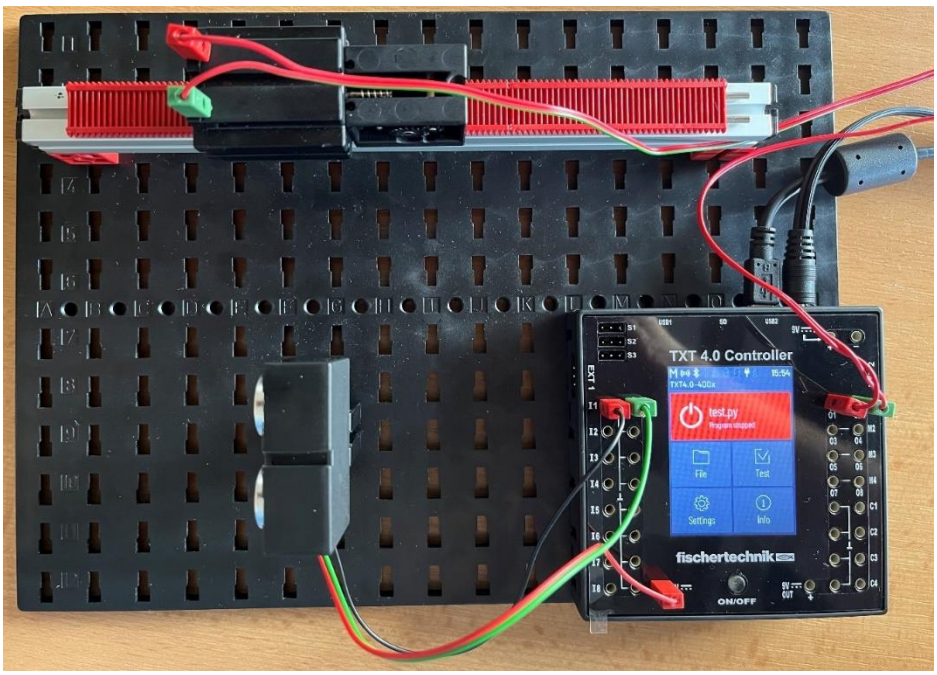
ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA STROJNÍ
KATEDRA PRŮMYSLOVÉHO INŽENÝRSTVÍ A MANAGEMENTU




# Základy robotiky – programování hardwarových modelů

Autoři:
Doc. Ing. Zdeněk Ulrych, Ph.D.
Ing. Miroslav Malaga


# Základy robotiky – programování hardwarových modelů

Autoři:
Doc. Ing. Zdeněk Ulrych, Ph.D.
Ing. Miroslav Malaga

Vydala:
Západočeská univerzita v Plzni
Univerzitní 8, 301 00 Plzeň

První vydání, 112 stran
Plzeň 2023

**ISBN 978-80-261-1144-3**

# Obsah

# 1 Úvod

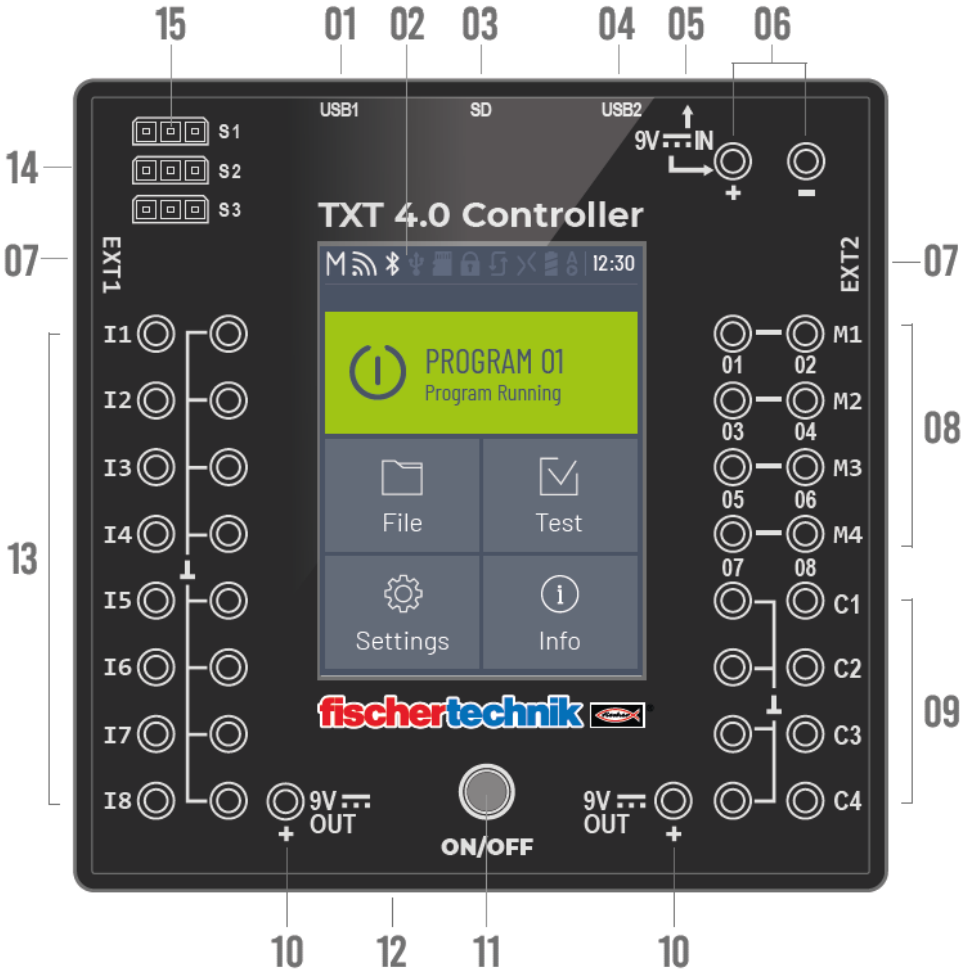
Pro cvičení k předmětu Počítačová podpora ve strojírenství se využívá stavebnice Fischertechnik 559888 ROBOTICS TXT 4.0 Base Set, díky které se studenti seznámí s tematickými moderními technologiemi. V případě předmětu KPV/PPVS jsou to hlavně základy kybernetiky, řídicích systémů, senzorové techniky, motorizace, automatizace, robotiky, digitální komunikace a programování. Okrajově pak s mobilní robotikou, konstruováním, mechanickými systémy a návrhem systémů. Kromě řízení modelů pomocí řídící jednotky Fischertechnik TXT 4.0 jsou uvedeny i příklady pro řízní pomocí jednotky BBC Micro:bit a I/O adaptéru.

# 2 Základní přehled stavebnice pro potřeby KPV/PPVS

Řídící jednotka slouží k ovládání postavených modelů. Řídící programy se vytváří pomocí vývojového prostředí ROBO PRO Coding a do řídící jednotky je lze nahrát přes USB kabel, nebo bezdrátově (WIFI, Bluetooth).

## 2.1 ROBOTICS TXT 4.0 Controller (řídící jednotka)

Řídící jednotka ROBOTICS TXT 4.0 Controller, viz následující obrázek a popis, je malý počítač s výkonem mobilního telefonu nižší řady s operačním systémem postaveným na Linuxu.

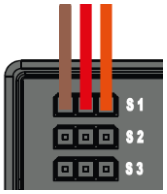



*Obrázek 2-1: ROBOTICS TXT 4.0 Controller (řídící jednotka) [1]*

1. USB A vstup, v případě uvedené stavebnice pro USB kameru.
2. Displej s dotykovým ovládáním.
3. Vstup pro Micro SD kartu. Lze využít pro rozšíření paměti.
4. Mini USB vstup. Slouží pro propojení řídící jednotky s počítačem a následně zajišťuje např. přenos zkompilovaných programů do řídící jednotky. Přes USB kabel není řídící jednotka napájena. Vždy je potřeba použít baterii, nebo napájecí zdroj.
5. Vstup pro napájení síťovým zdrojem
6. Vstupy pro napájení bateriovým setem (9 V). Dodržujeme pravidlo, že PLUS je červený drát, MINUS/ZEM je zelený drát.
7. Rozšiřující rozhraní – tato připojení lze použít k propojení dalších řídicích jednotek ROBOTICS TXT 4.0, čímž se rozšíří počet řídicích jednotek a celkový dostupný počet vstupů a výstupů.

Kromě toho obsahuje piny pro využití rozhraní I2C, například pro senzory komunikující pomocí této sběrnice.

8. Výstupy (M1 – M4, O1 – O8). Jsou to výkonové výstupy, které dodávají 9V elektřiny jakékoliv součásti, která je k nim připojena. Mělo by se jednat pouze o motory, lampy, bzučáky nebo elektromagnety. Připojení M1 – M4 jsou diferenciální výstupy, což umožňuje chod motorů v obou směrech. 01 – O8 se používá se zemí a může běžet 8 různých výstupu pouze v jednom směru.
9. Vstupy (C1 – C4). Snímají digitální stav 1/0. Vstupy pro rychlé počítání, záznam počítacích impulsů až do 1 kHz (1000 impulsů/s). Využívají se například pro encodéry na motorech – tedy pro krokový motor, pokud chceme řídit otáčky, nebo jej synchronizovat s jiným krokovým motorem. V pravém sloupečku je PLUS, v levém ZEM.
10. Napájení 9 V s neustálým napětím – neřídí se programem.
11. Zapnutí/vypnutí řídící jednotky.
12. Reproduktor.
13. Univerzální vstupy (I1 – I8). Lze je využít jako digitální, snímající digitální stav 0/1, nebo jako analogové, které čtou buď napětí, nebo proud. V levém sloupečku je PLUS, v pravém ZEM. Lze je nastavit pomocí softwaru ROBO Pro Coding. pro:
    - Digitální senzory (tlačítka, jazýčkové kontakty, fototranzistory) - Digitální 5 kΩ
    - Infračervené snímače dráhy – digitální 10 V
    - Analogové senzory 0-5 kΩ (NTC rezistory, fotorezistory, potenciometry)
    - Analogové senzory 0-10V (barevné senzory) zobrazují hodnotu v mV (milivoltech).
    - Ultrazvukové snímače vzdálenosti
14. V Originálních materiálech, ze kterých je obrázek převzatý, je chyba ukazující pozici 14 (materiály r. 2022). Na této pozici nic není 😊
15. Připojení servopohonů S1-S3. - 3pólová koncovka pro připojení servomotoru Fischertechnik 132292. Dbejte na správnou polaritu!




### 2.1.1 Možnosti napájení řídící jednotky baterií

Řídící jednotku lze napájet baterií. Tato možnost je vhodná hlavně pro samostatně pohyblivé projekty typu autíčko, vozík na materiál apod., nebo při práci se stavebnicí mimo dosah el. zásuvek. Zapojení při využívání napájení z baterie viz Obrázek . Je nutné zachovat správnost propojení, tedy PLUS baterie na PLUS řídící jednotky a MINUS baterie na MINUS řídící jednotky.


*Obrázek 2-2: Napájení řídící jednotky baterií*

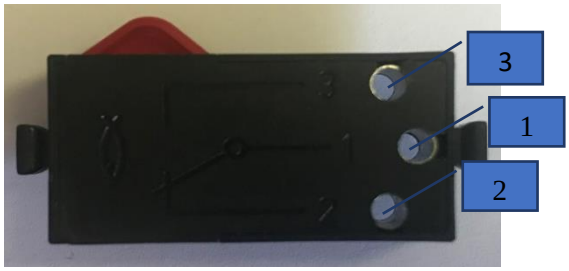
## 2.2 Základní prvky pro potřeby KVP/PPVS

V této kapitole jsou popsány základní sensory a aktuátory využívané se stavebnicí Fischertechnik.

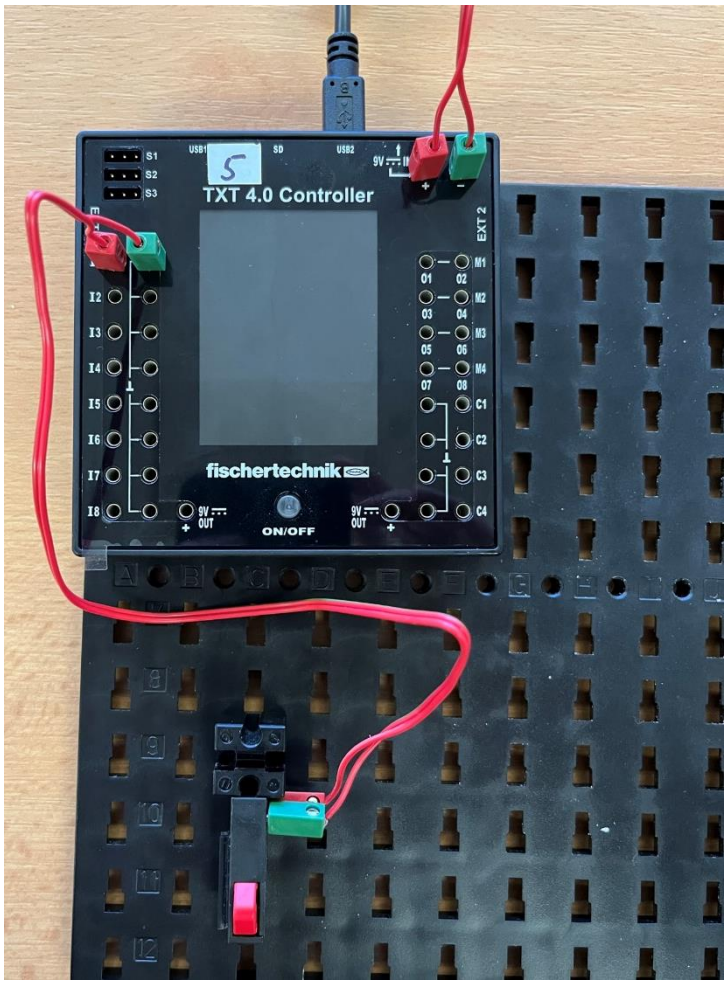
### 2.2.1 Spínač

V tomto případě se jedná přesně řečeno o koncový spínač. Je to zařízení, které v elektrotechnice slouží ke spínání a rozepínání elektrických obvodů v koncových polohách. Velmi jednoduše se dají v tomto případě využít i jako tlačítka, která např. spouští některou z naprogramovaných procedur. Využívají se jako vstupy informací, zapojujeme je tedy do řídící jednotky do pinů I(1–8).

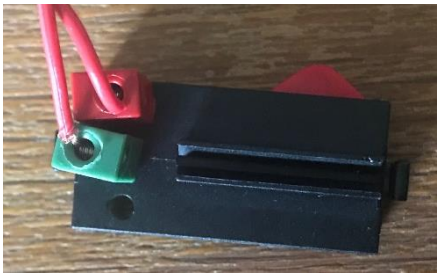
- Při zapojení 1–3 je obvod rozepnutý a stiskem tlačítka se uzavře.
- Při zapojení 1–2 je obvod zapnutý a stiskem tlačítka se rozepne.
- Pokud jej používáme jak tlačítko spouštěcí nějakou událost, využíváme na tlačítku piny 1 a 3.




*Obrázek 2-3: Spínač (Koncový spínač)*




*Obrázek 2-4: Spínač – ukázka možného zapojení do řídící jednotky na vstup I1*



*Obrázek 2-5: Spínač – detail zapojení do spínače*

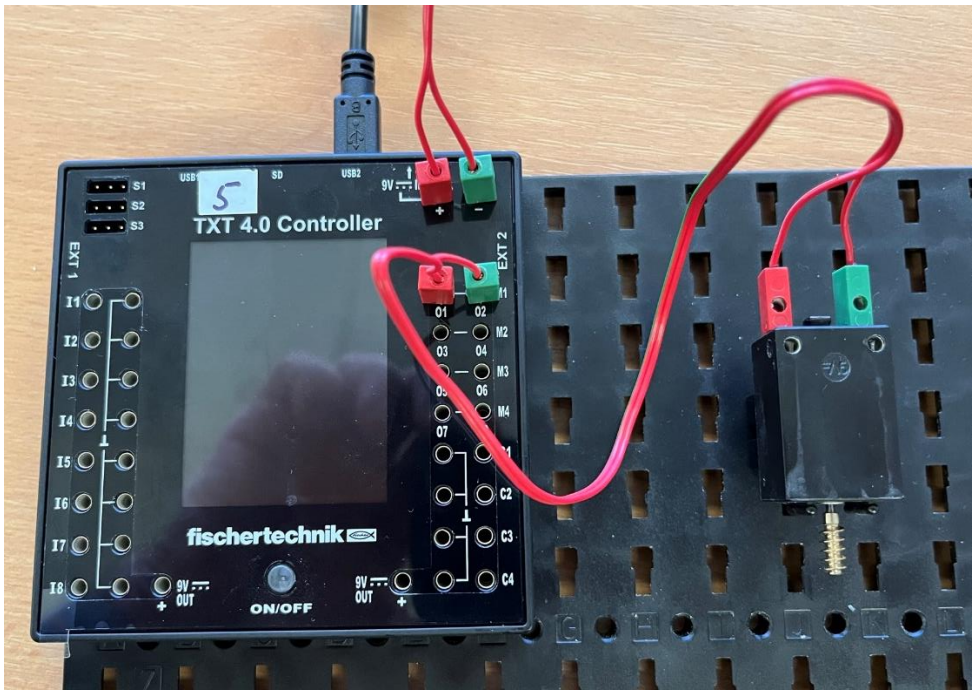
### 2.2.2 Mini motor a XS motor

Jedná se o jednoduché elektromotory, u kterých lze bez dalších přidaných prvků pouze řídit rychlost otáček a směr otáček. Motory mini a XS se liší pouze velikostí a možnostmi mechanického přichycení ke konstrukci. Zapojujeme je do pinů M(1–4).

Princip elektromotoru je založený na využití silových účinků magnetického pole. Zjednodušeně lze říci že využíváme vzájemné přitahování a odpuzování dvou elektromagnetů. Sílu a polaritu těchto elektromagnetů můžeme řídit velikostí protékajícího elektrického proudu (regulace rychlosti otáček).

*Obrázek 2-6: Mini motor*

*Obrázek 2-7: XS motor*



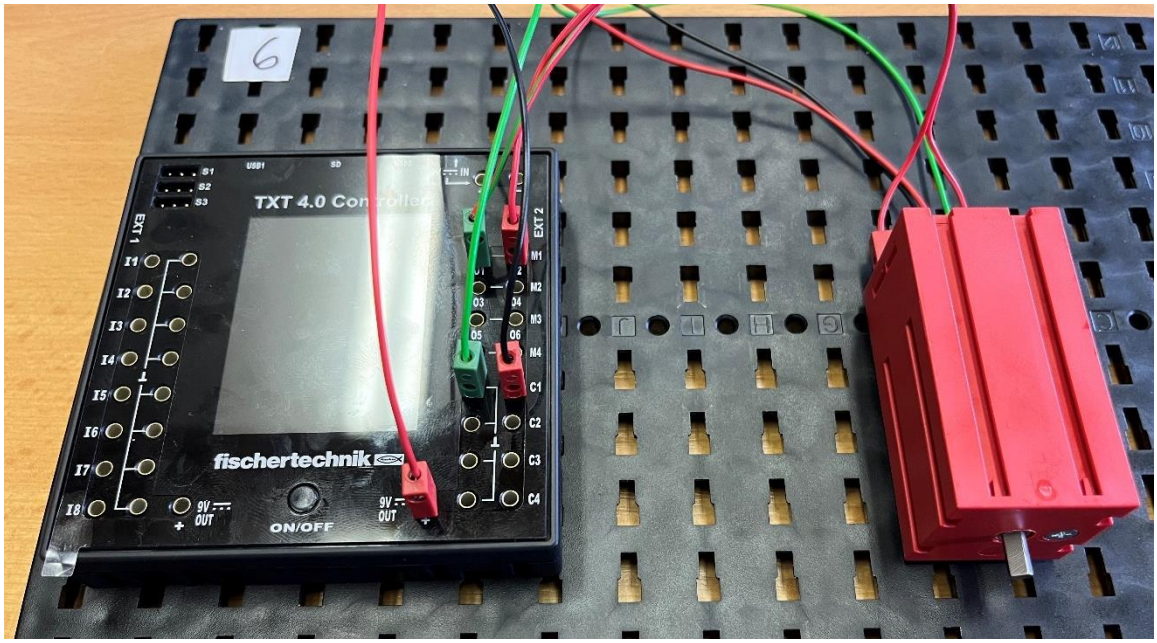

*Obrázek 2-8: Mini motor – ukázka možné zapojení na výstup M1*

### 2.2.3 Krokový motor

Jedná se o elektromotor, který je navíc vybaven převodovou hlavou, která snižuje otáčky na výstupu a zvyšuje točivý moment. Navíc je vybaven obvodem, který slouží k počítání otáček motoru, díky čemuž je bez dalších přidaných součástí možné přesně řídit chod motoru, nebo jej lze např. synchronizovat s druhým zapojeným motorem (stejného typu). Pro řízení otáček, nebo synchronizaci motoru je potřeba použít k normálnímu zapojení motoru ještě 3žílový kabel, kde:

- Červený kabel se zapojuje na +9V
- Zelený kabel je zem
- Černý kabel slouží pro přenos signálu a zapojuje se na C(1–4)

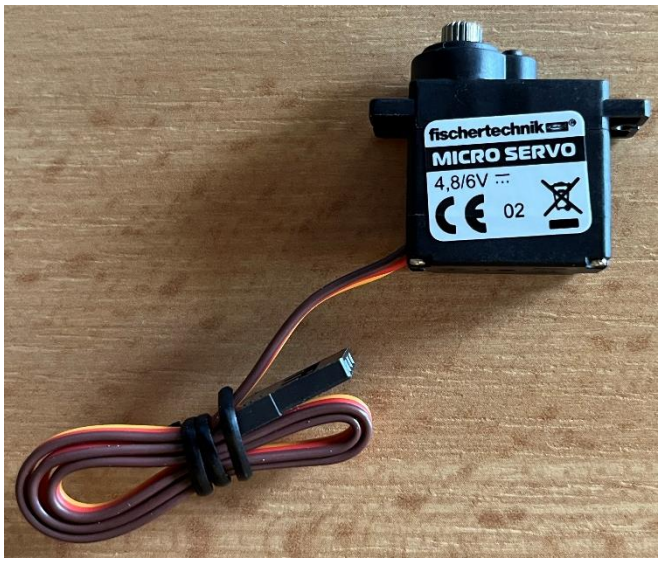
*Obrázek 2-9: Krokový motor*



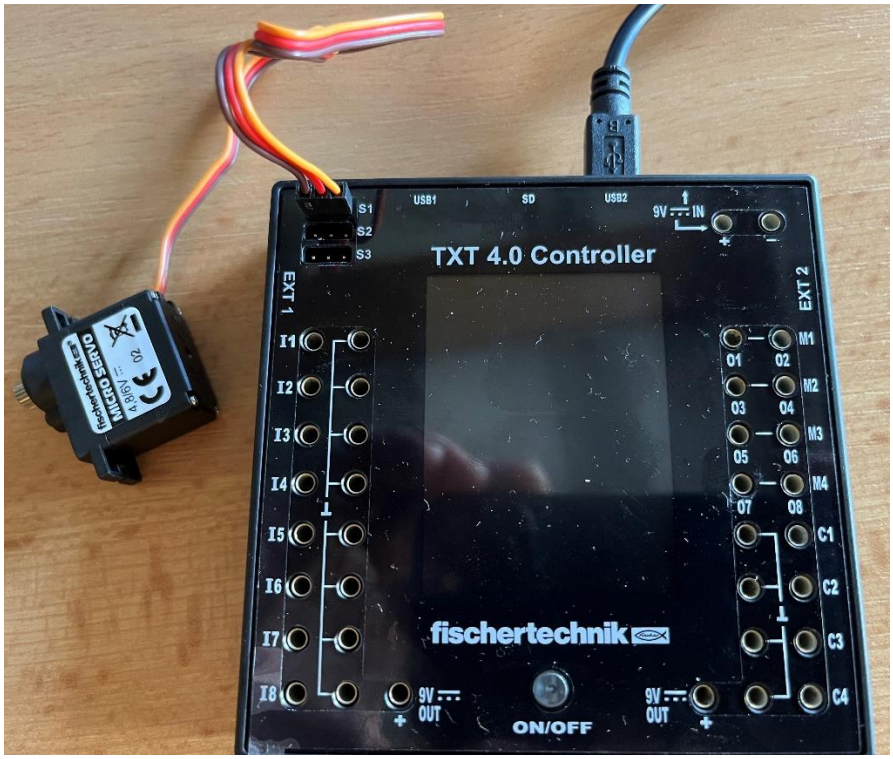

*Obrázek 2-10: Krokový motor – ukázka zapojení na výstupy M1(motor) a C1 + 9 V (řízení motoru)*

### 2.2.4 Servomotor

Servomotor je motor pro pohony (většinou elektrické), u kterých lze na rozdíl od běžného motoru nastavit přesnou polohu natočení osy. Ovládají se jím například posuvy u CNC strojů. Všechny RC (Radio controlled) modely používají malá modelářská serva. Pro připojení servomotoru k řídící jednotce Fischertechnik TXT 4.0 slouží výstupy S1 – S3. Předchozí generace jednotek (s názvem Fischertechnik TXT Controller) tyto výstupy nemá.




*Obrázek 2-11: Servomotor*

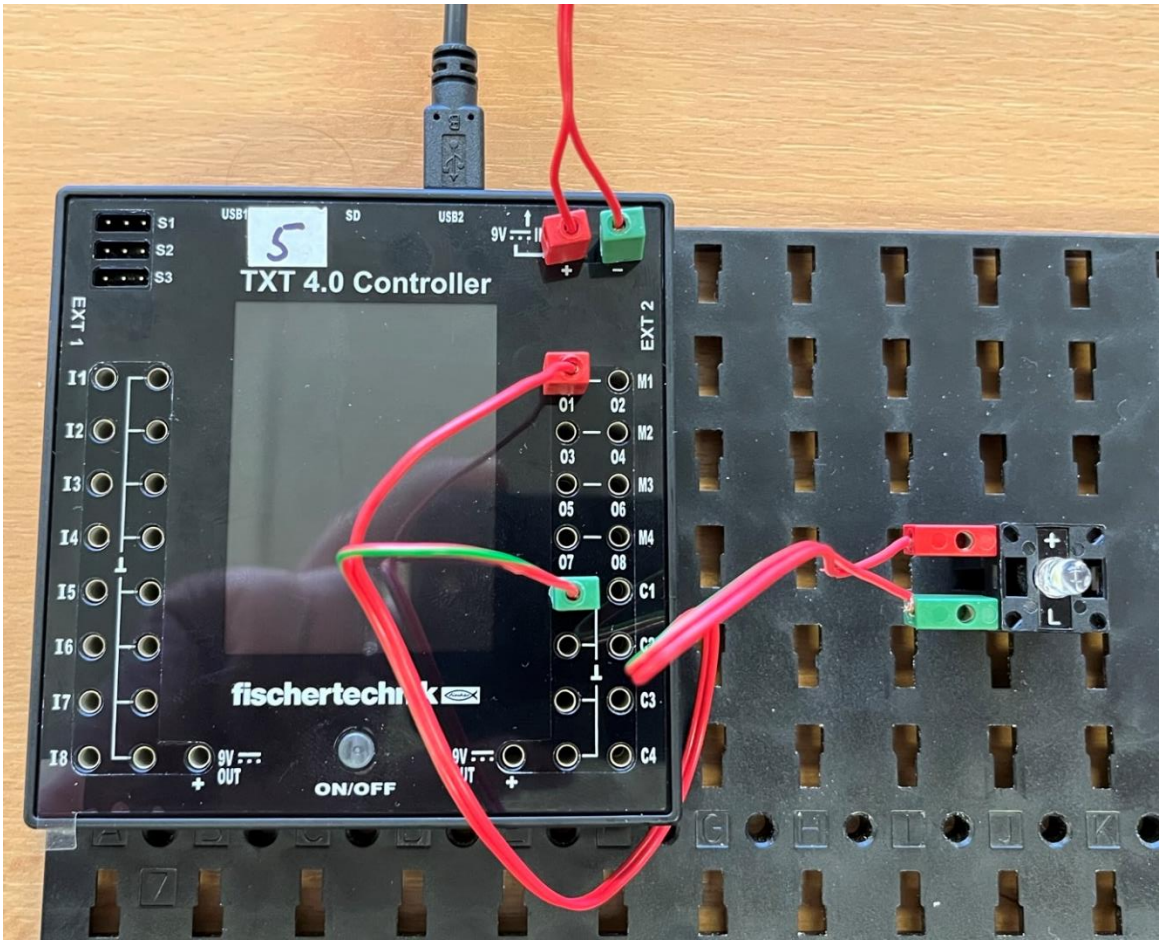



*Obrázek 2-12: Servomotor – ukázka zapojení na výstup S1*

### 2.2.5 LED

Light-Emitting Diode, česky elektroluminiscenční dioda, popř. světelná dioda. V elektrotechnice se jedná o diodu, která emituje světlo, případně infračervené, nebo ultrafialové záření. LED mohou vydávat jak teplé, tak studené světlo (záleží na typu LED). V tomto konkrétním případě emituje světlo. Zapojení je na piny M(1–4, nebo O(1-8) a zem. Je potřeba dbát na to, aby v případě LED bylo zapojeno PLUS řídící jednotky na PLUS diody!

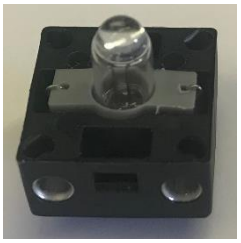

*Obrázek 2-13: LED komponenta*



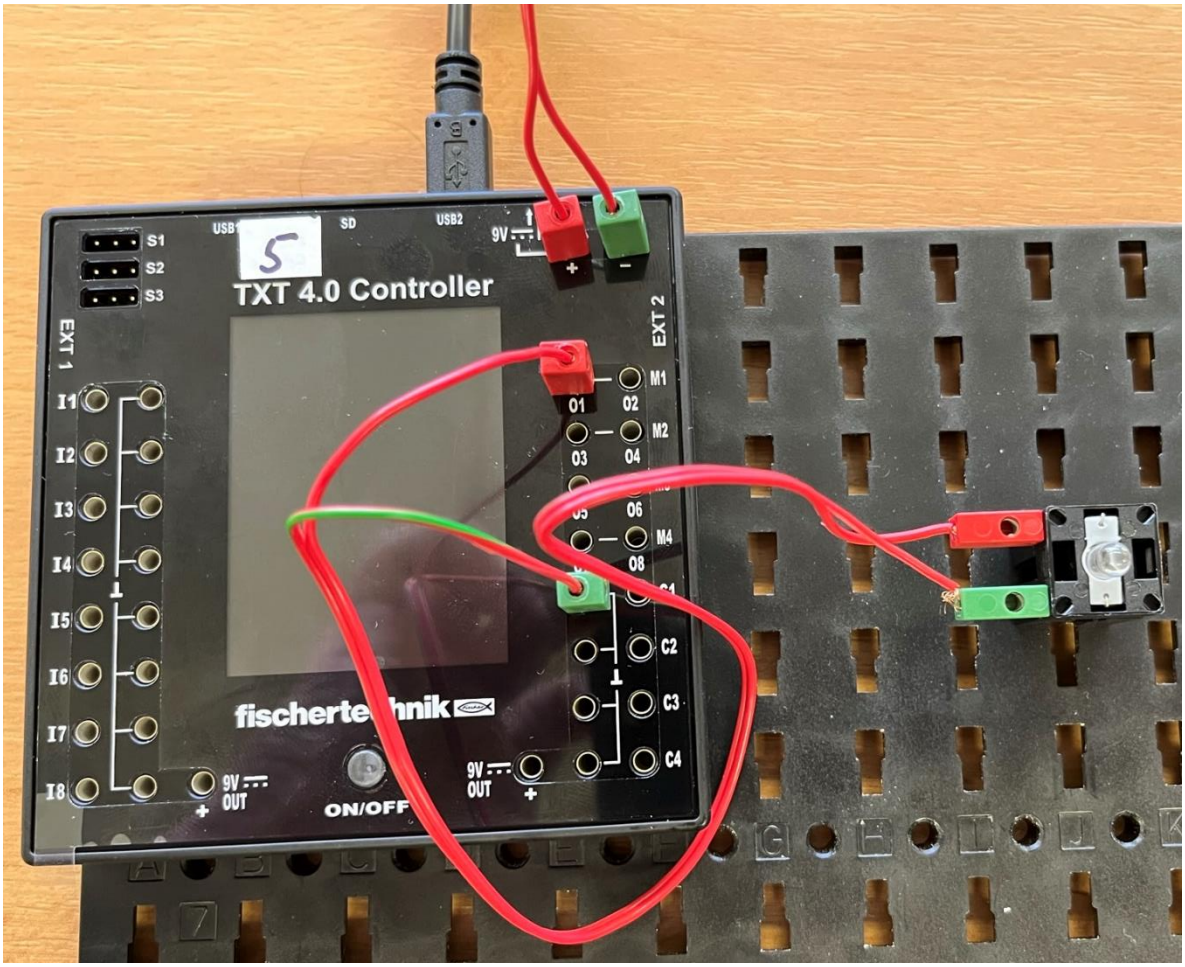

*Obrázek 2-14: LED – ukázka zapojení na výstup O1 a zem*

### 2.2.6 Žárovka

V tomto případě se jedná opravdu o obyčejnou žárovku, tedy nejjednodušší zařízení k přeměně elektrické energie na světlo. Funguje na principu zahřívání tenkého vodiče elektrickým proudem. Žárovka vždy vydává teplé světlo. Na řídící jednotku ji standardně napojujeme na piny O(1-8) a zem. Nestandardně lze zapojit i na výstupy M(1–4), ale pak se musí řídit přes element určený motoru.



*Obrázek 2-15: Žárovka*

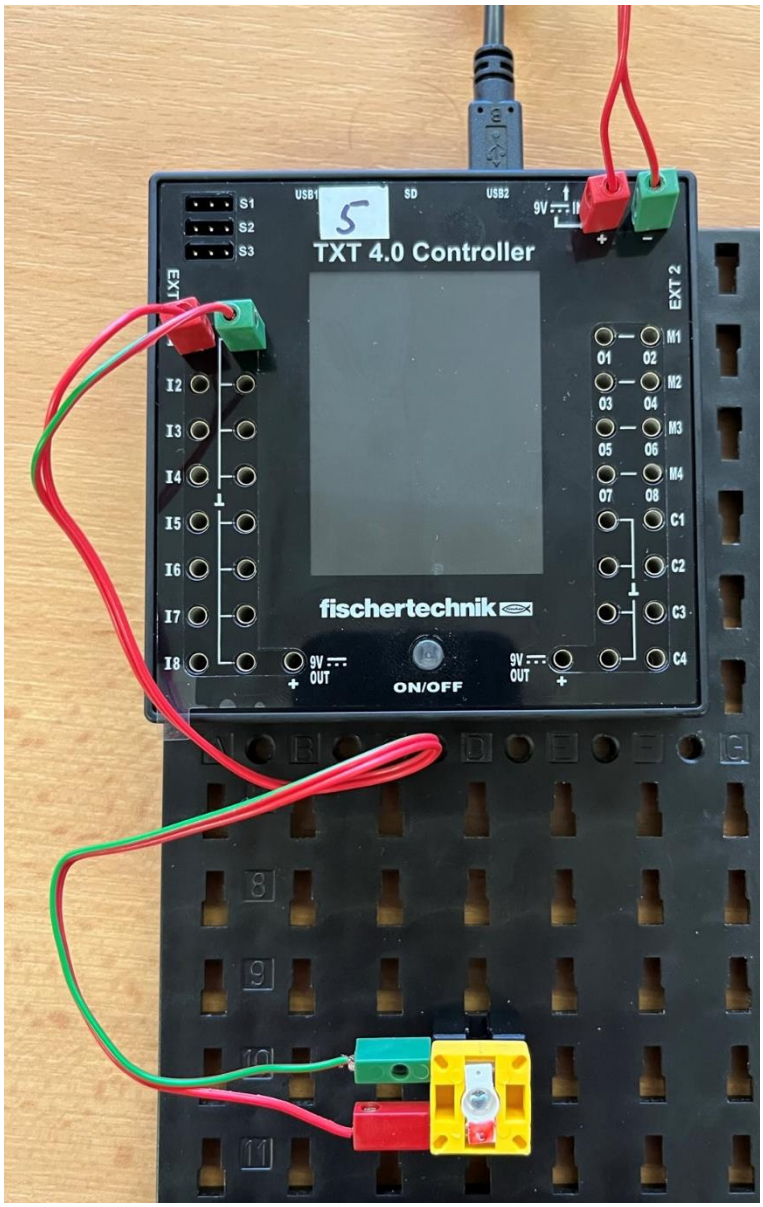



*Obrázek 2-16: Žárovka – ukázka zapojení na výstup O1 a zem*

### 2.2.7 Fototranzistor

Tato součástka slouží k vyhodnocení, jestli na ni dopadá světlo a bodle toho má stav 1, nebo 0. Lze ji např. tedy využít pro automaticky spouštěný větráček – svítí-li na fototranzistor světlo, bude automaticky spuštěn větráček. Při zapojení je potřeba dbát na to, aby PLUS bylo přivedeno na červeně označený vstup fototranzistoru! Protože slouží jako vstupní zařízení – dává nám informaci, jestli světlo dopadá/nedopadá (a to následně vyhodnocujeme), je potřeba jej zapojovat na piny I(1–8).

*Obrázek 2-17: Fototransistor*



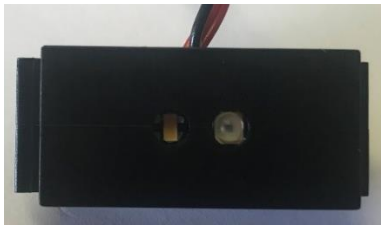

*Obrázek 2-18: Fototranzistor – ukázka zapojení na vstup I1*

### 2.2.8 Optický barevný senzor

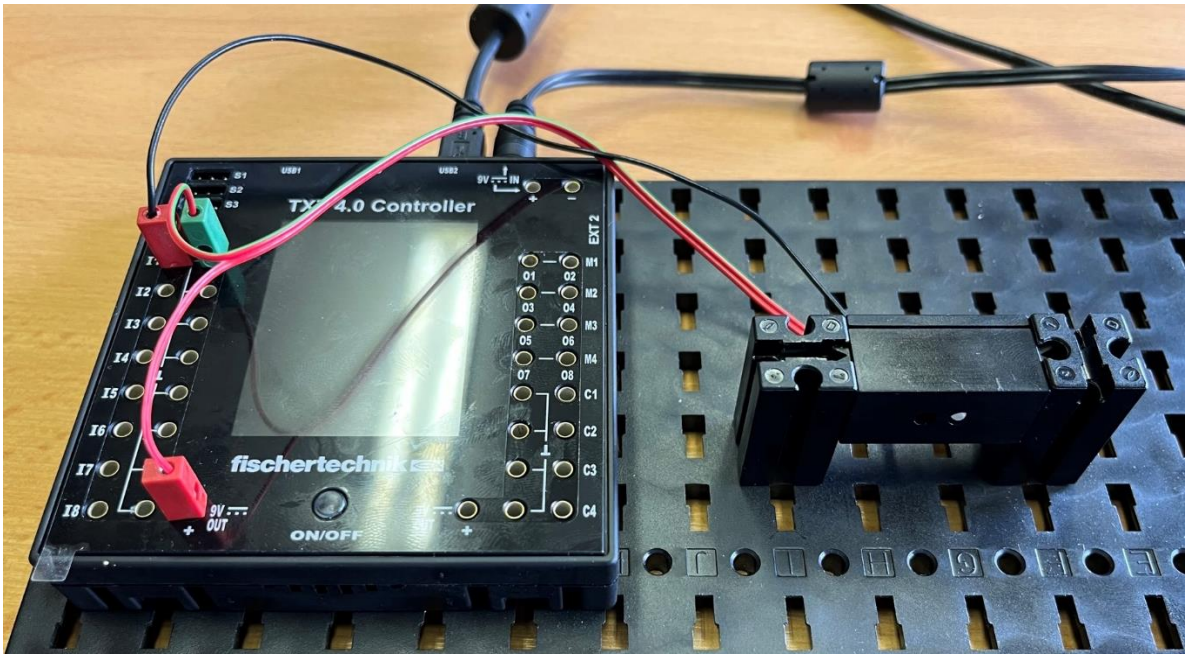
Optický barevný senzor měří vlastnosti povrchových barev, nebo světelných zdrojů. Hodnoty, získané senzorem se budou lišit v závislosti na osvětlení místnosti, vzdálenosti od snímaného povrchu a tvaru snímaného povrchu. Senzor se skládá z jasné LED, která prostor před senzorem osvětluje a receptoru, který zachycuje odražené světlo (princip je postavený na tom, že materiály absorbují různá množství světla na základě barvy, typu materiálu a textury povrchu). Z výše uvedeného vyplývá, že je potřeba jej vždy kalibrovat podle okolních podmínek, tedy ve vlastním kódu ošetřit, že naměřená (vrácená) hodnota optickým senzorem za daných podmínek odpovídá např. červené barvě. Optický senzor nám totiž nedá barvu, ale číselnou hodnotu, které my musíme barvu přiřadit. Neměla by to být jedna hodnota = barva, ale „rozumný interval hodnot“ = barva. Je to dáno tím, že při snímání barvy senzorem nám senzor může pokaždé vrátit lehce jinou hodnotu. Protože se jedná o zařízení, které hodnoty čte a předává je řídí jednotce, tak jej zapojujeme na piny I(1–8), nebo C(1-4). Toto zařízení vyžaduje zapojení na 9 V, proto může vypadat zapojení např. takto[1]:

[1] Bude ukázáno na konkrétním praktickém příkladu

- Černý kabel (sběr informací) – I1
- Zelený kabel (zem) – zem
- Červený kabel (+) – napájení +9 V



*Obrázek 2-19: Optický barevný senzor*

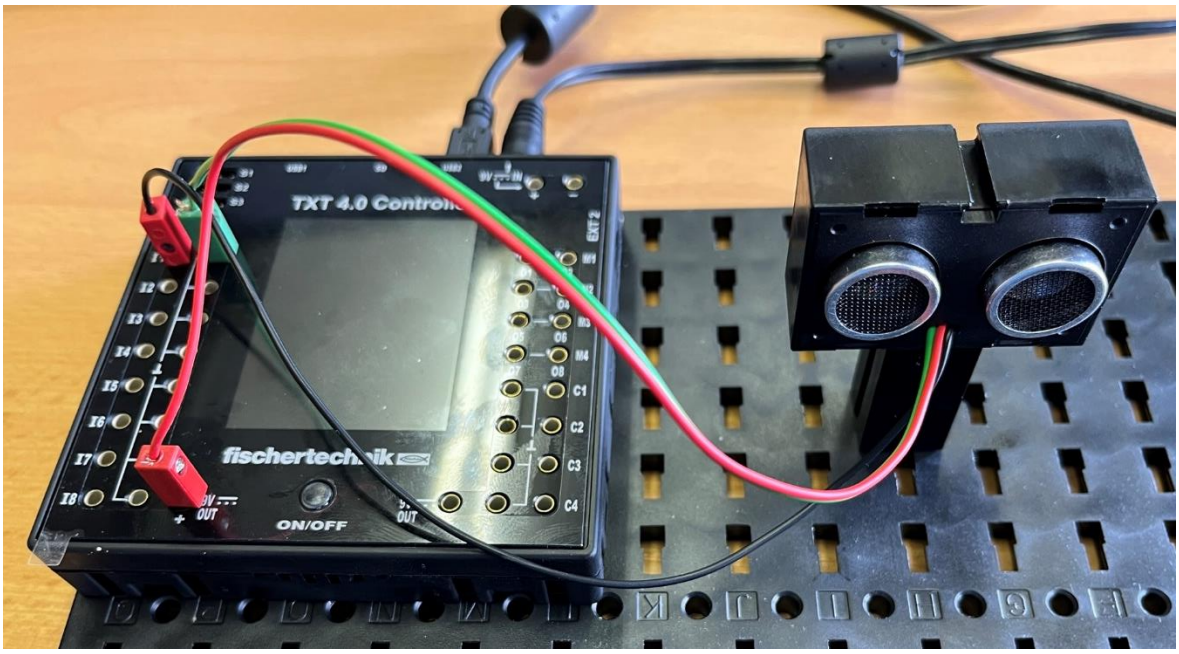



*Obrázek 2-20: Optický barevný senzor – ukázka zapojení na vstup C1 a +9V*

### 2.2.9 Ultrazvukový sensor vzdálenosti

Sensor vzdálenosti se skládá z ultrazvukového vysílače a přijímače. Jeho využití je např. u robotů a AGV, kde je potřeba s poměrně velkou přesností. Princip měření vzdálenosti ultrazvukem spočívá v měření doby od vyslání zvukového pulzu do jeho zachycení po odrazu od překážky. Na základě znalosti rychlosti šíření zvuku v daném prostředí můžeme vypočítat vzdálenost předmětu od snímače.

*Obrázek 2-21: Ultrazvukový sensor vzdálenosti*

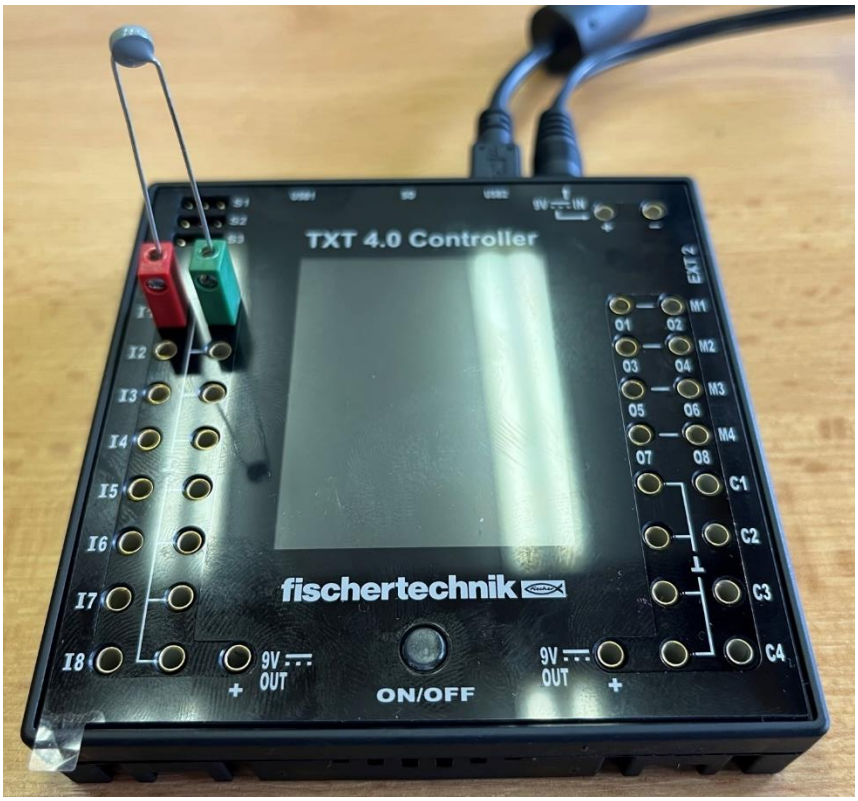



*Obrázek 2-22: Ultrazvukový sensor vzdálenosti – ukázka zapojení na vstup I1 a +9V*

### 2.2.10 NTC rezistor

Česky také Termistor. Jedná se o součástku, jejíž elektrický odpor je závislý na teplotě – proto je možné jej využít pro její měření. Abychom hodnoty získané z termistoru byli schopni převést např. na °C, musíme znát VA charakteristiku termistoru, která však není lineární. TZN. musíme mít k dispozici např. převodní tabulku od výrobce termistoru, nelze použít pro převod trojčlenku. NTC rezistor (nebo také negastor) je termistor s negativním teplotním koeficientem, tedy se zahřátím součástky odpor klesá.

*Obrázek 2-23: NTC rezistor*

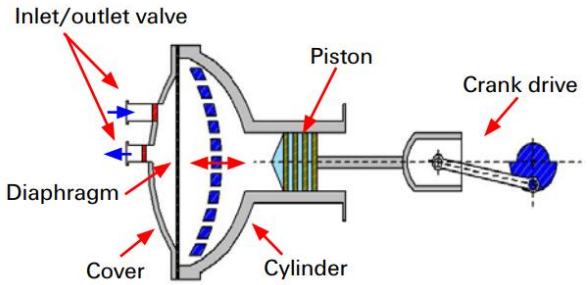



*Obrázek 2-24: NTC rezistor – ukázka zapojení na vstup I1*

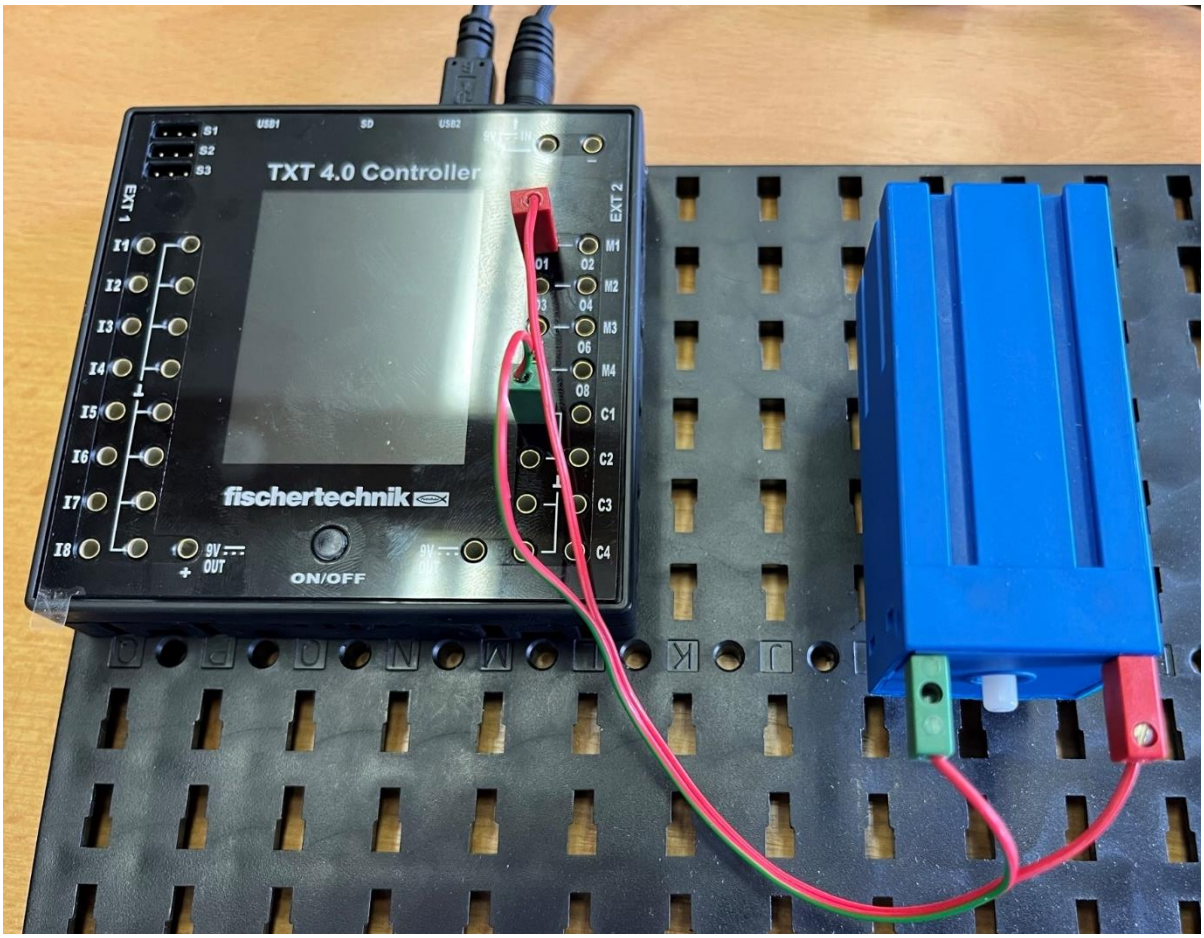
### 2.2.11 Vzduchový kompresor

Vzduchový kompresor je jednoduše řečeno motorová pumpa vzduchu. Větší, popř. složitější pneumatické systémy doplňují vzduchový kompresor vzduchovou nádrží, která slouží jako rezervoár pro udržení tlaku vzduchu např. i při vypnutí kompresoru. Kompresor je standardně možné zapojit jak na 9V+ pin a zem z pinu I8 (pak poběží kompresor neustále), tak i na piny O(1-4) a zem, kdy můžeme řídit jeho zapnutí/vypnutí jen na potřebnou dobu. Nestandardně lze zapojit i na výstupy M(1–4), ale pak se musí řídit přes element určený motoru.

*Obrázek 2-25: Kompresor*



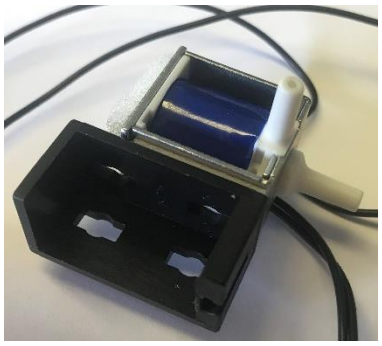

*Obrázek 2-26: Princip membránového kompresoru [2]*



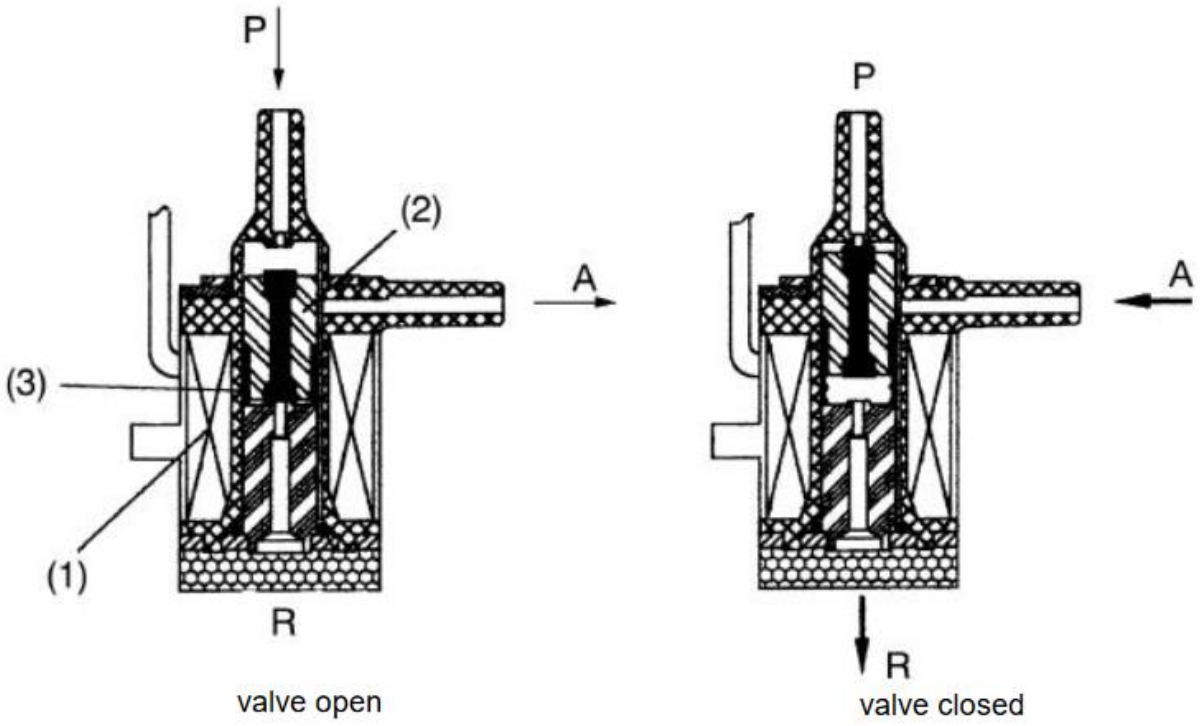

*Obrázek 2-27: Vzduchový kompresor – ukázka zapojení na výstup O1*

### 2.2.12 Solenoidový ventil

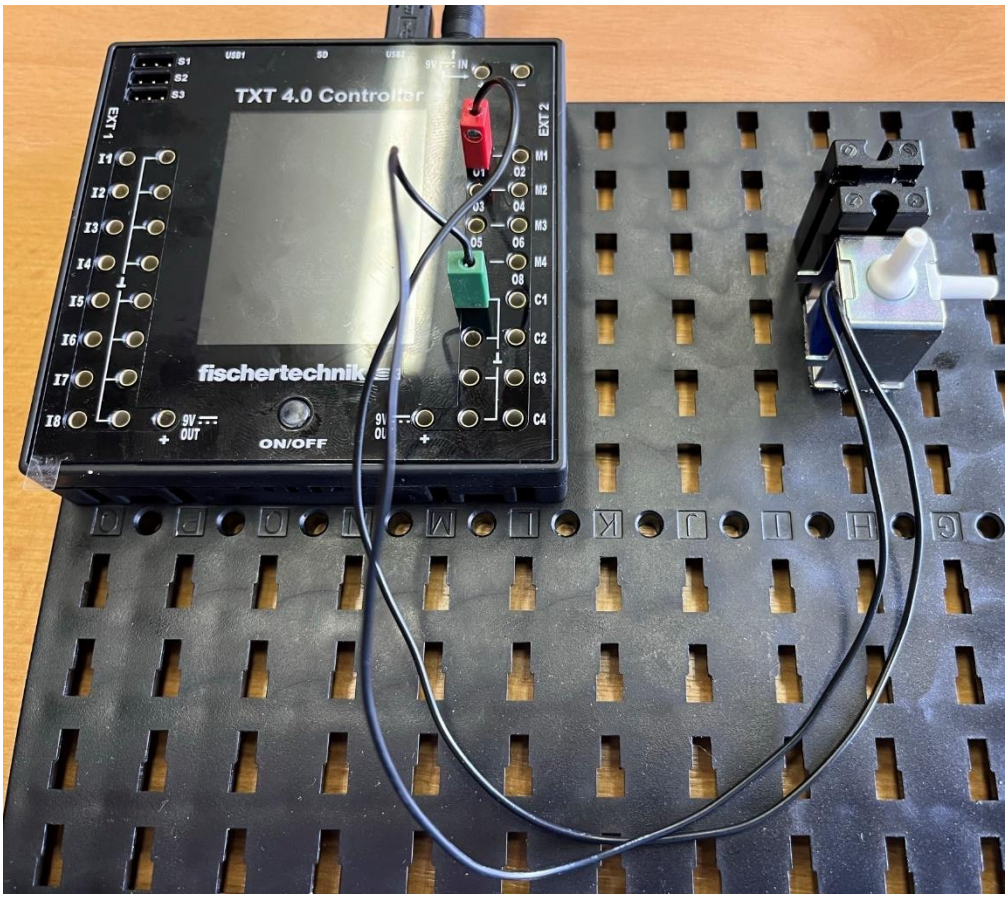
Solenoidový ventil je elektromagneticky ovládaný ventil. V tomto případě se jedná o 3/2 cestný ventil. Tzn. že v jedné poloze dochází k průtoku tekutiny a ve druhé poloze k odklonění přivedené tekutiny – tedy jsou 2 cesty kapaliny. Trojka v označení znamená, že jsou 3 vstupy/výstupy. První dva jsou na obrázku na první pohled patrné, třetí je zezadu ventilu překrytý molitanem, který slouží jako tlumič. Ventil se zapojuje do pinů O(1-8) a zem na řídící jednotce.



*Obrázek 2-28: Solenoidový ventil*




*Obrázek 2-29: Princip fungování solenoidového ventilu [1]*




*Obrázek 2-30: Solenoidový ventil – ukázka zapojení na výstup O1*

### 2.2.13 Pneumatický válec

Pneumatický válec je mechanické zařízení sloužící k převodu síly stlačeného vzduchu na mechanický pohyb. V tomto případě se jedná o jednočinný válec, tedy při přivedení vzduchu se pístnice vysune a její vratný pohyb obstará pružina uvnitř válce.

*Obrázek 2-31: Pneumatický válec (jednocestný)*

### 2.2.14 Vakuové sací zařízení

V tomto případě se jedná o speciální koncovku z velmi přizpůsobivé gumy. Při přiložení koncovky a vytvoření podtlaku se předmět „přisaje“ a je možné jej např. přemístit. Nutným předpokladem je, že uchopovaný předmět má vhodnou povrchovou strukturu.

*Obrázek 2-32: Vakuové sací zařízení*

### 2.2.15 USB kamera

Jedná se o klasickou webovou kameru s připojením do USB. Kamera disponuje manuálním zaostřením. Lze ji využít např. pro vyhodnocování předmětů, hledání/následování cesty apod.

*Obrázek 2-33: USB kamera*


*Obrázek 2-34: USB Kamera – ukázka zapojení*

# 3 Základní modely řízené kontrolérem Fischertechnik TXT 4.0

V této kapitole jsou ukázány základní příklady modelů včetně jejich řídících programů.

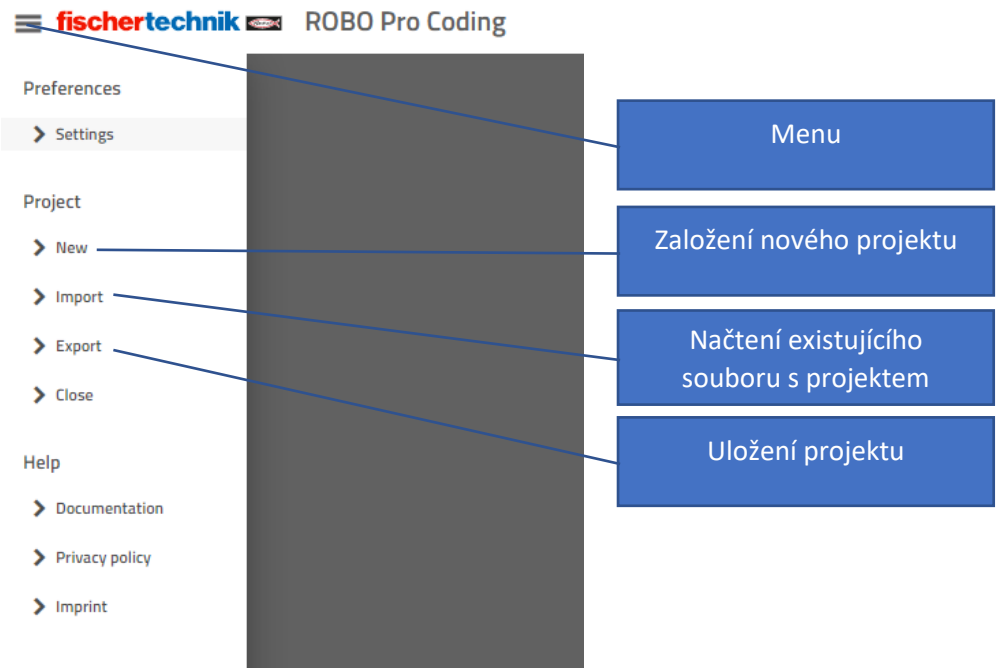
Ukázkové příklady pro modely a řídící jednotky Fischertechnik jsou tvořeny v Robo Pro Coding grafickým způsobem. Ukázané pythony kódy jsou pouze automaticky generované z grafického kódu.

## 3.1 Větráček – základní podrobné cvičení

Větráček je velmi jednoduché zařízení s jedním spínačem a motorem, který otáčí vrtulí. Pro sestavení tedy potřebujeme právě tyto dvě komponenty a řídící jednotku. Kromě základního zapojení si ukážeme i nastavení komunikace mezi počítačem a řídící jednotkou, otestování komponent (bez potřeby software), vytvoření jednoduchého programu a jeho spuštění na modelu větráčku.

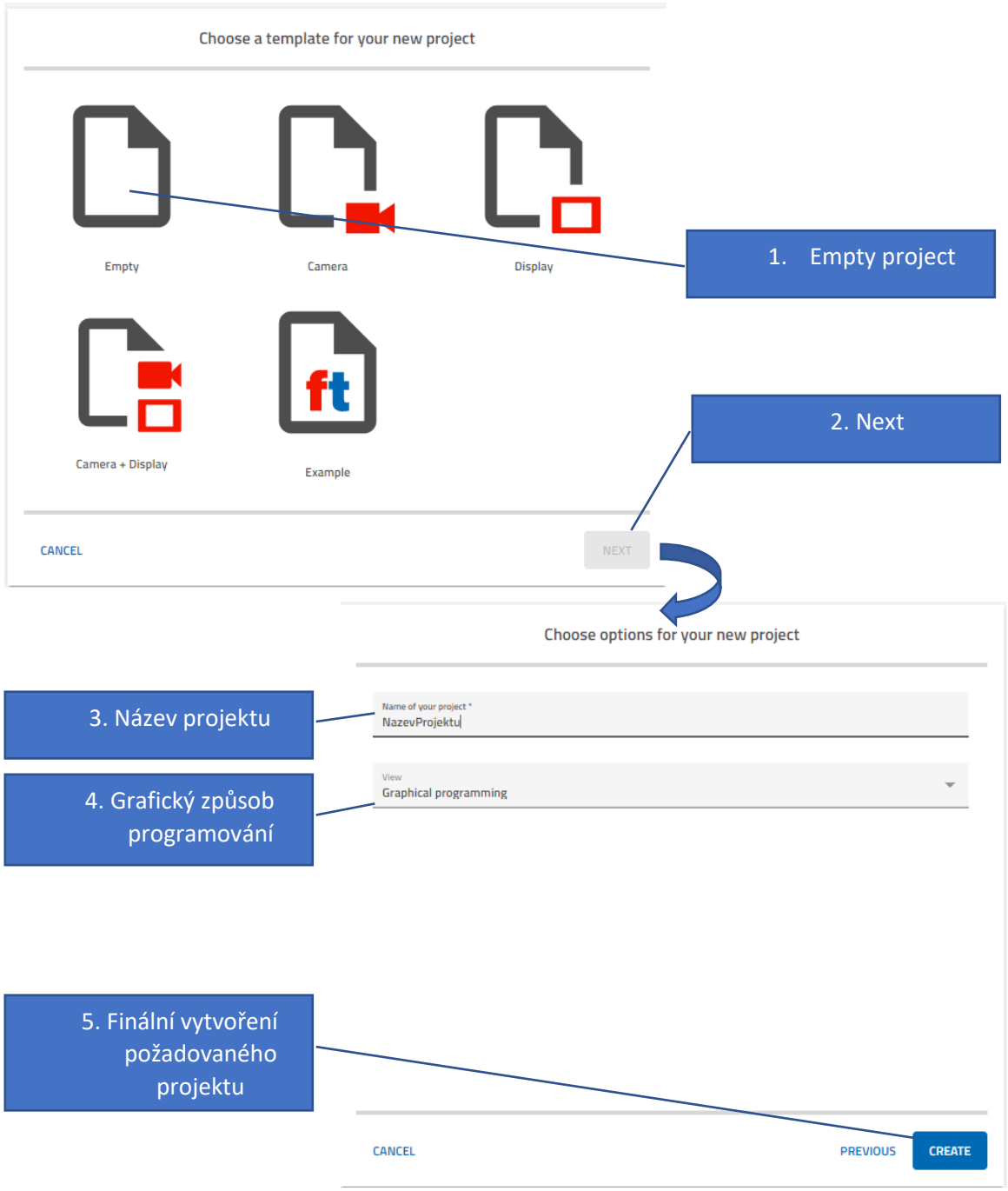
## 3.2 Založení nového, uložení a otevření existujícího projektu

Pokud se neotevře přímo okno vyzívající k volbě typu nového projektu, tak se nový projekt založí přes menu, možnost „NEW“. Pro uložení rozpracovaného projektu do souboru slouží volba „Export“, pro načtení souboru pak volba „Import“. Ukázka viz následující obrázek.




*Obrázek 3-1: ROBO PRO Coding – založení nového projektu, načtení existujícího projektu a uložení projektu do souboru*

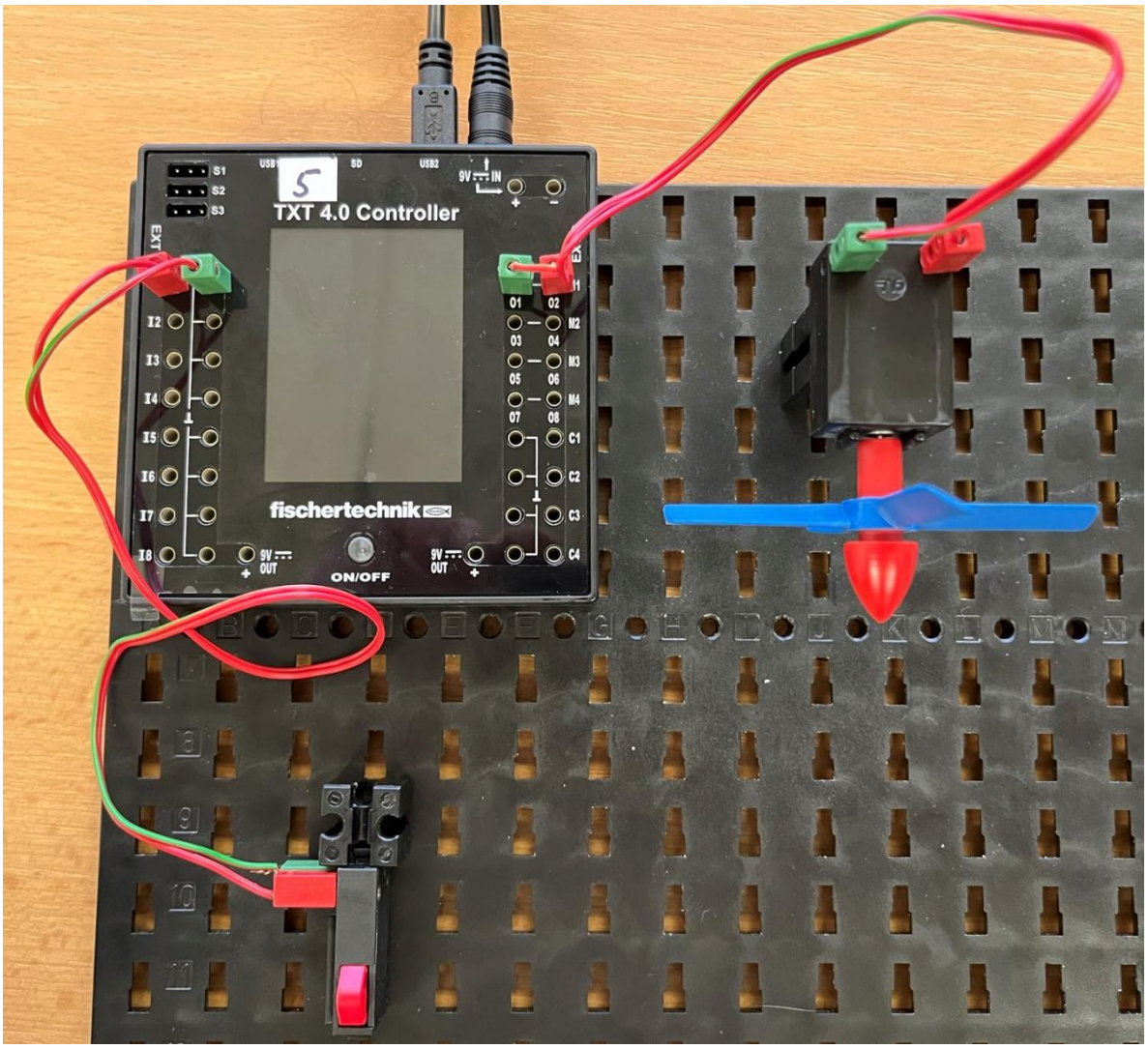
Při zakládání nového projektu (souboru s řídícím kódem) volíme pro naše potřeby možnost „Empty“ a v následujícím kroku volíme název projektu a grafický způsob programování, viz následující obrázek.




*Obrázek 3-2: Založení nového projektu v ROBO PRO Coding*

### 3.2.1 Větráček – zapojení komponent

Zapojení je v tomto případě triviální, na vstup I1 je přiveden spínač, na výstup M1 je přivedený motor větráčku, viz následující obrázek.




*Obrázek 3-3: Větráček – zapojení komponent*

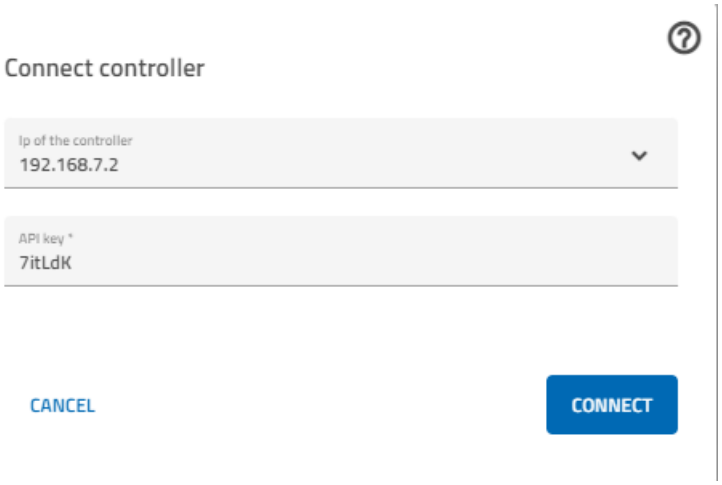
### 3.2.2 Nastavení způsobu komunikace mezi počítačem a řídící jednotkou

Při propojení řídící jednotky s počítačem a jejím zapnutí je vhodné zkontrolovat, že je nastaven správný způsob komunikace:

1. Na dotykové obrazovce TXT 4.0 Controller vybereme volbu **Settings** a v následujícím menu **API key**.


*Obrázek 3-4: TXT 4.0 Controller - Settings*

2. V Robo Pro Coding vybereme **Connect Controller**  . V nově otevřeném okně vybereme volbu USB a vyplníme **API key** zobrazené na TXT 4.0 Controller.




*Obrázek 3-5: Robo Pro Coding - Connect Controller*

3. Pokud se připojení podařilo, zpřístupní se zbývající tlačítka v prostředí ROBO Pro Coding pro ovládání, testování a debugování připojené řídící jednotky.

*Obrázek 3-6: Robo Pro Coding – ovládací prvky záhlaví*

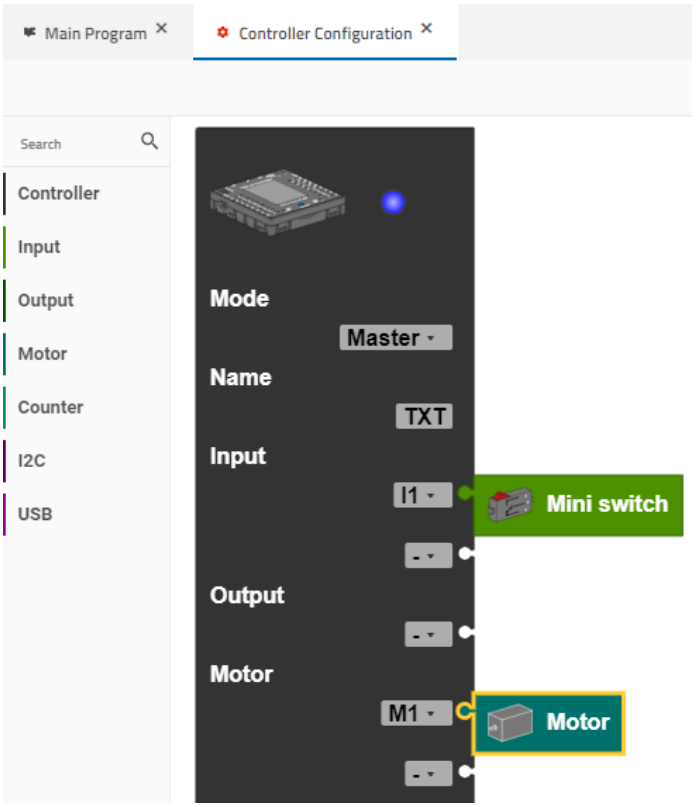
### 3.2.3 Otestování funkčnosti komunikace a komponent

Na tomto případě si ukážeme i jak lze otestovat, že propojení PC – řídící jednotka je funkční a zda komponenty jsou funkční a zapojeny takovým způsobem, aby mohly fungovat. Pro otestování není

potřeba vytvářet jakýkoliv program. Nejdříve však musíme udělat konfiguraci řídící jednotky v Robo Pro Coding.

Chcete-li v programu používat komponenty, jako jsou senzory a akční členy, musíte je připojit k řídicí jednotce, a to nejen fyzicky, ale i softwarově. Konfigurace Controlleru se vždy otevírá automaticky a je přístupná v horní části přímo vedle hlavního programu.

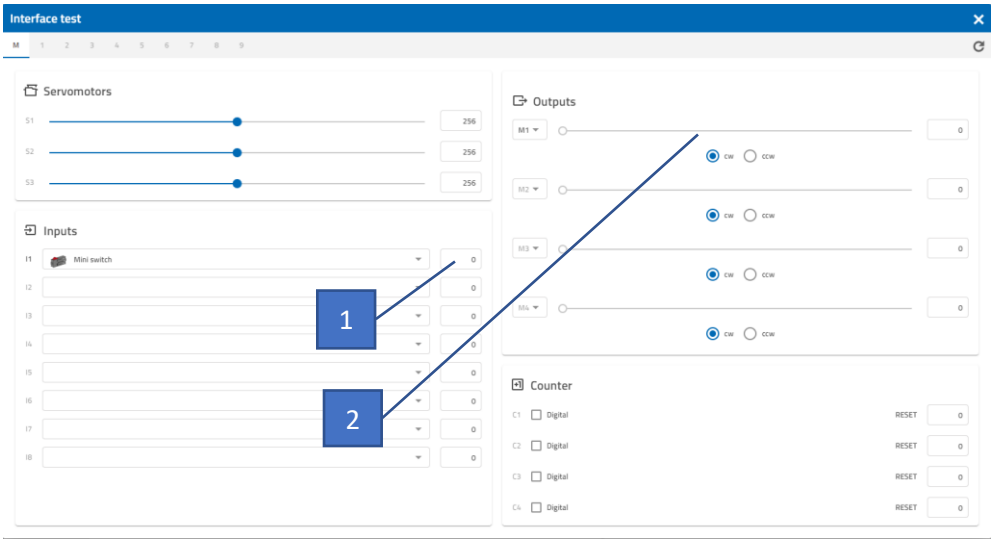
Nyní se v levé části zobrazuje Controller a všechny dostupné komponenty, které lze k němu připojit. Přetáhněte Controller do programovací oblasti. Poté můžete ke Controlleru připojit požadované komponenty přetažením a připojení na požadované vstupy/výstupy.




*Obrázek 3-7: Robo Pro Coding – Konfigurace Controlleru*

Okno interface je přístupné přes ikonu Test interface v horní liště. V otevřeném okně je možné si otestovat jakýkoliv vstup/výstup. Protože my používáme vstup I1 pro tlačítko a výstup M1 pro motor, tak test vypadá následovně:

1. Pokud je vše v pořádku, tak při stisknutí tlačítka na modelu se na vstupu I1 v okně testu interface přepíná hodnota z 0 na 1.
2. Pokud je vše v pořádku, tak u výstupu M1 si můžeme manuálně zapnout motor modelu v libovolném směru a libovolné intenzitě.

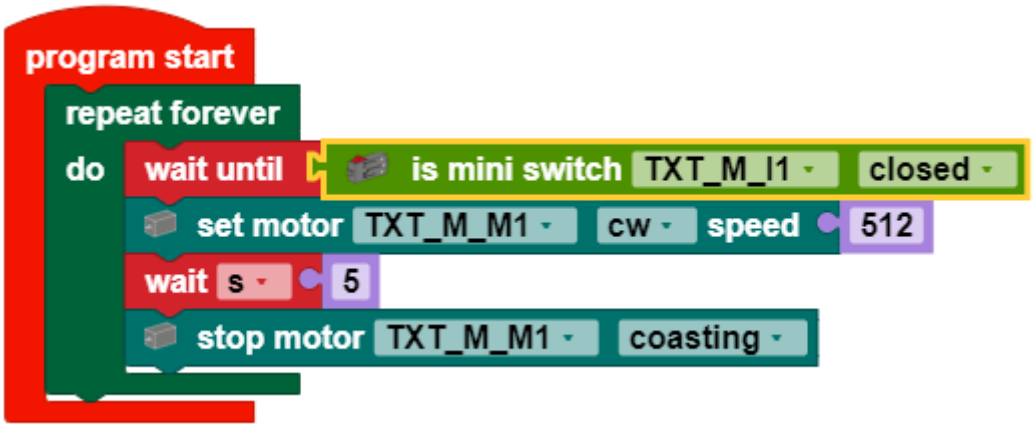



*Obrázek 3-8: Otestování komunikace s modelem a funkčnosti komponent*

Stejným způsobem by se testovaly i další zapojené komponenty a lze tak otestovat vše, co je k řídící jednotce připojeno.

### 3.2.4 Program pro řízení větráčku

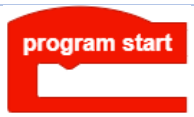
Cílem cvičení je oživit model takovým způsobem, aby po stisknutí tlačítka se větráček roztočil na maximální možnou rychlost, běžel 5 vteřin a po této době se vypl a čekal na další stisknutí tlačítka. Program tedy může vypadat např. následovně:




*Obrázek 3-9: Větráček – řídící program*

Přehled použitých elementů je v následující tabulce.

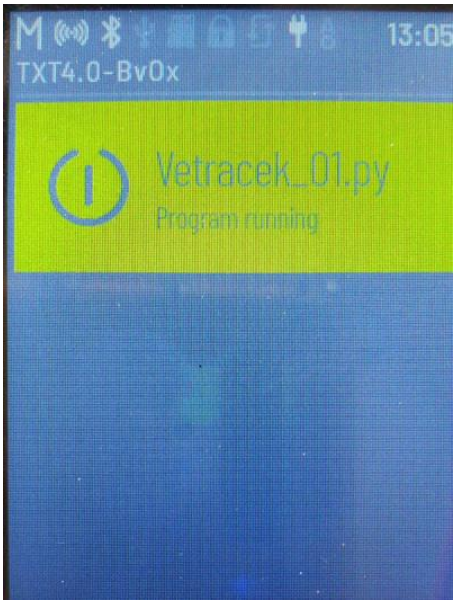
*Tabulka 1: Větráček – přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| start programu |   | 1x | | - | - |
| repeat forever |   | 1x | Processing – Loops | - | - |
| wait until |   | 1x | Processing – Util | - | - |
| is mini switch |   | 1x | Sensors – Input | I1 | - |
| set motor speed |   | 1x | Actuators – Motor | M1 | 1x start, |
| wait |   | 1x | Processing – Util | - | - |
| stop motor |   | 1x | Actuators – Motor | M1 | 1x stop |

### 3.2.5 Zkompilování programu a nahrání do řídící jednotky

Když je hotový program, aktivní připojení k řídící jednotce a připojený model, stačí přes tlačítko (Start program)  nechat program zkompilovat a nahrát do řídící jednotky. Zastavení programu se provede tlačítkem (Stop)  . Toto platí pro online komunikaci. Pokud je chtěno nahrát program do řídící jednotky, aby byl použitelný i po odpojení PC od jednotky, je potřeba použít tlačítko Upload program  .

Když se program zkompiluje a nahraje (v tomto případě v online modu), zobrazí se na displeji řídicí jednotky název programu v zeleném obdélníku s tlačítkem pro zastavení programu a program lze využít, tedy stisknout tlačítko a nechat se chladit, viz video Větráček – základní cvičení.



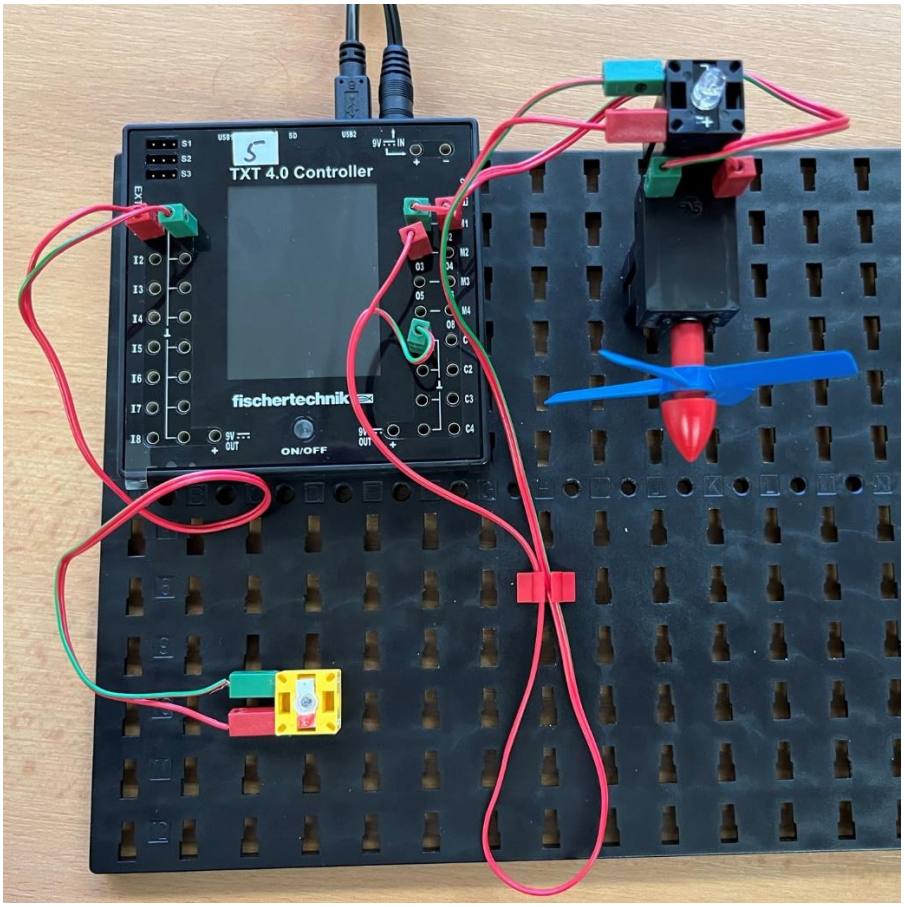

*Obrázek 3-10: Obrazovka řídící jednotky – program je zkompilován a nahrán do řídící jednotky*

## 3.3 Předělání větráčku na model větrné elektrárny

V tomto cvičení předěláme větráček na model větrné elektrárny. Ukážeme si, jak se při tom pracuje s optickými moduly, tedy žárovkou/led a světelným senzorem. V programování si ukážeme jak se program větví. Pro sestavení modelu je potřeba motor, světelný senzor a žárovka. Velkoryse přejdeme, že elektromotor otáčí vrtulí, a ne vrtule elektromotorem – je to jen model.

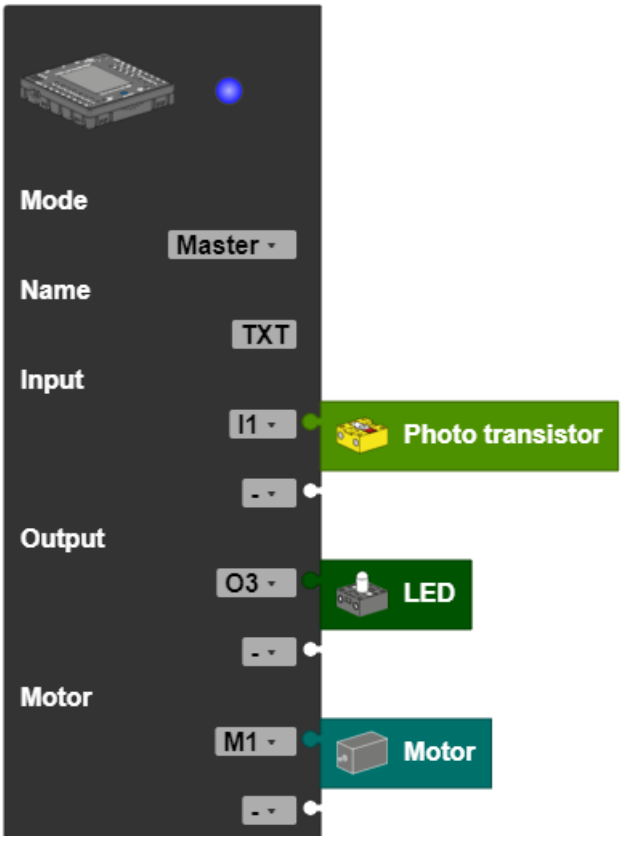
### 3.3.1 Větrná elektrárna – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstup I1 je přivedený světelný senzor, na výstup M1 je přivedený motor s vrtulí a na výstup O3 je přivedené světlo (s červenou krytkou).




*Obrázek 3-11: Větrná elektrárna – zapojení modelu*

Konfigurace kontroléru:



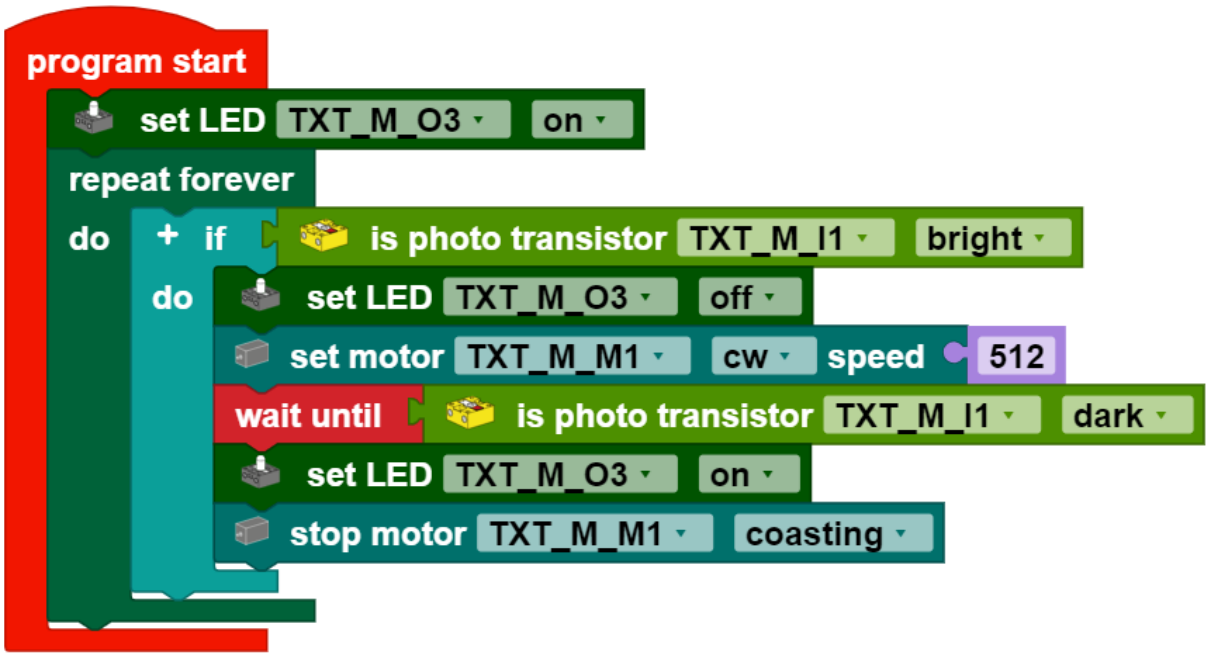

*Obrázek 3-12: Robo Pro Coding – Konfigurace Controlleru – větrná elektrárna*

### 3.3.2 Větrná elektrárna – řídící program

Jak chceme, aby se větrná elektrárna chovala…

1. V noci bude svítit červené světlo, aby nám do elektrárny nenarážela letadla. Vrtule bude v klidu, nechceme budit blízko bydlící občany.
2. Ve dne budeme šetřit světlo, letadla uvidí větrnou elektrárnu i bez něj, a vrtule se může točit.
3. Toto chování chceme mít uzavřené v cyklu, aby se vše spouštělo a vypínalo automaticky a my se o větrnou elektrárnu nemuseli denně starat.

Možnost, jak může vypadat program je na následujícím obrázku. Běh elektrárny je na videu Větrná elektrárna. Ze stejných součástek by šel vytvořit např. i sušák rukou, kdy by světelný senzor vyhodnocoval, jestli vidí světlo ze žárovky, nebo nevidí – tedy jsou mezi ním a žárovkou vloženy ruce a má se spustit jejich sušení.

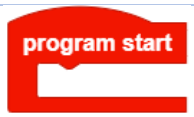



```
1   import time
2
3   from fischertechnik.controller.Motor import Motor
4   from lib.controller import *
5
6
7   TXT_M_O3_led.set_brightness(512)
8   while True:
9       if TXT_M_I1_photo_transistor.is_bright():
10          TXT_M_O3_led.set_brightness(0)
11          TXT_M_M1_motor.set_speed(int(512), Motor.CW)
12          TXT_M_M1_motor.start()
13          while True:
14              if (TXT_M_I1_photo_transistor.is_dark()):
15                  break
16              time.sleep(0.010)
17          TXT_M_O3_led.set_brightness(512)
18          TXT_M_M1_motor.coast()
```

*Obrázek 3-13: Větrná elektrárna – řídící program*

Přehled použitých elementů je v následující tabulce.

*Tabulka 2: Větrná elektrárna – přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| start programu |  | 1x | | - | - |
| repeat forever |  | 1x | Processing – Loops | - | - |
| If – else |  | 1x | Processing – Logic | - | - |
| is photo transistor |  | 2x | Sensors – Input | I1 | - |
| set LED brightness |  | 2x | Actuators – Output | O3 | 1x on, 1x off |
| wait until |  | 1x | Processing – Util | - | - |
| set motor speed |  | 1x | Actuators – Motor | M1 | 1x start, |
| stop motor |  | 1x | Actuators – Motor | M1 | 1x stop |

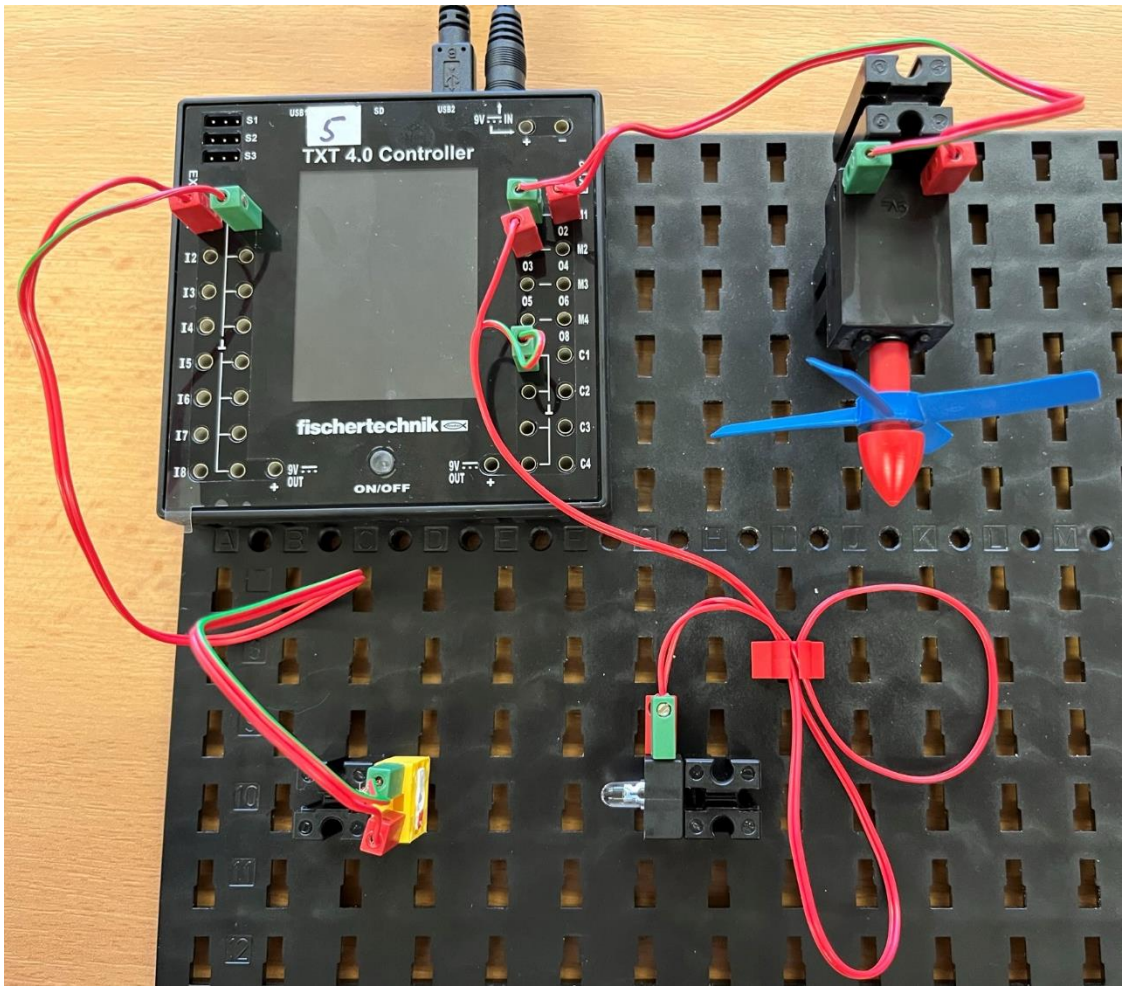
## 3.4 Vysoušeč rukou – princip světelné brány

V tomto cvičení využijeme části předchozích cvičení. Úkolem je vytvořit vysoušeč rukou, tedy zařízení, kde je světelná brána rozhodující o spuštění/nespuštění ventilátoru. Na cvičení se v tomto případě pouze lehce modifikuje model využitý v předchozím cvičení.

Světelná brána je využití komponenty emitující světlo (v tomto případě např. žárovka, nebo LED) na jedné straně a fototranzistor, který vyhodnocuje, zda na něj emitované světlo ze světelné komponenty dopadá, nebo ne, na straně druhé. Pokud světlo na fototranzistor dopadá, je větrák v klidu. Pokud ne, tedy mezi světelnou komponentu a fototranzistor jsou vloženy ruce, větrák se na určitou dobu sepne.

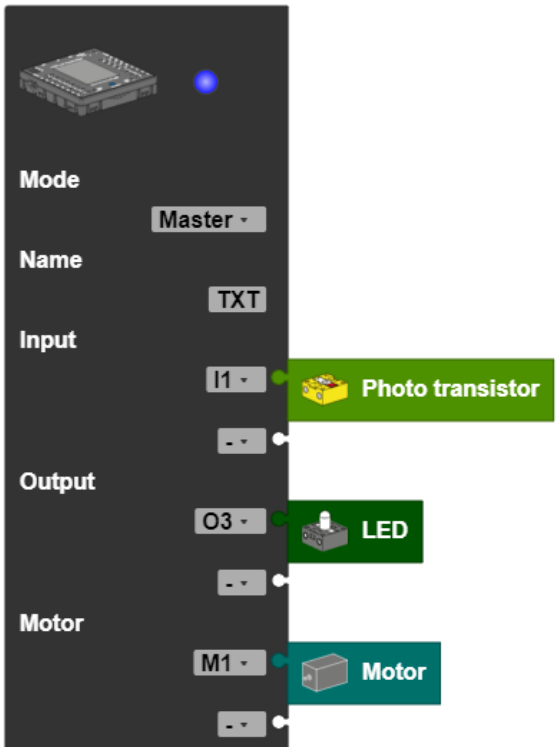
### 3.4.1 Vysoušeč rukou – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstup I1 je přivedený fototranzistor (světelný senzor), na výstup M1 je přivedený motor s vrtulí a na výstup O3/O4 je přivedené světlo, v tomto případě LED.




*Obrázek 3-14: Vysoušeč rukou – zapojení komponent*

Konfigurace kontroléru:

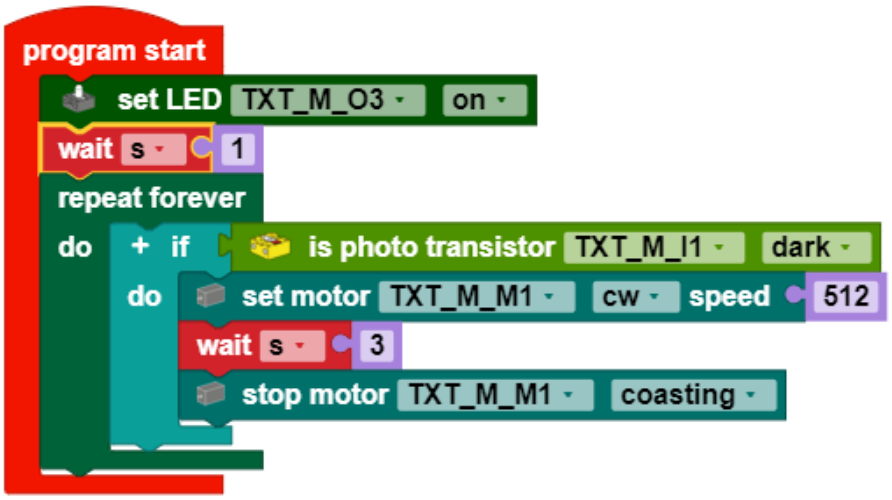



*Obrázek 3-15: Robo Pro Coding – Konfigurace Controlleru – vysoušeč rukou*

### 3.4.2 Vysoušeč rukou – řídící program

Jak chceme, aby se vysoušeč choval…

1. Pokud není mezi LED komponentou a fototranzistorem překážka, tak je vysoušeč v klidu
2. Pokud se mezi LED komponentu a fototranzistor vloží překážka (ruce), vysoušeč se na 3 sekundy spustí a bude vysoušet ruce.
3. Program bude v cyklu, tedy kdykoliv se vloží překážka mezi LED a fototranzistor, tak se vysoušeč opětovně zapne.

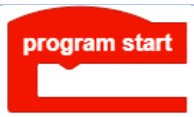



```
1   import time
2
3   from fischertechnik.controller.Motor import Motor
4   from lib.controller import *
5
6
7   TXT_M_O3_led.set_brightness(512)
8   time.sleep(1)
9   while True:
10      if TXT_M_I1_photo_transistor.is_dark():
11          TXT_M_M1_motor.set_speed(int(512), Motor.CW)
12          TXT_M_M1_motor.start()
13          time.sleep(3)
14          TXT_M_M1_motor.coast()
15
```

*Obrázek 3-16: Vysoušeč rukou – řídící program se světelnou bránou*

Přehled použitých elementů je v následující tabulce.

*Tabulka 3: Vysoušeč rukou – přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| start programu |   | 1x | | - | - |
| repeat forever |   | 1x | Processing – Loops | - | - |
| If do |   | 1x | Processing – Logic | - | - |
| is photo transistor |   | 1x | Sensors – Input | I1 | - |
| set LED brightness |   | 1x | Actuators – Output | O3 | 1x on |
| wait |   | 1x | Processing – Util | - | - |
| set motor speed |   | 1x | Actuators – Motor | M1 | 1x start, |
| stop motor |   | 1x | Actuators – Motor | M1 | 1x stop |

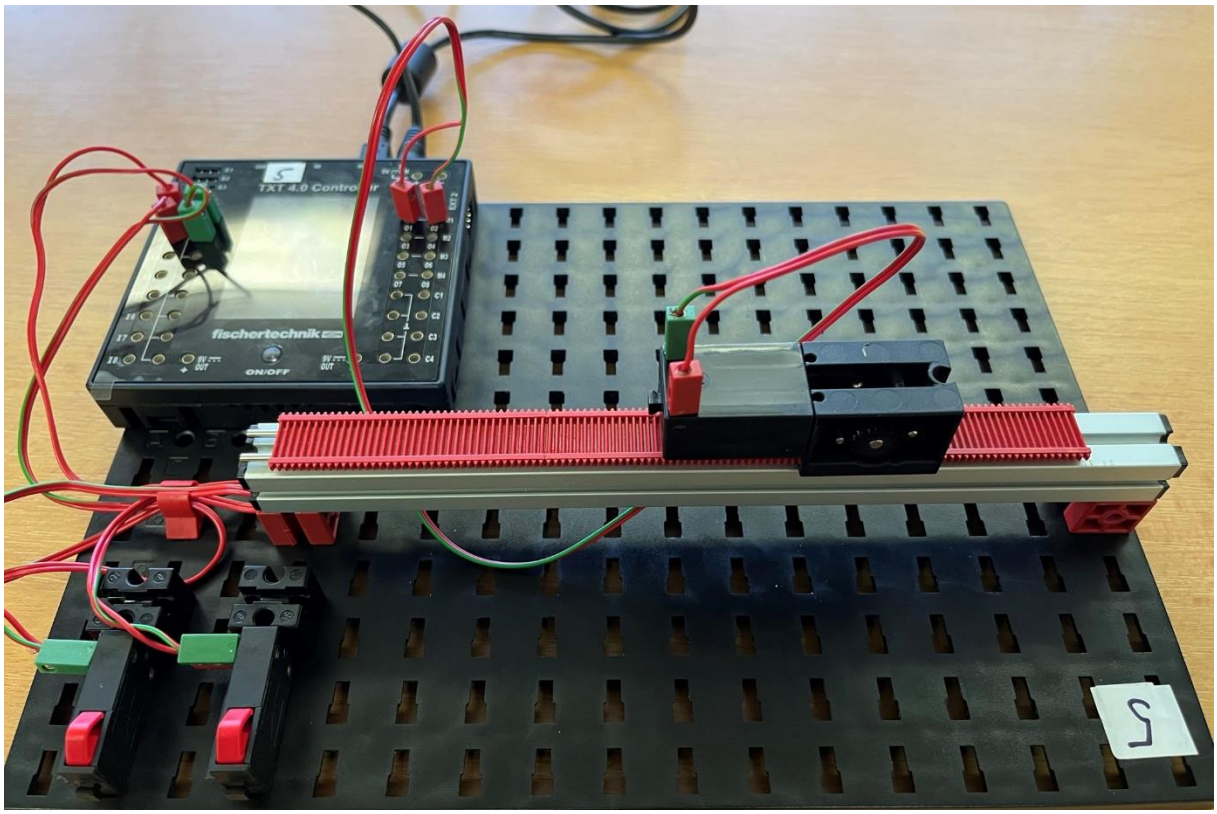
Ukázka běhu modelu viz video Vysoušeč rukou.

## 3.5 Manuálně ovládaný pojezd

V tomto cvičení si zopakujeme vyhodnocování podmínek (větvení programu). Cílem je mít manuálně ovládaný pojezd. Model se skládá ze dvou tlačítek a motoru. Motor je spojen s převodovkou, která přenáší otáčky na výstupu elektromotoru na ozubené kolečko, které je v kontaktu s ozubeným pásem pevně spojeným s nosníkem.

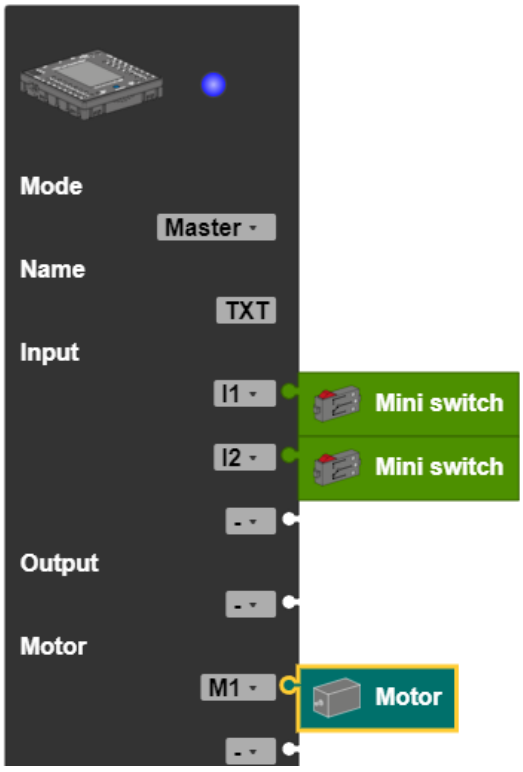
### 3.5.1 Manuálně ovládaný pojezd – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstupy I1 a I2 jsou přivedeny spínače, na výstup M1 je přivedený motor zajišťující posuv pojezdu.




*Obrázek 3-17: Manuálně ovládaný pojezd – zapojení modelu*

Konfigurace kontroléru:



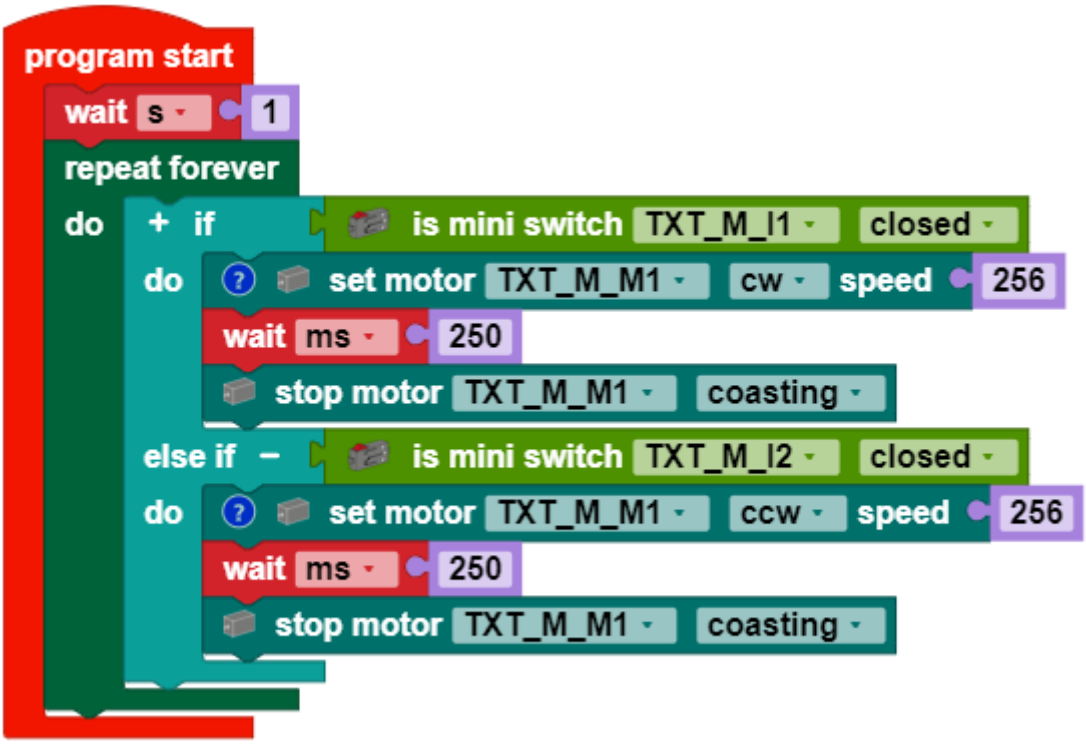

*Obrázek 3-18: Robo Pro Coding – Konfigurace Controlleru – manuálně ovládaný pojezd*

### 3.5.2 Manuálně ovládaný pojezd – ovládací program

Jak chceme, aby se pojezd choval…

1. Při stisknutí levého tlačítka se pojezd bude pohybovat doleva.
2. Při stisknutí pravého tlačítka se pojezd bude pohybovat doprava.
3. Při držení tlačítka se pojezd bude pohybovat bez přerušení po dobu držení tlačítka.
4. Program bude v cyklu, tedy kdykoliv stisknu jakékoliv tlačítko, tak se pojezd bude pohybovat odpovídajícím směrem.

Zde je dobré upozornit na drobnou „vychytralost“. Pokud se má pojezd pohybovat po celou dobu držení tlačítka, pomůžeme si nastavením malé hodnoty na dobu běhu motoru. Čím menší doba běhu motoru bude, tím rychleji bude pojezd reagovat na stisknutí/nestisknutí tlačítka. Možnost, jak může vypadat program je na následujícím obrázku. Funkční pojezd je na videu Manuálně ovládaný pojezd.

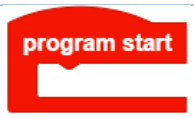



```
1   import time
2
3   from fischertechnik.controller.Motor import Motor
4   from lib.controller import *
5
6
7   time.sleep(1)
8   while True:
9       if TXT_M_I1_mini_switch.is_closed():
10          # tlačítko I1 stisknuto, pojezd pojede doleva
11          TXT_M_M1_motor.set_speed(int(256), Motor.CW)
12          TXT_M_M1_motor.start()
13          time.sleep(0.25)
14          TXT_M_M1_motor.coast()
15      elif TXT_M_I2_mini_switch.is_closed():
16          # tlačítko I2 stisknuto, pojezd pojede doprava
17          TXT_M_M1_motor.set_speed(int(256), Motor.CCW)
18          TXT_M_M1_motor.start()
19          time.sleep(0.25)
20          TXT_M_M1_motor.coast()
21
```

*Obrázek 3-19: Manuálně ovládaný pojezd – řídící program*

Přehled použitých elementů je v následující tabulce.

*Tabulka 4: Manuálně ovládaný pojezd – přehled použitých elementů*

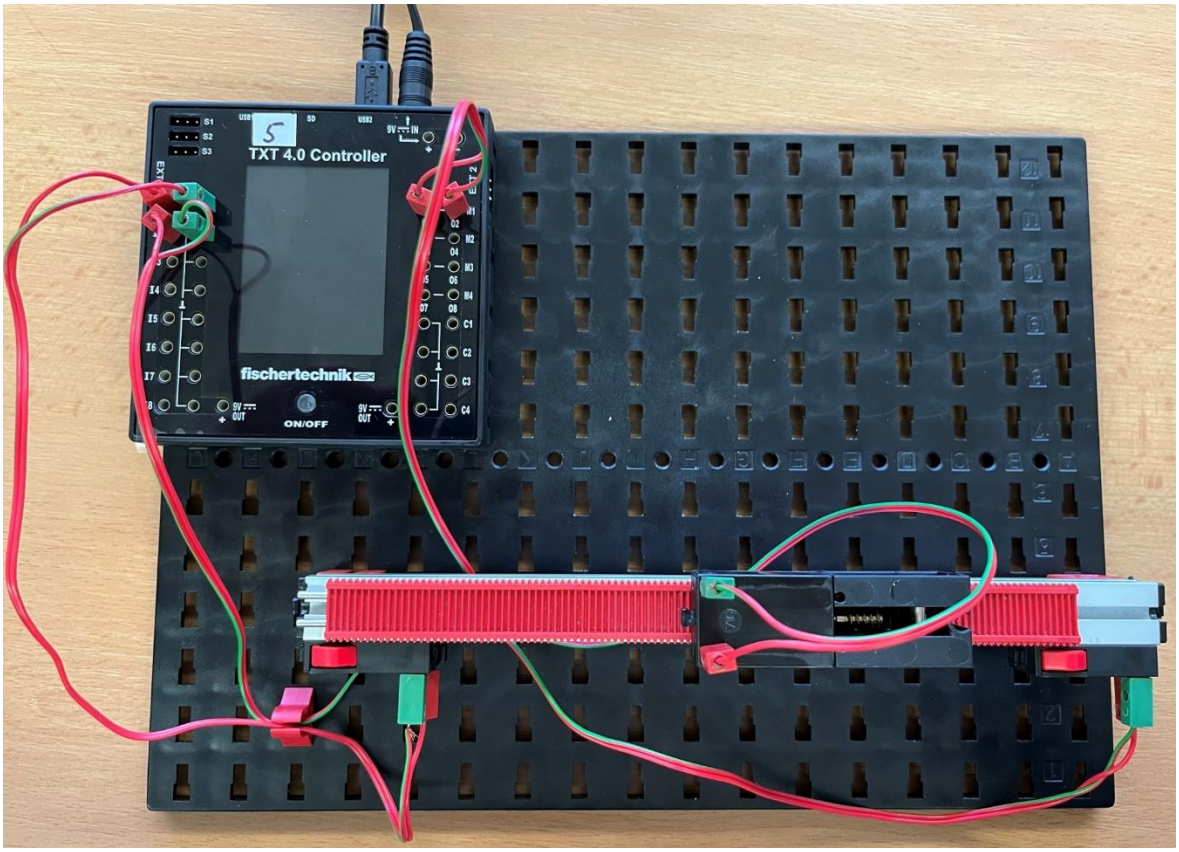
| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| start programu |   | 1x | | - | - |
| repeat forever |   | 1x | Processing – Loops | - | - |
| If do |   | 1x | Processing – Logic | - | Pomocí symbolu + se přidávají podmínky |
| is mini switch |   | 2x | Sensors – Input | I1, I2 | - |
| wait |   | 2x | Processing – Util | - | - |
| set motor speed |   | 2x | Actuators – Motor | M1 | Nastavují se různé směry otáčení |
| stop motor |   | 2x | Actuators – Motor | M1 | 2x stop |

## 3.6 Pojezd mezi pevně danými body

V tomto cvičení jednoduše modifikujeme předchozí model. Zde se naučíme vytvářet pro program a model ovládací panel, vytvářet a využívat podprogramy zde označované jako funkce[2], pracovat s elementy typu proměnná, Panel displayem, rozhodováním na základě podmínky a zopakujeme vyhodnocování podmínek (větvení programu), využijeme spínače jako skutečné koncové spínače. Cílem je mít pojezd, který po stisknutí tlačítka na ovládacím panelu (softwarové tlačítko) přesune pojezd na požadovanou pozici. Model se skládá ze dvou tlačítek a motoru. Motor je spojen s převodovkou, která přenáší otáčky na výstupu elektromotoru na ozubené kolečko, které je v kontaktu s ozubeným pásem pevně spojeným s nosníkem.

### 3.6.1 Pojezd mezi pevně danými body – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstupy I1 a I2 jsou přivedeny koncové spínače (I1 spínač vlevo, I2 spínač vpravo), na výstup M1 je přivedený motor zajišťující posuv pojezdu.

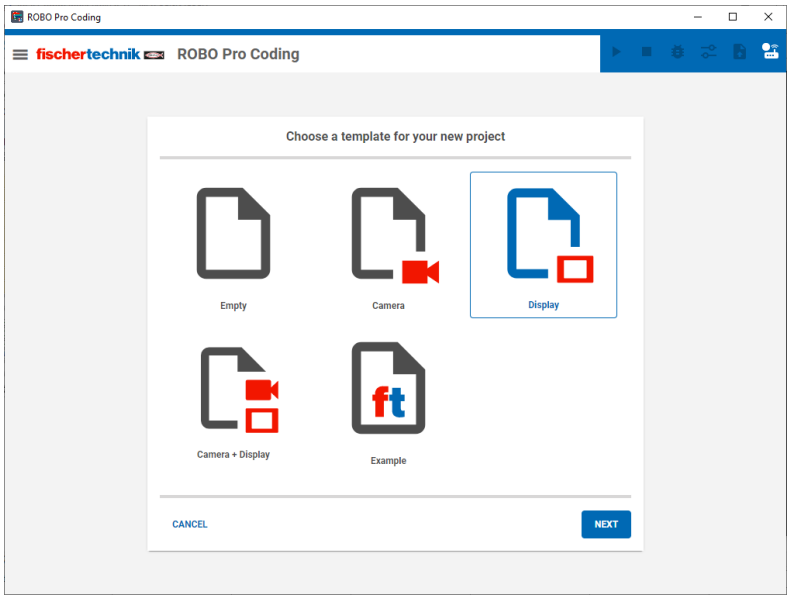



*Obrázek 3-20: Pojezd mezi pevně danými body – zapojení komponent*

**Založení nového projektu**

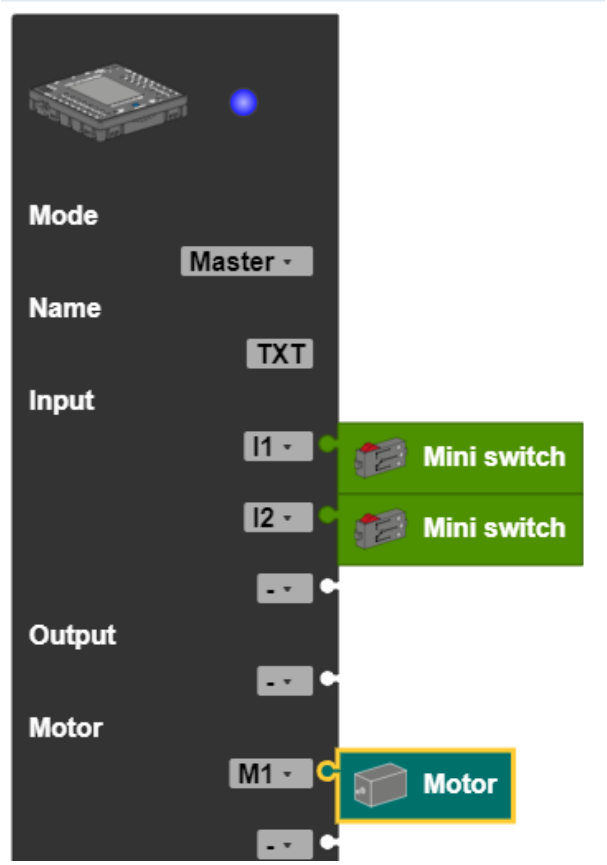
V projektu budeme potřebovat definovat ovládací prvky na display řídící jednotky. Z tohoto důvodu již při zakládání projektu vybereme volbu **Display**.

[2] „Když neodskočíš z cyklu programu, nikdy nebudeš v podprogramu.“ Kryton, Červený trpaslík, Spravedlnost




*Obrázek 3-21: Založení nového projektu*

Konfigurace kontroléru:




*Obrázek 3-22: Robo Pro Coding – Konfigurace Controlleru – Pojezd mezi pevně danými body*

### 3.6.2 Pojezd mezi pevně danými body – řídící program

Jak chceme, aby se pojezd choval…

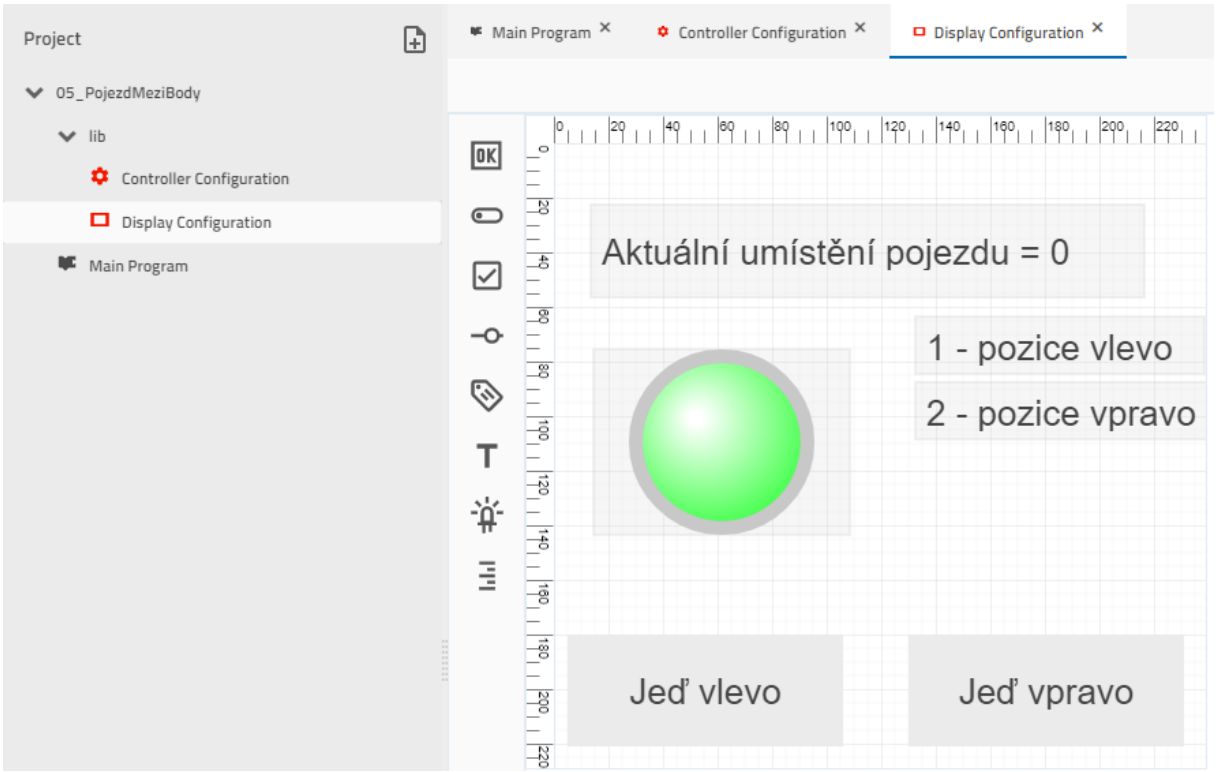
1. K ovládání chceme používat ovládací panel.
2. Při stisknutí vhodného softwarového tlačítka na ovládacím panelu se pojezd přesune ke koncovému spínači doleva.
3. Při stisknutí vhodného softwarového tlačítka na ovládacím panelu se pojezd přesune ke koncovému spínači doprava.
4. Program bude v cyklu, tedy kdykoliv stisknu jakékoliv softwarové tlačítko, tak se pojezd bude pohybovat odpovídajícím směrem a přesune se na požadované místo.
5. Na ovládacím panelu bude informace o tom, v které z koncových poloh se zrovna pojezd nachází.
6. Na ovládacím panelu bude informace o tom, zda pojezd se pohybuje pomocí stavového indikátoru.

V tomto případě už je program trochu komplikovanější. Bude potřeba vytvořit 2 funkce. Jeden pro zastavení vozíku, druhý pro hlídání lokace. Dále budeme muset nadefinovat událost pro každé tlačítko, které zajistí spuštění pojezdu ve správném směru. Hlavní program bude využívat obě funkce a bude mít naprogramovaný ovládací panel na řídící jednotce. Možnost, jak může program a funkce vypadat jsou na následujících obrázcích. Funkční pojezd je na videu Pojezd mezi pevně danými body.

#### *3.6.2.1 Ovládací panel*

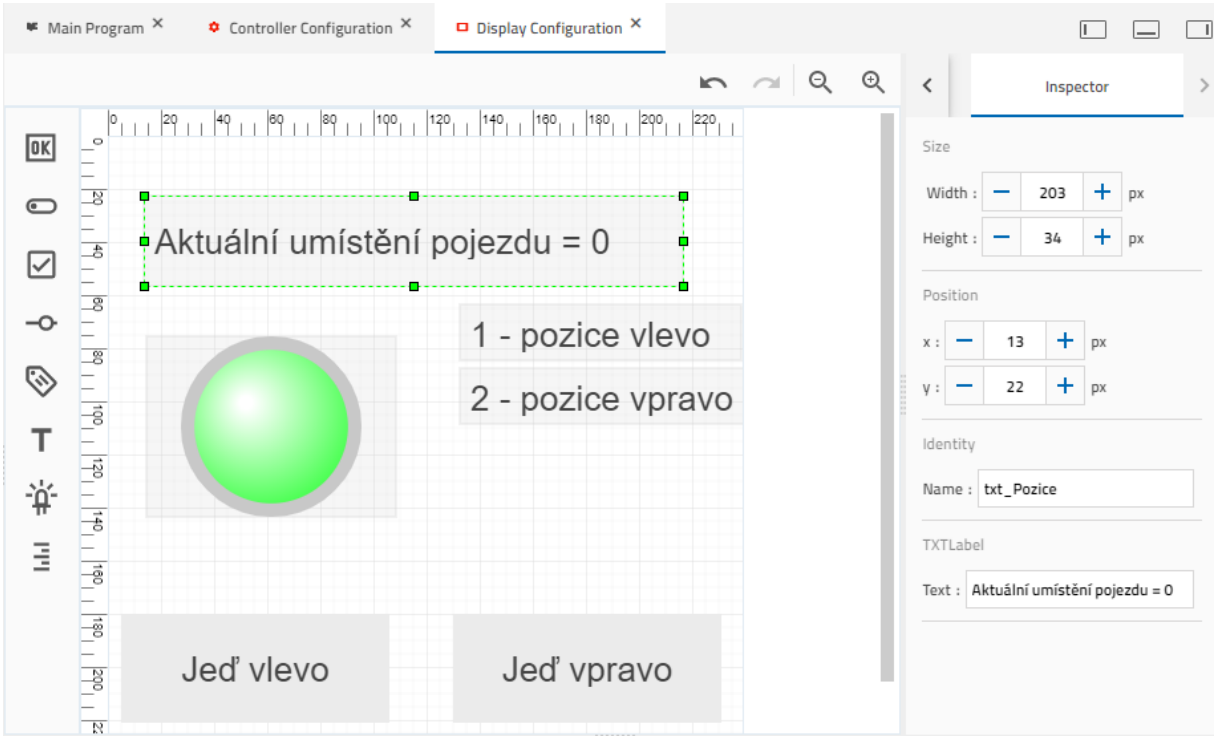
V projektu je potřeba vytvořit GUI ovládacího panelu. Klikneme na **Display Configuration** a tím se nám zobrazí možnost definovat jednotlivé ovládací prvky následně zobrazované na řídící jednotce.

Přehled použitých elementů pro GUI je součástí přehledu použitých elementů pro hlavní program.



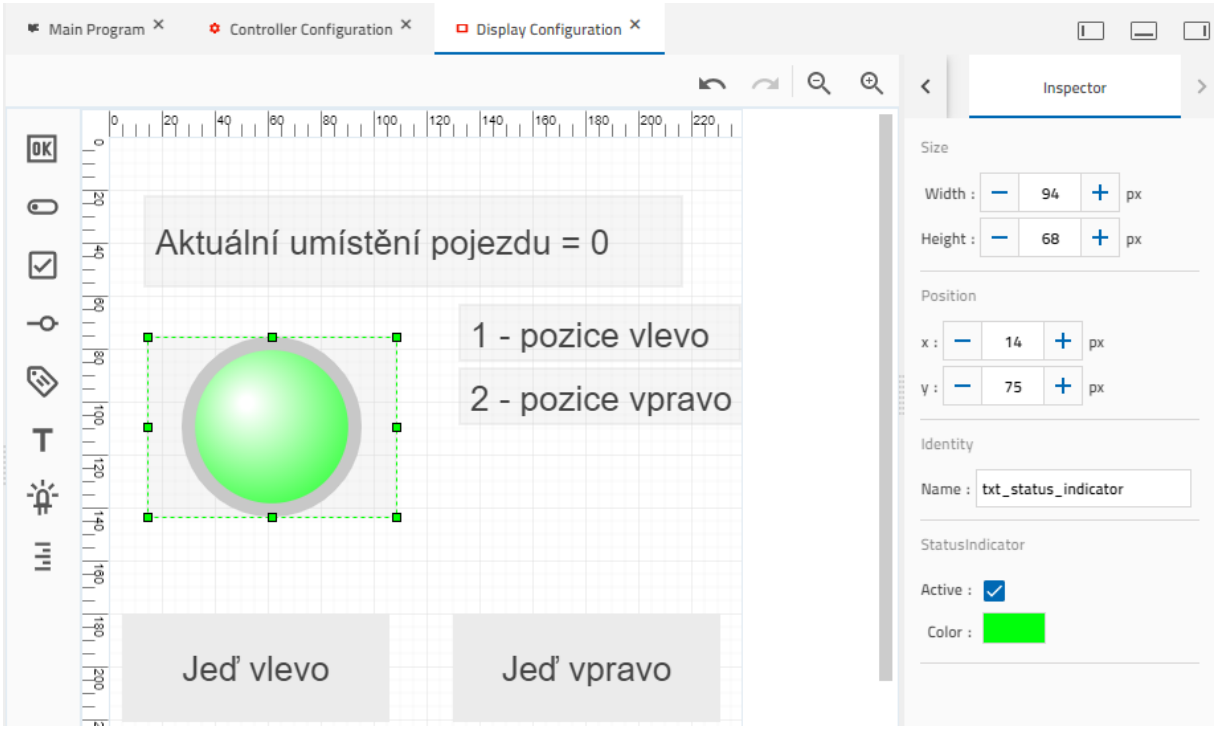

*Obrázek 3-23: Ovládací panel – takto může vypadat*

**TxtLabel** – nastavíme např. následovně s rozumným popiskem (např. „Aktuální umístění pojezdu = 0“).



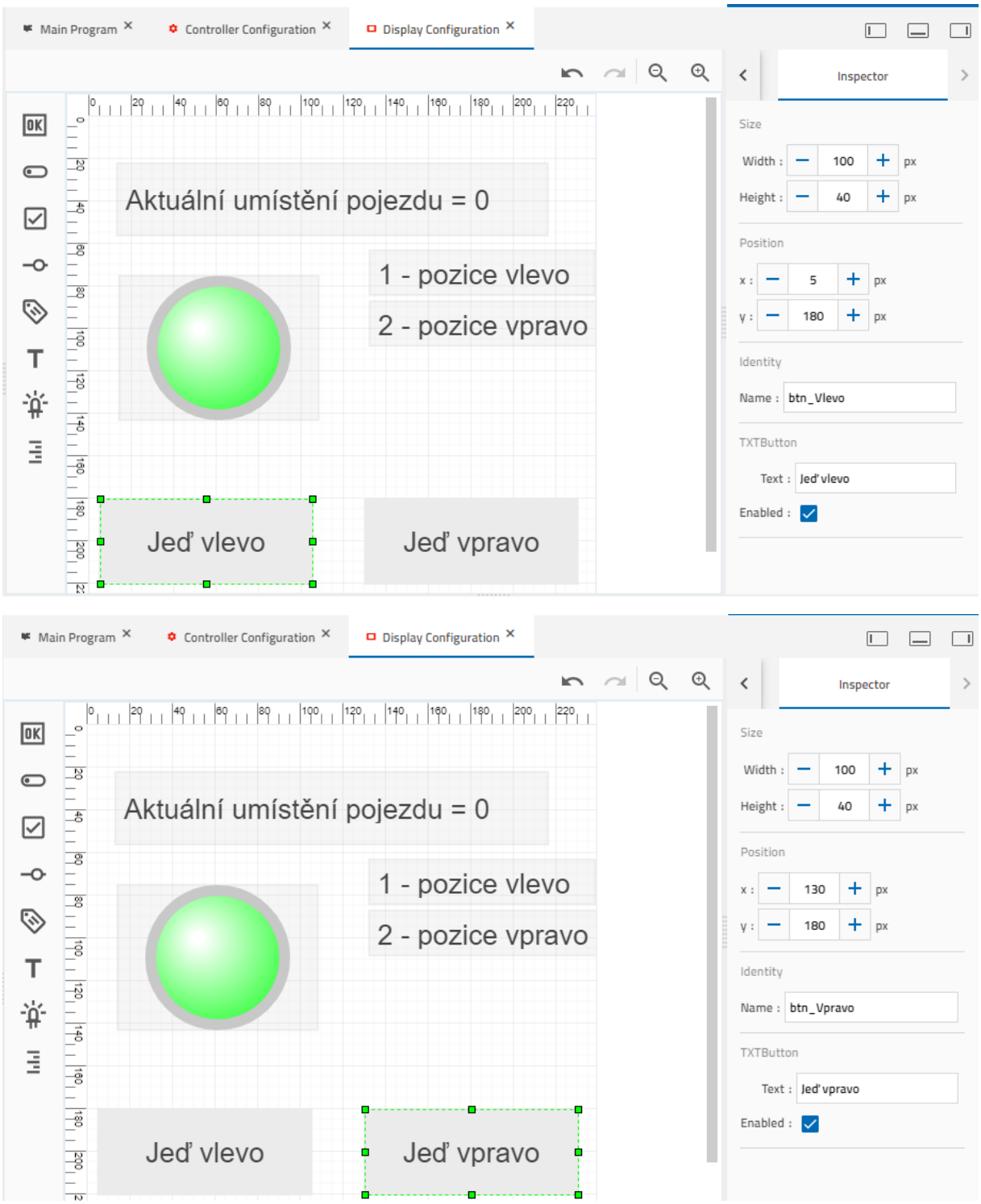

*Obrázek 3-24: Nastavení TxtLabel*

Indikátor stavu **StatusIndicator** nastavíme následovně.




*Obrázek 3-25: Nastavení StatusIndicator*

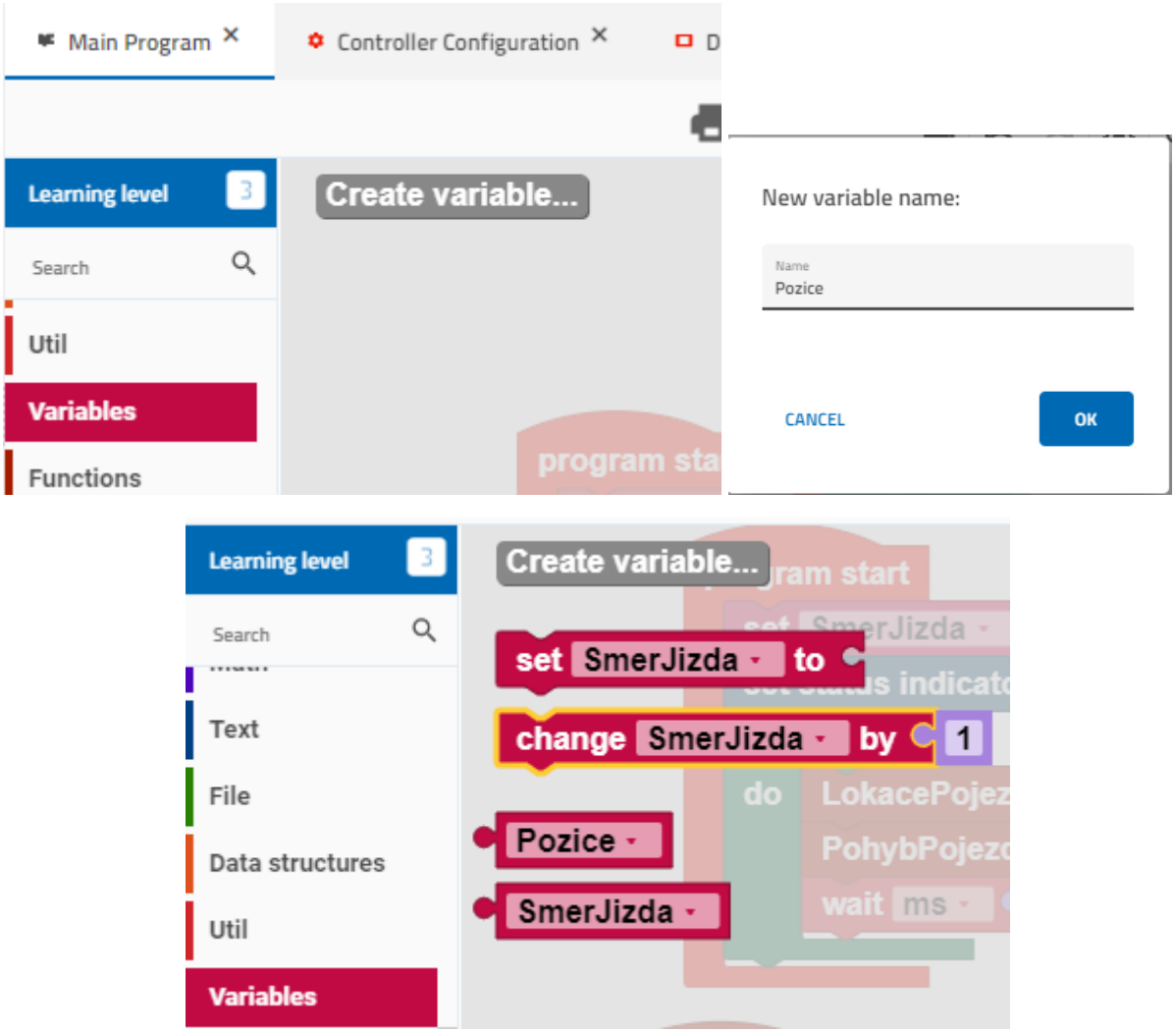
Tlačítka **TxtButton** nastavíme následovně.




*Obrázek 3-26: Nastavení TxtButton*

### *3.6.2.2 Deklarace proměnných*

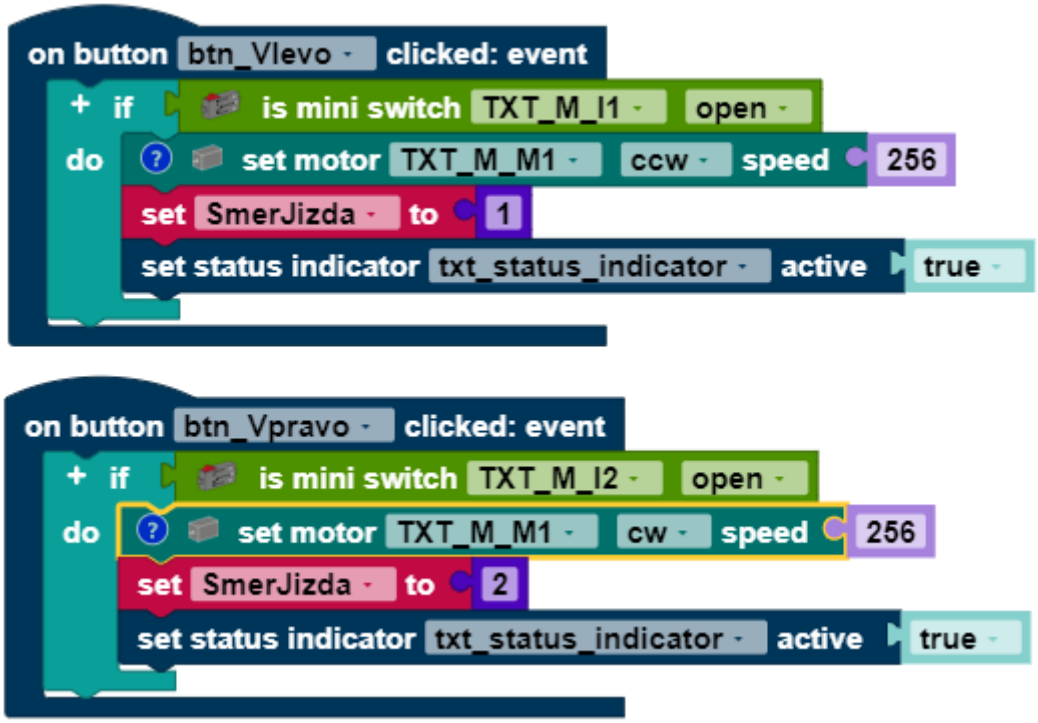
Deklarujeme si proměnou **Pozice**. Pomocí volby **Create variable…** v sekci **Variable** si deklarujeme novou proměnou **Pozice**. Stejným způsobem deklarujeme pomocnou proměnou **SmerJizda**, kam si budeme ukládat informaci, na jakou stranu jede pojezd.




*Obrázek 3-27: Deklarace proměnné Pozice*

### 3.6.2.3 Události na tlačítka

Při stisknutí tlačítka na displeji se vyvolá událost. Nyní naprogramujeme události pro obě dvě tlačítka.




```
def on_btn_Vlevo_clicked(event):
    global SmerJizda, Pozice
    if TXT_M_I1_mini_switch.is_open():
        # tlačítko I2 stisknuto, pojezd pojede doprava
        TXT_M_M1_motor.set_speed(int(256), Motor.CCW)
        TXT_M_M1_motor.start()
        SmerJizda = 1
        display.set_attr("txt_status_indicator.active", str(True).lower())

def on_btn_Vpravo_clicked(event):
    global SmerJizda, Pozice
    if TXT_M_I2_mini_switch.is_open():
        # tlačítko I1 stisknuto, pojezd pojede doleva
        TXT_M_M1_motor.set_speed(int(256), Motor.CW)
        TXT_M_M1_motor.start()
        SmerJizda = 2
        display.set_attr("txt_status_indicator.active", str(True).lower())
```

*Obrázek 3-28: Pojezd mezi pevně danými body – události na tlačítka*

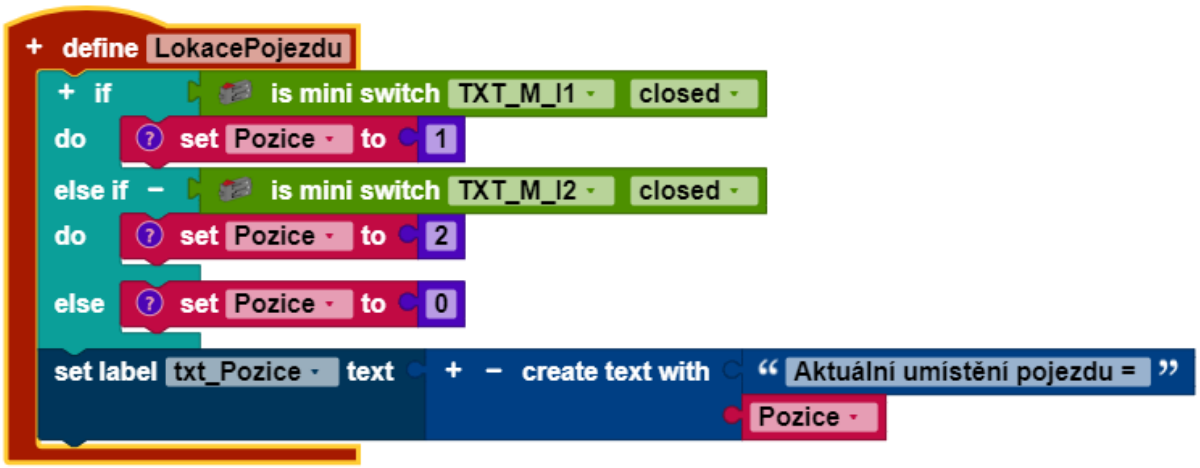
Přehled použitých elementů je v následující tabulce.

*Tabulka 5: Pojezd mezi pevně danými body – události na tlačítka – přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| událost na tlačítko |  | 2x | Actuators – Display | - | - |
| if – do |  | 2x | Processing – Logic | - | - |
| is mini switch |  | 2x | Sensors – Input | I1, I2 | - |
| set motor speed |  | 2x | Actuators – Motor | M1 | Nastavena poloviční rychlost |
| set to |  | 2x | Processing – Variables | - | - |
| a number |  | 2x | Processing – Math | - | - |
| set status indicator |  | 2x | Actuators – Display | - | - |

### 3.6.2.4 Funkce lokace pojezdu

Funkce lokace pojezdu slouží pro zjištění umístění pojezdu a vypsání umístění do GUI.

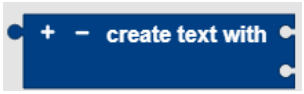



```
def LokacePojezdu():
    global SmerJizda, Pozice
    if TXT_M_I1_mini_switch.is_closed():
        # Pojezd se nachází vlevo
        Pozice = 1
    elif TXT_M_I2_mini_switch.is_closed():
        # Pojezd se nachází vpravo
        Pozice = 2
    else:
        # Pojezd se nachází mezi
        Pozice = 0
    display.set_attr("txt_Pozice.text", str('Aktuální umístění pojezdu = ' + str(Pozice)))
```

*Obrázek 3-29: Funkce lokace pojezdu*

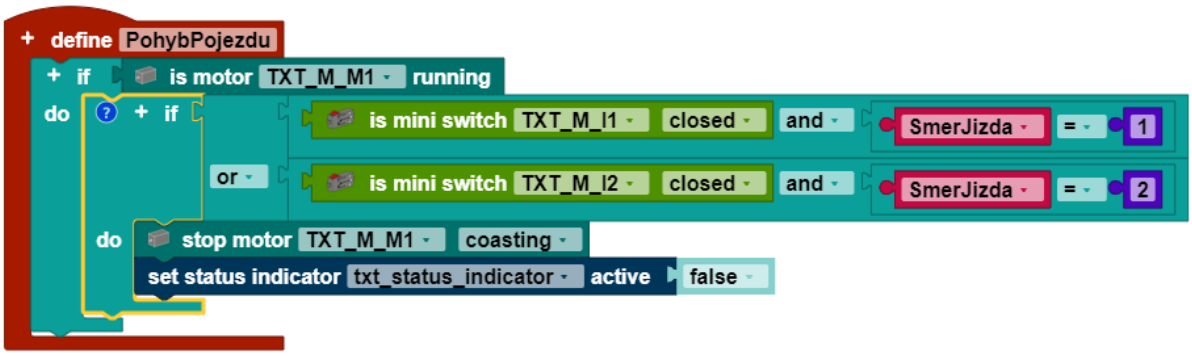
Přehled použitých elementů je v následující tabulce.

*Tabulka 6: Pojezd mezi pevně danými body – Funkce lokace pojezdu – přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| funkce |  | 1x | Processing – Functions | - | - |
| if - else |  | | Processing – Logic | - | - |
| is mini switch |  | 2x | Sensors – Input | I1, I2 | - |
| set to |  | 3x | Processing – Variables | - | - |
| a number |  | 3x | Processing – Math | - | - |
| set label text |  | 1x | Actuators – Display | - | - |
| create text with |  | 1x | Processing – Text | - | - |
| text |  | 1x | Processing – Text | - | - |
| variable |  | 1x | Processing – Variables | - | - |

### *3.6.2.5 Funkce zastavení pohybu pojezdu v cílové destinaci*

Funkce zastavení pohybu pojezdu zastaví pojezd v cílové destinaci, které pozná přes stisknutí adekvátního koncového spínače.




```
def PohybPojezdu():
    global SmerJizda, Pozice
    if TXT_M_M1_motor.is_running():
        # Test, zda pojezd dosáhl konce drah ve směru své jízdy
        if (TXT_M_I1_mini_switch.is_closed()) and SmerJizda == 1 or (TXT_M_I2_mini_switch.is_closed()) and SmerJizda == 2:
            TXT_M_M1_motor.coast()
            display.set_attr("txt_status_indicator.active", str(False).lower())
```

*Obrázek 3-30: Funkce pro pohyb pojezdu*

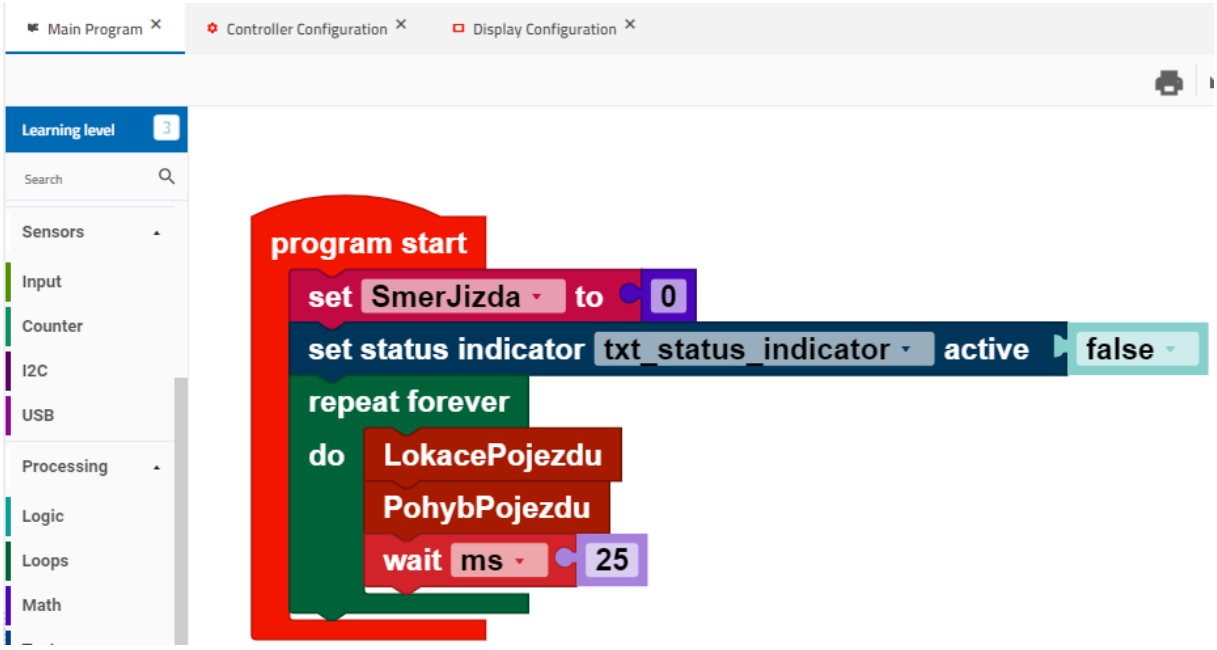
Přehled použitých elementů je v následující tabulce.

*Tabulka 7: Pojezd mezi pevně danými body - funkce pohybu pojezdu - přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| Subprogram |   | 1x | Processing – Functions | - | - |
| If - do |   | 2x | Processing – Logic | - | - |
| Logical operators |   | 3x | Processing – Logic | - | 1x OR, 2x AND |
| Comparative operators |   | 2x | Processing – Logic | - | - |
| is motor running |   | 1x | Actuators – Motor | M1 | - |
| is mini switch |   | 2x | Sensors – Input | I1, I2 | - |
| a number |   | 3x | Processing – Math | - | - |
| variable |   | 2x | Processing – Variables | - | - |
| stop motor |   | 1x | Actuators – Motor | M1 | - |
| set status indicator |   | 1x | Actuators – Display | - | - |

### *3.6.2.6 Vytvoření těla hlavního programu*

V tuto chvíli máme vytvořený ovládací panel pro ovládání události na tlačítka a 2 funkce. Poslední krok je funkce použít v těle hlavního programu, viz obr.:

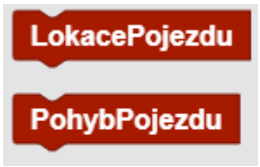



```
1   import time
2
3   from fischertechnik.controller.Motor import Motor
4   from lib.controller import *
5   from lib.display import *
6
7
8   SmerJizda = None
9   Pozice = None
10
11
12
13  def on_btn_Vlevo_clicked(event): ···
21
22  def on_btn_Vpravo_clicked(event): ···
30
31
32  def LokacePojezdu(): ···
44
45
46  def PohybPojezdu(): ···
53
54
55  display.button_clicked("btn_Vlevo", on_btn_Vlevo_clicked)
56  display.button_clicked("btn_Vpravo", on_btn_Vpravo_clicked)
57
58
59  SmerJizda = 0
60  display.set_attr("txt_status_indicator.active", str(False).lower())
61  while True:
62      LokacePojezdu()
63      PohybPojezdu()
64      time.sleep(0.025)
65
```

*Obrázek 3-31: Hlavní program s využitím funkcí*

Přehled použitých elementů je v následující tabulce.

*Tabulka 8: Pojezd mezi pevně danými body - program - přehled použitých elementů*

| Použitý element | Základní zobrazení elementu | Počet | Umístění elementu v nabídce | připojení na vstup (Input), výstup (Output/Motor) | Poznámka |
|---|---|---|---|---|---|
| start programu |   | 1x | | - | - |
| set to |   | 2x | Processing – Variables | - | - |
| a number |   | 2x | Processing – Math | - | - |
| set status indicator |   | 1x | Actuators – Display | - | - |
| repeat forever |   | 1x | Processing – Loops | - | - |
| volání funkcí |   | 1x | Processing – Functions | - | Funkce se nabízí v Functions programs až po jejich vytvoření |
| wait |   | 1 | Processing – Util | - | - |

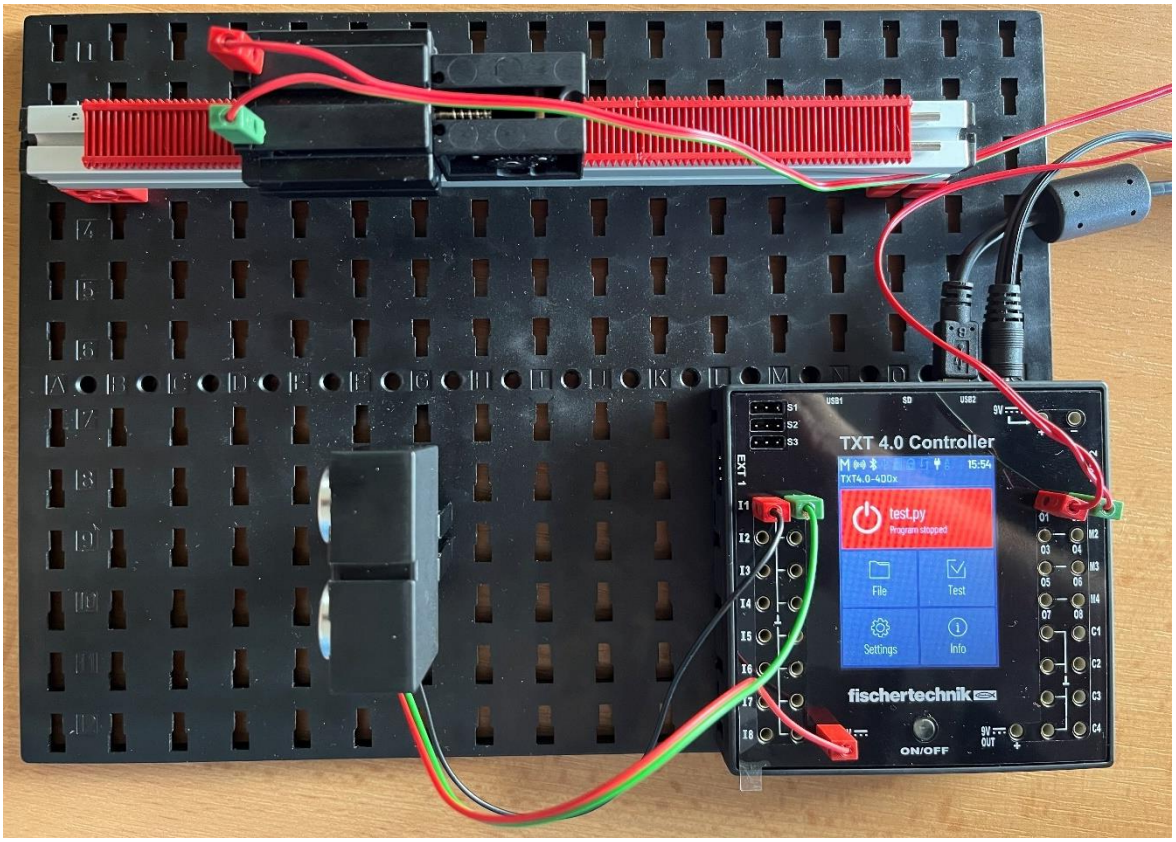
Ukázka, jak program funguje viz video Pojezd mezi pevně danými body.

## 3.7 Bezdotykově řízený pojezd

V tomto cvičení využijeme ultrazvukový sensor vzdálenosti pro řízení pojezdu. Ukážeme si, jak se z ultrazvukového sensoru vzdálenosti čtou data a jak lze přemapovat interval. Protože bude využíván displej, je potřeba založit projekt s displejem.

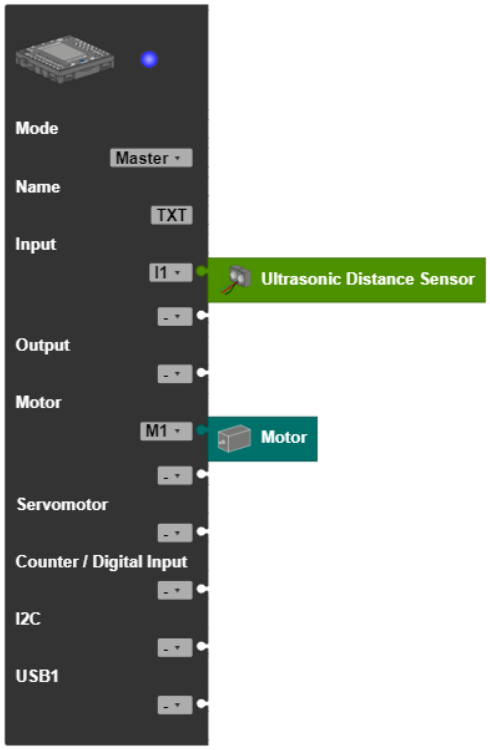
### 3.7.1 Bezdotykově řízený pojezd – zapojení komponent

Zapojení komponent je následující, viz obrázek. Ultrazvukový sensor vzdálenosti má 3 kabely, na vstup I1 je připojený černý kabel pro čtení údajů (vzdálenosti), zelený kabel je připojený na zem a červený kabel je připojený na neustále napájený pin +9V. Na výstup M1 je přivedený motor.




*Obrázek 3-32: Bezdotykově řízený pojezd – zapojení komponent*

Konfigurace kontroléru:




*Obrázek 3-33: Robo Pro Coding – Konfigurace Controlleru – Bezdotykově řízený pojezd*

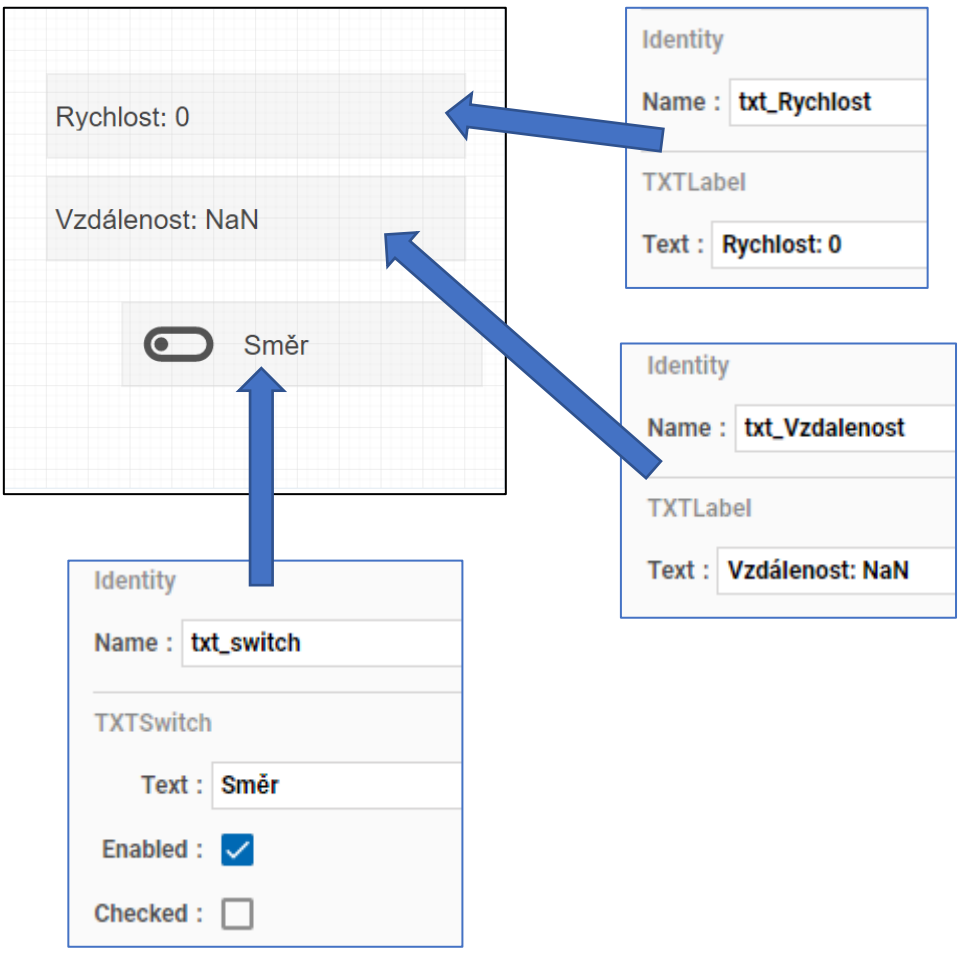
### 3.7.2 Bezdotykově řízený pojezd – řídící program

Jak chceme, aby se bezdotykově řízený pojezd choval…

1. Pojezd reaguje na překážku (ruku)
2. Při vzdálenosti ruky v rozsahu 10–0 cm jede pojezd doleva, výkon pojezdu je lineárně odpovídající ke vzdálenosti ruky. Tedy vzdálenost ruky 10 cm od sensoru je minimální rychlost, 0 cm je maximální rychlost pojezdu.
3. Při vzdálenosti ruky v rozsahu 25–15 cm jede pojezd doprava, výkon pojezdu je lineárně odpovídající ke vzdálenosti ruky. Tedy vzdálenost ruky 25 cm od sensoru je maximální rychlost, 15 cm je minimální rychlost pojezdu.
4. Při jakékoliv jiné vzdálenosti ruky od sensoru, než jsou dva výše uvedené intervaly ([0,10], [15,25]) pojezd stojí.
5. Na displeji řídící jednotky je vidět aktuální rychlost pojezdu, vzdálenost ruky od sensoru a směr je znázorněný přepínačem (switch) takovým způsobem, že když jede pojezd vlevo, je přepínač přepnutý doleva, pokud jede vpravo, je přepínač přepnutý doprava a pokud pojezd stojí, tak je přepínač na displeji zašednutý.

### 3.7.2.1 Informační panel

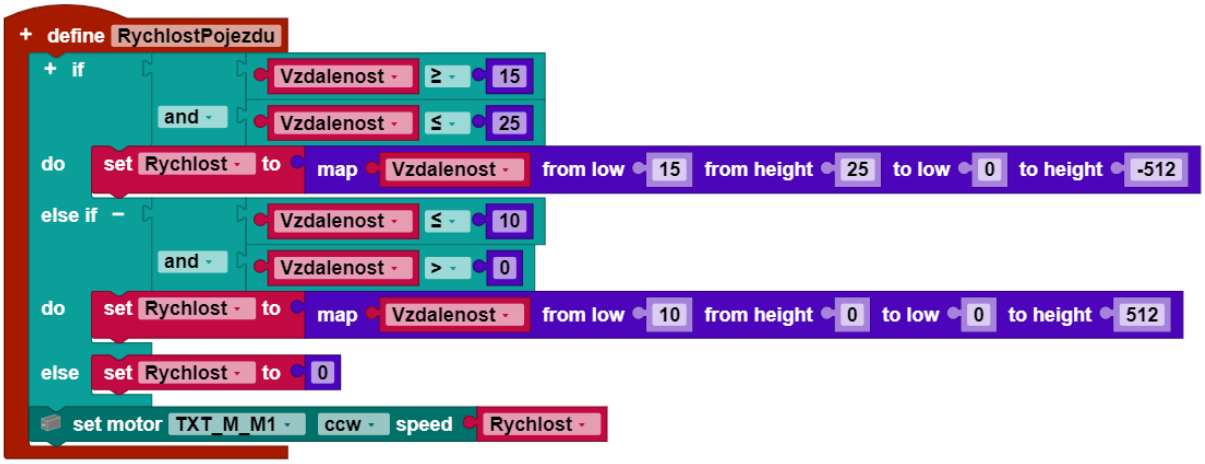
Informační panel v tomto případě zobrazí rychlost, vzdálenost a lze přepínat směr jízdy.




*Obrázek 3-34: Informační panel pro bezdotykově ovládaný pojezd*

### 3.7.2.2 Funkce nastavení rychlosti a směru pojezdu

Funkce nastavení rychlosti pojezdu nastavuje rychlost a směr pojezdu adekvátně vzdálenosti ruky od sensoru vzdálenosti.

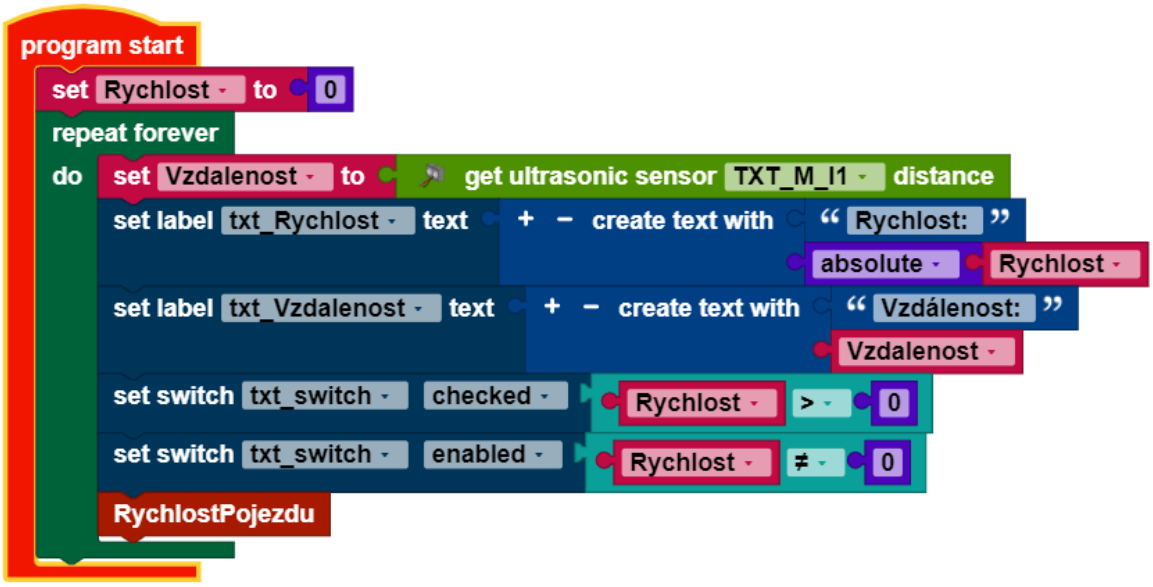



```
def RychlostPojezdu():
    global Rychlost, Vzdalenost
    if Vzdalenost >= 15 and Vzdalenost <= 25:
        Rychlost = ft_math.map(Vzdalenost, 15, 25, 0, -512)
    elif Vzdalenost <= 10 and Vzdalenost > 0:
        Rychlost = ft_math.map(Vzdalenost, 10, 0, 0, 512)
    else:
        Rychlost = 0
    TXT_M_M1_motor.set_speed(int(Rychlost), Motor.CCW)
    TXT_M_M1_motor.start()
```

*Obrázek 3-35: Funkce nastavení rychlosti a směru pojezdu*

### 3.7.2.3 Vytvoření těla hlavního programu

Hlavní program, kde se v cyklu načítá vzdálenost, nastavují se hodnoty do GUI a volá se podprogram RychostPojezdu.




```
import fischertechnik.utility.math as ft_math
import math

from fischertechnik.controller.Motor import Motor
from lib.controller import *
from lib.display import *


Rychlost = None
Vzdalenost = None


def RychlostPojezdu():
    global Rychlost, Vzdalenost
    if Vzdalenost >= 15 and Vzdalenost <= 25:
        Rychlost = ft_math.map(Vzdalenost, 15, 25, 0, -512)
    elif Vzdalenost <= 10 and Vzdalenost > 0:
        Rychlost = ft_math.map(Vzdalenost, 10, 0, 0, 512)
    else:
        Rychlost = 0
    TXT_M_M1_motor.set_speed(int(Rychlost), Motor.CCW)
    TXT_M_M1_motor.start()


Rychlost = 0
while True:
    Vzdalenost = TXT_M_I1_ultrasonic_distance_meter.get_distance()
    display.set_attr("txt_Rychlost.text", str('Rychlost: ' + str(math.fabs(Rychlost))))
    display.set_attr("txt_Vzdalenost.text", str('Vzdálenost: ' + str(Vzdalenost)))
    display.set_attr("txt_switch.checked", str(Rychlost > 0).lower())
    display.set_attr("txt_switch.enabled", str(Rychlost != 0).lower())
    RychlostPojezdu()
```

*Obrázek 3-36: Hlavní program bezdotykově řízeného pojezdu*

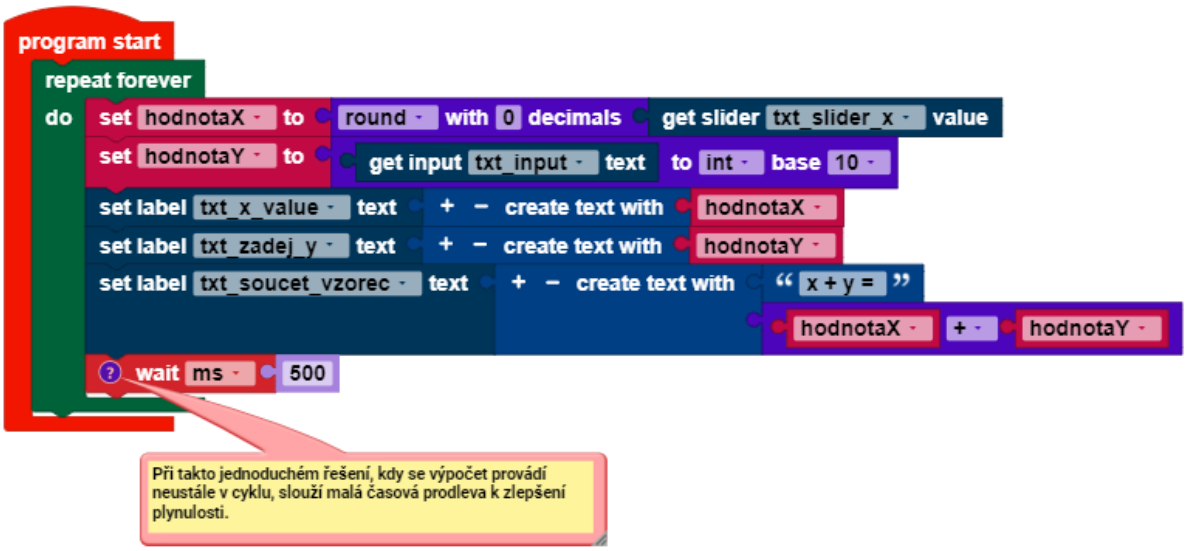
## 3.8 Kalkulačka

V tomto cvičení vytvoříme jednoduchou kalkulačku s jednoduchým grafickým rozhraním. Pro tento příklad není potřeba připojovat k řídící jednotce žádný hardware. Protože bude využíván displej, je potřeba založit projekt s displejem.

### 3.8.1 Kalkulačka – řídící program

Jak chceme, aby se kalkulačka chovala…

1. Kalkulačka bude zobrazovat součet dvou čísel.
2. Sliederem (hodnota x) se zadávají pouze celá čísla v intervalu [0,10]. Výchozí hodnota je „5“.
3. Sliderem zadaná hodnota se zobrazuje v labelu nad sliderem.
4. Číslo y se zadává pomocí kontrolky TXTInput (je jej tedy nutné přetypovat na int).
5. Přetypování hodnoty y nebude ošetřené.

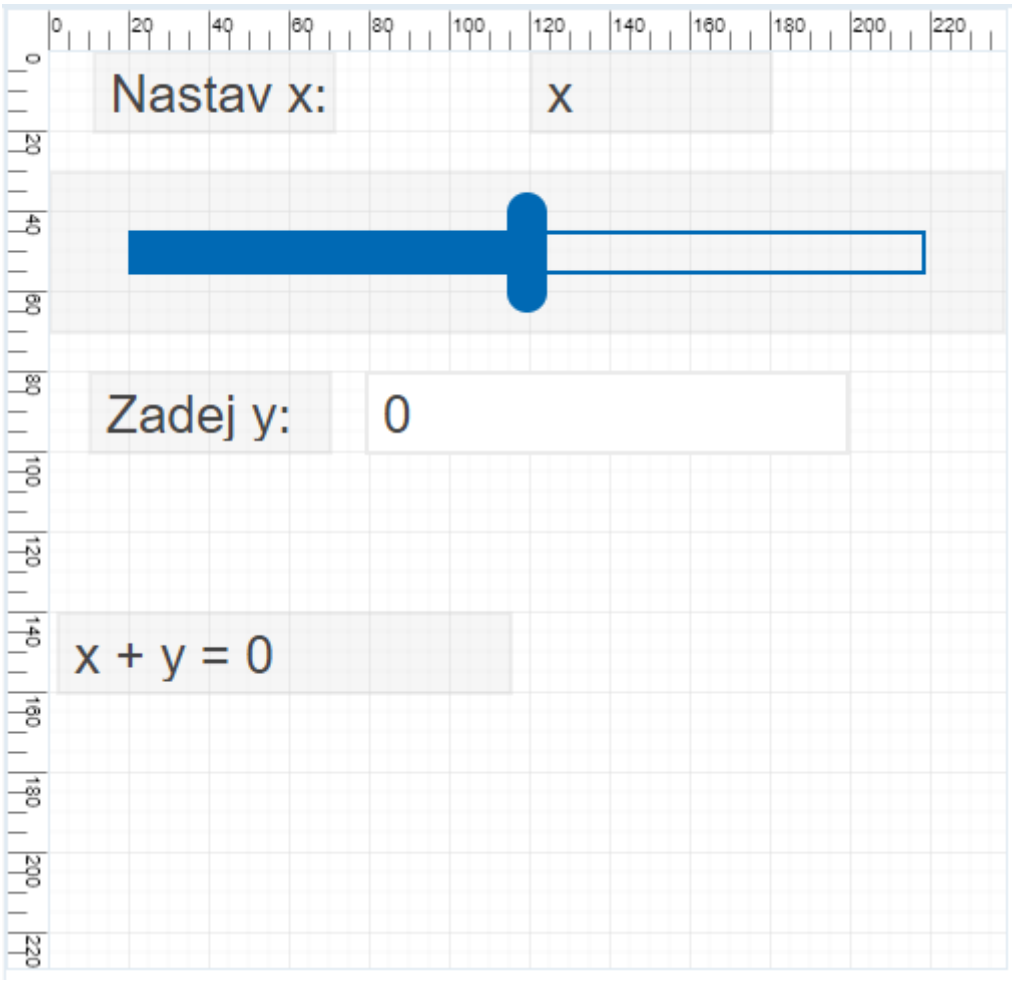



```
import math
import time

from lib.controller import *
from lib.display import *


hodnotaX = None
hodnotaY = None


while True:
    hodnotaX = round(display.get_attr("txt_slider_x.value"))
    hodnotaY = int(display.get_attr("txt_input.text"), 10)
    display.set_attr("txt_x_value.text", str(str(hodnotaX)))
    display.set_attr("txt_zadej_y.text", str(str(hodnotaY)))
    display.set_attr("txt_soucet_vzorec.text", str('x + y = ' + str(hodnotaX + hodnotaY)))
    # Při takto jednoduchém řešení, kdy se výpočet provádí neustále
    # v cyklu, slouží malá časová prodleva k zlepšení plynulosti.
    time.sleep(0.5)
```

*Obrázek 3-37: Kalkulačka – řídící program*

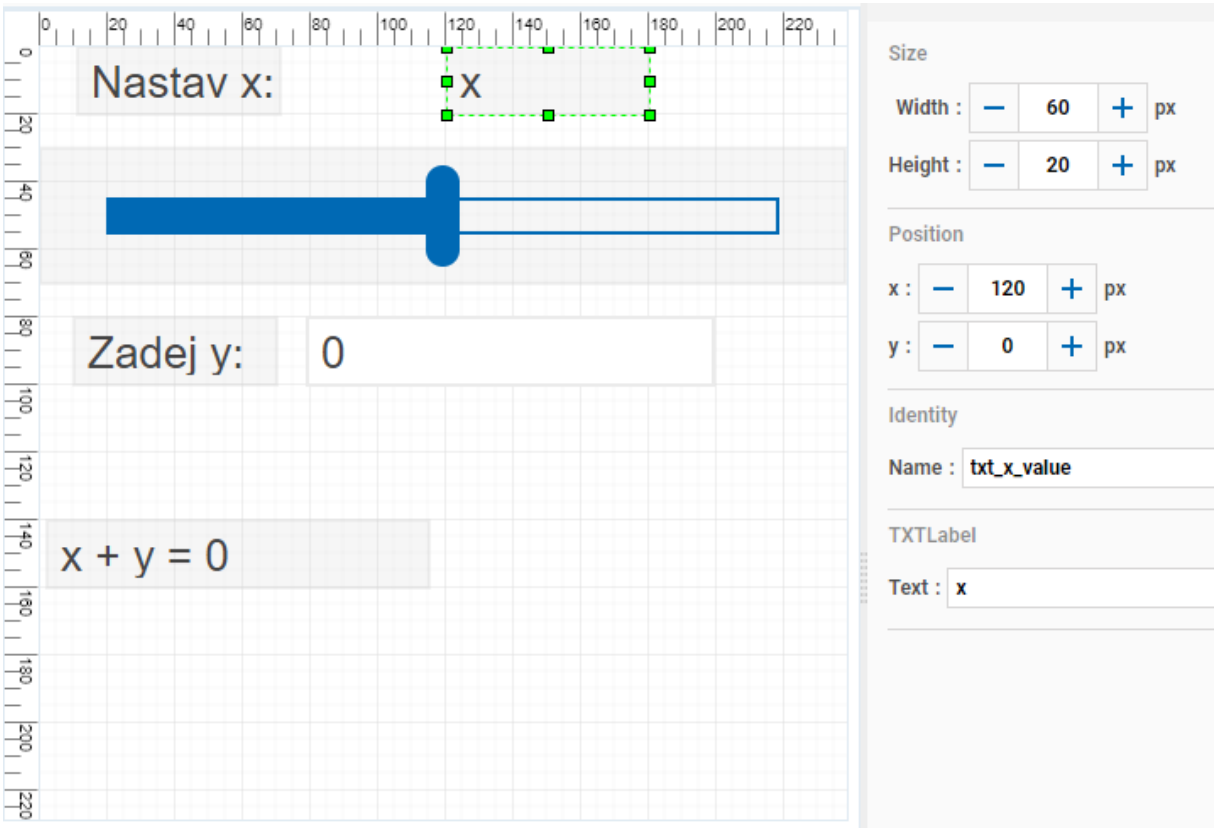
#### 3.8.1.1 Ukázka nastavení vybraných elementů

Grafické rozhraní bude dle následujícího obrázku.



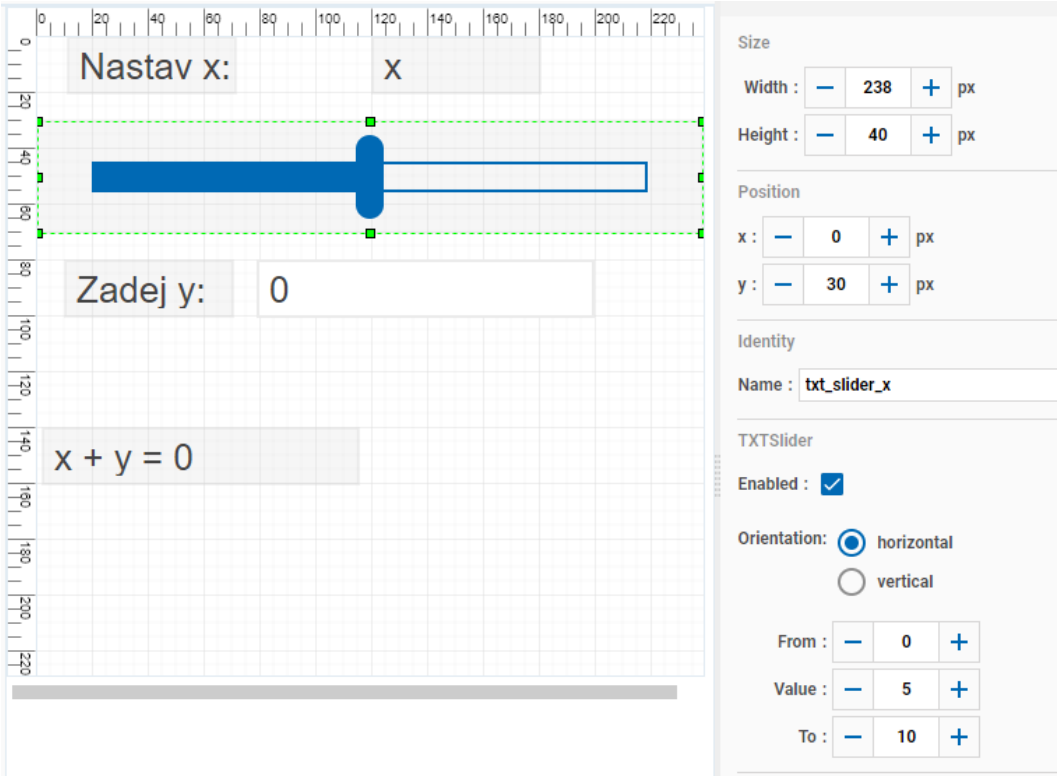

*Obrázek 3-38: Grafické rozhraní kalkulačky*

Pro label zobrazující hodnotu nastavenou na slideru nastavíme pouze jméno „txt_x_value“ a text (text není s ohledem na kód povinný, kvůli výchozí hodnotě slideru bude po spuštění programu přepsán):



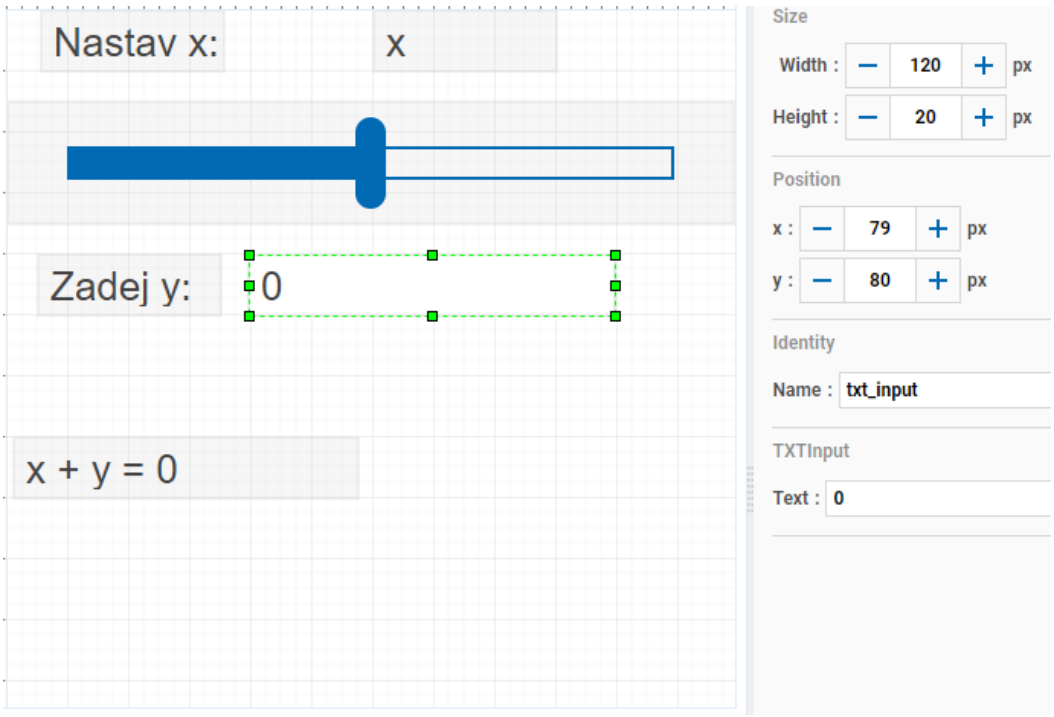

*Obrázek 3-39: Nastavení ukazatelů hodnot sliderů u kalkulačky*

U slideru nastavíme název „txt_slider_x“ a rozsah hodnot from = 0, to = 10 a výchozí hodnotu Value = 5:



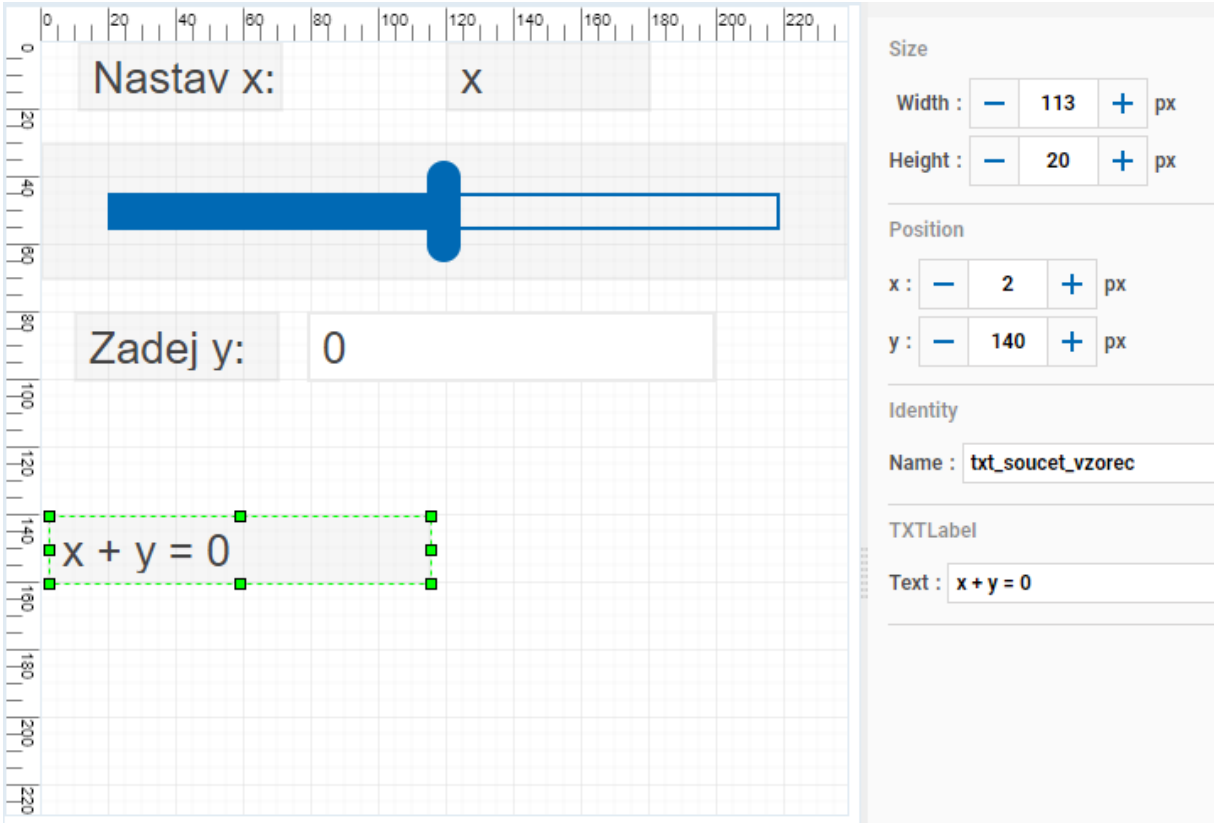

*Obrázek 3-40: Nastavení slideru u kalkulačky*

U TXTInputu nastavíme název na „txt_input“ a Text na „0“ (nulu). Text „Zadej y:“ je v klasickém labelu a slouží pouze jako informace o hodnotě, která se zadává TXTInputem.




*Obrázek 3-41: Nastavení TXTInputu u kalkulačky*

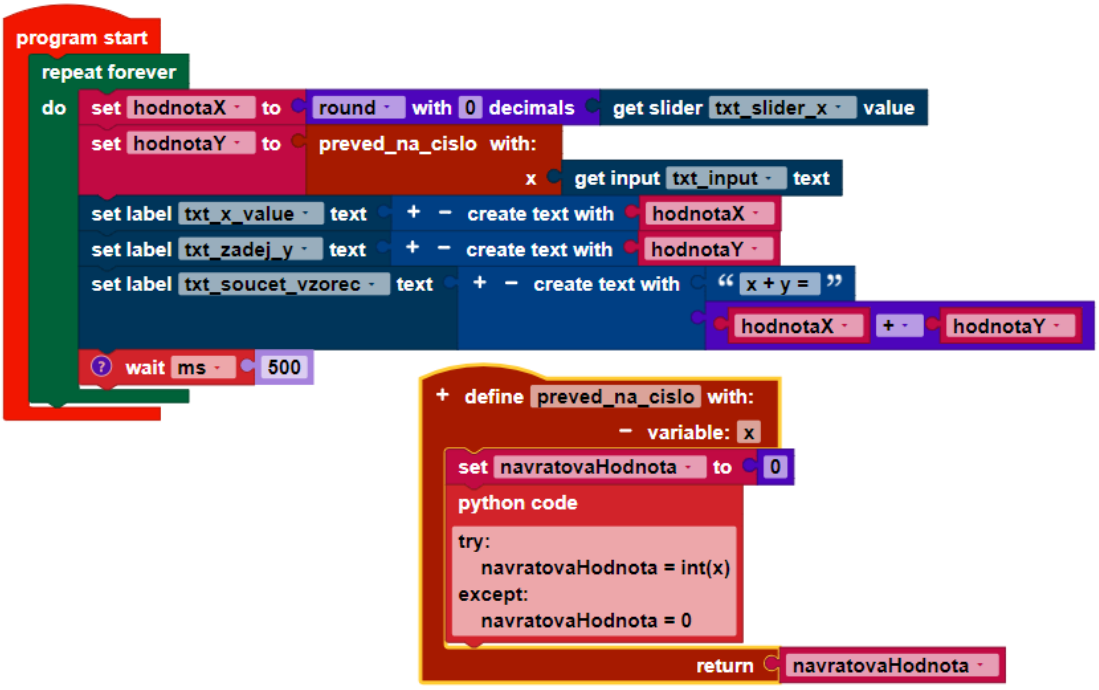
U labelu pro zobrazení výsledku nastavíme název „txt_soucet_vzorec“ a výchozí text „x + y = 0“, který z logiky řídícího kódu není povinný, protože po spuštění programu bude přepsán.




*Obrázek 3-42: Nastavení labelu určeného pro zobrazení výsledku součtu*

### 3.8.2 Kalkulačka – řídící program s přetypováním hodnoty

V předchozím programu nemáme ošetřený vstup z TXTInputu. TXTInput vrací zadaná data jako „string“. Je tedy potřeba přetypovat data přebraná z TXTInputu na integer. V aktuální verzi Robo Pro Coding není blok pro ošetření vyjímky, ale lze využít blok pro vložení vlastního kódu v Pythonu – toho je využito v ukázaní řešení. Pro přetypování je využit jednoduchý try-except (obdoba Try-Catch ze C#). Pokud se hodnotu nepodaří přetypovat na integer, bude hodnota z TXTInputu nastavavena na 0.




```
import math
import time

from lib.controller import *
from lib.display import *


x = None
navratovaHodnota = None
hodnotaX = None
hodnotaY = None


def preved_na_cislo(x):
    global navratovaHodnota, hodnotaX, hodnotaY
    navratovaHodnota = 0
    try:
        navratovaHodnota = int(x)
    except:
        navratovaHodnota = 0
    return navratovaHodnota


while True:
    hodnotaX = round(display.get_attr("txt_slider_x.value"))
    hodnotaY = preved_na_cislo(display.get_attr("txt_input.text"))
    display.set_attr("txt_x_value.text", str(str(hodnotaX)))
    display.set_attr("txt_zadej_y.text", str(str(hodnotaY)))
    display.set_attr("txt_soucet_vzorec.text", str('x + y = ' + str(hodnotaX + hodnotaY)))
    # Při takto jednoduchém řešení, kdy se výpočet provádí neustále
    # v cyklu, slouží malá časová prodleva k zlepšení plynulosti.
    time.sleep(0.5)
```

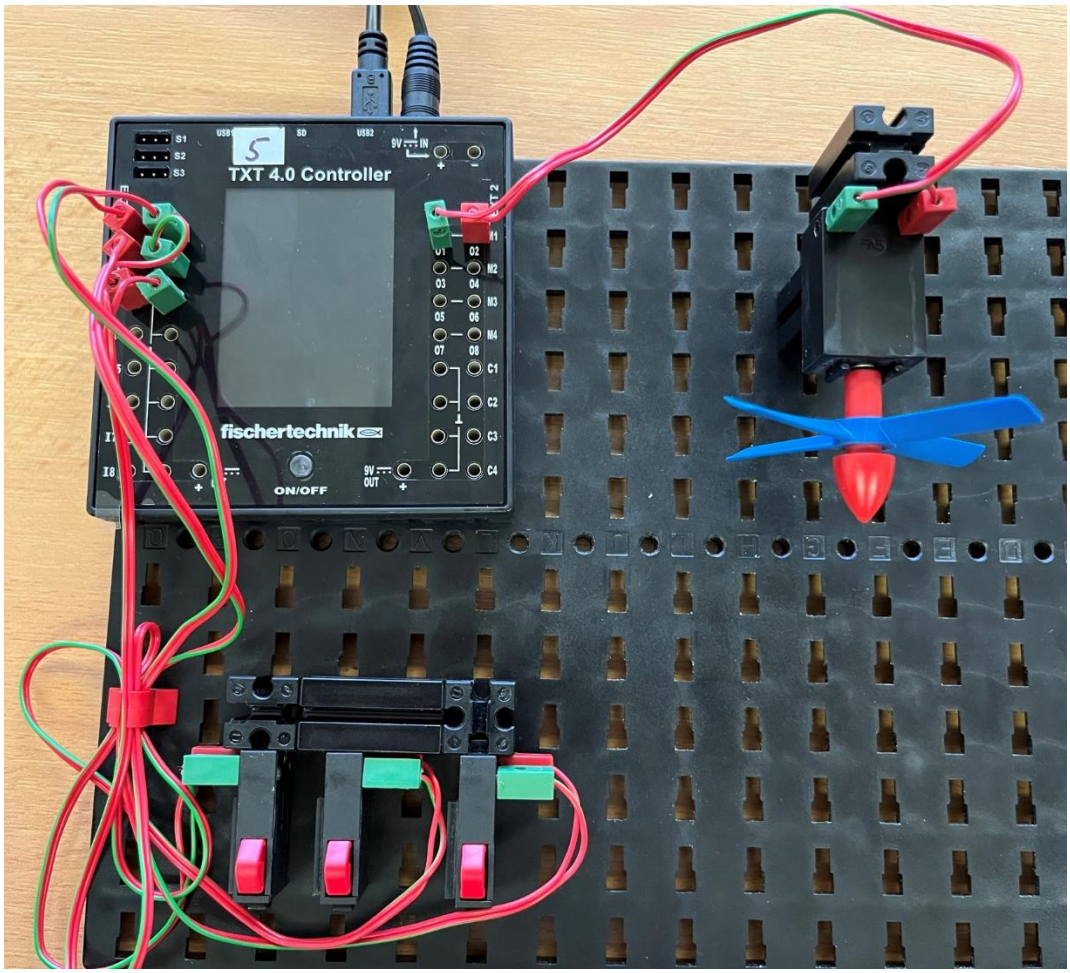
*Obrázek 3-43: Kalkulačka – s přetypováním stringu na integer*

## 3.9 Logický motor

V tomto cvičení se na chvíli vrátíme k větráku a naučíme se využívat logické operátory a matematické operace, zopakujeme si funkce a naučíme se předávat potřebné hodnoty do funkce a naopak z funkce je vracet do programu.

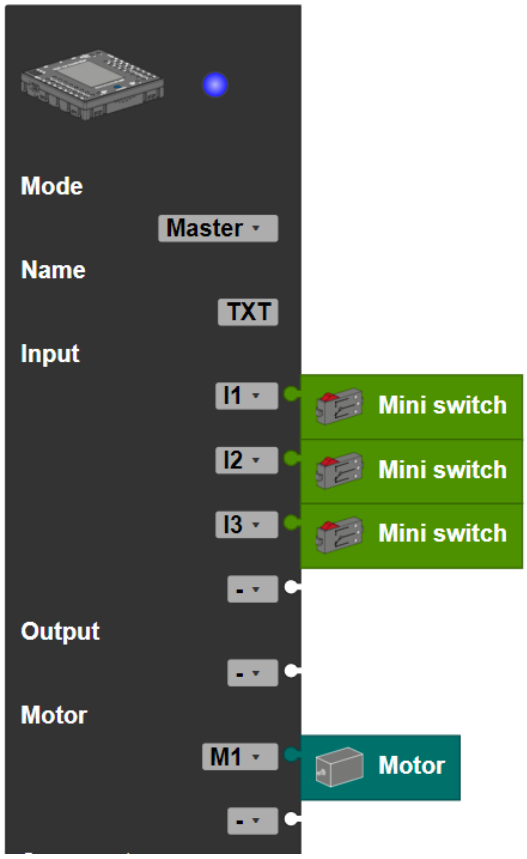
### 3.9.1 Logický motor – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstupy I1, I2 a I3 jsou přivedeny spínače. Na výstup M1 je přivedený motor.




*Obrázek 3-44: Logický motor – zapojení komponent*

Konfigurace kontroléru:




*Obrázek 3-45: Robo Pro Coding – Konfigurace Controlleru – Logický motor*

### 3.9.2 Logický motor – řídící program

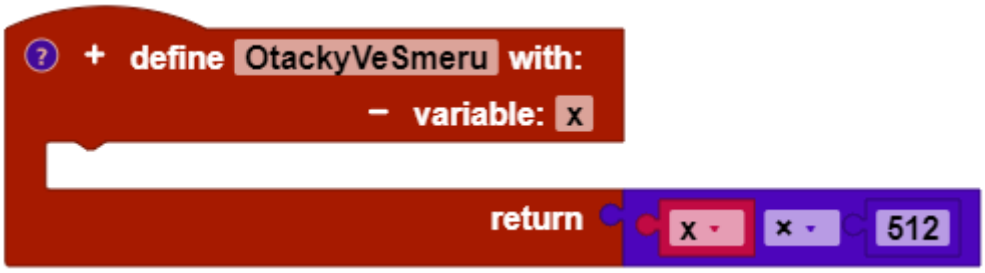
Jak chceme, aby se logický motor choval…

1. Při stisknutí pouze jednoho, jakéhokoliv tlačítka se nic neděje.
2. Tlačítko I1 je bezpečnostní
3. Tlačítko I2 spouští otáčky ve směru hodinových ručiček. Podmínka je, že k němu musí být stisknuto i tlačítko I1.
4. Tlačítko I3 spouští otáčky v protisměru hodinových ručiček s poloviční rychlostí. Podmínka je, že k němu musí být stisknuto i tlačítko I1.

Stejného výsledku lze dosáhnout propojením elementů spínačů s vhodně nadefinovanými podmínkami. Úkolem tohoto cvičení je si ukázat, že to lze i jinak a že existují logické operace a jak fungují.

Praktické využití je např. bezpečné spouštění stroje, kdy chceme mít jistotu, že obsluha má obě ruce mimo a nehrozí tak její poranění. Ukázka funkčnosti modelu viz video Logický motor.

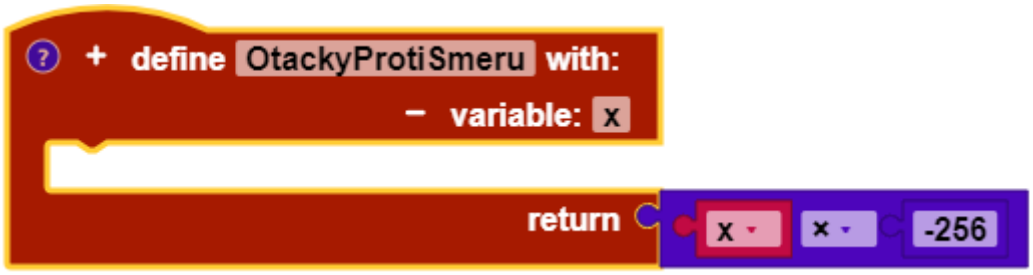
Funkce pro otáčky motoru ve směru hodinových ručiček:




```
# Do podprogramu vstupuje logická 0 (nestisknuly se správná
# tlačítka), nebo 1 (stisknuly se správná tlačítka).
# Vynásobíme to hodnotou 512 (maximální rychlost motoru) a
# předáme jako výstup podprogramu zpět do hlavního programu.
def OtackyVeSmeru(x):
  global Rychlost
  return x * 512
```

*Obrázek 3-46: Logický motor – funkce pro otáčky motoru ve směru hodinových ručiček*

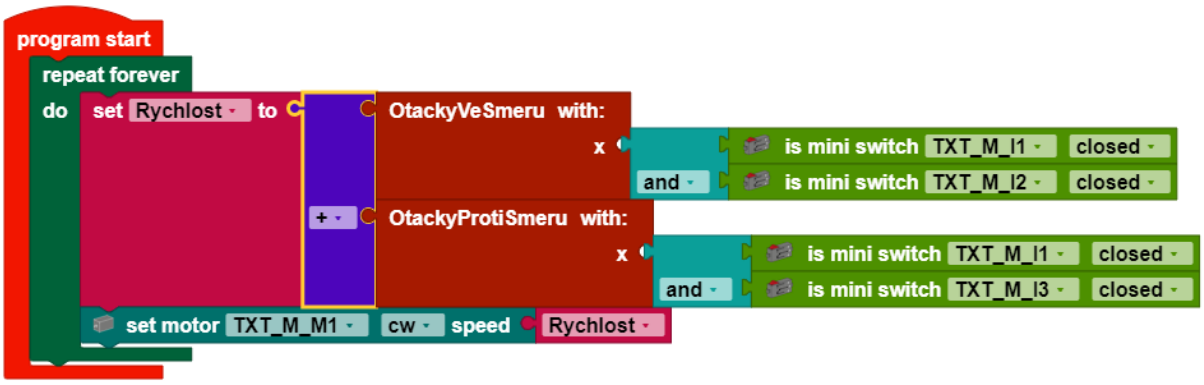
Funkce pro otáčky motoru v protisměru hodinových ručiček:




```
# Do podprogramu vstupuje logická 0 (nestisknuly se správná
# tlačítka), nebo 1 (stisknuly se správná tlačítka).
# Vynásobíme to hodnotou -256 (poloviční rychlost motoru, obrácené
# otáčky) a předáme jako výstup podprogramu zpět do hlavního programu.
def OtackyProtiSmeru(x):
  global Rychlost
  return x * -256
```

*Obrázek 3-47: Logický motor – funkce pro otáčky motoru v protisměru hodinových ručiček*

Hlavní program:




*Obrázek 3-48: Logický motor – Program*

## 3.10 Větráček II – návrat k základům a jejich rozšíření

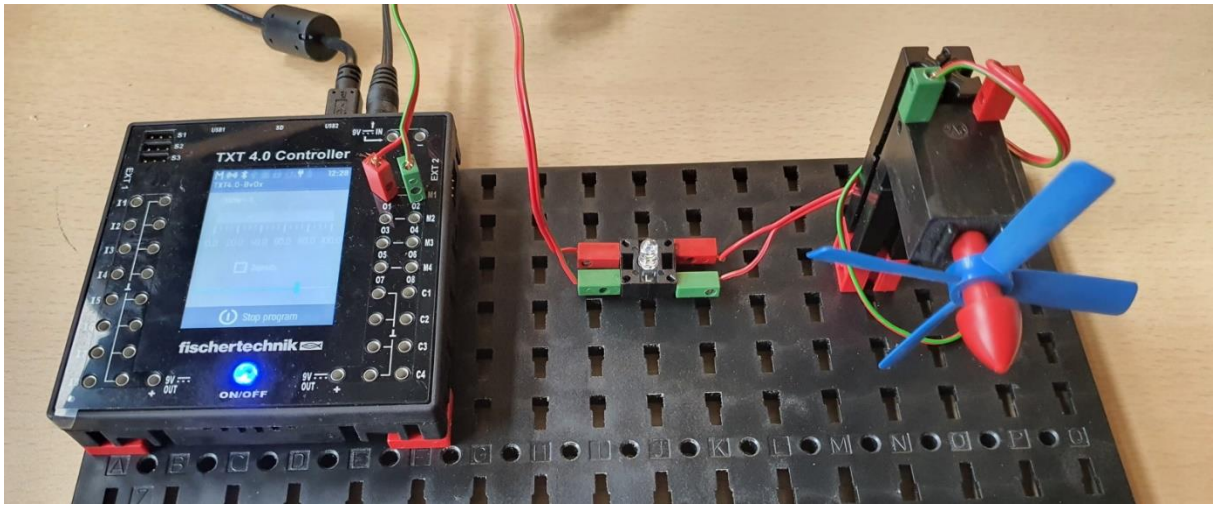
V tomto cvičení se naučíme pracovat s ukazatelem, Slidery (posuvníky), vytvářet GUI pro ovládání modelu na dotykovém displeji řídící jednotky a nahrání do řídící jednotky tak, aby program fungoval samostatně bez propojení řídící jednotky s počítačem. Navíc si ukážeme, že lze zapojovat komponenty paralelně.

### 3.10.1 Větráček II – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na výstup M1 je přivedena světelná komponenta a ta je dále propojena s motorem.

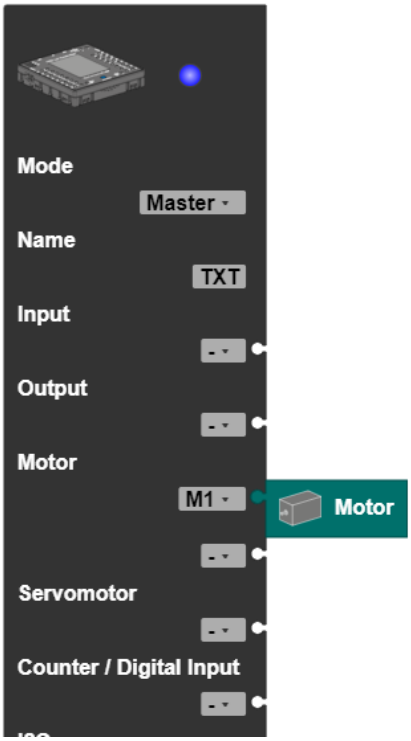
POZOR!!!

Při využití LED diody lze otáčet větráčkem pouze v jednom směru! Napětí musí jít na +LED a zem na -LED. Pokud by se zapojilo obráceně, nebo by se změnil směr otáček motůrku softwarově, tak bychom využívali LED v závěrném směru a pak by fungovala jako pojistka. Nedošlo by tak k jejímu rozsvícení ani roztočení motoru a mohlo by dojít k trvalému poškození LED (proražení LED v závěrném směru). V případě využití žárovky tento problém není.




*Obrázek 3-49: Větráček II – zapojení komponent*

Konfigurace kontroléru:



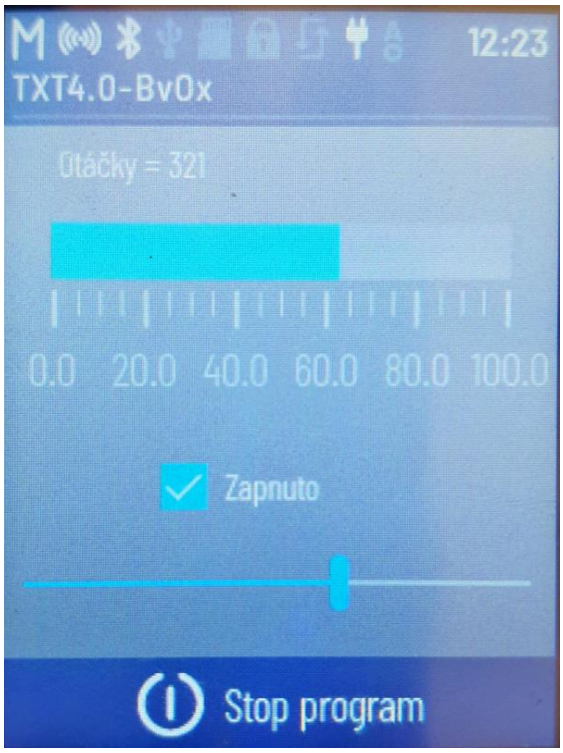

*Obrázek 3-50: Robo Pro Coding – Konfigurace Controlleru – Větráček II*

### 3.10.2 Větráček II – řídící program v režimu online propojení s počítačem

Jak chceme, aby se kontrolor choval…

1. Větráček bude možné zapnout pouze tehdy, když na řídící jednotce uživatel změní stav na zapnuto.
2. Při použití slideru (posuvníku) na řídící jednotce se bude adekvátně rozsvěcet/zhasínat světelná komponenta a roztáčet větráček.
3. Intenzita emitace světla světelnou komponentou a rychlost otáček větráčku bude zobrazen ukazatelem se stupnicí na řídící jednotce od 0 do 100 %.
4. Program bude v cyklu, aby bylo možné jej spouštět opakovaně.

Ukázka obrazovky řídící jednotky:

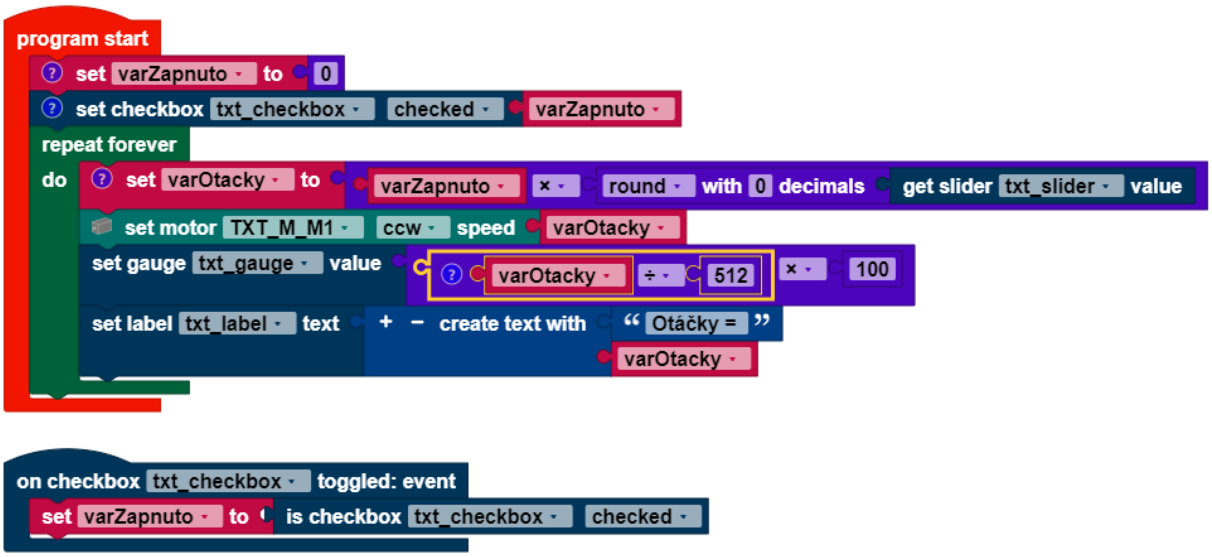



*Obrázek 3-51: Řídící jednotka – ukázka obrazovky – Větráček II*

**Co v tomto případě potřebujeme vědět…**

Intenzita svícení světelné komponenty a rychlost otáček motoru jsou dány rozsahem 0–512. Kdy při hodnotě 0 jsou komponenty vypnuty, při hodnotě 512 běží na plný výkon (největší svítivost světelné komponenty a nejvyšší otáčky motoru). Celý program je pak na následujícím obrázku.

U tohoto příkladu je třeba si uvědomit, že při spuštění programu je nutné definovat typ proměnné **varZapnuto**, aby bylo možné spočítat rychlost otáček do proměnné **varOtacky**. Deklarace typu se realizuje přiřazením hodnoty do proměnné. Také platí, že rychlost otáček motoru se nastavuje pomocí celých čísel. Z tohoto důvodu vypočtenou rychlost zaokrouhlujeme na celé číslo.







```python
import math

from fischertechnik.controller.Motor import Motor
from lib.controller import *
from lib.display import *


varZapnuto = None
varOtacky = None



def on_txt_checkbox_toggled(event):
    global varZapnuto, varOtacky
    varZapnuto = display.get_attr("txt_checkbox.checked")


display.checkbox_toggled("txt_checkbox", on_txt_checkbox_toggled)


# provede se deklarace typu proměnné a nastaví se počáteční hodnota
varZapnuto = 0
# podle hodnoty proměnné se nastaví počáteční stav pro CheckBox
display.set_attr("txt_checkbox.checked", str(varZapnuto).lower())
while True:
    # Nastaví se proměnné pro otáčky jen pokud je zaškrtnuto zapnuto.
    # Otáčky se zaokrouhlí na celá čísla podle nastavení Slider
    varOtacky = varZapnuto * round(display.get_attr("txt_slider.value"))
    TXT_M_M1_motor.set_speed(int(varOtacky), Motor.CCW)
    TXT_M_M1_motor.start()
    # Přepočet na procenta
    display.set_attr("txt_gauge.value", str((varOtacky / 512) * 100))
    display.set_attr("txt_label.text", str('Otáčky = ' + str(varOtacky)))
```

*Obrázek 3-52: Větráček 2 - řídící program*

Funkčnost modelu viz video Větráček 2.

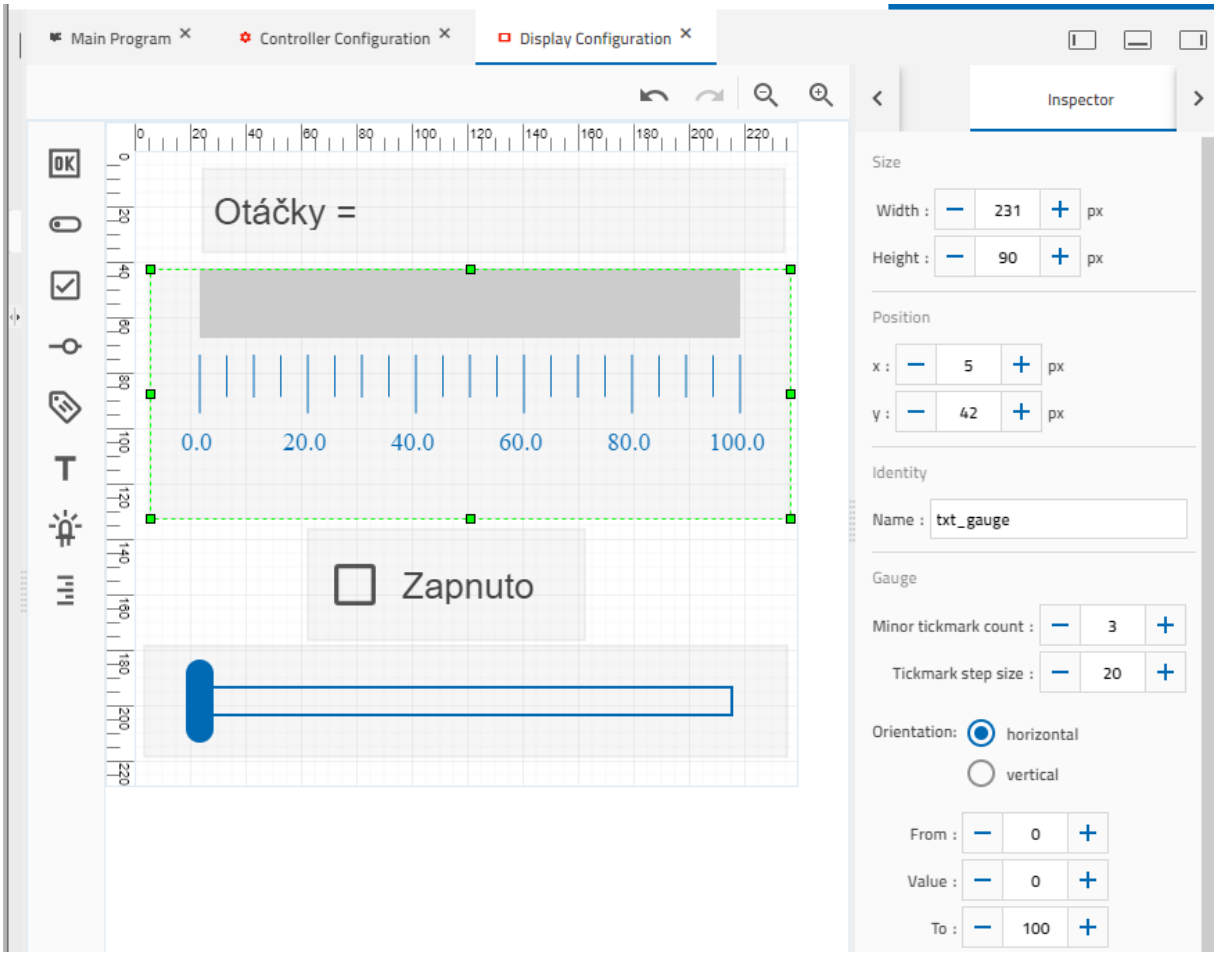
#### *3.10.2.1 Ukázka nastavení vybraných elementů*

V této kapitole je ukázáno grafické rozhraní pro model Větráček 2. a je ukázáno nastavení vybraných grafických prvků.

##### 3.10.2.1.1 Gauge

Element Gauge nastavíme:

- Minimum value = 0 hodnota vypnutí komponent
- Maximum value = 100 maximální výkon komponent
- Minor tickmark count = 3 vedlejší krok na stupnici
- Tickmark step size = 20 hlavní hodnoty ukazatele budou mít krok na stupnici = 20




*Obrázek 3-53: Větráček II – element Gauge*

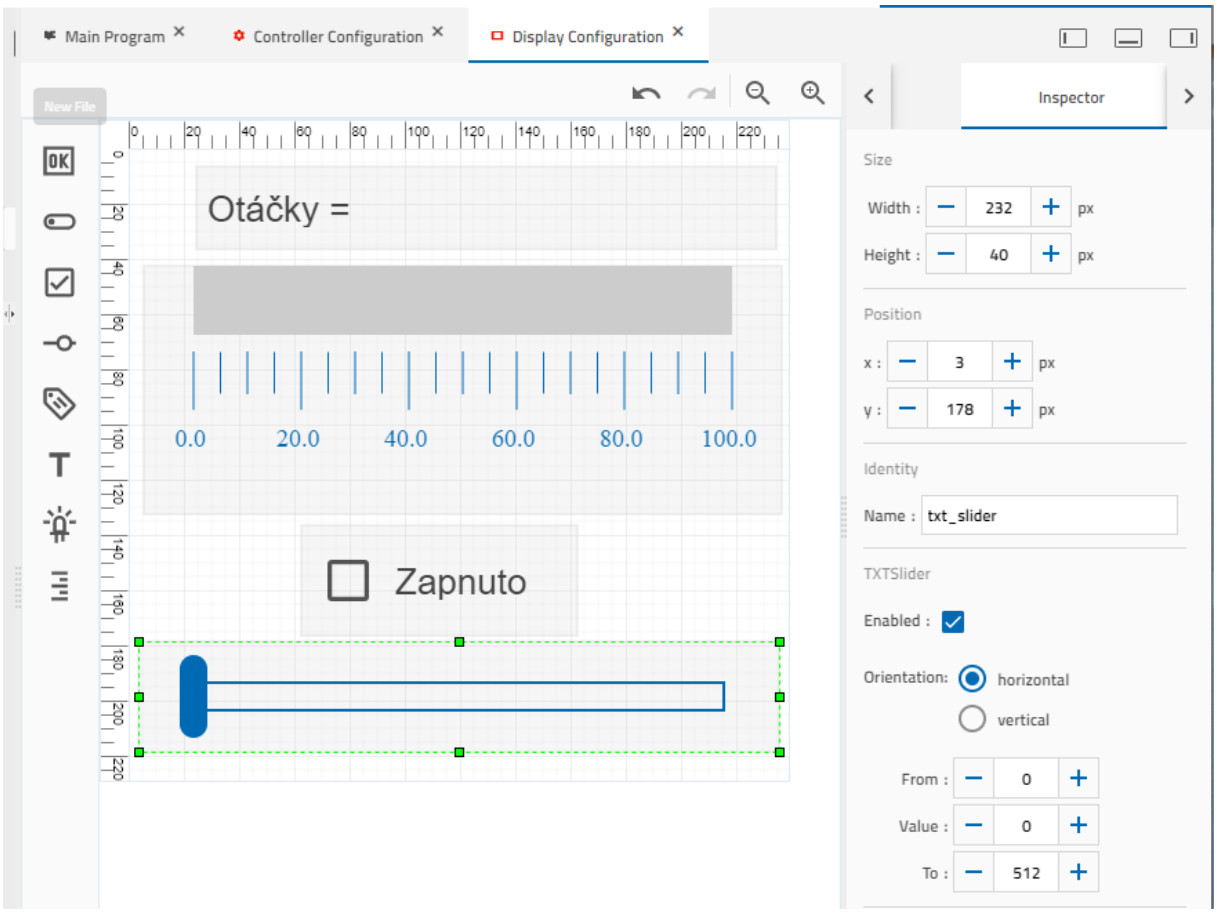
##### 3.10.2.1.2 Slider

Element **txtSlider** nastavíme:

From = 0 Při pozici jezdce úplně vlevo budou komponenty vypnuty.

To = 512 Při pozici úplně vpravo budou komponenty puštěny na maximum.

Value = 0 Při spuštění programu bude jezdec na pozici úplně vlevo.



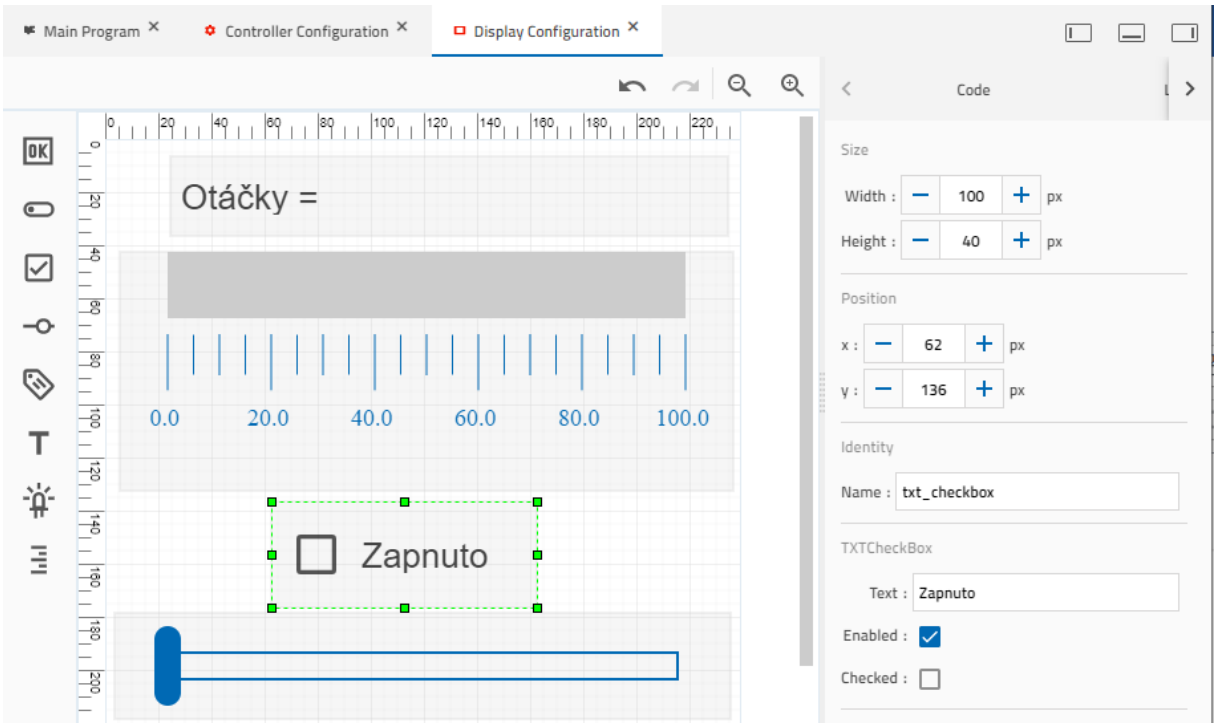

*Obrázek 3-54: Větráček II – element Slider*

#### 3.10.2.1.3 CheckBox

Element **txtCheckBox** nastavíme:

Enabled = true - Uživatel může měnit.

Checked = false - CheckBox není zaškrtnut



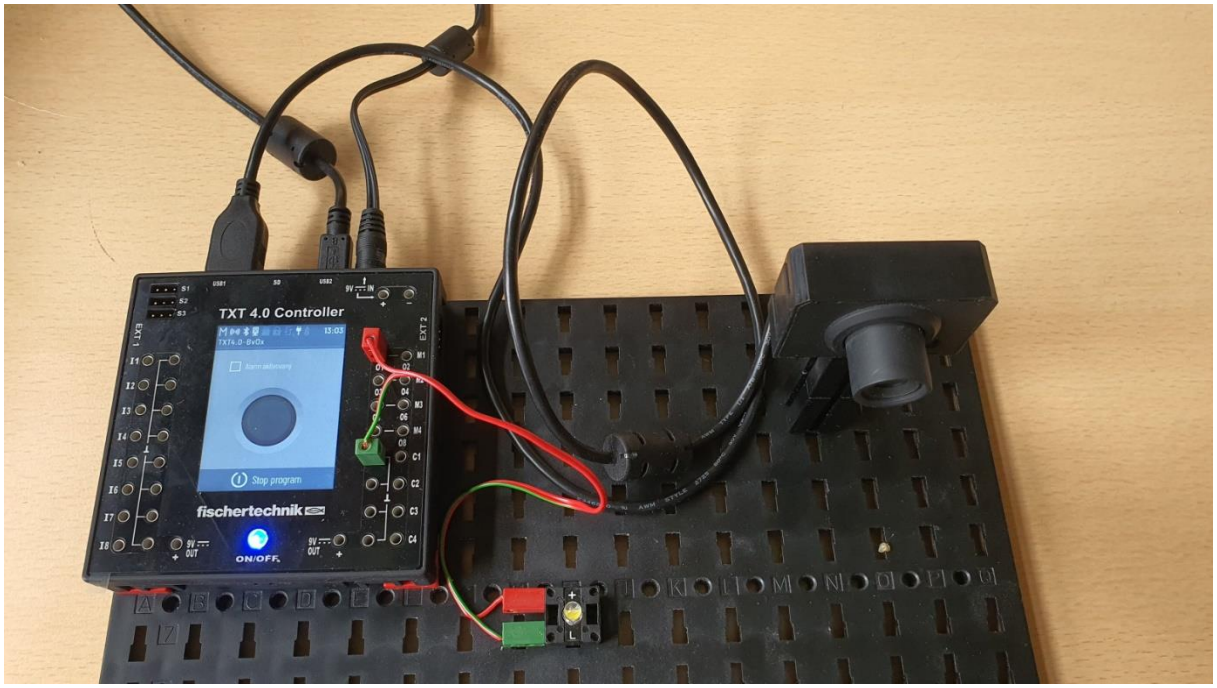

*Obrázek 3-55: Větráček II – element CheckBox*

## 3.11 Alarm

V tomto cvičení si ukážeme vyhodnocení pohybu před kamerou, využívání zvuků a zopakujeme si funkce.

### 3.11.1 Alarm – zapojení komponent

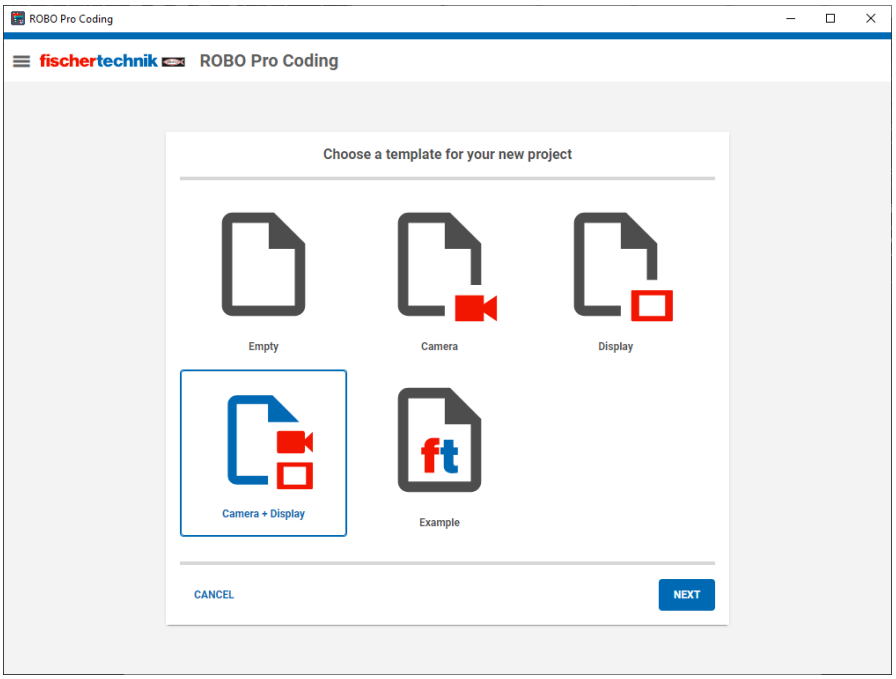
Zapojení komponent je následující, viz obrázek. Do TXT 4.0 Controlleru je zapojena kamera, na výstup O1 je přivedena LED s červenou krytkou. V tomto případě určitě nepoužijeme žárovku z důvodu potřeby rychlého blikání alarmu. Žárovka by neměnila stavy tak rychle a zároveň by k ní požadované chování nebylo šetrné.




*Obrázek 3-56: Alarm – zapojení komponent*

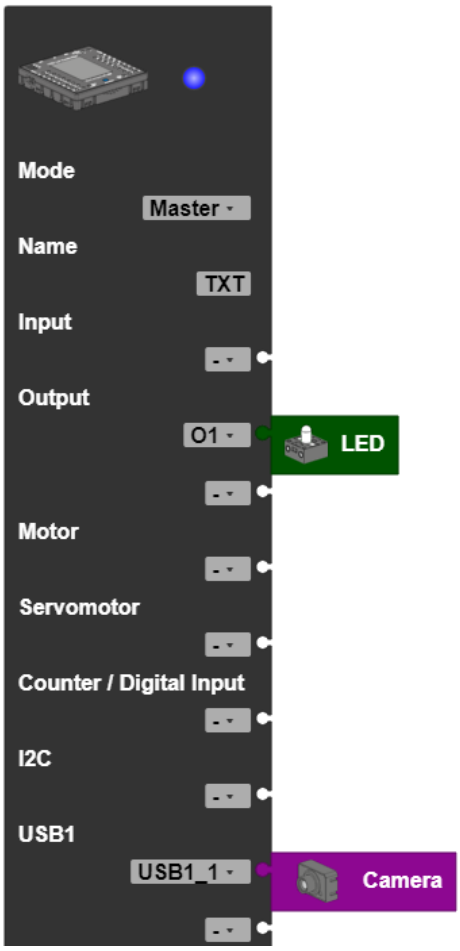
**Založení nového projektu**

V projektu budeme potřebovat definovat ovládací prvky na display řídící jednotky a zároveň získávat data z USB kamery. Z tohoto důvodu již při zakládání projektu vybereme volbu **Camera + Display**.




*Obrázek 3-57: Založení nového projektu*

Konfigurace kontroléru:

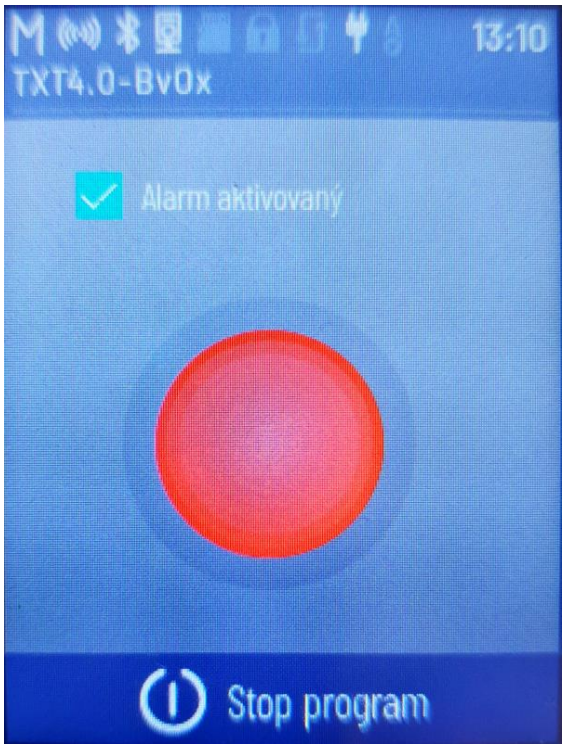



*Obrázek 3-58: Robo Pro Coding – Konfigurace Controlleru – Alarm*

### 3.11.2 Alarm – řídící program

Jak chceme, aby se alarm choval…

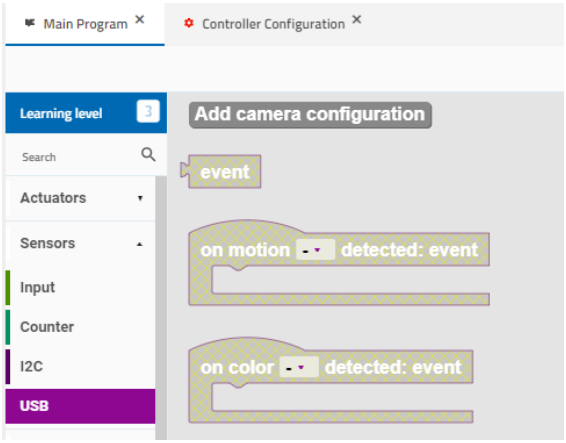
1. Na displeji řídící jednotky bude možné zapnout, nebo vypnout alarm.
2. Při vyhodnocení pohybu před kamerou se spustí zvuková výstraha a zároveň bude blikat červené světlo. Taktéž bude blikat červené světlo na řídící jednotce.
3. Program poběží ve smyčce, aby se neustále vyhodnocoval pohyb před kamerou a alarm mohl reagovat.




*Obrázek 3-59: Řídící jednotka – ukázka obrazovky – Alarm*

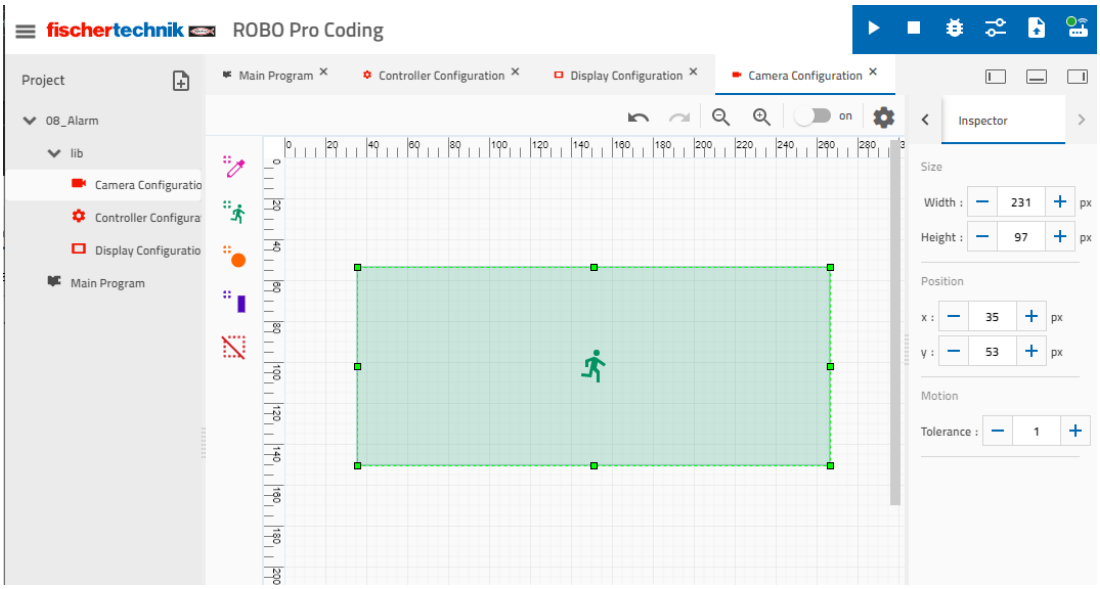
Nejprve je potřeba nastavit, co má kamera vyhodnocovat. Postup:

- V sekci Sensors -> USB klikneme na **Add camera configuration.**



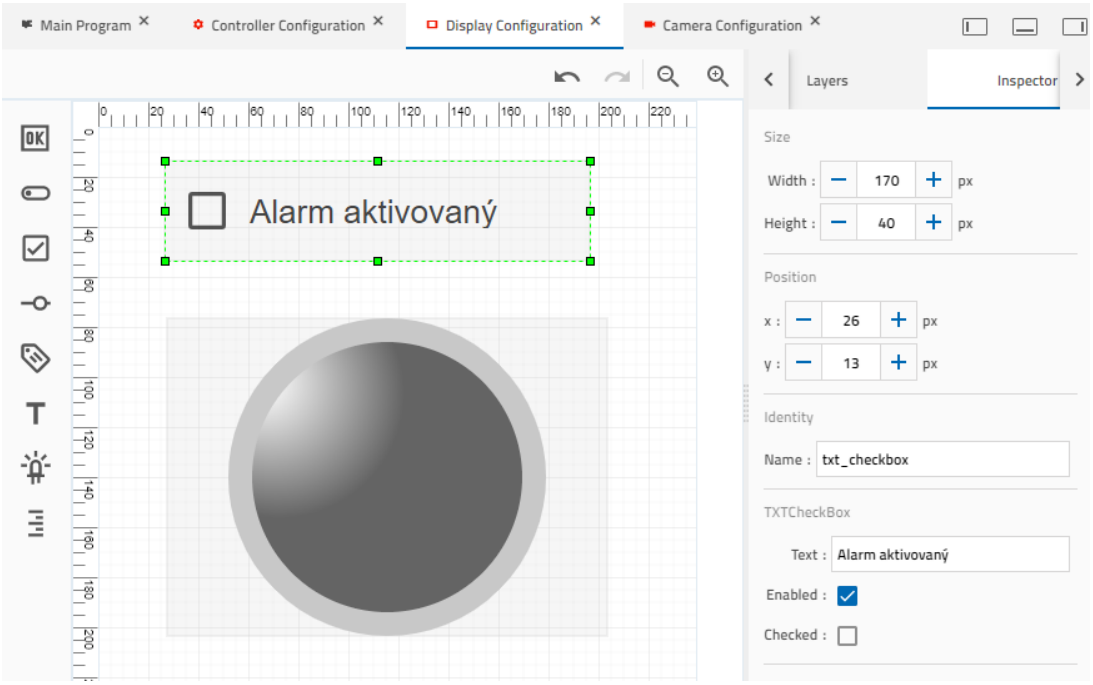

Obrázek 3-60: Alarm – přidání konfigurace kamery

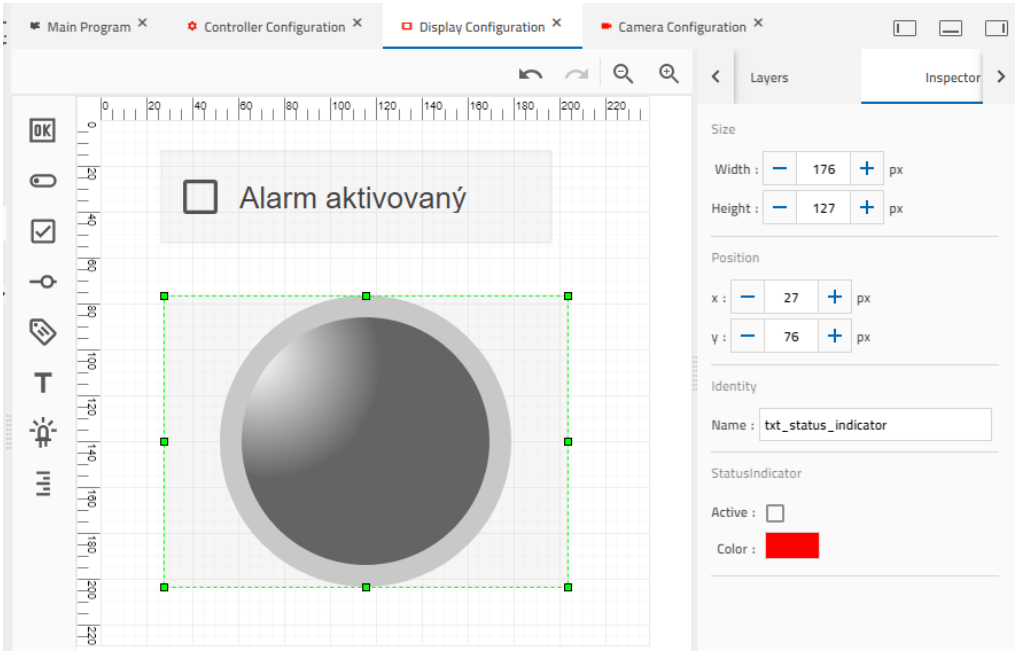
- V hlavním programu, na záložce **Camera Configuration**, vložíme komponentu **MotionDetector**, kterou zakreslíme do plochy symbolizující snímaný prostor kamery. Lze si při tom zaškrtávátkem „Activate Preview“ zobrazit místo bílé plochy obraz z kamery.
- Podle toho, jakou oblast označíme (velikost, umístění), tak tuto oblast bude kamera hlídat. Přes pravé tlačítko myši nad označením vybrané plochy (Inspector) si pak zobrazíme možnosti nastavení a nastavíme si, jak citlivá má být kamera při vyhodnocování pohybu. Pohyb je vyhodnocován na základě změny kontrastu obrazu, předdefinovaná hodnota 1 je naprosto dostačující pro citlivé vyhodnocování.




*Obrázek 3-61: Alarm – nastavení kamery pro využití v programu*

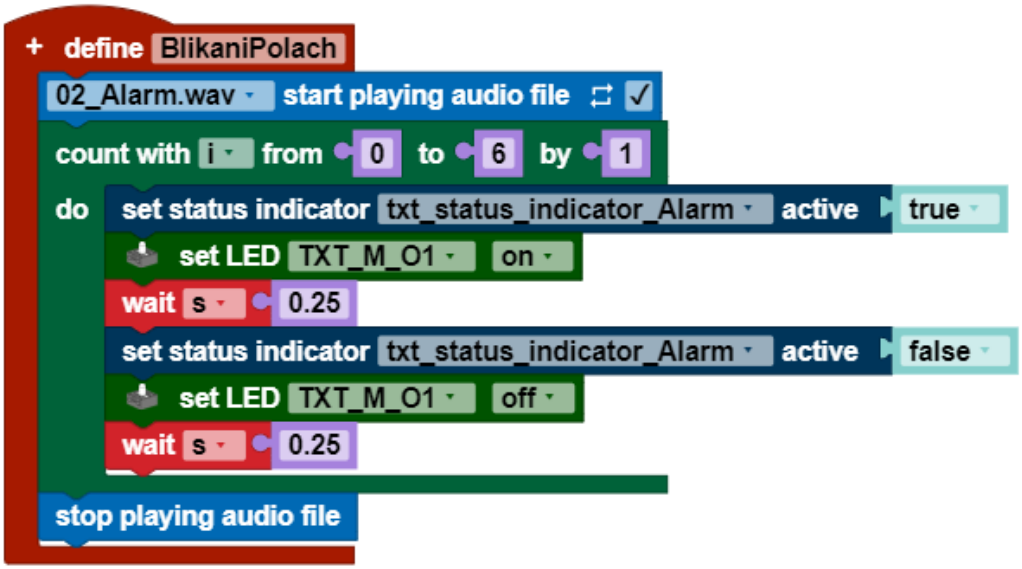
- Nastavíme si také konfiguraci displeje.






*Obrázek 3-62: Alarm – konfigurace displeje*

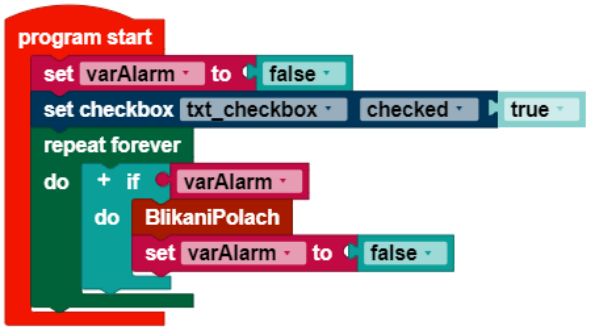
- Po nastavení kamery si vytvoříme funkci pro blikání LED a samotný program.




```
19
20  def BlikaniPolach():
21      global varAlarm, i
22      TXT_M.get_loudspeaker().play("02_Alarm.wav", True)
23      for i in range(7):
24          display.set_attr("txt_status_indicator.active", str(True).lower())
25          TXT_M_O1_led.set_brightness(512)
26          time.sleep(0.25)
27          display.set_attr("txt_status_indicator.active", str(False).lower())
28          TXT_M_O1_led.set_brightness(0)
29          time.sleep(0.25)
30      TXT_M.get_loudspeaker().stop()
```

*Obrázek 3-63: Alarm – funkce pro blikání LED*

- Hlavní program a deklarace události na detekci pohybu před kamerou:





```
1   import time
2
3   from lib.camera import *
4   from lib.controller import *
5   from lib.display import *
6
7
8   varAlarm = None
9   i = None
10
11
12  def motion_callback(event):
13      global varAlarm, i
14      if display.get_attr("txt_checkbox.checked"):
15          varAlarm = True
16
17  motion_detector.add_detection_listener(motion_callback)
18
19
20  def BlikaniPolach(): ···
31
32
33  varAlarm = False
34  display.set_attr("txt_checkbox.checked", str(True).lower())
35  while True:
36      if varAlarm:
37          BlikaniPolach()
38          varAlarm = False
39
```

*Obrázek 3-64: Alarm – program*

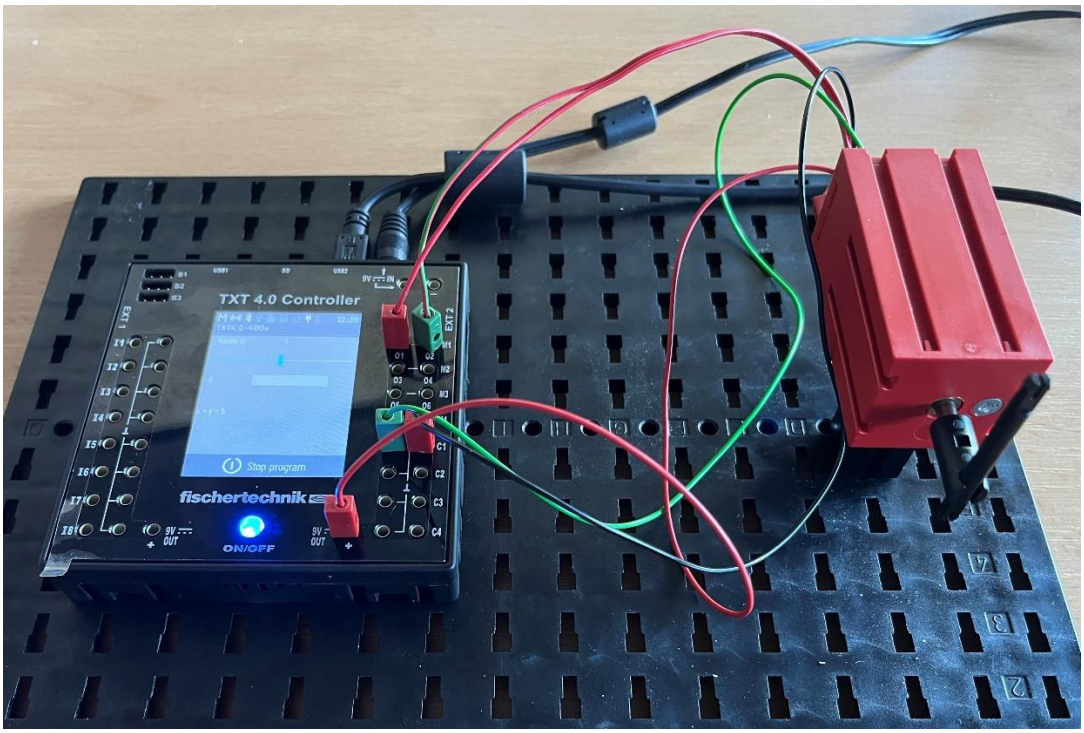
Nově je tu vložena komponenta pro přehrání zvuku. V tomto případě je zvolený zvuk 02-Alarm, který se opakuje, dokud bliká dioda a je zaškrtnuta možnost opakovaného přehrávání audia. Taktéž je nově použita událost, která nastane při detekci pohybu.

## 3.12 Hodiny

V tomto cvičení si ukážeme řízení krokového motoru s magnetickým snímačem pulsů.

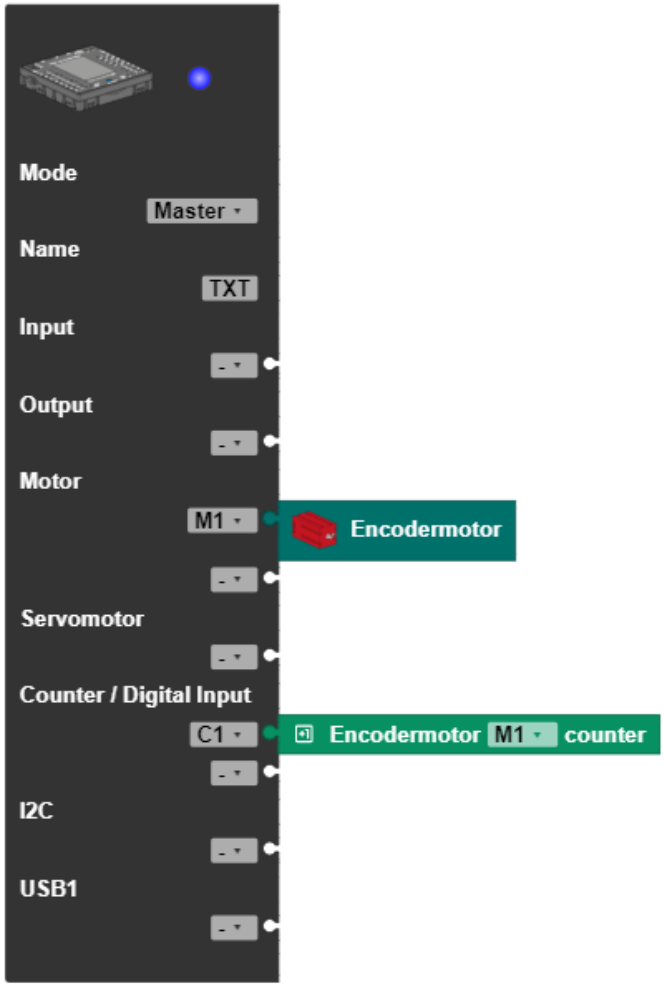
### 3.12.1 Hodiny – zapojení komponent

Zapojení komponent je následující, viz obrázek. Do TXT 4.0 Controlleru je zapojen čítač krokového motoru na +9V, zem a čítač rychlých pulsů C1, motorová část je připojena na výstup M1.




*Obrázek 3-65: Hodiny – zapojení krokového motoru*

**Konfigurace kontroléru:**




*Obrázek 3-66: Hodiny – konfigurace kontroléru*

### 3.12.2 Hodiny – řídící program

Jak chceme, aby se alarm choval…

1. Na displeji je možné softwarovými tlačítky zapnout nastavení ukazatele do poloh čtvrt/půl/třičtvrtě/celá rychlostí 150.
2. Po dosažení definované polohy se vrátí do původní pozice.
3. Bude možné pustit(na poloviční výkon) a vypnout neustálý běh motoru softwarovými tlačítky.

**Co v tomto případě potřebujeme vědět…**

Snímač použitého krokového motoru nám dovoluje poměrně přesně řídit natáčení osy motoru, případně synchronizovat více krokových motorů mezi sebou. Je potřeba vědět, že použitý motor v krokovém motoru dává 3 pulsy na otáčku. Protože je převodovaný v poměru 21:1, tak celkem dává přibližně 64 pulsů (3x21) na otočku výstupní osy krokového motoru.

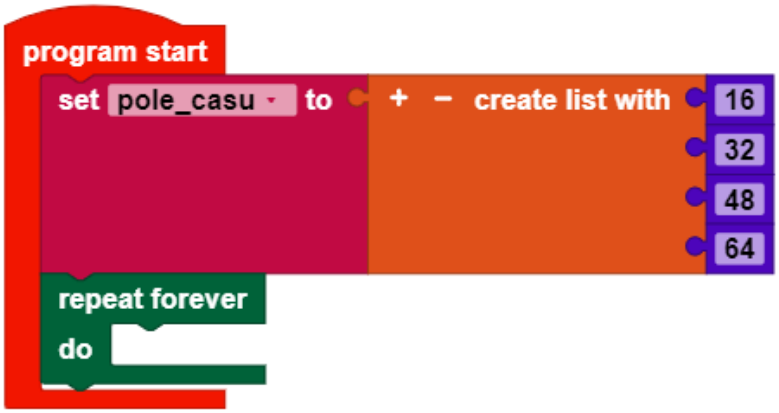
#### *3.12.2.1 Grafické rozhraní*

Grafické rozhraní obsahuje 1x label a 6x tlačítko.


*Obrázek 3-67: Hodiny – grafické rozhraní*

### *3.12.2.2 Hodiny – hlavní tělo programu*

V programu se definuje list se základními polohami ukazatele, tedy 16, 32, 48 a 64.

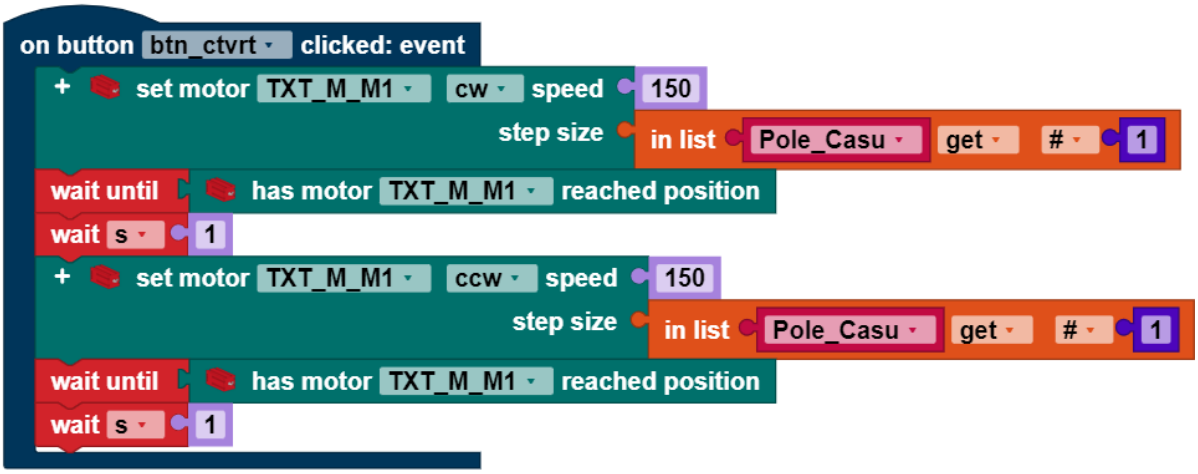



```
pole_casu = [16, 32, 48, 64]
while True:
  pass
```

*Obrázek 3-68: Hodiny – hlavní tělo programu*

### *3.12.2.3 Hodiny – event pro ukázku „čtvrt“*

Událost pro nastavení ukazatele do polohy čtvrt.



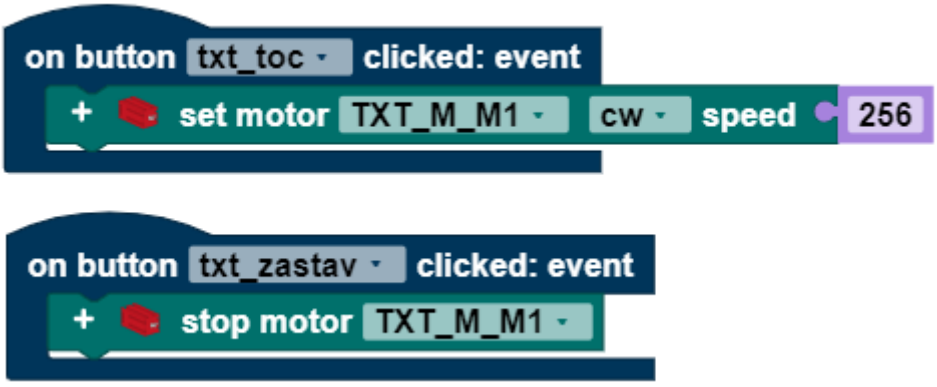

```
def on_btn_ctvrt_clicked(event):
    global Pole_Casu
    TXT_M_M1_encodermotor.set_speed(int(150), Motor.CW)
    TXT_M_M1_encodermotor.set_distance(int(Pole_Casu[0]))
    while True:
        if (not TXT_M_M1_encodermotor.is_running()):
            break
        time.sleep(0.010)
    time.sleep(1)
    TXT_M_M1_encodermotor.set_speed(int(150), Motor.CCW)
    TXT_M_M1_encodermotor.set_distance(int(Pole_Casu[0]))
    while True:
        if (not TXT_M_M1_encodermotor.is_running()):
            break
        time.sleep(0.010)
    time.sleep(1)
```

*Obrázek 3-69: Hodiny – event ukázky "čtvrt"*

Ostatní eventy pro ukázání půl/třičtvrtě/celá jsou jen upravenou variantou uvedeného eventu.

#### *3.12.2.4 Hodiny – eventy pro spuštění a zastavení motoru*

Události pro rozběhnutí motoru a zastavení pohybu motoru.




```
def on_txt_toc_clicked(event):
    global pole_casu
    TXT_M_M1_encodermotor.set_speed(int(256), Motor.CW)
    TXT_M_M1_encodermotor.start_sync()

def on_txt_zastav_clicked(event):
    global pole_casu
    TXT_M_M1_encodermotor.stop_sync()
```

*Obrázek 3-70: Hodiny – eventy pro spuštění a zastavení motoru*

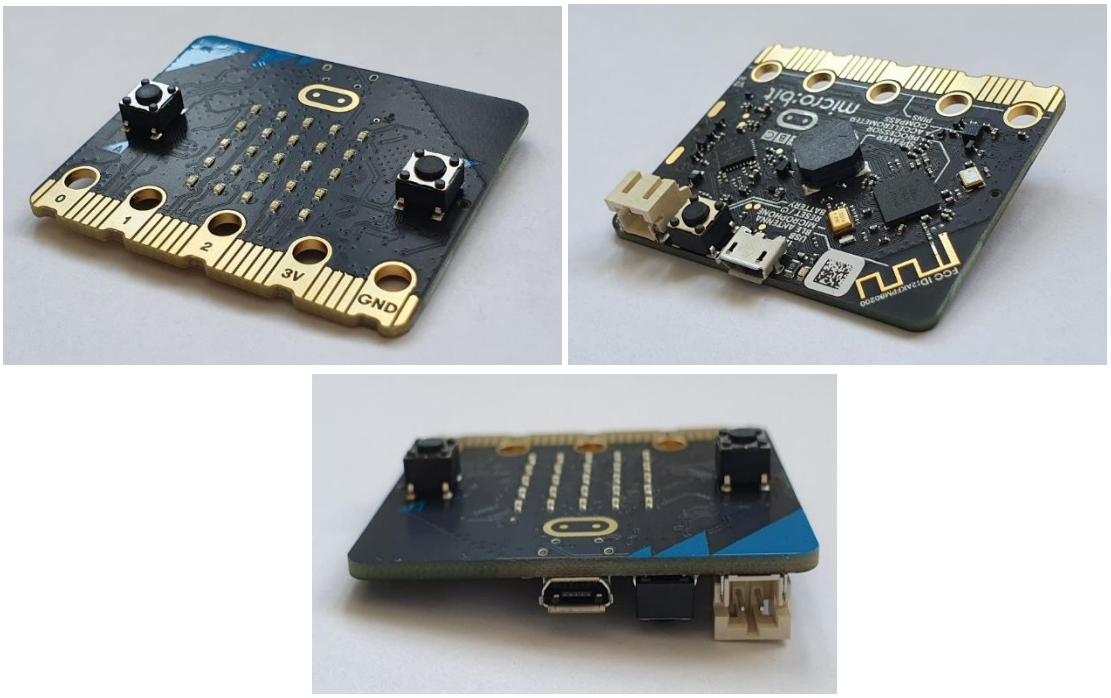
# 4 BBC micro:bit

BBC Micro:bit je malá řídící jednotka, jejíž první verze byla určena pro vzdělávání v oblasti informatiky a robotiky ve Velké Británii. Velmi rychle se stala populární po celém světě jak ro vzdělávání, tak např. pro využití různými bastlíři. Dnes pro tuto jednotku existuje velké množství rozhraní, které usnadňují její integraci např. se stavebnicemi Fischertechnik, Lego, Merkur atd. Ukázka řídící jednotky je na následujícím obrázku.

První verze označována jako V1 byla představena v roce 2016. V roce 2020 pak přišla nová generace označována jako V2, která má oproti V1 mikrofon, reproduktor, dotykové tlačítko v logu, větší paměť (RAM i ROM), výkonnější procesor a lehce upravený pinout Edge connectoru. V rámci tohoto textu a příkladů je využívána verze 2.

Základní technická specifikace řídící jednotky V2:

- Rozměry: 51,6 x 42 x 11,65 mm
- Procesor: Nordic nRF52833 s jádrem ARM Cortex-M4 (takt 64 MHz)
- RAM: 128 kB
- Vnitřní paměť: 512 kB typu flash
- Bezdrátová komunikace: Bluetooth 5.1
- Připojení: Micro USB 2.0
- Pracovní napětí 3,3V
- Napájecí napětí se může pohybovat v rozsahu 2 až 3,3 V
- Mezní rozsah napětí je 1,95 V až 3,6 V




*Obrázek 4-1: BBC Micro:bit*

## 4.1 Podrobný popis řídící jednotky BBC Micro:bit

Abychom mohli mikrokontrolér BBC Micro:bit naplno využívat, je potřeba se s ním důkladně seznámit 😊

### 4.1.1 Přední strana BBC Micro:bit

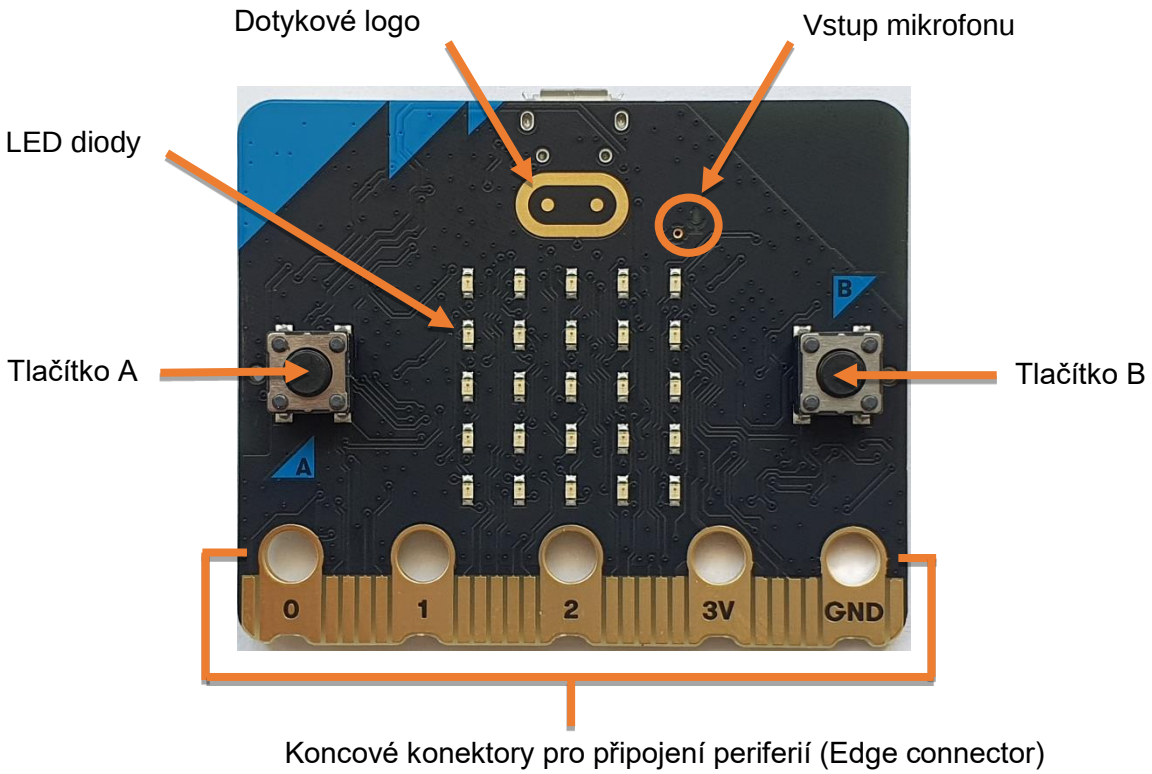
**Tlačítko A a B** – slouží uživateli ke spuštění naprogramované akce, která se skrývá pod stisknutím tlačítka, stisknutím a podržením, dvojitým stisknutím nebo stisknutím obou tlačítek naráz apod.

**LED diody** – jsou uspořádání do matice o rozměru 5 x 5 a svítí červeně. Uživatel tak může na této mini obrazovce vytvářet jednoduché hry, texty, signalizace... Je také možné nastavit intenzitu světla, které diody vyzařují. LED diody lze využít i pro měření množství světla v okolí řídící jednotky.

**Dotykové logo** – uživatel může poklepat nebo se jen dotknout dotykového loga a tím spustit naprogramovanou akci. Funguje podobně jako tlačítko jen s tím rozdílem, že je dotykové.

**Vstup mikrofonu –** slouží pro zachycení okolního zvuku, díky kterému je zařízení schopno vykonávat nejrůznější akce. Vedle mikrofonu je také LED indikátor, který svítí červeně v případě práce se zvukem.

**Koncové konektory pro připojení periferií** – celkově má Micro:bit 25 pinů, z toho pět velkých pinů které lze využít i jako dotyková tlačítka. Do pinu s označením 0, 1 a 2 je možné připojit vstupní / výstupní zařízení, a to nejčastěji pomocí krokosvorek. Pomocí pinu s označením 3V je možné napájet jiná zařízení jako elektromotor nebo jinou elektroniku. Popis Edge connectoru je na následujícím obrázku.




*Obrázek 4-2: BBC Micro:bit – popis zadní strany*

### 4.1.2 Zadní strana BBC Micro:bit

**Bluetooth anténa** – slouží k přijímání a odesílání signálů. Dokáže komunikovat s jinými micro:bity pomocí radiových vln a také s jinými zařízeními pomocí Bluetooth.

**Napájecí LED –** Pokud je micro:bit napájen, svítí červeně.

**Micro USB –** konektor pro propojení micro:bitu s počítačem a jeho napájení.

**USB LED –** bliká žlutě, pokud micro:bit komunikuje počítačem, tedy během přesouvání souborů z počítače do micro:bitu a obráceně.

**Resetovací tlačítko –** při stisknutí se restartuje program, který je nahraný v micro:bitu.

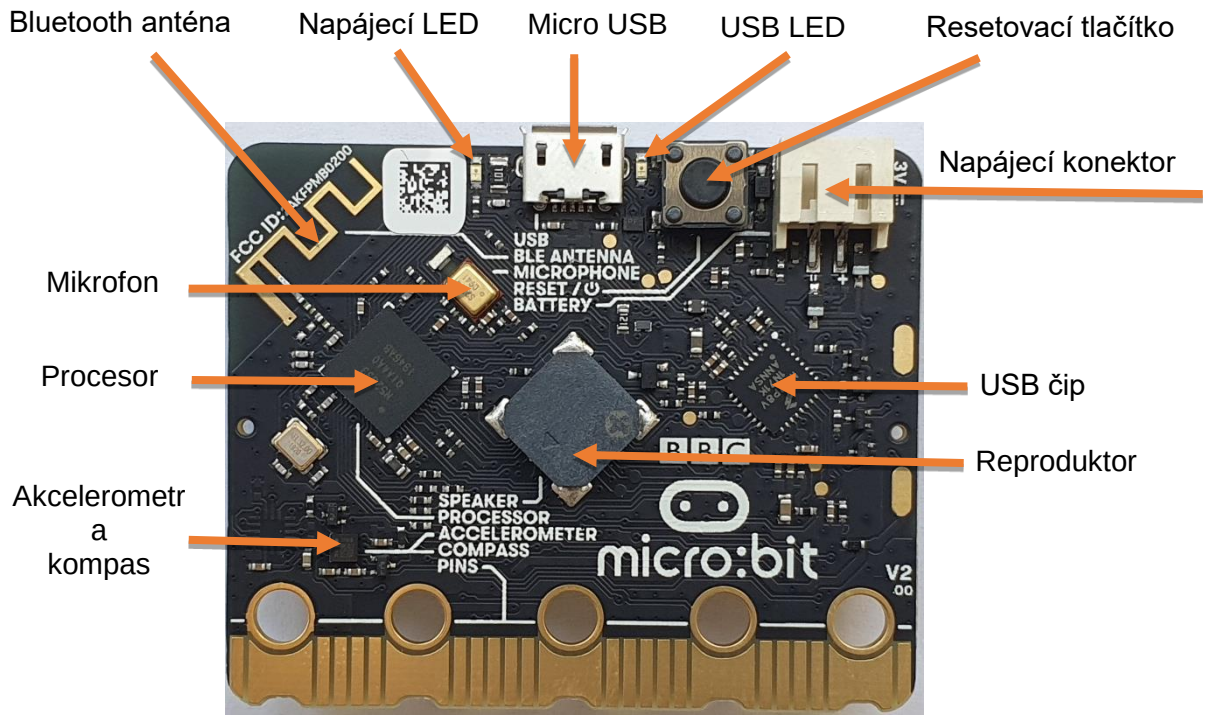
**Napájecí konektor –** slouží k napájení micro:bitu pomocí externího zdroje. Pak není nutné mít micro:bit připojený k počítači pomocí USB kabelu.

**USB čip –** umožňuje posílat data z počítače do micro:bitu a ukládání nového kódu.

**Reproduktor –** produkuje zvuk podle kódu programu.

**Akcelerometr a kompas –** akcelerometr měří zrychlení ve třech osách a funguje jako takový gyroskop. Kompas dokáže zjistit, kde se nachází sever a měří i sílu magnetického pole. Celkově funguje jako takový gyroskop.

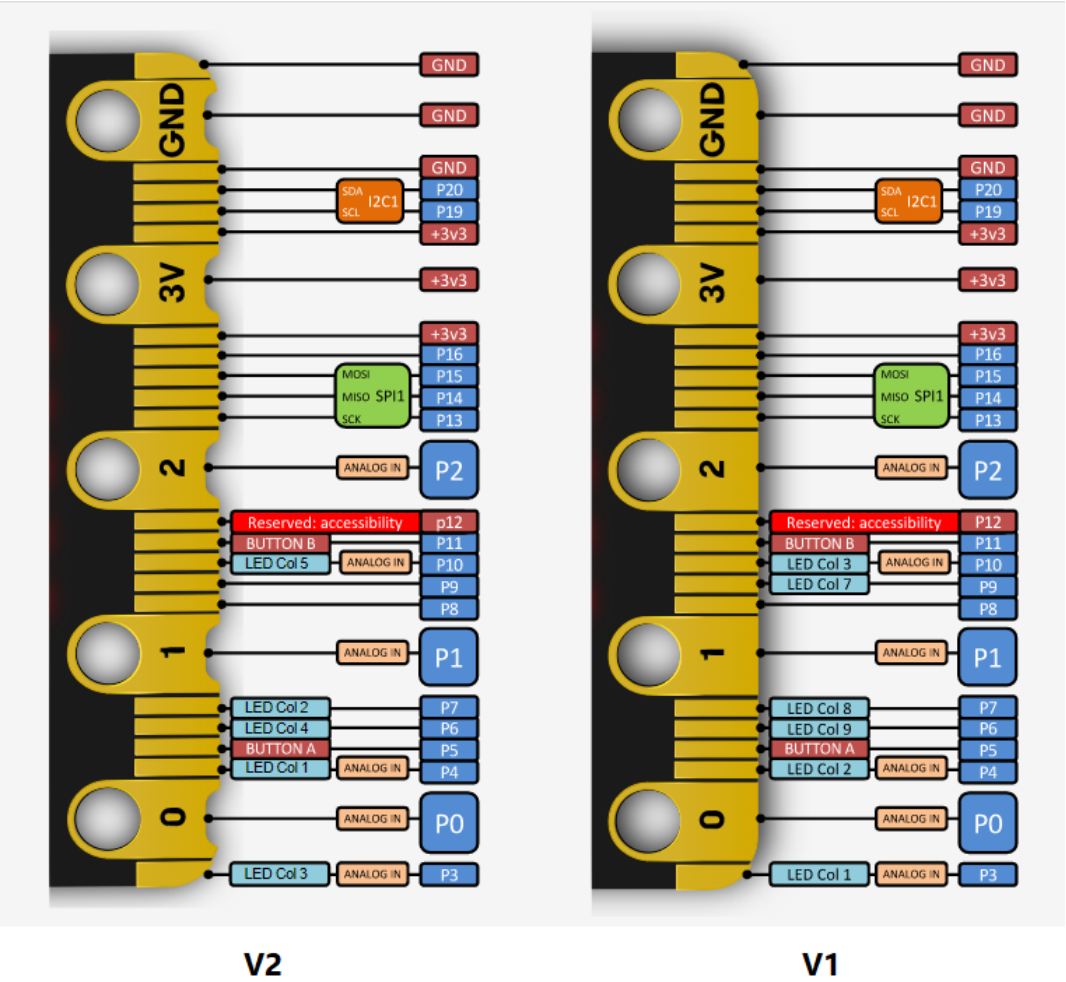
**Procesor –** provádí výpočty a logické operace. Je to takový mozek micro:bitu. Zároveň se v této oblasti nachází také senzor teploty.




*Obrázek 4-3: BBC Micro:bit – popis zadní strany*

### 4.1.3 Edge connector

Edge connector řídící jednotky obsahuje 20 pinů. Jejich funkcionalita se lehce liší mezi verzemi V1 a V2. Na následujícím obrázku jsou popsané piny pro obě verze.



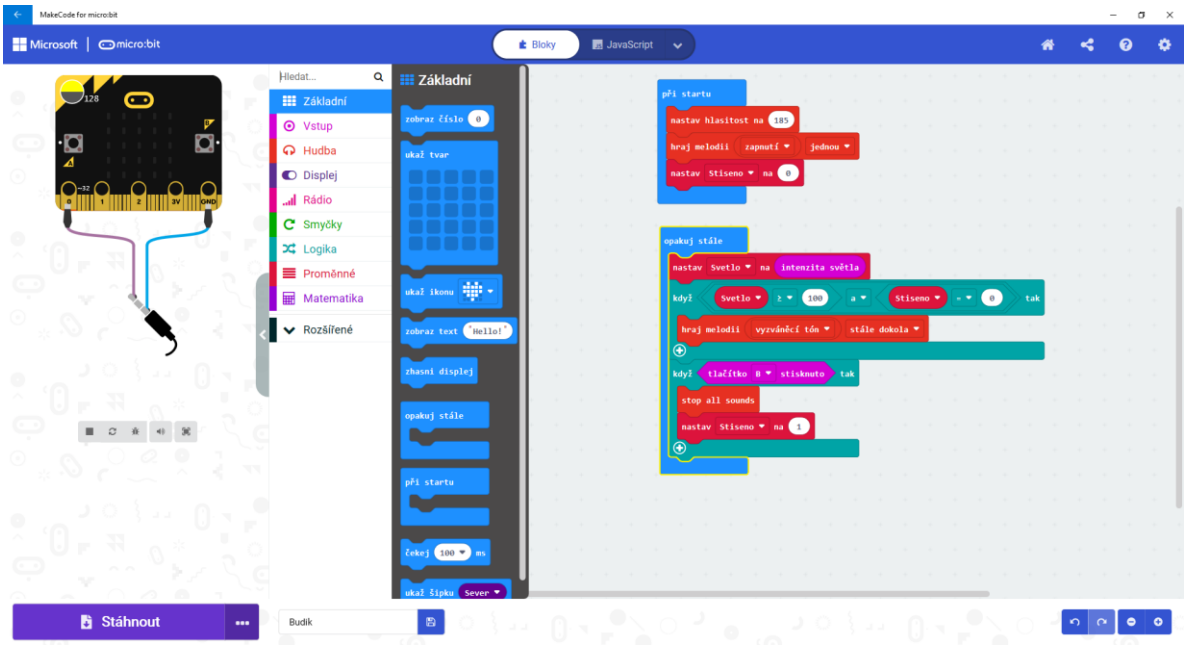

*Obrázek 4-4: BBC Micro:bit V1/V2 Edge connector pinout [3]*

## 4.2 Možnosti programování řídící jednotky BBC Micro:bit

Řídící jednotku lze programovat formou grafického programování ve webové aplikaci na stránkách micro:bitu (https://makecode.microbit.org/), popř. lze stáhnout aplikaci MakeCode z MS Store, nebo Apple store. Dostupné jsou oficiální i velké množství neoficiálních aplikací pro BBC Micro:bit pro iOS, iPad OS a Android. Další podporované možnosti/jazyky pro programování jednotky jsou:

- Python upravený přímo pro BBC Micro:bit, lze programovat ve webové verzi MakeCode
- Javascript, lze programovat ve webové verzi MakeCode
- standardní microPython, nejjednodušeji ve vývojovém prostředí Mu, nebo Thonny, dokumentace na https://microbit-micropython.readthedocs.io/en/v2-docs/
- Wiring, ve vývojovém prostředí pro Arduina
- C/C++

Ukázka vývojového prostředí MakeCode je na následujícím obrázku.




*Obrázek 4-5: Vývojové prostředí MakeCode pro BBC Micro:bit*

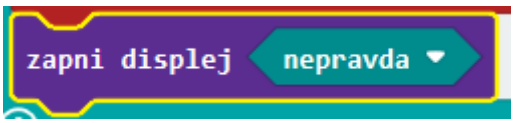
# 5 Digitální I/o adaptér Fischertechnik pro BBC Micro:bit

Adaptér v tomto případě slouží pro 2 věci:

1. Zjednodušuje fyzické propojení vstupů a výstupů řídící jednotky s postaveným modelem, který chceme pomocí řídící jednotky ovládat.
2. Zajišťuje oboustranný převod mezi úrovněmi napětí, kdy BBC Micro:bit funguje přibližně na 3V, komponenty stavebnice Fischertechnik na 9V a zároveň umožňuje napájení stavebnice a řídící jednotky. V tomto případě tak USB kabel může sloužit pouze k přenosu řídícího programu do řídící jednotky a po nahrání může být USB od řídící jednotky odpojeno.

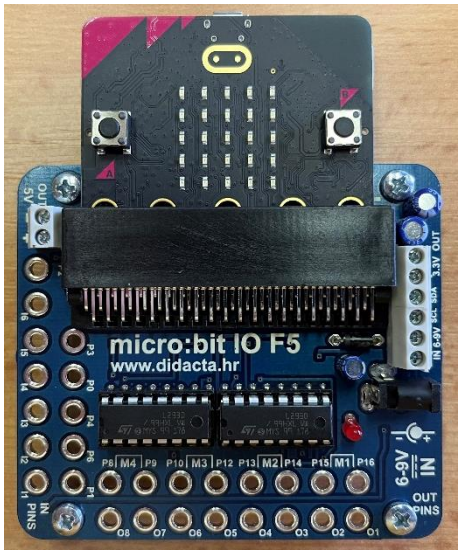
**POZOR!!!**

**Pokud se používají na adaptéru piny P3, P4, P6, P9 a P10, tak nepoužívat display na BBC Micro:bit! Jedná se o sdílené piny! Je vhodné jej i deaktivovat blokem „zapni displej“, viz obrázek.**



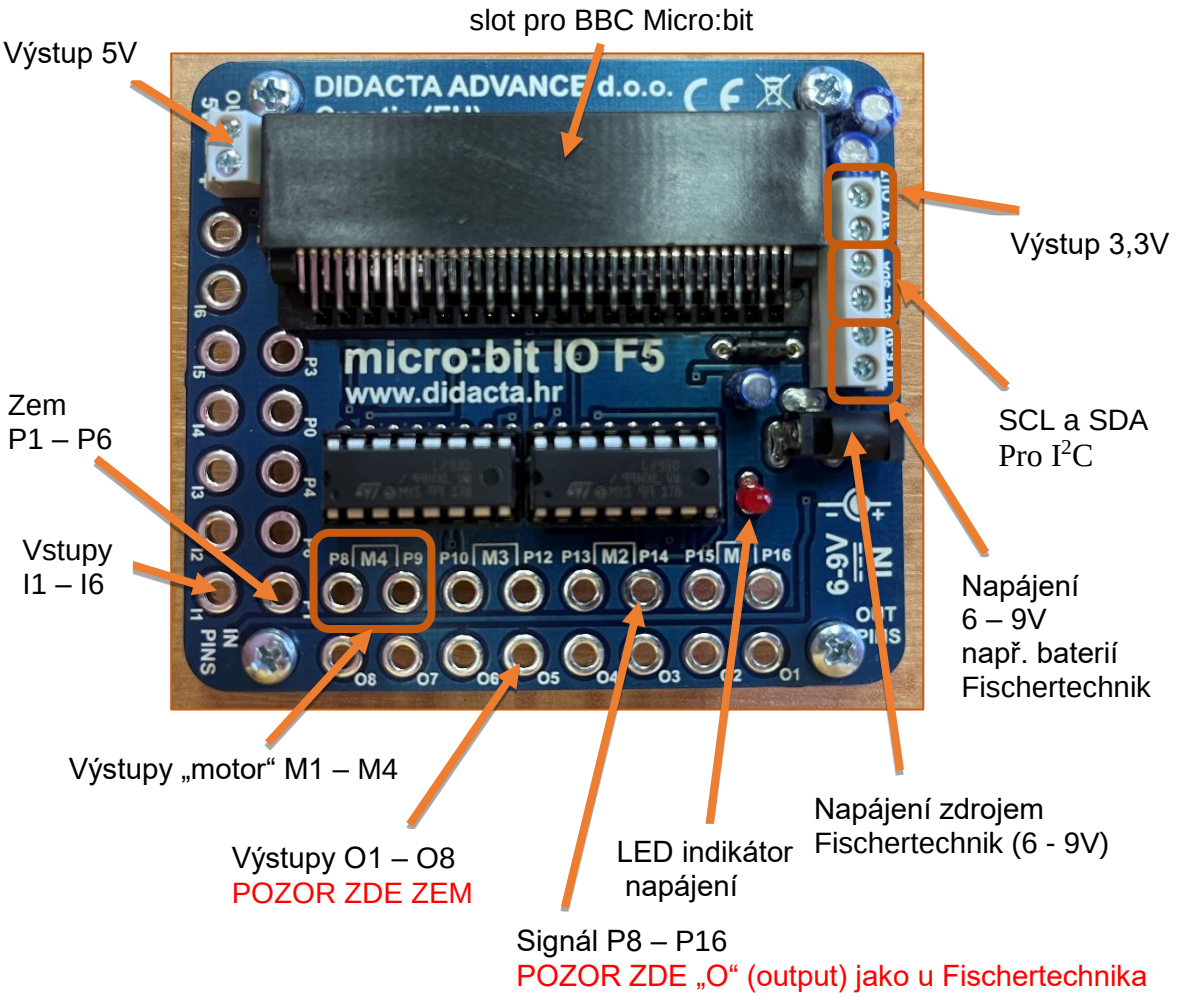

*Obrázek 5-1: BBC Micro:bit – blok pro deaktivaci/aktivaci displeje*

Existují varianty adaptéru i pro další řídící jednotky, jako je RasbperryPi, Arduino UNO, Arduino MEGA a další. Ukázka zapojeného BBC Micro:bit je na následujícím obrázku.




*Obrázek 5-2: BBC Micro:bit s digitálním I/o adaptérem Fischertechnik*

Popis jednotlivých částí adaptéru je na následujícím obrázku.




*Obrázek 5-3: Popis digitálního I/O adaptéru Fischertechnik pro BBC Micro:bit*

# 6 Ukázkové příklady pro BBC Micro:bit

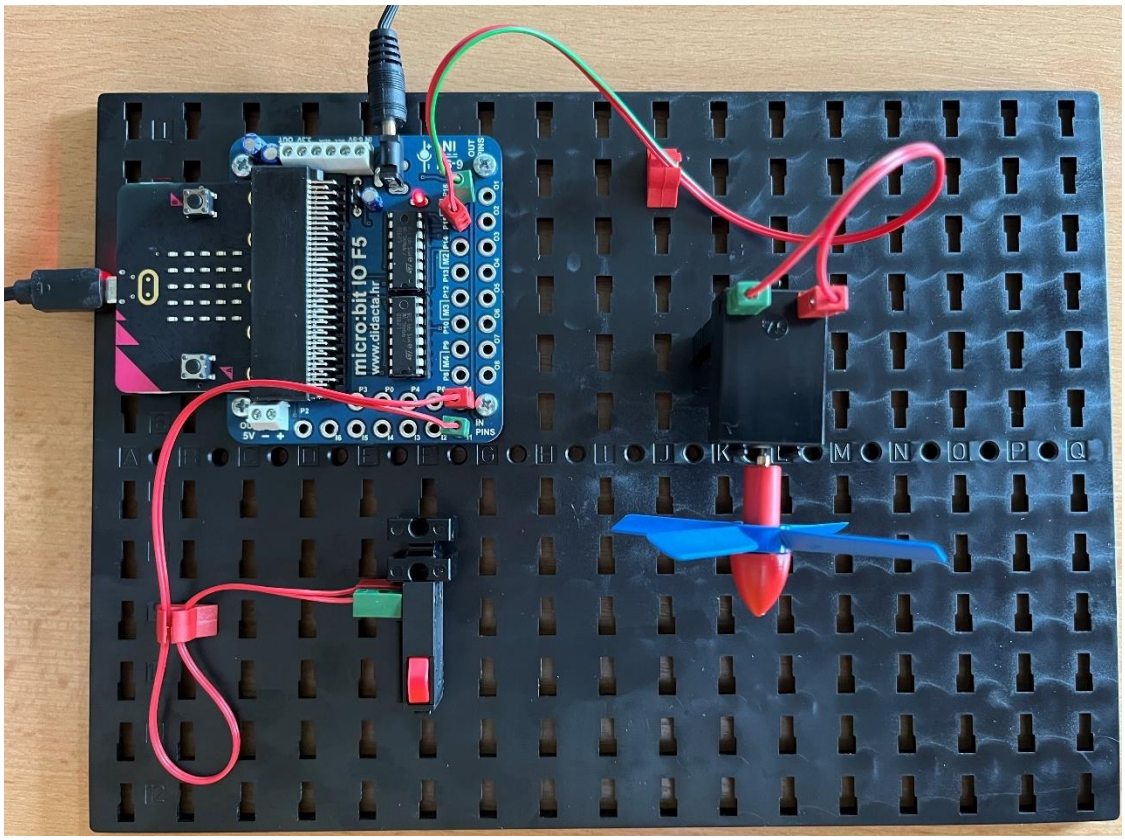
Ukázkové příklady pro BBC Micro:bit jsou tvořeny v MakeCode grafickým způsobem. Ukázané pythony kódy jsou pouze automaticky generované z grafického kódu.

## 6.1 Větráček s BBC Micro:bit

Větráček je velmi jednoduché zařízení s jedním spínačem a motorem, který otáčí vrtulí. Pro sestavení tedy potřebujeme právě tyto dvě komponenty a řídící jednotku.

### 6.1.1 Větráček – zapojení komponent

Zapojení je v tomto případě triviální, na vstup I1/P1 je přiveden spínač, na výstup M1 je přivedený motor větráčku, viz následující obrázek.

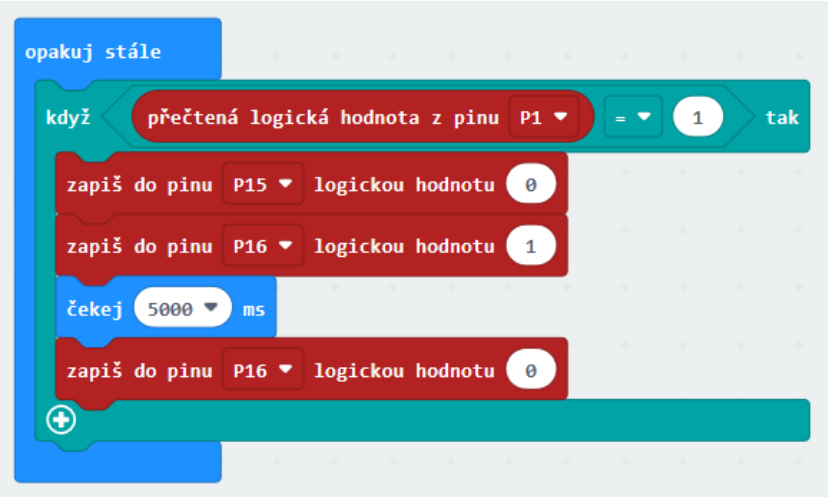



*Obrázek 6-1: Větráček s BBC Micro:bit – zapojení komponent*

### 6.1.2 Program pro řízení větráčku

Cílem cvičení je oživit model takovým způsobem, aby po stisknutí tlačítka se větráček roztočil na maximální možnou rychlost, běžel 5 vteřin a po této době se vypl a čekal na další stisknutí tlačítka. Po příkladech s Robo Pro Coding si řídící kód není potřeba rozebírat.

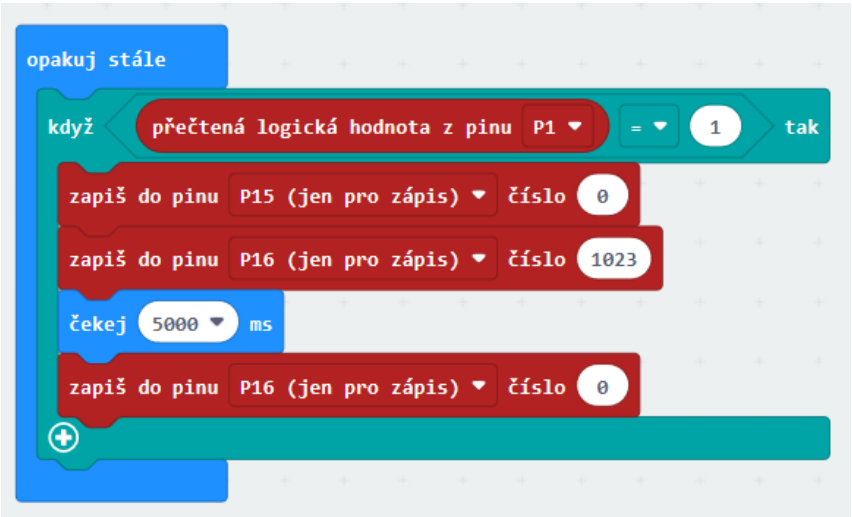
**Varianta 1**: Je použito digitálního zápisu/čtení do pinů pro připojený motor. Zapisujeme tedy 0/1. To znamená, že motor bude buď stát, nebo se roztočí na maximum. Nelze řídit výkon.




```
def on_forever():
    if pins.digital_read_pin(DigitalPin.P1) == 1:
        pins.digital_write_pin(DigitalPin.P15, 0)
        pins.digital_write_pin(DigitalPin.P16, 1)
        basic.pause(5000)
        pins.digital_write_pin(DigitalPin.P16, 0)

basic.forever(on_forever)
```

*Obrázek 6-2: Větráček s BBC Microbit, digitální hodnoty pro motor*

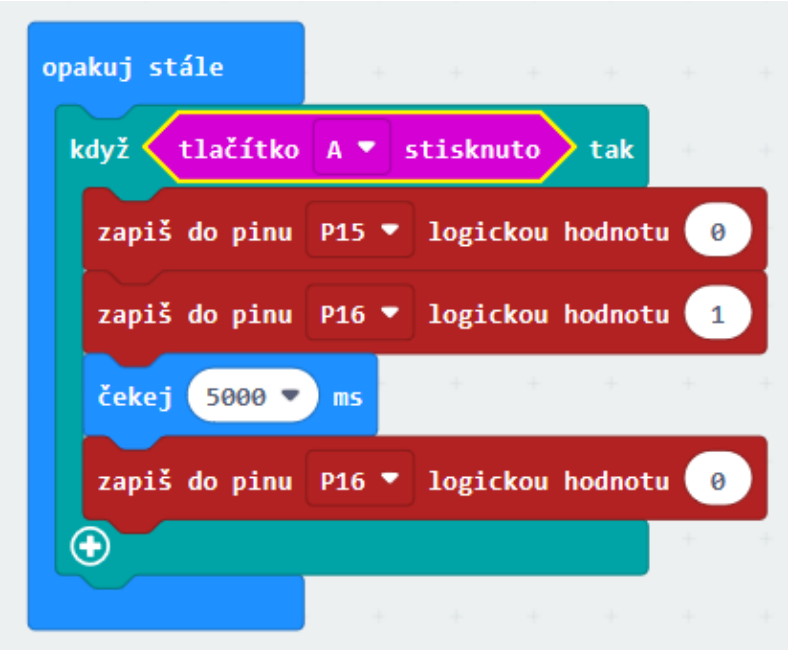
**Varianta 2**: Je použito analogového zápisu/čtení do pinů pro připojený motor. Zapisujeme tedy rozsah 0–1023 a v tomto rozsahu můžeme regulovat rychlost otáček/výkon motoru, hodnota 1023 je maximální výkon.




```
def on_forever():
    if pins.digital_read_pin(DigitalPin.P1) == 1:
        pins.analog_write_pin(AnalogPin.P15, 0)
        pins.analog_write_pin(AnalogPin.P16, 1023)
        basic.pause(5000)
        pins.analog_write_pin(AnalogPin.P16, 0)

basic.forever(on_forever)
```

*Obrázek 6-3: Větráček s BBC Microbit, digitální hodnoty pro motor*

**Varianta 3**: Je použito analogového zápisu/čtení do pinů pro připojený motor. Zapisujeme tedy logické hodnoty 0/1 a nemůžeme tak regulovat rychlost otáček/výkon motoru, hodnota 1 je maximální výkon. Místo tlačítka z modelu se používá tlačítko A na BBC Micro:bit.




```
def on_forever():
    if input.button_is_pressed(Button.A):
        pins.digital_write_pin(DigitalPin.P15, 0)
        pins.digital_write_pin(DigitalPin.P16, 1)
        basic.pause(5000)
        pins.digital_write_pin(DigitalPin.P16, 0)

basic.forever(on_forever)
```

*Obrázek 6-4: Větráček s BBC Microbit, digitální hodnoty pro motor spouštěný tlačítkem na BBC Micro:bit*

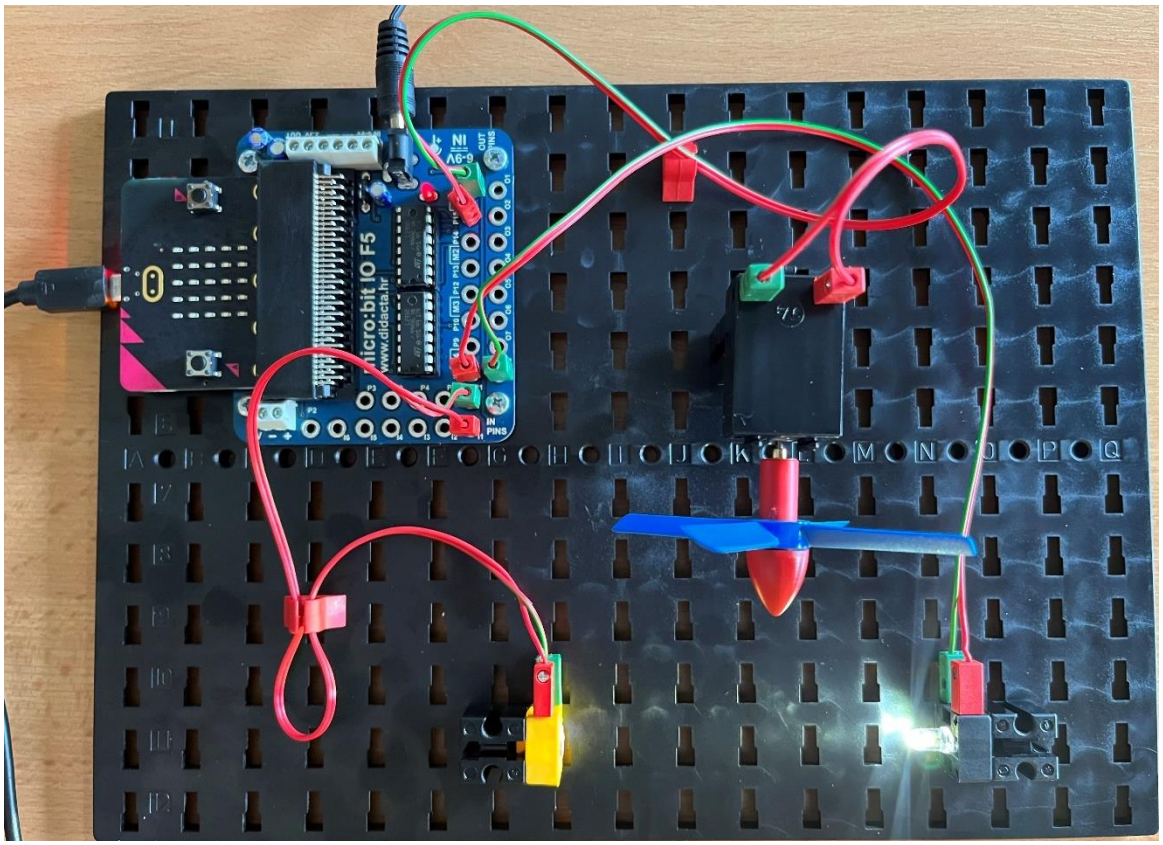
## 6.2 Vysoušeč rukou s BBC Micro:bit

V tomto cvičení využijeme části předchozích cvičení. Úkolem je vytvořit vysoušeč rukou, tedy zařízení, kde je světelná brána rozhodující o spuštění/nespuštění ventilátoru. Na cvičení se v tomto případě pouze lehce modifikuje model využitý v předchozím cvičení.

Světelná brána je využití komponenty emitující světlo (v tomto případě např. žárovka, nebo LED) na jedné straně a fototranzistor, který vyhodnocuje, zda na něj emitované světlo ze světelné komponenty dopadá, nebo ne, na straně druhé. Pokud světlo na fototranzistor dopadá, je větrák v klidu, pokud ne, běží na maximum.

### 6.2.1 Vysoušeč rukou – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstup I1/P1 je přivedený fototranzistor (světelný senzor), na výstup M1 je přivedený motor s vrtulí a na výstup P8/O8 je přivedené světlo, v tomto případě LED, tedy na P8 je přivedeno „+“.

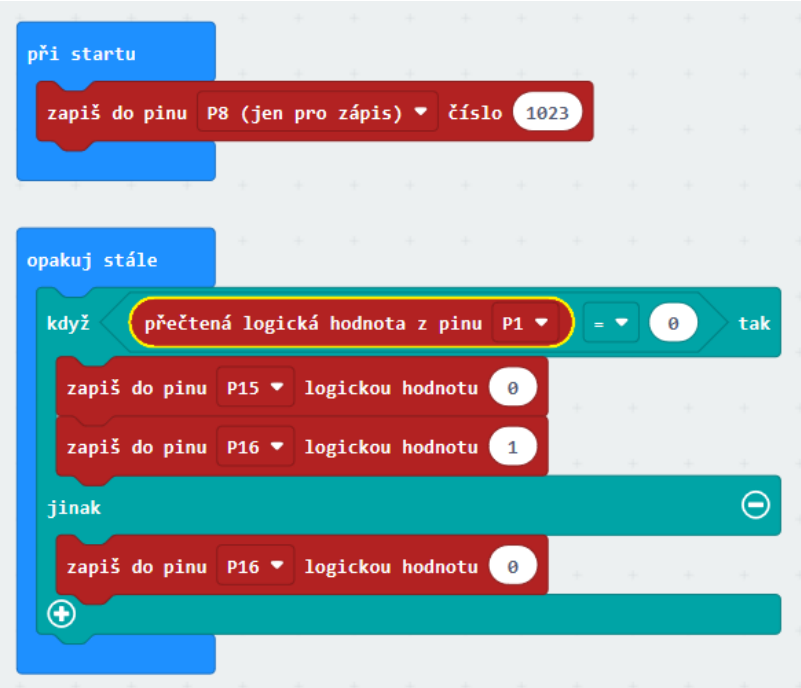



*Obrázek 6-5: Vysoušeč rukou s BBC Micro:bit – zapojení komponent*

### 6.2.2 Vysoušeč rukou – řídící program

Jak chceme, aby se vysoušeč choval…

1. Pokud není mezi LED komponentou a fototranzistorem překážka, tak je vysoušeč v klidu.
2. Pokud se mezi LED komponentu a fototranzistor vloží překážka (ruce), vysoušeč se spustí a bude vysoušet ruce. Na vyklizení prostoru pro ruce ihned reaguje vypnutím.
3. Program bude v cyklu, tedy kdykoliv se vloží překážka mezi LED a fototranzistor, tak se vysoušeč opětovně zapne.




```
pins.analog_write_pin(AnalogPin.P8, 1023)

def on_forever():
    if pins.digital_read_pin(DigitalPin.P1) == 0:
        pins.digital_write_pin(DigitalPin.P15, 0)
        pins.digital_write_pin(DigitalPin.P16, 1)
    else:
        pins.digital_write_pin(DigitalPin.P16, 0)
basic.forever(on_forever)
```

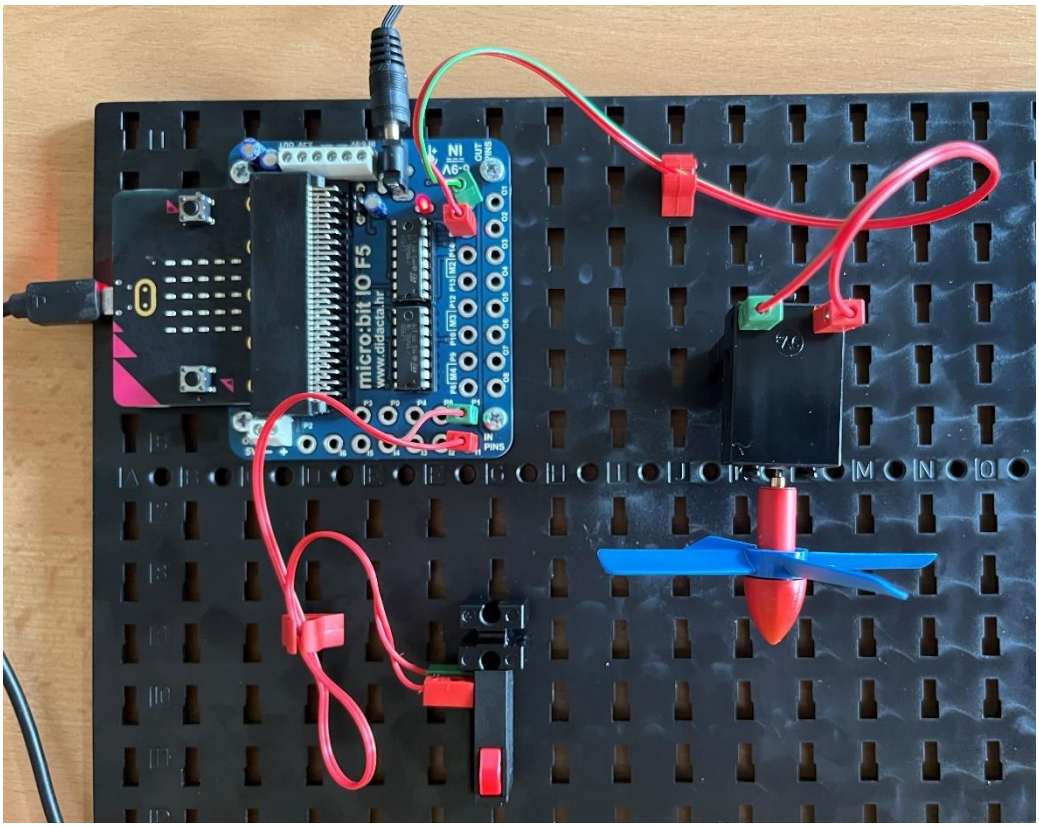
*Obrázek 6-6: Vysoušeč rukou s BBC Micro:bit – řídící program se světelnou bránou*

## 6.3 Logický motor s BBC Micro:bit

V tomto cvičení se vrátíme k větráku a ukážeme si, jak využívat logické operátory.

### 6.3.1 Logický motor – zapojení komponent

Zapojení komponent je následující, viz obrázek. Na vstup I1/P1 je přivedený spínač. Na výstup M1 je přivedený motor.




*Obrázek 6-7: Logický motor s BBC Micro:bit – zapojení komponent*

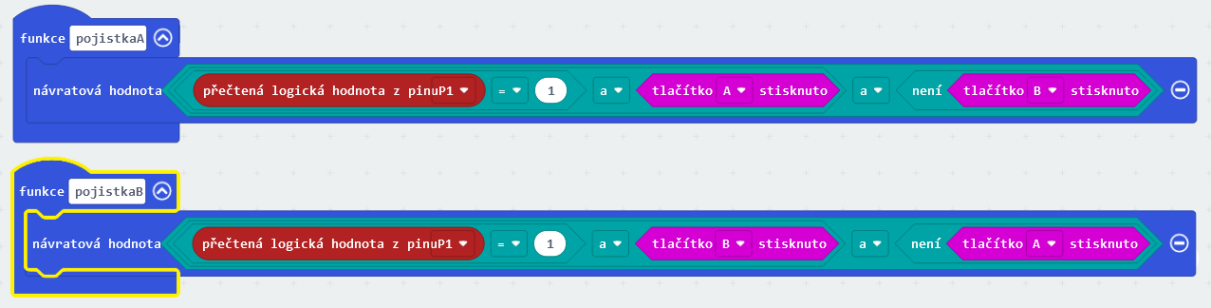
### 6.3.2 Logický motor – řídící program

Jak chceme, aby se logický motor choval…

1. Při stisknutí pouze jednoho, jakéhokoliv tlačítka se nic neděje.
2. Tlačítko I1/P1 na modelu je bezpečnostní
3. Tlačítko A na řídící jednotce spouští otáčky ve směru hodinových ručiček. Podmínka je, že k němu musí být stisknuto i tlačítko na modelu připojené na I1/P1.
4. Tlačítko B na řídící jednotce spouští otáčky v protisměru hodinových. Podmínka je, že k němu musí být stisknuto i tlačítko na modelu připojené na I1/P1.
5. Při stisknutí všech 3 tlačítek se nesmí stát nic a musíme to ošetřit, abychom se nesnažili poslat napětí na P15 i P16, které dohromady tvoří M1. Mohli bychom tím ublížit motoru, řídící jednotce nebo adaptéru.

Praktické využití je např. bezpečné spouštění stroje, kdy chceme mít jistotu, že obsluha má obě ruce mimo a nehrozí tak její poranění. Ukázka funkčnosti modelu viz video Logický motor.

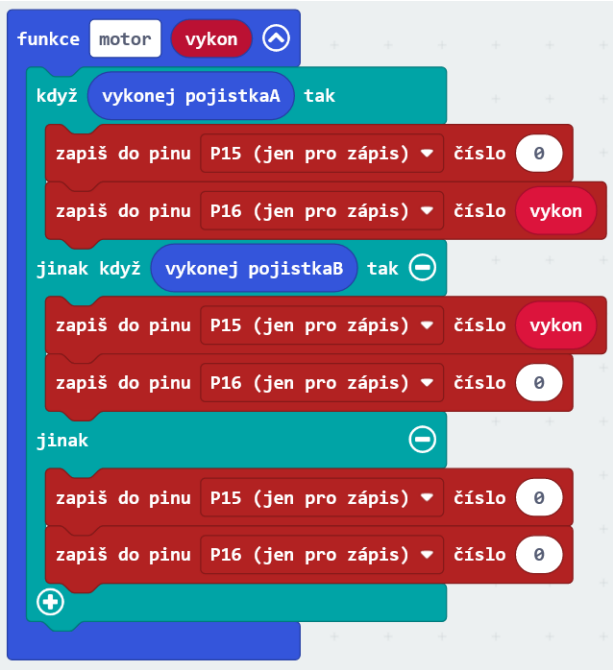
### Funkce pro vyhodnocování stisknutých tlačítek




```
def pojistkaA():
    return pins.digital_read_pin(DigitalPin.P1) == 1 and input.button_is_pressed(Button.A) and not (input.button_is_pressed(Button.B))
def pojistkaB():
    return pins.digital_read_pin(DigitalPin.P1) == 1 and input.button_is_pressed(Button.B) and not (input.button_is_pressed(Button.A))
```

*Obrázek 6-8: Logický motor s BBC Micro:bit – funkce pro vyhodnocování tlačítek*

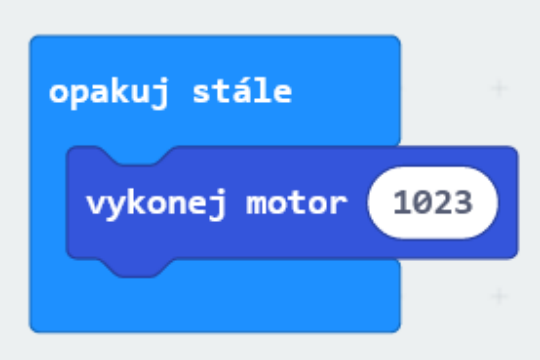
### Funkce pro ovládání motoru s předáváním hodnoty požadovaného výkonu




```
def motor(vykon: number):
    if pojistkaA():
        pins.analog_write_pin(AnalogPin.P15, 0)
        pins.analog_write_pin(AnalogPin.P16, vykon)
    elif pojistkaB():
        pins.analog_write_pin(AnalogPin.P15, vykon)
        pins.analog_write_pin(AnalogPin.P16, 0)
    else:
        pins.analog_write_pin(AnalogPin.P15, 0)
        pins.analog_write_pin(AnalogPin.P16, 0)
```

*Obrázek 6-9: Logický motor s BBC Micro:bit – funkce pro ovládání motoru*

**Hlavní program**




```
def on_forever():
    motor(1023)

basic.forever(on_forever)
```

*Obrázek 6-10: Logický motor s BBC Micro:bit – hlavní program*

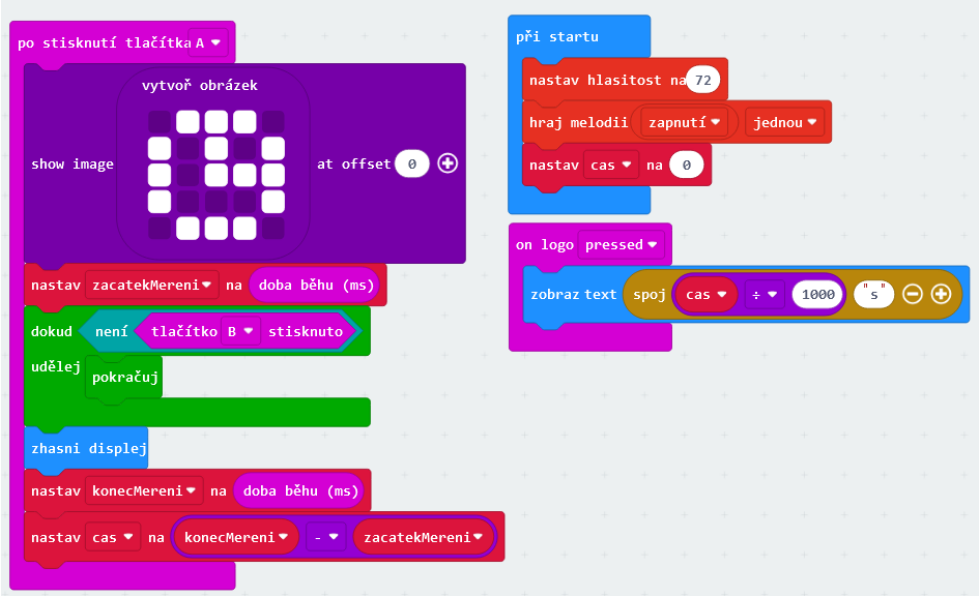
## 6.4 Stopky

V tomto cvičení se zaměříme na BBC Micro:bit. Není potřeba připojovat žádný model.

### 6.4.1 Stopky – program

Jak chceme, aby se stopky chovaly…

1. Po nahrání a spuštění programu na BBC Micro:bit nám zahraje znělka, která nás informuje že je vše připraveno.
2. Po stisku tlačítka A na BBC Micro:bit začne běžet časomíra a o tom nás po celou dobu informuje symbol hodin na displeji (LED matice).
3. Po stisku tlačítka B na BBC Micro:bit se časomíra zastaví.
4. Po stisknutí dotykového tlačítka (loga BBC Micro:bit) se zobrazí naměřená hodnota na displeji (LED matice).
5. Stopovat čas bude možné opakovaně.




```
def on_button_pressed_a():
    global zacatekMereni, konecMereni, cas
    images.create_image("""
        . # # # .
                # . # . #
                # . # # #
                # . . . #
                . # # # .
    """).show_image(0)
    zacatekMereni = input.running_time()
    while not (input.button_is_pressed(Button.B)):
        continue
    basic.clear_screen()
    konecMereni = input.running_time()
    cas = konecMereni - zacatekMereni
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_logo_pressed():
    basic.show_string("" + str(cas / 1000) + "s")
input.on_logo_event(TouchButtonEvent.PRESSED, on_logo_pressed)

konecMereni = 0
zacatekMereni = 0
cas = 0
music.set_volume(72)
music.start_melody(music.built_in_melody(Melodies.POWER_UP), MelodyOptions.ONCE)
cas = 0
```

*Obrázek 6-11: Stopky s BBC Micro:bit – řídící program*

# 7 Seznam obrázků

# 8 Seznam tabulek

# 9 Reference

[1] fischertechnik, „ROBOTICS TXT 4.0 CONTROLLER,“ 10 11 2021. [Online]. Available: https://www.fischertechnik.de/en/service/elearning/teaching/txt-40-controller. [Přístup získán 24 08 2022].

[2] fischertechnik, „ROBOTICS TXT Electropneumatic,“ [Online]. Available: https://www.fischertechnik.de/en/service/elearning/playing/txt-electropneumatic. [Přístup získán 16 09 2019].

[3] micro:bit pins, „Microsoft MakeCode for micro:bit,“ Copyright © 2022 Microsoft, [Online]. Available: https://makecode.microbit.org/device/pins. [Přístup získán 9 30 2022].