0.1 CONDITIONAL STATEMENTS

Before we dive into S-R-A Loop lets take a first look to IF-statemen. `IF-statement` allows us to execute some code when the condition is **`true`**. Such navigation of execution of the code is essential in programming and as such one of the fundamental structures of the field. Lets test test the bumper push-button-switch if it is working properly…

0.1.1 Tasks:

1. Construct the bumper of the robot with push-button-switch as is shown in this video instructions.
2. And connect the push-button-switch (PBSW) terminals with module RobDuino according to tbl. 1:

Table 1: Connection of push-button-switch to the Robduino module.

PBSW con.	RobDuino connectors
No. 1	A0
No. 2	GND
No. 3	+5V

3. Test the push-button-switch in the bumper with next prog. 1:

Program 1: Conditional Statements.

```
const int BUMPER_PIN          = A0;
const int TEST_BUMPER_LED_PIN = 3;
void setup()
{
  pinMode(BUMPER_PIN, INPUT);
  pinMode(TEST_BUMPER_LED_PIN, OUTPUT);
}

void loop()
{
  bool bumperIsPressed = digitalRead(BUMPER_PIN);
  if ( bumperIsPressed ) digitalWrite(TEST_BUMPER_LED_PIN, HIGH);
}
```

2. Then... complete the program to turn OFF the LED when the bumper is not touching anything.
3. Next... Change IF statements into single one IF-THEN-ELSE statement.

0.1.2 Questions:

1. Check if the LED on the output terminal D3 is ON when the bumper is pressed.
2. Measure the voltage potencial at the terminal A0 when the bumper is pressed.
3. Explain when the curly braces `{}` are necessary in the if-statement.

0.1.3 Summary:

0.1.3.1 IF Statement If statement can be written in several forms. The easiest one is:

```
1   if (value_one) statement1;
```

In this case the variable named `value_one` can hold some numerical number. If `value_one` is **`true`** or greater than `0` the program will execute `statement1`. But this simple example is not used so often due its simplicity. We rather use it in this form:

```
1   if ( value_one == value_two ){
2    statement1;
3    statement2;
4   }
```

In this case `value_one` can be any number and the `statement1` and `statement2` will be executed if the `value_one` will be equal to `value_two`. These command can be expanded into IF-ELSE form:

```
1   if ( value_one == value_two ){
2    statement1;
3    statement2;
4   }else{
5    statement3;
6   }
```

0.1.3.2 Condition operators Also other logical condition operators can be used:

- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`
- Equal to `a == b`
- Not Equal to: `a != b`

And some more conditional statements are available in C++:

- **`if`** - to specify a block of code to be executed, if a specified condition is true

- **`else`** - to specify a block of code to be executed, if the same condition is false
- **`else`** - if to specify a new condition to test, if the first condition is false
- **`switch`** - to specify many alternative blocks of code to be executed

0.1.4 Issues:

0.1.4.1 **<++>** <++>