mert szükség volt a 0.15mm-es huzalvastagság, illetve huzaltávolságra (érintéspanelek fésűs kialakítása miatt), aminek a gyártása az ETT-s NYÁK gyártás lehetőségein kívül esik.

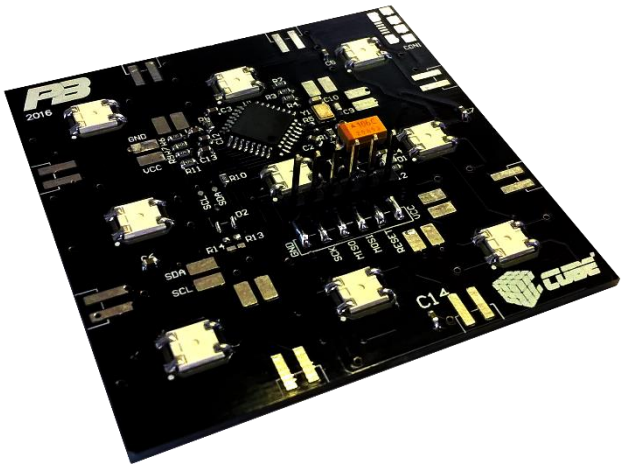
Az elkészült NYÁK-okon forrasztás előtt még ki kellett alakítani a LED-ek helyét. Ez azt jelenti, hogy olyan bemélyedéseket kellett a NYÁK-ba marni, melyekbe a LED-ek belefekszenek, így besüllyesztve kerülnek beültetésre. Erre 2 okból volt szükség:

- Mivel a LED-ek fejjel lefelé történő beforrasztással lettek a NYÁK-ra tervezve (átvilágítanak a NYÁK-on), ezért ez megkönnyíti a beültetést, azzal hogy a LED-ek lábai a pad-ekkel egyszintre kerülnek.
- Kék LED-ek színe miatt: a kék LED-eket a NYÁK-on keresztül átvilágíttatva azt tapasztaltam, hogy közel fehér szín látszik a NYÁK túloldalán. Ez valószínűleg azért van, mert gerjeszt valamit a kék fény a NYÁK-ban (FR4), aminek a hatására additív színkeveréssel fehéret kapunk. Ennek kiküszöbölés céljából csökkenthetünk a NYÁK vastagságán, így redukálható a gerjesztő hatás intenzitása és kékhez közelebb álló színt kaphatunk.

A NYÁK-ok helyeinek bemarását CNC-vel végeztem, melyhez a szükséges G kódot kézzel írtam meg.

Beültetés

Az áramkör úgy lett megtervezve, hogy csak olyan alkatrészek szerepelnek rajtva (0603 a legkisebb), melyek kézzel és forrasztópákával kényelmesen beültethetők. A 6db oldallap forrasztását párhuzamosan végeztem, ezzel növelve a munka sebességét és hatékonyságát.



Az oldallapok közötti csatlakozók beültetése jelentette a legnagyobb nehézséget. Ezeket az egyik oldalra beültetve, majd a csatlakozó másik felét rádugva, a két panelt egymáshoz merőlegesen tartva lehetett a legkönnyebben beforrasztani. A jobb oldali ábrán a kész, beültetett panel látható.

Élesztés

Beültetés után az áramkört mikroszkóp alatt ellenőriztem, mely során nem találtam hibát. Az áramkör feszültség alá helyezése és beindítása során nem volt gond, a mikrokontrollert is elsőre programozni tudtam.

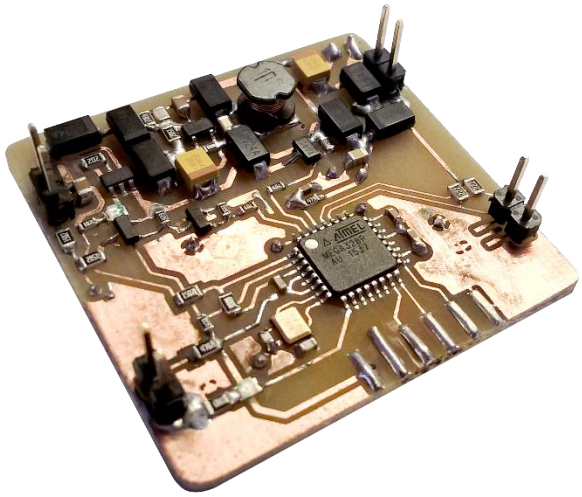
Vezérlőpanel

Gyártás

A vezérlőpanelt annak érdekében, hogy gördülékenyen haladhasson a fejlesztés, nem NYÁK gyártónál gyártattam le, hanem a SEM műhelyében (Schönherz Elektronikai Műhely) fotolitográfiás eljárással magam készítettem el. Ennek eredményeképp a vezérlőpanelen nincsen forrasztásgátló réteg, illetve a furatgalván hiányában a beültetést a viák huzalarabkákkal történő helyettesítésével (kétoldalról beforrasztott rézdrótok) kellett kezdeni.

Beültetés

Az oldalpanelhez hasonlóan, ezen a NYÁK-on is csak 0603 és annál nagyobb méretű alkatrészek kaptak helyet, így ennek a beültetése is kézzel történt. A forrasztásgátló és a pozíciószita hiányában ennek a NYÁK-nak a forrasztása során fokozott figyelmet kellett fordítani az alkatrészek pozíciójára és polaritására.



A jobb oldali ábrán a beültetett NYÁK fotója található, mely még a alkoholos lemosást megelőzően készült így a folyasztószer maradványai láthatóak az áramkörön.

Élesztés

Az áramkör élesztése során ellenőrizve lett a tápot és töltést szolgáltató egységek működőképessége és terhelhetősége:

- Töltő modul
  - Maximális töltőáram: **487mA**
  - Töltéslekapcsolás feszültségszint: **4.2V**
  - A töltő helyesen működik.
- Step-up converter
  - Kimeneti feszültség: **5.1V (3.5V in), 5.09V (4.2 V in)**
  - Maximális terhelhetőség: **300mA**
  - Kimeneti zaj: **±45mV**
  - A Step-up megfelelően működik.

A mikrokontrollerhez a programozót csatlakoztatva elsőre programozható volt.

Firmware fejlesztés

A firmware fejlesztést az oldalpanelek kódjának megírásával kezdtem, mivel ezek fognak SLAVE eszközként működni a rendszerben, illetve a MASTER őket vezéreli, illetve az oldalpanelek adatai alapján végzi a műveleteket.

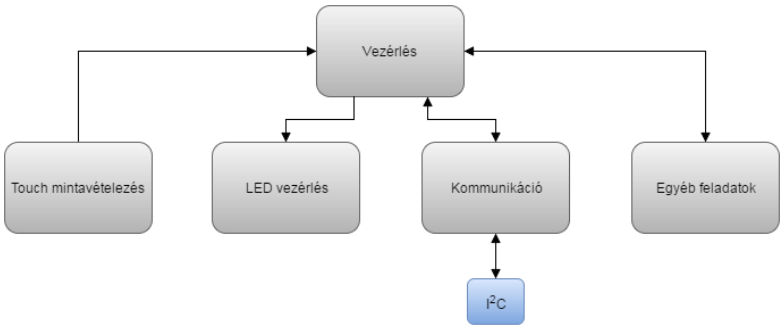
A firmware-k mindegyike *Atmel Studio 6* fejlesztőkörnyezetben készült.

Oldalpanel firmware (SLAVE)

Feladat, blokkdiagramm

Az oldalpanelnek az alábbi feladatokat kell ellátnia:

- Kommunikáció I^2C-n
- Érintőpanelek mintavételezése
- LED-ek vezérlése
- Egyéb feladatok
  - Kalibráció
  - Ki-bekapcsolás
  - Állapot megmondása

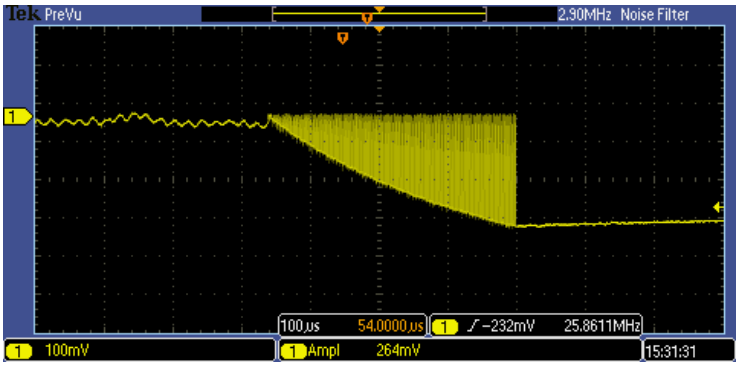


A felsorolt feladatoknak megfelelően firmware vázlatos blokkdiagramja az ábrán látható. Az egyes feladatok ütemezéséréért és vezérléséért egy központi vezérlés felelős.

Érintés mintavételezése

Az érintőpanelek mintavételezését egy saját könyvtárral oldottam meg, mely az *Érintésérzékelés* fejezetben ismertetett elven mintavételezi az érintőfelületek állapotát. Az elkészített könyvtár tartalmazza a következőket:

- Folyamatos (auto) kalibráció (kis határokon belül)
- Kalibrációs utasítás (teljes kalibrálás)
- Folyamatosan futó taszk-ként mintavételezés
- Eredmények tárolása egy globálisan hozzáférhető tömbben
- Paraméterekkel beállíthatóság



Az elkészített könyvtár működését ellenőriztem, illetve oszcilloszkóp segítségével is megvizsgáltam a kapcsolásban szereplő kondenzátorokon mérhető feszültséget. A jobb oldali ábrán látszik, hogy a kondenzátor az elvnek megfelelően valóban „felpumpáljuk” a fésűs minta által kialakított pF-os nagyságrendű kapacitáson keresztül. Itt látható, hogy a kondenzátornak a komparátor bemenetére kötött oldalán valóban negatív feszültséget mérhetünk a „pumpálás” után.

LED vezérlés

A LED-ek vezérlésére egy kész könyvtárat használtam, mely nem a saját munkám. Mivel a LED-eknek elég nagy sebességgel kell küldeni az adatot, ez egy .h fájl formájában asm utasításokkal van megírva. *A könyvtár eredetét nem ismerem, már korábban, más projektekkel kapcsolatban is használtam.*

Kommunikáció (I^2C)

A kommunikáció megvalósítására szintén saját I^2C könyvtárat hoztam létre, mely megszakításalapon működik, annak érdekében, hogy minél kevesebb CPU időt vegyen igénybe. A könyvtár az alábbi tulajdonságokkal rendelkezik:

- Saját handler struktúrával működik, melyet a felhasználási helyen kell létrehozni és címként átadni
- Megszakításalapú működés (Könyvtár IT handlerét kell elhelyezni a TWI ISR-ben)
- Beállítható buffer méretek
- Küldés, fogadás utasítások

A könyvtár működőképes állapotban van, azonban egyelőre vannak olyan jelenségek, melyek bizonyos szituációkban (bizonyos típusú felhasználások esetén) még hibát, lassabb adatátvitelt eredményezhetnek.

Vezérlés

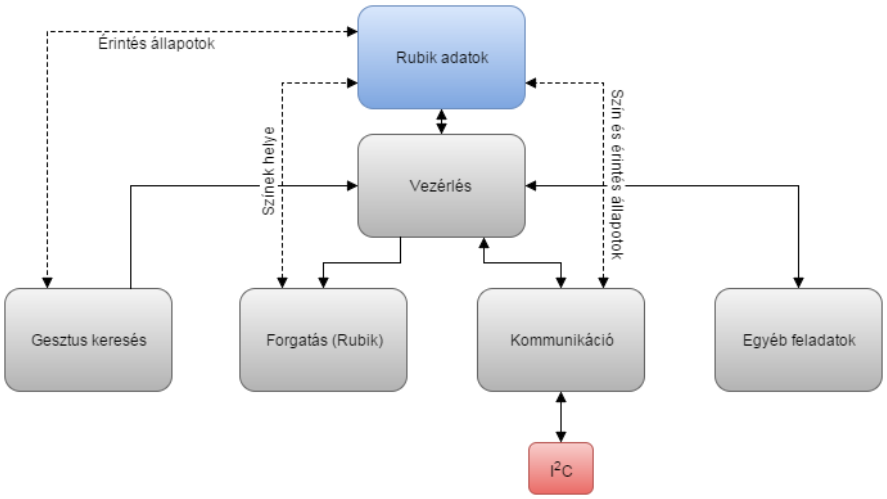
A main-ben megírt vezérlő algoritmus feladata, hogy fogadja az I^2C-n bejövő adatokat, illetve azok alapján elvégzi a szükséges műveleteket és adott esetben elküldi a megfelelő választ. Emellett futtatuja folyamatosan az érintés mintavételezését, hogy lekérdezés esetén friss adat álljon készen.

Vezérlőpanel firmware (MASTER)

Feladatai

- Teljes működés vezérlése
- Oldalak érintés állapotainak folyamatos lekérdezése
- Gesztusok keresése az érintésadatokban
- Kocka állásának forgatása
- Tápvezérlés
- Töltés detektálás
- Akkumulátorszint mérés
- Megjelenítendő színadatok küldése az oldalaknak

Áttekintő blokkdiagramm



A blokkdiagrammon látható, hogy van egy közös adathalmaz, melyet minden taszk globálisan használni tud. Ide két fontos dolog tartozik: az érintés adatokat tároló tömb, a kocka jelenlegi állapotát (színek helyét) tároló tömb.

A MASTER firmware működésének lényege, hogy egy vezérlő algoritmus segítségével, mely a main fájlon belül van implementálva, folyamatosan (~10ms) lekérdezi minden oldal érintésadatát. Ezekben az adatokban keres a program egyik része gesztusokat. Amennyiben talál valahol valamilyen forgatásra irányuló gesztust, a vezérlés átadja azt a forgató algoritmusnak mely elvégzi a szükséges forgatás(oka)t.

Érintésadatok tárolása, gesztuskeresés

Az érintésadatok tárolásánál a ram korlátozott mérete jelenti az egyik legnagyobb nehézséget. Mivel egy gesztus időmenzióban (a tér és a sík mellett) történik, így szükségünk van eltárolni az előző érintés adatokat is. Tehát nem elegendő számunkra az éppen aktuálisan megérintett felületek vizsgálata. Azonban, ha kiszámoljuk, rájövünk hogy elég sok adatot kell eltárolnunk: Ha mondjuk 1s-et akarunk (ez minimum kell): 10ms-ként lekérdezve (ez minimum szükséges) 100 új állapot/másodperc. Mivel a kockánk és ezzel együtt az érintőpadjeink térben vannak, ezért praktikus ezt egy 3D-s tömbben eltárolni, azonban ez egy 5x5x5-ös tömböt jelent. Tehát ha minden új érintésadatnak egy bytet le szeretnénk foglalni, akkor az 5x5x5x100 byte/s. Viszont ennyi területünk nincsen.

A másik probléma az adatokban való gesztuskeresés. Itt pl. olyan gesztusokat szeretnénk keresni, hogy valaki 3-4 felületen végig húzza az ujját (ez egy forgatás). Tételezzük fel, hogy találunk egy ilyen mintát, azonban még azt is el kell döntenünk, hogy ez melyik tengely menti, milyen irányú, melyik sík elforgatását jelenti.

Ezeket a problémákat 2 ötlet együttes alkalmazásával sikerült kiküszöbölnem:

Az idősíknak ne foglaljunk egy új dimenziót a tömbnél, hanem használjuk a bitmélységet. Tehát mivel úgyis csak azt akarjuk eltárolni, hogy az adott felületet épp fogják-e vagy nem, így nem kell ennek egy egész byte, használhatjuk a biteket. Így ha egy 1 byte méretű 5x5x5-ös tömböt foglalunk (az „csak” 125 byte), akkor abban még tudunk időben visszamenőleg 8 állapotot tárolni. Ennek egy másik előnye, hogy így mindig csak elshiftelni kell minden bytet. Így már csak az idősík nagyságát kéne lekicsinyítenünk.

A másik ötlet, hogy ne tároljunk el minden állapotot, csak amikor változás történik és akkor is csak akkor, ha felfutó él. Azért elegendő mindig csak felfutó élt figyelni, mert ha belegondolunk így egy gesztust pontosan érzékelni tudunk. Pl. van 3 felületünk, először megfogják az elsőt (felfutó él), jön a következő (felfutó él) ekkor persze már az elsőnél 0-t shiftelünk be és így tovább.

Tehát ha ezt a két ötletet együtt alkalmazzuk, akkor elfér 125 bytera az összes adat és a mintakeresés is egyszerűbb, mivel csak elshiftelt biteket kell keresnünk.

A **gesztuskeresést** továbbra is az a probléma nehezíti, hogy az adataink térben vannak. Ezt úgy sikerült leegyszerűsítenem, hogy az algoritmus tengelyenként megy végig: minden tengely körüli palástot kiteríti és az így kapott 12x3 méretű mátrixban keres vízszintesen 3 egymás után végig érintett helyet. Ez azért praktikus, mert így oldalak közötti gesztus is lehetséges, illetve könnyű megállapítani mit kell forgatni: a tengely azáltal van meghatározva, hogy épp melyik tengelyt vizsgáljuk, a sík, hogy a mátrix hányadik sorában vagyunk az irány pedig, hogy milyen irányban találtuk meg az adott sorban a gesztust.

Forgatás

A forgatás egy külön algoritmussal, egy függvényként van megvalósítva. Mivel 5x5x5-ös tömbben tároljuk a színeket is, ezért amennyiben nem valamelyik középső síkot akarjuk forgatni akkor a külsőt és az eggyel beljebbit kell forgatnunk.

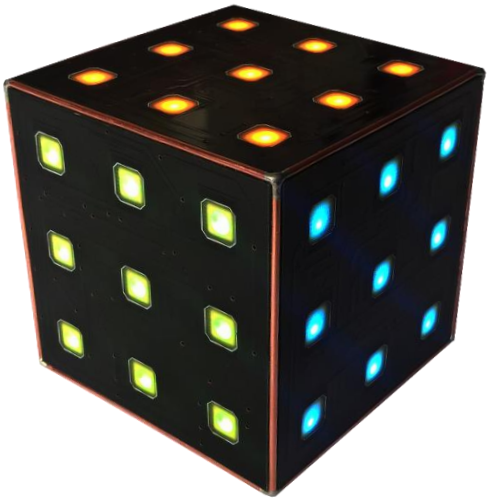
Eredmény

A fejlesztés eredményeképp a tervezett elektronikus Rubik-kocka játszható, bemutatható szinten elkészült, azonban a jelenlegi állapot még nem a teljesen kész rendszert takarja.

Elkészült részek

- Rendszerterv
- Fizikai felépítés terve
- Kocka működéséhez szükséges áramkörök tervei
- NYÁK-ok, amik a kockához lettek tervezve
    - Kapcsolás
    - NYÁK
    - Gyártás
    - Beültetés
    - Élesztés
- A rendszer fizikailag összeillesztve
- Vezérlőpanel táp részének bemérése
- Érintésérzékelés megvalósítása
- Kommunikáció az oldalak és vezérlőpanel között
    - Saját könyvtárakkal
- Oldalpanelek firmware-ének első verziója, mely tartalmazza:
    - Kommunikáció
    - Adatként kapott színek megjelenítés
    - Érintés érzékelés és annak továbbítása
    - Státusz lekérdezhetőség
    - Kalibrációs utasítás
- Vezérlőpanel firmware-ének alap funkciói
    - Kommunikáció
    - Érintés lekérdezés
    - Szín adatok kiküldése
    - Gesztus keresés
    - Forgatás
    - Rendszer összeillesztve tesztelve

Elkészült eszköz kinézete



Irodalomjegyzék

Leírások

Atmel QTouch User Guide
http://www.atmel.com/images/doc8207.pdf

QMatrix Technology White Paper
http://www.atmel.com/images/qmatrix_white_paper_100.pdf

QTouch Schematic and Layout Checklist
http://www.atmel.com/Images/Atmel-42094-QTouch-Schematic-and-Layout-Checklist_ApplicationNote_AT02259.pdf

I^2C standards
http://www.nxp.com/documents/user_manual/UM10204.pdf

Adatlapok

LM27313 (Step-up)
http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=lm27313&fileType=pdf

ATmega328P (MCU)
http://www.atmel.com/images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168a-168pa-328-328p_datasheet_complete.pdf

WS2812B (LED)
https://cdn-shop.adafruit.com/datasheets/WS2812B.pdf

MCP73832 (Töltő IC)
https://www.sparkfun.com/datasheets/Prototyping/Batteries/MCP73831T.pdf