# BNF DESCRIPTION

**<program>** ::= START<stmts>FINISH

**<stmts>** ::= <stmts><stmt> | <stmt>

**<stmt>** ::= <matched_stmt> | <unmatched_stmt>

**<matched_stmt>** ::= if <LP>  <logic_exp> <RP> <matched_stmt> else <matched> | <non_if_stmt>

**<unmatched_stmt>** ::= if <LP> <logic_exp> <RP> <stmt> | if <LP> <logic_exp> <RP> <matched> else <unmatched>

**<non_if_stmt>** ::= <loop_stmt> | <assign_stmt> | <var_declaration> | <comment> | <func_call_stmt> | <func_declaration> | <return_stmt>

**<assign_stmt>** ::= <lside> <equal> <rside>

**<rside> :**:= <logical_expression>

**<logical_expression> :**:= <or_expression>

**<or_expression>** ::= <or_expression> <or_relation><and_expression> | <and_expression>

**<and_expression>** ::= <and_expression><and_relation><comparison_expression> | <comparison_expression>

**<comparison_expression>** ::= <comparison_expression><comparator><add_sub_expression> | <add_sub_expression>

**<add_sub_expression>** ::= <add_sub_expression> <plus>  <mul_div_expression> | <add_sub_expression> <minus> <mul_div_expression> | <mul_div_expression>

**<mul_div_expression>** ::= <mul_div_expression> <multiply> <pow_expression> | <mul_div_expression> <divide> <pow_expression> | <pow_expression>

<**pow_expression>** ::= <pow_expression> <pow_sign> <element> | <element>

**<element>** ::= <LP> <logical_expression> <RP> | <var_id> | <integer> | <double_value> | <array_init> | <string_init> | <connect_init>

**<lside> :**:= <var_id> | <var_declaration>
**<comment>** ::=<comment_string>

**<code_block>** ::= // <stmts> \\

**<string_init>** ::= <string>

# ARRAYS

**<array_init>** ::= <LSB><array_elements><RSB>

**<array_elements>** ::= <array_elements> <comma> <array_element> | <array_element>

**<array_element>** ::= <integer> | <double_value> | <string>

# DECLARATIONS

**<var_declaration> :**:= <type_id> <var_id> | <type_id> array <var_id>

**<type_id**> ::= <int> | <double> | <string_type>| <connection>

**<var_id>** ::= <identifier>

**<func_name>** ::= <identifier>

**<int>** ::= int

**<double> ::**= double

**<string_type>** ::= string

**<array>** ::= array

**<connection>** ::= connection

# LOOPS

**<loop_stmt>** ::= <for_stmt> | <while_stmt>

**<for_stmt>** ::= for <LP> <var_id> in  <var_id> <RP> <codeblock> | for <LP> <assign_stmt> where <logical_expression> with <assign_stmt> <RP> <codeblock> | for <LP> <assign_stmt> where <logical_expression> <RP> <codeblock>

**<while_stmt>** ::= while <LP> <logical_expression> <RP>  <code_block>

# FUNCTIONS

**<func_call_stmt>** ::= <primitive_func_call> | <non_primitive_func_call>

**<primitive_func_call>** ::= <read_temp> | <read_hum> | <read_air_p> | <read_air_q> | <read_light> | <read_sound_lvl> | <read_timestamp_from_timer> | <send_integer_to_connection> | <read_integer_from_connection> | <connect>

**<non_primitive_func_call>** ::= <func_name> <LP> <parameter_list_on_call>? <RP>

**<parameter_list_on_call>** ::= <rside> |  <parameter_list_on_call> <comma> <rside>

**<func_declaration>** ::=  func <func_name> <LP> <parameter_list>? <RP>

**<return_stmt>** ::= return <var_id> | return <double_value> | return <integer> | return <string>

**<parameter_list>** ::= <var_declaration> | <parameter_list> <comma> <var_declaration>

**<read_temp>** ::= read_temp<LP><RP>

**<read_hum>** ::= read_humidity<LP><RP>

**<read_air_p>** ::= read_air_pressure<LP><RP>

**<read_air_q>** ::= read_air_quality<LP><RP>

**<read_light>** ::= read_light<LP><RP>

**<read_sound_lvl>** ::= read_sound_level<LP><double_value><RP> | read_sound_level<LP><var_id><RP>

**<read_timestamp_from_timer>** ::= read_timer<LP><RP>

**<send_integer_to_connection>** ::= send_int<LP><var_id><comma><var_id><RP> | send_int<LP><var_id><comma><integer><RP>

**<read_integer_from_connection>** ::= read_int<LP><var_id><RP>

**<connect_init>** ::= connect<LP><var_id><RP> | connect<LP><string><RP>

# SWITCHES

**<control_switches>** ::= <switch_state>

**<switch_states>** ::= <turn_on> | <turn_off>

**<turn_on> ::=** SwitchON1 | SwitchON2 **|** SwitchON3 | SwitchON4 | SwitchON5 | SwitchON6 | SwitchON7 | SwitchON8 | SwitchON9 | SwitchON10

**<turn_off>** ::= SwitchOFF1 | SwitchOFF2 **|** SwitchOFF3 | SwitchOFF4 | SwitchOFF5 | SwitchOFF6 | SwitchOFF7 | SwitchOFF8 | SwitchOFF9 | SwitchOFF10

# SYMBOLS

**<digit>** ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

**<integer>** ::= <digit> | <integer> <digit>

**<double_value>** ::= <integer><dot><integer>

**<string>** ::= “<string_characters>”

**<comment_string>** ::= #<string_characters>#

**<string_characters>** ::= <character> | <string_characters> <character>

**<character>** ::= <lowercase_letter> | <uppercase_letter> | <digit> | <symbols> | <space>

**<lowercase_letter>** ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

**<uppercase_letter>** ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

**<symbols>** ::= ! | ‘ | ^ | # | + | $ | % | & | / | { | ( | [ | ) | ] | = | ? | * | \ | - | “ | | | : | . |

**<space>** ::= SPACE

# TOKENS

**<identifier>** ::= <lowercase_letter> | <identifier><lowercase_letter>  | <identifier><underscore> | <identifier><digit>

**<and>** ::= &

**<and_relation>** ::= <and><and>

**<or>** ::= |

**<or_relation>** ::= <or><or>

**<equal>** ::= =

**<comma>** ::= ,

**<LP>** ::= (

**<RP>** ::= )

**<LSB>** ::= [

**<RSB>** ::= ]

**<minus>** ::= -

**<underscore>** ::= _

**<plus>** ::= +

**<multiply>** ::= *

**<divide>** ::= /

**<hashtag>** ::= #

**<pow_sign>** ::= ^

**<comparator>** ::= > | < | >= | <= | ==