Discussion Document for Shannon Town’s Entry to the .IE Digital Town Awards

Author: Tony Walsh (kilnageer@gmail.com / 0876 488 522)
Version: v0.4
Last Changed: Tuesday 6th April, 2021 12:18

Overview

Thanks for taking an interest in this initiative and taking the time to open and read this document. I have organised it as follows:

- **Executive Summary**: One paragrah explaination of everything!
- **Grant Award Proposal**: The 500 words of our award application.
- **Example System**: General explantion of a small demonstration system.
- **Governance**: How will the network will be managed
- **Appendices**: For the technically minded, detailed implementation details.

NOTE: This document has been written using LibreOffice open source software and saved as in Microsoft Word .docx format. My appologies if it does not look perfect in your word processor.

Executive Summary

We all use radio frequencies each day. Whether it’s our mobile phone or the WiFi in our house. But if we wanted to say make an air quality sensor for the Shannon Wetlands it would cost us at least 10 euros a month to base it on mobile phone technology and it would eat up the battery forcing us to recharge it frequently. Well, LoRaWAN is a free wireless technology that is ideal for such a sensor and a lot more. It would last for months without charging or indefinitely with a small solar panel and would not cost a penny to run. So I propose we submit a Large Town entry proposing the €9,000 (or €5,000 runner up) grant money be used to buy €40 LoRaWAN nodes (which are used as sensors) and €200 LoRaWAN Gateways that connect these nodes to the internet so the sensor data can be shared with the community. We would then have environment data on an ongoing basis (sensors send data every 30 mins) that we can use to plan our town. This could include air quality sensors outside our schools, a weather station at the Model Aircraft Airfield, water depth and quality at the Wetlands and more. And the beauty of this system is as it is open (i.e. not propriety to any manufacturer or network operator) so we can extend and enhance it as our community groups see fit. I believe we would be the first town in Ireland to have such an environmental network infrastructure. It would also be a fantastic educational resource. Want to know more? Then please read the rest of this document :) By the way, The Things Network which is the community based web portal my proposal is based on has 17754 gateways and 140632 developers worldwide so we are in safe hands :) https://www.thethingsnetwork.org/

Grant Award Proposal

Please visit the following website for full details: https://www.weare.ie/ie-digital-town-awards-judging-criteria/

Their criteria and points for the Digital Vision (Strategy) award is to describe the **ambition**, **thinking** and **rationale** behind the initiative (10 points), as well as the expected **outcomes** and the **impact** it will have on the town (10 points). Explain how it contributes to the overall digital development of the town (10 points) and changing of attitudes/culture (10 points).

The entry must be 500 words or less and address all the above issues. Below is my entry (please let more how I could make this more effective). I will also attach to my online entry a document of endorsements by different Shannon Community group chairs. But that won't be used for judging.

The entry must be submitted by Friday 23rd April 2021

The **ambition** of this proposal is for the residents of Shannon Town to become actively engaged in monitoring, maintaining and improving their immediate environment. My **thinking** is that wireless tecnology has moved on in leaps and bounds over the last 10 years and The Things Network, formed in 2015 is a mature, worldwide technology which harnesses a long range, free to use wireless protocol (LoRaWAN) to allow people to monitor sensors so let's use it so see how own town is doing environmentally. The **rationale** being "you cannot manage if you don't measure" as we all know too well from our experience of the continued importance of the Covid-19 test and trace process.

The initial **outcome** of winning the award is heightened awareness that we in Shannon are actively monitoring the quality of our environment, long-term Shannon could become the poster child of enviromental based community planning with access to accurate before/after data for all initiatives taken in by the town. The **impact** of this will be a greater sense of environmental empowerment to the residents of Shannon.

By installing the handful or so gateways and dozens of assocaited sensor nodes of the Shannon LoRaWAN Area Network (*SLÁN*) we would be **developing** a resilient, future-proofed digital infrastructure unique in Ireland. Something our community can grow and enhance ourselves which being an open source project facillitates. In addition, the opportunities for educational involvement are immense as people can design ther own gateways and sensors in addition to buying off-the-shelf products. And as the historical data begins to accumulate we will have an excellent opportunities for studying it.

I see having such a network in Shannon **changing** so much. It will promote engagement between all community groups. From 10 year olds in CoderDojo in the library "data wrangling" sensor data in Scratch, to Mens' Shed members helping build and install attractive, weatherproof housings for sensors and gateways, to schools monitoring the traffic polution outside their gates, to the model airclub having an online weather station so members can check windspeeds before travelling to the airfield. Each allotment holder could monitor the unique micro-climate of their plot from ground temperature to slug detection! The wetlands could monitor water levels and quality. And I'm sure each resident of shannon you ask would think of 10 more uses. The resulting **attittude** and environmental **culture** change would be seismic.

Winning this award would install pride in the residents of Shannon Town that their town’s technology matches it’s already incredible industrial zone technology and we can start to saying *SLÁN* to a healthier, more enviromently aware community.

(429 words)

(This entry is made online at https://dotiedigitaltownawards.secure-platform.com/a/)

Example System

Say we wanted so see how much sunlight Tola Park is getting each day. What our LoRaWAN system would need is four components:

- a LoRaWan **Node** to take and transmit wirelessly light readings every 30 minutes
- a LoRaWAn **Gateway** to receive these readings every 30 mins and publish them to a Cloud based **Database**
- a Cloud based **Database**
- a **Website** where we can view these stored readings

I feel all environmental data collected by the Shannon LoRaWAN Area Network (*SLÁN*) should be made publically available. However, at the end of this section I will discuss security.

I will now talk out each component in general. For low-level, hands-on, implementation details please see Appendix A.

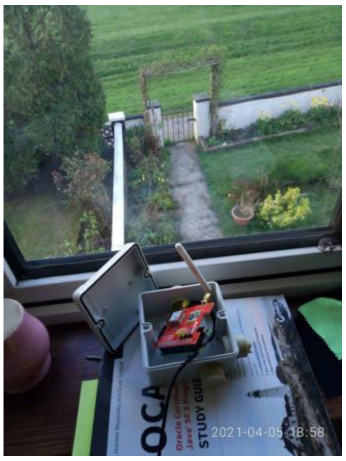
Node

These small battery powered units are the sensors of the system. They typically wake up every half an hour, measure something, transmit it to a **Gateway** and then go back to sleep. The demo system uses one I made myself to measure the light level in Tola Park here's a picture:

And here's a link to a commercially available ones (https://heltec.org/project/htcc-ac02/) which I've ordered 5 of :) With a picture:

Gateway

These units need to be powered by the mains and have internet access. The demo one I made myself and it connects to the internet over WiFi. Here’s a picture.

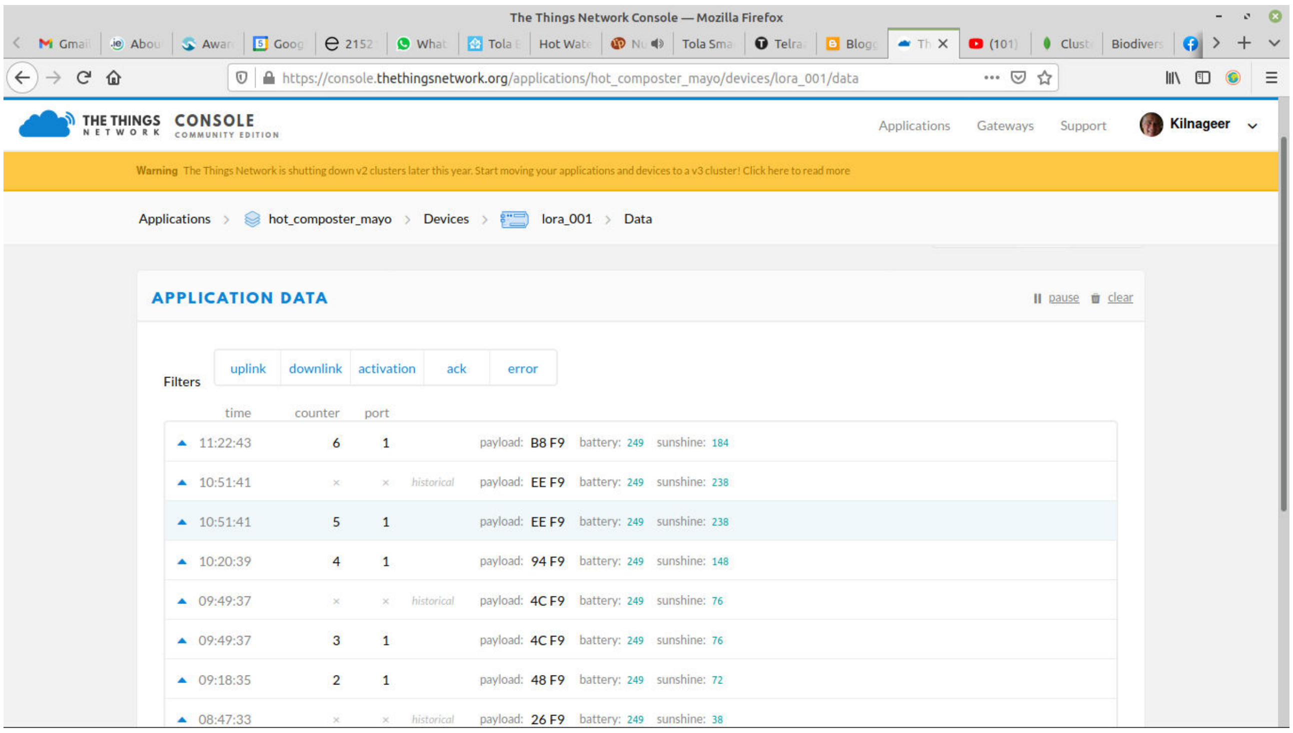


Ideally this would be mounted high up on an external wall with a TV style aerial.

Again there are commercially available ones (https://heltec.org/project/ht-m00/) costing around 50 euro. Here’s a picture:

But this is just the hardware side of the Gateway. It also needs a portal on the internet to which it can send the **Node** sensor data it receives. For this I choose The Things Network (TTN). I have been using this service for over two years and it is very well estblished and supported. See https://www.thethingsnetwork.org/ for more details. Currently, in Ireland there are 54 TTN **Gateways** in Ireland. Worldwide there are 17648.

Here’s a screen shot of the live sensor data coming in:



The Things Network Console — Mozilla Firefox

https://console.thethingsnetwork.org/applications/hot_composter_mayo/devices/lora_001/data

Warning The Things Network is shutting down v2 clusters later this year. Start moving your applications and devices to a v3 cluster! Click here to read more

Applications > hot_composter_mayo > Devices > lora_001 > Data

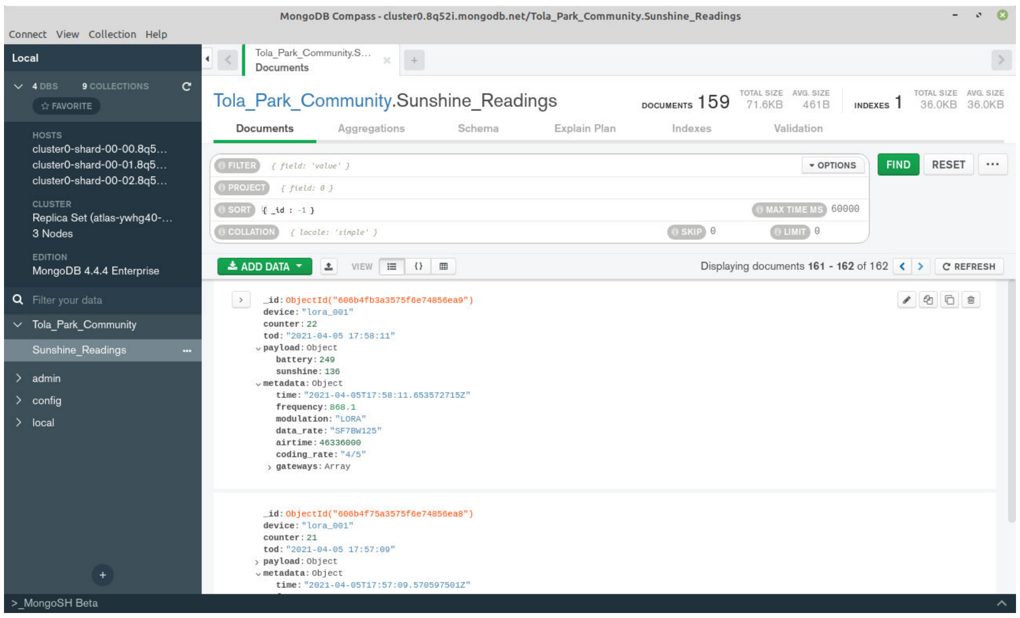
APPLICATION DATA

Filters: uplink | downlink | activation | ack | error

time	counter	port		
11:22:43	6	1		payload: B8 F9 battery: 249 sunshine: 184
10:51:41	×	×	historical	payload: EE F9 battery: 249 sunshine: 238
10:51:41	5	1		payload: EE F9 battery: 249 sunshine: 238
10:20:39	4	1		payload: 94 F9 battery: 249 sunshine: 148
09:49:37	×	×	historical	payload: 4C F9 battery: 249 sunshine: 76
09:49:37	3	1		payload: 4C F9 battery: 249 sunshine: 76
09:18:35	2	1		payload: 48 F9 battery: 249 sunshine: 72
08:47:33	×	×	historical	payload: 26 F9 battery: 249 sunshine: 38

Database

So now we can log in to the The Things Network (TTN) portal and see our Tola Park sensor data arrive live every 30 minutes. But this portal only stores this data for a week. So for long term analysis and ease of access we need to ensure this sensor data is also being stored to a **Database**. For the demo I am using MongoDB Atlas. Whether this is the best choice I don't know. There are a of of choices out there. But it is free and allows us to store 500MB of data in the cloud from 100 sensors which I feel should do us for a few years. After this we can always upgrade of move to another service. It also provides a Charts service which allows us to create live graphs that can be embedded in websites. It has a website as well as a remote concole called Compass. Here's a screenshot of both:


MongoDB Compass - cluster0.8q52i.mongodb.net/Tola_Park_Community.Sunshine_Readings

Tola_Park_Community.Sunshine_Readings

DOCUMENTS 159 TOTAL SIZE 71.6KB AVG. SIZE 461B INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB

HOSTS
cluster0-shard-00-00.8q5…
cluster0-shard-00-01.8q5…
cluster0-shard-00-02.8q5…

CLUSTER
Replica Set (atlas-ywhg40-…
3 Nodes

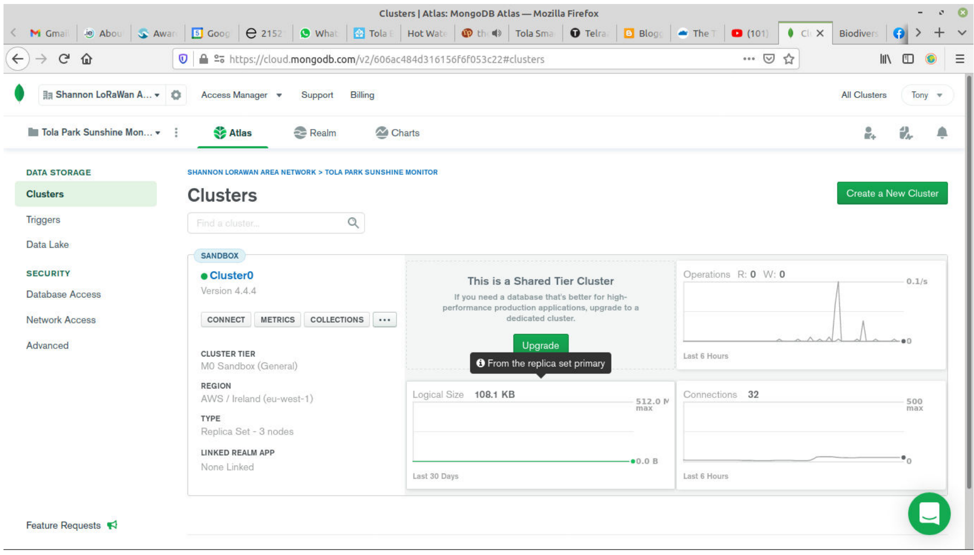
EDITION
MongoDB 4.4.4 Enterprise

FILTER { field: 'value' }
PROJECT { field: 0 }
SORT { _id : -1 }
COLLATION { locale: 'simple' }
MAX TIME MS 60000
SKIP 0
LIMIT 0

Displaying documents 161 - 162 of 162

```
_id: ObjectId("606b4fb3a3575f6e74856ea9")
device: "lora_001"
counter: 22
tod: "2021-04-05 17:58:11"
payload: Object
   battery: 249
   sunshine: 136
metadata: Object
   time: "2021-04-05T17:58:11.653572715Z"
   frequency: 868.1
   modulation: "LORA"
   data_rate: "SF7BW125"
   airtime: 46336000
   coding_rate: "4/5"
   gateways: Array

_id: ObjectId("606b4f75a3575f6e74856ea8")
device: "lora_001"
counter: 21
tod: "2021-04-05 17:57:09"
payload: Object
metadata: Object
   time: "2021-04-05T17:57:09.570597501Z"
```


Clusters | Atlas: MongoDB Atlas — Mozilla Firefox

https://cloud.mongodb.com/v2/606ac484d316156f6f053c22#clusters

SHANNON LORAWAN AREA NETWORK > TOLA PARK SUNSHINE MONITOR

Clusters

SANDBOX
Cluster0
Version 4.4.4

CLUSTER TIER
M0 Sandbox (General)

REGION
AWS / Ireland (eu-west-1)

TYPE
Replica Set - 3 nodes

LINKED REALM APP
None Linked

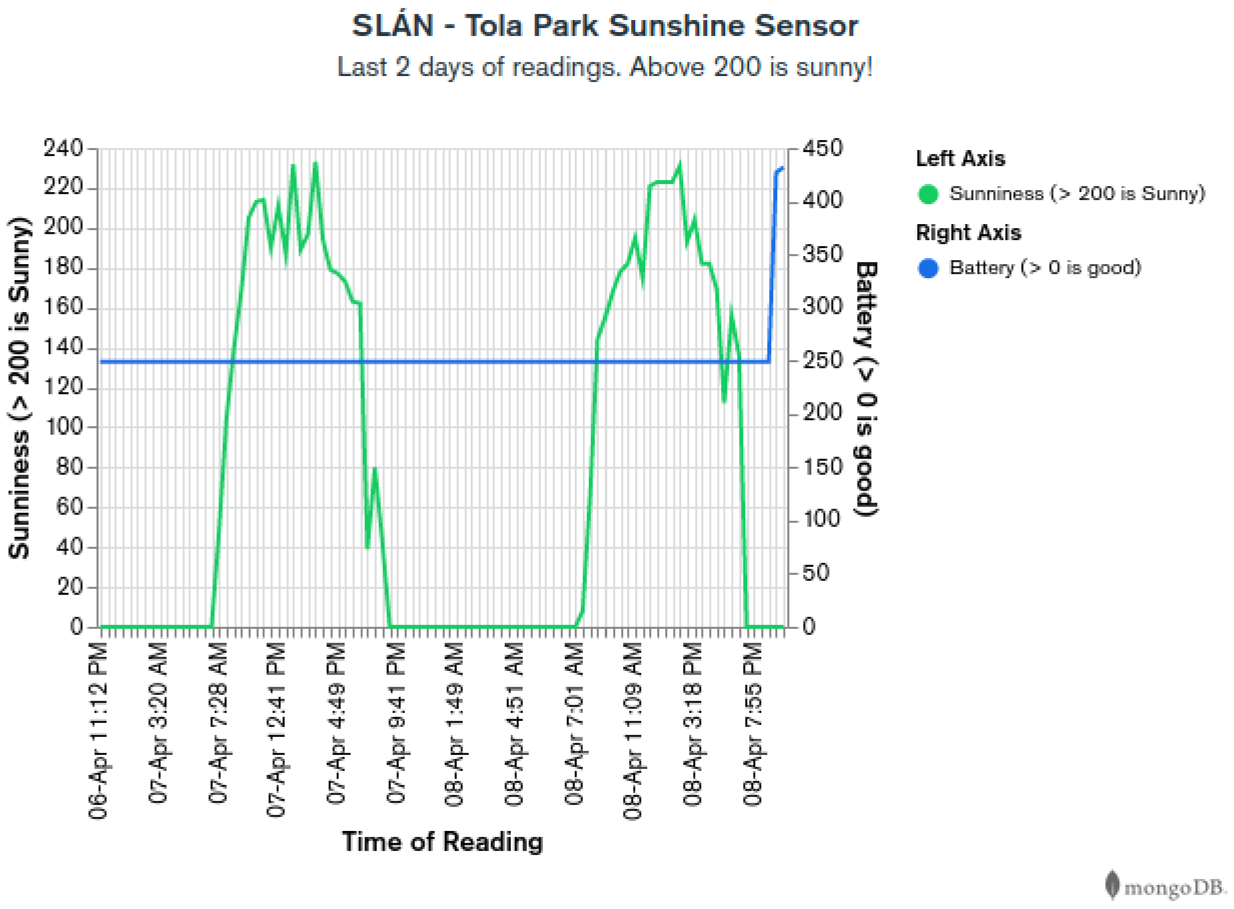
This is a Shared Tier Cluster
If you need a database that's better for high-performance production applications, upgrade to a dedicated cluster.

From the replica set primary

Operations R: 0 W: 0
Logical Size 108.1 KB
Connections 32

Website

The final component of our demo system is a website that allows historic **Node** sensor data to be retrieved from the **Database** and displayed graphically. For the demo I am using the graphcally service called Charts provided by MongoDB Atlas as well as accessing it on a popular smart home platform called Home Assistant. But the beauty of the MongoDB Atlas **Database** is the data can be accessed easily with a few lines of program code (see Appendix D). This makes it ideal for school projects, etc. An example of an auto-updating, embedded chart can be seen at: https://shannontownwetlands.ie/biodiversity-map
Here's an example graph:



Security

This generals falls into three categories: authentication, integrity, privacy.

The TTN uses a protocol that ensures bogus or misformed sensor data does not get accepted. Thus allaying authenticity and data integrity concerns.

It is down to the creator of the sensor and the TTN application it talks to to decide whether they want to encrypt the data the sensor is transmitting. I’m against this.

Governance

I expect the *SLÁN* network to be run by a commitee who will award funds for more equipment on receipt of formal written requests. I proforma application form will be created. For example:

Applicant Name:

Contact Phone and E-mail:

Community Group (If applicable):

How many nodes and gateways are you requesting?:

What will they be used for?:

How will you organise the collected LoRaWAN Node sensor data into The Things Network (TTN) Applications?:

Training material willl also be created and posted to Youtube for general community use. This should include the subjects:

SLÁN Overview. How does it all work?

You want to get a The Things Network (TTN) Gateway. What next?

You want to get a The Things Network (TTN) Node. What next?

How do you access the *SLÁN* database of environmental sensor data?

The next Appendix sections goes into all the above again but in much more detail. You’ve been warned! :)

Appendix A - System Implememtation

Node

For this example system I used a Heltec WiFi LoRa 32(V2) microcomputer (MCU) (https://heltec.org/project/wifi-lora-32/) whose analogue to digital converter (ADC) pin is connected to a voltage divider circuit composed of a light dependent resistor (LDR) and a 10K Ohm resistor. It will be housed in weather resistant (IP67) box along with a lithium polymer (LiPo) rechargeable battery. A glass window in this box will let the LDR measure sunlight strength. A connector will allow a 898MHz aerial to be attached and another connector will allow periodic charging of the node (either by a 5v USB power bank every few months or a permanent solar panel). The MCU will be programmed using Arduino via its micro USB connector to transmit its ADC voltage every 31 minutes to a LoRaWAN **Gateway**.

The Arduino code it is running is listied in Appendix C.

It appears to use 50 mSec of airtime every 60 secs. This, in theory, allows 1000 **Nodes** to operate on the single channel **Gateway**.

Experiments I did using Factory Test peer-to-peer code got good communications between my house in Tola Park to inside the library. This represents a range of some 400m through buildings using short stub antennae. I expected testing with tuned Yagi antenna with the **Gateway** mounted on the gable end of my house may well achieve 1000m which would start to cover the Shannon Wetlands.

I have 5 prebuilt Nodes on order from Heltec as well as 4 development boards. This should allow 10 nodes to be deployed by July 2021. These will be registered on the V3 TTN portal https://eu1.cloud.thethings.network/oauth/login.

Gateway

For this example system I used used a Draguino dual channel Raspberry Pi LoRaWAN HAT (https://www.dragino.com/products/lora/item/106-lora-gps-hat.html) and a Raspberry Pi Zero (RPiZW) (https://www.raspberrypi.org/products/raspberry-pi-zero-w/). It will be housed in weather resistant (IP67) box along with a lithium polymer (LiPo) with external connectors for constant 5v power at 205 mA (aprox.) (estimated from 10.3 mA HAT + 25mA GPS + 170 mA RpiZW https://raspi.tv/2017/how-much-power-does-pi-zero-w-use) and a 868MHz aerial.

The RpiZW runs the Raspbian Linux operating system and expects to have WiFi access to SSID "LoRaDemo" password "LoRaWAN". It's software has been modified to work as a LoRaWAN gateway be following these instructions:
http://wiki.dragino.com/index.php?title=Use_Lora/GPS_HAT_%2B_RaspberryPi_to_set_up_a_Lora_Node#Build_a_single_channel_LoRaWAN_gateway

I also have an 8 channel **Gateway** that need to be built. This is will be attached to a tuned Yagi antenna and mounted on the gable end of my house pointed towards the Town Park. I expect this **Gateway** to allow access to Nodes deloyed in the Shannon Wetlands. This will beregistered on V3 of the TTN portal https://eu1.cloud.thethings.network/oauth/login.

Database

To test that the **Node** and **Gateway** work we need to connect them to a Cloud based database. This project uses The Things Network (TTN) to do this. This free, community based, web portal allows users to register, create projects (call applications) and register all their Nodes for that project. Users can also register their **Gateway**. The Node connects using the TTN generated Device ID and Application ID. (TTN generates these IDs when a user creates a new application and for each node/device it registers to it). A similar method is used to register the **Gateway**. The TTN portal allows the user to monitor the LoRaWAN data received from each of their Nodes and Gateways. But this is transitory data. So this project enabled data storage so that the data can be stored in a database for historic access.

In passing, I am currently using V2 of the TTN portal. I expect to use V3 of this **Gateway** portal https://eu1.cloud.thethings.network/oauth/login by the end of April 2021 which may allow for better sharing of the **Node** sensor data.

Website

The final component of our system is to create a website that allows historic data to be retrieved from the TTN database and displayed graphically. test that the **Node** and **Gateway** work we need to connect them to a Cloud based database. This project uses The

Appendix B - Further Reading

History and overview of The Things Network (TTN)
https://www.thethingsnetwork.org/

Appendix C – Node Arduino Code

This is rough code which will be tidied up and put under Github control by the end of April 2021.

```
/*******************************************************************************
 * Copyright (c) 2015 Thomas Telkamp and Matthijs Kooijman
 * Copyright (c) 2018 Terry Moore, MCCI
 *
 * Permission is hereby granted, free of charge, to anyone
 * obtaining a copy of this document and accompanying files,
 * to do whatever they want with them without any restriction,
 * including, but not limited to, copying, modification and redistribution.
 * NO WARRANTY OF ANY KIND IS PROVIDED.
 *
 * This example sends a valid LoRaWAN packet with payload "Hello,
 * world!", using frequency and encryption settings matching those of
 * the The Things Network.
 *
 * This uses ABP (Activation-by-personalisation), where a DevAddr and
 * Session keys are preconfigured (unlike OTAA, where a DevEUI and
 * application key is configured, while the DevAddr and session keys are
 * assigned/generated in the over-the-air-activation procedure).
 *
 * Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
 * g1, 0.1% in g2), but not the TTN fair usage policy (which is probably
 * violated by this sketch when left running for longer)!
 *
 * To use this sketch, first register your application and device with
 * the things network, to set or generate a DevAddr, NwkSKey and
 * AppSKey. Each device should have their own unique values for these
 * fields.
 *
 * Do not forget to define the radio type correctly in
 * arduino-lmic/project_config/lmic_project_config.h or from your BOARDS.txt.
 *
 *******************************************************************************/


#define VERSION_STRING "v1.3 Monday 5th April 2021 21:37"
// v1.3 has 30 min rather tha 1 min updates



 // References:
 // [feather] adafruit-feather-m0-radio-with-lora-module.pdf

#define LMIC_DEBUG_LEVEL 2

#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

//
// For normal use, we require that you edit the sketch to replace FILLMEIN
// with values assigned by the TTN console. However, for regression tests,
// we want to be able to compile these scripts. The regression tests define
// COMPILE_REGRESSION_TEST, and in that case we define FILLMEIN to a non-
// working but innocuous value.
//
#ifdef COMPILE_REGRESSION_TEST
# define FILLMEIN 0
```

```
#else
# warning "You must replace the values marked FILLMEIN with real values from the TTN control panel!"
# define FILLMEIN (#dont edit this, edit the lines that use FILLMEIN)
#endif

// LoRaWAN NwkSKey, network session key
// This should be in big-endian (aka msb).
static const PROGMEM u1_t NWKSKEY[16] = { HIDDEN };

// LoRaWAN AppSKey, application session key
// This should also be in big-endian (aka msb).
static const u1_t PROGMEM APPSKEY[16] = { HIDDEN };

// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace
// The library converts the address to network byte order as needed, so this should be in big-endian (aka msb) too.
static const u4_t DEVADDR = HIDDEN ; // <-- Change this address for every node!

// These callbacks are only used in over-the-air activation, so they are
// left empty here (we cannot leave them out completely unless
// DISABLE_JOIN is set in arduino-lmic/project_config/lmic_project_config.h,
// otherwise the linker will complain).
void os_getArtEui (u1_t* buf) { }
void os_getDevEui (u1_t* buf) { }
void os_getDevKey (u1_t* buf) { }

//static uint8_t mydata[] = "Hello, world!";
static uint8_t mydata[] = { 0, 249 }; // Sunshine level 0..255 and battery level 0..255
static osjob_t sendjob;

// Schedule TX every this many seconds (might become longer due to duty
// cycle limitations).
const unsigned TX_INTERVAL = (31 * 60); // 60; Let's increase it to 31 mins form 1 minutes

// Pin mapping
// Adapted for Feather M0 per p.10 of [feather]
const lmic_pinmap lmic_pins = {
.nss = 18,
.rxtx = LMIC_UNUSED_PIN,
.rst = 14,
.dio = {26, 34, 35},
};


int sensorValue;


void onEvent (ev_t ev) {

    // Get ADC ready on pin 36
    adcAttachPin(13);
    analogSetClockDiv(255); // 1338mS

    sensorValue = analogRead(A0); // read analog input pin 36 on Heltec LoRa V2
    Serial.print(sensorValue, DEC); // prints the value read 0..4095
    Serial.print(" \n"); // prints a space between the numbers

    mydata[0] = (uint8_t)(sensorValue >> 4); // need to divide by 6 to get into a single byte
    Serial.print(mydata[0], DEC); // prints the value read 0..255
    Serial.print(" \n"); // prints a space between the numbers

    Serial.print(os_getTime());
    Serial.print(": ");
```

```
switch(ev) {
    case EV_SCAN_TIMEOUT:
        Serial.println(F("EV_SCAN_TIMEOUT"));
        break;
    case EV_BEACON_FOUND:
        Serial.println(F("EV_BEACON_FOUND"));
        break;
    case EV_BEACON_MISSED:
        Serial.println(F("EV_BEACON_MISSED"));
        break;
    case EV_BEACON_TRACKED:
        Serial.println(F("EV_BEACON_TRACKED"));
        break;
    case EV_JOINING:
        Serial.println(F("EV_JOINING"));
        break;
    case EV_JOINED:
        Serial.println(F("EV_JOINED"));
        break;
    /*
    || This event is defined but not used in the code. No
    || point in wasting codespace on it.
    ||
    || case EV_RFU1:
    ||     Serial.println(F("EV_RFU1"));
    ||     break;
    */
    case EV_JOIN_FAILED:
        Serial.println(F("EV_JOIN_FAILED"));
        break;
    case EV_REJOIN_FAILED:
        Serial.println(F("EV_REJOIN_FAILED"));
        break;
    case EV_TXCOMPLETE:
        Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
        //mydata[0]++; // incrementing dummy data


        if (LMIC.txrxFlags & TXRX_ACK)
          Serial.println(F("Received ack"));
        if (LMIC.dataLen) {
          Serial.println(F("Received "));
          Serial.println(LMIC.dataLen);
          Serial.println(F(" bytes of payload"));
        }
        // Schedule next transmission
        os_setTimedCallback(&sendjob, os_getTime()+sec2osticks(TX_INTERVAL), do_send);
        break;
    case EV_LOST_TSYNC:
        Serial.println(F("EV_LOST_TSYNC"));
        break;
    case EV_RESET:
        Serial.println(F("EV_RESET"));
        break;
    case EV_RXCOMPLETE:
        // data received in ping slot
        Serial.println(F("EV_RXCOMPLETE"));
        break;
    case EV_LINK_DEAD:
        Serial.println(F("EV_LINK_DEAD"));
        break;
    case EV_LINK_ALIVE:
            Serial.println(F("EV_LINK_ALIVE"));
            break;
        /*
        || This event is defined but not used in the code. No
        || point in wasting codespace on it.
        ||
        || case EV_SCAN_FOUND:
        ||    Serial.println(F("EV_SCAN_FOUND"));
        ||    break;
        */
        case EV_TXSTART:
            Serial.println(F("EV_TXSTART"));
            break;
        case EV_TXCANCELED:
            Serial.println(F("EV_TXCANCELED"));
            break;
        case EV_RXSTART:
            /* do not print anything -- it wrecks timing */
            break;
        case EV_JOIN_TXCOMPLETE:
            Serial.println(F("EV_JOIN_TXCOMPLETE: no JoinAccept"));
            break;
        default:
            Serial.print(F("Unknown event: "));
            Serial.println((unsigned) ev);
            break;
    }
}

void do_send(osjob_t* j){
    // Check if there is not a current TX/RX job running
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println(F("OP_TXRXPEND, not sending"));
    } else {
        // Prepare upstream data transmission at the next possible time.
//        LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
        LMIC_setTxData2(1, mydata, 2, 0);
        Serial.println(F("Packet queued"));
    }
    // Next TX is scheduled after TX_COMPLETE event.
}

void setup() {
//    pinMode(13, OUTPUT);
    SPI.begin(5, 19, 27, 18);

    while (!Serial); // wait for Serial to be initialized
    Serial.begin(115200);
    delay(100);       // per sample code on RF_95 test

    delay(200);                  // wait a 0.2s
    Serial.printf("\n\nLoRaWAN Sunlight Intensity Sensor LDR/2.2KR and HelTec WiFi LoRa 32(V2) %s \n\n",
VERSION_STRING );

    Serial.println(F("Starting"));
    delay(3000);

    //SPI.begin(5, 19, 27);

    #ifdef VCC_ENABLE
    // For Pinoccio Scout boards
    pinMode(VCC_ENABLE, OUTPUT);
    digitalWrite(VCC_ENABLE, HIGH);
```

```
delay(1000);
#endif

// LMIC init
os_init();
// Reset the MAC state. Session and pending data transfers will be discarded.
LMIC_reset();

// Set static session parameters. Instead of dynamically establishing a session
// by joining the network, precomputed session parameters are be provided.
#ifdef PROGMEM
// On AVR, these values are stored in flash and only copied to RAM
// once. Copy them to a temporary buffer here, LMIC_setSession will
// copy them into a buffer of its own again.
uint8_t appskey[sizeof(APPSKEY)];
uint8_t nwkskey[sizeof(NWKSKEY)];
memcpy_P(appskey, APPSKEY, sizeof(APPSKEY));
memcpy_P(nwkskey, NWKSKEY, sizeof(NWKSKEY));
LMIC_setSession (0x13, DEVADDR, nwkskey, appskey);
#else
// If not running an AVR with PROGMEM, just use the arrays directly
LMIC_setSession (0x13, DEVADDR, NWKSKEY, APPSKEY);
#endif

#if defined(CFG_eu868)
// Set up the channels used by the Things Network, which corresponds
// to the defaults of most gateways. Without this, only three base
// channels from the LoRaWAN specification are used, which certainly
// works, so it is good for debugging, but can overload those
// frequencies, so be sure to configure the full frequency range of
// your network here (unless your network autoconfigures them).
// Setting up channels should happen after LMIC_setSession, as that
// configures the minimal channel set. The LMIC doesn't let you change
// the three basic settings, but we show them here.
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI);      // g-band
LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);      // g-band
LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK, DR_FSK), BAND_MILLI);      // g2-band
// TTN defines an additional channel at 869.525Mhz using SF9 for class B
// devices' ping slots. LMIC does not have an easy way to define set this
// frequency and support for class B is spotty and untested, so this
// frequency is not configured here.
#elif defined(CFG_us915) || defined(CFG_au915)
// NA-US and AU channels 0-71 are configured automatically
// but only one group of 8 should (a subband) should be active
// TTN recommends the second sub band, 1 in a zero based count.
// https://github.com/TheThingsNetwork/gateway-conf/blob/master/US-global_conf.json
LMIC_selectSubBand(1);
#elif defined(CFG_as923)
// Set up the channels used in your country. Only two are defined by default,
// and they cannot be changed. Use BAND_CENTI to indicate 1% duty cycle.
// LMIC_setupChannel(0, 923200000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);
// LMIC_setupChannel(1, 923400000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI);

// ... extra definitions for channels 2..n here
#elif defined(CFG_kr920)
// Set up the channels used in your country. Three are defined by default,
// and they cannot be changed. Duty cycle doesn't matter, but is conventionally
```

```
    // BAND_MILLI.
    // LMIC_setupChannel(0,  922100000,  DR_RANGE_MAP(DR_SF12,  DR_SF7),   BAND_MILLI);
    // LMIC_setupChannel(1,  922300000,  DR_RANGE_MAP(DR_SF12,  DR_SF7),   BAND_MILLI);
    // LMIC_setupChannel(2,  922500000,  DR_RANGE_MAP(DR_SF12,  DR_SF7),   BAND_MILLI);

    // ... extra definitions for channels 3..n here.
    #elif defined(CFG_in866)
    // Set up the channels used in your country. Three are defined by default,
    // and they cannot be changed. Duty cycle doesn't matter, but is conventionally
    // BAND_MILLI.
    // LMIC_setupChannel(0,  865062500,  DR_RANGE_MAP(DR_SF12,  DR_SF7),   BAND_MILLI);
    // LMIC_setupChannel(1,  865402500,  DR_RANGE_MAP(DR_SF12,  DR_SF7),   BAND_MILLI);
    // LMIC_setupChannel(2,  865985000,  DR_RANGE_MAP(DR_SF12,  DR_SF7),   BAND_MILLI);

    // ... extra definitions for channels 3..n here.
    #else
    # error Region not supported
    #endif

   LMIC_disableChannel(1);
   LMIC_disableChannel(2);
   LMIC_disableChannel(3);
   LMIC_disableChannel(4);
   LMIC_disableChannel(5);
   LMIC_disableChannel(6);
   LMIC_disableChannel(7);
   LMIC_disableChannel(8);

    // Disable link check validation
    LMIC_setLinkCheckMode(0);

    // TTN uses SF9 for its RX2 window.
    LMIC.dn2Dr = DR_SF9;

    // Set data rate and transmit power for uplink
    LMIC_setDrTxpow(DR_SF7,14);

    // Get ADC ready on pin 36
    adcAttachPin(13);
    analogSetClockDiv(255);  // 1338mS

    // Start job
    do_send(&sendjob);
}

void loop() {
    unsigned long now;
    now = millis();
    if ((now & 512) != 0) {
      digitalWrite(13, HIGH);
    }
    else {
      digitalWrite(13, LOW);
    }

#if 0


        sensorValue = analogRead(A0); // read analog input pin 36 on Heltec LoRa V2
        Serial.print(sensorValue, DEC); // prints the value read 0..4095
        Serial.print("  \n"); // prints a space between the numbers

        mydata[0] = (uint8_t)(sensorValue >> 4);  // need to divide by 6 to get into a single byte
```

```
          Serial.print(mydata[0],   DEC);  // prints the value read  0..255
          Serial.print("  \n"); // prints a space between the numbers

          delay(1000);  // wait 100ms for next reading
#else

   os_runloop_once();
#endif
}
```

Appendix D – Node.js Code to Access MonoDB Atlas Databases

This is rough code that will be refined and placed under Github by the end of Aprl 2021.

```
pi@tola_ttn_gateway:~/thingsconf2019/collector $ cat collector.js
// collector.js
// connects to personal TTN feed using Node.js SDK
// stores data in mongodb database
// Author - Nic Burkinshaw nic@thinnovation.co.uk

const ttn = require('ttn');
const config = require('./config');

const { insertDocument } = require('./mongo.js');

ttn.data(config.ttn.user, config.ttn.password)
   .then(function (client) {
       client.on("uplink", function (devID, payload) {
           var messageObject = {
               device: payload['dev_id'],
               counter: payload['counter'],
               tod: (new Date()).toJSON().slice(0, 19).replace(/T/g, ' '),
               payload: payload['payload_raw'],
               metadata: payload['metadata']
           };
           insertDocument(messageObject)
       })
   })
   .catch(function (error) {
       console.error("Error", error)
       process.exit(1)
   })

function exitHandler(options, err) {
   if (err) {
       console.error('Application exiting...', err);
   }
   process.exit();
}

process.on('exit', exitHandler.bind(null, { cleanup: true }));
process.on('SIGINT', exitHandler.bind(null, { exit: true }));
process.on('SIGUSR1', exitHandler.bind(null, { exit: true }));
process.on('SIGUSR2', exitHandler.bind(null, { exit: true }));
process.on('uncaughtException', exitHandler.bind(null, { exit: true }));
```