**Basics for Project Zero (Quick Reference)**

For this project we will be using two files that have predefined code: MotorUtilities.nxc and SensorUtilities.nxc. Below you will find descriptions of how the different sections of code can be used.

Keep in mind:

* NXC is **case sensitive**. If a variable or keyword is uppercase, typing it in lowercase will cause an error when you compile.
* All open parentheses and open curly braces must eventually be closed with a closing parenthesis or curly brace.
* Most lines of code will be followed by a semicolon. Forgetting a semicolon is a common mistake that will cause compile errors.

**Variables**

* There are several different data types in NXC, but the main two you will be using in this activity are **int** and **bool**. The **int** data type represents an integer value while the **bool** type is used to store “true” or “false” and is very helpful for conditional statements.
* Before you use a variable you will want to **declare** and **instantiate** it.
  + To **declare** a variable you first put its data type followed by a variable name

Ex: **bool** isOdd;

* + To i**nitialize** a variable you put the variable name equal sign and then the value.

Ex: isOdd = false;

* You can combine these steps into one line

Ex: **bool** isOdd = false;

**Constants**

If you want to use a value that will never change you can use a constant. Constants are defined at the top of your file and follow this format:

Ex: #**define** PI\_ESTIMATE 3.14

It’s good practice to put the name of the constant in uppercase, making it easy to tell whether something is a constant or a regular variable. When you’re defining a constant you do not need to state its data type. Also note that you do not need to put a semicolon at the end of these lines!

**Basic Functions**

* To move forward : MotorsForward(MOTOR\_ports of wheels, power, turn ratio);

MotorsForward(MOTOR\_AC, 45, 0 );

* To move Backwards : MotorsReverse(MOTOR\_ports of wheels, power, turn ratio);

MotorsReverse(MOTOR\_AC, 45, 0);

* To state how long a function should work: Wait(time in ms);

Wait(1000);

* To Turn: Move forward or backward with just one wheel
  + To turn when moving backward

MotorsReverse(MOTOR\_C, 75, 0);

Wait(360);

* + To turn when moving forward

MotorsForward(MOTOR\_A, 75, 0);

Wait(360);

* To rotate the motors a certain number of degrees: RotateMotors(MOTOR\_ports of wheels, power, degrees)

RotateMotors(MOTOR\_AC, 60, 180);

Note: You do not need to use Wait after RotateMotors, it will stop once it has rotated the correct number of degrees.

* To power off wheels: Off(OUT\_Ports of wheels);

Off(MOTOR\_AC);

* To just power off the wheels (not braking):  Coast(MOTOR\_ports of wheels)

Coast(MOTOR\_AC);

**To Use Sensors**

* To use the **light sensor**:
  + You need to add the following line at the very top of your program:

**#include** "SensorUtilities.nxc"

* + You need to add a THRESHOLD value to test the light sensor against. Add this after any include statements that you have, for example

**#define** THRESHOLD1 30

* + You need to declare a variable to store the sensor value when it is scanning.

**int** sensorValue = **CheckSensor**(S1);

* + You need to configure the sensor in your program (for example in task main)

**ConfigSensor**(S1, SENSOR\_TYPE\_LIGHT\_ACTIVE,

SENSOR\_MODE\_PERCENT, **false**);

* + Then you use the value of sensorValue (or whatever you called your variable) in your code for comparisons or statements such as:

**if**(sensorValue > THRESHOLD1)

{

MotorsReverse(MOTOR\_C, 75, 0);

}

* To use the **sound sensor**:
  + You need to add the following line at the very top of your program

**#include** "SensorUtilities.nxc"

* + You need to add a THRESHOLD value to test the light sensor against. Add this after any include statements that you have, for example

**#define** THRESHOLD1 30

* + You need to declare a variable to store the sensor value when it is scanning.

**int** sensorValue = **CheckSensor**(S1);

* + You need to configure the sensor in your program (for example in task main)

ConfigSensor(S1, SENSOR\_TYPE\_SOUND\_DB,

SENSOR\_MODE\_PERCENT, false);

* + Then you use the value of sensorValue (or whatever you called your variable) in your code for comparisons or statements such as:

**if**(sensorValue > THRESHOLD1)

{

MotorsReverse(MOTOR\_C, 75, 0);

}

* To use the **touch sensor**:
  + You need to add the following line at the very top of your program

**#include** "SensorUtilities.nxc"

* + You need to declare a variable to store the sensor value when it is scanning.

**int** sensorValue = **CheckSensor**(S1);

* + You need to configure the sensor in your program (for example in task main)

ConfigSensor(S1, SENSOR\_TYPE\_TOUCH,

SENSOR\_MODE\_BOOL, false);

* + Then you use the value of sensorValue (or whatever you called your variable) in your code for comparisons or statements such as:

**if**(sensorValue)

{

MotorsForward(MOTOR\_AC, 75, 0);

Wait(2000);

}

* To use the **ultrasonic sensor**:
  + You need to add the following line at the very top of your program

**#include** "SensorUtilities.nxc"

* + You need to add a THRESHOLD value to test the light sensor against. Add this after any include statements that you have, for example

**#define** THRESHOLD1 30

* + You need to declare a variable to store the sensor value when it is scanning.

**int** sensorValue = **CheckSensor**(S1);

* + You need to configure the sensor in your program (for example in task main)

ConfigSensor(S1, SENSOR\_TYPE\_LOWSPEED\_9V,

SENSOR\_MODE\_RAW, false);

* + Then you use the value of sensorValue (or whatever you called your variable) in your code for comparisons or statements such as:

**if**(sensorValue > THRESHOLD1)

{

MotorsReverse(MOTOR\_C, 75, 0);

}

**Loops and Conditions**

For a **while** statement: while (condition) { set of statements}

* + NOTE that if you want an infinite loop then put while (true)

**while** (turn\_time < 100)

{

MotorsForward(MOTOR\_A, 75, 0););

Wait(2000);

turn\_time = turn\_time + 20;

}

For an **if** statement: if(condition){1 or more statements}

**if** (Random() > 0)

{

MotorsReverse(MOTOR\_AC, 75, 0);

Wait(2000);

}

To use **do while** loop: which means that you are DOing a bunch of code for as long as the WHILE condition is met.

**do**

{

move\_time = Random(1000);

MotorsForward(MOTOR\_A, 75, 0);

}

**while** (move\_time < 800);

**Playing Sound**

To play a tone: PlayTone (frequency, duration)

PlayTone(440, 2000); //play tone for 2 seconds