Středoškolská technika 2022

Setkání a prezentace prací středoškolských studentů na ČVUT

URX

Jan Holý a Mikuláš Tycar

Gymnázium Christiana Dopplera

Zborovská 621/45, Praha 5 – Malá Strana

Poděkování

Chtěli bychom poděkovat tátovi od Honzy za cenné rady a skvělé pracovní podmínky, které nám vytvořil a pravidelnou dávku motivace, nejen v průběhu vývoje naší konzole, ale i v průběhu psaní této práce. Dále bych rád poděkoval Mikulášovi za to, jaký je to kamarád a člověk, jsem rád, že jsem ho potkal a věřím, že toho společně ještě hodně dokážeme.

Anotace

Tato práce se zabývá tvorbou vlastní 3D tisknuté open-source studijní/herní konzole URX. Výsledkem práce je produkt, který nabízíme na našich webových stránkách. Produkt jsme se rozhodli zpřístupnit tzv. open source, takže je pro každého nadšence volně stažitelný a použitelný, včetně kódu, 3D modelu. Naše práce obsahuje technický návrh, design, softwarový kód, marketing a končí samotnou finální realizací produktu. Dalším cílem projektu bylo vytvoření edukační pomůcky pro gymnázia, střední školy, integrované školy, případně první semestry VŠ v oborech programování, výpočetní technika, elektrotechnika, elektronické počítače, automatizace, číslicová technika, 3D modelování a další technické předměty.

Klíčová slova

Open-source; Arduino; 3D tisk; Wiring; Blender, Onshape

Annotation

This thesis deals with the creation of a custom 3D printed open-source learning/gaming console URX. The result of the work is a product that we offer on our website. We decided to make the product open source, so it is freely downloadable and usable by any enthusiast, including the code, 3D model. Our work includes the technical proposal, design, software code, marketing and ends with the actual final implementation of the product. Another goal of the project was to create an educational tool for programming schools, high schools, integrated schools, or the first semesters of universities in the fields of programming, computer science, electrical engineering, electronic computers, automation, digital technology, 3D modeling and other technical subjects.

Keywords

Open-source; Arduino; 3D print; Wiring; Blender, Onshape

Obsah

1 ÚVOD

Studijní/herní konzole (dále jen „**konzole“**) jsou tady s námi od roku 1972, To je dlouhá doba, ve které se vystřídaly téměř 4 lidské generace. Mezi tím samozřejmě vznikly spousty a spousty konzolí od stolních, přenosných, až po domácí. Jelikož jsme s herními konzolemi vyrůstali a vyrůstáme, tak jsme se rozhodli, že si takovou studijně-herní konzoli zkusíme postavit.

Během karantény jsme se začali učit s 3D tiskárnou a s mikrořadičem - Arduinem. Tvorba vlastní konzole začala 4. 9. 2021 a dokončili jsme ji 19. 9. 2022.

Náš prvotní cíl byl vytvořit si **studijní - herní konzoli**, kde každý z nás (Honza a Miki) disponuje jinými schopnostmi, rozhodli jsme se, že je zkusíme prolnout s cílem finálního produktu, který bude nejen hezký, ale i použitelný. Nakonec se nám podařilo udělat produkt, který nyní nabízíme k prodeji.

V rámci vývoje jsme si uvědomili, že náš produkt může být skvělou didaktickou pomůckou pro gymnázia, střední školy, integrované školy, případně první semestry VŠ v oborech programování, výpočetní technika, elektrotechnika, elektronické počítače, automatizace, číslicová technika, 3D modelování a další technické předměty.

V této práci se nejdříve seznámíme s hardwarem a softwarem konzole, poté se podíváme na problémy tvorby robustního produktu.

2 URX – HARDWARE

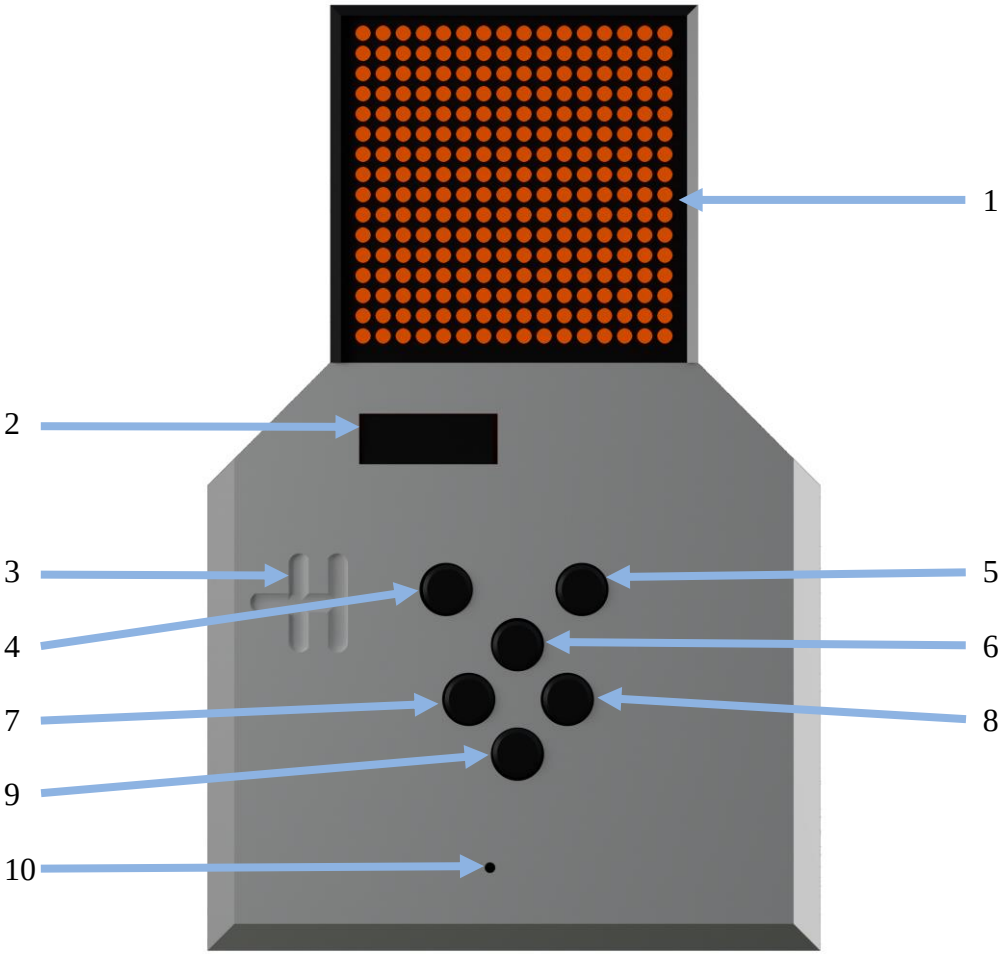
V této kapitole si představíme veškerý hardware od samotného vzhledu, až po návrh plošných spojů.

Veškerý hardware jsme se rozhodli zpřístupnit zdarma jako tzv. open-source hardware.

2.1 URX popis

V kapitole se seznámíme se vzhledem konzole.

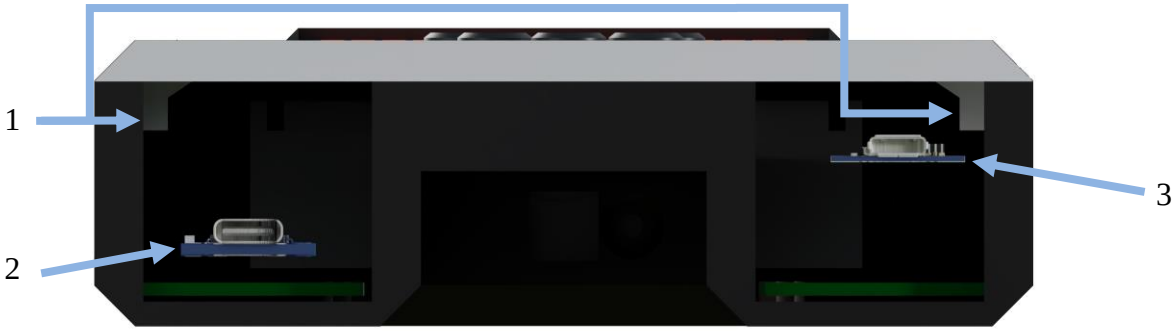
2.1.1 Pohledy z vnějšku



Obr. 1: Konzole URX – pohled zepředu, Render z programu Blender

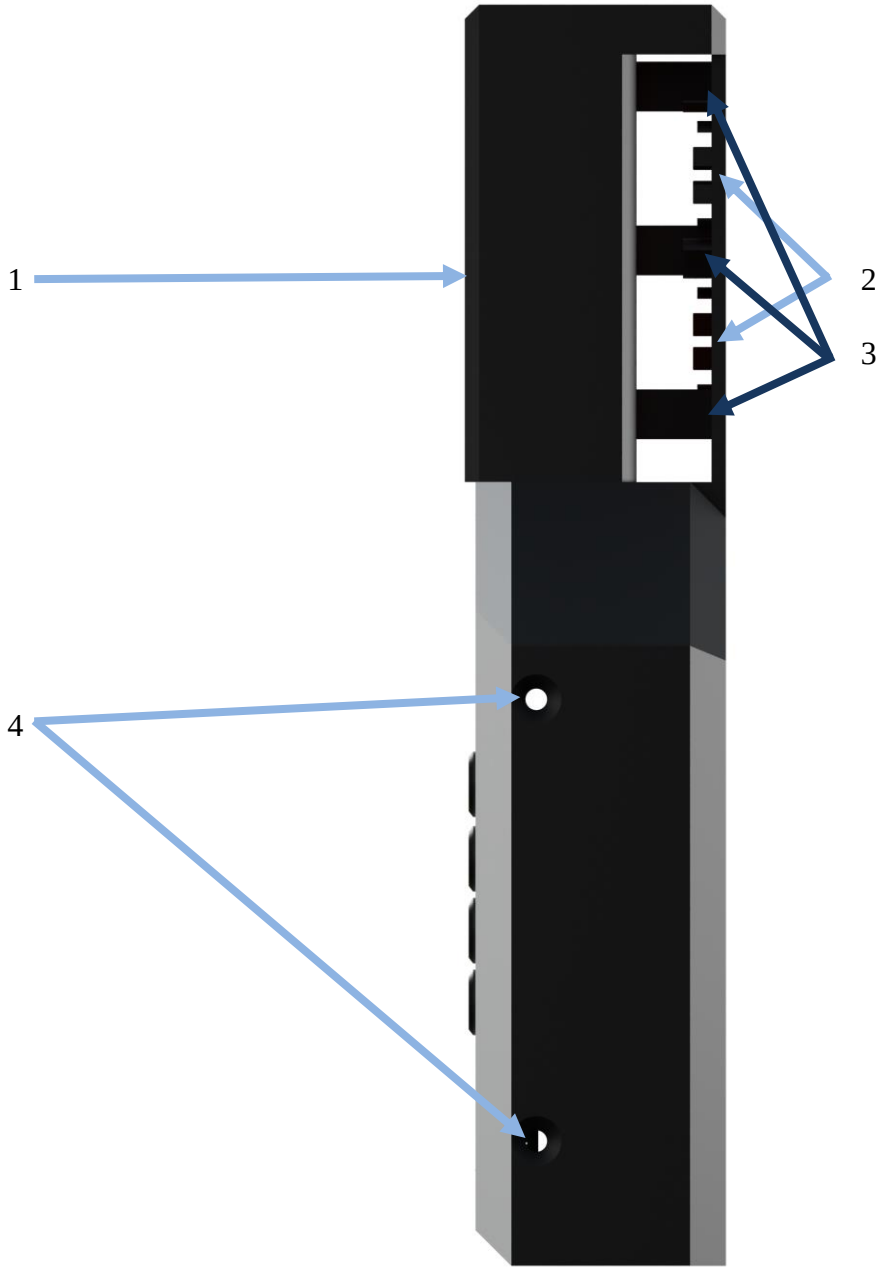
1. LED dot matrix 16x16 displej
2. 0.91" 128x32 OLED displej
3. Naše Logo -
4. Back button (B)
5. Accept button (A)
6. Tlačítko nahoru
7. Tlačítko doleva
8. Tlačítko doprava

9. Tlačítko dolů
10. Otvor pro prostup zvuku z piezoměnič („ reproduktur“)



Obr. 2: Konzole URX – pohled zdola, Render z programu Blender

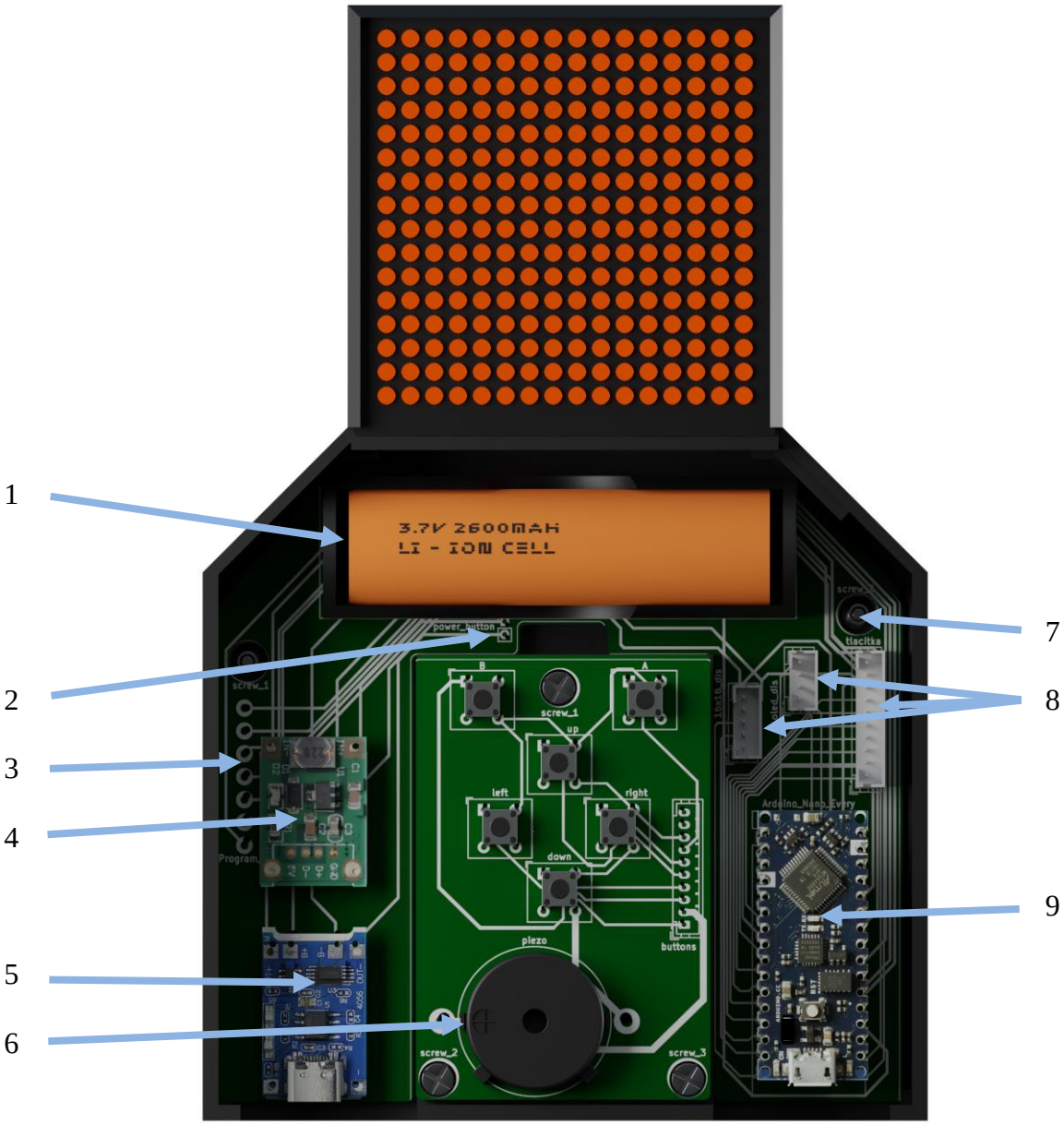
1. 3D tisknuté držáky na horní desce pro přišroubování horní části se spodní částí.
2. Li-ion USB – C charger
3. Arduino nano every – připojovací konektor na programování, update firmware.



Obr. 3: Konzole URX – pohled zprava, Render z programu Blender

1. Ochranný okraj displeje
2. Cvočky na debug kabely – konektory pro připojení dalších libovolných periférií
3. Podpěry pro displej
4. Otvory pro přišroubování horní části konzole ke spodní části

2.1.2 Pohled z vnitřku



Obr. 4: Konzole URX – pohled zevnitř, Render z programu Blender

1. Li-ion baterie (18650) – 2600 mAh
2. Porty pro zapnutí a vypnutí konzole
3. Programovatelné piny
4. Step-up DC-DC měnič
5. Li-ion USB – C charger
6. Piezoměnič („ reproduktor“)
7. Obrácené šroubky, které drží plošný spoj k 3D tisknuté části pomocí matičky
8. JST PH 2,0 mm - konektory
9. Arduino nano every

2.2 Výroba

Modely jsme vyrobili v programu Onshape. Celkem je potřeba 6 modelů na zhotovení konzole: **bot, top, tablet, secure_tablet, power_tablet, battery_holder**. V této kapitole se dozvíte, jak jsme modely tiskli a s jakými problémy jsme se setkali.

Modely jsou nyní ve verzi, které jsou plně přizpůsobené, jak negativním efektům 3D tisku, tak pozitivním efektům 3D tisku. Filament, který jsme vybrali na tisk hlavních částí, je PETG. Volba tohoto typu filamentu je z těchto důvodů:

- Jednoduše se tiskne – nestringuje, má slušnou přilnavost k tiskové podložce, nekroutí se, nemění při tisku velikost a tvar
- Má pro naše použití vhodné fyzikální a chemické vlastnosti – odolné proti potu, minimálně hydroskopický, pružný, vysoká tepelná odolnost.
- Cenově dostupný

Dále používáme filament Filamentum Flexfill 98A, který používáme na tisk ohebné části – tlačítek. Jeho použití je z důvodů pružných vlastností.

Části konzole se tvořily pomocí postupu pokus a omyl. Respektive postupem, který se nedá nikde získat, ale musíte si na něj přijít sami, jedná se o těžko přenosné know-how. Jednou z tisknutelných částí, jejíž realizace byla plná zkoušek a pokusů byla právě tlačítka a podpěry displeje.

2.3 Návrh a výroba plošných spojů

Plošné spoje jsme navrhovali v programu KiCad.

Používáme dva plošné spoje:

1. main board (hlavní deska), - deska pro umístění Arduina, nabíjecího modulu

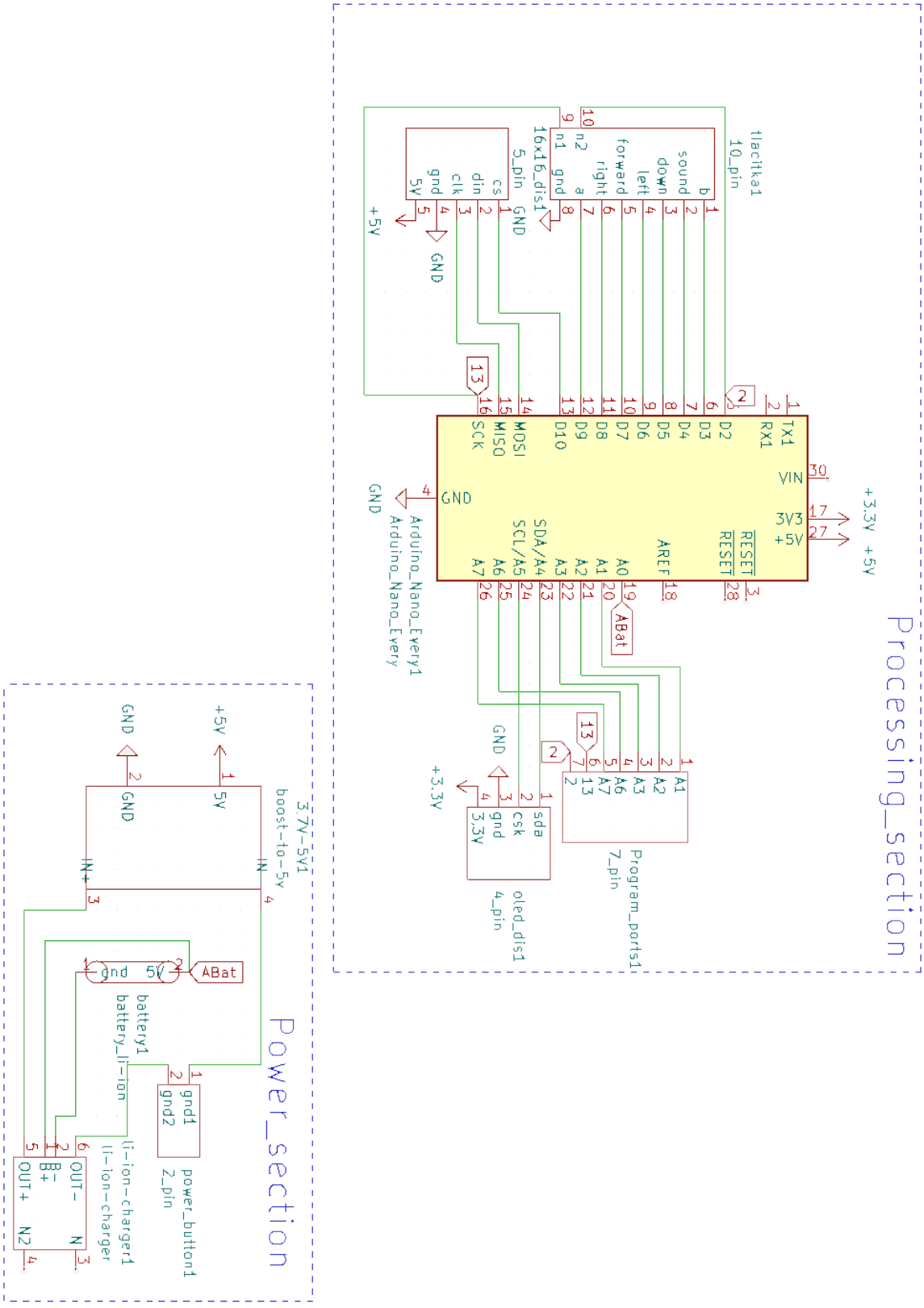
2. button board (tlačítková deska) – deska pro umístění tlačítek ovládacích tlačítek

Kvůli potřebě atypického tvaru plošného spoje jsme se rozhodli si nechat vyrobit plošné spoje v jedné z největších firem pro tvorbu plošných spojů – JLCPCB.

Původně jsme chtěli výrobu plošných spojů nechat na českých firmách, bohužel tyto jsou limitovány použitou technologií a možností výroby plošných spojů pouze ve tvaru obdélníku či čtverce, což pro nás nebylo vyhovující.

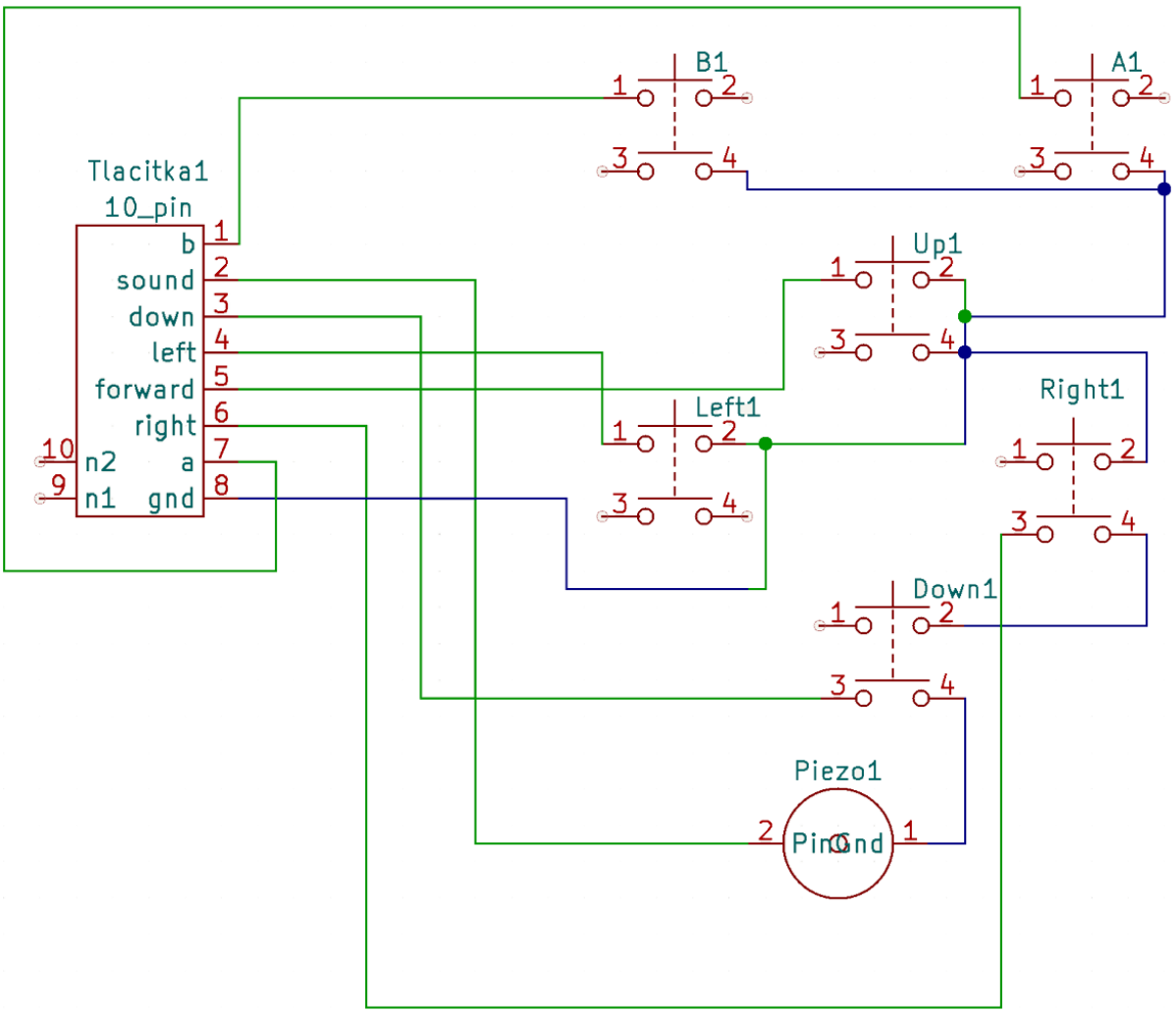
Tvorba prošla spoustou iterací, kvůli tomu, že jsme na začátku neuměli plošné spoje vyrobit, potýkali jsme se s problémy:

- Zrcadlení
- Tloušťka vodivých spojů
- Interference
- Rozměry pájecích plošek
- Rozložení komponent, proti rušení



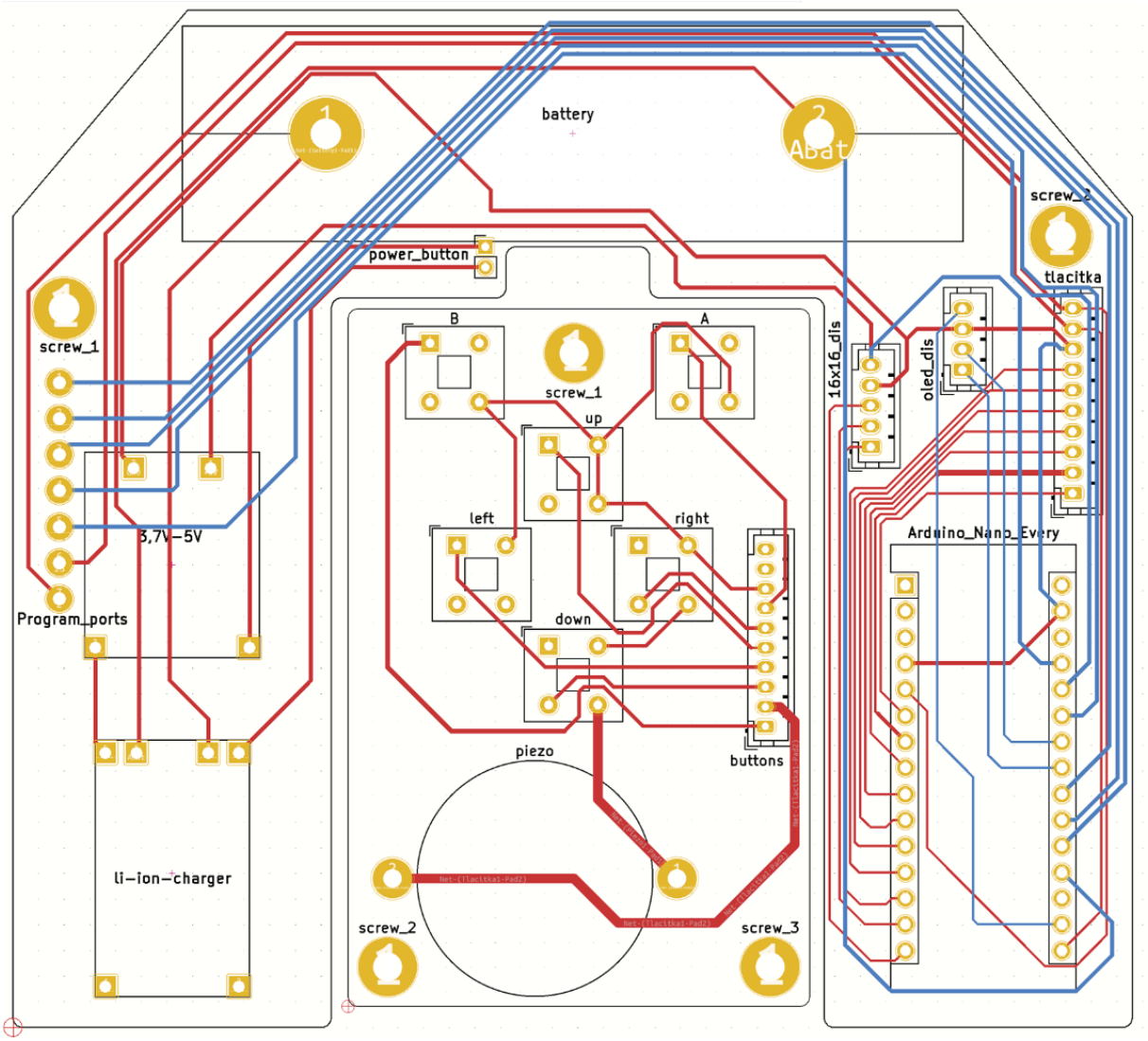
Obr. 5: Schéma main board, KiCad

Na obrázku vidíme použití speciálních typů konektorů JST PH, které jsou na propojení dvou nezávislých částí.



Obr. 6: Schéma button board, KiCad

Na obrázku vidíme jednoduché zapojení tlačítek a piezoměniče („ reproduktoru“) ke konektorům, které se propojí s main board.



Obr. 7: Plošný spoj, KiCad

Zde jdou vidět obě desky. Dále vidíme: červeně horní vrstvu a modře spodní vrstvu. Zde jsme se potkali s problémem rušení u piezoměniče, řešením bylo zvětšení šíře vodivých plošek.

2.4 Specifikace

Tab. 1: Specifikace Arduina nano every

Arduino nano every	
Mikrokontrolér	ATMega4809
Operační napětí	5V
Hodinový takt	20MHz
CPU Flash Memory	48KB
SRAM	6KB

Arduino nano every	
EEPROM	256byte
I^2C	1
SPI	1
UART	1
Interrupts	Všechny digitální piny
DC proud na pin I_P	20mA

Údaje ze stránek Arduina: https://store.arduino.cc/products/arduino-nano-every

Tím, že má Arduino nano every pouze jednu I^2C a jednu SPI komunikaci, tak jsou oba displeje jiné komunikace: Led dot matrix displej – SPI, OLED 0,91“ – I^2C.

Tab. 2: Specifikace Led dot matrix 8x8 displeje

LED dot matrix 8x8 (v konzoli jsou 4)	
Forward Current I_F	25mA
Peak Forward Current I_{PF}	150mA
Reverse Voltage V_R	5V

Údaje z datasheet: https://cdn-shop.adafruit.com/datasheets/BL-M07C881.PDF

Tab. 3: Specifikace Step-up DC-DC měniče

Step-up DC-DC měnič	
Vstupní napětí V_I	1V~5V
Výstupní napětí V_O	5V
Dissipation Power I_D	600mA

Údaje ze stránek Aliexpress:
https://www.aliexpress.com/item/4001101711737.html?spm=a2g0o.order_list.0.0.21ef1802zovYiK

Tab. 4: Výdrž baterie

Výdrž baterie v klidu	**Výdrž baterie v zátěži**
I_K – Proud v klidu t_k – výdrž v klidu (v hodinách)	I_Z – Proud v zátěži t_z – výdrž v zátěži (v hodinách)
$I_K = I_F*4 + I_P*13 + I_D$	$I_Z = I_{PF}*4 + I_P*13 + I_D$
$I_K = 100 + 260 + 600$	$I_Z = 600 + 260 + 600$
$I_K = 960mA$	$I_Z = 1460mA$
$t_k = 2600mAh/I_K$	$t_z = 2600mAh/I_Z$
$t_k = 2,7h$	$t_z = 1,78h$

Tab. 5: Specifikace Li-ion USB – C charger

Li-ion USB – C charger	
Proud	1A
Vstupní napětí	4,5V-5,5V
Baterie plně nabita	4,2V ± 1%

Údaje ze stránek Aliexpress:
https://www.aliexpress.com/item/4001196670805.html?spm=a2g0o.order_list.0.0.21ef1802vfbqA9

2.4.1 Problémy s hardwarem

Museli jsme vyřešit 3 hlavní problémy, které se týkaly hardwaru:

- LED displej píská,
- Li-ion USB-C charger se při nabíjení zahřívá,
- Mikrospínače se dají při nadlehčení zmáčknout bez toho, aby tlačítko cvaklo.

Ad 1) Problém řešíme tím, že jsme v nastavení vytvořili změnu jasu. Při této změně dochází k výrazné redukci rušivého pískání. Samozřejmě s důsledkem, že na intenzivním světle se citelně zhorší viditelnost.

Ad 2) Problém jsme vyřešili změnou materiálu z PLA, které začíná měnit své mechanické vlastnosti (měknout) mezi 50 - 60 °C na materiál PETG, které začíná měknout až u teploty kolem 85 °C. Dále jsme udělali výraznou změnu v návrhu konzole, tuto jsme otevřeli, aby lépe větrala. Nakonec jsme udělali testy, které naleznete níže (Tab. 6).

Ad 3) Problém jsme řešili změnou firmwaru => podařilo se nám výrazně eliminovat, ale stále je v řešení.

Tab. 6: Měření teplot na konzolích – nabíjecí proces

URX 0.4				**URX 1 - černý**		
Čas	Baterie [°C]	Arduino [°C]	Li-ion[1] [°C]	Baterie [°C]	Arduino [°C]	Li-ion[1] [°C]
9:30	22,7	23,5	22	22,6	23,2	22
10:00	27,3	27,4	45	25,3	25,1	35,3
10:30	28,9	28,1	40	27,4	25,9	34,3
11:00	27,3	29,6	37,9	25,1	26,9	31,7
11:30	28,9	27,4	33,6	25,3	24,4	30
12:00	26	26,6	31,6	25,1	24,3	28,5
URX 1 – oranžová (otevřená)						
Čas	Baterie [°C]	Arduino [°C]	Li-ion[1] [°C]			
9:30	22	23	X			
10:00	26,5	24,4	X			
10:30	28,9	25,5	X			
11:00	26,4	25,3	X			
11:30	25,8	24,1	X			
12:00	25	24	X			

[1]*Li-ion USB – C charger. Měření probíhalo pomocí bezkontaktního digitálního teploměru. Hodnoty v tabulce jsou průměr z 3 měření, kdy se konzole úplně vybili a poté nabili.*

3 URX – SOFTWARE

Software jsme programovali v programovacím jazyku Wiring což je upravená verze C++. Knihovnu jsme programovali v čistém C a C++.

V konzoli se pro ukázku jejich možností nachází 3 hry:

- SNAKE (had)
- GALAXIAN
- DANCE MAN
- Dále se v konzoli nachází sekce nastavení, které obsahuje možnost změny intenzity displeje, zapnutí/vypnutí zvuku, obtížnosti a high score.

Celý kód (stejně jako konzoli) jsme se rozhodli nabídnout jako open-source a umístili jsme ho na platformu GitHub, takže ho může kdokoliv libovolně doprogramovat, nebo rovnou přeprogramovat celou konzoli.

V této kapitole Vám ukážeme celý kód hada a vysvětlíme, jak funguje.

3.1 Ukázka Hada

Kód 1: Ukázka hada

```
//Main function for game Snake
void Snake() {
  //Local variables Snake
  allTime = millis();
  //Movement: left....x = 1, right....x = -1, forward....y = 1, down....y = -1.
  if ((mode == 1 or mode == 2) and active == 1) {
    noTone(soundPin);
    int mod = mode;
    SettingGameSnake();
    mode = -mod;
    displayOled("Your Score: " + String(bodyLength - 3 + points) + "\nHigh Score: " + String(read2EEPROM(addressSnake1)));
  } else if (mode == 1 and y != 0) {
    displayOled("Snake SUPER\nFor start press A");
    displayLed(foodY, foodX, 0);
    GenerateFood();
    foodXSuper = foodX;
    foodYSuper = foodY;
    GenerateFood();
    wave = -1;
    mode = 2;
    y = 0;
    delay(100);
  } else if (mode == 2 and y != 0) {
    displayOled("Snake CLASSIC\nFor start press A");
    mode = 1;
    y = 0;
    delay(100);
  }
  if ((x == 0 and y == 0) and startingLedOn == true and abs(mode) != mode) {
    displayLed(shiftedY, shiftedX, 1);
    //Serial.print()
  } else if (startingLedOn == true and abs(mode) != mode) {
    displayLed(shiftedY, shiftedX, 0);
    startingLedOn = false;
  }
  //Turning Snake ON
  if (isGameOn == 1 and (x != 0 or y != 0)) {
    isGameOn = 0;
    lastTime = allTime;
  }
  if (active == -1) {
    goThrough = true;
    isGameOn = 1;
    active = 0;
    allwaysX = 0;
    allwaysY = 0;
    x = 0;
    y = 0;
    displayOled("Your Score: " + String(bodyLength - 3 + points) + "\nHigh Score: " + String(read2EEPROM(addressSnake1)));
    GameOver(bodyLength - 3 + points, 1);
  }
  if (isGameOn == 0 and active == 1 and startingLedOn == false) {
    isGameOn = 1;
    y = 0;
    x = 0;
    noTone(soundPin);
  }
  if (((bodyLength - 4 + points) >= read2EEPROM(addressSnake1) and congratsState == true) and congrats <= 3) {
    PlaySound(9, congrats);
    congratsState = false;
  }
```

```
if (isGameOn == 0) {
  // Smart blinking food led
  if (mode == 2 or wave == -2 and timer != 0) {
    displayLed(foodYSuper, foodXSuper, millis() % 100 < 50 ? 1 : 0);
    displayLed(foodY, foodX, millis() % 1000 < 500 ? 1 : 0);
  } else {
    displayLed(foodYSuper, foodXSuper, 0);
    displayLed(foodY, foodX, millis() % 1000 < 500 ? 1 : 0);
  }
  // This will run every delayTime
  if (allTime - lastTime >= delayTime) {
    if (abs(mode) != mode) {
      if (allwaysY != -y and y != 0) {
        allwaysY = y;
        allwaysX = 0;
      } else if (allwaysX != -x and x != 0) {
        allwaysX = x;
        allwaysY = 0;
      }
    } else {
      int y1 = random(-1, 2);
      int x1 = random(-1, 2);
      if (allwaysY != -y1 and y1 != 0) {
        allwaysY = y1;
        allwaysX = 0;
      } else if (allwaysX != -x1 and x1 != 0) {
        allwaysX = x1;
        allwaysY = 0;
      }
    }
    // Making head bigger
    shiftedY -= allwaysY;
    shiftedX -= allwaysX;
    Warp();
    if (Array[shiftedY][shiftedX] > 0) {
      displayOled("Your Score: " + String(bodyLength - 3 + points) + "\nHigh Score: " + String(read2EEPROM(addressSnake1)));
      GameOver((bodyLength - 3 + points), 1);
    }
    if (allwaysX != 0 or allwaysY != 0) {
      Array[shiftedY][shiftedX] = bodyLength + 1;
    }
    //This will run only if game snake is on - if you are not dead
    if (gameState == 2) {
      //Big for loop that run for all pixel in array
      for (int row = 0; row < 16; row++) {
        for (int col = 0; col < 16; col++) {
          // if there is a body segment, decrement it's value
          if (Array[row][col] > 0 ) {
            Array[row][col]--;
            //If This pixel is zero (I know that zero means that it is snake ending) turn pixel off
            if (Array[row][col] == 0) {
              displayLed(row, col, 0);
              //If pixel is the biggest turn pixel on
            } else if (Array[row][col] == bodyLength) {
              displayLed(row, col, 1);
            }
          }
        }
      }
      if (soundDelay == 0 and (allwaysX != 0 or allwaysY != 0)) {
        if (wave == -2) {
          PlaySound(11);
```

```
    } else {
      PlaySound(1);
    }
  }
  //If snake eated the food
  if (Array[shiftedY][shiftedX] == Array[foodY][foodX] && Array[shiftedY][shiftedX] != 0) {
    noTone(soundPin);
    PlaySound(2);
    if (abs(mode) != mode) {
      bodyLength += 1;
      delayTime -= 0.5;
    }
    foodX = -1;
    foodY = -1;
    GenerateFood();
    if (wave == 0 and mode == -2) {
      wave = -2;
      foodXSuper = foodX;
      foodYSuper = foodY;
      GenerateFood();
      timer = 20;
      displayOled("Super food: " + String(timer));
    } else if (wave != -2) {
      wave--;
    }
    if (abs(mode) != mode and wave != -2) {
      displayOled("Your Score: " + String(bodyLength - 3 + points) + "\nHigh Score: " + String(read2EEPROM(addressSnake1)));
    }
  }
  if (Array[shiftedY][shiftedX] == Array[foodYSuper][foodXSuper] && Array[shiftedY][shiftedX] != 0) {
    noTone(soundPin);
    PlaySound(3, 0);
    for (int row = 0; row < 16; row++) {
      for (int col = 0; col < 16; col++) {
        //Making Snake timer longer
        if (Array[row][col] > 0 ) {
          Array[row][col] += 2;
        }
      }
    }
    timer = 0;
    if (abs(mode) != mode) {
      bodyLength += 2;
      delayTime -= 0.5;
    }
    points++;
    foodYSuper = -1;
    foodXSuper = -1;
  }
  if (wave == -2 and timer != 0) {
    displayOled("Super food: " + String(timer + 1));
    timer--;
  } else if (wave == -2) {
    displayLed(foodYSuper, foodXSuper, 0);
    foodYSuper = -1;
    foodXSuper = -1;
    wave = random (2, 5);
    displayOled("Your Score: " + String(bodyLength - 3 + points) + "\nHigh Score: " + String(read2EEPROM(addressSnake1)));
  }
  //Resetting timer
  lastTime = allTime;
  if (abs(allwaysX)==allwaysX and abs(allwaysY)==allwaysY) {
```

```
            bool bytes[5] = {allwaysY,0,0,allwaysX,0};
            transmit(bytes);
          } else {
            bool bytes[5] = {0,abs(allwaysY),abs(allwaysX),0,0};
            transmit(bytes);
          }
          delay(100);

        }
      }
    }
}
```

3.2 Časy

V hadu jsou řešeny časy dvěma způsoby:

První způsob počítání času od zapnutí Arduina

- Pomocí prvního způsobu je řešen main loop Hada ve kterém probíhá celá hra (strana 19 – if (allTime - lastTime >= delayTime)).
- V proměnné allTime (strana 18 – nahoře) je pomocí funkce millis() uložen čas od zapnutí Arduina a poté pomocí proměnné lastTime (strana 19 – nahoře), kam se ukládá minulá hodnota allTime a nakonec pomocí proměnné delayTime, kde je uložen čas, kdy se má zpustit hlavní program.

Druhý způsob pomocí funkce delay(), která zastaví běh programu na určitý počet milisekund.

3.3 Módy

Hra had má 2 módy.

- Snake ClASSIC – mód, který vychází z originálního hada na Nokii 1997,
- Snake SUPER – mód ve kterém se nachází nový druh jídla, který bliká, má časový limit a zvětší hada o 2 body a přidá skóre o 3.

V kódu je to řešené pomocí proměnné mode.

3.4 Zvuk

Zvuk je řešen pomocí piezoměniče, do kterého nahráváme PWM signál, pomocí funkce tone(). Samotná funkce tone() je schovaná ve funkci PlaySound(), která usnadňuje práci se zvukem viz str. 22.

Kód 2: Funkce PlaySound

```
void PlaySound(int sound, int i = 0) {
  if (soundState == true and abs(mode) != mode) {
    if (sound == 1) {
      tone(soundPin, 70);
      soundDelay = 100;
    } else if (sound == 11) {
      tone(soundPin, 150);
      soundDelay = 100;
    } else if (sound == 2) {
      tone (soundPin, 260);
      soundDelay = 250;
    } else if (sound == 3) {
      tone (soundPin, 500);
      soundDelay = 150;
    } else if (sound == 4) {
      noTone(soundPin);
      tone (soundPin, 170);
      soundDelay = 100;
    } else if (sound == 5) {
      tone (soundPin, 90);
      soundDelay = 90;
    } else if (sound == 9) {
      const float noty[5] = {130.813, 164.814, 195, 998, 329.621};
      tone(soundPin, noty[i]);
      soundDelay = 200;
    }
  }
}
```

Funkce PlaySound()

3.5 Vykreslování na OLED displej

Vykreslování na OLED displej je umožněné pomocí funkce displayOled() do které stačí napsat text (nový řádek pomocí symbolu‘\n‘) viz str 23.

Kód 3: Funkce displayOled

```
void displayOled(String writing = "") {
  u8g2.firstPage();
  do {
    int index = writing.lastIndexOf('\n');
    int length = writing.length();
    String first = writing.substring(0, index);
    u8g2.setFont(u8g2_font_helvR12_tr);
    u8g2.setCursor(0, 13);
    u8g2.print(first);
    u8g2.setCursor(0, 28);
    index = writing.lastIndexOf('\n');
    length = writing.length();
    String second = writing.substring(index, length);
    u8g2.print(second);
  } while ( u8g2.nextPage() );
}
```

Funkce displayOled

3.6 Vykreslování na LED displej

Vykreslování na LED displej je umožněné pomocí 3 funkcí:

- První funkce: displayLed() rozsvítí LED pomocí kartézské soustavy,
- Druhá funkce: displayClear() funkce zhasne celý displej,
- Třetí funkce: displayImage() dokáže pomocí 16x16 nebo 8x8 (binární pole) pole vykreslit obrázek viz níže.

```
void displayLed(int row, int col, int state) {
  if (row < 8 and col < 8) {
    lc.setLed(1, 7 - col, row, state);
  } else if (row < 8 and col >= 8) {
    lc.setLed(3, 7 - col % 8, row, state);
  } else if (row >= 8 and col < 8) {
    lc.setLed(0, 7 - col, row % 8, state);
  } else if (row >= 8 and col >= 8) {
    lc.setLed(2, 7 - col % 8, row % 8, state);
  }
}
```

Kód 4: Funkce displayLed

Funkce displayLed()

Kód 5: Funkce displayClear

```
void displayClear() {
  for (int index = 0; index < 4; index++) {
    lc.clearDisplay(index);
  }
}
```

Funkce displayClear()

Kód 6: Funkce displayImage

```
void displayImage(const word* image, int displaySizeX = 8, int displaySizeY = 0, int display = 0, int offsetX = 0, int offsetY = 0) {
  int MirrorX = displaySizeX - 1;
  int MirrorY = displaySizeY - 1;
  for (int index = 0; index < 2; index++) {
    for (int rows = 0; rows < displaySizeX; rows++) {
      if (displaySizeX == 8 and displaySizeY == 0) {
        byte byte1 = (pgm_read_word(&image[(rows) + (index * displaySizeX)]) >> displaySizeX) & 0xFF;
        byte byte2 = pgm_read_word(&image[(rows) + (index * displaySizeX)]) & 0xFF;
        lc.setRow(index + 1 + index, MirrorX - rows, byte1);
        lc.setRow(index + index, MirrorX - rows, byte2);
      } else {
        for (int cols = 0; cols < displaySizeY; cols++) {
          lc.setLed(display, MirrorX - rows - offsetX, MirrorY - cols - offsetY, bitRead(pgm_read_word(&image[rows]), cols));
        }
      }
    }
  }
}
```

Funkce displayImage()

3.7 Ukládání dat do EEPROM

Pro ukládání dat trvale, jako jsou skóre, nastavení zvuku nebo intenzita displeje používáme zabudovanou EEPROM, která má 256 bytů.

K čtení 1 bytu dat používáme knihovnu od Arduina EEPROM.h, kde využíváme funkci EEPROM.read() a pro čtení 2 bytů používáme funkci, která rozpůlí číslo a uloží ho do dvou adres viz níže.

Kód 7: Funkce read2EEPROM

```
int read2EEPROM(int address) {
  byte byte1 = EEPROM.read(address);
  byte byte2 = EEPROM.read(address + 1);
  int result = (byte1 << 8) + byte2;
  return result;
}
```

Funkce read2EEPROM()

4 Z PROJEKTU NA PRODUKT

Hlavní výzva projektu byla udělat konzoli ze stavu „jsme rádi, že to funguje," na stav prodejného produktu. Tento proces nám zabral nejvíce času.

Pár základních věcí co jsme museli předělat:

1. jednoduchost opravy konzole – tedy tak, aby to zvládli nejen odborníci, ale i studenti a začínající nadšenci.

2. jednoduchost předělání hardwaru – myšlenou je právě možné využití pro studium, kdy jednoduchou změnou komponent či rozložení, mohou být rozšířeny možnosti konzole.

3. jednoduchost rozložení na komponenty a složení – prostě aby to dokázal složit a opravit opravu každý

4. tisknutelnost na různých 3D tiskárnách – na trhu je nyní nepřeberné množství výrobců tiskáren, model je navržen tak, aby bylo možné tisknout na libovolné FDM tiskárně

5. srozumitelný kód – tedy optimalizovat zdrojový kód tak, aby se v něm vyznal i začínající kodér.

4.1 Tým

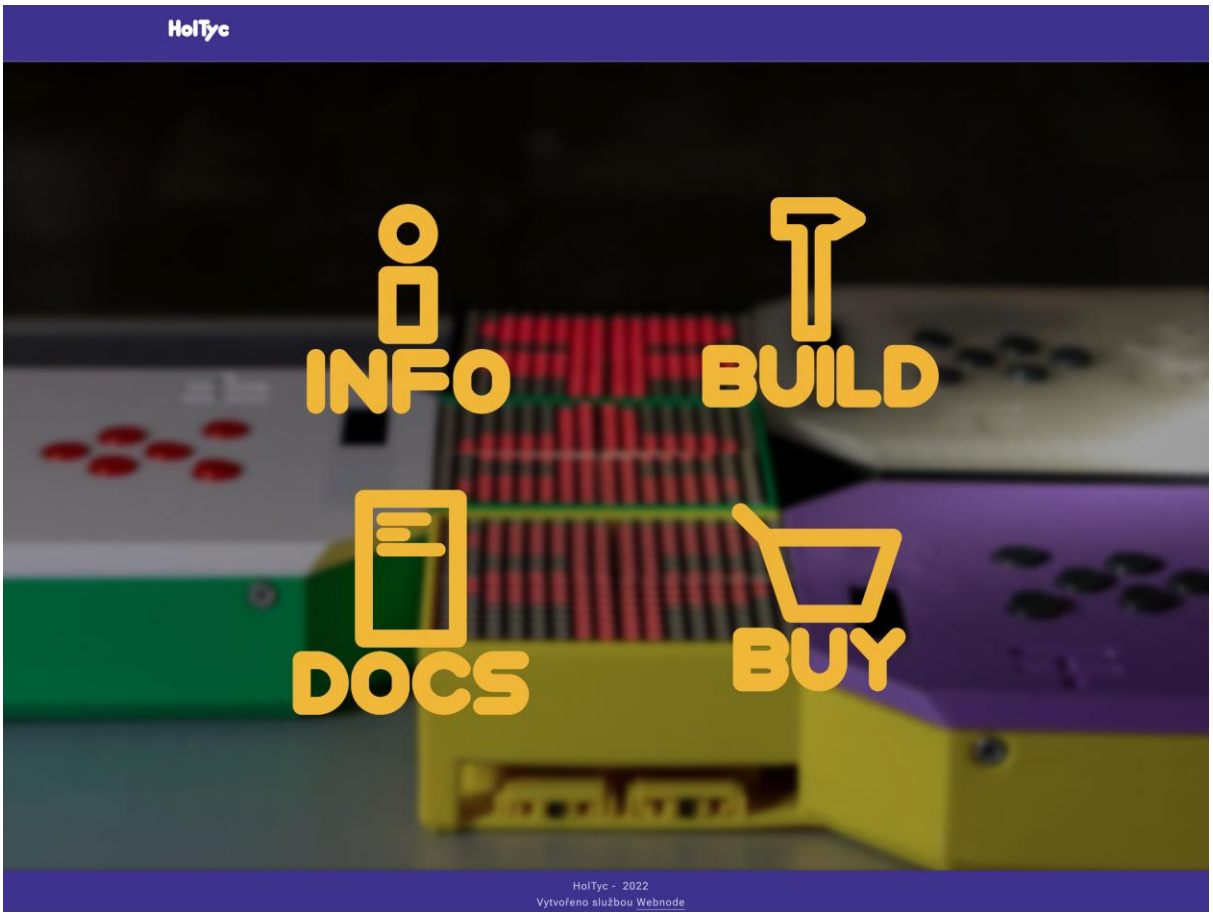
Na začátku se náš tým skládal z Jana Holé a Mikuláše Tycara. K dnešnímu dni se náš tým skládá ze 7 lidí – Jan Holý, Mikuláš Tycar, Štěpán Holý, Adam Ivaniškin, Tobiáš Kopecký, Sevastian Sokolov, Tomáš Jodl.

Práce jednotlivých lidí:

- Jan Holý – vývoj, vedoucí týmu
- Mikuláš Tycar – design, vedoucí týmu
- Štěpán Holý – pájení, tvorba konzolí
- Adam Ivaniškim – marketing
- Tobiáš Kopecký – pájení, tvorba konzolí
- Sevastian Sokolov – marketing
- Tomáš Jodl – programování

4.2 Web

V rámci vývoje došlo k vytvoření vlastních webových stránek. Stránky jsme vytvořili pomocí nástroje Webnode: www.holtyc.com.



Obr. 8: Webové stránky

Na webových stránkách nabízíme 2 varianty konzole: 1. složená konzole, 2. rozložená konzole.

4.3 Varianta 1

Nabízenou variantu prodáváme za 2800 Kč. Balení obsahuje složenou konzoli URX, brožurku a limitovaný box viz kapitola Marketing.

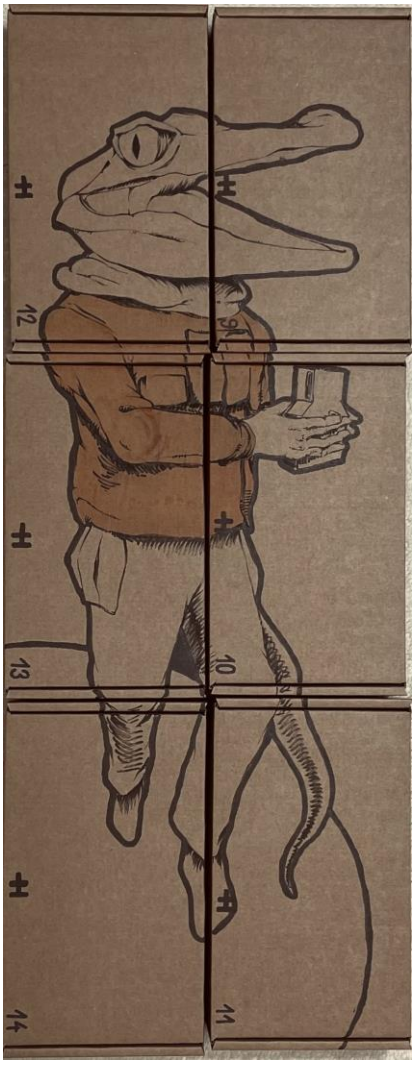
4.4 Varianta 2

Nabízenou variantu prodáváme za 1500 Kč. Balení obsahuje rozloženou konzoli URX. V balení chybí dvě hlavní části tisku. To je hlavní položka v redukci ceny. Umožňuje samozřejmě každému, aby si tiskl konzoli v libovolných barevných kombinacích. V příloze naleznete návod, jak složit konzoli URX.

4.5 Marketing

Marketing je základ úspěšného produktu. Nejsme marketéři, a proto zkoušíme metody, které si myslíme, že by mohly fungovat: 1. založili jsme si profil na sociálních médiích (Youtube, Instagram, TikTok), 2. Jsme pro první variantu vytvořili limitované boxy, které jsou v sérii po 4-6 viz níže.

Obr. 9: 2. série limitovaných boxů.



Obr. 10: 3. série limitovaných boxů

5 ZÁVĚR

Cílem této práce bylo vytvořit prodejný produkt – studijní/herní konzoli URX.

Vývoj probíhal v následujících etapách:

1. Testování a tvorba kódu hada – 4 měsíce.
2. Vytváření nového designu konzole a webu – 3 měsíce
3. Dokončení designu – nové plošné spoje, jednoduchá modularita, marketing zvětšování týmu – 5 měsíců

Podařilo se nám vytvořit konzoli URX. Celý vývoj nám zabral přibližně 12 měsíců. Veškeré náklady vyšly na 16 583 Kč. Prodali jsme 3 kusy v první variantě a získali jsme 8 900 Kč.

Výsledný produkt je zcela ideální pro využití v rámci vzdělávání a to nejen nadšencům mimo školský systém, tedy tzv. bastlířům, ale zejména v rámci školního vzdělávání.

Výhody a přednosti vidíme následující:

1) Rozvoj schopností studentů v oborech: 3D modelování, programování, elektronika, řídící technika, automatizace
2) V rámci integrovaného školství – učební obory – možnost výuky pájení, osazování součástek, leptání a tvorba desky plošných spojů a následné oživení celého systému pomocí připraveného kódu (tento dáváme zdarma v licenci open-source)
3) Práce s mikrořadiči – pochopení jednotlivých částí počítače ve zmenšeném vydání
4) Výuka programování přímo metodou Jana Ámose Komenského – tedy „ škola hrou“ = teorie rovnou spojená s praktickou a vizuální ukázkou, možnost přímé interakce LED, piezoměniče, řada volných vstupních/výstupních portů
5) Připravené debug konektory, pro okamžitou možnost rozšíření možností konzole o instalaci doplňkových periférií – větráčky, další displeje, možnosti propojení konzolí
6) Možnosti rozšíření a libovolné modifikace komponent, plošné desky, 3D modelu, kódu
7) Volná licence pro využití komukoliv, kdo bude mít zájem

8) V rámci naší představy: ideální jako semestrální projekt v předmětech – elektronické systémy, elektronika, výpočetní technika, programování, modelování atd. Kdy si studenti mohou vyzkoušet modelování v CAD programech, dále tisk na 3D tiskárnách typu FDM (FFF) a následně s osazováním součástek na desku plošných spojů. Dále připojení mikrořadiče s možností dohrání vhodného kódu, který dáváme v současné plně funkční podobě zdarma (jako open-source).

9) Produkt nabízí nepřeberné množství využití v rámci vzdělání a věříme, že v případě použití bude mí slušný úspěch.

Do budoucna doufáme ve vytvoření nezávislé komunity, která umožní rozvoj a vývoj projektu – využíváme aktivně platformu Discord. Zdrojový kód jsme dali volně ke stažení přes GitHub - https://github.com/Hanzalt/Urx_documentation/wiki

Plánujeme vytvořit konzoli URX 2 s lepším monochromatickým LCD displejem s vyšším rozlišením.

V rámci našeho projektu vznikl základ pro naši budoucí firmu HOLTYC viz. web www.holtyc.com

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PCB	Printed Circuit Board – Deska plošných spojů
EEPROM	Electrically Erasable Programmable Read-only Memory (paměť)
LED	Light-Emitting Diode
USB	Universal Seriál Bus (komunikační protokol)
PETG	Polyethylen terephthalate glycol (plast)
PLA	Polylaktidová vlákna (plast)
I^2C	Inter-Integrated Circuit (komunikační protokol)
SPI	Serial Peripheral Interface (komunikační protokol)
UART	Universal asynchronous receiver-transmitter (komunikační protokol)
FLASH	Paměť s libovolným přístupem zapisovatelná (použití uložení firmware)
LI-ION	Lithium-iontový akumulátor
I	Proud
V	Napětí
t	Čas

POUŽITÁ LITERATURA

1. HW Knihovny C++ - https://github.com/wayoda/LedControl

2. HW Knihovy C++ - https://github.com/olikraus/u8g2

SEZNAM OBRÁZKŮ, TABULEK A KÓDU PROGRAMU

PŘÍLOHA 1: SLOŽENÍ A ROZLOŽENÍ KONZOLE

Rozložení

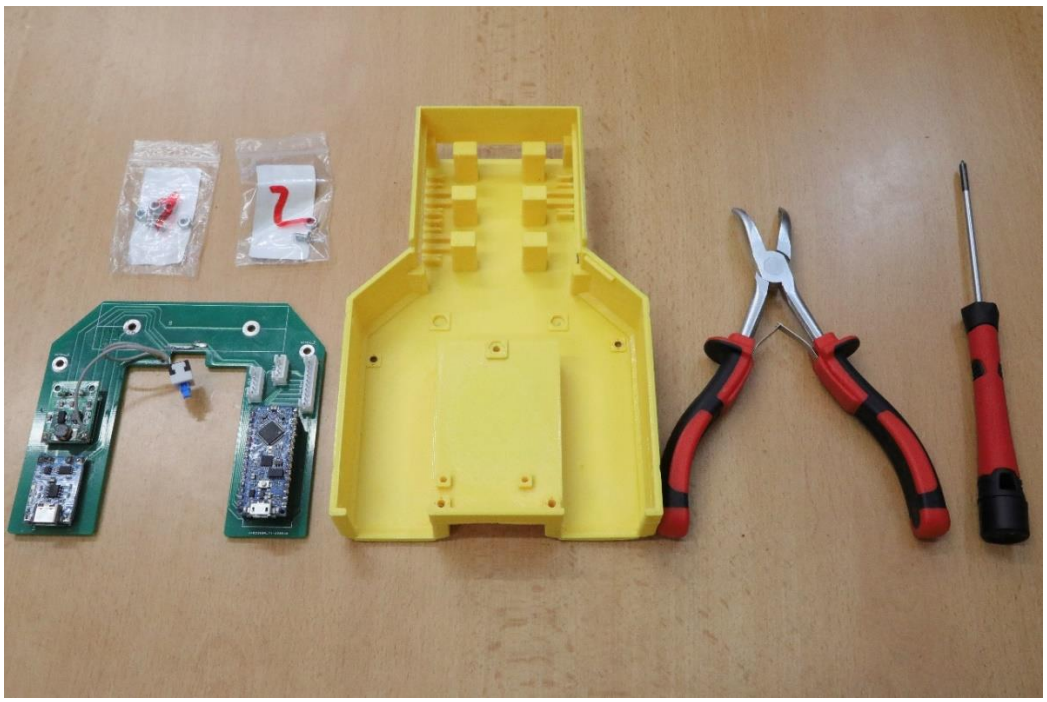
Potřebné věci: Plochý šroubovák, Křížový šroubovák

1. Odšroubujte vršek a opatrně ho zvedněte.
2. Odpojte OLED displej od hlavní desky.
3. Pomocí plochého šroubováku vyndejte přepážku u OLED displeje. Lehoučce prstem zatlačte na OLED displej a pomalinku ho vytahujte. NÉ ZA KABELY.
4. Odpojte velký displej a tlačítka od hlavní desky.
5. Vyšroubujte tlačítkovou desku a opatrně ji dejte stranou, tak aby se neutrhli programovatelné kabely (female kabely zboku).
6. Vyndejte matici, která se nachází pod horním šroubem tlačítkové desky
7. Vyndejte baterku pomocí plochého šroubováku.
8. Vyšroubujte držák na baterku a dejte ho stranou.
9. Pomocí plochého šroubováku vycvakejte zácvaky u programovatelných kabelů (female kabely zboku), poté tyto kabely vyndejte pomocí plochého šroubováku.
10. Vyndejte velký displej.
11. Tlačítkovou desku dejte stranou.
12. Odšroubujte šroub, který drží zapínací tlačítko. POZOR NA MATIČKY PŘIDRŽUJTE SI JE PRSTEM.
13. Vytáhněte zapínací tlačítko
14. Odšroubujte hlavní desku. POZOR NA MATIČKY PŘIDRŽUJTE SI JE PRSTEM.
15. Pomocí plochého šroubováku vyndejte matičky, které jsou pod deskou.

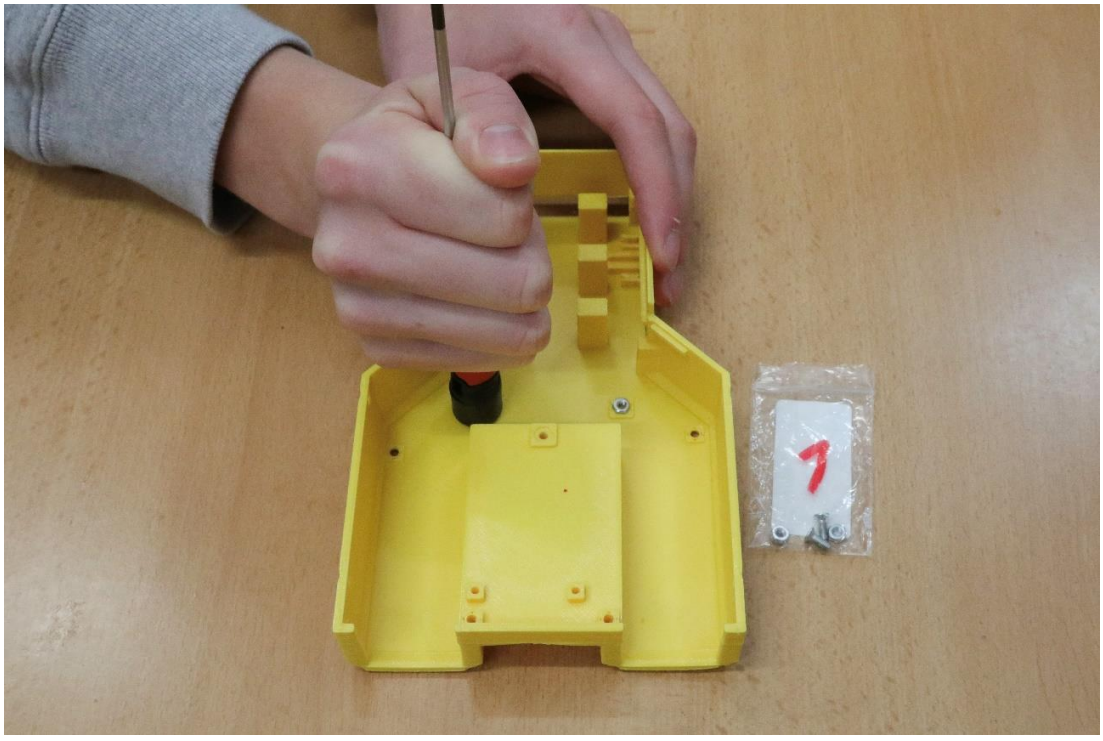
Složení

Potřebné věci: Plochý šroubovák, Křížový šroubovák

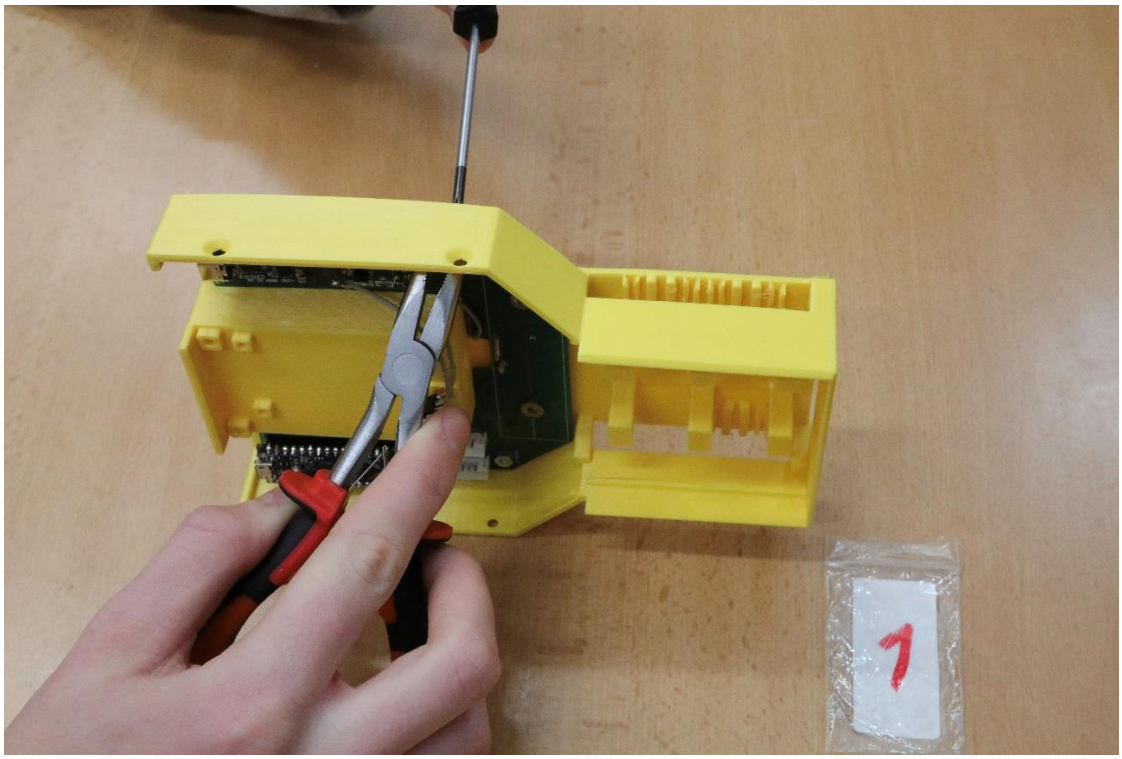
1. 1. Část – hlavní deska



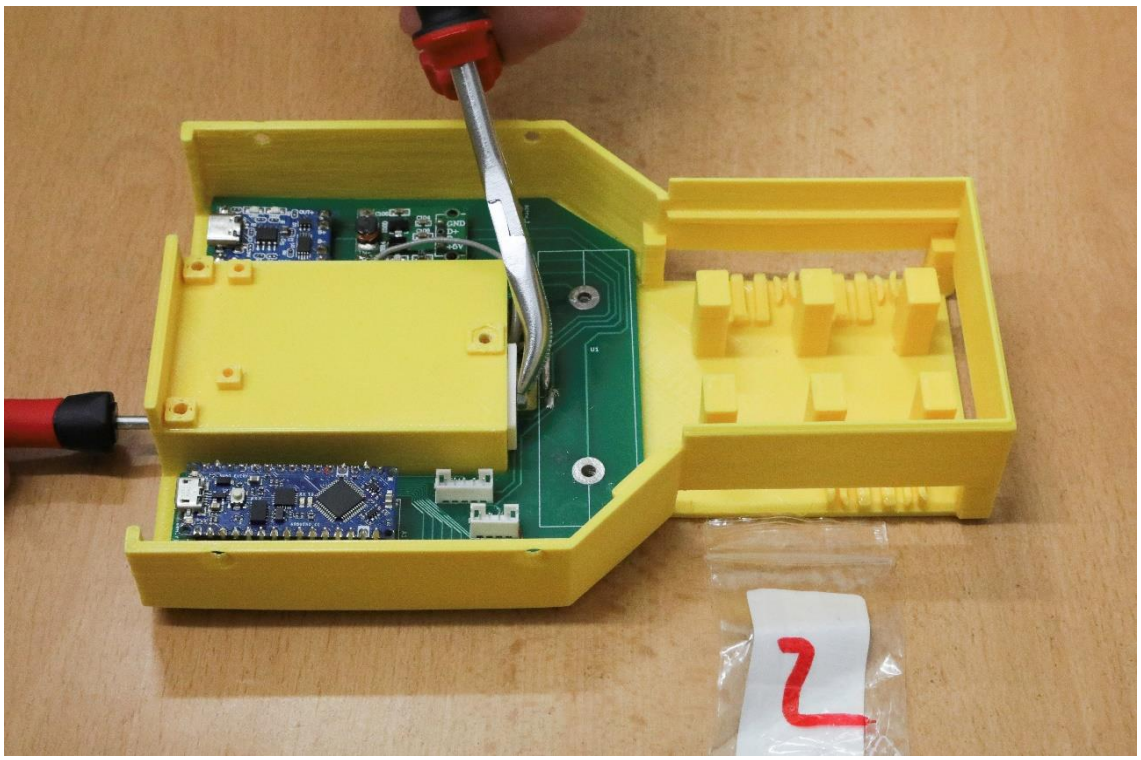
2. Dejte do spodních šestiúhelníků matičky, a to tak, že je nalisujte pomocí druhé strany šroubováku.



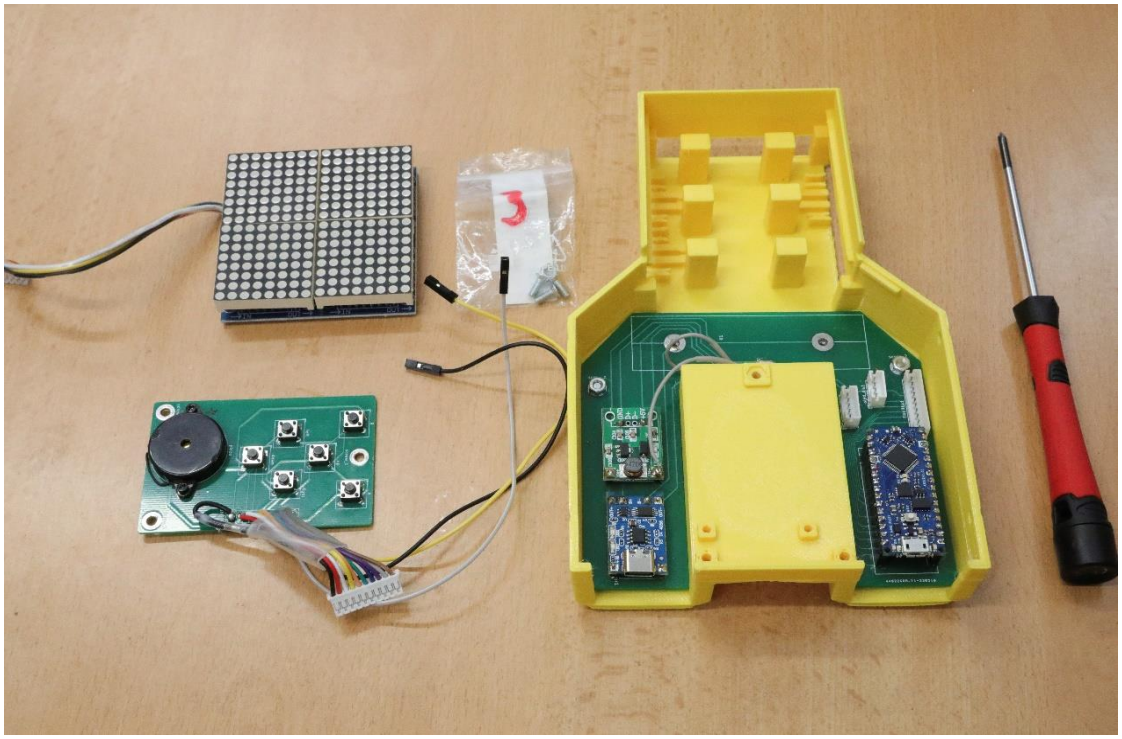
3. Našroubujte hlavní desku, budete muset přidržovat matičku na druhé straně, takže nejlepší je postavit si konzoli a jednou rukou šroubovat a druhou přidržovat matičku, aby se neprotáčela.



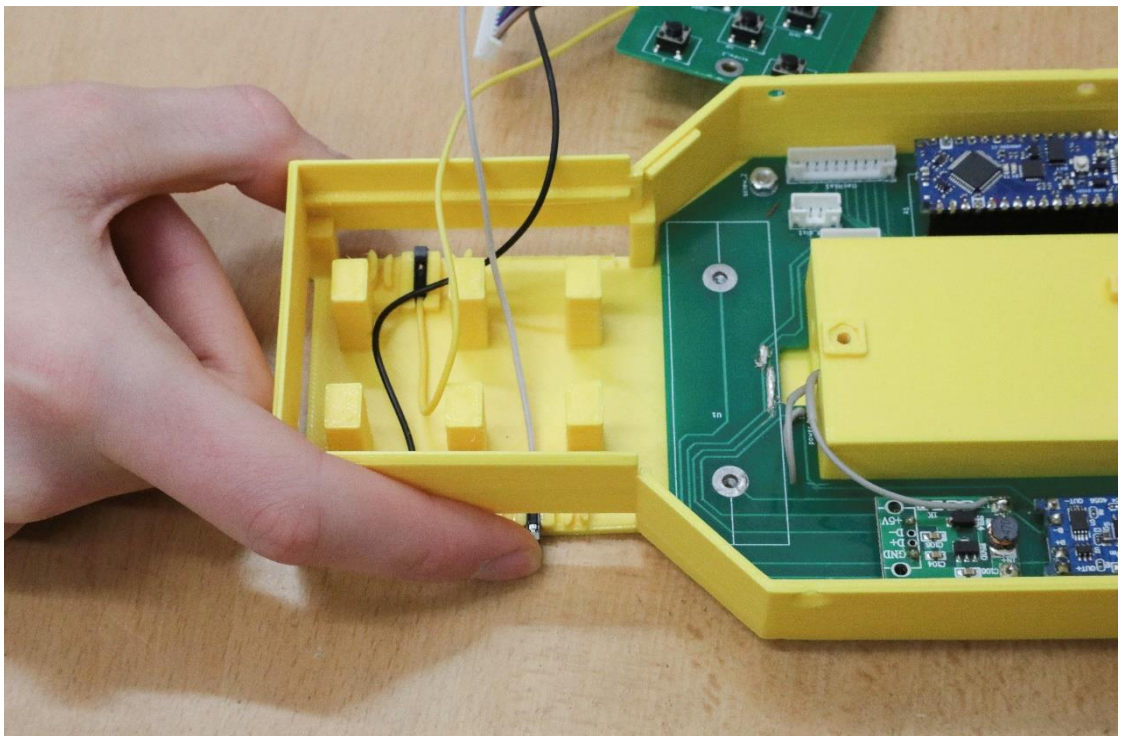
4. Dejte zapínací tlačítko do díry a pomocí tištěné destičky a matičky ho zajistěte.



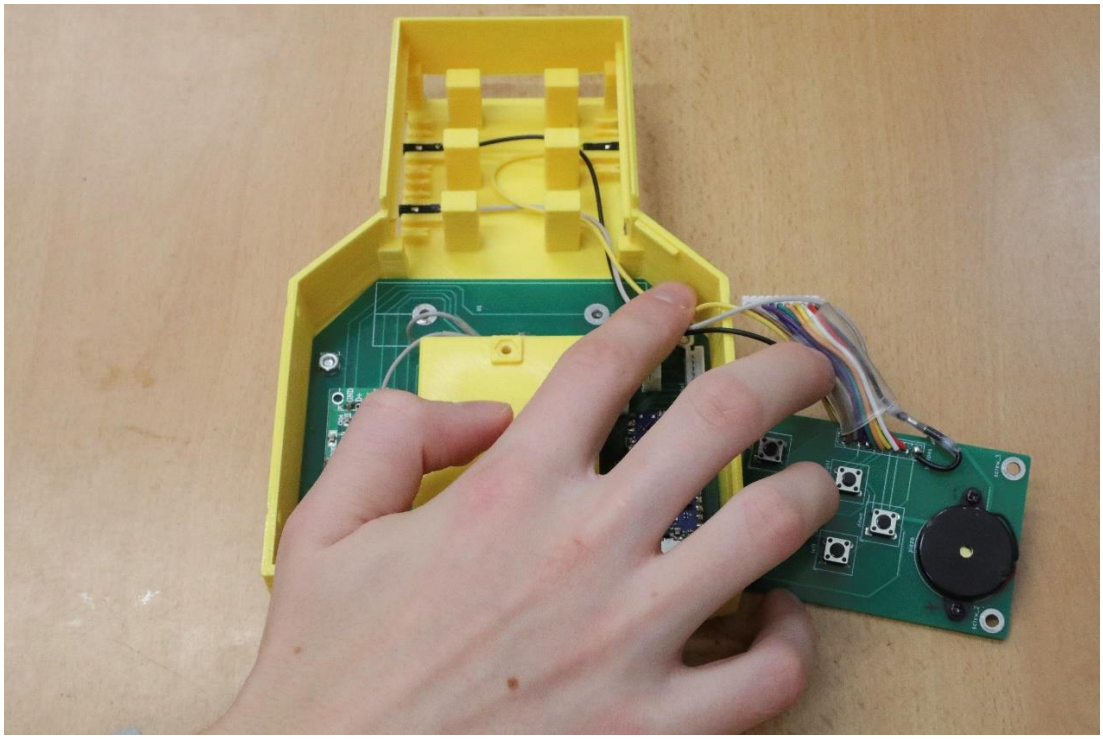
5. 2 Část – 16x16 displej a tlačítková deska



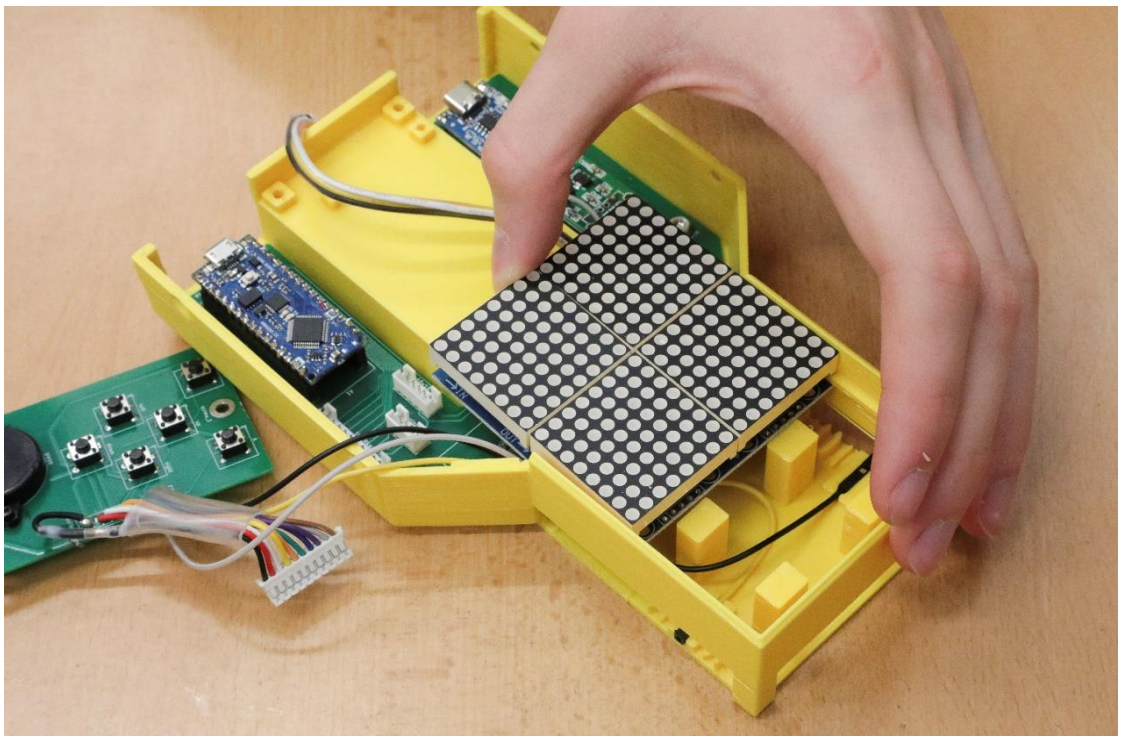
6. Zasuňte debug kabely do prostorů pro debug kabely.



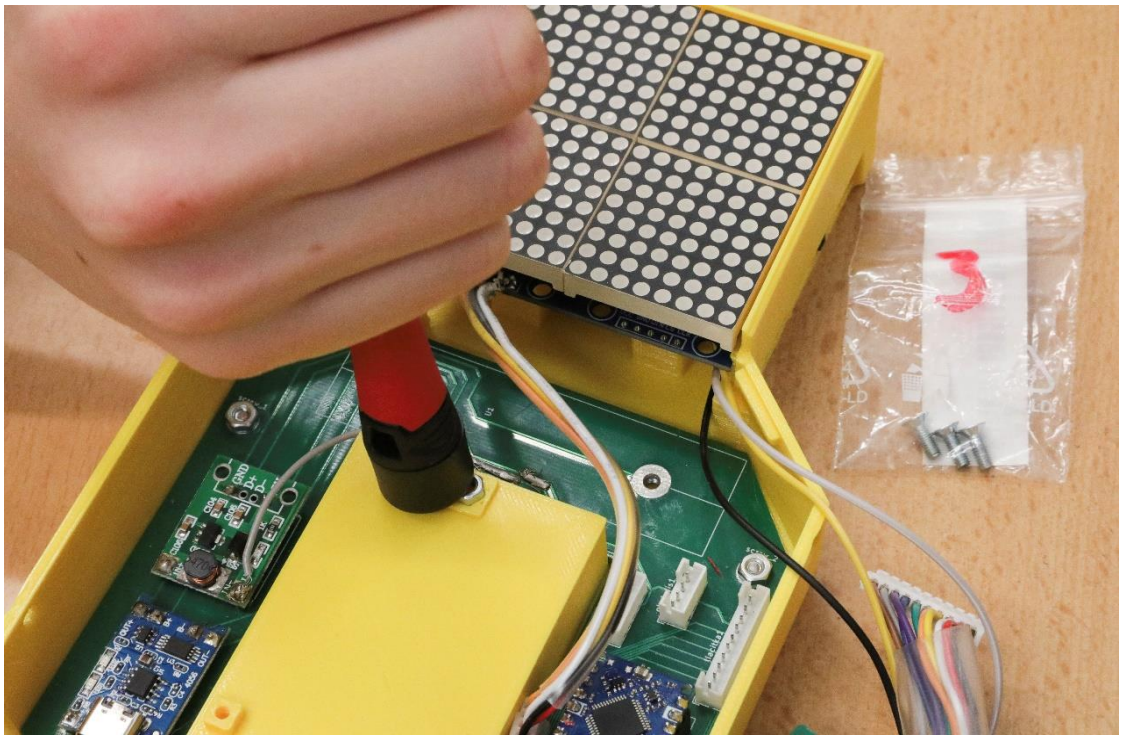
7. Přidržte si opatrně kabely, tak aby bylo možné zasunout displej pod drážky.



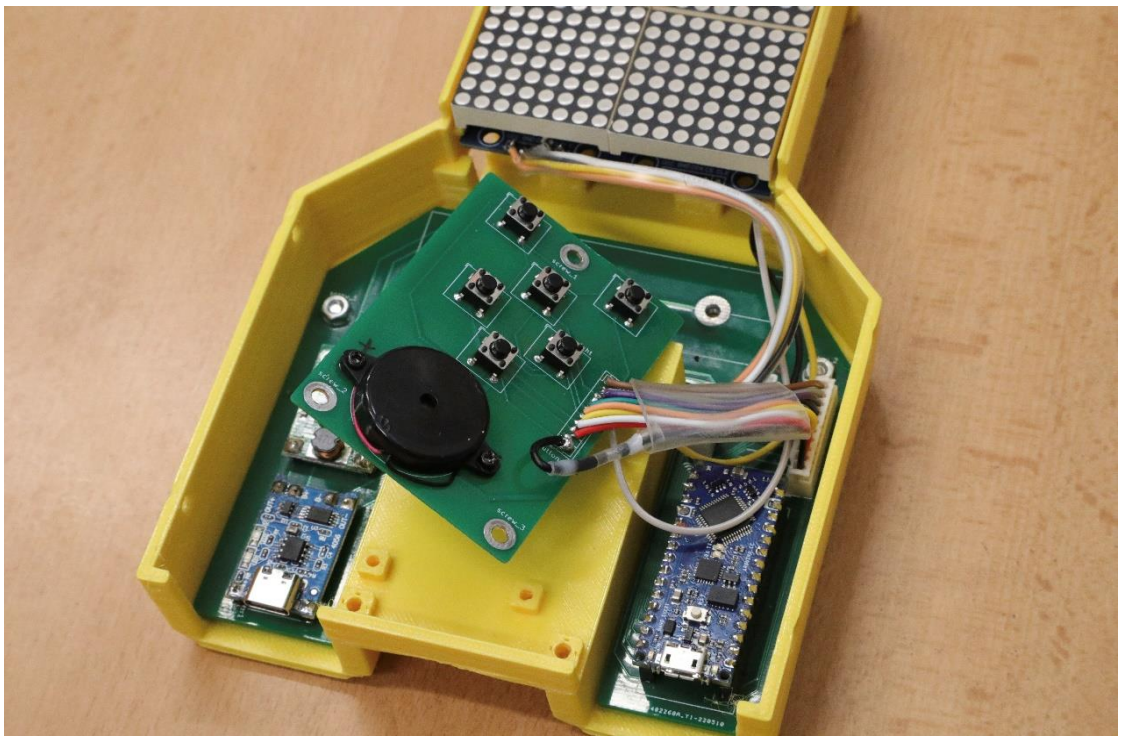
8. Zasuňte displej. Dejte si pozor ať je pod drážkami.



9. Dejte matičku do horního šestiúhelníku a pomocí druhé strany šroubováku jí tam nalisujte.

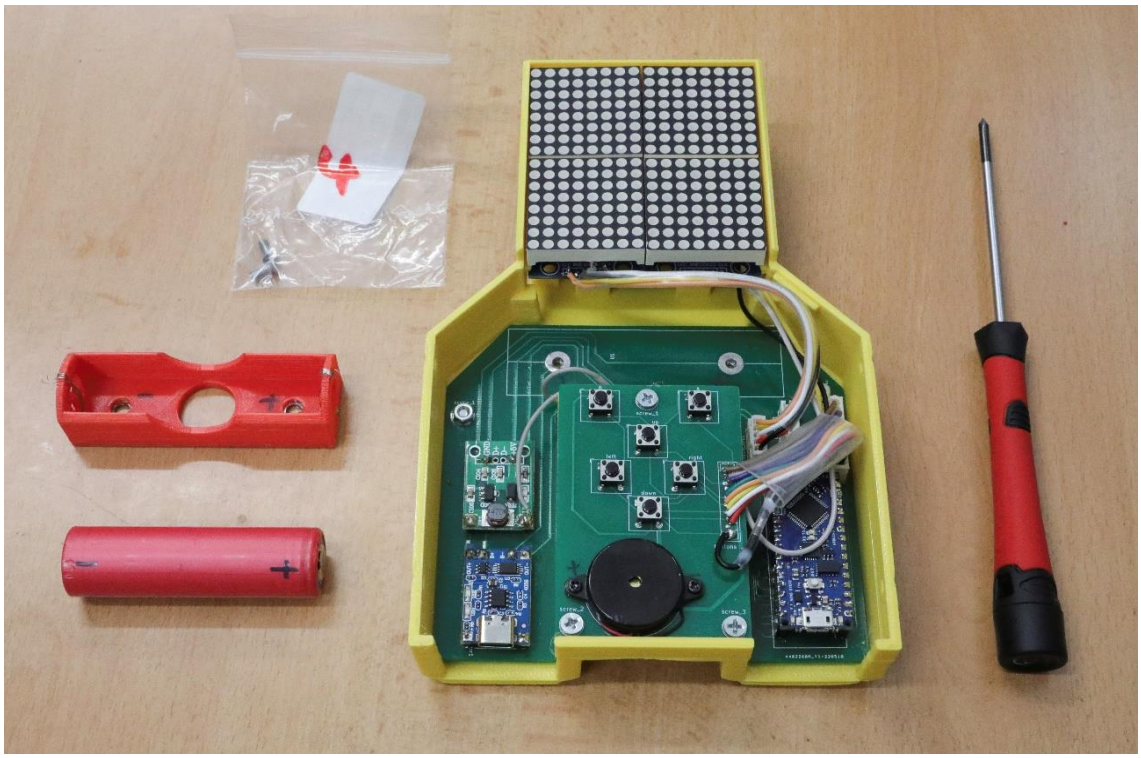


10. Zacvakejte displej a tlačítka do protikusů.

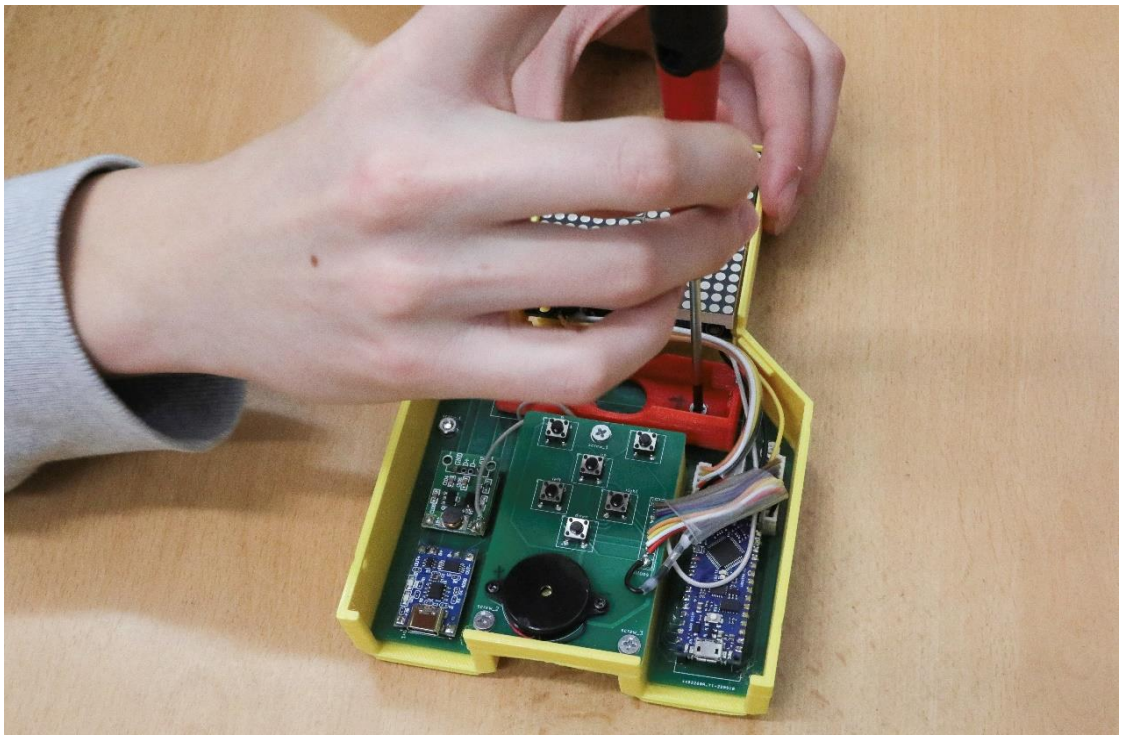


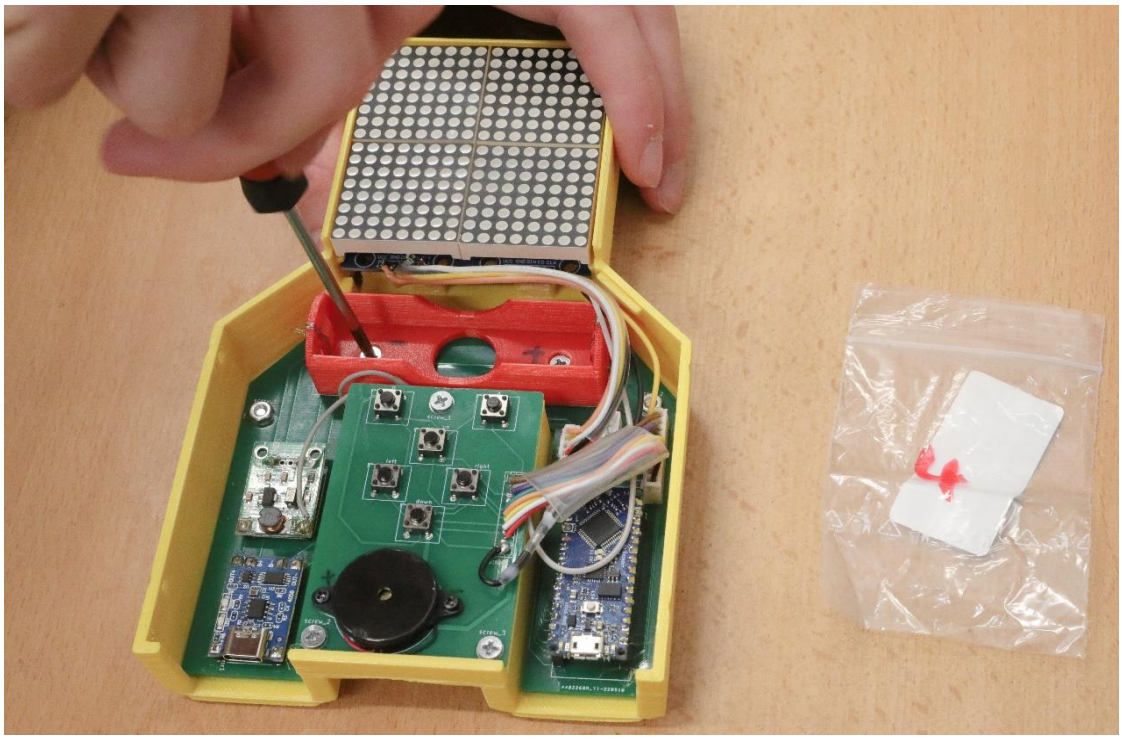
11. Našroubujte tlačítkovou desku.

12. 3 Část - baterie

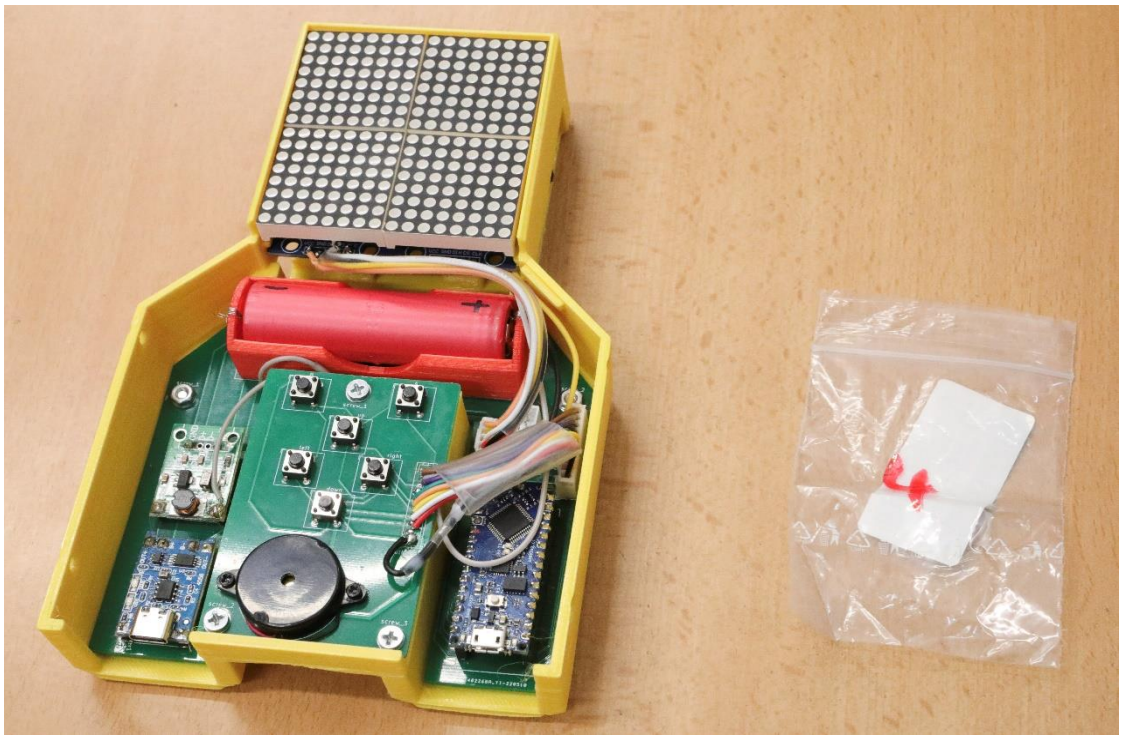


13. Našroubujte držák na baterku do hlavní desky.

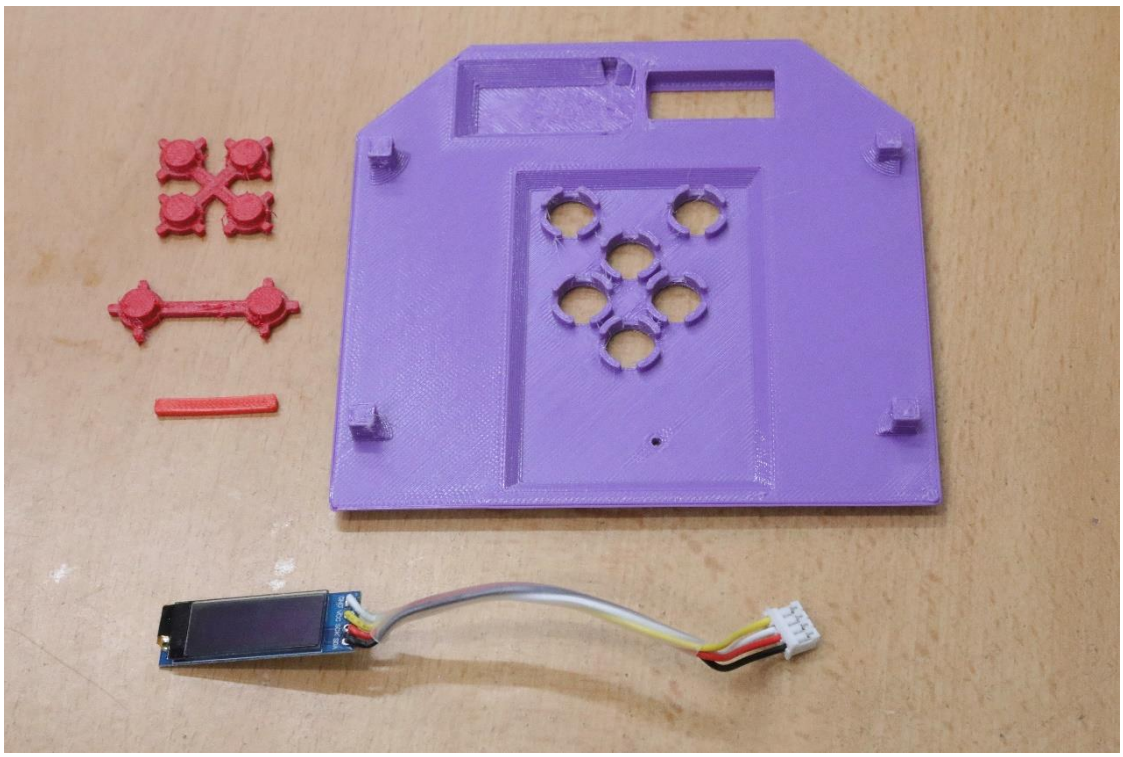




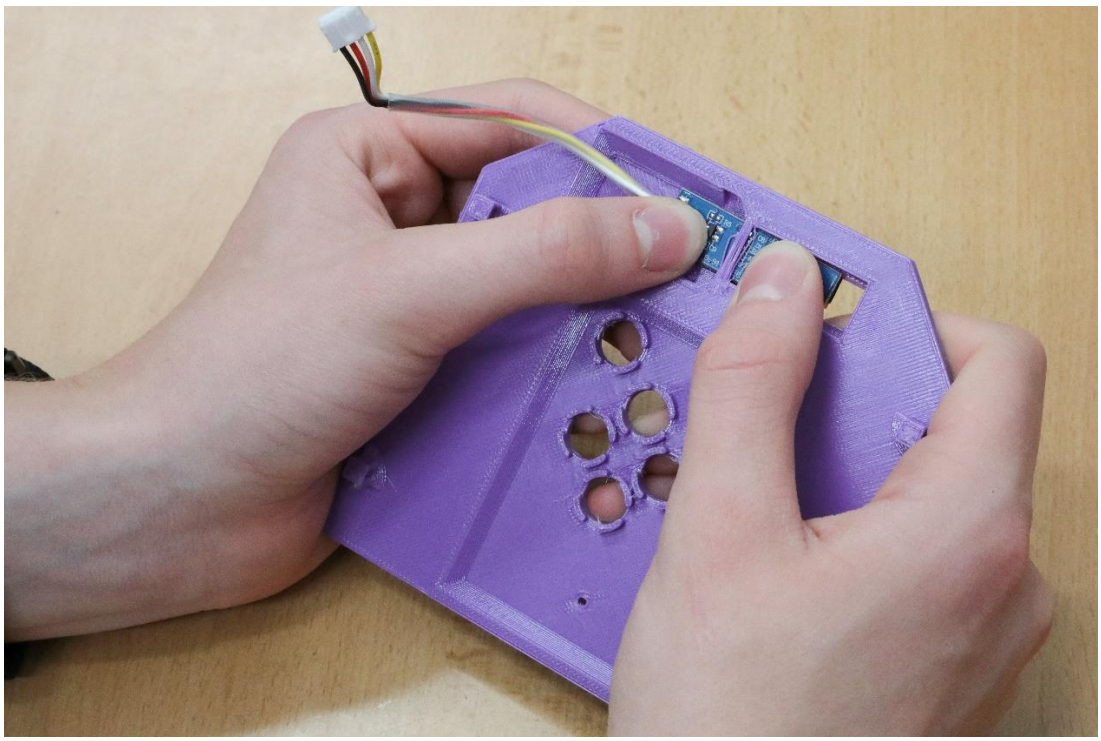
14. Dejte baterku do držáku na baterku



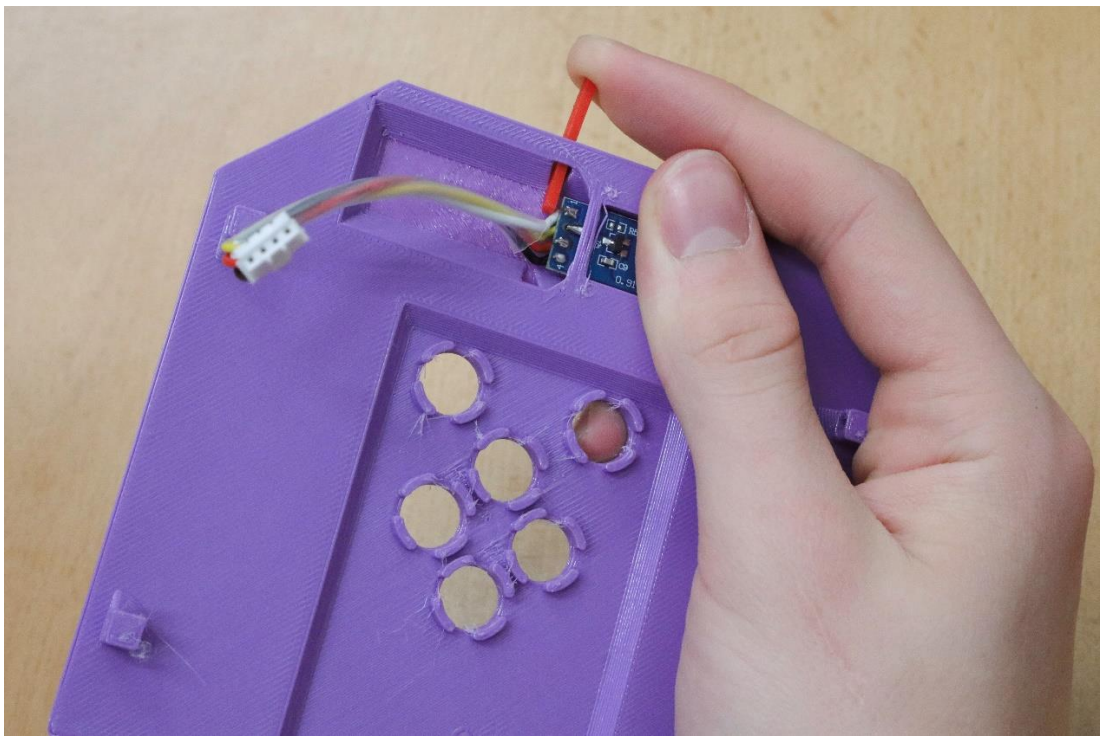
15. 4 Část – Oled displej



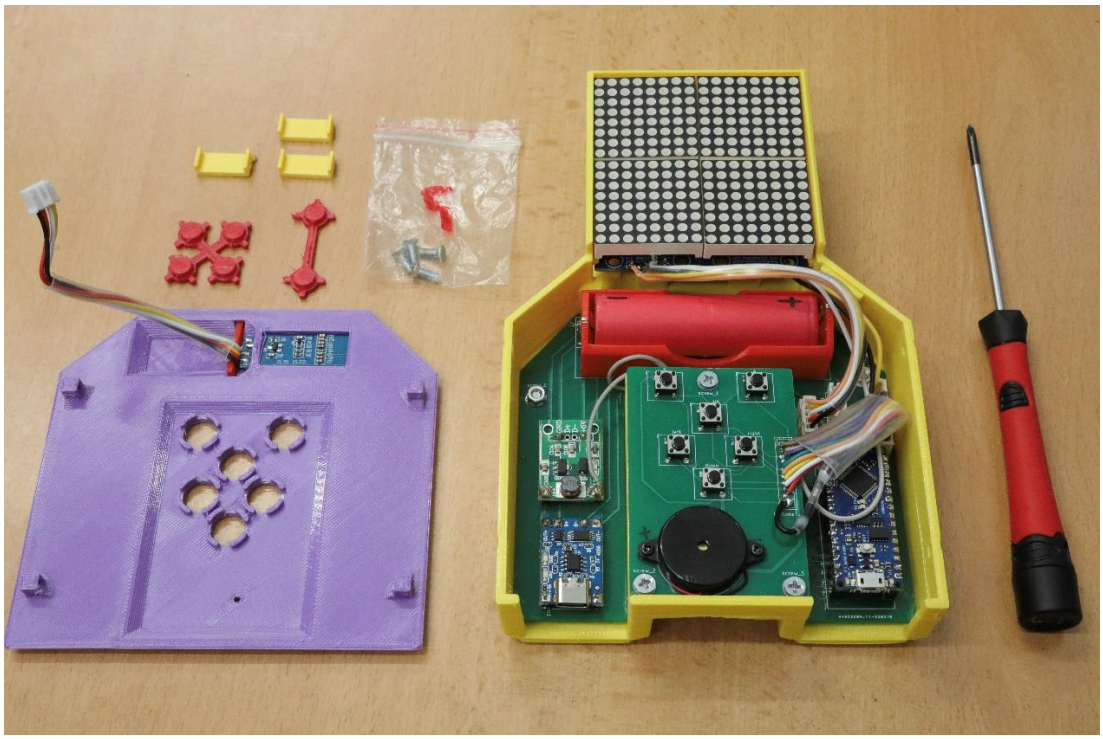
16. Pomocí malého tlaku a posunu displejem ho nasaďte do horní desky.



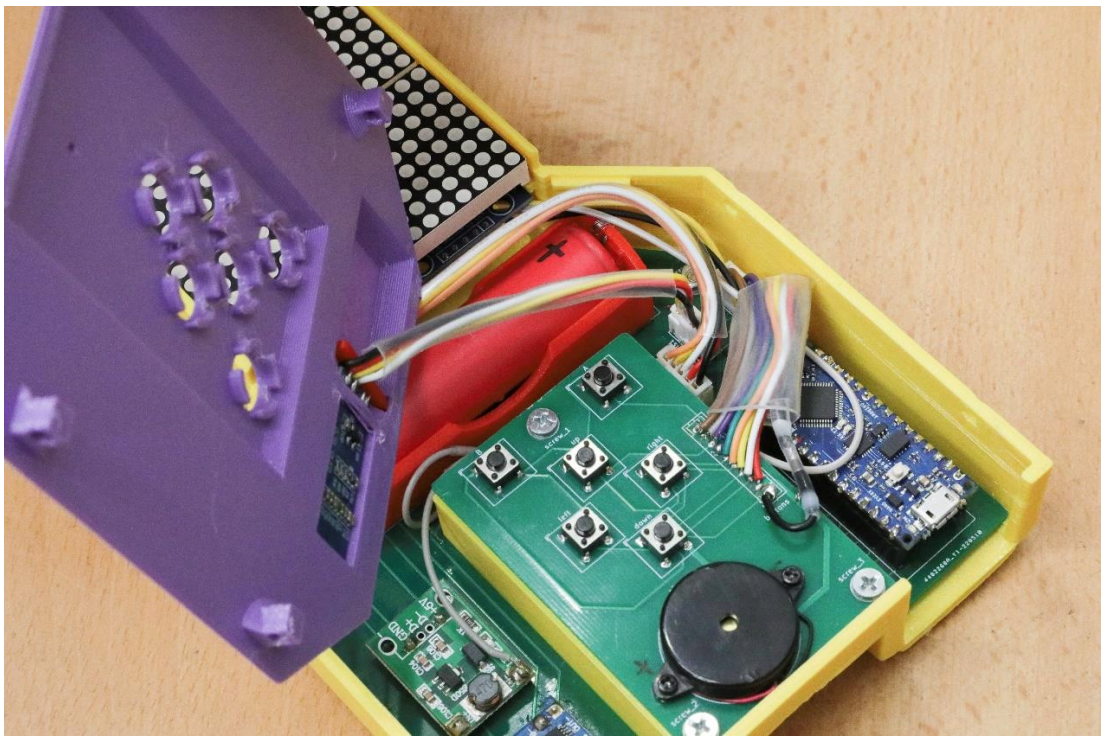
17. Dejte zácvaku do díry pro zácvaku, aby displej nevypad.



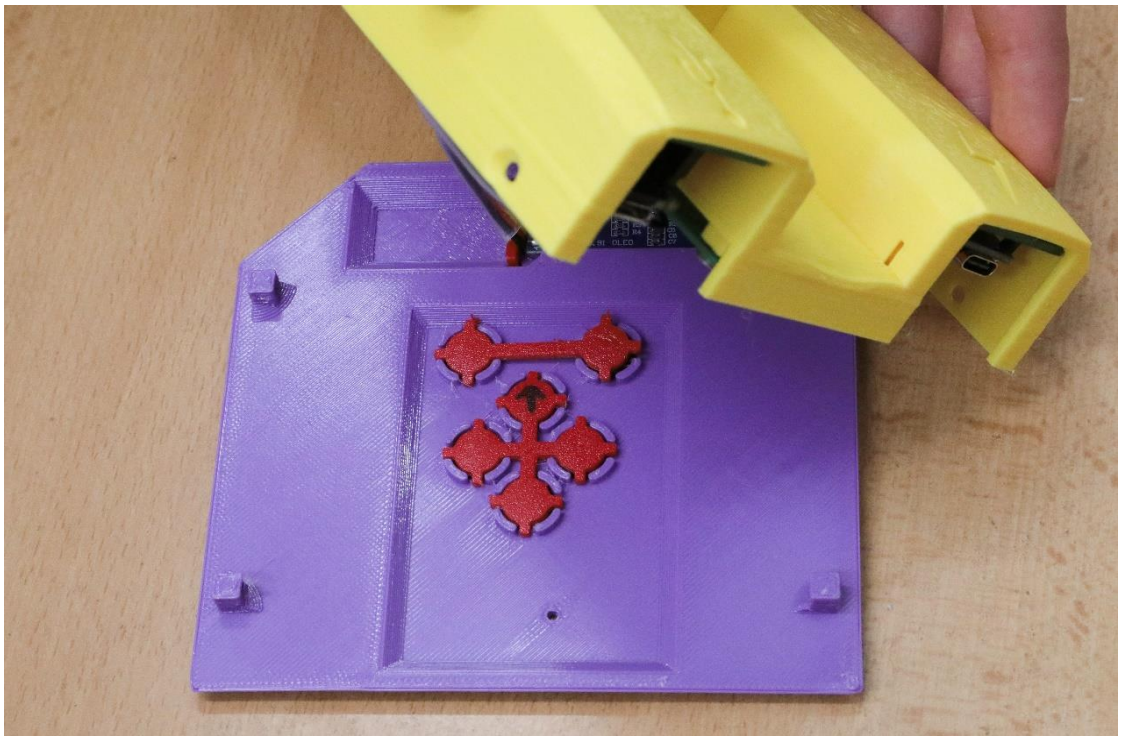
18. 5 Část – uzavření konzole



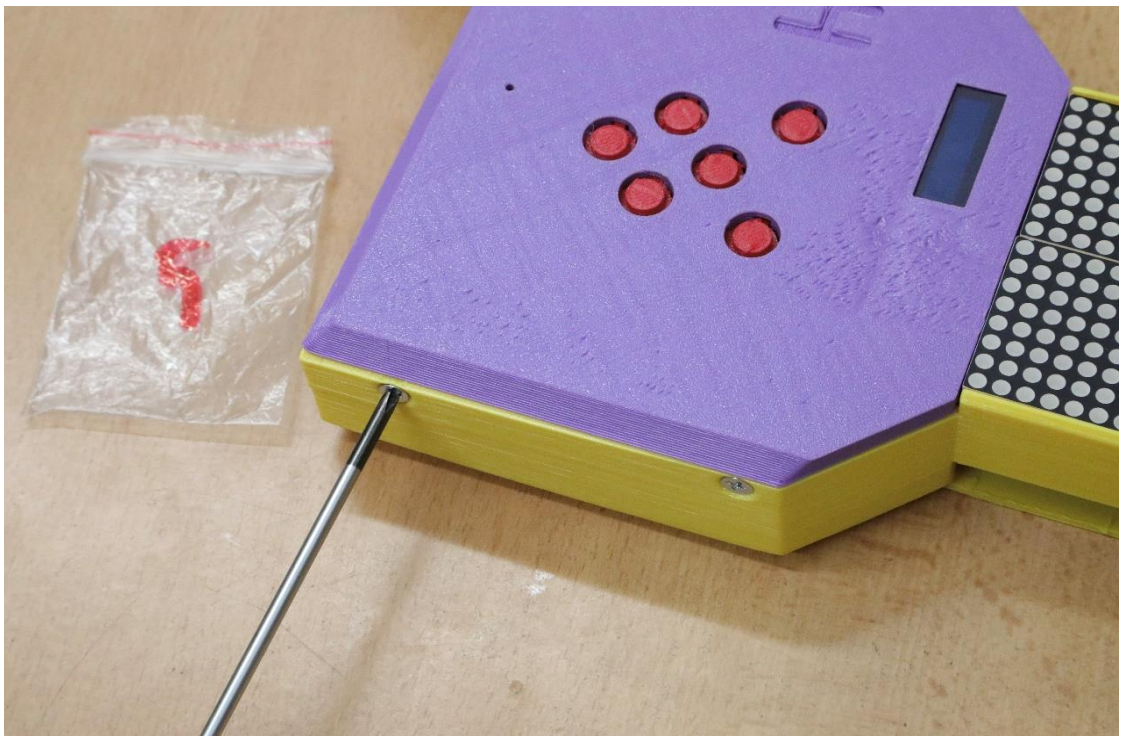
19. Zacvakejte kabel Oled displeje do protikusu



20. Dejte do míst pro tlačítka ohebnou část tlačítek. Šipkou nahoru stejně jako na obrázku.

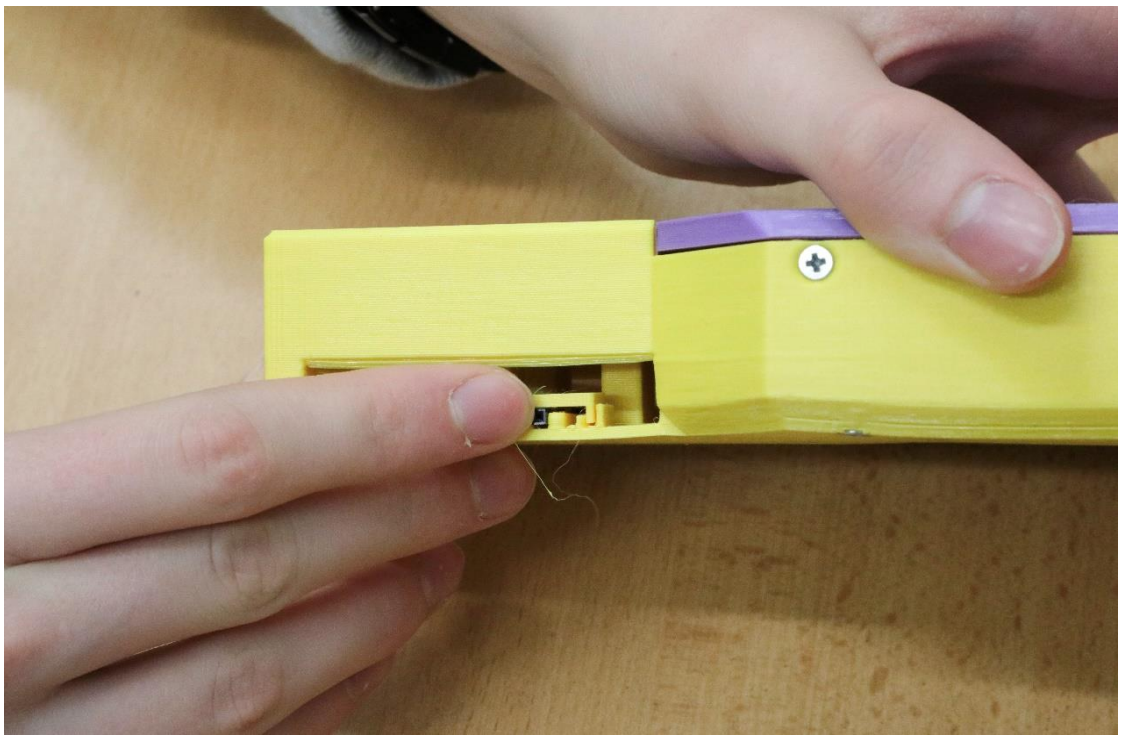


21. Přišroubujte vršek ke spodku.

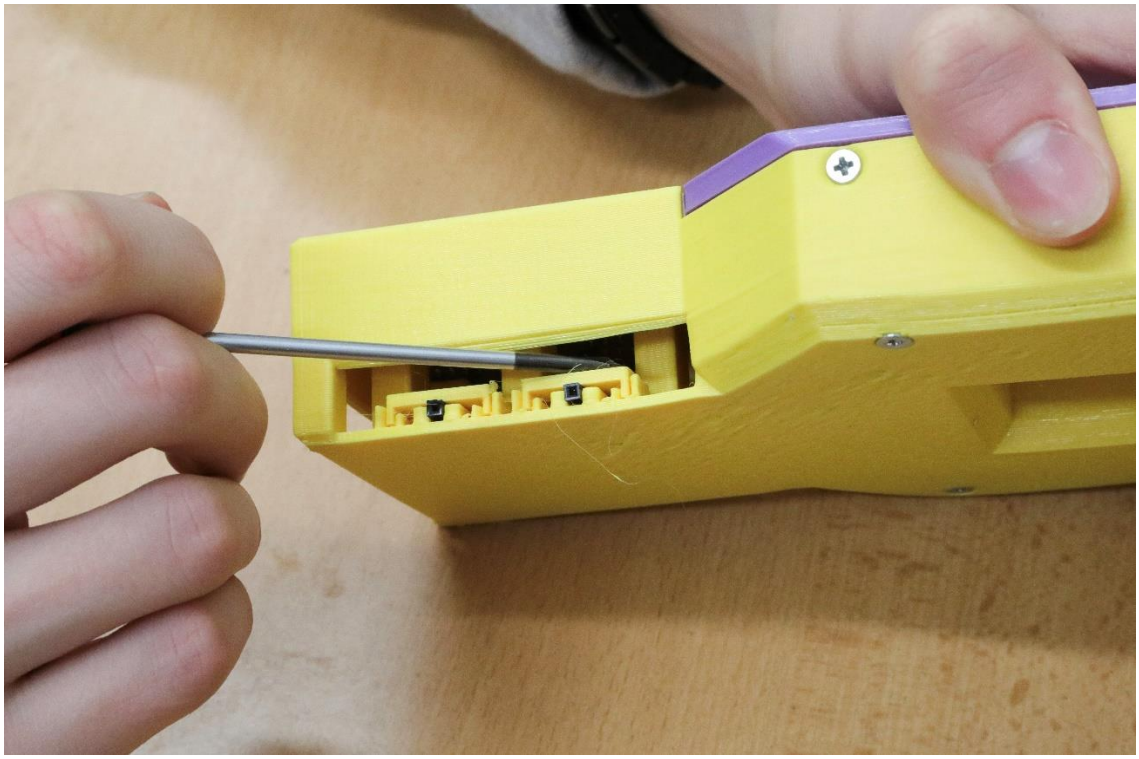


22. Podle toho jak moc to utáhnete, můžete ovlivnit, jak moc se nahne spodek k vrchní části. Mělo by to vypadat jako na obrázku.

23. Zasuňte krytku debug kabelů do drážky.



24. Můžete si pomoct šroubovákem, aby to nevypadlo.



PŘÍLOHA 2: KÓD – GALAXIAN

```
void GalaxianGame() {
  allTime = millis();
  if (active == -1) {
    y = 0;
    active = 0;
    noTone(soundPin);
    GameOver(points, 4);
  }
  if ((mode == 1 or mode == 2) and active == 1) {
    noTone(soundPin);
    int mod = mode;
    SettingGameGalaxian();
    mode = -mod;
    displayOled("Your Score: " + String(points) + "\nHigh Score: " + String(read2EEPROM(addressGalaxian1)));
  } else if (mode == 1 and y != 0) {
    displayOled("Galaxian Classic\nFor start press A");
    noteCounter=0;
    soundDelay1=0;
    noTone(soundPin);
    mode = 2;
    y = 0;
    delay(100);
  } else if (mode == 2 and y != 0) {
    displayOled("Galaxian song\nFor start press A");
    mode = 1;
    y = 0;
    delay(100);
  }
  if (mode==1) {
    playSong("BloodyTears");
  }
  if (y != 0 and abs(mode)!=mode) {
    displayLed(15, 0, 0);
  } else if (abs(mode)==mode) {
    for (int row = 0; row < 3; row++) {
      for (int col = 0; col < 2; col++) {
        displayLed(row + shiftedY, col, pgm_read_byte(&(ship[row][col])));
      }
    }
  }
  if (abs(mode)!=mode) {
    if (mode==-1) {
      playSong("BloodyTears");
    }
    if (allTime - lastTime >= 200) {
      shiftedY -= y;
      y = 0;
      Warp();
      for (int row = 0; row < 3; row++) {
        for (int col = 0; col < 2; col++) {
          Array[row + shiftedY][col] = pgm_read_byte(&(ship[row][col])) + 1;
        }
      }
      if (active == 1) {
        active = 0;
        if (mode==-2) {
          PlaySound(4);
        }
        ShootG(shiftedY + 1);
      }
      // Cele vykreslovani hry
      for (int row = 0; row < 16; row++) {
```

```
      for (int col = 0; col < 16; col++) {
        if (Array[row][col] > 0) {
          Array[row][col]--;
          if (Array[row][col] == 0) {
            displayLed(row, col, 0);
          } else if (Array[row][col] == 1) {
            int displayShip = 0;
            for (int i = 0; i < helpingVariable * 2; i++) {
              if (i % 2 == 0) {
                if (row == EnemysG[i] and col == EnemysG[i + 1]) {
                  displayShip = 1;
                  if (EnemyDouble[i] == 1 and abs(EnemysG[i]) == EnemysG[i]) {
                    displayLed(EnemysG[i], EnemysG[i + 1] + 1, 0);
                    if (mode==-2) {
                      PlaySound(5);
                    }
                    EnemysG[i] = -EnemysG[i];
                    EnemysG[i + 1] = -EnemysG[i + 1];
                    points += 1;
                    displayOled("Your Score: " + String(points) + "\nHigh Score: " + String(read2EEPROM(addressGalaxian1)));
                  } else if (EnemyDouble[i] > 1 and abs(EnemysG[i]) == EnemysG[i]) {
                    if (mode==-2) {
                      PlaySound(5);
                    }
                    EnemyDouble[i] -= 1;
                    displayLed(EnemysG[i], EnemysG[i + 1], 0);
                    points += 1;
                    displayOled("Your Score: " + String(points) + "\nHigh Score: " + String(read2EEPROM(addressGalaxian1)));
                  }
                  if ((EnemysG[i] != 17 and EnemysG[i + 1] != 17) and EnemysG[i] != abs(EnemysG[i])) {
                    wave += 1;
                    EnemysG[i + 1] = 17;
                    EnemysG[i] = 17;
                  }
                  for (int j = col; j < 16; j++) {
                    Array[row][j] = 0;
                  }
                }
              }
            }
            if (displayShip != 1) {
              displayLed(row, col, 1);
            }
          }
        }
      }
    }
  }
  if (allTime - lastTime2 >= delayTime) {
    if (wave >= helpingVariable) {
      wave = 0;
      moveEnemy1 += 1;
      helpingVariable = EnemyG(moveEnemy1);
    }
    for (int i = 0; i < helpingVariable * 2; i++) {
      if (i % 2 == 0) {
        if (EnemysG[i] != 17 and EnemysG[i + 1] != 17) {
          displayLed(EnemysG[i], EnemysG[i + 1] + 1, 0);
          if (EnemyDouble[i] > 1) {
            for (int j = 1; j < EnemyDouble[i]; j++) {
              displayLed(EnemysG[i], EnemysG[i + 1] - j, 1);
```

```
                }
              }
              displayLed(EnemysG[i], EnemysG[i + 1], 1);
              EnemysG[i + 1]--;
            }
            if (EnemysG[i + 1] <= 0 + EnemyDouble[i] - 1) {
              GameOver(points, 4);
              break;
            }
          }
        }
        lastTime2 = allTime;
      }
    }
}
```

PŘÍLOHA 3: KÓD – DANCE MAN

```
void DanceMan() {
  allTime = millis();
  if (active == -1) {
    GameOver();
  }
  if (abs(mode) == mode and active == 1) {
    noTone(soundPin);
    int mod = mode;
    helpingVariable = -mode;
    SettingGameDanceMan();
    helpingVariable = 1;
    mode = -mod;
    if (mode == -2 ) {
      displayOled("Song: Uno");
    } else if (mode == -1 ) {
      displayOled("Song: Among us");
    } else if (mode == -3) {
      displayOled("Song: DOOM");
    } else if (mode == -4) {
      displayOled("Song: Key. cat");
    }
  } else if (mode == 1 and y != 0) {
    mode = 3 + y;
    y = 0;
    delay(100);
  } else if (mode == 2 and y != 0) {
    mode -= y;
    y = 0;
    delay(100);
  } else if (mode == 3 and y != 0) {
    mode -= y;
    y = 0;
    delay(100);
  } else if (mode == 4 and y != 0) {
    mode = 2 + y;
    y = 0;
    delay(100);
  }
  if (mode == 1) {
    displayOled("Song: Among us\nFor start press A");
    displayImage(ArrowUp);
  } else if (mode == 2) {
    displayOled("Song: Uno\nFor start press A");
    displayImage(ArrowDown);
  } else if (mode == 3) {
    displayOled("Song: DOOM\nFor start press A");
    displayImage(ArrowLeft);
  } else if (mode == 4) {
    displayOled("Song: Key cat\nFor start press A");
    displayImage(ArrowRight);
  }
  //Serial.println(mode);
  if (abs(mode) != mode) {
    if (helpingVariable == 1) {
      allwaysY = random(0, 4);
      const word* nazev[] = {ArrowUp, ArrowRight, ArrowDown, ArrowLeft};
      displayImage(nazev[allwaysY]);
      helpingVariable = 0;
      allwaysX = 170;
      x = 0;
      y = 0;
    } else if (allwaysX <= 180) {
```

```
      bool hit = false;
      if (x == 1 or x == -1 or y == 1 or y == -1) {
        if (x == 1 and allwaysY == 3) {
          hit = true;
        } else if (x == -1 and allwaysY == 1) {
          hit = true;
        } else if (y == 1 and allwaysY == 0) {
          hit = true;
        } else if (y == -1 and allwaysY == 2) {
          hit = true;
        } else if (allwaysY == -2) {
          hit = true;
        } else {
          noTone(soundPin);
          GameOver();
        }
      }
      if (hit == true) {
        displayClear();
        if (allwaysX <= 170 and allwaysX > 110) {
          points += allwaysX / 10;
          displayOled("Perfect: " + String(allwaysX / 10) + "\nAll Points: " + String(points));
        } else if (allwaysX <= 110 and allwaysX > 50) {
          points += allwaysX / 10;
          displayOled("Good: " + String(allwaysX / 10) + "\nAll Points: " + String(points));
        } else if (allwaysX <= 50) {
          points += allwaysX / 10;
          displayOled("Ok: " + String(allwaysX / 10) + "\nAll Points: " + String(points));
        }
        x = 0;
        y = 0;
        allwaysX = 0;
        helpingVariable = 1;
        //delay(100);
      }
    }
    if (allTime - lastTime >= 10 and allwaysX != 1 and allwaysX != 0) {
      lastTime = millis();
      allwaysX--;
    }
    if (mode == -1 and gameState == 3) {
      playSong("AmongUs");
    } else if (mode == -2 and gameState == 3) {
      playSong("Uno");
    } else if (mode == -3 and gameState == 3) {
      playSong("Doom");
    } else if (mode == -4 and gameState == 3) {
      playSong("KeyboardCat");
    }
  }
}
```

PŘÍLOHA 4: (ONLY IN ENGLISH) HOW TO UPDATE URX OS

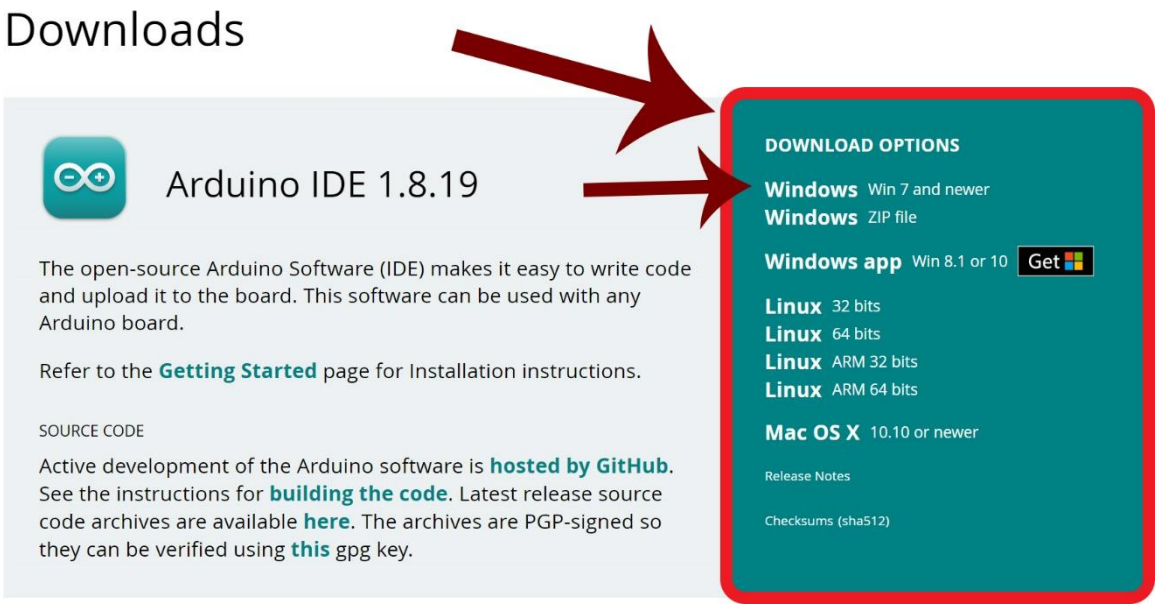
Tutorial in points

1. Download Arduino software
2. Download URXos and URXos library
3. Import URXos library
4. Download Arduino nano every board config
5. Plug in URX to your computer
6. Upload code

Tutorial in pictures

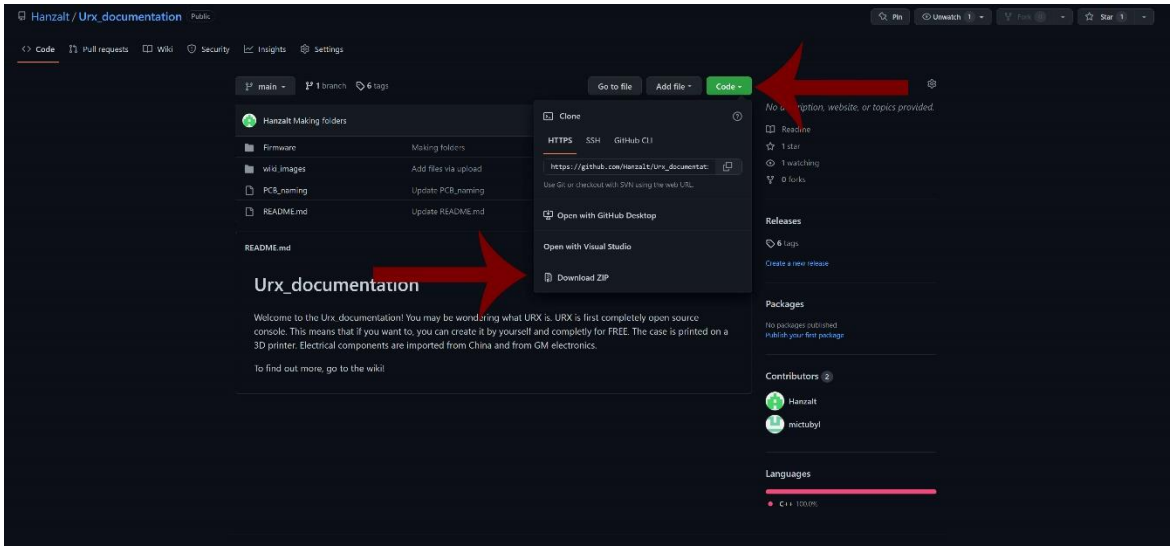
1. Download Arduino

1. Click on this link: https://www.arduino.cc/en/software and download Arduino software for your os (I prefer Arduino 1.8.9 over 2.0)
2. If you have windows download: **Windows Win 7 and newer** not **Windows app**
3. Download and install arduino



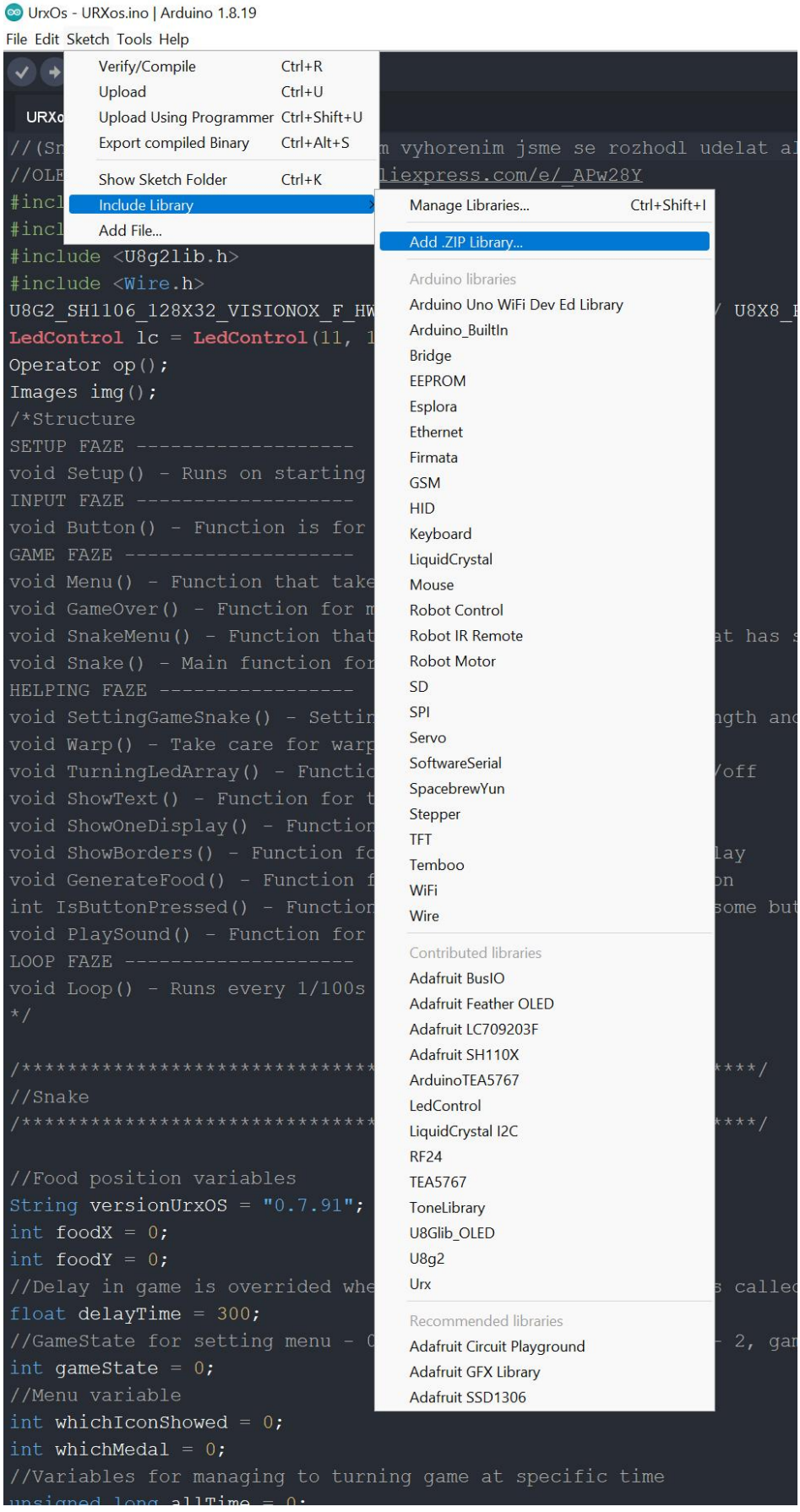
2. Download URXos and URXos library

1. Click on this link: https://github.com/Hanzalt/Urx_documentation
2. Click **Code** in green box
3. Download **zip**
4. After downloading zip file, extract all the files



3. Import URXos library

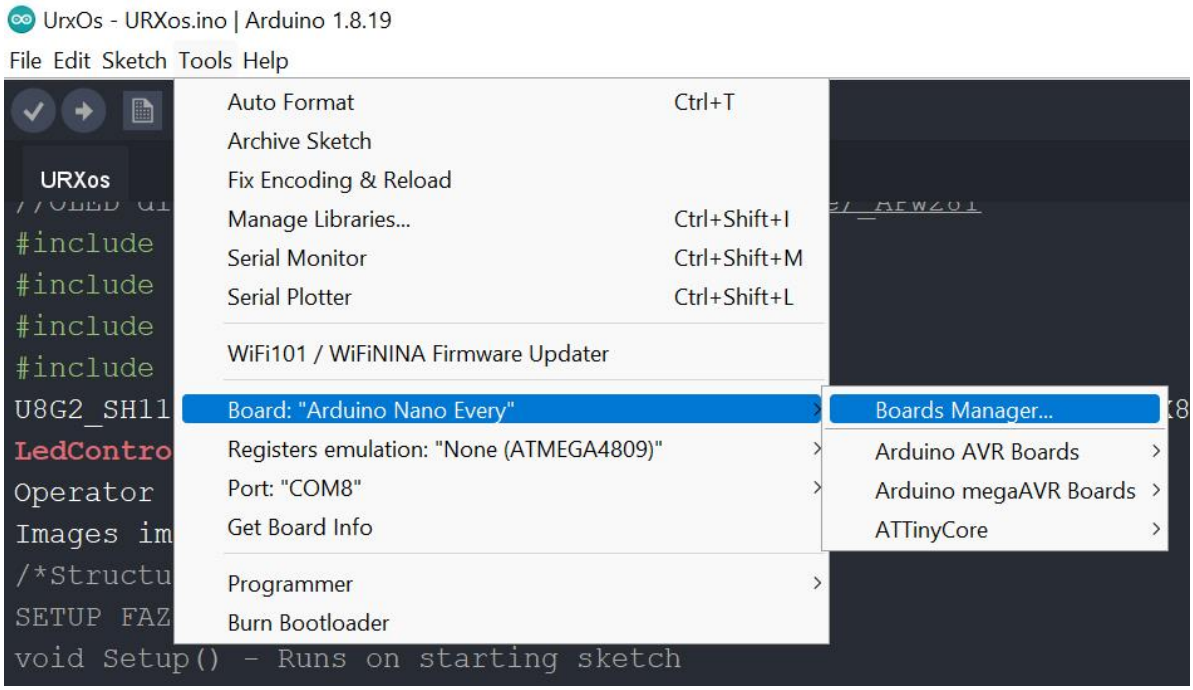
1. URXos and URXos library is in folder **Firmware**
2. Open URXos in Arduino IDE (just double click file URXos)
3. Add library: Click Sketch -> Include Library -> Add .ZIP Library...
4. Then find URXos library: Firmware -> Urxos_library -> URXos_library_X-Y-Z.zip

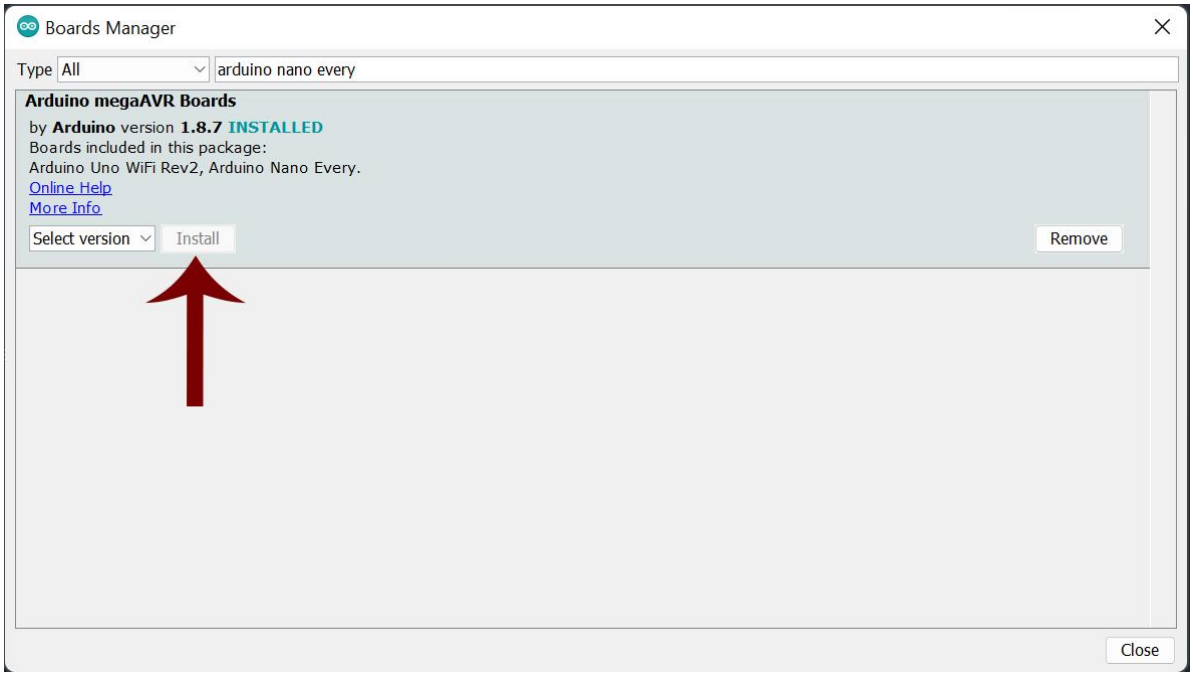

UrxOs - URXos.ino | Arduino 1.8.19
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Arduino Uno WiFi Dev Ed Library
Arduino_BuiltIn
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
HID
Keyboard
LiquidCrystal
Mouse
Robot Control
Robot IR Remote
Robot Motor
SD
SPI
Servo
SoftwareSerial
SpacebrewYun
Stepper
TFT
Temboo
WiFi
Wire
Contributed libraries
Adafruit BusIO
Adafruit Feather OLED
Adafruit LC709203F
Adafruit SH110X
ArduinoTEA5767
LedControl
LiquidCrystal I2C
RF24
TEA5767
ToneLibrary
U8Glib_OLED
U8g2
Urx
Recommended libraries
Adafruit Circuit Playground
Adafruit GFX Library
Adafruit SSD1306
#include <U8g2lib.h>
#include <Wire.h>
LedControl lc = LedControl(11, 1
Operator op();
Images img();
/*Structure
SETUP FAZE ------------------
void Setup() - Runs on starting
INPUT FAZE ------------------
void Button() - Function is for
GAME FAZE ------------------
void Menu() - Function that take
void GameOver() - Function for m
void SnakeMenu() - Function that
void Snake() - Main function for
HELPING FAZE ----------------
void SettingGameSnake() - Settin
void Warp() - Take care for warp
void TurningLedArray() - Functio
void ShowText() - Function for t
void ShowOneDisplay() - Function
void ShowBorders() - Function fo
void GenerateFood() - Function f
int IsButtonPressed() - Function
void PlaySound() - Function for
LOOP FAZE ------------------
void Loop() - Runs every 1/100s
*/
//Snake
//Food position variables
String versionUrxOS = "0.7.91";
int foodX = 0;
int foodY = 0;
float delayTime = 300;
int gameState = 0;
//Menu variable
int whichIconShowed = 0;
int whichMedal = 0;
//Variables for managing to turning game at specific time

Your Arduino IDE probably looks different, because I have downloaded One Dark theme from this link: https://github.com/konrad91/OneDarkArduino

4. Download Arduino nano every board config

1. Open board manager: Click Tools -> Board: "Something" -> Boards Manager...
2. Then type: arduino nano every and install Arduino megaAVR Boards

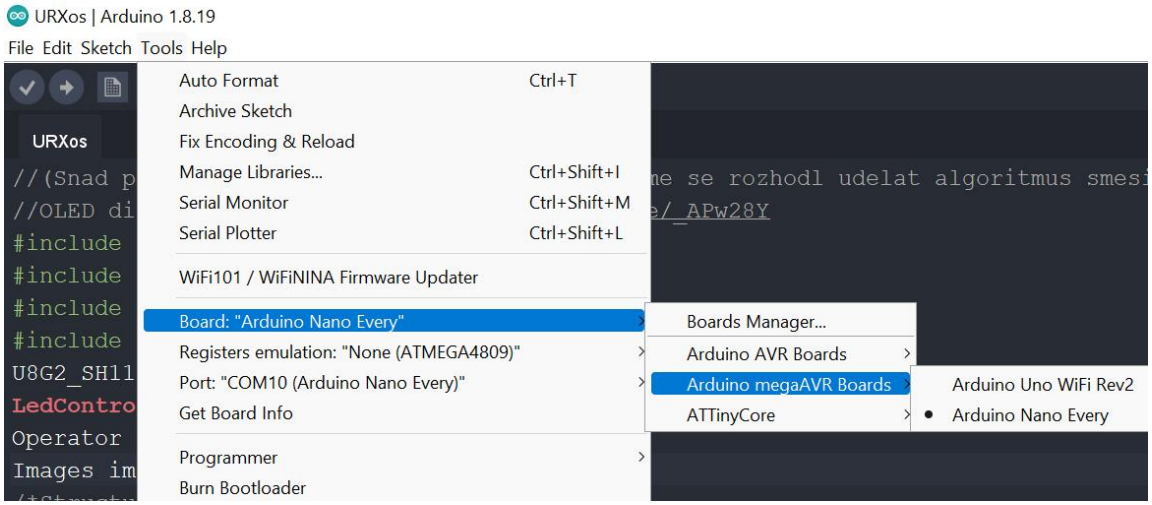


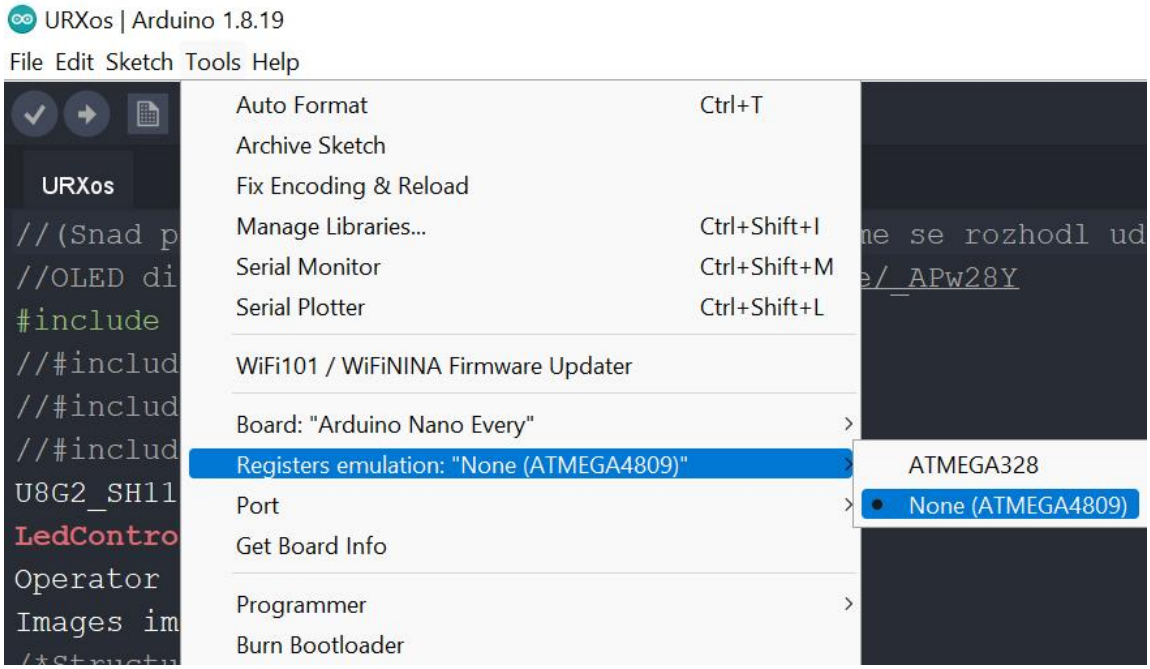


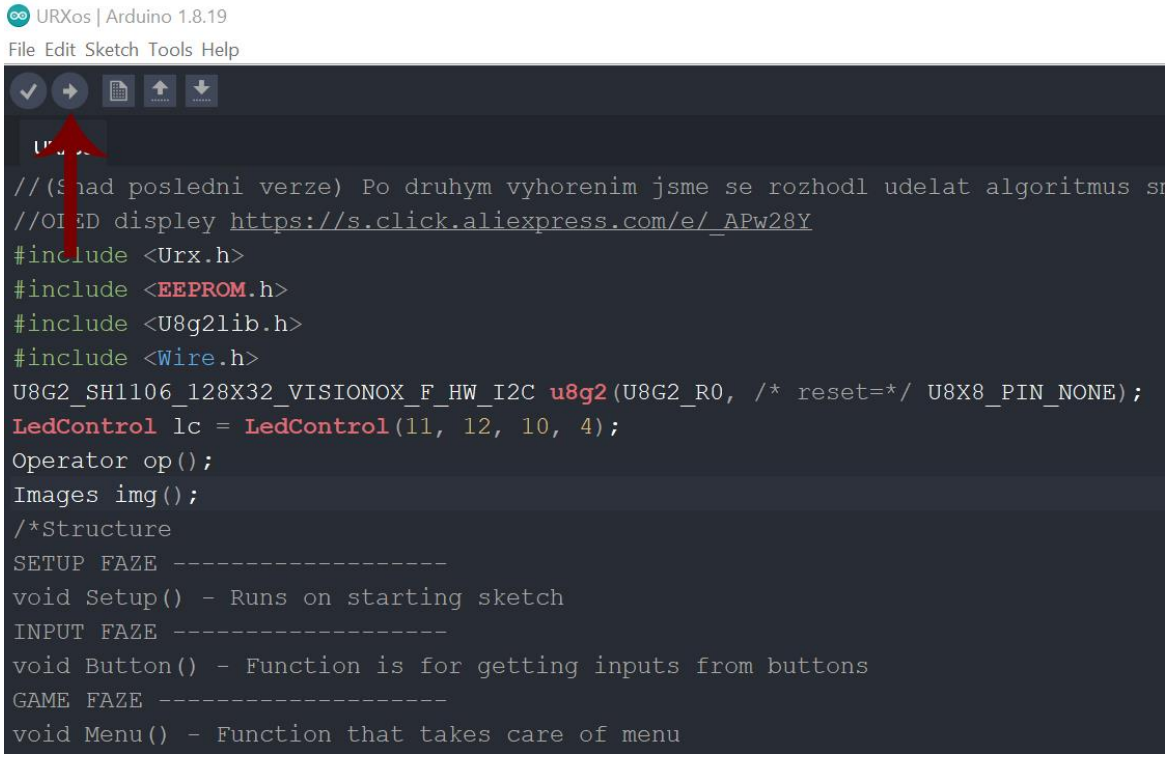
5. Plug in URX to your computer

1. First connect cable to URX (micro usb on the right)
2. Set board information: In Tools bar set

- Board -> Arduino megaAVR Boards -> Arduino Nano every
- Registers emulation -> None (ATMEGA4809)
- Port -> Port where you have arduino connected

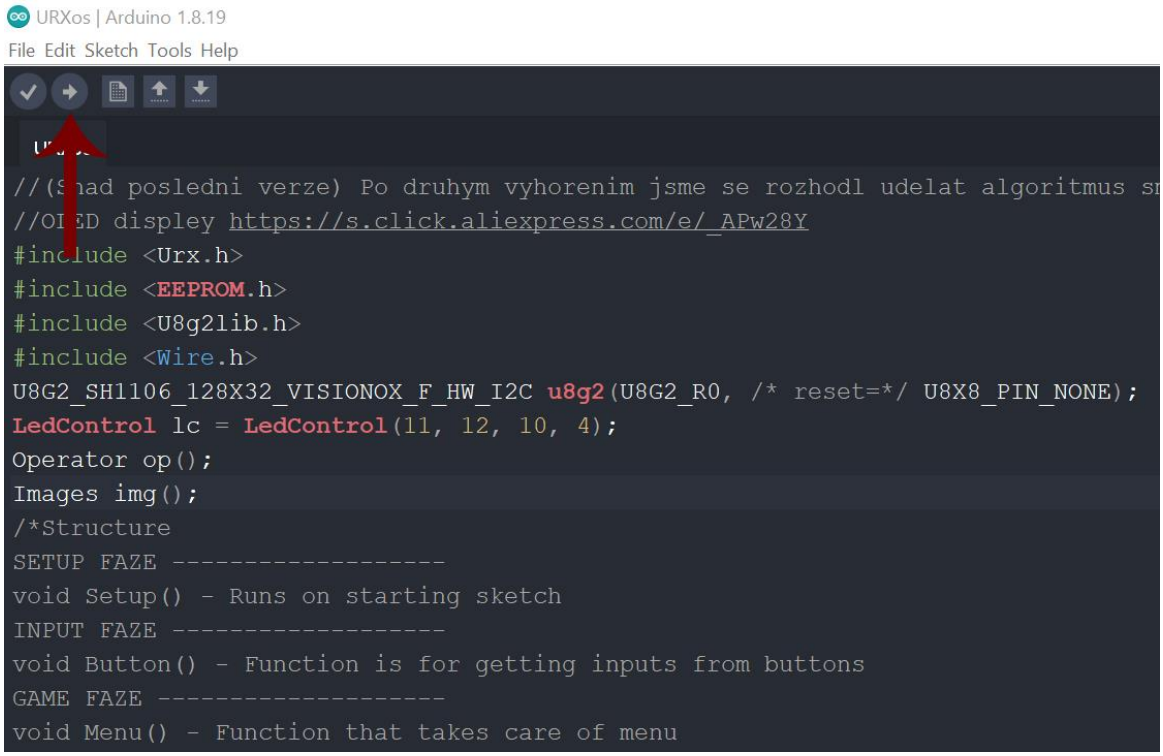






6. Upload code

1. Click on -> Upload (Arrow to the right)
2. If you see this ERROR: *avrdude: jtagmkII_initialize(): Cannot locate "flash" and "boot" memories in description,* don't worry it's an internal error



HURRAH!! If you have followed all the steps you should have the new firmware in your URX, but if you have any problems just go to our discord and ask community what to do.