# BrakelZigbee

Zodra er een nieuwe seriële data binnenkomt voegt deze “klasse” de data toe aan een xbee\_data struct.

Van de data die binnenkomt wordt het volgende eruit gefilterd:

* 64-bits Netwerk adres
* Temperatuur, luchtvochtigheid en lichtintensiteit, deze worden omgezet van bytes naar int’s

Vervolgens worden deze gegevens opgeslagen in een lokale variabele genaamd ‘zigbeeData’.

## Methodes

* void setNTP(NTP \*rtc);
  + Deze methode zorgt ervoor dat de goede tijd wordt meegegeven aan het ZigbeePacket, en dat deze overeen komt met de rtc-chip.

# Buffer

De ‘Buffer’ is de klasse die de data in de arduino beheerd. De Buffer wordt alleen benaderd door de BufferManager zodat er geen conflicten komen zodra er 2 “threads” dezelfde data willen lezen.

## Enums

* Readers
  + SDCARD = 0;
    - Deze wordt intern gebruikt
  + DATABASE = 1;
    - Deze wordt intern gebruikt

## Methodes

* Buffer();
  + Deze methode zorgt ervoor dat er een instance kan worden aangemaakt in de BufferManager.
* boolean isEmpty(char process);
  + deze methode geeft aan of de buffer voor een bepaald process leeg is.
* boolean store(xbee\_data data);
  + deze methode slaat de xbee\_data op in de buffer.
* Boolean read0(xbee\_data \*data);
  + Deze methode leest de eerst volgende xbee\_data uit de buffer voor process 0 en slaat deze op in de meegegeven pointer.
* Boolean read1(xbee\_data \*data);
  + Deze methode leest de eerst volgende xbee\_data uit de buffer voor process 1 en slaat deze op in de meegegeven pointer.
* Void showBuffer();
  + Dit is een debug-methode die de inhoud van de buffer uitprint.

# BufferManager

De ‘BufferManager’ zorgt ervoor dat er verschillende methodes tegelijk de buffer kunnen aanroepen, zonder dat er conflicten ontstaan.

## Methodes

* BufferManager();
  + Deze methode zorgt ervoor dat er een instance kan worden aangemaakt.
* Boolean read(xbee\_data \*data, char process);
  + Deze methode zorgt ervoor dat de eerst volgende xbee\_data wordt uitgelezen uit de Buffer voor het desbetreffende process.
* Boolean store(xbee\_data data);
  + Deze methode roept de “store” methode aan van de Buffer. Deze methode zet tijdelijk de interrupts uit zodat de buffer “ongestoord” de data kan opslaan!
* Boolean isEmpty(char process);
  + Deze methode roept de “isEmpty(char process)” methode van de Buffer aan.
* Void showBuffer();
  + Deze methode roept de debug-methode “showBuffer” van de Buffer aan.

# DBClient

De ‘DBClient’ zorgt voor de connectie naar de database, het opbouwen van de post-packeten en de het versturen ervan.

## Methodes

* setBufferManager(BufferManager \*b);
  + Deze methode zorgt ervoor dat de DBClient dezelfde instance van de SDManager gebruikt als de rest van het programma.
* Void setSDManager(SDManager \*sd);
  + Deze methode zorgt ervoor dat de DBClient dezelfde instance van de SDManager gebruikt als de rest van het programma.
* Void setEthernetClient(EthernetClient \*client);
  + Deze methode zorgt ervoor dat de EthernetClient wordt geset zodat er verbinding met het internet gemaakt kan worden.
* Void dbClientSend();
  + Deze methode zorgt ervoor dat er verbinding gemaakt wordt, een post-packet wordt aangemaakt en vervolgens wordt verzonden.
* Void getResponse();
  + Deze methode ontvangt een response zodra er ene post-packet verstuurd is.

# MemoryFree

‘MemoryFree’ is een ‘debug-klasse’ die weergeeft hoeveel geheugen er nog vrij is op de Arduino. Deze library is compatible met de meeste Arduino soorten (Mega, Uno etc.).

# ParallaxLCD

ParallaxLCD zorgt voor een makkelijke benadering van het LCD-scherm.

## Methodes

* LCDSetup();
  + Deze methode initialiseert het LCD-scherm.
* LCDUpdate();
  + Deze methode update het scherm met nieuwe informatie en haalt oude/irrelevante informatie weg.
* Bool LCDAddError(LCDMessage newError)
  + Deze methode voegt een nieuwe error toe aan het scherm.
* Bool LCDAddMessage(LCDMessage new LCDMessage);
  + Deze methode voegt een nieuwe message toe aan het scherm.
* Bool LCDRemoveMessage(String oldLCDMessage);
  + Deze methode verwijderd een oude message van het scherm.
* Bool LCDAddBrokenNode(char\* newNode);
  + Deze methode voegt een message toe van een kapotte node aan het scherm.
* Bool LCDRemoveBrokenNode(String oldNode);
  + Deze methode verwijderd een message van een kapotte node van het scherm.
* Void LCDBeep();
  + Deze methode laat de luidspreker in het scherm 1x piepen.
* Void LCDWrite(String message);
  + Deze methode zet een ‘speciale’ message op het scherm.

# RTCLIB

# SDManager

# Scheduler

De ‘Scheduler’ zorgt voor het automatisch uitvoeren van methodes na een vooraf bepaalde interval.

## Methodes

* Bool addSchedulerEvent(void(\*event)(void), unsigned long interval, unsigned char id);
  + Deze methode voegt een methode toe aan de lijst van methodes die aangeroepen worden.
* Void initScheduler(void);
  + Deze methode initialiseert de scheduler.
* Bool removeSchedulerEvent(unsigned char id);
  + Deze methode verwijderd een methode van de lijst van methodes die aangeroepen worden.
* Void handleScheduler();
  + Deze methode update de scheduler en roept de methodes aan die aangeroepen moeten worden.

# Timers

Deze klasse zorgt ervoor dat er eenvoudig met alle timers op de Arduino Mega gewerkt kan worden.

Note: Timer 0 wordt niet gebruikt want deze wordt gebruikt voor de “millis()” methode van arduino. Als we deze wel zouden gebruiken klopt de DataTime(die we gebruiken in de RTC) niet meer. Ook is Timer 0 een 8bit Timer.

Note2: Timer 2 wordt niet gebruikt omdat dit een 8bit Timer is ipv een 16bit Timer. Timer 2 wordt ook gebruikt voor de “tone()” methode van Arduino.

Note3: Timer 5 (Timer 1 op de Arduino Uno) wordt gebruikt door de Servo Library van Arduino.

# Methodes

* Void Initialize(int milliseconden, int timer);
  + Deze methode initialiseert de gekozen timer op het gekozen aantal milliseconden.
* Void setInterval(int milliseconden, int timer);
  + Deze methode veranderd de interval van de gekozen timer.
* Void attachtInterruptTimer1/3/4/5(void(\*isr)());
  + Deze methodes setten de callback van de desbetreffende timers.

# Types

Deze klasse is er zodat er over het hele project met de xbee\_data struct gewerkt kan worden.

## Enums

* Xbee\_data
  + DateTime timeStamp;
  + Int temperature;
  + Int lightIntensity;
  + Int humidity;
  + Int CO2;
  + Unsigned long nodeAddrLow;
  + Unsigned long nodeAddrHigh;