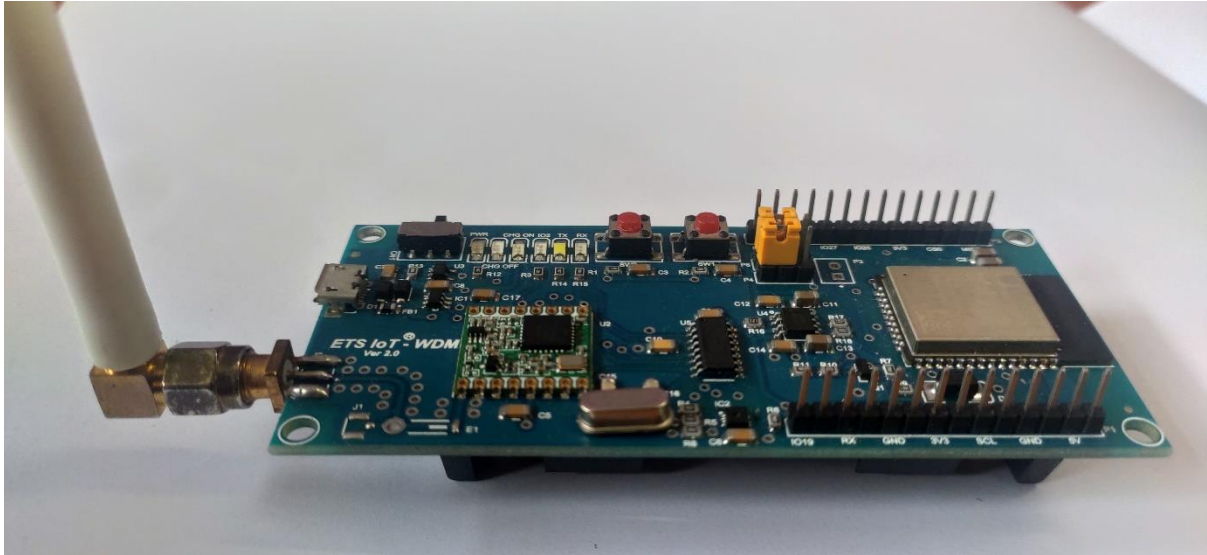


# **ETS IoT® - WDM Ver 2.0**

## **Document Ver 1.0**



### **What is WDM?**

WDM is the Wireless Development Module suitable to complete your Proof of Concept in a faster way. It supports Multiple Wireless Communication Protocol as follows

- Bluetooth Classic
- Bluetooth Low Energy (BLE ver 4.2)
- Wi-fi
- LoRa
- LoRaWAN

### **Features:**

- LoRaWAN 1.0.3 Class A
- LoRaWAN Activation Method Support: ABP & OTAA
- Low power consumption
- Suitable to Interface Different Sensors
- Onboard SHT31 – Temperature and Humidity Sensor
- Bands: IN865
- Arduino Programmable
- LMIC Library Compatible
- Option for Battery Powered Device (3.7V - 18650 rechargeable lithium Polymer battery) – Not Included in the Pack
- Onboard Battery Recharge option

## Specification:

### MCU Specifications:

- CPU: Xtensa dual-core 32-bit LX6 microprocessor, up to 240MHz
- ROM: 448KB for booting and core functions
- SRAM: 520KB for booting and instructions
- SRAM: 16 KB in RTC
- SPI Flash: 4MB
- Ultra-Low Power (ULP) Co-processor
- Crystal oscillator: 40 MHz
- 8x Hybrid Digital IO with Special Functions
- Special Functions: 1x I2C, 1x SPI, 1x UART
- 4x Hybrid Analog & Digital IO: 4 No's
- 2x Hybrid Analog & Digital IN: 2 No's
- Analog Resolution: 8,10,12-bit configurable
- Pulse Width Modulation (PWM)
- Onboard Temperature Sensing (typ., -40°C to 90°C with Accuracy  $\pm 0.3$  °C)
- Onboard Humidity Sensing (typ., 0%RH to 100%RH with Accuracy  $\pm 2\%$  RH)
- Onboard LED: 1xRED
- Baud rate configurable
- 802.11 b/g/n Wi-Fi
- Bluetooth Classic and Bluetooth Low Energy (BLE) in the 2.4GHz band
- General ISM < 1GHz LoRa™ Transceiver 868MHz Surface Mount
- Onboard Antenna for Wi-fi & Bluetooth
- Open source software

### LoRa Specification:

- LoRa Chip: RF96
- Data Rate: 300kbps
- Power Output: 20dBm
- Sensitivity: -148dBm
- Current Transmitting: 120mA
- Operating Temperature: -20°C ~ 70°C
- RF Family/Standard: General ISM < 1GHz
- Protocol: LoRa™
- Modulation: FSK, GFSK, GMSK, MSK, OOK
- Frequency: 865-867 MHz
- Antenna Type - External Antenna via SMA / I-Pex connector
- Supply Voltage: 1.8V ~ 3.7V
- Receiving Current: 12.1mA
- Transmitting Current: 120mA
- Operating Temperature: -20°C ~ 70°C

### **Wi-Fi:**

- 802.11b/g/n
- Bit rate: 802.11n up to 150 Mbps
- A-MPDU and A-MSDU aggregation
- 0.4  $\mu$ s guard interval support
- Center frequency range of operating channel: 2412 ~ 2484 MHz

### **Bluetooth Specification:**

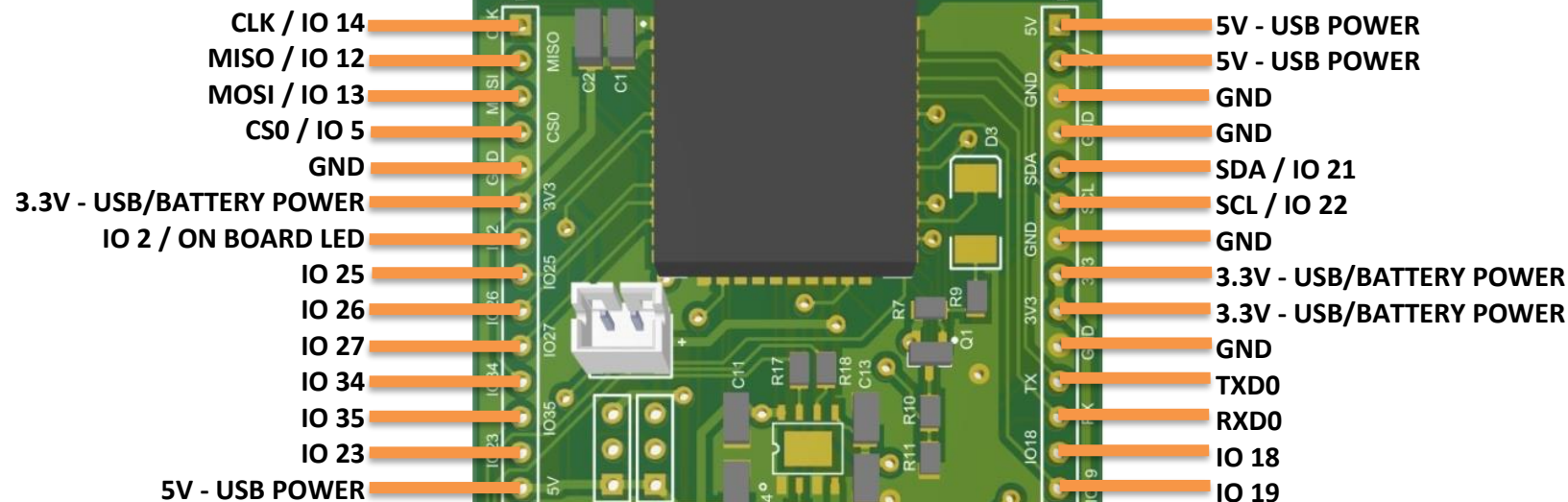
- Bluetooth v4.2 BR/EDR and BLE specification
- Class-1, class-2 and class-3 transmitter
- Adaptive Frequency Hopping (AFH)

### **Common DC Characteristics:**

- Supply Voltage: 5 V
- Operating Voltage: 3.0 - 3.6 V
- Minimum current delivered by power supply: 500 mA
- Battery Voltage: 3.7 V Li-Poly
- operating temperature range:  $-40^{\circ}\text{C} \sim 85^{\circ}\text{C}$
- Wake up from GPIO interrupt, timer, ADC measurements

### **Applications:**

- Home Automation
- Smart Building
- Industrial Automation
- Smart Agriculture



**PIN LAYOUT**  
**ETS IoT - WDM**  
**Ver 2.0**

## ETS IoT® – WDM Ver2.0 Pin Description

Pin No.	Pin Name	IO No.	Type	Function
1	CLK	14	I/O	GPIO, ADC, RTC, SPI_CLK, LoRa_SPI_CLK
2	MISO	12	I/O	GPIO, ADC, RTC, SPI_MISO, LoRa_SPI_MISO
3	MOSI	13	I/O	GPIO, ADC, RTC, SPI_MOSI, LoRa_SPI_MOSI
4	CS0	5	I/O	GPIO, SPI_CS0
5	GND	--	GND	GROUND
6	3V3	--	PWR	3.3V Power Supply while connecting Battery (or) USB
7	IO2	2	I/O	GPIO, ADC, RTC, On Board LED
8	IO25	25	I/O	GPIO, ADC, RTC
9	IO26	26	I/O	GPIO, ADC, RTC
10	IO27	27	I/O	GPIO, ADC, RTC
11	IO34	34	I	Input_Pin, ADC, RTC
12	IO35	35	I	Input_Pin, ADC, RTC
13	IO23	23	I/O	GPIO
14	5V	--	PWR	5V Power Supply while connecting USB Only
15	IO19	19	I/O	GPIO
16	IO18	18	I/O	GPIO
17	RXD0	3	I/O	GPIO, U0RXD
18	TXD0	1	I/O	GPIO, U0TXD
19	GND	--	GND	GROUND
20	3V3	--	PWR	3.3V Power Supply while connecting Battery (or) USB
21	3V3	--	PWR	3.3V Power Supply while connecting Battery (or) USB
22	GND	--	GND	GROUND
23	SCL	22	I/O	GPIO, I2C_SCL, Also Configured for Onboard SHT31 SCL
24	SDA	21	I/O	GPIO, I2C_SDA, Also Configured for Onboard SHT31 SDA
25	GND	--	GND	GROUND
26	GND	--	GND	GROUND
27	5V	--	PWR	5V Power Supply while connecting USB Only
28	5V	--	PWR	5V Power Supply while connecting USB Only

### Note:

- I/O – Input/Output, I – Input, PWR – Power Supply, GND – Ground
- IO No. can be used for Programming the WDM
- IO15 is Internally Connected to LoRa\_NSS
- IO17 is Internally Connected to LoRa\_Reset
- IO4, IO33, IO32 is Internally Connected to LoRa DIO0, DIO1, DIO2 respectively

# Arduino Configuration for Getting Started with WDM

## Prerequisite:

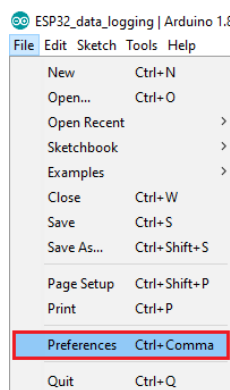
1. **ETS IoT® - WDM Ver 2.0**  
(Buy at: <https://www.enthutech.in/shop/product/wdm-ets-iot-wdm-1591>)
2. **PC / Laptop Installed with ArduinoIDE Ver 1.8.15**  
(Download at: <https://www.arduino.cc/en/software>)
3. **Driver for CH340G**  
(Download at: <https://www.enthutech.in/shop/product/wdm-ets-iot-wdm-1591>)
4. **LMIC Library Files & Sample Codes**  
(Download at: <https://www.arduino-libraries.info/libraries/mcci-lo-ra-wan-lmic-library>)

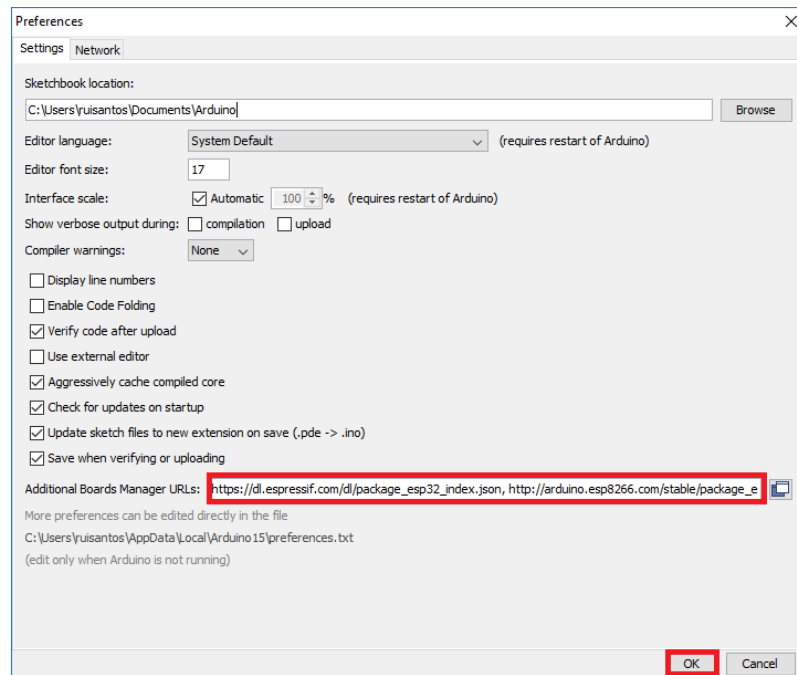
## Procedure for installing and uploading the code in WDM module:

- Arduino IDE Version 1.8.15 is used to program the WDM module
- Use the following link to download the Arduino IDE Version 1.8.15  
<https://downloads.arduino.cc/arduino-1.8.15-windows.exe>

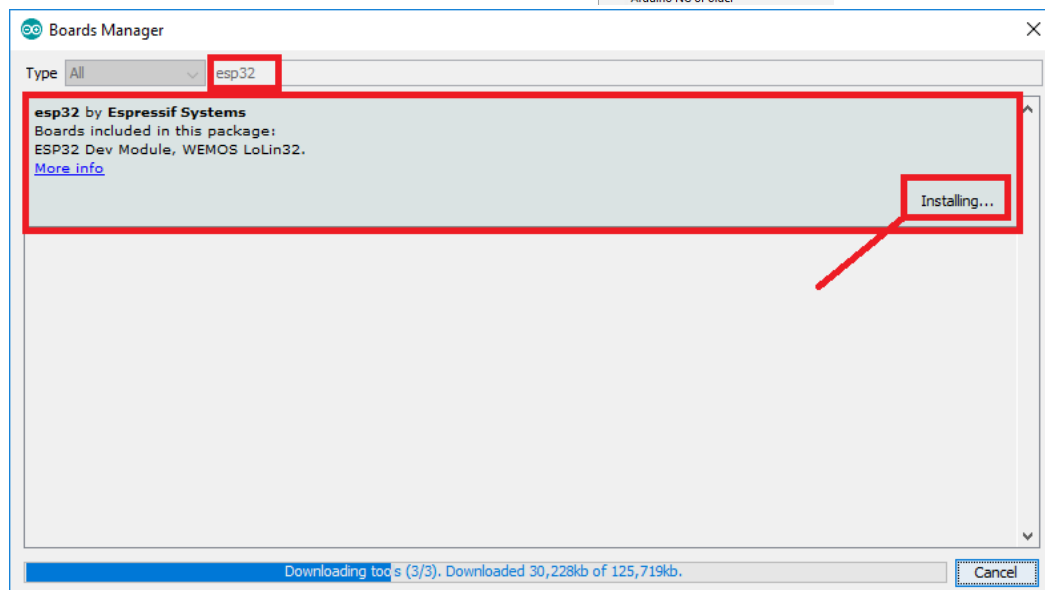
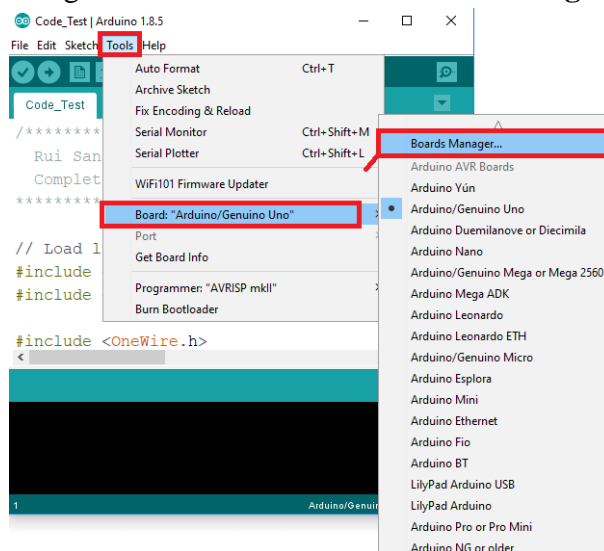


- After installing the ArduinoIDE we need to install the ESP32 board support package in the Arduino IDE software
- Open the Arduino IDE and go to **Files -> Preferences** and paste the below link [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json) in Additional Board manager URL field



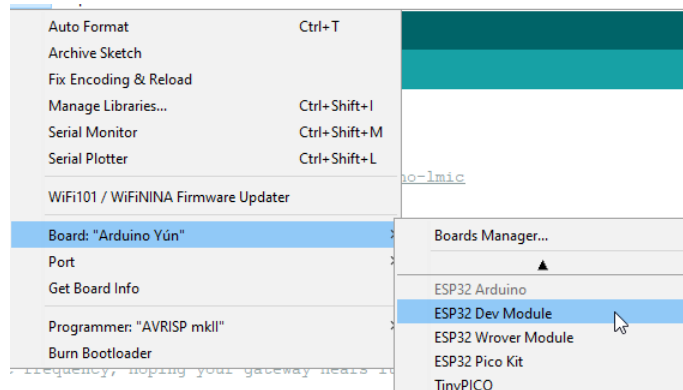


➤ After pasting the link and go to **Tools -> Boards -> Board Manager**

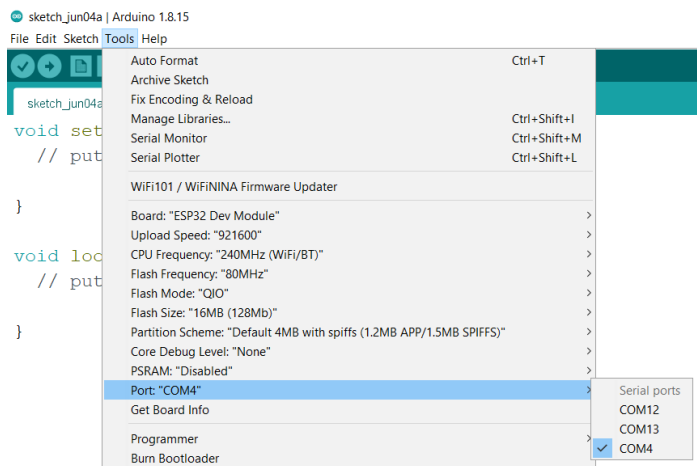




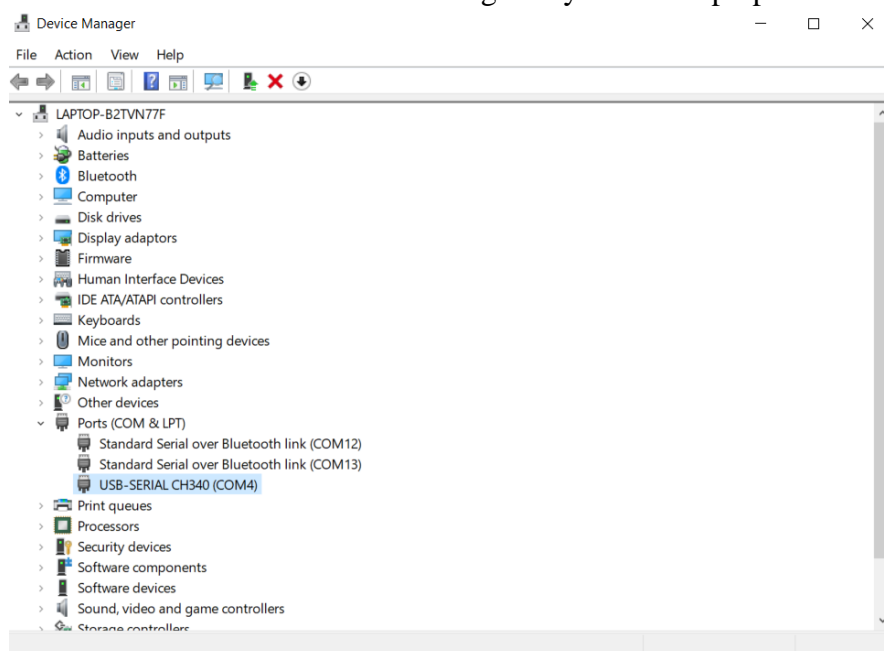
- After installing the ESP32, Go to **Tools >> Board >> ESP32 Dev Module**



- Connect WDM with PC/Laptop Via USB Cable
- Make ON OFF Switch in WDM to ON Position
- In ArduinoIDE, Go to Tools and select all the settings for ESP32 Dev Module as per below image

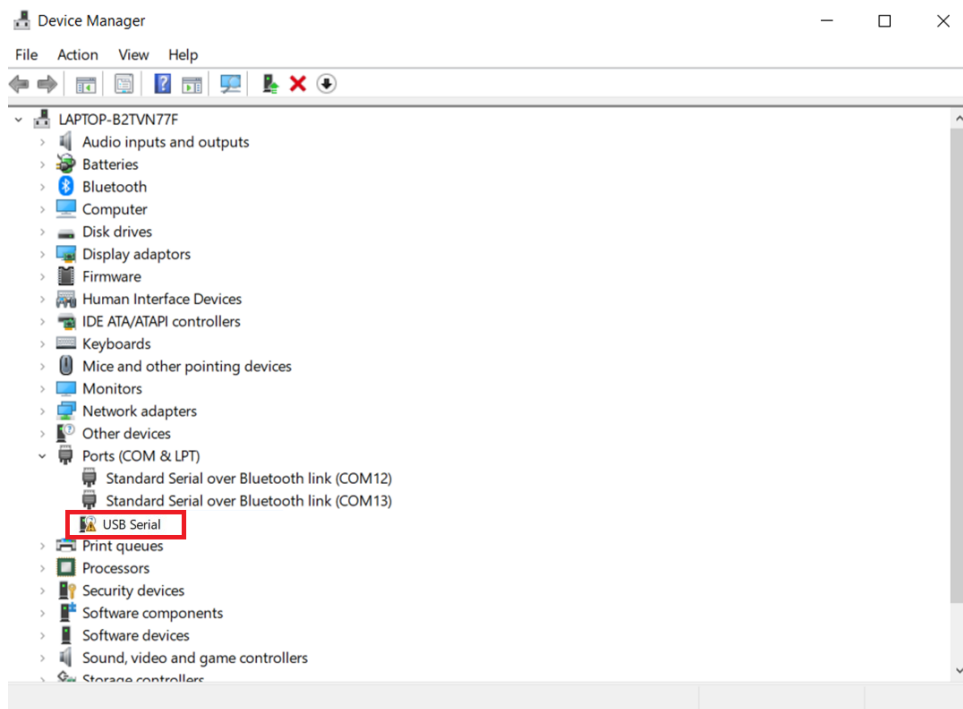


- Here It was showing 3 COM Ports. To Identify right COM Port or to check status of Device driver Installation Check Device Manager in your PC/Laptop





- As per above Image, WDM is connected with USB-SERIAL CH340(COM4)
- In case, if the driver is not installed you will get the COM Port as follows (USB Serial Warning)

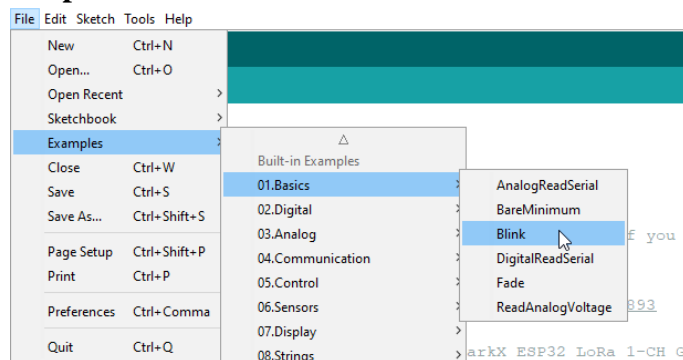


For Installing Driver: Download CH340G Driver for Arduino available at Documents section of <https://www.enthutech.in/shop/product/wdm-ets-iot-wdm-1591>

- After Installing, Select right port at Tools >> Port

## TO UPLOAD SKETCH IN WDM

- Go to Files >> Examples >> Basics >> Blink

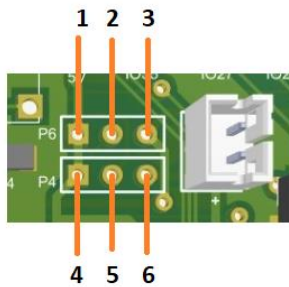


- In Code, replace “LED\_BUILTIN” as 2 in all the places. Where Onboard LED is connected to IO2 of ESP32 as below

```
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(2, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(2, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);           // wait for a second
  digitalWrite(2, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);           // wait for a second
}
```

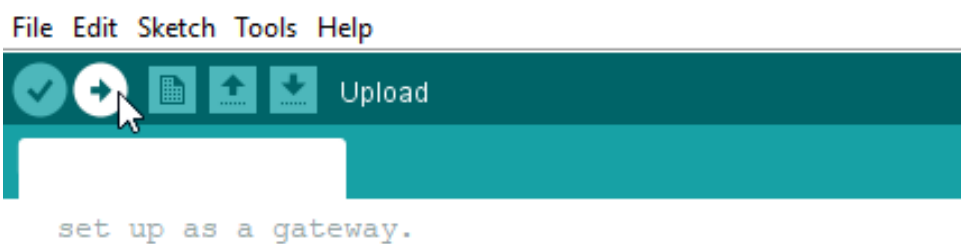
- Before Uploading Code, Check Jumper Option for P6 & P4



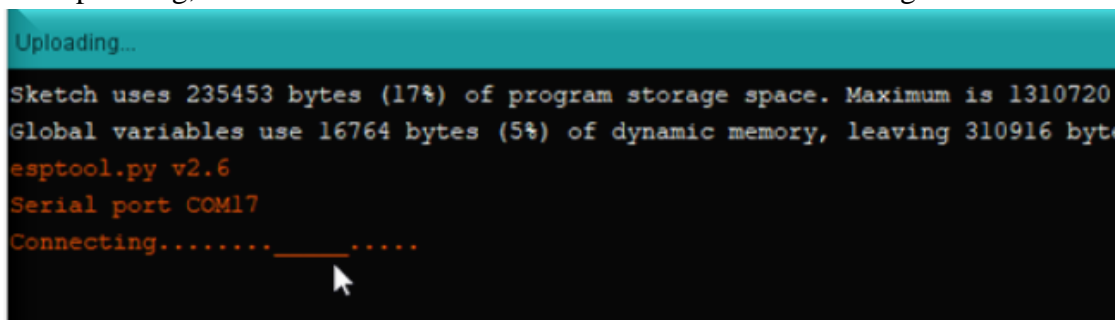
Jumper Position (P4 & P6)	Operation
1-2 (P6)	PWR LED for +5V
2-3 (P6)	PWR LED for +3.3V
4-5 (P4)	USB Power / Programming
5-6 (P4)	Battery Power

**Note:** Keep P6 Jumper Open to make PWR LED OFF when connected to Battery it will save power.

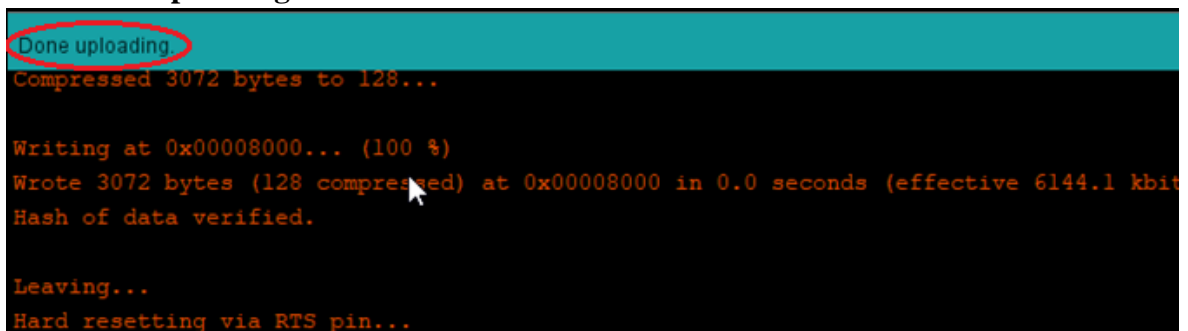
- After Changing above, click the upload icon in the Arduino IDE by clicking Upload Icon as below



- Your code will compile and then it will upload into the WDM module
- While uploading, in the bottom of the Arduino IDE it will show a message as below



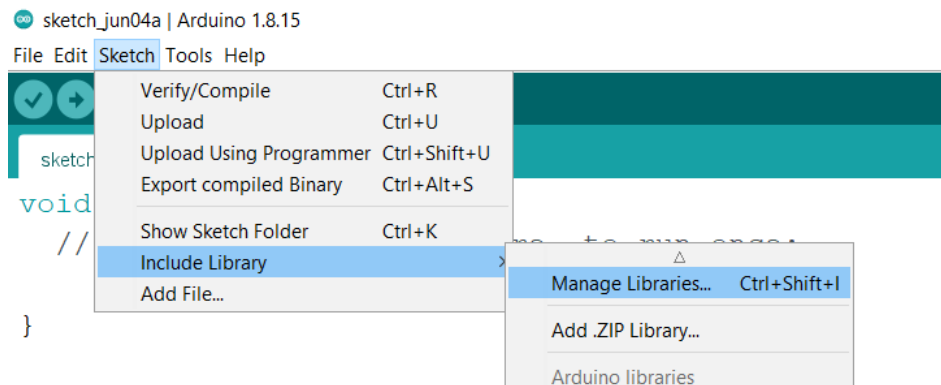
- Once it was trying to connect with Board, press and hold the BOOT button (**Don't Release**) and Reset the device by Pressing & Releasing RST Button then release BOOT button after releasing RST Button. If the coding is uploaded to the WDM module it will show **"Done Uploading"**



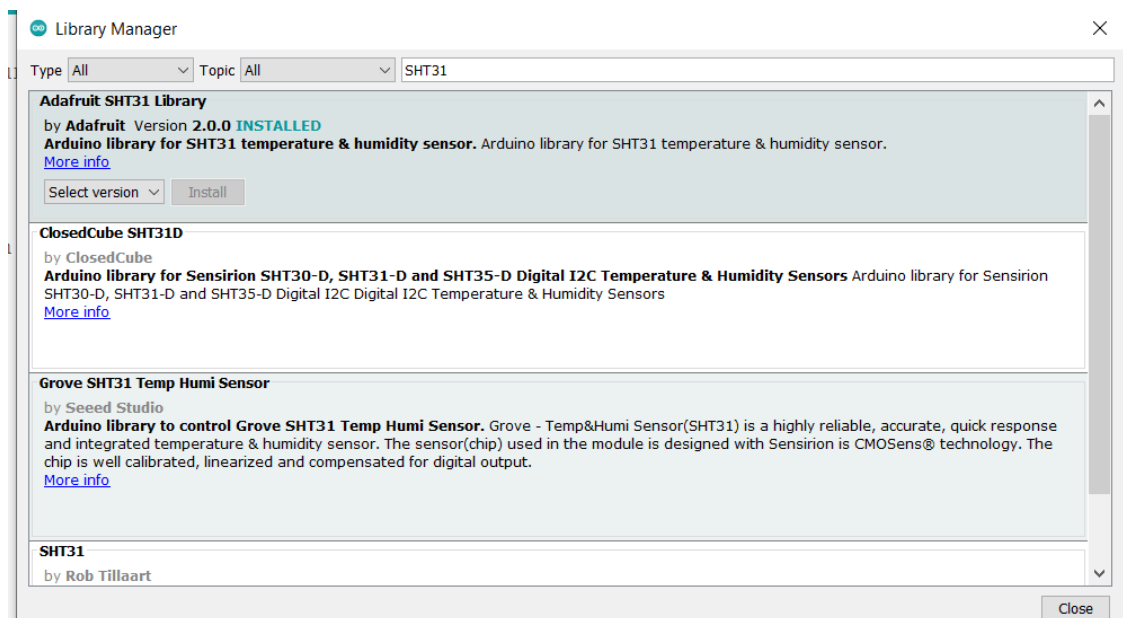
- After Uploading code, press Reset Button in WDM to start executing the code
- Now Check IO2 LED in the WDM board started to Blink as per your code.

## Onboard SHT31 Sensor Configuration in WDM

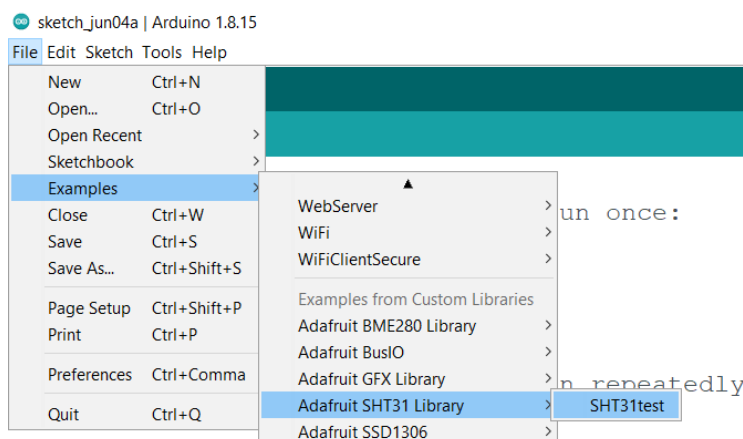
- To Install Library for SHT31 Library, Go to Sketch >> Include Library >> Manage Library



- In Manage Library, Search for SHT31 and Select Adafruit SHT31 Library and Install it.



- After Installing, Open SHT31test code from Examples >> Adafruit SHT31 Library



- Comment the lines highlighted to avoid heater enabled test

```

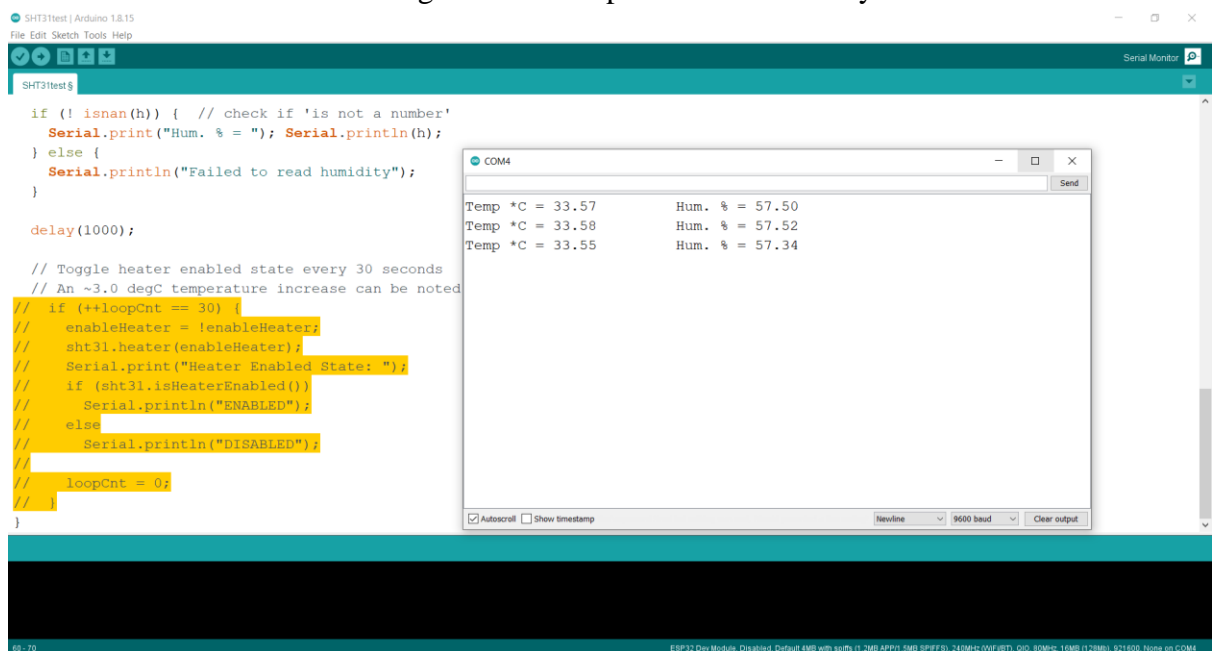
if (!isnan(h)) { // check if 'is not a number'
  Serial.print("Hum. % = "); Serial.println(h);
} else {
  Serial.println("Failed to read humidity");
}

delay(1000);

// Toggle heater enabled state every 30 seconds
// An ~3.0 degC temperature increase can be noted w/
// if (++loopCnt == 30) {
//   enableHeater = !enableHeater;
//   sht31.heater(enableHeater);
//   Serial.print("Heater Enabled State: ");
//   if (sht31.isHeaterEnabled())
//     Serial.println("ENABLED");
//   else
//     Serial.println("DISABLED");
//   loopCnt = 0;
// }
}

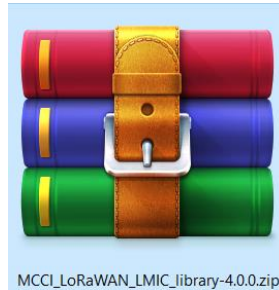
```

- Save & Upload the sketch then press RST Button to see results as below in Serial Monitor. Wait for some time to get stable Temperature & Humidity

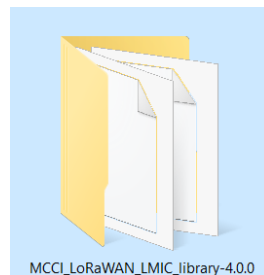


## LMIC LoRaWAN Library Customization for WDM

- Download LMIC Library from <https://www.arduinolibraries.info/libraries/mcci-lo-ra-wan-lmic-library>
- Current Version using for this Demo is [MCCI LoRaWAN LMIC library-4.0.0.zip](#)



- Extract the zip as Folder



- Go to location .....\\MCCI\_LoRaWAN\_LMIC\_library-4.0.0\\project\_config and open lmic\_project\_config.h and make changes as follows for Indian Frequency and save it.

```
// project-specific definitions
// #define CFG_eu868 1
// #define CFG_us915 1
// #define CFG_au915 1
// #define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP /* for as923-JP */
// #define CFG_kr920 1
#define CFG_in866 1
#define CFG_sx1276_radio 1
// #define LMIC_USE_INTERRUPTS
```

- Go to location .....\\MCCI\_LoRaWAN\_LMIC\_library-4.0.0\\src\\hal and open hal.cpp and find below line

### **Before Change:**

```
static void hal_spi_init () {
    SPI.begin();
}
```

- Make changes in SPI.begin() as follows and save it

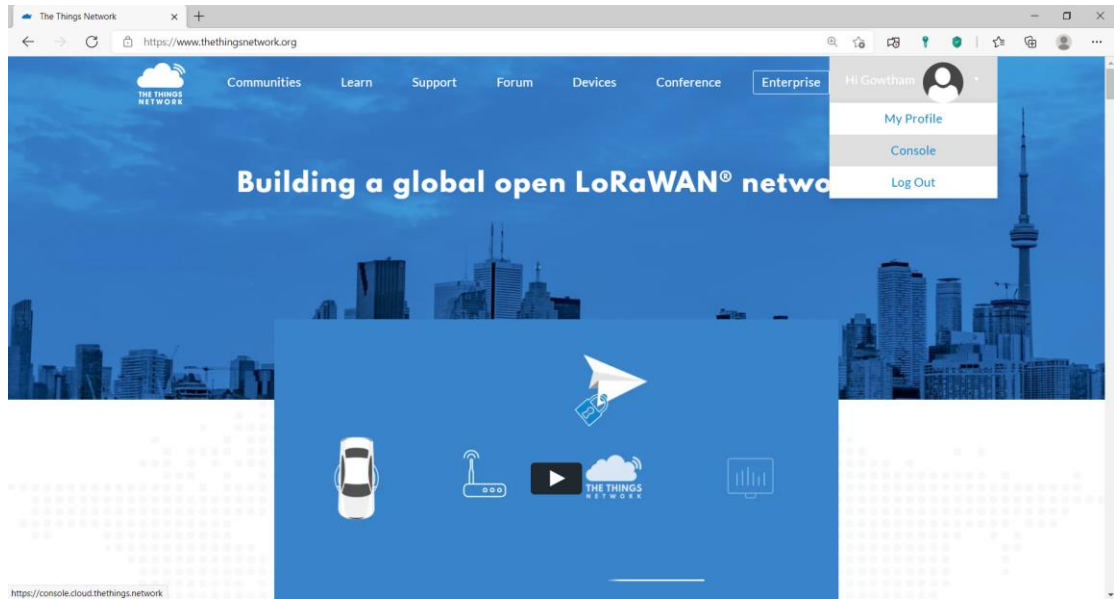
```
static void hal_spi_init () {
    SPI.begin(14,12,13,15);
}
```

**Note: The Pin definition for SPI Pins were as follows CLK – 14, MISO – 12, MOSI – 13, NSS – 15**

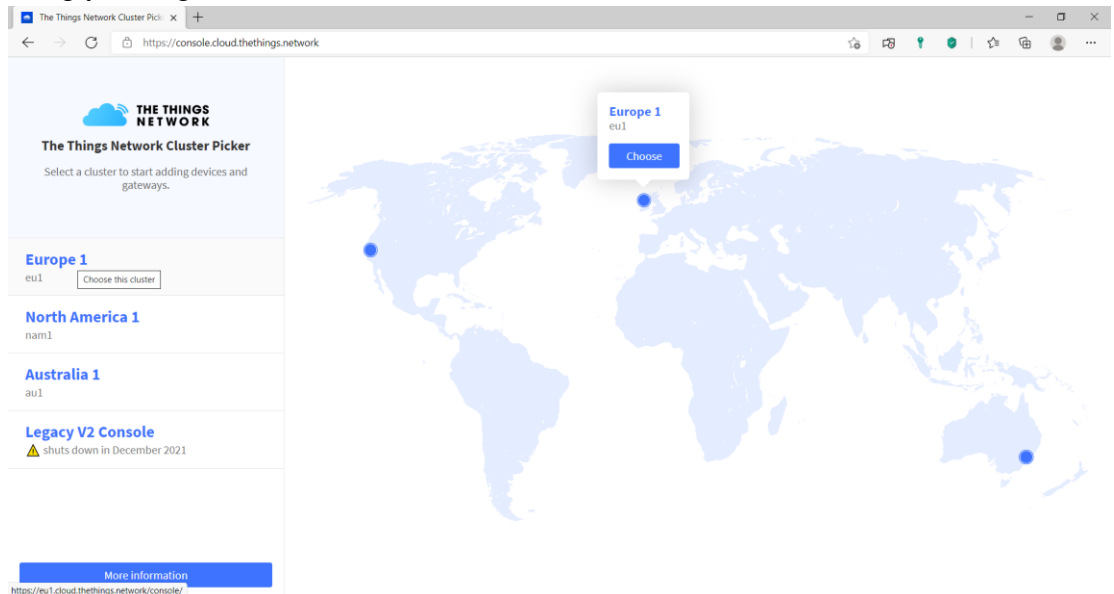
- Copy and paste this folder MCCI\_LoRaWAN\_LMIC\_library-4.0.0 in following location .....\\Documents\\Arduino\\libraries

## Registering Application & WDM Device in ABP mode with TTN v3

- Create a login in “<https://www.thethingsnetwork.org/>” and go to console



- Choose your cluster. I am going to choose Europe 1 & Login with The ThingsID using your registered.



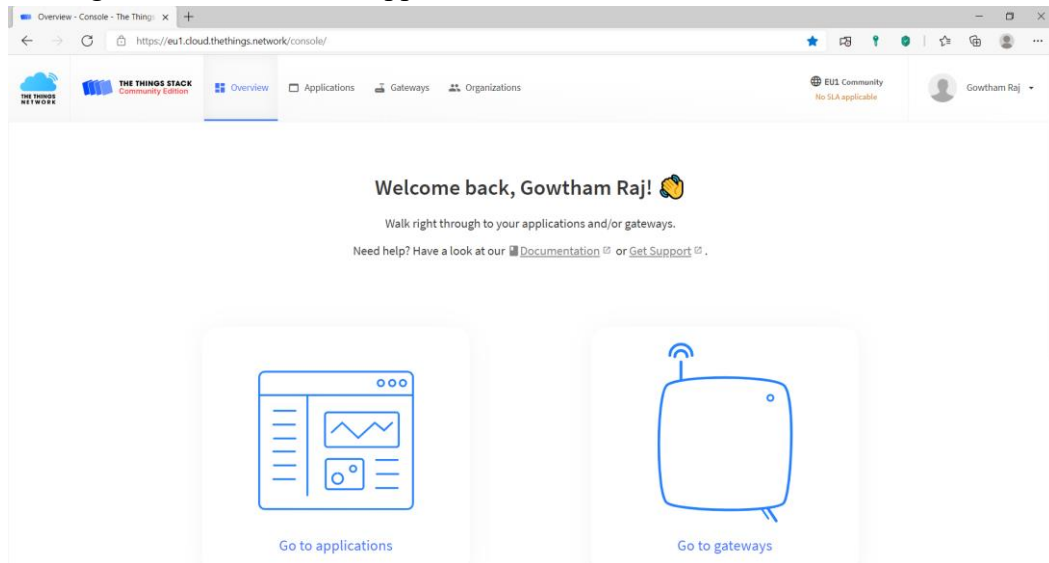
### The Things Network Account

Please login to continue

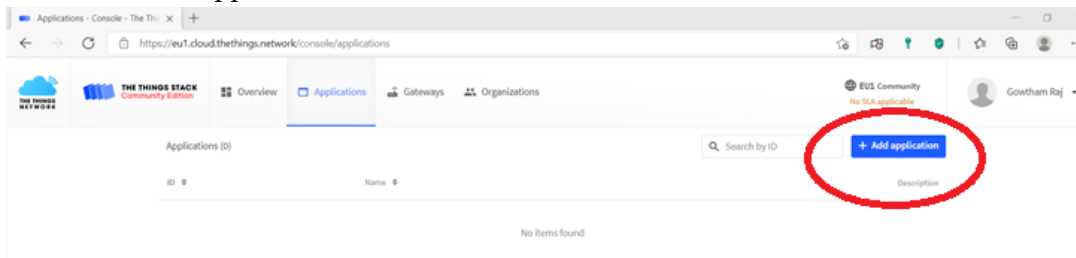
Login with The Things ID

Login using credentials

- After Login, enter into “Go to applications”



- Click on “Add application” and



- Give necessary details and create application.

### Add application

Owner \*

Application ID \*

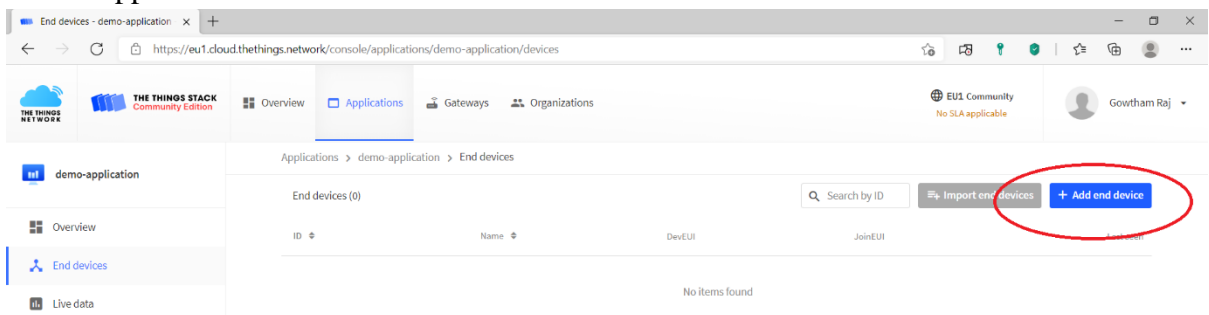
Application name

Description

Optional application description; can also be used to save notes about the application

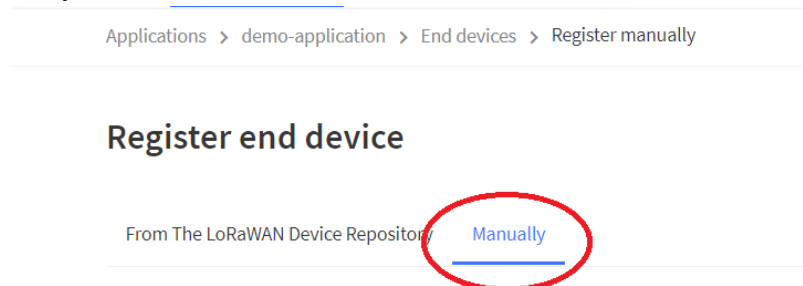
Create application

- Inside Applications Go to End Devices and click on “Add end device”





➤ Go to “Manually” Section



- Give following and click start
  - Select “Activation by personalization (ABP)”
  - LoRaWAN Version: “MAC V1.0.3”
  - Network & Application Server address: “eu1.cloud.thethings.network”
- Enter following details & Enter Network Layer Settings
  - End device ID: <Enter ID for your device> eg.node1
  - DevEUI: <Enter 8byte Unique ID for Device>
  - End Device Name: WDM
  - End Device Description: <give anything for your identification>
- Give Network Layer Settings as follows and Enter Application Layer Settings
  - Frequency Plan: India 865-867 MHz
  - Don’t Select anything for LoRaWAN Class Capabilities. WDM will support only for Class A


**LoRaWAN class capabilities ?**

- ☐ Supports class B
- ☐ Supports class C

- Generate “Device address” automatically by clicking icon mentioned
- Generate “NwkSKey” automatically by clicking icon mentioned
- Go to advanced Settings and enable Reset Frame counters. Keep all other settings default.

**Resets Frame Counters**

☒ Enabled

 Resetting is insecure and makes your device susceptible for replay attacks

- Enter Application Layer settings as follows and Add End Device
  - Don’t Enable Skip payload encryption and decryption

**Skip payload encryption and decryption**

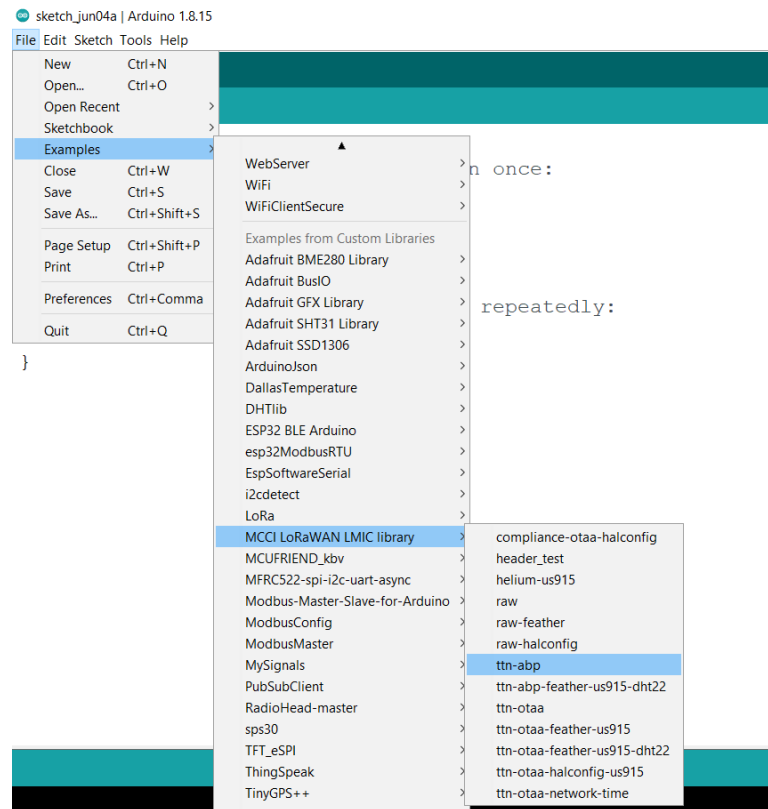
☐ Enabled

Skip decryption of uplink payloads and encryption of downlink payloads

- Generate “AppSKey” automatically by clicking icon mentioned
- Now we have Registered the Application & Device with TTN V3.

## ABP Uplink With WDM to TTN V3

- Open ArduinoIDE 1.8.15 and go to File >> MCCI\_LoRaWAN\_LMIC\_library >> ttn-abp



- Need to change the LoRaWAN Keys (NWKSKEY, APPSKEY, DEVADDR) highlighted below

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

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

// 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 = FILLMEIN ; // <-- Change this address for every node!
```

- Copy the Keys from TTI V3 Devices Page as below and replace it in code in following format

**Note:**

- *Click the eye icon & < > icon to change the view of keys*
- *While copying NwkSKey & AppSKey Ensure the keys are in MSB Position as highlighted in following image*

Overview - WDM - The Things Network | MCC1 LoRaWAN LMIC library - A | +

https://eu1.cloud.thethings.network/console/applications/demo-application/devices/node1

demo-application

- Overview
- End devices
- Live data
- Payload formatters
- Integrations
- Collaborators
- API keys
- General settings

Overview

General information

End device ID: node1

Description: WDM

Created at: Jun 4, 2021 18:27:38

Activation information

AppEUI: n/a

DevEUI: 64 89 29 28 37 AB CD EF

Session information

Device address: 26 0B 9A B0

NwkSKey: 0x0D, 0x88, 0xF9, 0xDB, 0xAC, 0x3E, 0x40, 0x95, 0xD3, 0x58, 0x40, 0x4A, 0xF2, 0xA1, 0x47, 0xE1

SNwkSIntKey: .....

NwkSEncKey: .....

AppSKey: 0x82, 0x90, 0x44, 0xDE, 0xE3, 0x5D, 0xD3, 0xB6, 0x4B, 0x73, 0xB2, 0xA7, 0xE3, 0xBC, 0x72, 0x28

< Hide sidebar

- After Changing the LoRaWAN Keys your code should like this,

```
// LoRaWAN NwkSKey, network session key
// This should be in big-endian (aka msb)
static const PROGMEM u1_t NWKSEKEY[16] = { 0x0D, 0x88, 0xF9, 0xDB, 0xAC, 0x3E, 0x40, 0x95, 0xD3, 0x58, 0x40, 0x4A, 0xF2, 0xA1, 0x47, 0xE1 };

// LoRaWAN AppSKey, application session key
// This should also be in big-endian (aka msb)
static const u1_t PROGMEM APPSKEY[16] = { 0x82, 0x90, 0x44, 0xDE, 0xE3, 0x5D, 0xD3, 0xB6, 0x4B, 0x73, 0xB2, 0xA7, 0xE3, 0xBC, 0x72, 0x28 };

// 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 = 0x260B9AB0; // <-- Change this address for every node!
```

**Note:**

- *Don't enter the above keys as same. Enter the keys as per your TTN account*

- Next, need to change the pin mapping as per WDM board  
Identify the following lines and change the pin map as follows

```
const lm3c_pinmap lm3c_pins = {
    .nss = 15,
    .rxtx = LM3C_UNUSED_PIN,
    .rst = 17,
    .dio = {4, 33, 32},
};
```

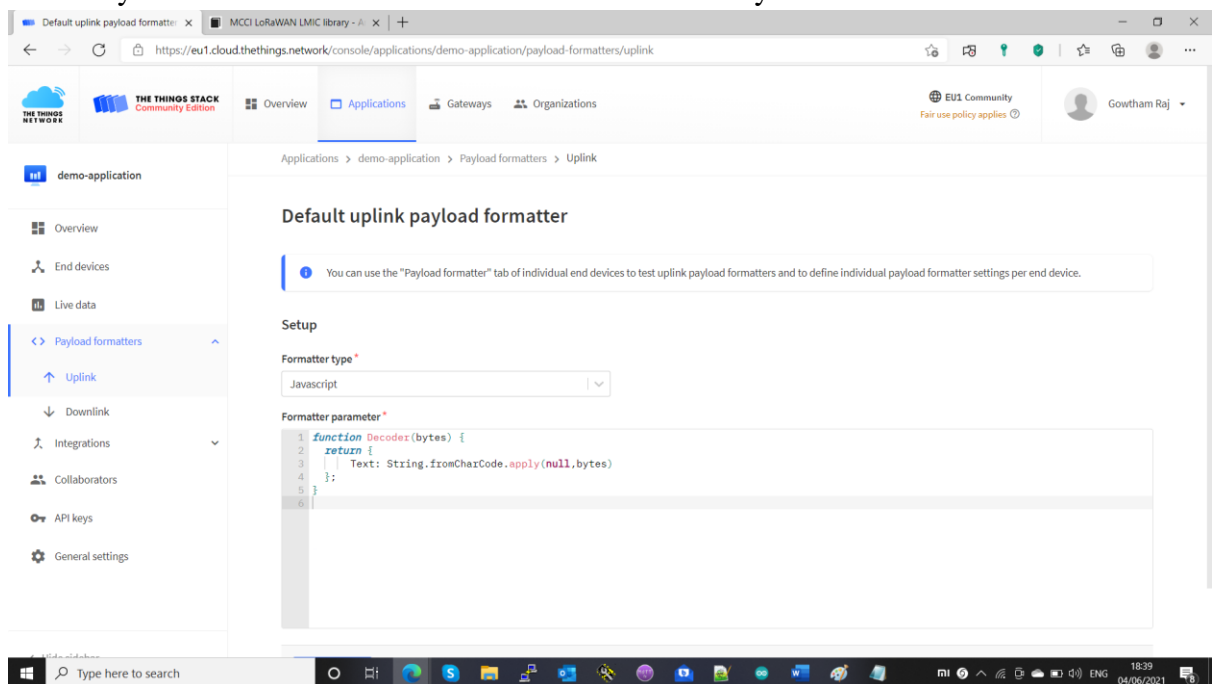
## ➤ Comment following lines to disable European Frequency

```
#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
```

## ➤ Uncomment Following lines to enable Indian Frequency

```
// ... 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);
```

## ➤ Enter Payload Decoder & save it to Decode the received bytes in TTN V3



The screenshot shows the TTN V3 console interface. The left sidebar contains navigation options: Overview, End devices, Live data, Payload formatters (selected), Uplink, Downlink, Integrations, Collaborators, API keys, and General settings. The main content area is titled 'Default uplink payload formatter' and includes a setup section with a dropdown menu for 'Formatter type' set to 'Javascript'. Below this is a code editor showing a JavaScript function:

```
1 function Decoder(bytes) {
2   return {
3     Text: String.fromCharCode.apply(null, bytes)
4   };
5 }
6
```

- Now Upload the sketch in WDM and Reset the device to get data in TTN V3

The screenshot shows the Arduino IDE interface. The sketch is being uploaded to an ESP32 Dev Module. The Serial Monitor is open, showing the output of the sketch. The output includes boot information, configuration parameters, and a successful transmission of a packet.

```

// Next TX is scheduled after TX_COMPLETE event.

}

void setup() {
  // pinMode(13, OUTPUT);
  while (!Serial); // wait for Serial to be initialized
  Serial.begin(115200);
  pinMode(VCC_ENABLE, OUTPUT);
  digitalWrite(VCC_ENABLE, HIGH);
  delay(1000);
  Serial.println(F("Starting"));

  #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
  LMIC_reset();
}

// Serial Monitor Output:
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1216
ho 0 tail 12 room 4
load:0x40078000,len:9720
ho 0 tail 12 room 4
load:0x40080400,len:6352
entry 0x400806b8
Starting
10850: EV_TXSTART
Packet queued
141485: EV_TXCOMPLETE (includes waiting for RX windows)
Leaving...
Hard resetting via RTS pin...
  
```

- Results in TTN V3 as follows

The screenshot shows the TTN V3 console interface. The 'Live data' tab is selected, displaying a table of application data. The table has columns for Time, Entity ID, Type, and Data preview. A single data entry is shown, indicating a successful transmission of a packet.

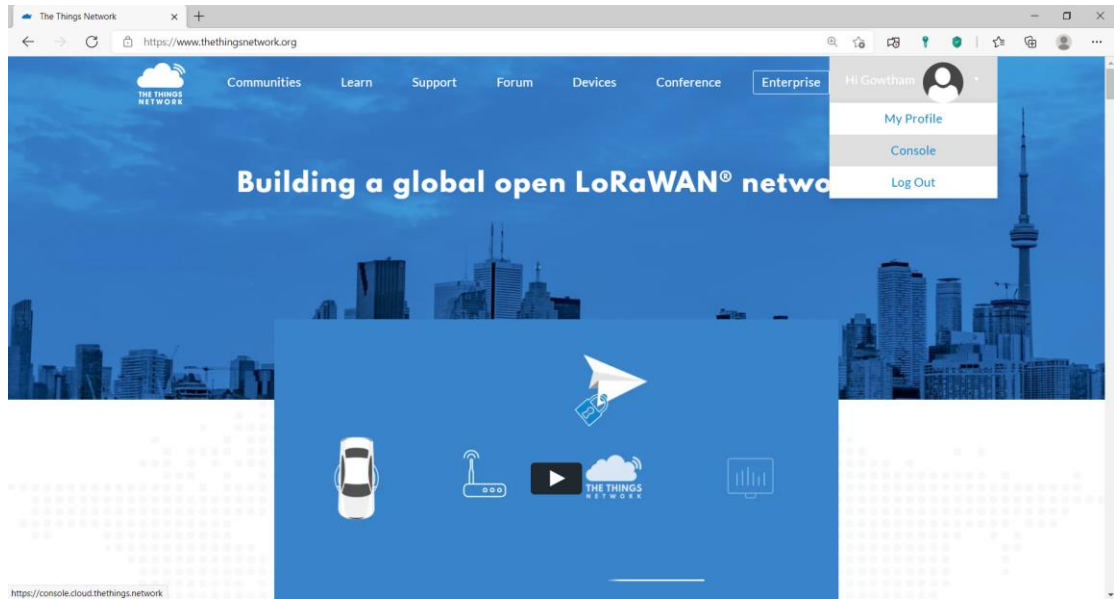
Time	Entity ID	Type	Data preview
19:49:13	node1	Forward uplink data message	Payload: { Text: "Hello, world!" } 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21 FPort: 1 SNR: 10.2 RSSI: -45 Band

**Note:**

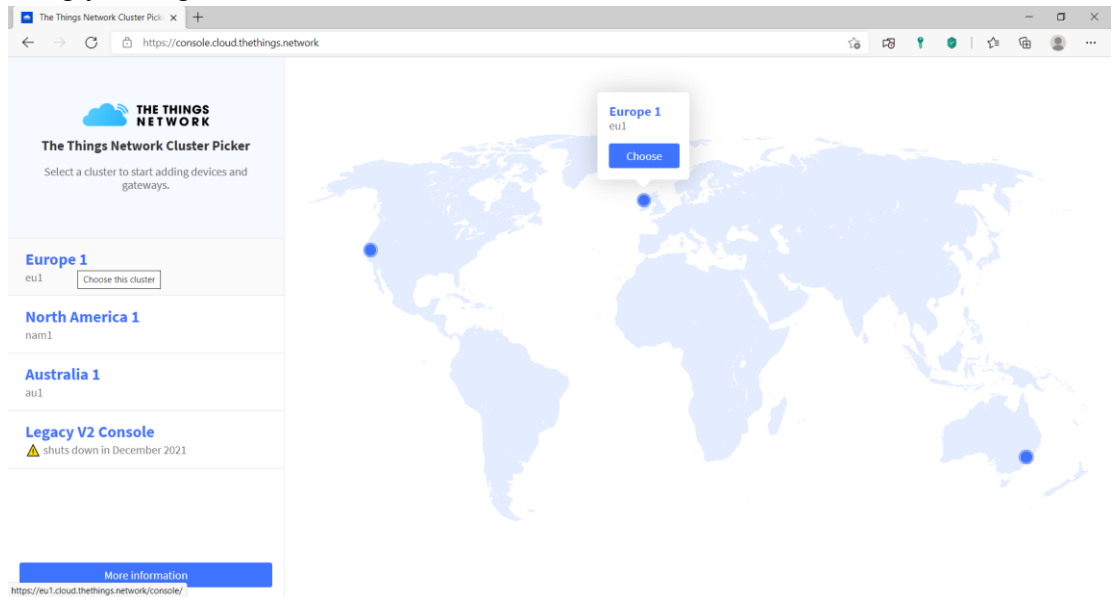
- Gateway should be in Coverage, Live & Connected to TTNv3 for getting data as above

## Registering Application & WDM Device in OTAA mode with TTN v3

- Create a login in “<https://www.thethingsnetwork.org/>” and go to console



- Choose your cluster. I am going to choose Europe 1 & Login with The ThingsID using your registered.



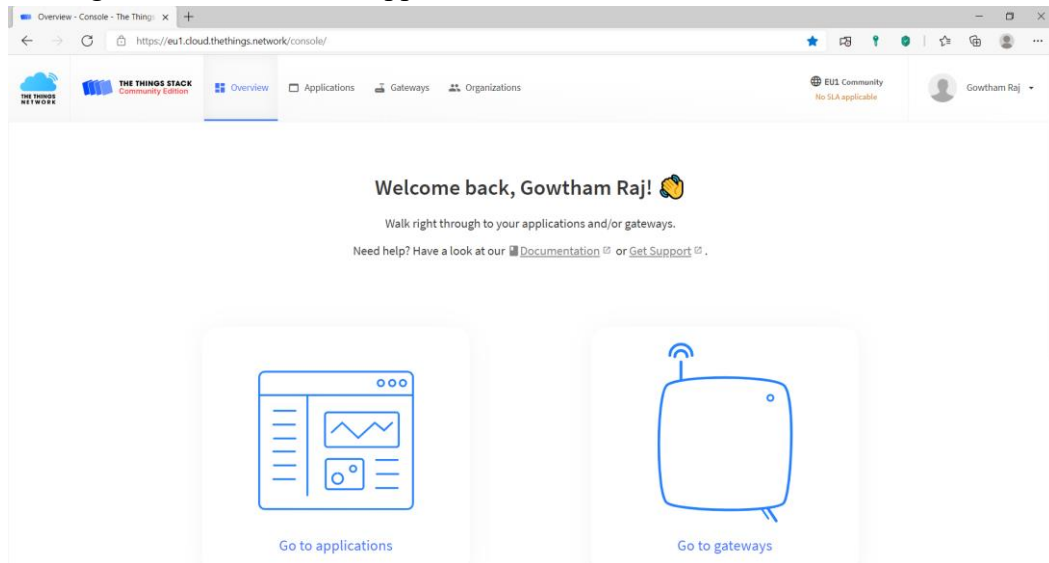
### The Things Network Account

Please login to continue

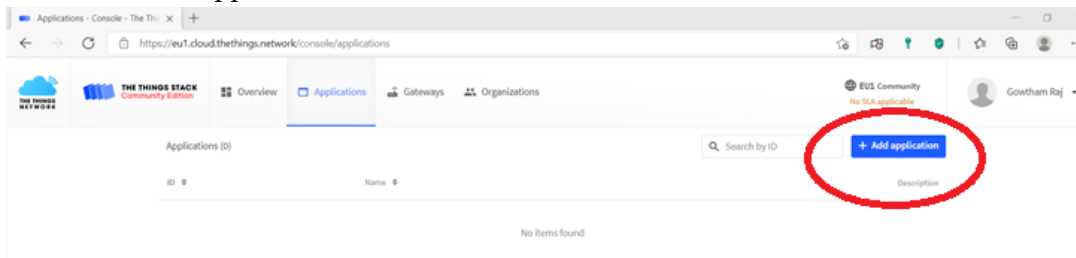
Login with The Things ID

Login using credentials

- After Login, enter into “Go to applications”



- Click on “Add application” and



- Give necessary details and create application.

### Add application

Owner \*

Application ID \*

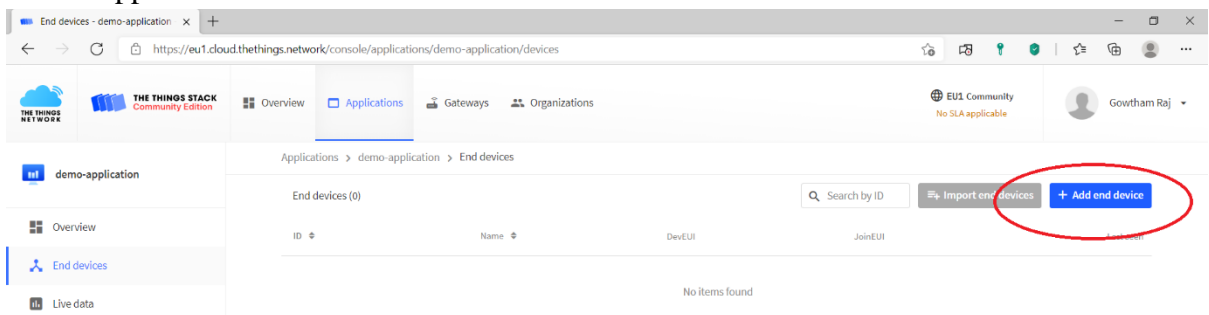
Application name

Description

Optional application description; can also be used to save notes about the application

Create application

- Inside Applications Go to End Devices and click on “Add end device”





➤ Go to “Manually” Section

Applications > demo-application > End devices > Register manually

### Register end device

From The LoRaWAN Device Repository

Manually

➤ Give following and click start

#### Preparation

##### Activation mode \*

- ☒ Over the air activation (OTAA)
- ☐ Activation by personalization (ABP)
- ☐ Multicast
- ☐ Do not configure activation

##### LoRaWAN version ? \*

MAC V1.0.3

##### Network Server address

eu1.cloud.thethings.network

##### Application Server address

eu1.cloud.thethings.network

##### External Join Server ?

☐ Enabled

##### Join Server address

eu1.cloud.thethings.network

Start

➤ Enter following details & Enter Network Layer Settings

- End device ID: <Enter ID for your device> eg.node2
- DevEUI: <Enter 8byte Unique ID for Device>
- End Device Name: WDM
- End Device Description: <give anything for your identification>

➤ Give Network Layer Settings as follows and Enter Join Settings

- Frequency Plan: India 865-867 MHz
- Don't Select anything for LoRaWAN Class Capabilities. WDM will support only for Class A

##### LoRaWAN class capabilities ?

- ☐ Supports class B
- ☐ Supports class C

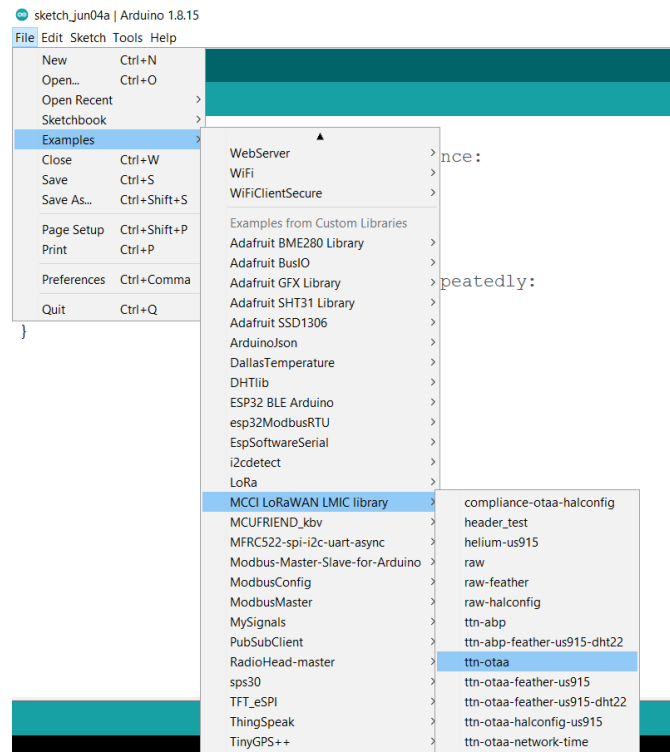
➤ Enter Join settings as follows and Add End Device

- Generate “AppKey” automatically by clicking icon mentioned

➤ Now we have Registered the Application & Device with TTN V3.

## OTAA Uplink With WDM in TTN V3

- Open ArduinoIDE 1.8.15 and go to File >> MCCI\_LoRaWAN\_LMIC\_library >> ttn-otaa



- Need to change the LoRaWAN Keys (APPEUI, DEVEUI, APPKEY) highlighted below

```
// 0x70.
static const ul_t PROGMEM APPEUI[8]={ FILLMEIN };
void os_getArtEui (ul_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const ul_t PROGMEM DEVEUI[8]={ FILLMEIN };
void os_getDevEui (ul_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
static const ul_t PROGMEM APPKEY[16] = { FILLMEIN };
void os_getDevKey (ul_t* buf) { memcpy_P(buf, APPKEY, 16);}
```

- Copy the Keys from TTI V3 Devices Page as below and replace it in code in following format

**Note:**

- *Click the eye icon & < > icon to change the view of keys*
- *While copying APPEUI & DEVEUI Ensure the keys are in LSB Position as highlighted in following image*
- *While copying APPKEY Ensure the keys are in MSB Position as highlighted in following image*

Applications > demo-application > End devices > WDM

**WDM**  
ID: node2

Last seen info unavailable ↑ n/a ↓ n/a

Overview Live data Messaging Location Payload formatters Claiming

**General information**

End device ID node2

Description WDM

Created at Jun 4, 2021 20:11:43

**Activation information**

AppEUI 0x92, 0x28, 0x67, 0x53, 0x76, 0x0... lsb <> [icon]

DevEUI 0x29, 0x88, 0xCD, 0xAB, 0x29, 0x2... lsb <> [icon]

Root key ID n/a

AppKey 0x39, 0xB4, 0x9D, 0x3E, 0x88... msb <> [icon]

NwkKey n/a

**Session information**

No data available

- After Changing the LoRaWAN Keys your code should like this,

```
static const ul_t PROGMEM APPEUI[8]={ 0x92, 0x28, 0x67, 0x53, 0x76, 0x02, 0x49, 0x67 };
void os_getArtEui (ul_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const ul_t PROGMEM DEVEUI[8]={ 0x29, 0x88, 0xCD, 0xAB, 0x29, 0x20, 0x29, 0x68 };
void os_getDevEui (ul_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
static const ul_t PROGMEM APPKEY[16] = { 0x39, 0xB4, 0x9D, 0x3E, 0x88, 0xF8, 0x4F, 0x64, 0xF6, 0x23, 0xB6, 0x4B, 0x33, 0xD1, 0x4A, 0x1F };
void os_getDevKey (ul_t* buf) { memcpy_P(buf, APPKEY, 16);}
```

**Note:**

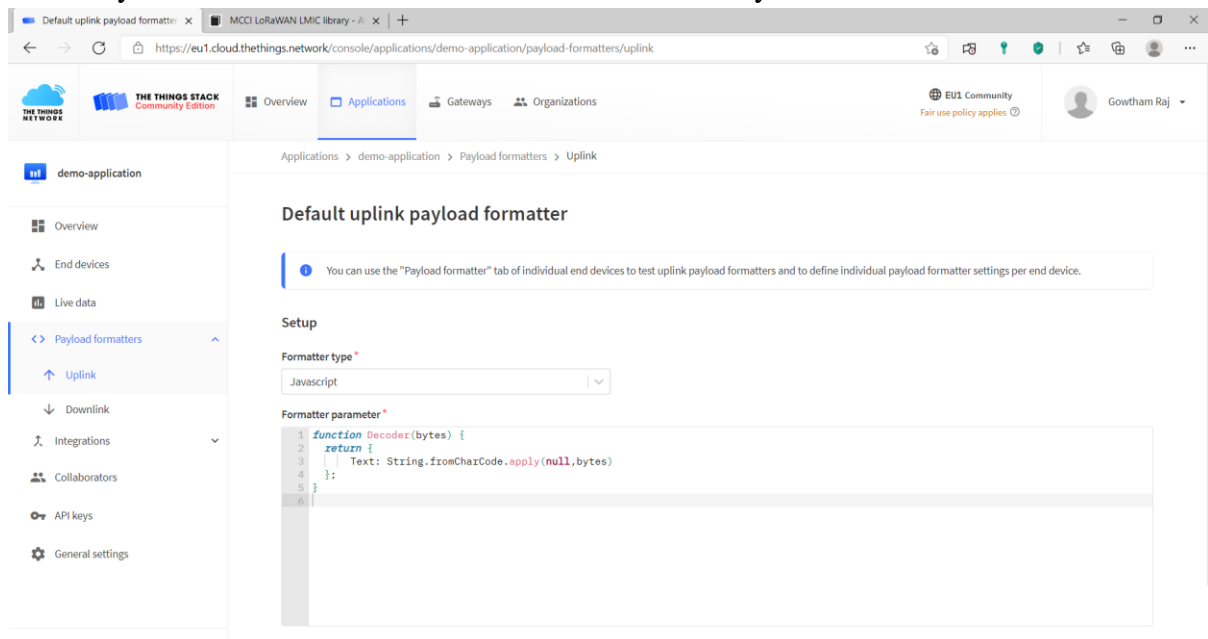
- **Don't enter the above keys as same. Enter the keys as per your TTN account**

- Next, need to change the pin mapping as per WDM board  
Identify the following lines and change the pin map as follows

```
const lm323_pinmap lm323_pins = {
    .nss = 15,
    .rxtx = LM323_UNUSED_PIN,
    .rst = 17,
    .dio = {4, 33, 32},

};
```

## ➤ Enter Payload Decoder & save it to Decode the received bytes in TTN V3



Default uplink payload formatter

You can use the "Payload formatter" tab of individual end devices to test uplink payload formatters and to define individual payload formatter settings per end device.

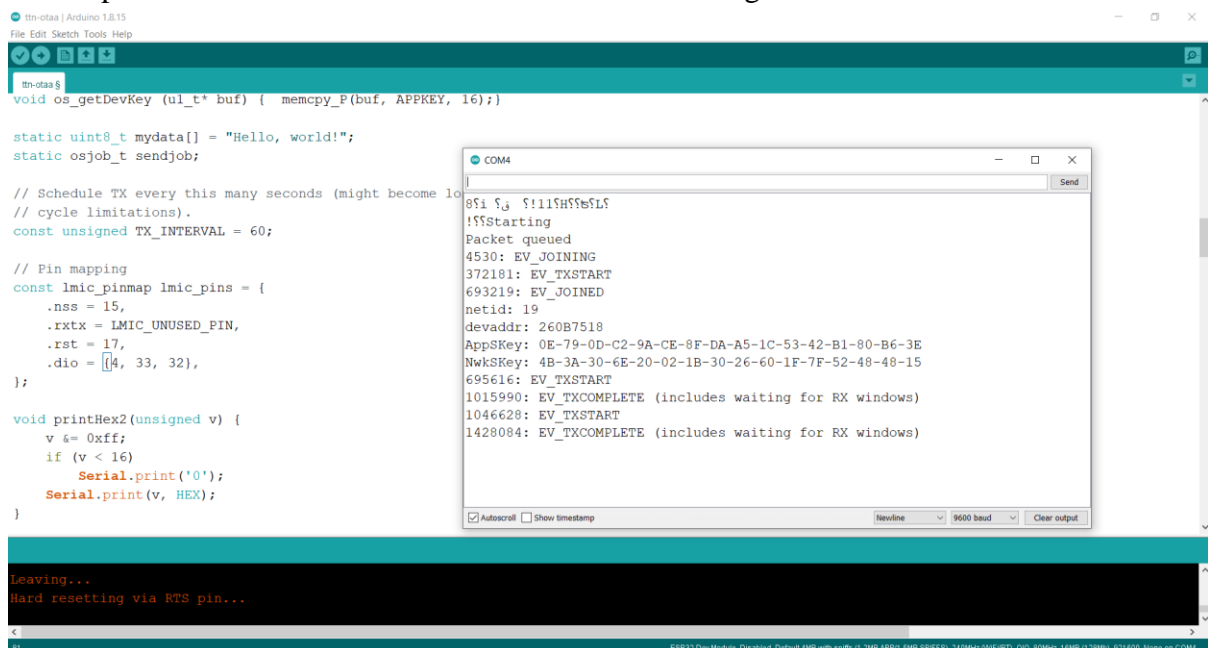
**Setup**

Formatter type\*  
Javascript

Formatter parameter\*

```
1 function Decoder(bytes) {
2   return {
3     Text: String.fromCharCode.apply(null, bytes)
4   };
5 }
6
```

## ➤ Now Upload the sketch in WDM and Reset the device to get data in TTN V3



```
void os_getDevKey (ul_t* buf) { memcpy_P(buf, APPKEY, 16);}

static uint8_t mydata[] = "Hello, world!";
static osjob_t sendjob;

// Schedule TX every this many seconds (might become lo
// cycle limitations).
const unsigned TX_INTERVAL = 60;

// Pin mapping
const lmic_pinmap lmic_pins = {
  .nss = 15,
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 17,
  .dio = {4, 33, 32},
};

void printHex2(unsigned v) {
  v &= 0xff;
  if (v < 16)
    Serial.print('0');
    Serial.print(v, HEX);
}

leaving...
Hard resetting via RTS pin...
```

COM4

```
051 53 5115H555L5
!55Starting
Packet queued
4530: EV_JOINING
372181: EV_TXSTART
693219: EV_JOINED
netid: 19
devaddr: 260B7518
AppSKey: 0E-79-0D-C2-9A-CE-8F-DA-A5-1C-53-42-B1-80-B6-3E
NwkSKey: 4B-3A-30-6E-20-02-1B-30-26-60-1F-7F-52-48-48-15
695616: EV_TXSTART
1015990: EV_TXCOMPLETE (includes waiting for RX windows)
1046628: EV_TXSTART
1428084: EV_TXCOMPLETE (includes waiting for RX windows)
```

## ➤ Results in TTN V3 as follows

The screenshot displays the 'The Things Stack Community Edition' interface. The top navigation bar includes 'Overview', 'Applications' (selected), 'Gateways', and 'Organizations'. The user 'Gowtham Raj' is logged in. The left sidebar shows the application 'demo-application' with sub-options: Overview, End devices, Live data (selected), Payload formatters, Integrations, Collaborators, API keys, and General settings. The main area shows the 'Live data' view for 'demo-application'. It features a table with columns: Time, Entity ID, Type, and Data preview. The table contains three entries: a 'Forward uplink data message' at 20:34:14, and two 'Accept join-request' events at 20:34:08 and 20:33:58, all for 'node2'. A console message at 19:35:03 states 'Events cleared' and 'The events list has been cleared'. The 'Data preview' for the first entry shows a JSON payload: { Text: "Hello, world!" } and various signal metrics.

Time	Entity ID	Type	Data preview
20:34:14	node2	Forward uplink data message	Payload: { Text: "Hello, world!" } 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21 FPort: 1 SNR: 10.5 RSSI: -64 Band
20:34:08	node2	Accept join-request	
20:33:58	node2	Accept join-request	
19:35:03		Console: Events cleared	The events list has been cleared

**Note:**

- Gateway should be in Coverage, Live & Connected to TTNv3 for getting data as above