# PRACOVNÍ LIST – POLE, CYKLUS FOR

V TÉTO ČÁSTI BUDETE POKRAČOVAT V PROGRAMOVÁNÍ SOUSTAVY LED DIOD. NAUČÍTE SE OPTIMALIZOVAT PROGRAMOVÝ KÓD POMOCÍ PROGRAMOVÝCH STRUKTUR JAKO JE POLE A CYLUS FOR.

## CO SE NAUČÍTE

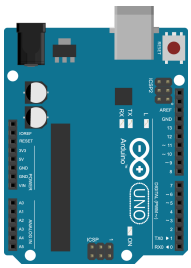
① Definovat pole pro Arduino.
② Využívat pole při definici pinů.
③ Programovat cyklus **`For`**.

## CO BUDETE POTŘEBOVAT

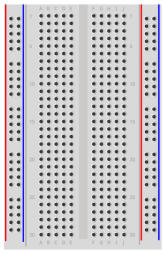
① LED diodu - 8x.
② Rezistor 220Ω – 8x.
③ Desku Arduino.
④ Kontaktní pole.
⑤ Vodiče typu samec-samec.

RGB led – 8 kusů

Rezistor 220Ω – 8 kusů



Deska Arduino



Kontaktní pole

POUŽITÉ SOUČÁSTKY

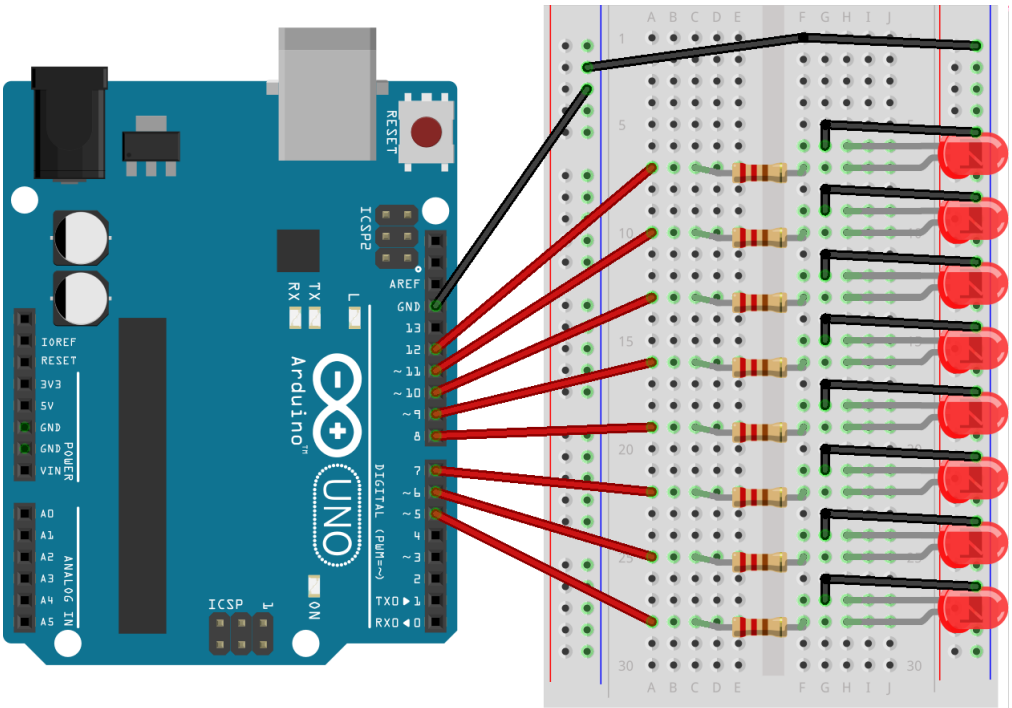
## A JDĚTE NA TO …

① Pokud máte složený elektronický obvod z minulé hodiny, můžete se pustit rovnou do programování. V opačném případě obvod musíte opět složit podle přiloženého schématu.

**DEJTE SI POZOR**

- **Pozor si dejte na to, jak zapojujete LED diody. Delší vývod musí být připojen přes rezistor k pinu. Kratší vývod je připojen na zem (pin GND).**
- **Dejte si pozor na hodnotu rezistoru. Zkontrolujte si, že je barevně označen v pořadí červená, červená, modrá černá, zlatá.**
- **Všimněte si, jak je zapojen vodič zemnění. Pro přehlednost je veden na druhou stranu kontaktního pole. Následně je zemnění vedeno ke každé LED diodě zvlášť (černý vodič).**

② Otevřete programový kód z minulé hodiny, kde je definována vlastní funkce pro jezdící světlo.



## DEKLARACE POLÍ

Deklarace polí lze provádět několika způsoby:

```
int myInts[6];
int myPins[] = {2, 4, 8, 3, 6};
int mySendVals[6] = {2, 4, -8, 3, 2};
char message[6] = "hello";
```

① **`myInt[6]`** – deklarace pole bez jeho inicializace.

② **`myPins[]`** – deklarace pole bez uvedení jeho konkrétní velikosti, kompilátor nejdříve spočítá prvky a pak pole vytvoří o odpovídající velikosti.

③ **`mySendVals[6]`** – deklarace pole s jeho přesným počtem prvků.

④ **`message[6]`** – u pole datového typu **`char`** musí být provedena jeho inicializace minimálně jedním prvkem.

## PŘÍSTUP K POLI

Prvky v poli jsou indexovány vždy od nuly. Tzn. první prvek v poli je na indexu 0. To znamená, že pole o deseti prvcích má poslední prvek s indexem 9.

```
// přístup k prvkům od nultého indexu
mySendVals[0] == 2;  // index 0 tj. první prvek
mySendVals[1] == 4;  // index 1 tj. druhý prvek
mySendVals[2] == -8; // atd.

// přidělení hodnoty pole
mySendVals[0] = 12;  // na index 0 bude přidělena hodnota 12

// získání hodnoty z pole
x = mySendVals[4];  // do x se uloží hodnota z indexu 4 tj. 2
```

### OTÁZKY PRO VÁS

- **Když se podíváte na otevřený programový kód, jak byste v něm využili pole?**
- **V čem byste spatřovali výhodu ve využití pole?**

### ÚKOLY PRO VÁS

- **A) Upravte otevřený program s vlastní funkcí tak, aby čísla pinů byla nahrazena odkazem na prvky pole.**

**B) Změňte směr běžícího světla z opačné strany.**

## CYKLUS FOR

Cyklus **`for`** se používá k opakování bloku příkazů, uzavřených do složených závorek. Využívá čítače inkrementů (přírůstků), který se používá pro ukončení průchodu cyklu. Cyklus **`for`** je vhodný pro jakékoliv opakující se operace a je často používán v kombinaci s **poli**, pro jejich průchod.

## SYNTAXE

```
for (inicializace; podmínka; přírůstek) {
      // blok příkazů
}

// praktická ukázka
void loop()
{
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

V první řadě nastane **`inicializace`**, a to minimálně jednou. V každém průchodu je testována **`podmínka`**. Pokud podmínka nabude hodnoty **`True`**, provede se blok příkazů

a zvětší se **`přírůstek`**. Podmínka je opět testována. Když podmínka nabude hodnoty **`False`**, smyčka se ukončí.

### OTÁZKA PRO VÁS

➔ **Které části programového kódu by se hodilo využít v cyklu for, aby se kód zjednodušil?**

### ÚKOLY PRO STUDENTY

➔ **C) Předchozí úkol, ve kterém jste čísla pinů nahradili prvky pole, upravte tak, abyste použili příkaz cyklu for a světlo diod probíhalo z jedné strany na druhou, neustále dokola.**

➔ **D) Upravte programový kód tak, aby se běžící světlo pohybovalo z jedné strany na druhou a zpět.**