5.6 Flow control

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…

5.6.1 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.

5.6.2 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.

5.6.3 Summary:

5.6.3.1 Conditional statements Conditional statements in C++ allow you to execute different blocks of code based on whether a condition is true or false. There are several different types of conditional statements in C++, including if, if-else, and switch.

Here is an example of how to use an if statement in C++:

```
int x = 10;

if (x > 5) {
  // This code will be executed if x is greater than 5
  printf("x is greater than 5");
}
```

In this example, the if statement checks whether the value of x is greater than 5. If it is, the code block inside the curly braces will be executed. If it is not, the code block will be skipped.

Here is an example of how to use an if-else statement in C++:

```
int x = 10;

if (x > 5) {
  // This code will be executed if x is greater than 5
  printf("x is greater than 5");
} else {
  // This code will be executed if x is not greater than 5
  printf("x is not greater than 5");
}
```

In this example, the if-else statement first checks whether the value of x is greater than 5. If it is, the code block inside the first set of curly braces will be executed. If it is not, the code block inside the second set of curly braces will be executed.

Here is an example of how to use a switch statement in C++:

```
int x = 2;

switch (x) {
  case 1: printf("x is 1"); break;
  case 2: printf("x is 2"); break;
  case 3: printf("x is 3"); break;
  default: printf("x is something else"); break;
}
```

In this example, the switch statement checks the value of x and executes the code block corresponding to the first case label that matches the value. The break statements are used to exit the switch statement once a match is found. If no match is found, the code block for the default label is executed.

5.6.3.2 IF Statement

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

```
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:

```
if ( value_one == value_two ){
 statement1;
 statement2;
}
```

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:

```
if ( value_one == value_two ){
 statement1;
 statement2;
}else{
 statement3;
}
```

5.6.3.3 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`

5.6.4 Issues:

5.6.4.1 <++> <++>