# DNOC Remote

# Software Design considerations and class definitions

## Main Loop

An Arduino program has two mandatory elements (and does not have any Main as C or C++ does):

setup ()

{

// is called once only at power up or general reset

}

loop()

{

// is called repeatedly; as soon as loop completes it is called again

}

Arduino is C++ based but a subset of full C++. It is single threaded, except that hardware timers will interrupt the main thread.

## Utilities class

WHDUtilities

All class methods (?)

flash ([led = 13], [repeat=false], [on1, off1, on2, off2, on3, off3, on4, off4]=[500, 500,0...]);

NO - use class instance per LED.

Eg

internalLED = new WHDUtilities::LED(13);

internalLED.flashOnce(250,250,250,1000,250,250,250); dash dash pause dash dash

internalLED.flashRepeat(250,250,250,1000,250,250,250, 1000); dash dash pause dash dash pause...

internalLED.off();

internalLED.on();

internalLED.status(); // 0 is off, 1 is ON, 2 is flash once, 3 is flash repeat. CANNOT recover pattern.

Relay class, 4 instances?

On

off

on for x ms

off for x ms

# Remote Unit SMS commands

Note these commands are those sent by SMS, but it is expected that the same format will be used for future implementation of web based service (web services), and via the Serial port over USB, and if BlueTooth is ever implemented.

The devise sends out SMS messages as alerts or as status statements, as described below. The device can be configured by sending SMS messages to it, in a tightly defined syntax.

## Security Setup

Each device of course has its own phone number, but to provide adequate security certain commands must start with a five digit security PIN. This PIN is randomly generated and sent to a pre-coded phone number when the device starts and determines that no device ID code has been stored. This will happen on first startup or after executing the Factory Rest command or after some software upgrades. A new random PIN will be sent every time the devise is reset or powered up until the set Device ID command is received. The sequence is:

1. Power up or reset
2. Device sends two random 5 digit PINs to a preset phonenumber, the 'public' and 'master' PIN numbers
3. That number (and ONLY that number) sends back the 'Set Device ID command'
4. Device is then ready to receive configuration commands.
5. If a “Factory reset” command is received, restart from step 1.

Note the fixed phone number is a constant defined in DNOC.H program header. Initially it is mine (Simon Hewitt), and can be assigned to any team member when production starts. The intention is that it will be a TextLocal number for fully automated web control.

The 'Master' PIN is needed to configure the site data, the alarms and commands such 'Factory reset', and should be known to a limited group of people.

The 'Public' PIN allows querying of the device status and allows silencing of alarms

## Command Syntax

The command syntax is simple and similar to the long standing 'AT' command set for controlling modems. The command looks like:

PPPPP CCC <parameters>

where PPPPP is the 5 digit security PIN.

CCC is the three letter command

<parameters> are the specific parameters for the command

The first letter of the command indicates what type of command it is:

R – Request, ask the unit to send some data by return

C – Set the configuration of the device

S – Security commands

D – Disable alarms

(in the examples, 01012 is the low level security PIN, 98989 is the high level security pin)

Commands:

| Command | Meaning | Parameters | Security Level | Example |
| --- | --- | --- | --- | --- |
| RAL | Request Alarm | Unit sends all active alarms | Low | 01012 RAL |
| Rnn | Details of channel nn | NN is the monitored channel, 01 upwards. Optionally add A for alarms only, C for configuration only, B for Both. Default is A. | Low | 01012 R02 A |
| Dnn | Disable alarms on channel NN | MMM Minutes – how long the alarm is disabled for, 000 is forever. | Medium | 01012 D02 30 |
|  |  |  |  |  |
| SLU | Security – add low security telephone number | Number in the format 004412345123456 | High | 98989 SLU 00447770333444 |
| SHU | Security – add High security phone number | Number in the format 004412345123456 | High | 98989 SHU 004412345654321 |
| S?? | List all authorised phone numbers |  | HIGH | 98989 S?? |
|  |  |  |  |  |
| CST | Configure Site | ID <5 digit ID number> | High | 98989 CST ID 12345 |
|  |  | NAME <site name> |  | 98989 CST NAME East Lothian A |
|  |  |  |  |  |
| CSA | Configure switch alarm | <Alarm Number (1..8)> <Alarm Name> | High | 98989 CSA 1 Main Relay |
|  |  | <alarm Number (1..8)> <Alarm on OPEN/CLOSED> | High | 98989 CSA 2 OPEN |
|  |  | <alarm Number (1..8)> <Alarm repeat at (minutes)> | High | 098989 CSA 1 120 |

# Class Libraries

## WHD\_Util

Utility routines such as string bashing

## DNOC\_Rmac

The classes that describe the RMAC unit functions

Classes:

* Site – Single instance per unit, describes the core unit such as SiteID
* DigitalAlarm – One per no-volt switch alarm – strictly ON or OFF hence digital
* AnalogAlarm – One per Analogue voltage monitor.
* EnvironmentAlarm – Temperature and Humidity, inherits from AnalogAlarm, just different device addressing
* AuthUsers – Authorised ‘Users’, just telephone numbers at present, may be extended later.

## DNOCR\_Config

Single class with many methods to read and write the Flash config files.

## LFlash

Debug classes to allow LinkIt code to be compiled and run on OS X under XCodce. Not 100% perfect, but relicates many LinkiIt class methods.

# Future Features, bugs , todo

1. Hash encrypt the PINs (as they are clear text in the config files) Note the config files can ONLY be accessed with physical access to the device .
2. Log files. Manage. Most recent X – 31 is good.

# Testing

Features to be tested

## Log and Config files

|  |  |  |  |
| --- | --- | --- | --- |
| # |  |  |  |
| 1 | Log files |  |  |
| 1.1 | New log file created each day |  |  |
| 1.2 | /LOGS folder created if it does not exists |  |  |
| 1.3 | Log files > 31 days old deleted |  |  |
| 1.4 | Log files can be read by Windows and Mac machines as USB devices |  |  |
| 1.5 | Log file dateTimestamp is correct |  |  |
|  |  |  |  |
| 2 | Config files |  |  |
| 2.1 | If NO config files, does fatalError |  |  |
| 2.2 | If .CFG only exists, copies to .DAT and starts OK |  |  |
| 2.3 | On config change .DT is copied to .BAK |  |  |
| 2.4 | Detects missing atributes |  |  |
| 2.4.1 | Wrong class name |  |  |
| 2.4.2 | Wrong attribute name |  |  |
| 2.4.3 | Missing endClass tag |  |  |
| 2.4.4 | Illegal bool value (not true or false) |  |  |
| 2.4.5 | Invalid numeric value |  |  |
| 2.4.6 | Invalid date value (?) |  |  |
|  |  |  |  |

# Startup

|  |  |  |  |
| --- | --- | --- | --- |
| # |  |  |  |
| 3 | Startup |  |  |
| 3.1 | FatalError flashes blue and red LEDs |  |  |
| 3.2 | Reads config files |  |  |
| 3.3 | Blue LED flashes until GSM established, or… |  |  |
| 3.4 | Rapid Blue flashing if no GSM signal |  |  |
|  |  |  |  |
|  |  |  |  |
| 2 |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |