ROČNÍKOVÁ PRÁCE

Obor: Matematický seminář

Vývoj robota II: měření pH, Bluetooth ovládání

Autor: **Martin Znamenáček**
Konzultant: **Mgr. Čestmír Mniazga**
Škola: **Gymnázium ALTIS s.r.o., Dopplerova 351, 110 00 Praha**
Kraj: **Praha**
Ročník: **7. A (septima), 2020/2021**

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů.

Nemám závažný důvod proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 27. 5. 2021

Martin Znamenáček

Poděkování

Tímto bych velice rád poděkoval svému konzultantovi panu Mgr. Čestmíru Mniazgovi za nápad na samotnou ročníkovou práci a rovněž pak bezednou trpělivost, a to i v této době koronavirové.

Anotace

Ve své práci jsem se i nadále zabýval vývojem robota, kterého jsem tento rok opatřil o pH metr, pro možné domácí či laboratorní měření. Rovněž jsem přidal Bluetooth čip (k bezdrátovému ovládání) a pochopitelně i zdroj energie, 9V baterii.

Software jsem vyvíjel v rámci prostředí Arduino, Bluetooth ovládání pak skrze aplikaci Arduino bluetooth controller z Obchodu Play.

Výsledkem pak byl čtyřnohý pavouk se schopností bezdrátově měřit pH jakéhokoliv roztoku, a to s relativně malou chybovostí.

Klíčová slova

Arduino; pH; Bluetooth; programování; bezdrátové ovládání; robotika.

OBSAH

1 ÚVOD

V práci se soustředím na změny v kódu a způsob zapojení nových součástek. Krátce se také zmiňuji o tom, na jakém principu samotné pH měření funguje.

Zároveň se soustřeďuji na nové možnosti díky přidanému bezdrátovému Bluetooth modulu a rozebírám jeho využití v aplikaci pro chytré telefony.

V závěru rovněž rozebírám další možné vylepšení pro napravení nedostatků či nově vytvořené příležitosti, které jsem nestihl otestovat v tomto roce.

2 CÍLE PRÁCE

- Umístit a zprovoznit pH modul
- Otestovat a zkalibrovat pH měření
- Umožnit bezdrátové ovládání celého robota
- Umožnit bezdrátový feedback z měření robota
- Bezdrátové ovládání zautomatizovat v podobě aplikace pro chytré telefony
- Přidat nové pozice a rotace pro lepší manipulaci robota

3 PROSTŘEDÍ

Veškerý kód pro řízení mikrokontroléru robota byl vyvíjen zapomocí softwaru Arduino IDE.

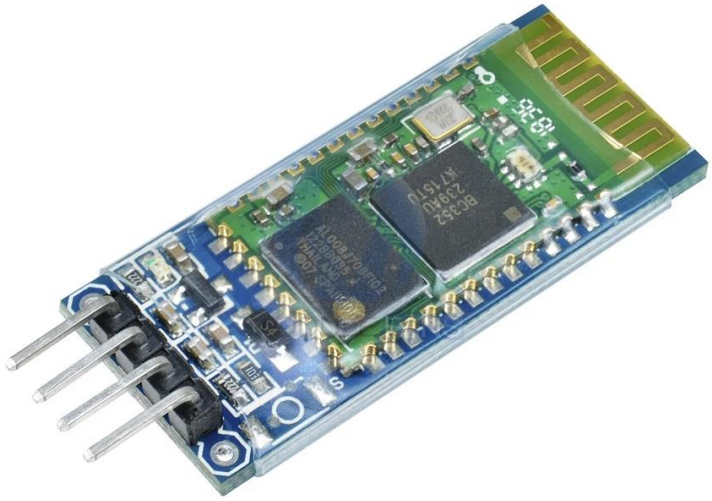
Bluetooth je pak spojený zapomocí vlastní aplikace pro chytré telefony, kterou jsem vyvíjel v prostředí MIT App Inventoru.

4 SOUČÁSTKY

Celkové náklady pro vylepšení robota činily cirka 1 650 Kč. (pH proba - 350 Kč, pH modul – 1 100 Kč, 9V adaptér – 100 Kč, jumper drátky – 100 Kč).

4..1 Bluetooth modul

Jedná se o standardní modul HC-06, který umožňuje bezdrátovou komunikaci s Arduino zařízením.

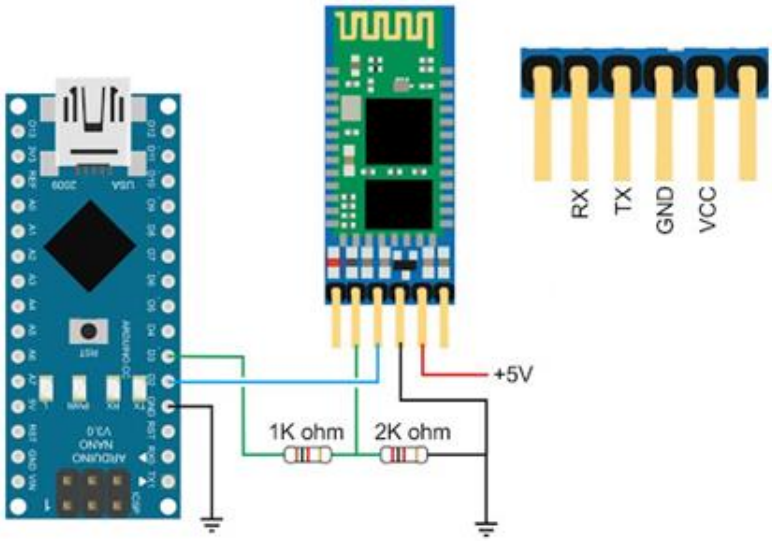


Obrázek 1: Bluetooth čip HC-06

Je třeba podotknout, že jsem se ihned po zapojení setkal s problémem nahrání kódu - je totiž třeba nezapomenout, že HC-06 sdílí porty TX a RX s NANO čipem, takže se kód nikdy nepovede arduino čipu předat. Vždy před nahráním nového kódu je tedy třeba modul HC-06 vypojit a po úspěšném nahrání opět zapojit. Dalším háčkem byl fakt, že TX vývod z NANO čipu se zapojuje do RX portu Bluetooth modulu a zase naopak RX vývod NANO čipu do TX portu Bluetooth modulu.

Tento Bluetooth modul dále vyžaduje usměrnění napájení z 5 V na cirka 3.3 V, čehož jsem docílil dvěma rezistory - 1K a 2K, protože napětí 5 V tak bude usměrněno 2K rezistorem a součtem obou rezistorů 3K, kdy $5 \cdot 2 \div 3$ činí asi 3,33 V.

Zjednodušené konečné zapojení, jež jsem pouze přenesl na bastlicí desku, pak vypadalo takto:



Obrázek 2: Schéma zapojení HC-06 na NANO čip

4..2 9V napájecí modul

Protože jsem se rozhodl udělat robota bezdrátově ovladatelného, musel jsem najít jednoduchý a lehký zdroj energie. Rozhodl jsem se pro velice primitivní adaptér na jednu 9V baterii, která je schopna napájet celého robota po dobu mnoha desítek minut. Jedná se sice o relativně neefektivní způsob napájení, neboť se komponenty pohybují od 3,3 do 5 V, a tak se baterie vybíjí a potenciálně přehřívá daleko rychleji, ale na plošce robota mi již nevybylo místo na kupříkladu Li-Po socket a příslušnou baterii. 9V baterie se naopak perfektně zapasovala do šasi pavouka.

Obrázek 3: 9V napájecí kabel

4..3 pH sonda

Pokud chceme změřit pH kapaliny, je třeba zjistit koncentraci vodíkatých kationtů [H^+]. Pakliže je roztok kyselý, obsahuje velmi mnoho vodíkatých kationtů, naopak, čím je roztok více alkalický, tím je koncentrace kationtů H^+ nižší.

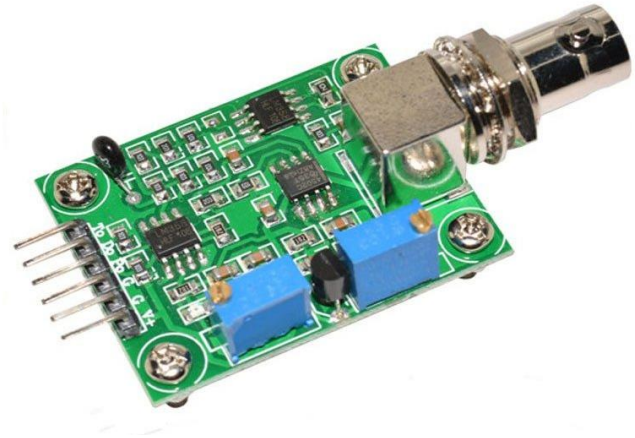
Tyto kationty pak v kapalině vytváří elektrický potenciál, tedy napětí, které je pH proba schopna detekovat. De facto se tedy jedná o velmi přesný voltmetr, který na bázi potenciálního rozdílu, tedy diference mezi nastaveným vzorkem a měřeným roztokem dedukuje reálné pH.

Dále je třeba podotknou, že se jedná o relativně přesný měřič, při ideálních podmínkách (při 5-45 °C) je uváděna odchylka ± 0,05 až 0,4 pH.

Obrázek 4: pH sonda

4..4 pH modul

Vlastní vyhodnocování výsledků pak činí pH modul, který se přes BNC konektor připojuje k výše zmíněné pH sondě.



Obrázek 5: pH modul

Aby ale pH modul správně fungoval, je potřeba jej nejprve zkalibrovat. K tomu slouží modrý trimr, jenž se nachází hned vedle BTC konektoru. Proces kalibrování je pak velice jednoduchý, leč zdlouhavý - šroubkem na trimru se utahuje, či povoluje tak dlouho, dokud nezačne měření být přesné. Míra utažení totiž značí, v jakém rozsahu (od 2.4 do 4.99 V) se pohybuje modulem vysílané napětí, jedná se totiž o variabilní rezistor.

Kalibraci lze ještě učinit v samotném k arduino programu, kde se přesnost měření ovlivní skrze přičítanou konstantu.

Protože je ideální modul kalibrovat třemi různými vzorky, tak jsem se rozhodl použít neutrální vody (pH cirka 7), 8% kyseliny ethanové, tedy octu, s pH asi 3 a v neposlední řadě jsem ještě využil mýdla, které jsem rozpustil ve vodě a naměřil pH něco kolem 10.

5 pH KÓD

Samotný pH kód je velice jednoduchý - nejprve si stanovíme všechny konstanty, které jsou potřeba k převodu hustoty vodíkatých kationtů na pH.

K ideální hodnotě *pH_kalibrace* jsem se dostal již zmíněným postupným testováním. Napětí na měřiči, je od dodavatele stanoveno na 5 V, kanál, přes který modul komunikuje je pak analogový, jež umožňuje 10-bit rozlišení, tedy 2^{10}, což činí finální hodnotu *kanal_merice*, a to 1024.

Vlastní sonda také potřebuje až několik desítek vteřin, aby se měření ustálilo, a tak jsem pro ještě přesnější výsledky vždy naměřil 10 hodnot, které jsem uložil do provizorního pole. Všech 10 měření jsem pak mezi sebou porovnal, aby se hodnoty uskupily od nejmenší po největší, přičemž jsem se nakonec zbavil vždy po 3 výchylkách, a to na maximální i minimální škále. 4 zbylé hodnoty jsem zprůměroval a použil vzorec pro výpočet skutečného pH.

Tento výpočet určuje jednoduchý vzorec: $\frac{-5{,}7 \cdot U_{měř} \cdot A_{read}}{kanal} + kalibrace$

Nakonec jsou výsledky posílány buďto do USB-portem připojeného sériového počítače, či skrze Bluetooth například do telefonu. Přesnost vypisovaných výsledků jsem nastavil na 2 desetinná místa, aby korespondovala s odchylkou. Četnost při neustálém měření jsem pak nastavil na 2 výsledky za každou 1 sekundu.

```
1  const int pH_pin = A0;
2  const float pH_kalibrace = 25.032;
3  const byte U_merice = 5;
4  const int kanal_merice = 1024;
5  const int cetnost_vysledku = 120;
6  float pH = 0;
7  float pH_signal[10];
8  int zaloha;
9  unsigned long int pH_prumer = 0;
10
11 void pH_mereni(int pH_meric) {
12     if (pH_meric) {
13         delay(60000 / cetnost_vysledku);
14         for (int i = 0; i < 10; i++) {
15             pH_signal[i] = analogRead(pH_pin);
16             delay(5);
17         }
18         for (int i = 0; i < 9; i++) {
19             for (int j = i + 1; j < 10; j++) {
20                 if (pH_signal[i] > pH_signal[j]) {
21                     zaloha = pH_signal[i];
```

```
                    pH_signal[i] = pH_signal[j];
                    pH_signal[j] = zaloha;
                }
            }
        }
        for (int i = 3; i < 7; i++) {
            pH_prumer += pH_signal[i];
        }
        pH = ((-5.7 * U_merice * (float) pH_prumer / kanal_merice / 4) +
          pH_kalibrace);
    }
}

void pH_seriovy_vypis(boolean vypsat, byte pH_desetinna_mista) {
    if ((vypsat == true) && (pH > 0)) {
        Serial.print(F("pH: "));
        Serial.println(pH, pH_desetinna_mista);
    }
}
```

6 MĚŘICÍ POZICE

Abych mohl robota použít jakožto pomocníka například při chemických experimentech, musel pro něj vytvořit měřicí pozici, kdy se nejprve de facto napřáhne, tedy zaměří pH sondu a v dalším kroku příslušnou nohu ohne k zemi, a tedy eventuálně ponoří probu do měřeného roztoku.

Pro volný pohyb jedné z nohou jsem zvolil výchozí napřímenou sedavou pozici *vstat(0)*, kterou jsem vyvíjel ještě minulý rok, a ze které se robot připraví do nově vytvořených měřicích pozic, které pochopitelně využívají nastavovače servomotorů rovněž z minulého roku.

```
void zacilit(boolean zacileni) {
    if (zacileni) {
        nastavit_pozici(0, 60, 110, 40, 1);
    }
}

void ponorit(boolean ponoreni) {
    if (ponoreni) {
        nastavit_pozici(0, 60, 110, -50, 1);
    }
}
```

Ze sedavé pozice, kdy má pavouk velmi nízko položené těžiště, a tak perfektní stabilitu, vyzdvihnu nohu č. 0, kterou napřímím, tedy zacílím pro budoucí ponoření do kapaliny.

Jak si lze všimnout, mění se pouze prostorová *z*-souřadnice - noha je tedy nejprve rovnoběžná se zemí a později se o 90° sklopí a zanoří do připravené nádoby s kapalinou.

Obrázek 6: Zacílení

Obrázek 7: Ponoření

7 ROTACE

Aby se s robotem daleko lépe manipulovalo, rozhodl jsem se zavést rotační pohyb, a to jak doleva, tak i doprava.

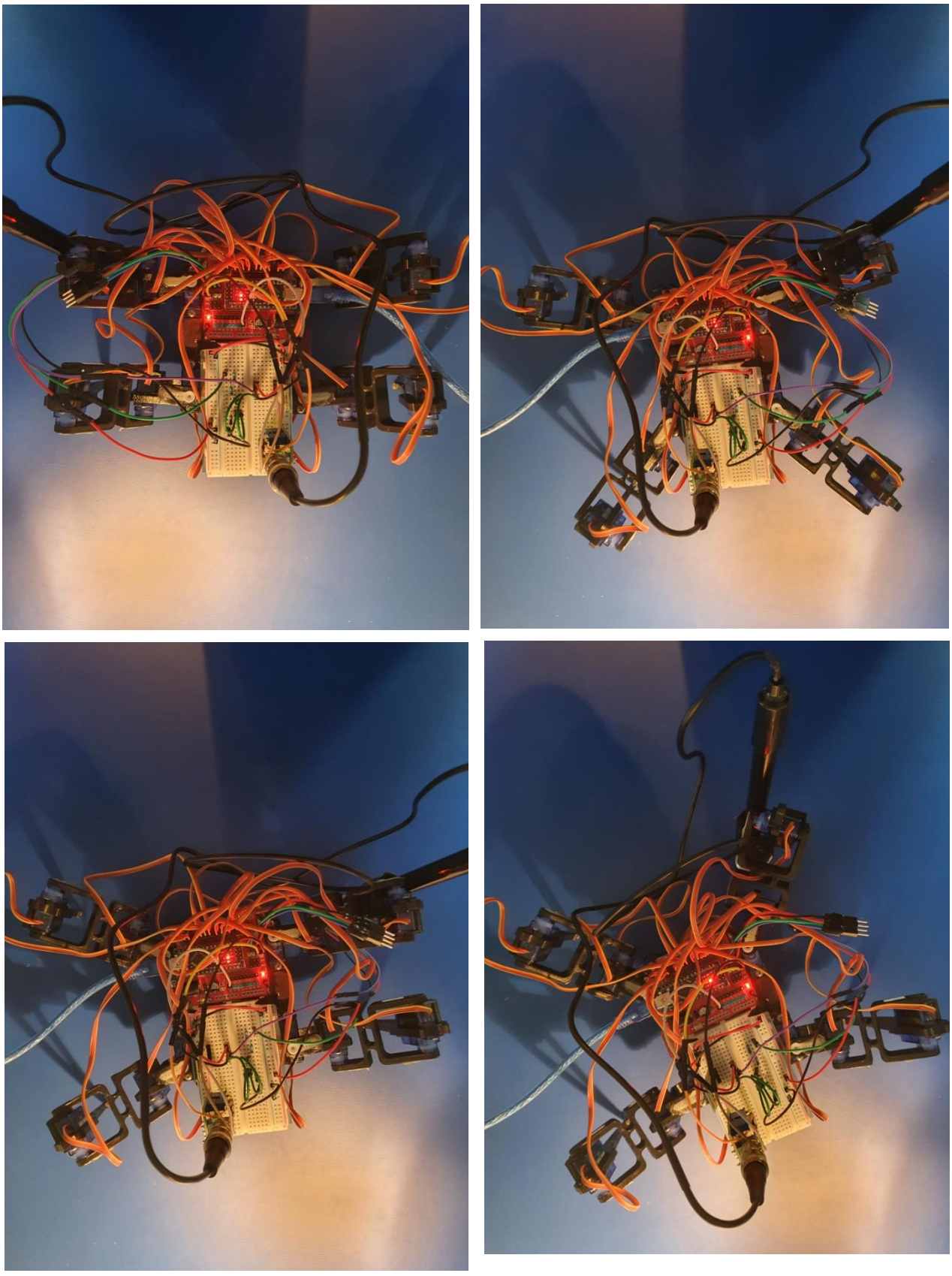
Kromě z minulého roku nadefinovaných konstant jsem si ještě přerýsováním naměřil kýžené rotace pro přetočení a otočení nohy operující na *x* a *y* souřadnici, které jsem v dalším bloku používal k nastavení nových délek pohybu motorů části nohou.

```
1   const float x_pretoceni = 62.1;
2   const float x_otoceni = 31.1;
3   const float y_pretoceni = 20.0;
4   const float y_otoceni = 56.8;
```

Robot se nejprve odpíchne, aby se nepřevážil, dále napřáhne obě zadní nohy, opět se odpíchne, přisune buďto pravou, anebo levou zadní nohu k přední části, opět se odpíchne, konečně přisune druhou zadní nohu a připraví se na další otáčení.

Protože takto robot ale neudělá směr úplně dokola, je potřeba jej ještě připravit na eventuální pokračování v rotaci, při kterém se může část nohy na y-souřadnici propadnout, od čehož slouží podmínka pro nohy 3-Y vlevo a 2-Y vpravo, která zaručí, že bude pohyb plynulý, bez veliké ztráty rovnováhy.

Rotace doleva a doprava jsou si navzájem inversní, takže se u obou děje to stejné (proto také bylo možné celou *void* funkci krásně iterovat), ale vždy pro nohy na opačné straně.



Série obrázků 8: Pohyb doleva (při čtení pozpátku de facto doprava)

```
1  void otocit_vlevo(unsigned int krok) {
2      rychlost_pohybu = rychlost_otaceni;
3      while (krok-- > 0) {
4          if (soucasna_pozice[3][1] == y_start) {
5              otoceni(3, 1, 0, 1, +1, 1, 0, 1, 0, 0, 0);
6          } else {
7              otoceni(0, 2, 1, 0, -1, 0, 1, 1, 0, 0, 0);
8          }
9      }
10 }
11
12 void otocit_vpravo(unsigned int krok) {
13     rychlost_pohybu = rychlost_otaceni;
14     while (krok-- > 0) {
15         if (soucasna_pozice[2][1] == y_start) {
16             otoceni(2, 0, 1, 0, +1, 0, 0, 0, 1, 0, 1);
17         } else {
18             otoceni(1, 3, 0, 1, -1, 0, 0, 0, 0, 1, 1);
19         }
20     }
21 }
22
23 void otoceni(byte a, byte b, byte c, byte d, byte e, byte f, byte g, byte fg, byte h,
       byte i, byte hi) {
24     nastavit_pozici(a, x_start + x_krok, y_start, z_start, 1);
25     nastavit_pozici(0, x_otoceni * c + x_pretoceni * d - x_krok * e, y_otoceni * c +
       y_pretoceni * d, (z_krok * f + z_start * g) * fg + (z_krok) * hi, 0);
26     nastavit_pozici(1, x_otoceni * d + x_pretoceni * c - x_krok * e, y_otoceni * d +
       y_pretoceni * c, (z_krok) * fg + (z_krok * h + z_start * i) * hi, 0);
27     nastavit_pozici(2, x_otoceni * c + x_pretoceni * d + x_krok * e, y_otoceni * c +
       y_pretoceni * d, (z_krok) * fg + (z_krok * i + z_start * h) * hi, 0);
28     nastavit_pozici(3, x_otoceni * d + x_pretoceni * c + x_krok * e, y_otoceni * d +
       y_pretoceni * c, (z_krok * g + z_start * f) * fg + (z_krok) * hi, 1);
29     nastavit_pozici(a, x_otoceni + x_krok, y_otoceni, z_krok, 1);
30     nastavit_pozici(0, x_otoceni * c + x_pretoceni * d + x_krok * e, y_otoceni * c +
       y_pretoceni * d, z_krok, 0);
31     nastavit_pozici(1, x_otoceni * d + x_pretoceni * c + x_krok * e, y_otoceni * d +
       y_pretoceni * c, z_krok, 0);
32     nastavit_pozici(2, x_otoceni * c + x_pretoceni * d - x_krok * e, y_otoceni * c +
       y_pretoceni * d, z_krok, 0);
33     nastavit_pozici(3, x_otoceni * d + x_pretoceni * c - x_krok * e, y_otoceni * d +
       y_pretoceni * c, z_krok, 1);
34     nastavit_pozici(b, x_otoceni + x_krok, y_otoceni, z_start, 1);
35     nastavit_pozici(0, x_start + e * x_krok, y_start + y_krok * c, (z_krok) * fg +
       (z_krok * i + z_start * h) * hi, 0);
36     nastavit_pozici(1, x_start + e * x_krok, y_start + y_krok * c, (z_krok * g +
       z_start * f) * fg + (z_krok) * hi, 0);
37     nastavit_pozici(2, x_start - e * x_krok, y_start + y_krok * d, (z_krok * f +
       z_start * g) * fg + (z_krok) * hi, 0);
```

```
38      nastavit_pozici(3, x_start - e * x_krok, y_start + y_krok * d, (z_krok) * fg +
          (z_krok * h + z_start * i) * hi, 1);
39      nastavit_pozici(b, x_start + x_krok, y_start, z_krok, 1);
40  }
```

8 BLUETOOTH OVLADAČ

Zde se nachází tělo kódu, jedná se o hlavní look funkci, která neustále skenuje pro možné zadané příkazy a eventuálně vysílá povely k jejich vykonání.

Pakliže je zaslána číslice, nastaví ji jako novou iterační hodnotu, která se používá například k definici počtu opakování pohybů vpřed, zároveň se hodila na testování a vylaďování kódu, či ověřování výpočtů pro pohyb.

```
1   char BT_prikaz;
2   char BT_zmena;
3   int BT_iterace = 1;
```

```
4   void loop() {
5       if (Serial.available()) BT_prikaz = Serial.read();
6       if (isDigit(BT_prikaz)) {
7           BT_iterace = BT_prikaz - '0';
8           Serial.println(F("NOVÁ HODNOTA ITERACE NASTAVENA NA: "));
9           Serial.println(BT_iterace);
10          BT_prikaz = 'z';
11      }
```

Všechny funkce jsou vždy typické svým unikátním znakem, který je aktivuje, přičemž jejich funkčnost se odvíjí od již popisovaných funkcí.

Funkce *v* pro pohyb vpřed; *q* pro sesednutí a následné *f* a *u*, tedy zacílení, respektive ponoření nohy opatřené o pH probu.

```
12      if (BT_prikaz == 'v') {
13          Serial.println(F(" ... CHŮZE VPŘED ... "));
14          vstat(1);
15          pohyb_vpred(BT_iterace);
16      }
```

```
17      if (BT_prikaz == 'q') {
18          Serial.println(F("... SEDNUTÍ ... "));
19          vstat(0);
20      }
```

```
21      if (BT_prikaz == 'f') {
22          Serial.println(F("... ZACÍLENÍ ..."));
23          zacilit(1);
24      }
```

```
25      if (BT_prikaz == 'u') {
26          Serial.println(F("... PONOŘENÍ ..."));
27          ponorit(1);
28      }
```

Pro rotační pohyby slouží navzájem si inversní *l* vlevo a *r* vpravo.

```
29      if (BT_prikaz == 'l') {
30          Serial.println(F(" ... OTÁČENÍ VLEVO ... "));
31          otocit_vlevo(BT_iterace);
32      }
```

```
33      if (BT_prikaz == 'r') {
34          Serial.println(F(" ... OTÁČENÍ VPRAVO ... "));
35          otocit_vpravo(BT_iterace);
36      }
```

Jakožto pojistka či ukončovací funkce slouží *s*.

```
37      if (BT_prikaz == 's') {
38          Serial.println(F("UKONČUJI FUNKCE... 0 (VYP)"));
39          vstat(0);
40          pH_mereni(0);
41      }
```

Asi jediné složitější je *p*, které zahájí měření pH, avšak proti omezení náporu na modul a zatížení celého systému se vždy po dokočení 1 měření ukončí - samo totiž aktivuje příkaz *s*.

```
42      if (BT_prikaz == 'p') {
43          Serial.println(F(" ... MĚŘENÍ pH ... "));
44          pH_mereni(1);
45          pH_seriovy_vypis(1, 2);
46          BT_prikaz = 's';
47      }
```

Na konci každého jednoho *loopu* je potom stanovený Bluetooth příkaz na *z*, které nemá žádnou definici, tedy ukončí či přeruší všechny ostatní příkazy a započne *loop* od samého počátku, a to zcela načisto.

```
48      BT_prikaz = 'z';
49  }
```

9 VÝVOJ APLIKACE K TELEFONU

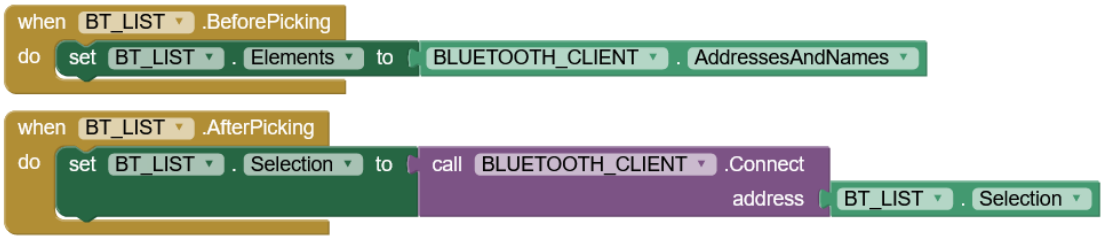
Abych nemusel robota ovládat pouze přes terminál, tak jsem se rozhodl naprogramovat ještě mobilní aplikaci. Zvolil jsem zdarma dostupného online prostředí App Inventor Massachusettského technologického institutu (na doméně http://ai2.appinventor.mit.edu).

Jde o jednoduchý DnD (Drag and Drop) systém, který nevyžaduje valnou znalost programovacího jazyka, avšak pracuje v blocích, tedy se skládá jako puzzle.

Nejprve jsem si vytvořil provizorní grafiku s 9 tlačítky (pro příkazy - ovládání) a textový box (v horní části obrazovky), který vypisuje přijatá data v textové podobě.

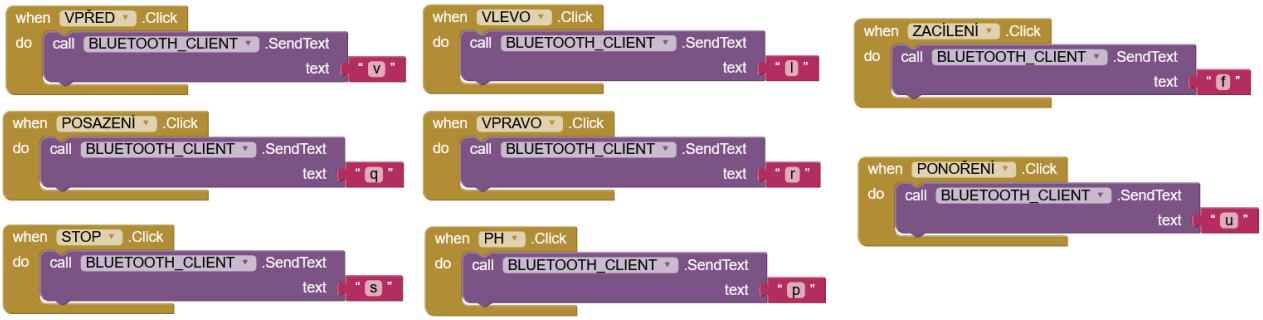
Obrázek 9: Grafika aplikace

Vytvořil jsem tlačítko pro výběr dostupných a již spárovaných Bluetooth zařízení. Nejprve se tedy zobrazí list všech možných přijímačů, ze kterého se vybere kýžené HC-06.



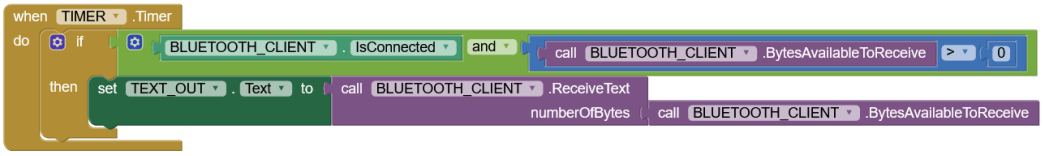
Obrázek 10: Puzzle bloky pro příjem a zvolení BT zařízení

Dále jsem vygeneroval celkem 8 tlačítek pro všechny předem popsané příkazy. Funguje tak, že se stisknutím tlačítka pošle do Bluetooth terminálu znak, který si Arduino již převede na *void* funkci, kterou spěšně vykoná.



Obrázek 11: Převedení tlačítka na příkaz pro Arduino

V neposlední řadě se ještě zkontroluje, jestli není k dispozici, k přijetí, nějaký objem dat - pakliže ano, vypíše se v horním textovém boxu. Protože se jedná o periodicky vykonávající se příkaz, je třeba nejprve užít časovače, timeru.



Obrázek 12: Kontrola dostupných dat

10 POTENCIÁLNÍ BUDOUCÍ VYUŽITÍ

Například by robot mohl sloužit jako automatický pomocník či vyhodnocovač výsledků třeba při chemických titracích. Lze robota totiž naprogramovat tak, aby graficky znázornil naměřené pH hodnoty a došel k tzv. kalibrační křivce, která popisuje vývoj pH na základě množství titračního činidla.

Tento graf bychom pak mohli převést na server a z něho, třeba skrze třeba Python, automaticky vyhodnotit výsledky - skrze jednoduché algoritmy lze například z kalibrační křivky vyvodit, zda jsme pracovali se slabou či silnou kyselinou a zásadou, a to na bázi počátečních a konečných hodnot pH, dle jejich relativní vzdálenosti od y-souřadnic, tedy minima 0 - maxima 14. Zároveň lze jednoduše určit, jestli se jednalo o monosytnou kyselinu či zásadu, a to podle počtu inflexních bodů - pro monosytnou bude 1, a to přímo v equilibriu.

Robot je ale v současném stádiu téměř přetížen váhově a co se kapacity NANO čipu týče - nezbývá moc volných pinů a maximálního možného množství operovatelných modulů je téměř zaplněné. Pro budoucí vylepšení by tak nejspíše bylo potřeba navrhnout PCB desku pro zlehčení kabeláže a zmenšení použité plochy. Zároveň by nebylo od věci zakoupit výkonnější čip, třeba Arduino Mega.

11 ODKAZY

Veškerý Arduino kód (.ino) - zaslaný emailem

Aplikace (apk.) vytvořená v prostředí MIT App Inventor - dostupná na Google Drive:
https://drive.google.com/file/d/1vyofM_O3k_dUVFmOXO9upuOtu61h9Idp/view?usp=sharing