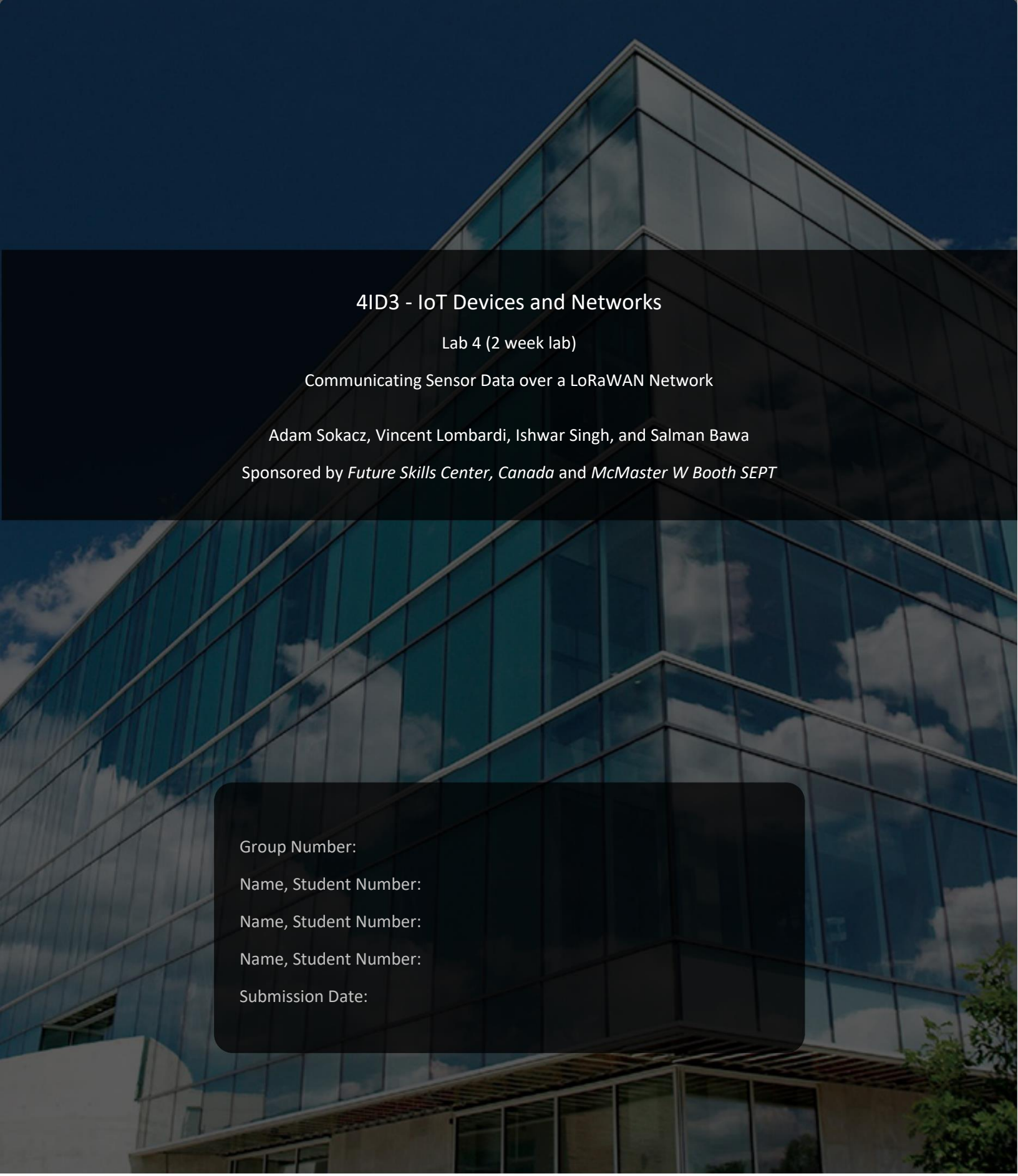


4ID3 - IoT Devices and Networks

Lab 4 (2 week lab)

Communicating Sensor Data over a LoRaWAN Network

Adam Sokacz, Vincent Lombardi, Ishwar Singh, and Salman Bawa

Sponsored by *Future Skills Center, Canada* and *McMaster W Booth SEPT*

Group Number:

Name, Student Number:

Name, Student Number:

Name, Student Number:

Submission Date:

W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY

McMaster-Mohawk Bachelor of Technology Partnership

McMaster University ENGINEERING

mohawk

Objective

In this lab, sensor data will be collected on the MKRWAN development board and transmitted over LoRaWAN to The Things Stack using a LoRaWAN Indoor Gateway as an intermediary. The Application Layer will be integrated with MQTT. Each uplink will be automatically committed to a local MySQL server for data collection and analysis.

Contents

Feedback

Q1 - What would you rate the difficulty of this lab?

(1 = easy, 5 = difficult)

1 **2** **3** **4** **5**

Comments about the difficulty of the lab:

Q2 - Did you have enough time to complete the lab within the designated lab time?

YES **NO**

Q3 - How easy were the lab instructions to understand?

(1 = easy, 5 = unclear)

1 **2** **3** **4** **5**

List any unclear steps:

Q4 - Could you see yourself using the skills learned in this lab to tackle future engineering challenges?

(1 = no, 5 = yes)

1 **2** **3** **4** **5**

Additional Resources

Lab GitHub Repo (https://github.com/sokacza/4ID3)

MySQL Basics (https://youtu.be/Cz3WcZLRaWc)

NodeRED Fundamentals Tutorial (https://youtu.be/3AR432bguOY)

Arduino MKRWAN (https://youtu.be/UmcA2moPWAY)

Pre-Lab Questions

Q1 - Name 6 advantages of LoRaWAN over other access technologies.

(Suggested: Short paragraph)

Q2 - LoRaWAN specifies 3 types of devices: Class A, Class B, and Class C. Compare and contrast each in a table.

(Suggested: Table)

Q3 - LoRaWAN end devices support 2 activation methods. Name these methods. Compare and contrast them in a table.

(Suggested: Table)

Q4 - What is The Things Network and how does it differ from LoRaWAN itself? What layer of the OSI model does each lie?

(Suggested: Short paragraph)

Post-Lab Questions

Q1 - Draw a diagram to identify each component of the IoT network produced in this lab and describe the information being exchanged between the components.

(Suggested: Sketch)

Q2 - What is Cisco SD-WAN? Compare and contrast it with The Things Network in a table.

(Suggested: Table)

Q3 - You have a database named `KGB_TOP_SECRET` and a table in that database named `LOGIN_CREDENTIALS`. What is the SQL query to **use** that database?

(Suggested: 1 Sentence)

Q4 - You have a database named `KGB_TOP_SECRET` and a table in that database named ` LOGIN_CREDENTIALS `. What is the SQL query to **select**/print all columns of that table to the terminal?

(Suggested: 1 Sentence)

Q5 - You have a database named `KGB_TOP_SECRET` and a table in that database named ` LOGIN_CREDENTIALS `. There are two columns in this table: `USERNAME`, `PASSWORD`. What is the SQL query to **insert** a new set of **values**, a row into the table that includes a username and password string fields? (Keep in mind ` symbol vs ‘ symbol)

username = ‘#TheRealJamesBond’

password = ‘007’

(Suggested: 1 Sentence)

Q8 – We can use SQL Queries to **connect**, **insert** data into, and **read** from a MySQL database. You have the following MySQL database:

IOT_DEVICES			
ID	DEVICE_NAME	SENSOR_NAME	SENSOR_VALUE
1	DeviceC	Temperature	24.6
2	DeviceF	Temperature	28.9
3	DeviceU	Temperature	102.3

Notice that the temperature in row #3 is much higher than other devices. Write a **SQL query** that will **select** all the database rows **where** a temperature is **greater than 30.0** degrees.

Expected output:

IOT_DEVICES			
ID	DEVICE_NAME	SENSOR_NAME	SENSOR_VALUE
3	DeviceU	Temperature	102.3

(Suggested: 1 – 2 sentences)

Q6 - We saw these terms a lot in this lab: **EUI**, **API Key**. What do each of these terms mean in the context of LoRa and LoRaWAN communication?

(Suggested: Short paragraph)

Q7 - In this lab, we used the **915 MHz** LoRaWAN frequency, which is a legal frequency band in North America. Using online resources, calculate the theoretically suggested antenna length when transmitting at this frequency. Show your calculations below.

(Suggested: Short paragraph)

Q6 – When looking at a LoRaWAN **data frame**, you notice that you receive the **payload data** as a **hex character string**.

[6c, 6f, 72, 61, 77, 61, 6e, 20, 72, 6f, 63, 6b, 73]

a) Convert each hex character to decimal form using the following online calculator:

https://www.rapidtables.com/convert/number/hex-to-decimal.html

[…]

b) Convert each decimal character to its ASCII representation using the following online calculator:

https://onlineasciitools.com/convert-decimal-to-ascii

[…]

Q8 - Write a brief LinkedIn post about key learning takeaways from this lab.

(Suggested: Short paragraph)

Exercise A Results:

Exercise B Results:

Exercise C Results:

Setting up the Workspace

Using Git Bash or Windows Terminal, navigate to your lab repository and issue the following commands:

git pull origin main

<Create your Lab04 folder>

<Create your Lab4 README.md file>

git add .

git commit -m "Starting Lab04"

git push origin main

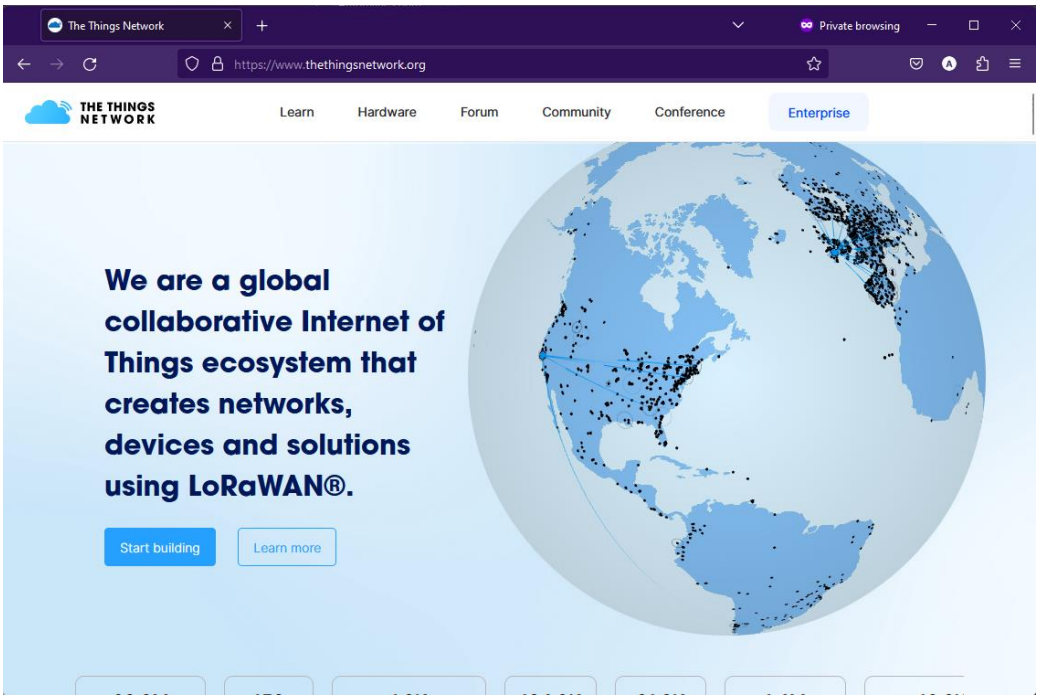
git pull origin main

Setting Up a LoRaWAN Gateway

Logging into ThingStack

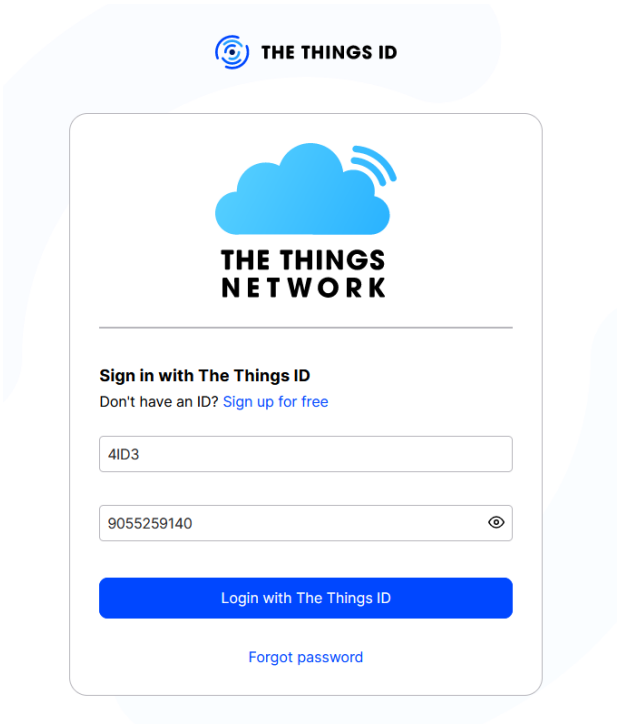
Navigate to The Things Network:

https://www.thethingsnetwork.org/



Press **Login**.

Use the credentials provided by the lab instructor.



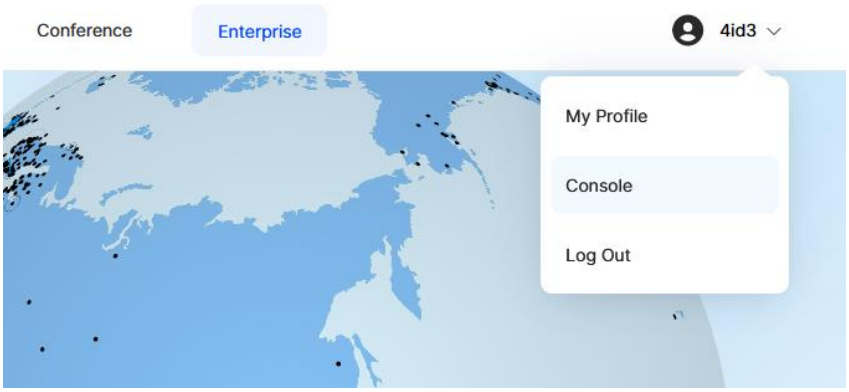
Login Credentials:

Username: 4ID3

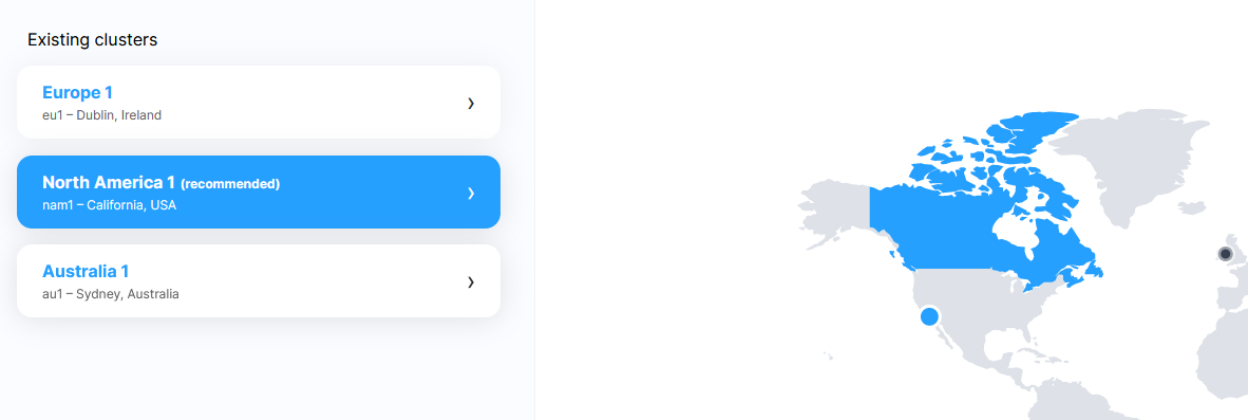
Password: 9055259140

**McMaster general phone number*

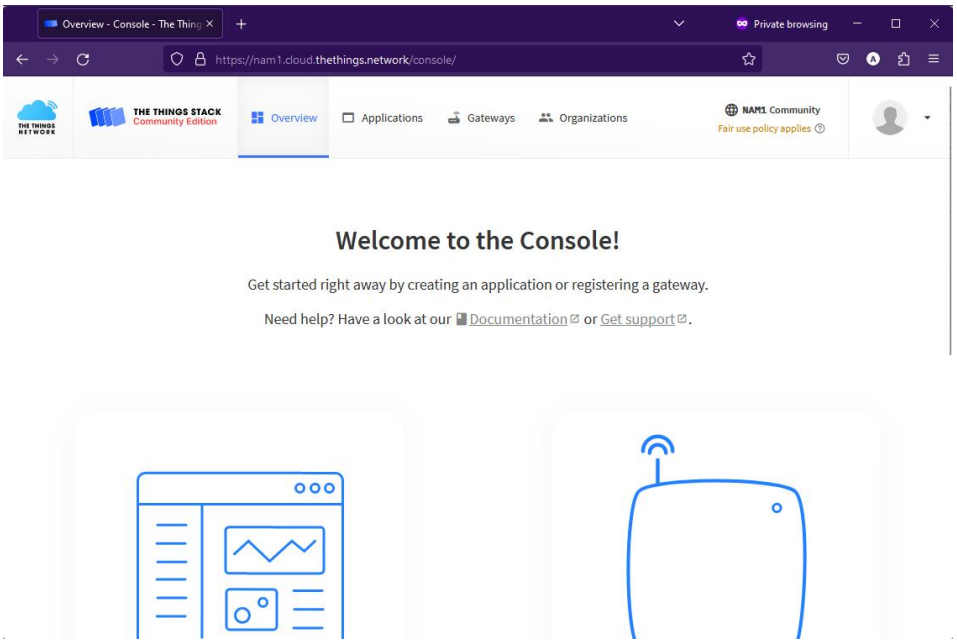
Once logged in, navigate to the **Console**.



Select the **North American Cluster**.



You will be redirected to the welcome page.



Notice the **region** in the top right corner.

Connecting the Gateway to The Things Stack

Gateway Configuration

Keep the RESET button (small button at the back of the gateway next to the USB-C port) pressedfor 5 seconds until the LED blinks rapidly from GREEN to RED and vice versa for a couple of times.

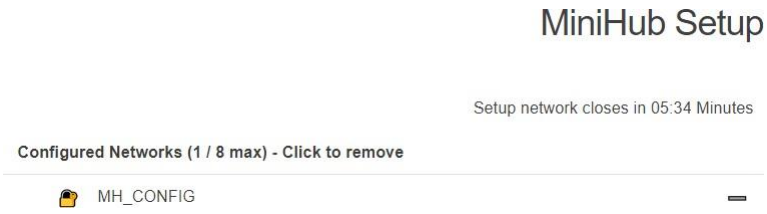
Hold the **setup button** (at the top of the gateway, next to the LED) for **10 seconds** until the LED **rapidly blinks RED**.

The gateway now exposes a **Wi-Fi access point** whose **SSID** is **MINIHUB-xxxxxx**, where xxxxxx is the last **6 digits of the gateway EUI**. The password for this network is on the back of the device.

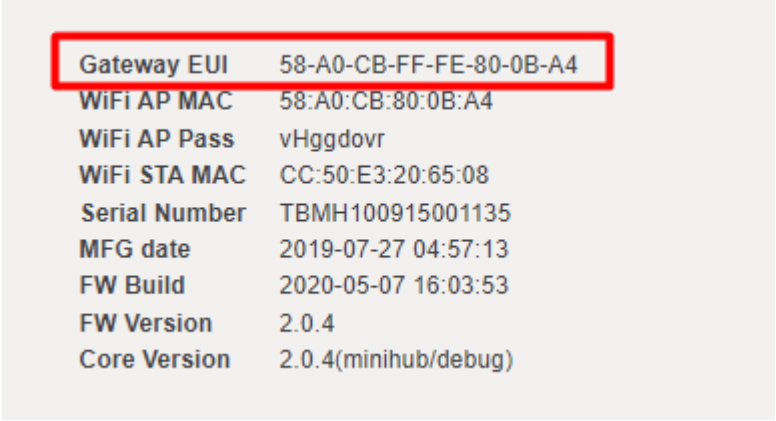
Connect to the network on your phone on computer. Note: If you are connecting to the devicedirectly on your computer you will lose access to the Internet since you are communicating directly with the device.

Once connected, go to the following **URL** in your browser: http://192.168.4.1

A page like the following should load:

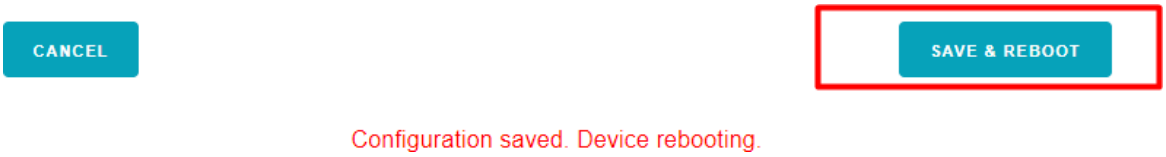


IMPORTANT: Take note of the **Gateway EUI** at the bottom of the page. This is necessary for adding the gateway to The Things Stack in the next steps. It is recommended to copy this table and save it to a notepad file on your desktop.



Find your Wi-Fi network in the list of scanned networks and enter your Wi-Fi password then press **Save & Reboot.**

You should see a prompt like this:



If your configuration is correct,

The gateway will blink **GREEN** for a few seconds until it connects to the selected WiFi network.

Then, it will blink from **GREEN to RED** and vice versa for a few seconds while it connects to the server and fetches the necessary configuration.

Please allow **5-10 minutes** for the gateway to pick up the **new configuration**.

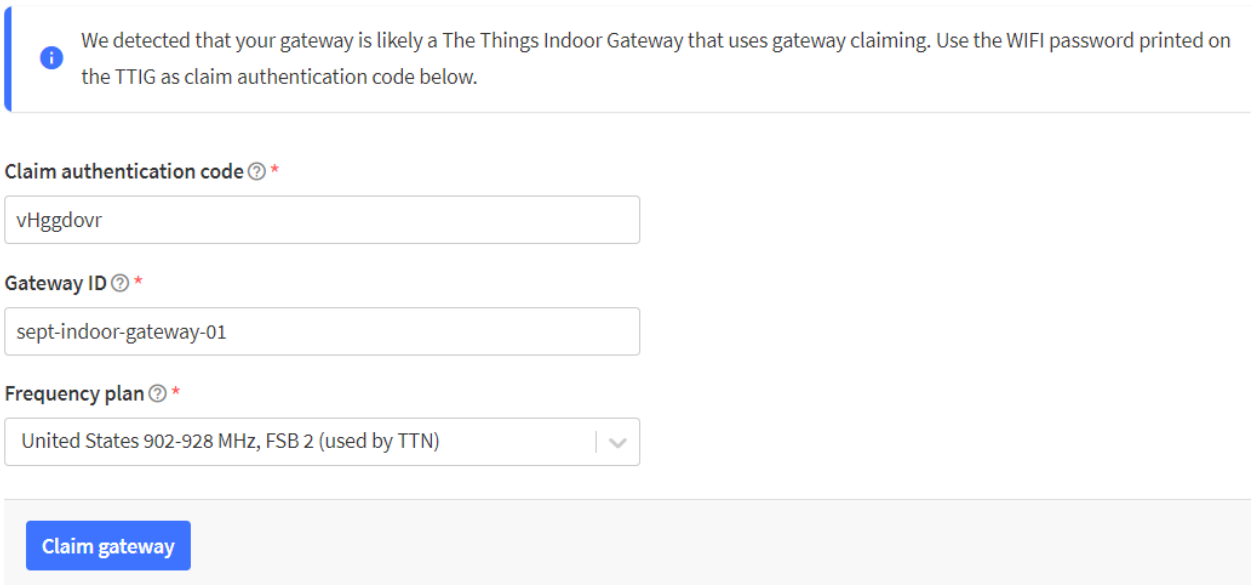
Registering the Gateway

Now that the Gateway is connected to our internet, we can register it in the console.

Go to **Gateways** in the top menu, and click **+ Register Gateway** to reach the gatewayregistration page.

Enter the **Gateway EUI** you copied earlier.

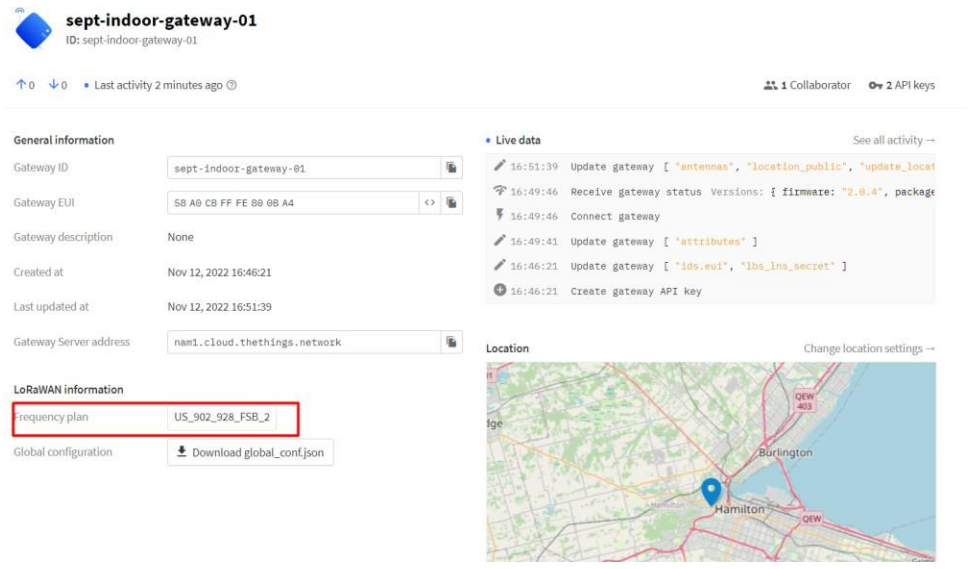
Use the **Wi-Fi password** of the device that was previously used to connect to its network as the **Claim Authentication Code**. Then give the gateway a unique ID/name and set the Frequency Plan to United States **902-928 MHz, FSB 2.**



If your inputs are correct, a new gateway will be created, and you will be redirected to the gateway overview page of your newly created gateway.

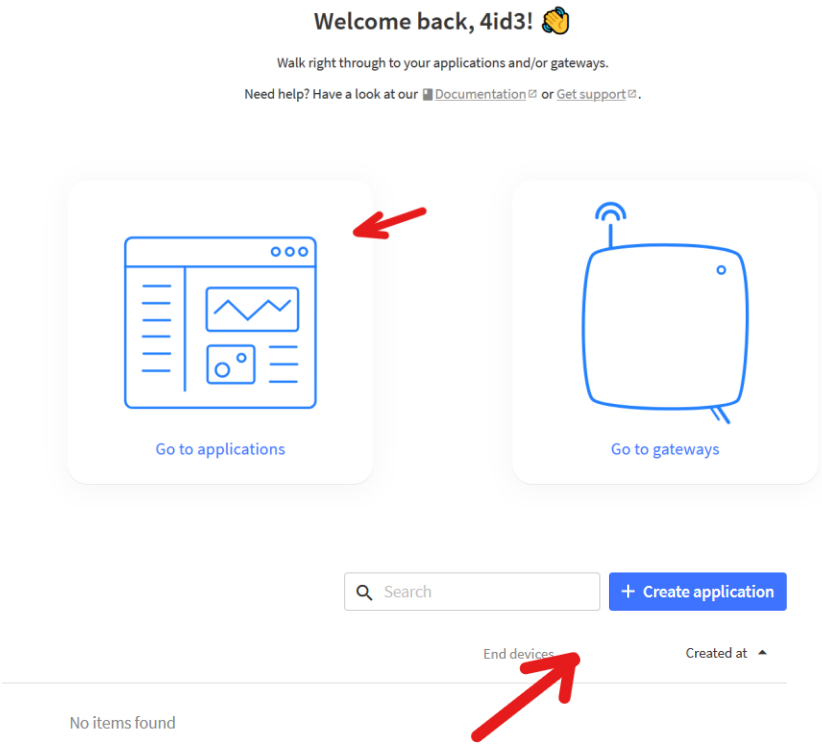
If this is the first time your gateway is being powered on/connected to WiFi, it might pick up a **new firmware** depending on when it was last updated. This is indicated by **alternating GREEN/RED blinks** of the LED. Please **leave the gateway powered** on when this happens.

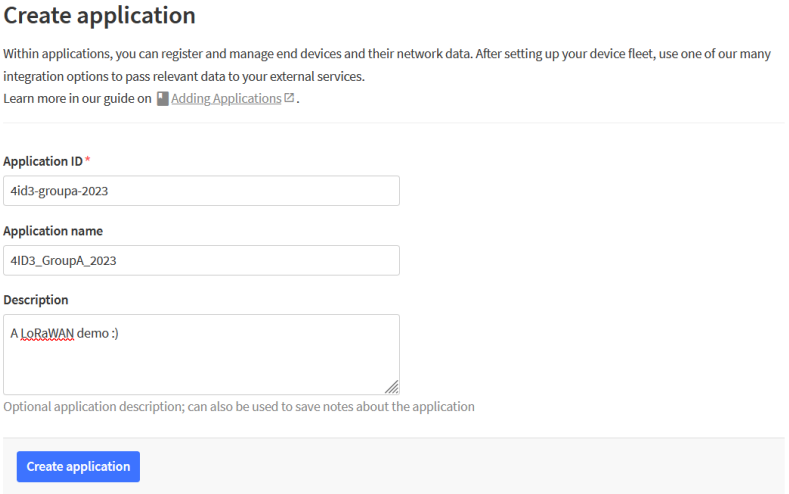
After a **few minutes**, the light on the Indoor Gateway should go **solid green**, and you should see the flow of packets in the console.

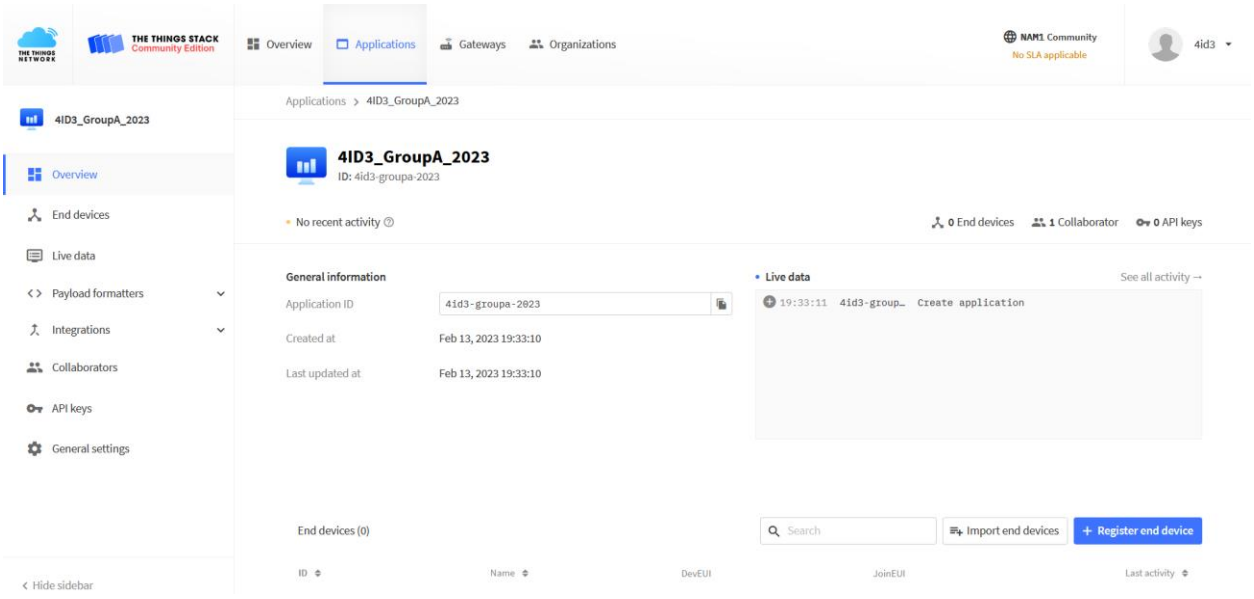


Creating an Application

Go to **Applications** in the top menu, and click **+ Add Application** to reach the application registration page. Fill the application ID. The other fields are optional. In this example, we aremaking an application called **4ID3_GroupA_2023.**






THE THINGS STACK
Community Edition
Overview
Applications
Gateways
Organizations
NAM1 Community
No SLA applicable
4id3
4ID3_GroupA_2023
Overview
End devices
Live data
Payload formatters
Integrations
Collaborators
API keys
General settings
Hide sidebar
Applications > 4ID3_GroupA_2023
4ID3_GroupA_2023
ID: 4id3-groupa-2023
No recent activity
0 End devices
1 Collaborator
0 API keys
General information
Application ID
4id3-groupa-2023
Created at
Feb 13, 2023 19:33:10
Last updated at
Feb 13, 2023 19:33:10
Live data
See all activity →
19:33:11 4id3-group… Create application
End devices (0)
Search
Import end devices
Register end device
ID
Name
DevEUI
JoinEUI
Last activity

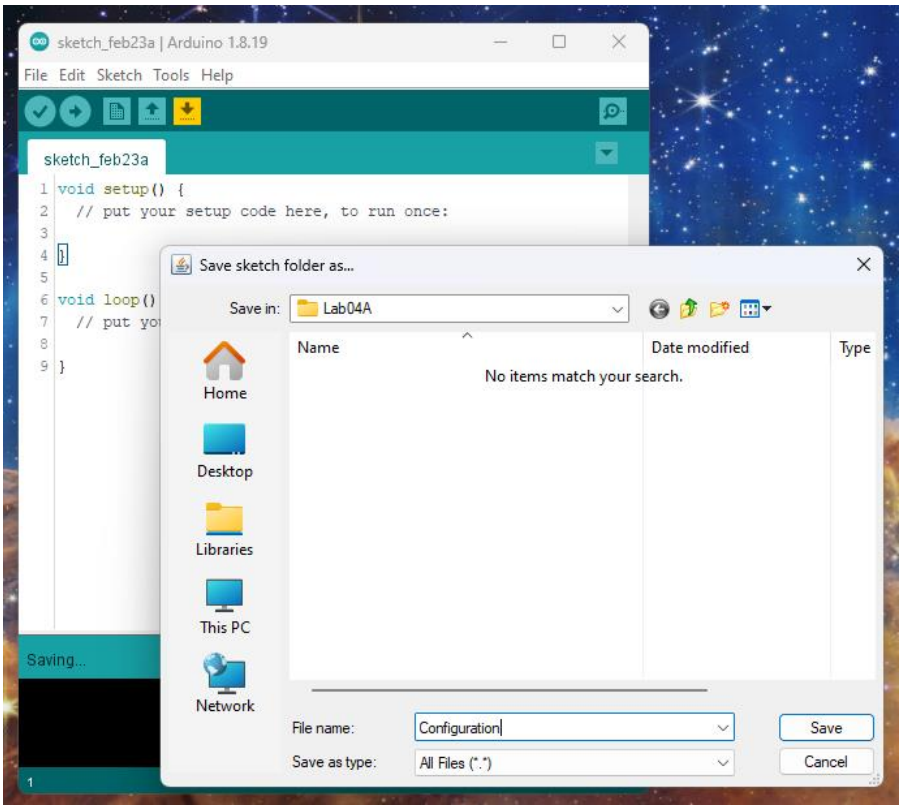
Creating a LoRaWAN Field Device

Finding your Device Configuration

There are a few libraries we need for this project:

- TheThingsNetwork
  - https://github.com/TheThingsNetwork/arduino-device-lib
  - Download and Install as ZIP
- Adafruit Unified Sensor by Adafruit Inc.
  - Install from Arduino library manager
- DHT Sensor Library by Adafruit Inc.
  - Install from Arduino library manager

Create a new Arduino project called **Configuration**.

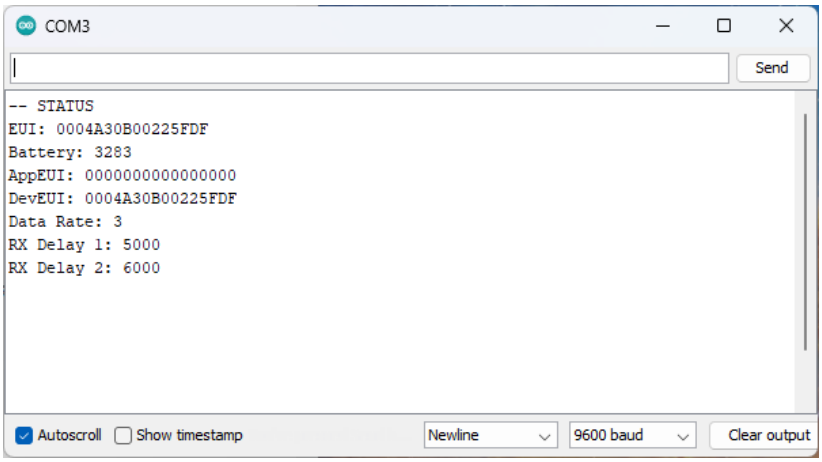


Set up your sketch as such:

Configuration

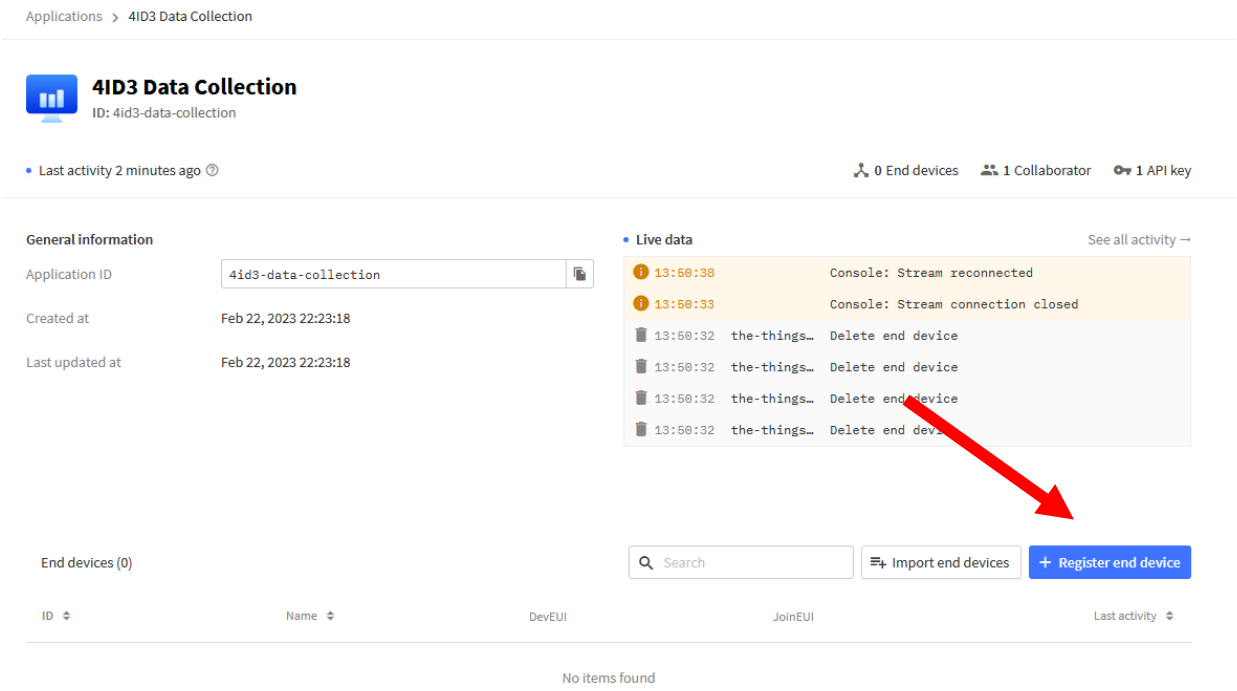
```
#include <TheThingsNetwork.h>

#define loraSerial Serial1
#define debugSerial Serial
#define freqPlan TTN_FP_US915

TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);

void setup(){
  loraSerial.begin(57600);
  debugSerial.begin(9600);

  while (!debugSerial) {}

  debugSerial.println("-- STATUS");
  ttn.showStatus();
}
void loop()
{
}

```

Upload it to your board, open the **serial monitor** and note the Device **EUI**.

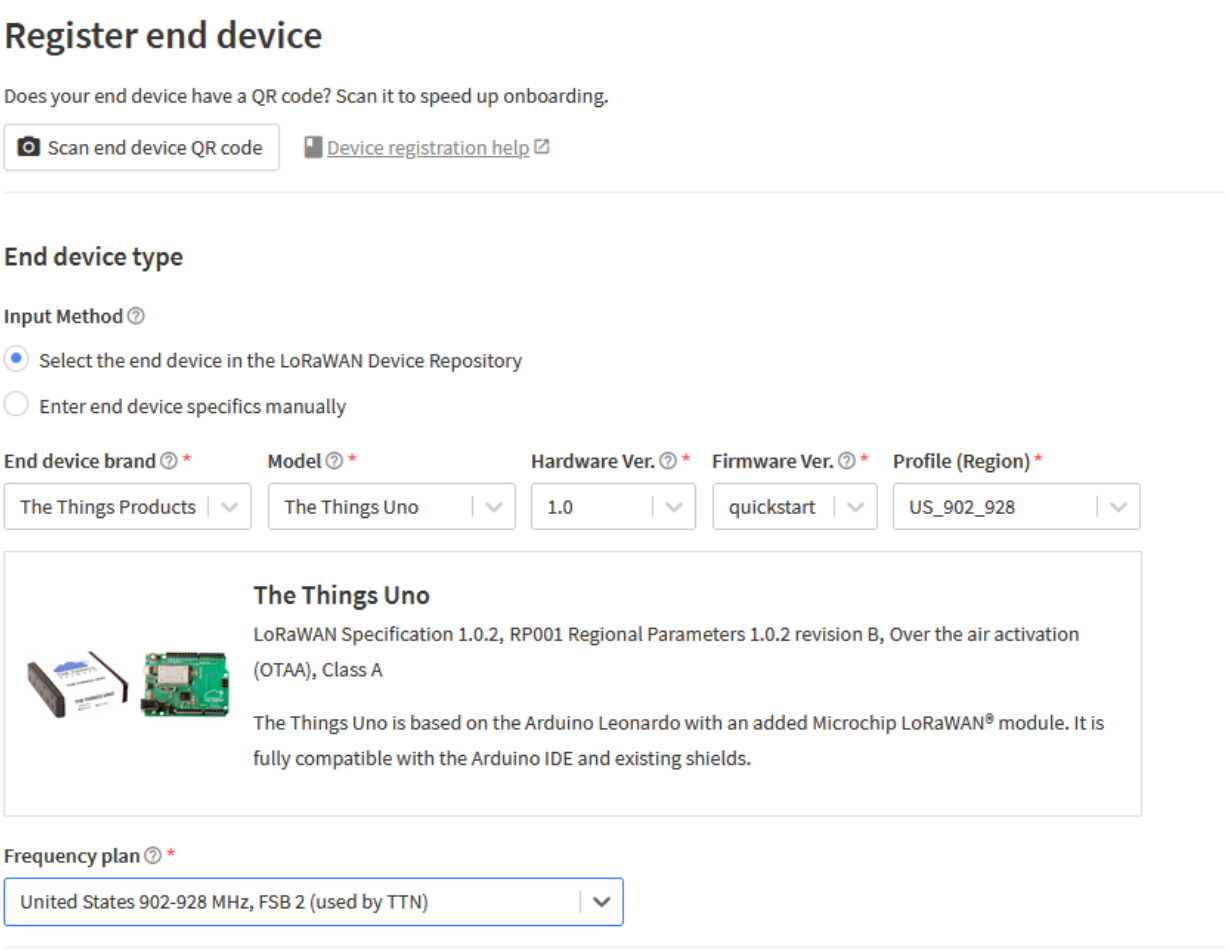


Registering your Device in your Application

Navigate to your application dashboard and click **Register End Device**.



Use the following configuration:



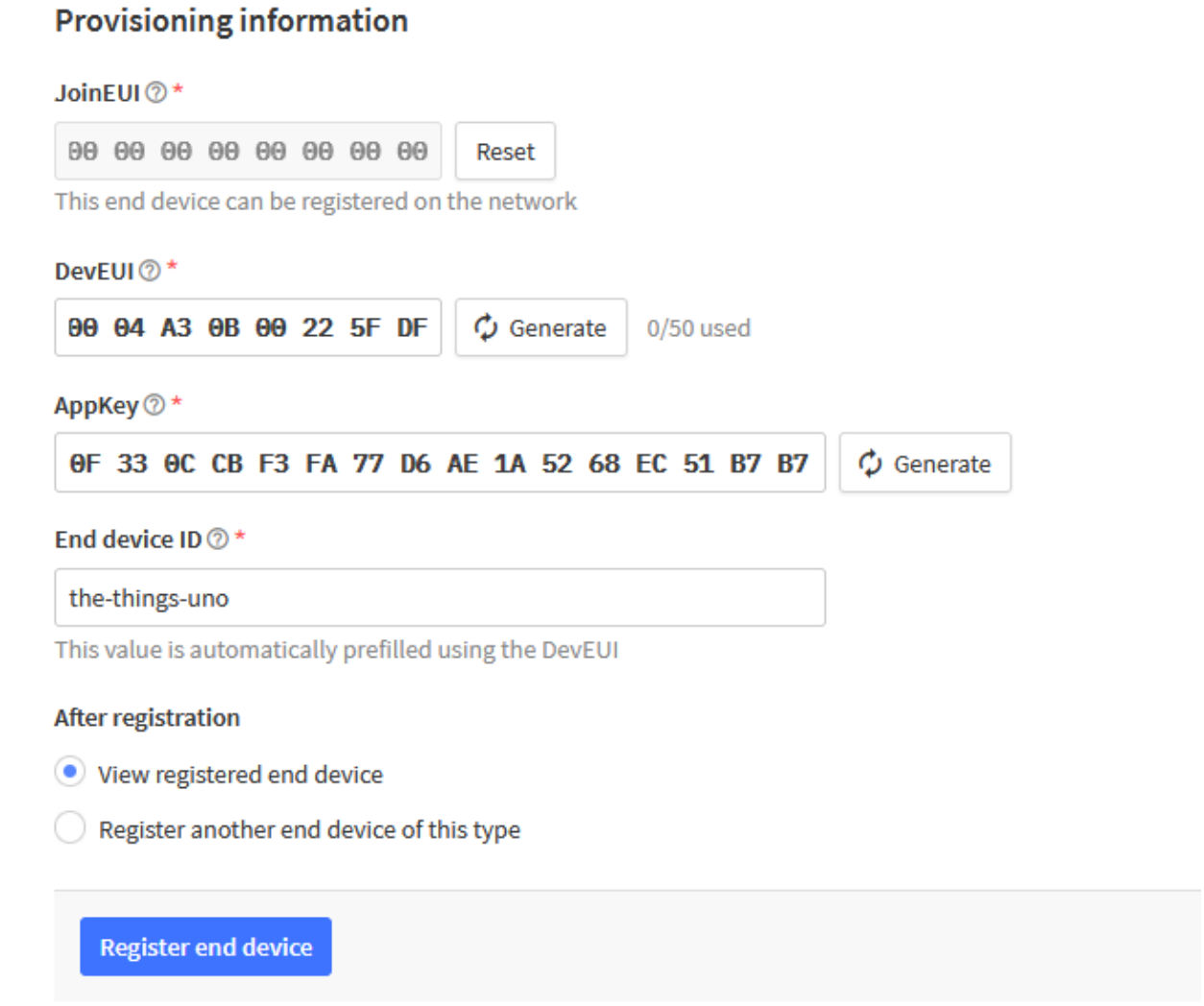
For the **Provisioning Information**:

JoinEUI - All 0's

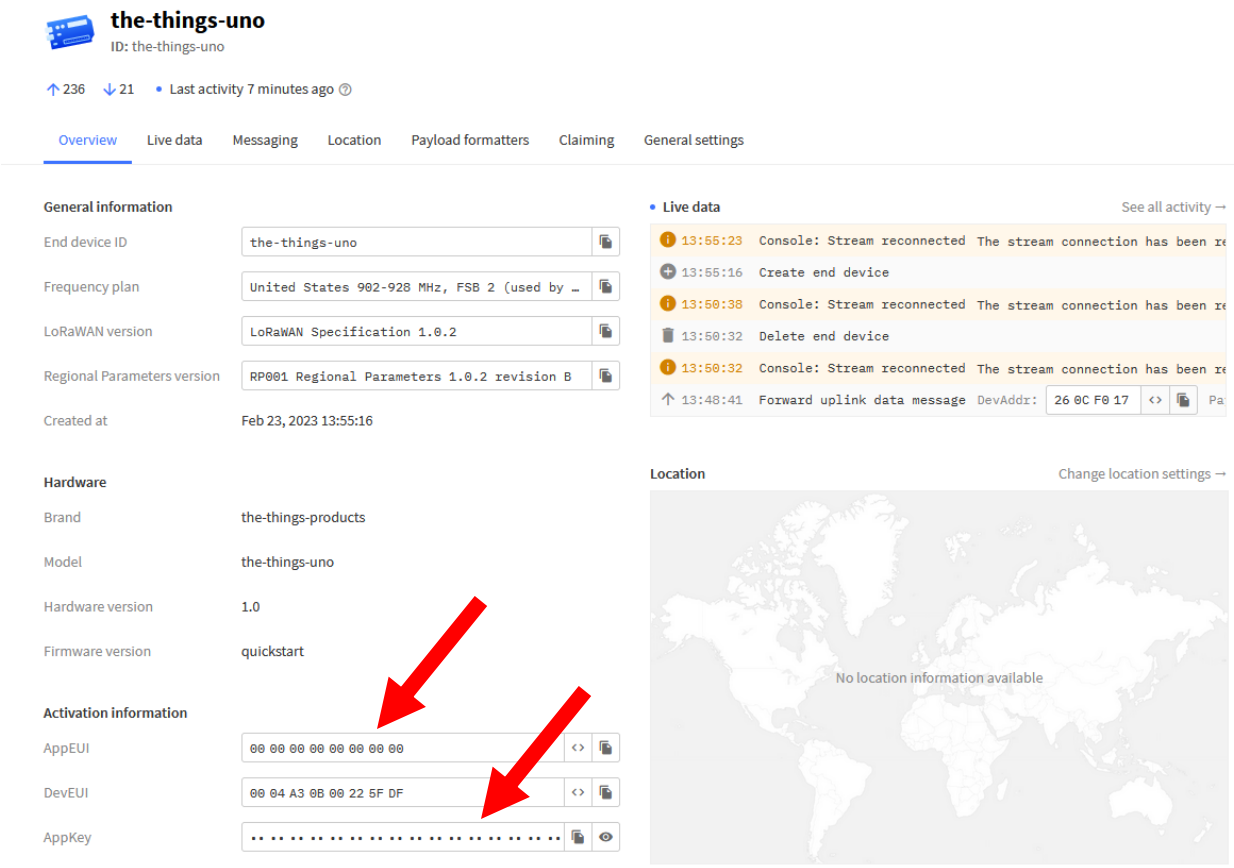
DevEUI - From Arduino Configuration Script

AppKey - Generate

End Device ID - Write a unique one, suggested format: **4id3-groupa-devicea**



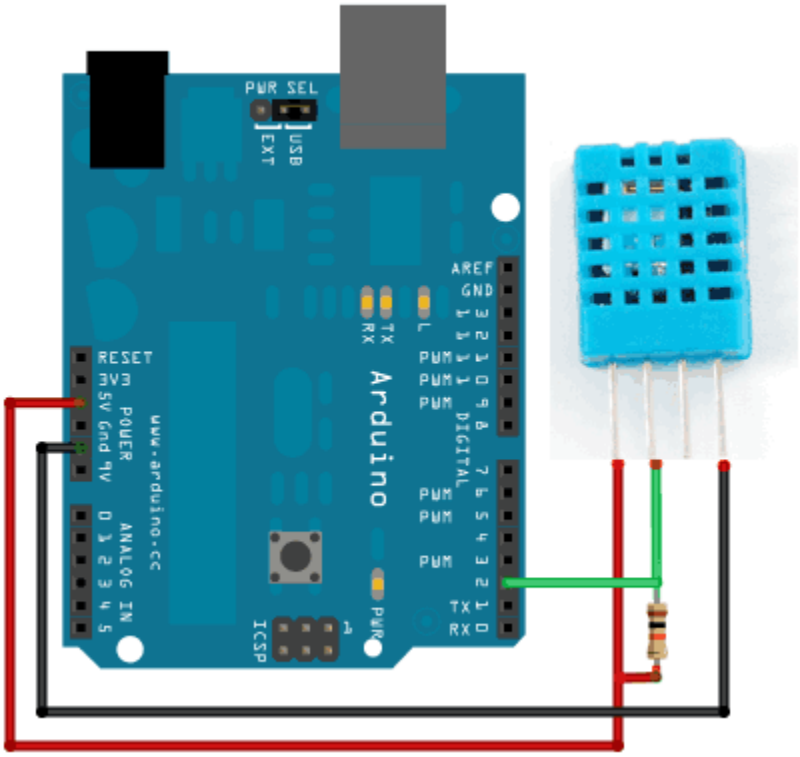
Press **Register End Device**.



Note down your **APPEui** and **APPKey**. You will need these values for your **Field Device** to connect to The Things Network.

Field Device

Make the following connections on your **The Things Uno** development board:

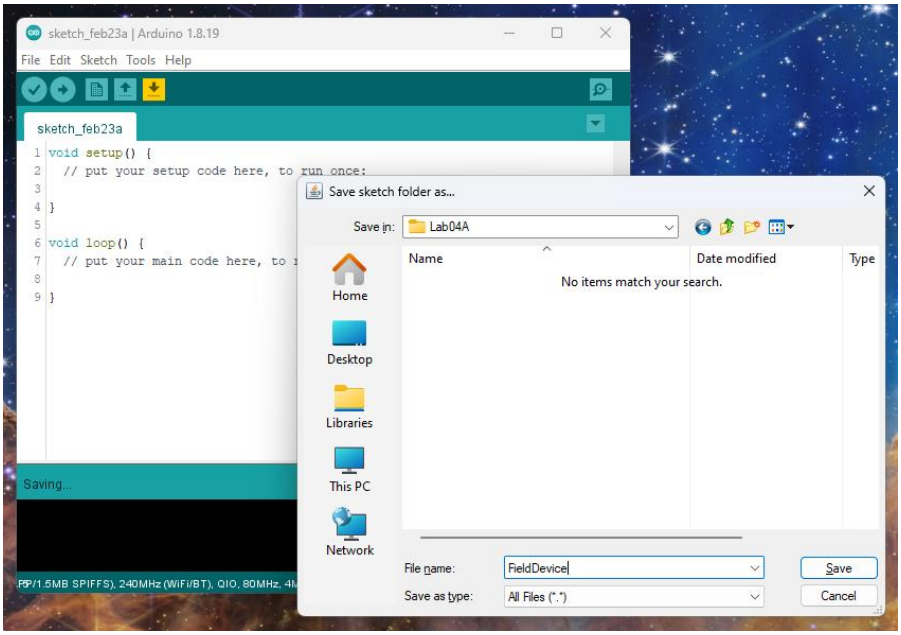


DHT11 VCC -> Things Uno +5V

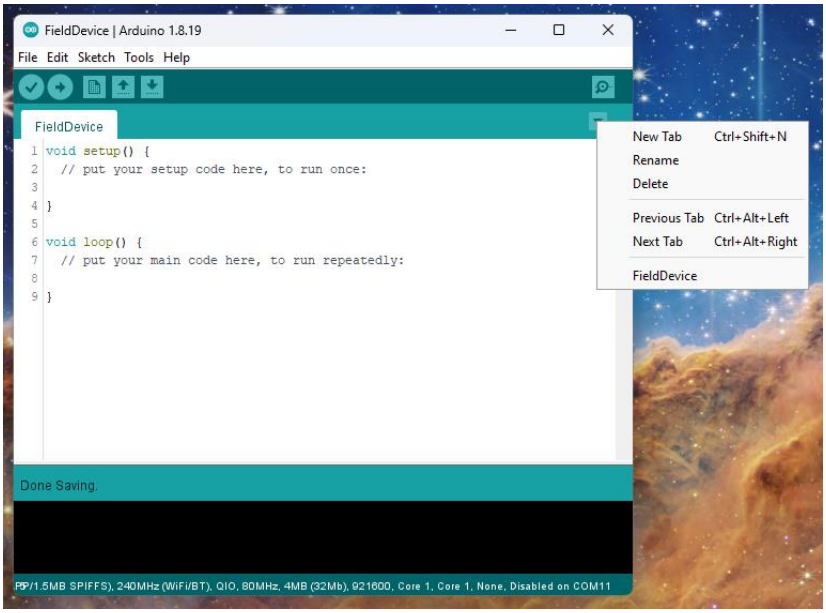
DHT11 GND -> Things Uno GND

DHT11 Signal -> Things Uno DIO 2 PWM

Create a new Arduino project called **Lab04A** in your **4ID3/Lab04** folder. Name the project **FieldDevice**.



Create a new header file called **FieldDevice.h**. Here, we'll store all the project configurations for easy access.



Include your libraries.

FieldDevice | FieldDevice.h §

```
#include <TheThingsNetwork.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

```

Next, we need to configure what digital pin our DHT11 sensor will be connected to.

FieldDevice | FieldDevice.h §

```
#include <TheThingsNetwork.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

//  Macros
#define DHTPIN 2
#define DHTTYPE DHT11
#define DELAY_BETWEEN_SAMPLES_MS 5000
#define GROUP_NAME "GroupA"
#define DEVICE_ID "DeviceA"

```

Next, lets instantiate the DHT object and provide a macro to differentiate the different serial objects in use.

FieldDevice | FieldDevice.h

```
#include <TheThingsNetwork.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

//  Macros
#define DHTPIN 2
#define DHTTYPE DHT11
#define DELAY_BETWEEN_SAMPLES_MS 5000
#define GROUP_NAME "GroupA"
#define DEVICE_ID "DeviceA"

DHT_Unified dht(DHTPIN, DHTTYPE);
#define loraSerial Serial1
#define debugSerial Serial

```

Configure your APP EUI and APP Key.

FieldDevice | FieldDevice.h

```
DHT_Unified dht(DHTPIN, DHTTYPE);
#define loraSerial Serial1
#define debugSerial Serial

// Set your AppEUI and AppKey
#define freqPlan TTN_FP_US915
const char *appEui = "0000000000000000";
const char *appKey = "3A7F0DA10347755BDD4566B7DCD4AE2A";
TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);

unsigned long startTime = millis();

```

In the implementation file, include your header file.

FieldDevice | FieldDevice.h

```
#include "FieldDevice.h"

```

Next, set up your **setup()** function as such:

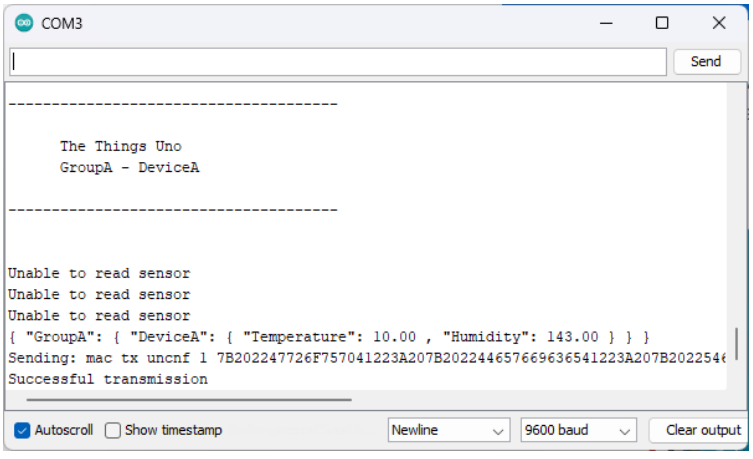
And once the specified interval has elapsed, poll your DHT11 sensor for temperature and humidity.
Reinterpret this data as a string with 2 decimal points.
Concatenate together a JSON formatted object containing your data.
Iteratively copy the string to a new character array.
Publish the character array to your LoRaWAN Things Stack Application.

FieldDevice | FieldDevice.h

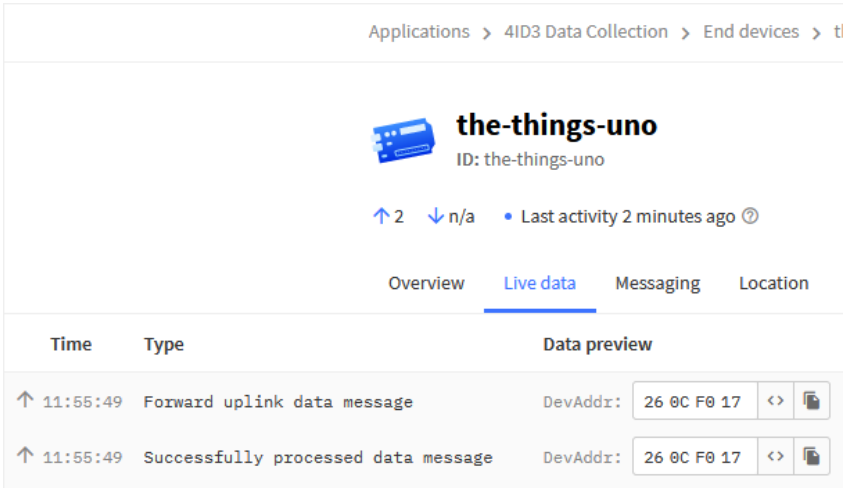
```
24
25 void loop() {
26
27   if(millis() - startTime > DELAY_BETWEEN_SAMPLES_MS){
28
29     sensors_event_t event, dhtHumEvent;
30     dht.temperature().getEvent(&event);
31     float temp = round(event.temperature * 10) / 10.0;
32     dht.humidity().getEvent(&event);
33     float hum = round(event.relative_humidity * 10) / 10.0;
34     delay(200);
35
36     if(!isnan(temp) and !isnan(hum)){
37
38       String msg = "{ \"" + String(GROUP_NAME) +  "\": { \"" + String(DEVICE_ID)
39                  + "\": { \"Temperature\": " + String(temp)
40                  + " , \"Humidity\": " + String(hum) + " } } }";
41       debugSerial.println(msg);
42
43       char msgData[msg.length()];
44       for(int i = 0; i < msg.length(); i++){
45         msgData[i] = msg[i];
46       }
47
48       ttn.sendBytes(msgData, sizeof(msgData));
49     }
50
51     else{
52       debugSerial.println("Unable to read sensor");
53     }
54
55     startTime = millis();
56   }
57
58 }
59
```

Build and upload your project to your **Arduino Leonardo** (The Things Uno) board and open your **serial monitor**.

If you receive **Unable to read sensor**, check your connections.



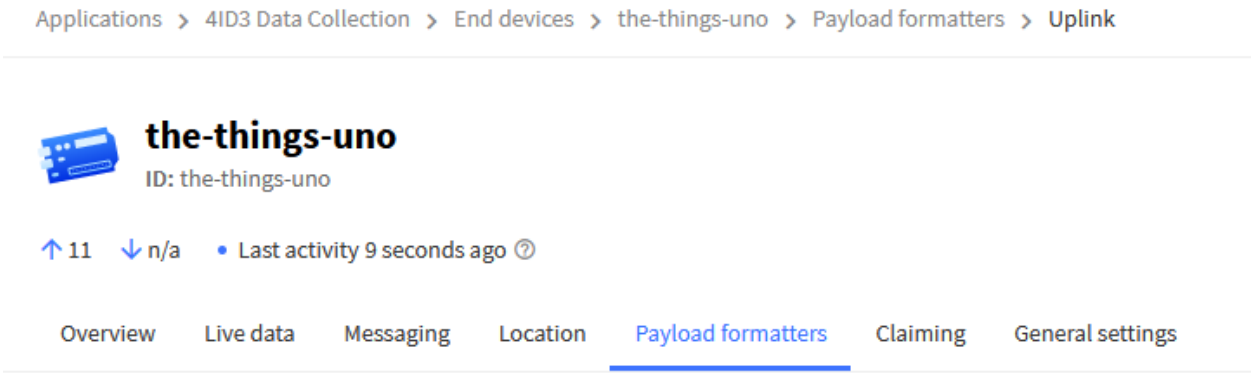
Navigate to your **Application > End Device** on **The Things Stack**, and verify that you are receiving data **uplinks**.

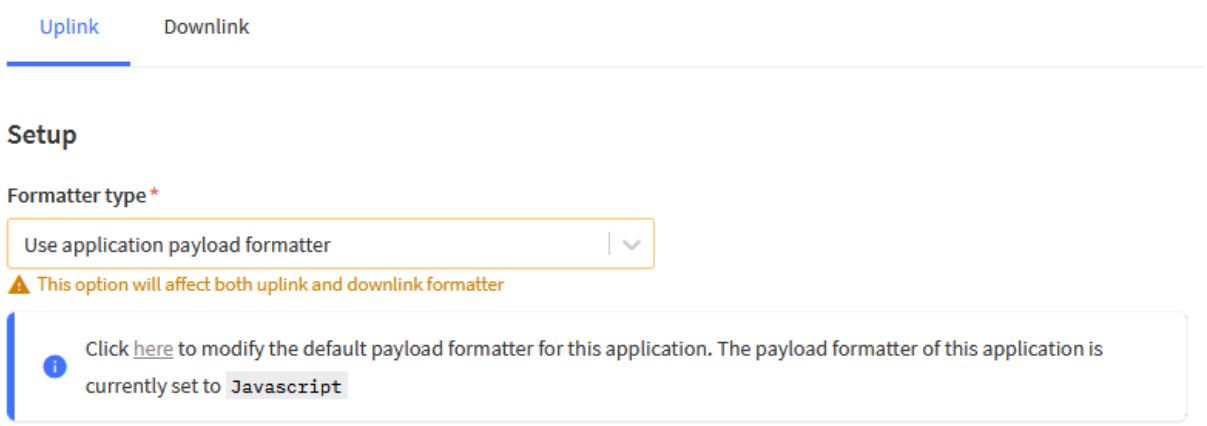


We structured our message in the following JSON format:

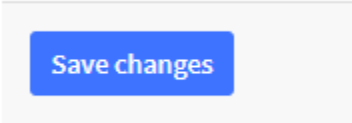
{ "GroupA": { "DeviceA": { "Temperature": 22.4 , "Humidity": 42 } } }

Because of this, we need to use JavaScript to parse the data in our Things Stack Application.
First, we need to tell the device to use the **Application Uplink Formatter**.

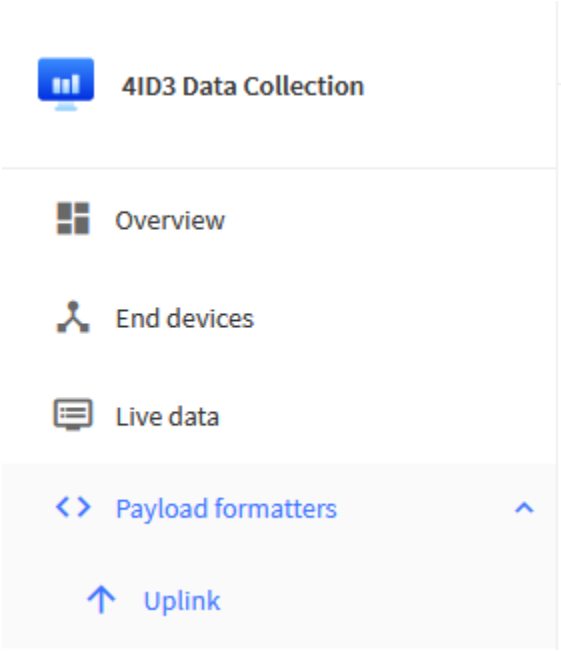




Ensure that you save changes.



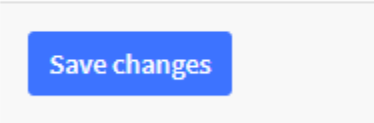
Next, navigate to your Application's Uplink Formatter.



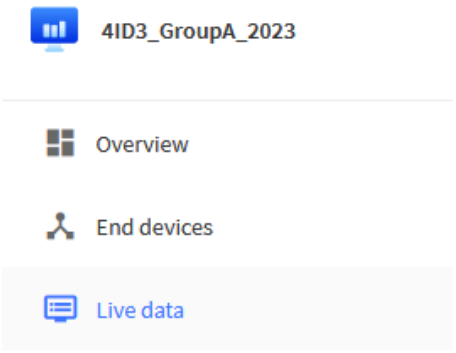
Configure to parse the JSON object.


```
function decodeUplink(input) {
let str = "";
for(let i = 0; i < input.bytes.length; i++){
str += String.fromCharCode(input.bytes[i]);
}
return {
data: JSON.parse(str)
};
}
```

Press **Save changes**.



Navigate back to **Live data** and view the data streams.



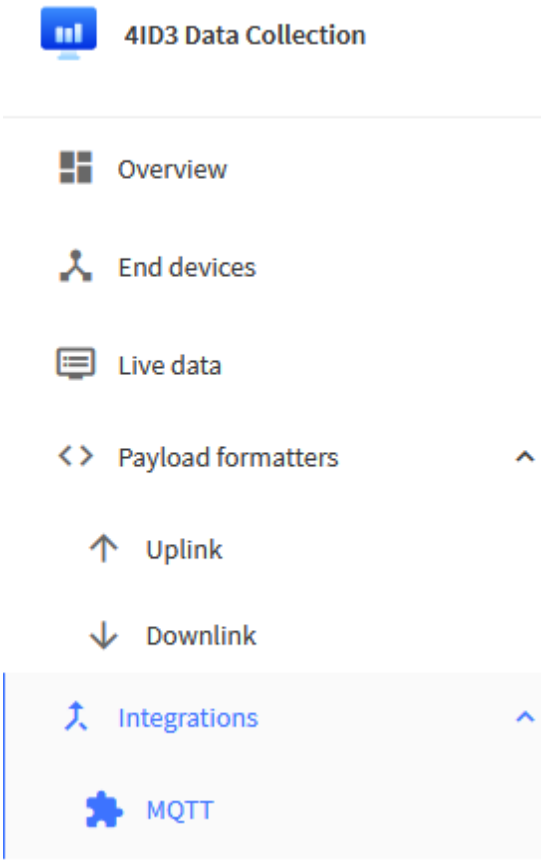
View the uplink.

Click on the **most recent uplink** to view the entire JSON encoded uplink.

Notice that your data is in the **uplink_message/decoded_payload** field.

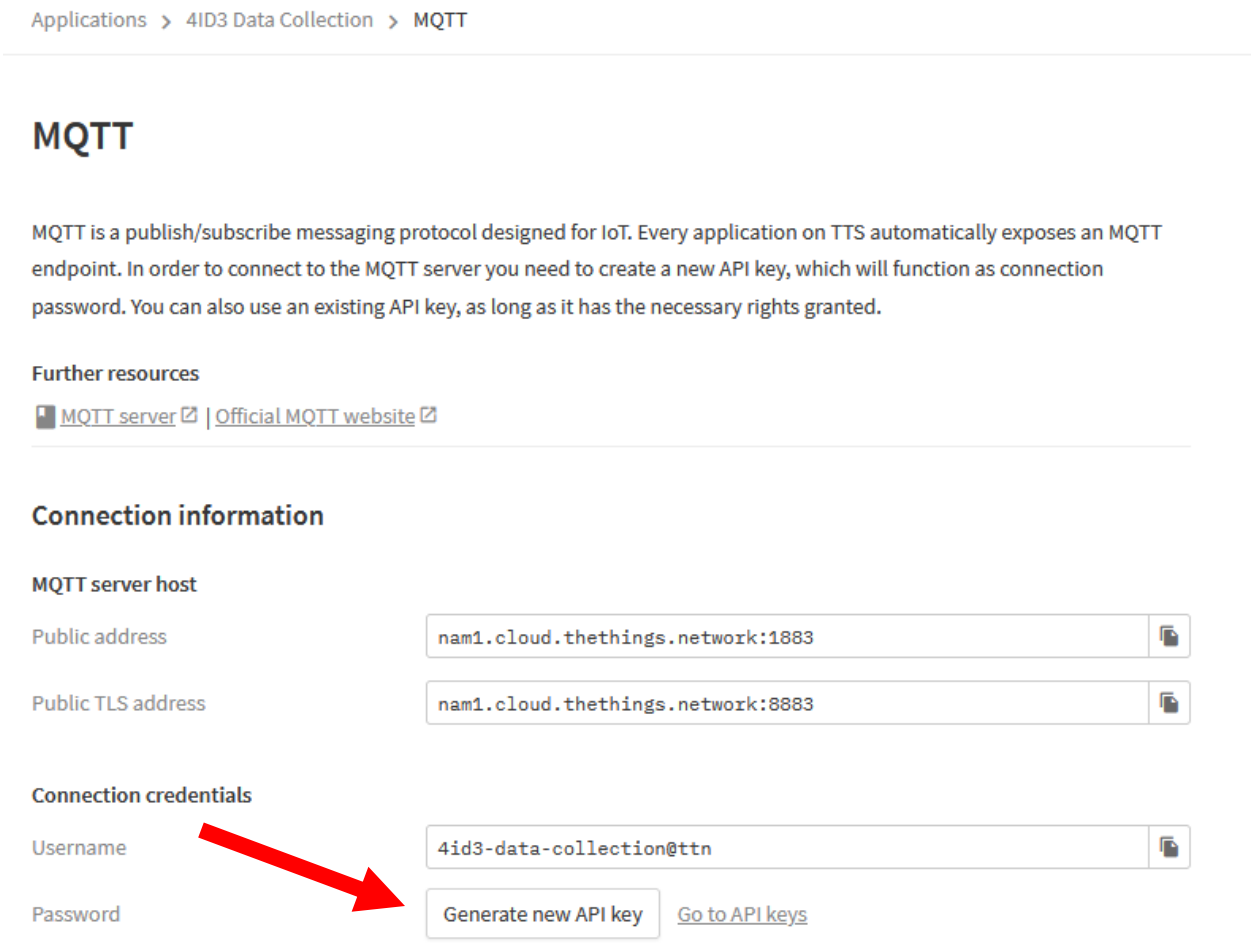
MQTT Integration

Click on **Integrations > MQTT** in the **Application Menu**.

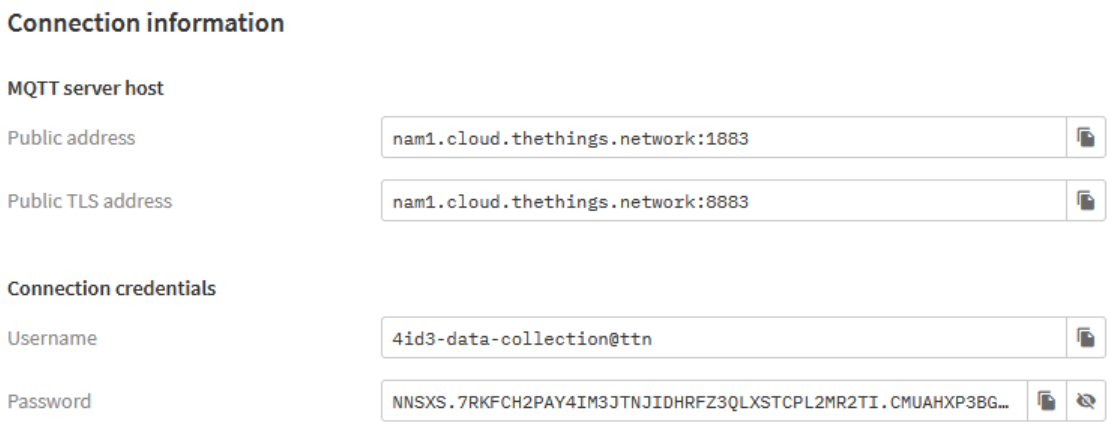


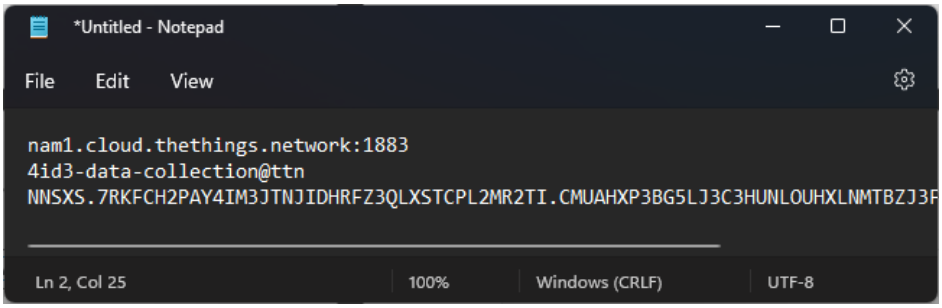
Click **Generate new API key**.

Copy these



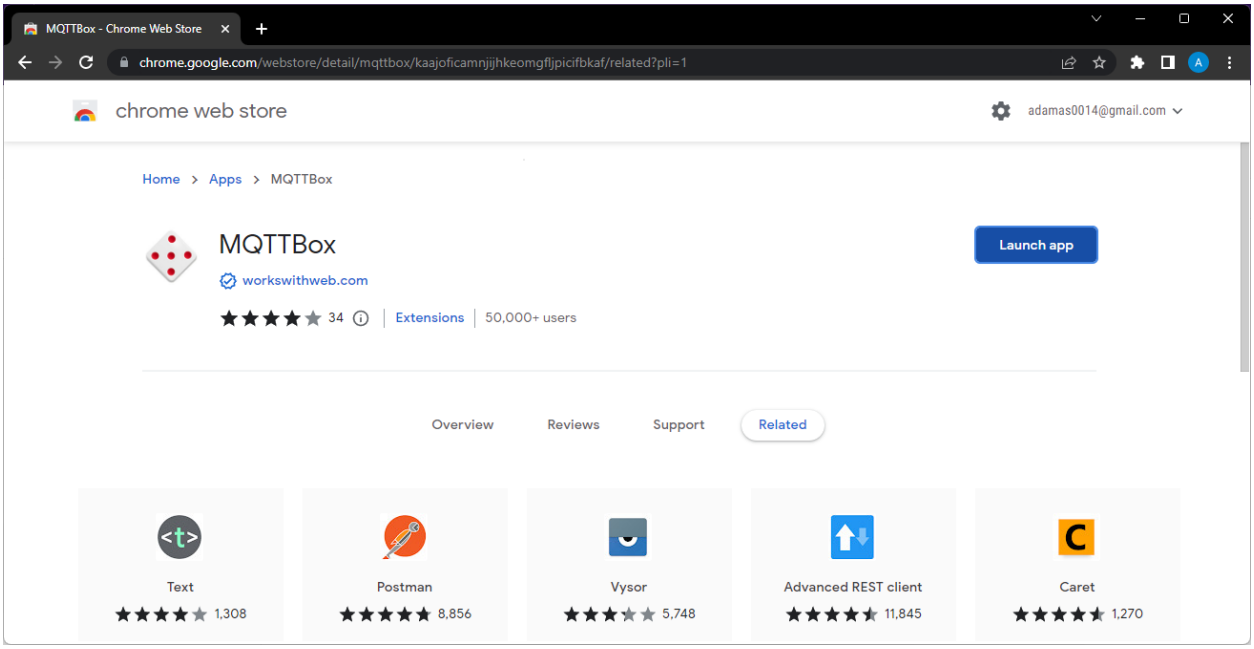
Copy these credentials into a nodepad file.





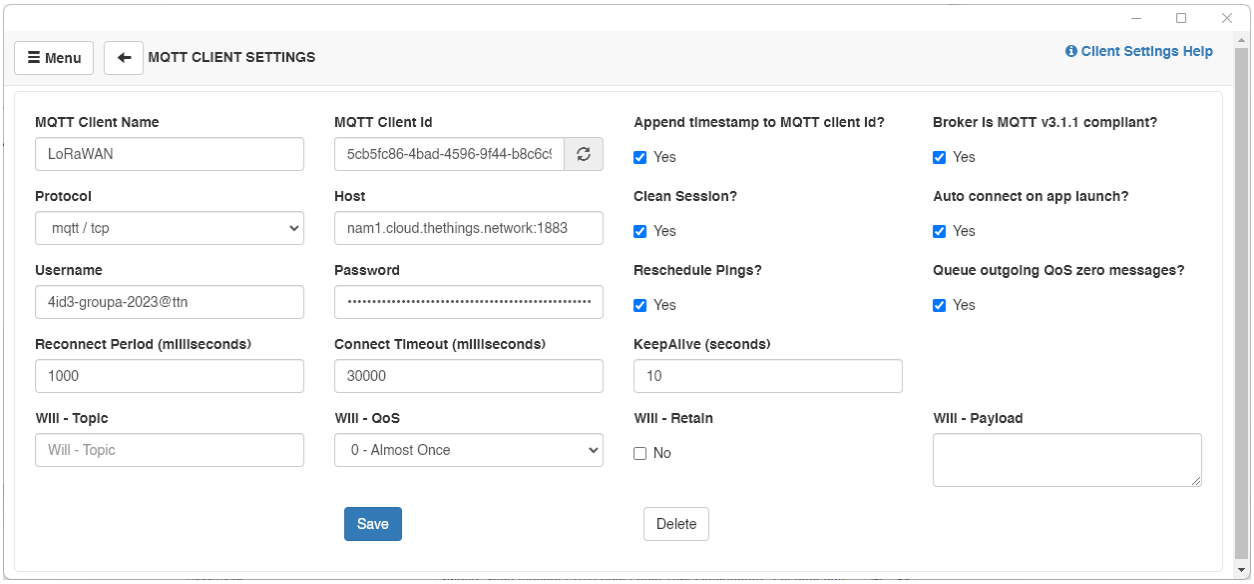
Using **a chromium based browser**, install the extension named **MQTTBox**.

https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf/related?pli=1



Launch **MQTTBox**.

Copy the information from the MQTT Integrations page to connect to The Things Network MQTT server.



Client Name = <Anything>

Protocol = mqtt / tcp **NOTE: this is not secure, for security in the real world, use mqtt / tls.*

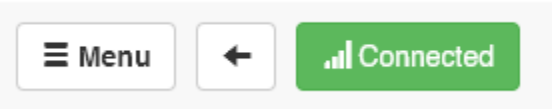
Username: <The username you copied>

Host = <The host URL you copied>

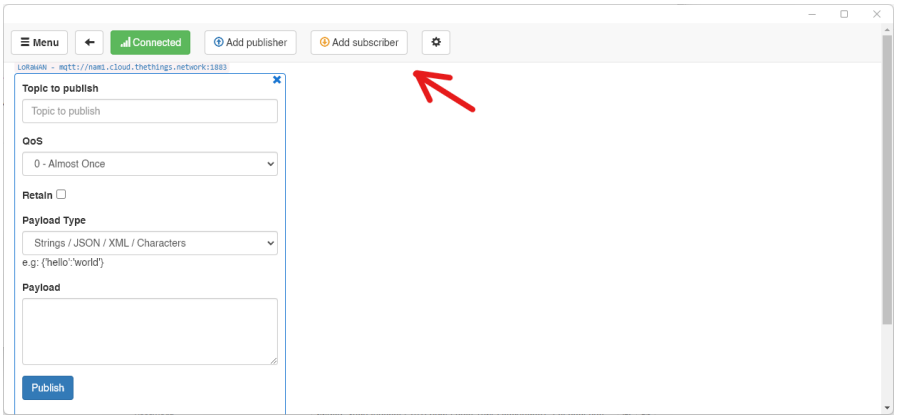
Password = <The API key you copied>

Press **Save**.

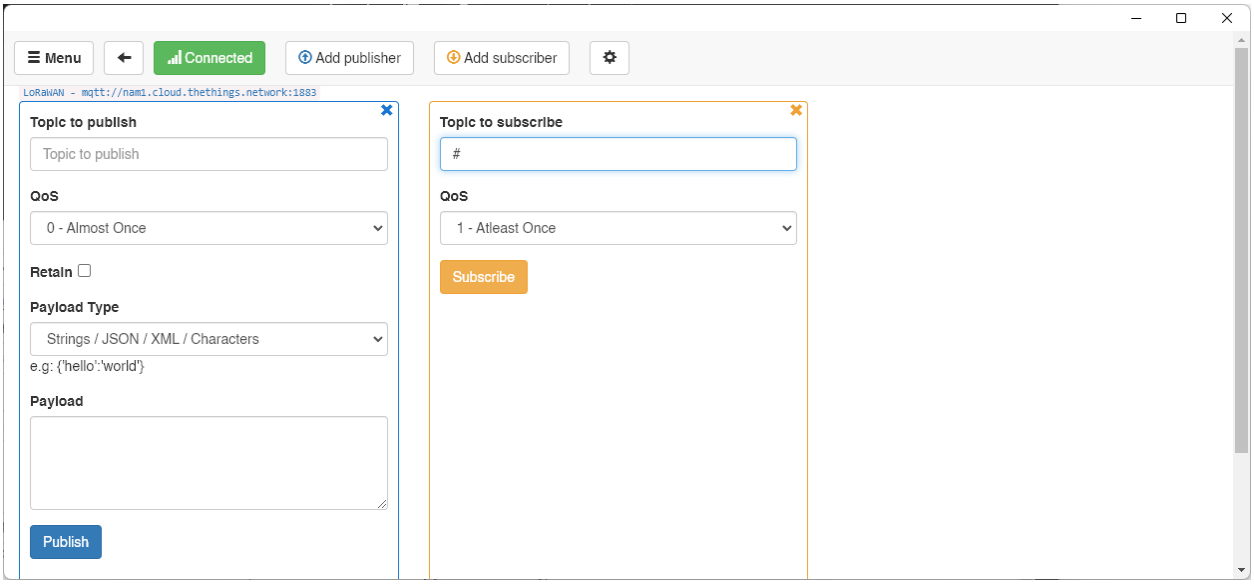
Ensure that you connect after a few seconds. If not, verify your credentials are correct.



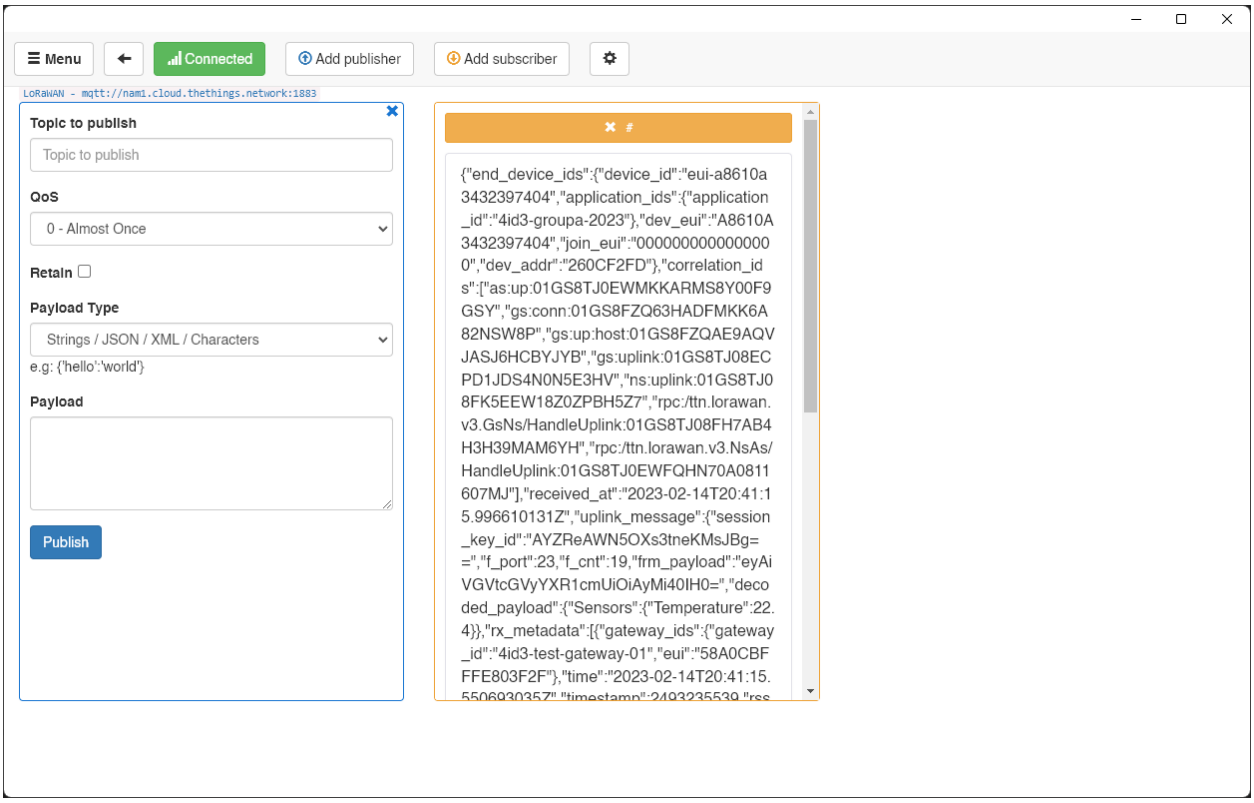
Click **Add Subscriber**.



Subscribe to the topic **#** , which means **all**. Press **Subscribe**.



Watch as the data populates in your MQTT subscription.



Data Collection and Analysis

Its amazing that we can see our sensor data in MQTT, but seeing everything may be overwhelming and it isn't useful to see it in a raw format like this when building an IoT network. We need to implement a way to:

a) Store the data
b) Visualize the data

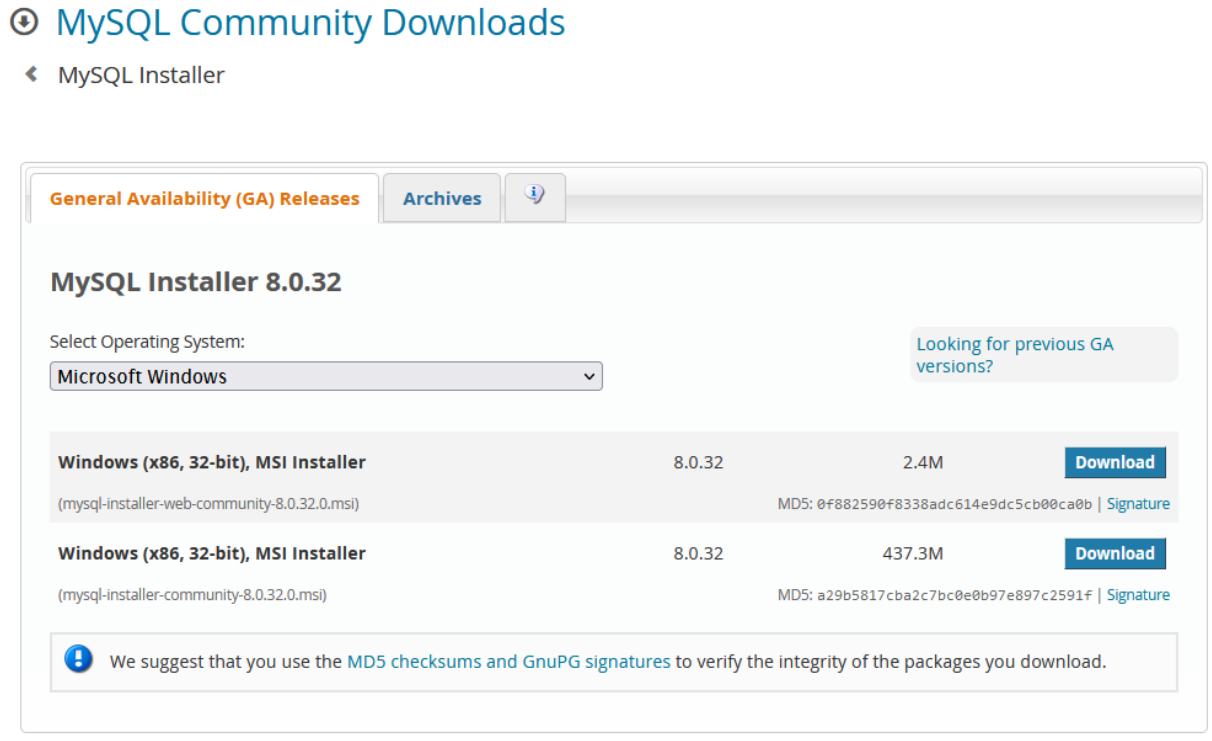
For storing the data, we'll switch it up from previous weeks. Instead of using a MongoDB database, we will use a MySQL database, which is a **relational database**. Non-relational databases are really good for quickly dumping data into without worrying about the relationship between the data. Meanwhile relational databases are more common because they are **very specific** on how they store data, and they can be parsed **extremely quickly** using commands called **SQL Queries**. There is an entire diagram format called an **ERD (Entity Relationship Diagram)** which describes these relationships.

This lab will only **demonstrate** how to use SQL queries in an **IoT application**. Further study will be required to master it.

Installing MySQL & MySQL Connector

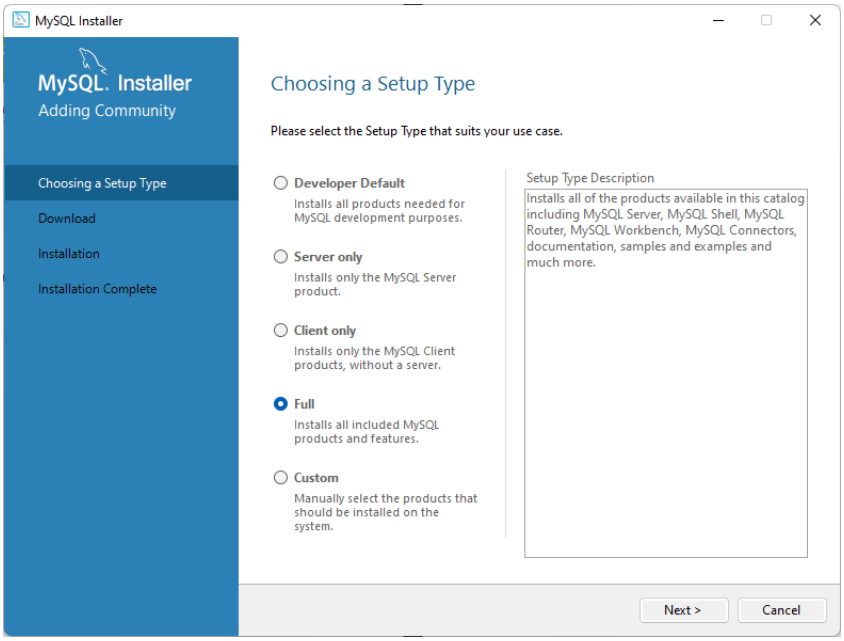
Navigate to the following web page and download the MySQL Installer. Install the **second** option.

https://dev.mysql.com/downloads/installer/

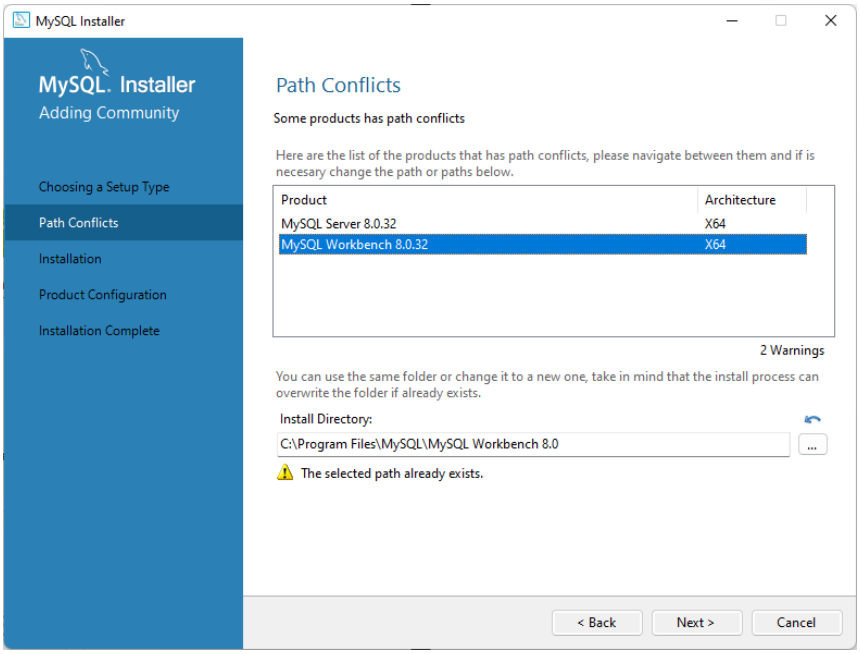


Run the installer. It will prompt for **Administrator** privileges.

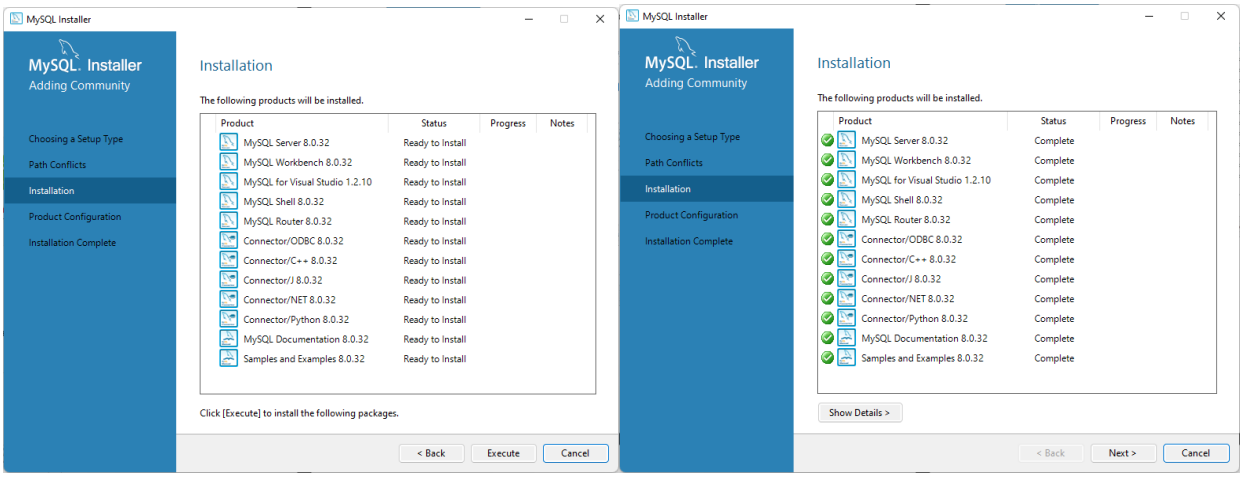
Select **Full**.

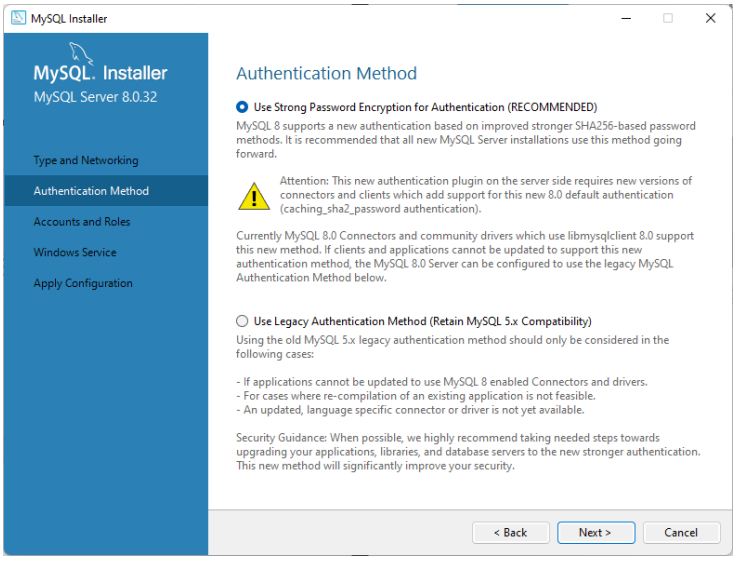


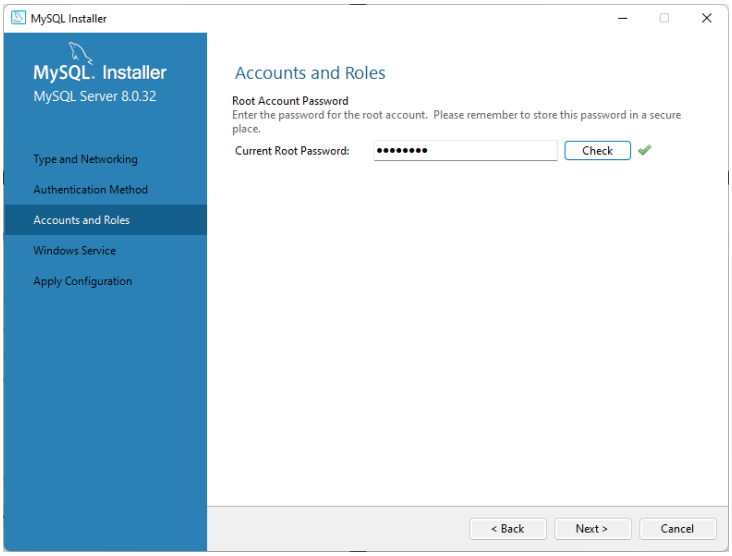
Select **MySQL Workbench**.

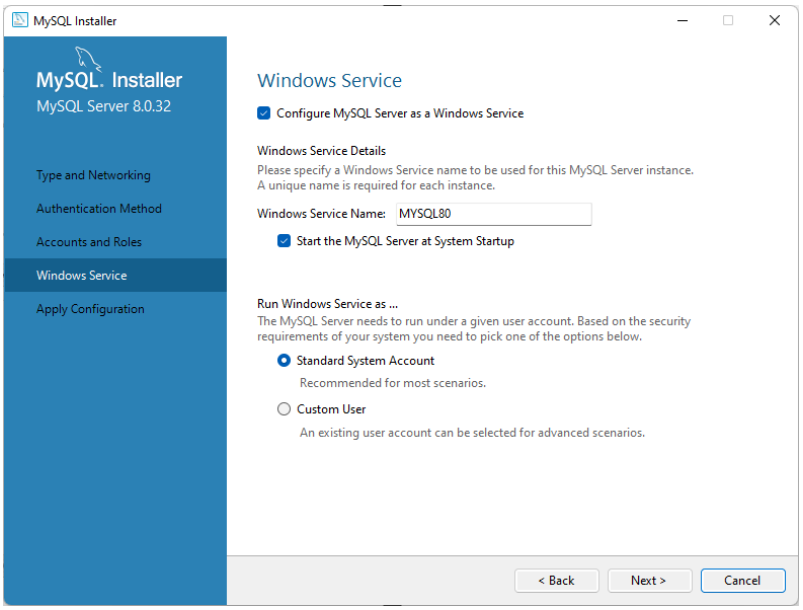


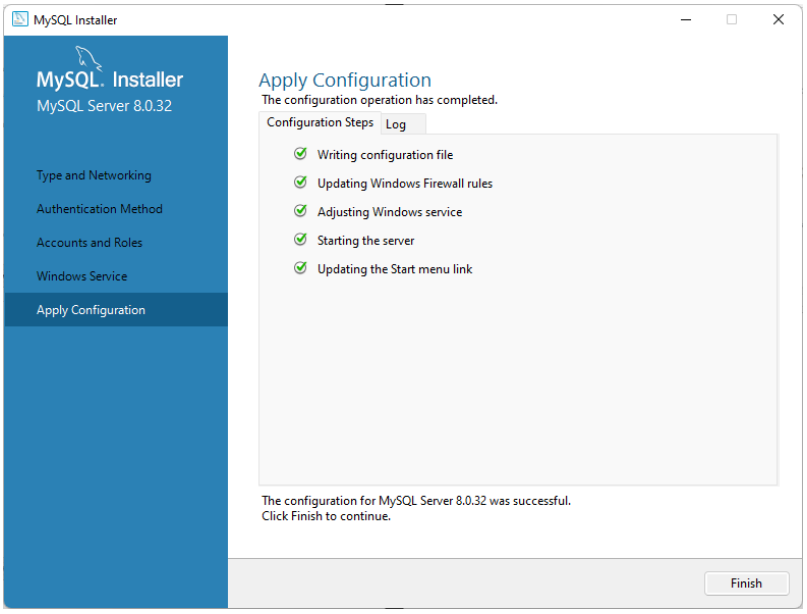
Press **Execute**.

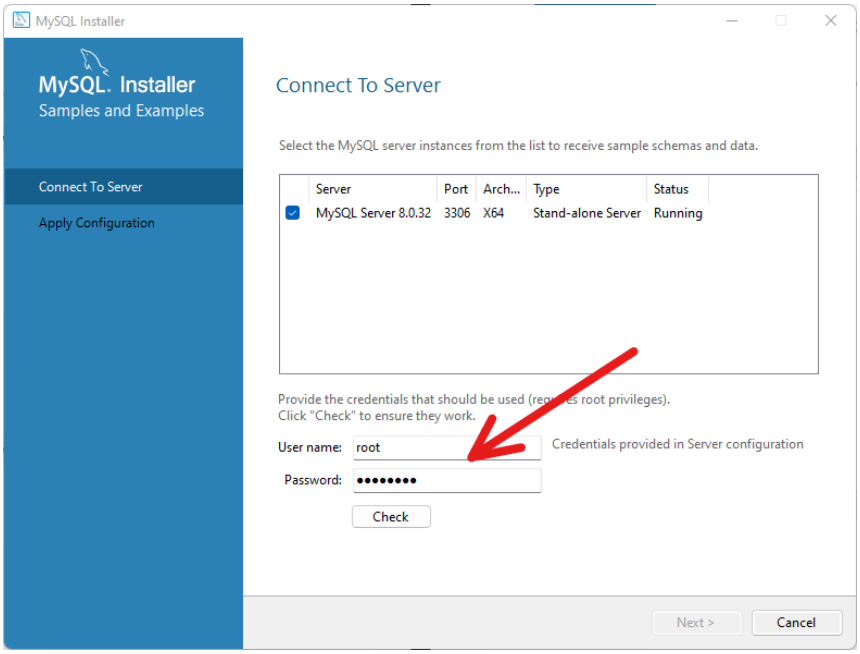


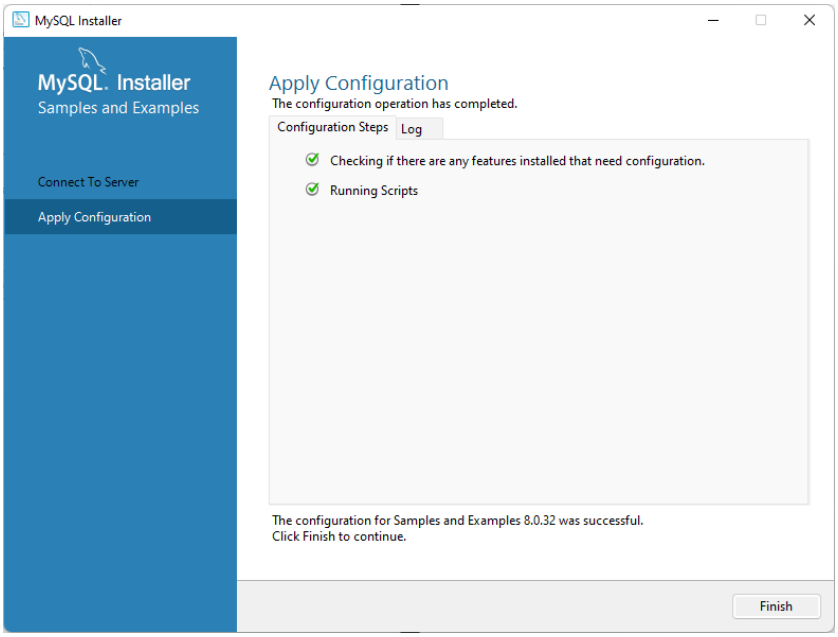

MySQL Installer
MySQL. Installer
MySQL Server 8.0.32
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Authentication Method
Use Strong Password Encryption for Authentication (RECOMMENDED)
MySQL 8 supports a new authentication based on improved stronger SHA256-based password methods. It is recommended that all new MySQL Server installations use this method going forward.
Attention: This new authentication plugin on the server side requires new versions of connectors and clients which add support for this new 8.0 default authentication (caching_sha2_password authentication).
Currently MySQL 8.0 Connectors and community drivers which use libmysqlclient 8.0 support this new method. If clients and applications cannot be updated to support this new authentication method, the MySQL 8.0 Server can be configured to use the legacy MySQL Authentication Method below.
Use Legacy Authentication Method (Retain MySQL 5.x Compatibility)
Using the old MySQL 5.x legacy authentication method should only be considered in the following cases:
- If applications cannot be updated to use MySQL 8 enabled Connectors and drivers.
- For cases where re-compilation of an existing application is not feasible.
- An updated, language specific connector or driver is not yet available.
Security Guidance: When possible, we highly recommend taking needed steps towards upgrading your applications, libraries, and database servers to the new stronger authentication. This new method will significantly improve your security.
< Back
Next >
Cancel

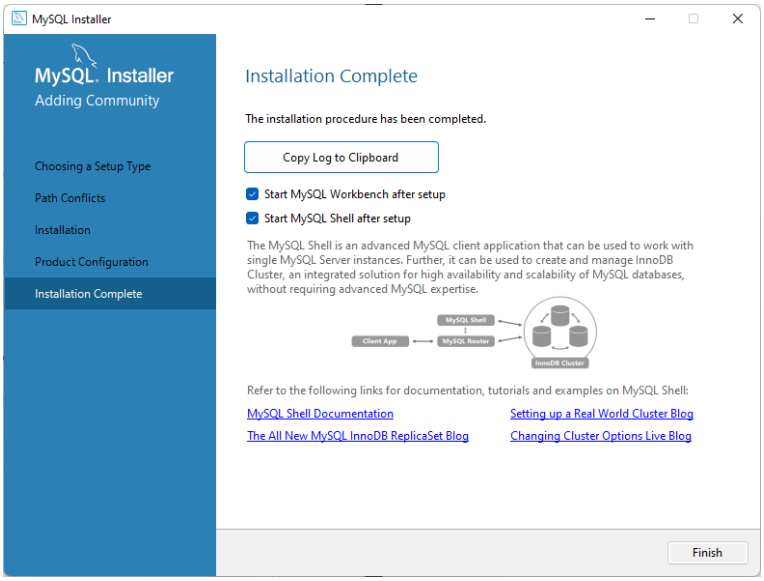

MySQL Installer
MySQL. Installer
MySQL Server 8.0.32
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Accounts and Roles
Root Account Password
Enter the password for the root account. Please remember to store this password in a secure place.
Current Root Password:
Check
< Back
Next >
Cancel

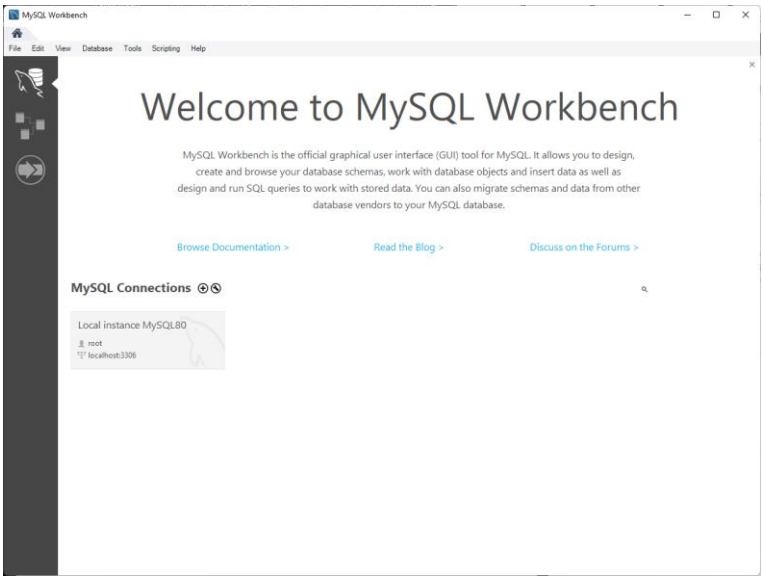

MySQL Installer
MySQL. Installer
MySQL Server 8.0.32
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Windows Service
Configure MySQL Server as a Windows Service
Windows Service Details
Please specify a Windows Service name to be used for this MySQL Server instance.
A unique name is required for each instance.
Windows Service Name: MYSQL80
Start the MySQL Server at System Startup
Run Windows Service as ...
The MySQL Server needs to run under a given user account. Based on the security requirements of your system you need to pick one of the options below.
Standard System Account
Recommended for most scenarios.
Custom User
An existing user account can be selected for advanced scenarios.
< Back
Next >
Cancel


MySQL Installer
MySQL. Installer
MySQL Server 8.0.32
Type and Networking
Authentication Method
Accounts and Roles
Windows Service
Apply Configuration
Apply Configuration
The configuration operation has completed.
Configuration Steps
Log
Writing configuration file
Updating Windows Firewall rules
Adjusting Windows service
Starting the server
Updating the Start menu link
The configuration for MySQL Server 8.0.32 was successful.
Click Finish to continue.
Finish


MySQL Installer
MySQL. Installer
Samples and Examples
Connect To Server
Apply Configuration
Connect To Server
Select the MySQL server instances from the list to receive sample schemas and data.
Server
Port
Arch...
Type
Status
MySQL Server 8.0.32
3306
X64
Stand-alone Server
Running
Provide the credentials that should be used (requires root privileges).
Click "Check" to ensure they work.
User name:
root
Credentials provided in Server configuration
Password:
Check
Next >
Cancel


MySQL Installer
MySQL. Installer
Samples and Examples
Connect To Server
Apply Configuration
Apply Configuration
The configuration operation has completed.
Configuration Steps
Log
Checking if there are any features installed that need configuration.
Running Scripts
The configuration for Samples and Examples 8.0.32 was successful.
Click Finish to continue.
Finish


MySQL Installer
MySQL. Installer
Adding Community
Choosing a Setup Type
Path Conflicts
Installation
Product Configuration
Installation Complete
Installation Complete
The installation procedure has been completed.
Copy Log to Clipboard
Start MySQL Workbench after setup
Start MySQL Shell after setup
The MySQL Shell is an advanced MySQL client application that can be used to work with single MySQL Server instances. Further, it can be used to create and manage InnoDB Cluster, an integrated solution for high availability and scalability of MySQL databases, without requiring advanced MySQL expertise.
MySQL Shell
Client App
MySQL Router
InnoDB Cluster
Refer to the following links for documentation, tutorials and examples on MySQL Shell:
MySQL Shell Documentation
Setting up a Real World Cluster Blog
The All New MySQL InnoDB ReplicaSet Blog
Changing Cluster Options Live Blog
Finish


MySQL Workbench
File Edit View Database Tools Scripting Help
Welcome to MySQL Workbench
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.
Browse Documentation >
Read the Blog >
Discuss on the Forums >
MySQL Connections
Local instance MySQL80
root
localhost:3306

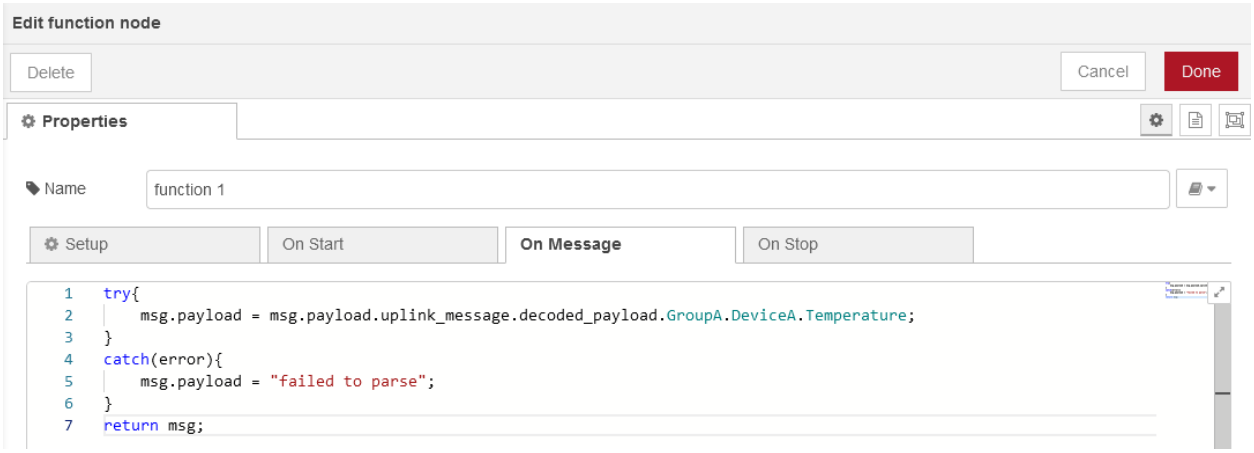
Exercise A

Create a **NodeRED** flow to parse the MQTT message and graph it on your dashboard. Include a screenshot with your report. Save your flow to your GitHub.

HINT:

*For this challenge, you will need to add a **Function Node** into your **flow** that instead of returning **msg.payload**, it returns **msg.payload.uplink_message.decoded_payload.GroupA.DeviceA.Temperature**. Plug this function block into your **graph** node.*

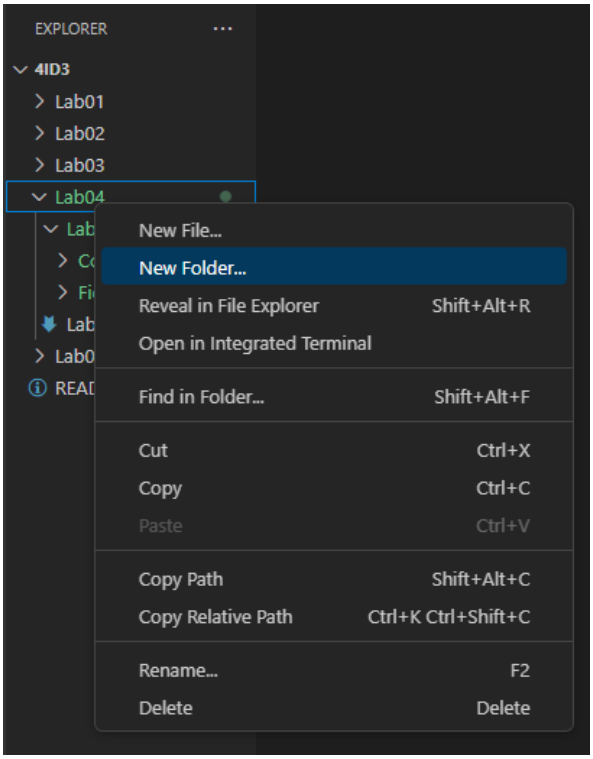
https://nodered.org/docs/user-guide/messages



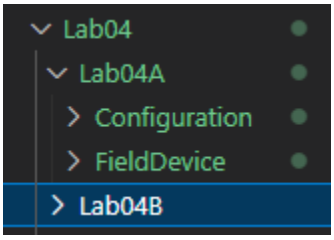
Data Collection

Python Virtual Environment

Open the **4ID3/ folder** using **VSCode**.



Name it **Lab04B**.

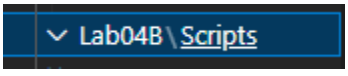


Unlike in previous labs, today we are not going to install Python libraries globally on our PC using PIP. Instead, we are going to create a **Python Virtual Environment** (VENV).

This wasn't done in previous labs to keep things simple as many people are at different levels of understanding when it comes to python.

However, if you work on python projects in the future, this is how it should be set up for each project.

Create another folder inside of **Lab04B** called **scripts/.**

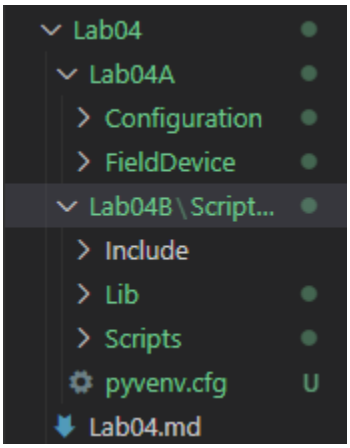


Navigate into this folder and run the command:

python -m venv venv

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3> cd .\Lab04\Lab04B\Scripts\
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python -m venv venv
PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```

Notice that many files have been created.

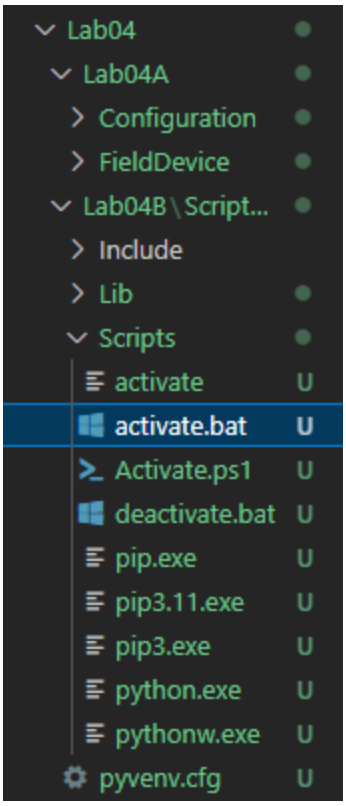


If you received an error, run the following using pip in PowerShell as Administrator and try again:

pip install virtualenv

You have now created an isolated python environment in your PC. Libraries that were installed previously have no impact here, and libraries installed here have no impact on your PC.

To use this environment, you must source the **venv/Scripts/activate.bat** script.



To start using this **virtual environment**, you must first source the **activate.bat** file in **each new teminal window.**

.\venv\Scripts\activate

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> .\venv\Scripts\activate
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> pip list
Package    Version
---------- -------
pip        22.3.1
setuptools 65.5.0

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```

Notice how only 2 python packages are installed in this environment.

Go ahead and upgrade pip as recommended.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\adama\onedrive\documents\repositories\4id3\lab04\lab04b\scripts\venv\lib\site-packages (22.3.1)
Collecting pip
  Downloading pip-23.0.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 10.9 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.3.1
    Uninstalling pip-22.3.1:
      Successfully uninstalled pip-22.3.1
Successfully installed pip-23.0.1
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```

From here, while in the virtual environment, please install the following packages using pip:

pip install mysql-connector-python

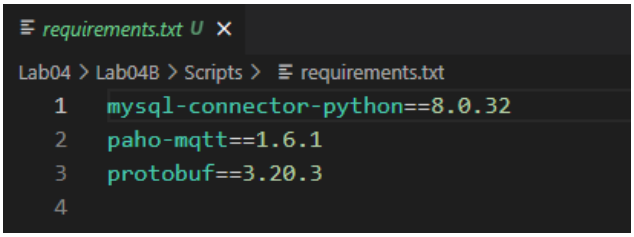
pip install paho-mqtt

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> pip install mysql-connector-python paho-mqtt
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.32-cp311-cp311-win_amd64.whl (7.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.9/7.9 MB 10.1 MB/s eta 0:00:00
```

Now, so other people can install these packages in one easy command, we'll generate a **requirements.txt** file.

pip freeze > requirements.txt

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> pip freeze > requirements.txt
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```



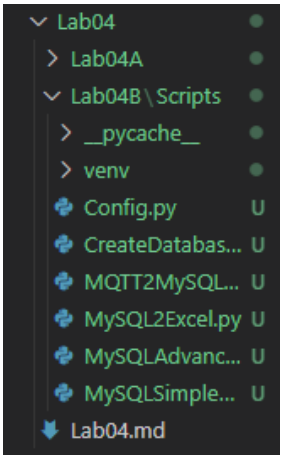
To load packages from this file, use the following command:

pip install -r requirements.txt

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> pip install -r .\requirements.txt
Requirement already satisfied: mysql-connector-python==8.0.32 in c:\users\adama\onedrive\documents\repositories\4id3\lab04\lab
04b\scripts\venv\lib\site-packages (from -r .\requirements.txt (line 1)) (8.0.32)
Requirement already satisfied: paho-mqtt==1.6.1 in c:\users\adama\onedrive\documents\repositories\4id3\lab04\lab04b\scripts\ve
nv\lib\site-packages (from -r .\requirements.txt (line 2)) (1.6.1)
Requirement already satisfied: protobuf==3.20.3 in c:\users\adama\onedrive\documents\repositories\4id3\lab04\lab04b\scripts\ve
nv\lib\site-packages (from -r .\requirements.txt (line 3)) (3.20.3)
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```

Now copy the following **Python scripts** from the **lab GitHub repo** into **Lab04/Lab04B/Scripts**:

- Config.py
- CreateDatabase.py
- MQTT2MySQL.py
- MySQL2Excel.py
- MySQLSimpleParser.py
- MySQLAdvancedParser.py



Open **Config.py** and enter your **individual credentials**. These credentials will be imported into every other Python file.

Config.py U

Lab04 > Lab04B > Scripts > Config.py > ...

```
1   HOST_IP = "localhost"
2   USER = "root"
3   PASSWORD = "9055259140"
4   GROUP_NAME = "GroupA"
5   DEVICE_ID = "DeviceA"
6
7   MQTT_HOSTNAME = "nam1.cloud.thethings.network"
8   MQTT_USERNAME = "4id3-data-collection@ttn"
9   MQTT_API_KEY = "NNSXS.7RKFCH2PAY4IM3JTNJIDHRFZ3QLXS
10  MQTT_PORT = 1883
11  MQTT_TOPIC = "#"
12
```

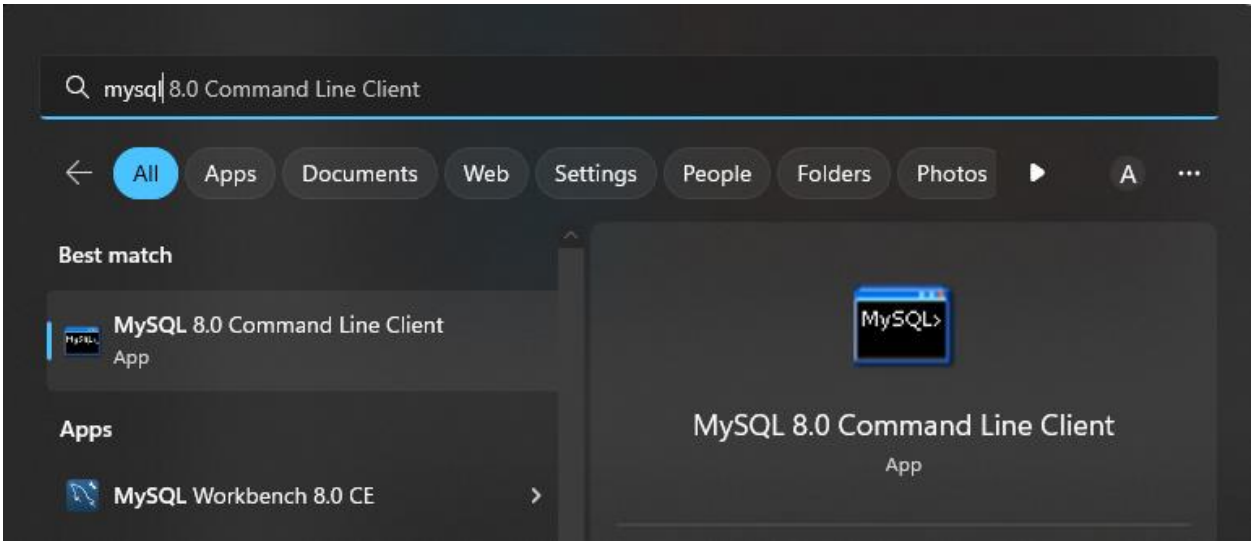
Creating the Database

Once completed, run the **CreateDatabase.py** script from the terminal. Ensure that you have your venv sourced.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python .\CreateDatabase.py
   -> ('sys',)
   -> ('test_database',)
   -> ('world',)
Created Database
Connected to database
Dropped table
Created table
[]
Inserted into table
[]
 [ (1, 'Test', 'Test', 'Test') ]

Selected data from table
MySQL connection is closed
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```

To verify that the database has been generated, open **MySQL 8.0 Command Line Client**.



Enter your password and issue the following queries. Ensure that the test insert was successful.

USE *GroupA*

*SELECT * FROM DeviceA;*

```
mysql> use GroupA
Database changed
mysql> select * from DeviceA;
+----+-----------+--------+---------+
| id | Timestamp | Sensor | Reading |
+----+-----------+--------+---------+
|  1 | Test      | Test   | Test    |
+----+-----------+--------+---------+
1 row in set (0.00 sec)

mysql>
```

From here, we can also try an insert directly.

INSERT INTO `GroupA`.`DeviceA` (`Timestamp`, `Sensor`, `Reading`) VALUES ("Test2", "Test2", "Test2");

*SELECT * FROM `GroupA`.`DeviceA`;*

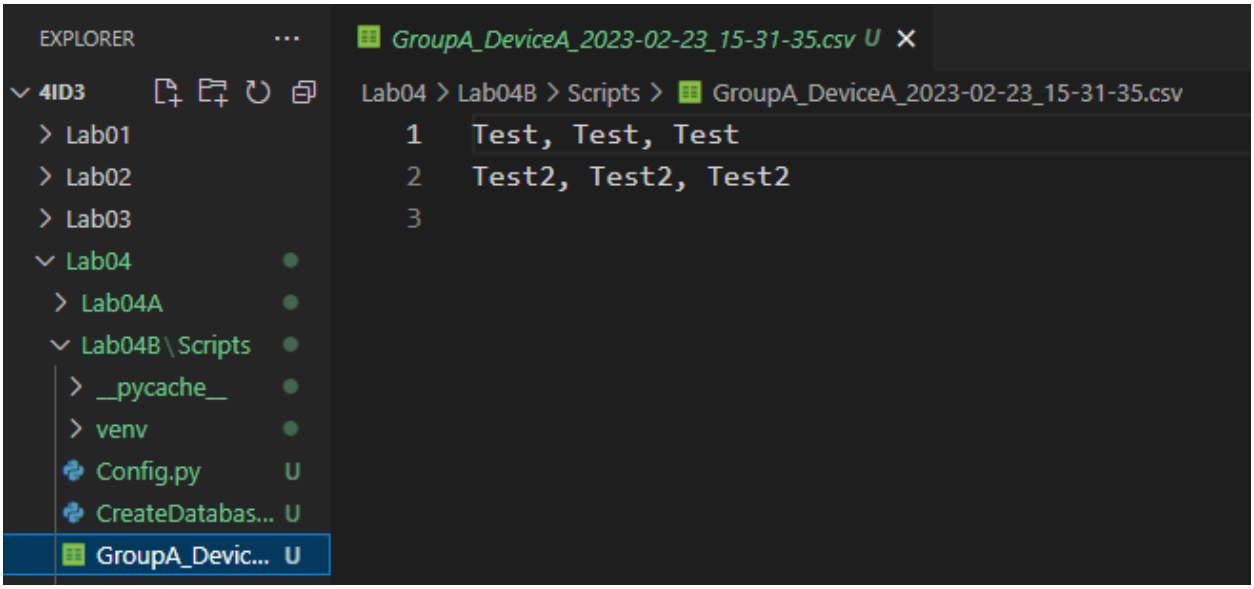
```
mysql> INSERT INTO `GroupA`.`DeviceA` (`Timestamp`, `Sensor`, `Reading`) VALUES ("Test2", "Test2", "Test2");
Query OK, 1 row affected (0.01 sec)

mysql> select * from `GroupA`.`DeviceA`;
+----+-----------+--------+---------+
| id | Timestamp | Sensor | Reading |
+----+-----------+--------+---------+
|  1 | Test      | Test   | Test    |
|  2 | Test2     | Test2  | Test2   |
+----+-----------+--------+---------+
2 rows in set (0.00 sec)
```

Converting Database to Excel

Next, using the VSCode terminal, launch **MySQL2Excel.py**.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python .\MySQL2Excel.py
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts>
```

Observe that a new file has been generated.



Connecting MQTT to your Local MySQL Database

Ensure that your **Field Device** is still publishing data and launch **MQTT2MySQL.py**.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python .\MQTT2MySQL.py

-------
CONFIGURATION
-------
IP: nam1.cloud.thethings.network
PORT: 1883
TOPIC: #

Connecting to MQTT
.
.
.
.
.
.
.
Connected to  0
['2023-02-23 15:35:42', 'Humidity', '159.8' ] MQTT -> MySQL ( `GroupA`.`DeviceA` )
Response from MySQL: []


['2023-02-23 15:35:42', 'Temperature', '10' ] MQTT -> MySQL ( `GroupA`.`DeviceA` )
Response from MySQL: []
```

Ensure that they are being inserted correctly.

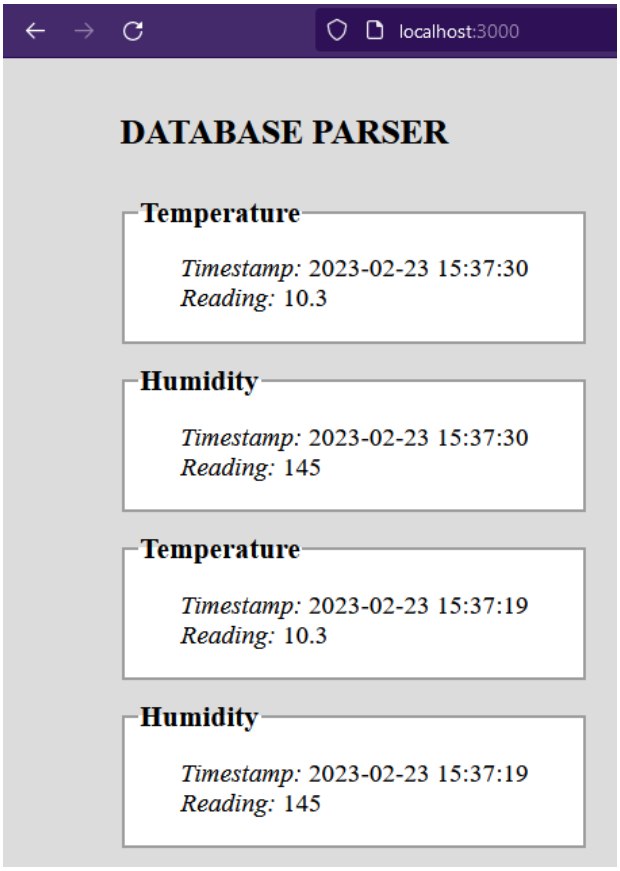
```
mysql> select * from `GroupA`.`DeviceA`;
+----+---------------------+-------------+---------+
| id | Timestamp           | Sensor      | Reading |
+----+---------------------+-------------+---------+
|  1 | Test                | Test        | Test    |
|  2 | Test2               | Test2       | Test2   |
|  3 | 2023-02-23 15:35:42 | Humidity    | 159.8   |
|  4 | 2023-02-23 15:35:42 | Temperature | 10      |
|  5 | 2023-02-23 15:35:54 | Humidity    | 159.8   |
|  6 | 2023-02-23 15:35:54 | Temperature | 10.1    |
|  7 | 2023-02-23 15:36:11 | Humidity    | 146     |
|  8 | 2023-02-23 15:36:11 | Temperature | 10.3    |
+----+---------------------+-------------+---------+
8 rows in set (0.00 sec)
```

Parsing the Database (Simple)

Run the **MySQLSimpleParser.py** script then, using a **web browser**, navigate to **localhost:3000**.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python .\MySQLSimpleParser.py


--------------------
Simple HTTP IoT Server
---        ------------------


HTTP server running on 0.0.0.0 port 3000


Server Ready

127.0.0.1 - - [23/Feb/2023 15:40:34] "GET / HTTP/1.1" 200 -
```

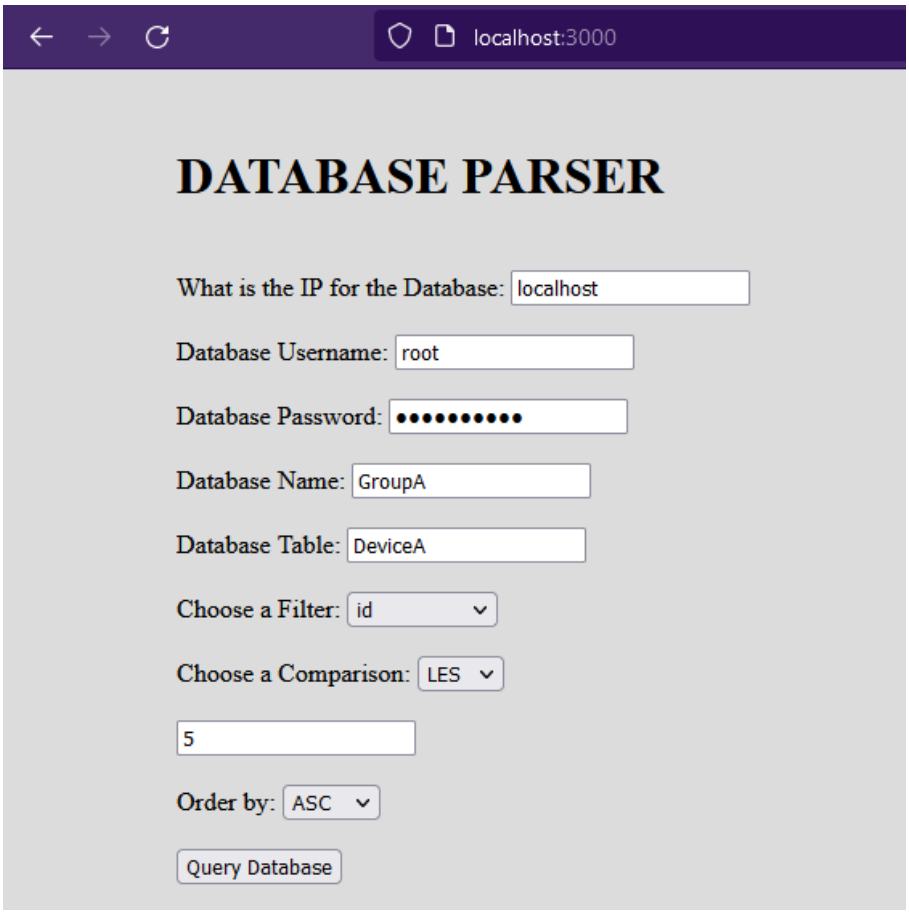
Each time the page is reloaded, a GET request is sent to the HTTP server. During the callback function of the GET request, the database is queried for new data and this new data is formatted as an HTML document and server to the client's web browser.



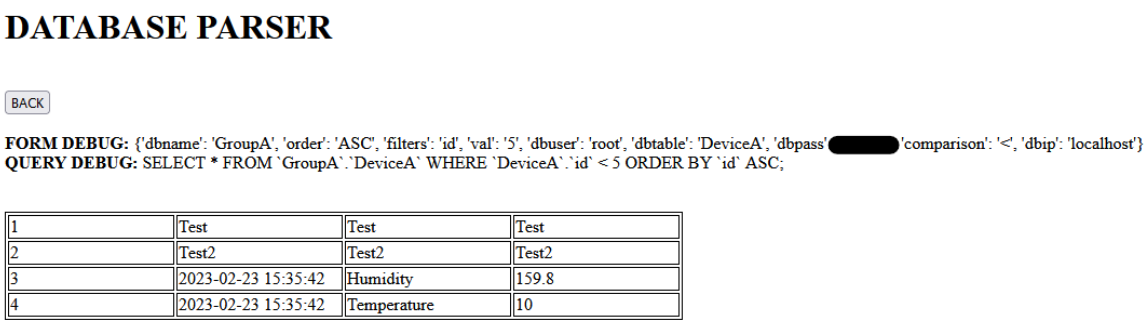
Parsing the Database (Advanced)

Run the **MySQLAdvancedParser.py** script then, using a **web browser**, navigate to **localhost:3000**.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

(venv) PS C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts> python .\MySQLAdvancedParser.py
C:\Users\adama\OneDrive\Documents\Repositories\4ID3\Lab04\Lab04B\Scripts\MySQLAdvancedParser.py:16: DeprecationWarning: 'cgi' is deprecated and slated for removal in Python 3.13
  import cgi
serving at port 3000
ERROR:root:Host: localhost:3000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/110.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://localhost:3000/data
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1


127.0.0.1 - - [23/Feb/2023 16:02:55] "GET / HTTP/1.1" 200 -
```



Choose your parameters and press **Query Database**.



DATABASE PARSER

BACK

FORM DEBUG: {'dbname': 'GroupA', 'order': 'ASC', 'filters': 'id', 'val': '5', 'dbuser': 'root', 'dbtable': 'DeviceA', 'dbpass' [redacted] 'comparison': '<', 'dbip': 'localhost'}
QUERY DEBUG: SELECT * FROM `GroupA`.`DeviceA` WHERE `DeviceA`.`id` < 5 ORDER BY `id` ASC;

1	Test	Test	Test
2	Test2	Test2	Test2
3	2023-02-23 15:35:42	Humidity	159.8
4	2023-02-23 15:35:42	Temperature	10

A table should be automatically generated based on your entered parameters.

Exercise B

Use **Microsoft Excel** to graph the data **queried** from your database (*MySQL2Excel.py*). Paste the graph with your lab submission.

Installing Visual Studio

In this section, we will explore creating a basic desktop form application that connects to MySQL and uses the IoT data that has been collected in previous sections.

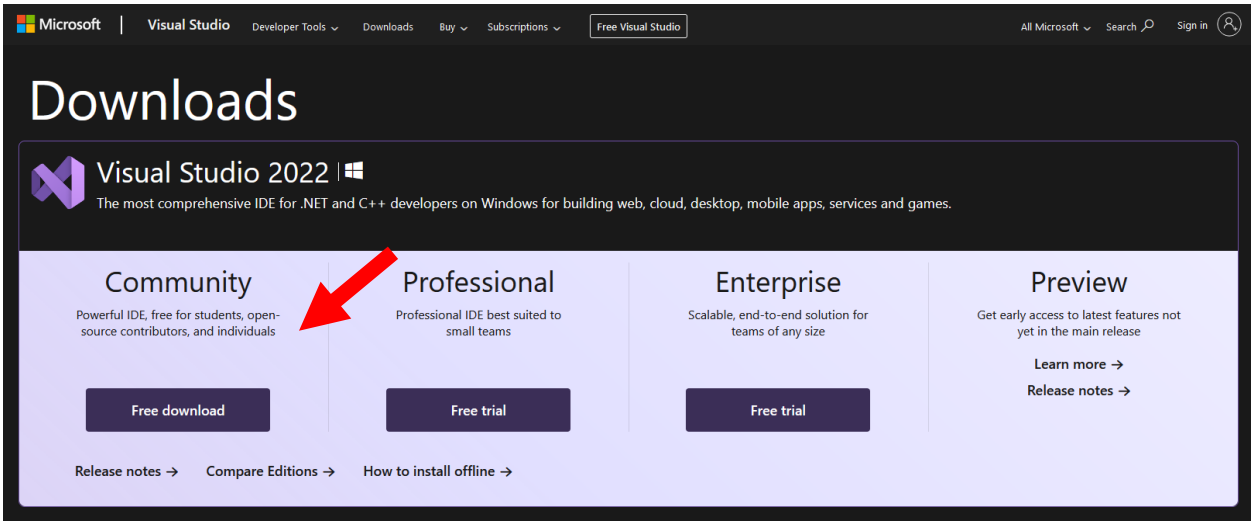
Desktop apps are an alternative to the web applications built in the previous section. Instead of a web server serving your web browser an HTML document with sensor data, a desktop application is a program specifically compiled for your computer. In our case, as a .EXE file which runs on Windows.

We will be learning the basics of 1-page form applications in C# using .NET WinForms, and gathering user data to create database queries.

Don't be intimidated by C#. It has a similar syntax to C++ and you'll be led through the entire process. If you wish to use a desktop application in your project, or are just interested in learning more, I strongly suggest on watching a few YouTube videos.

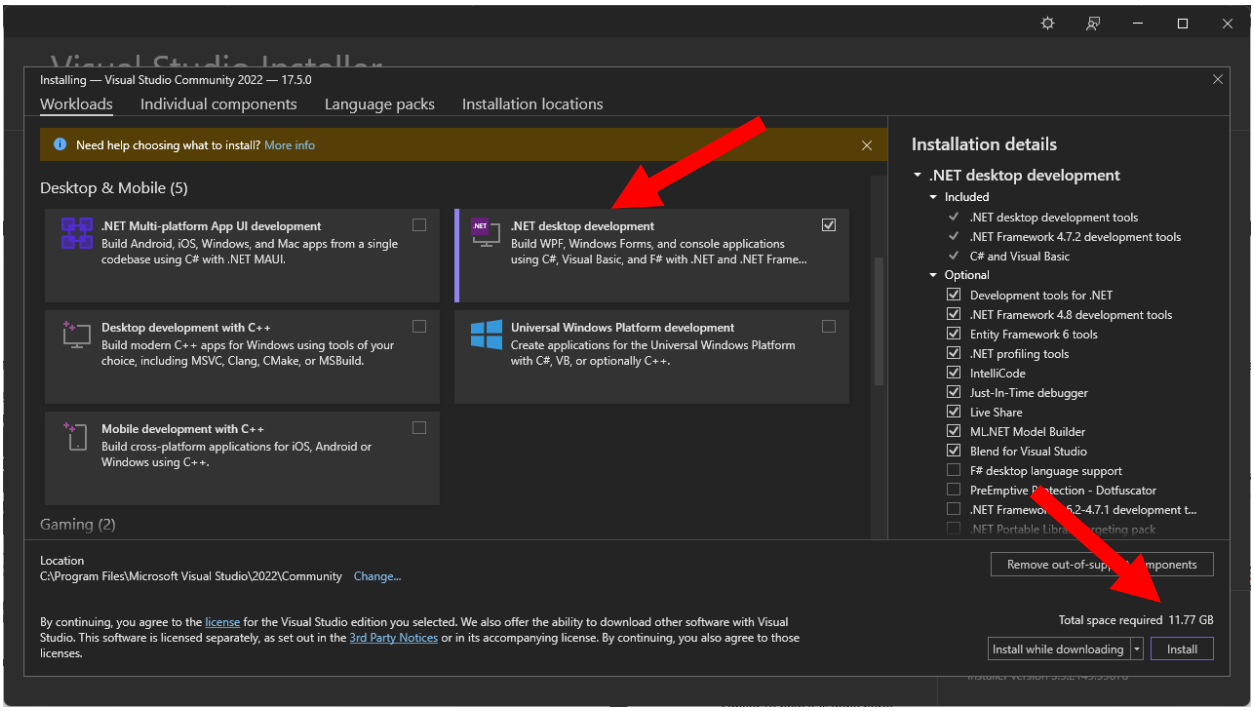
Navigate to the Microsoft Visual Studio download page.

https://visualstudio.microsoft.com/downloads/

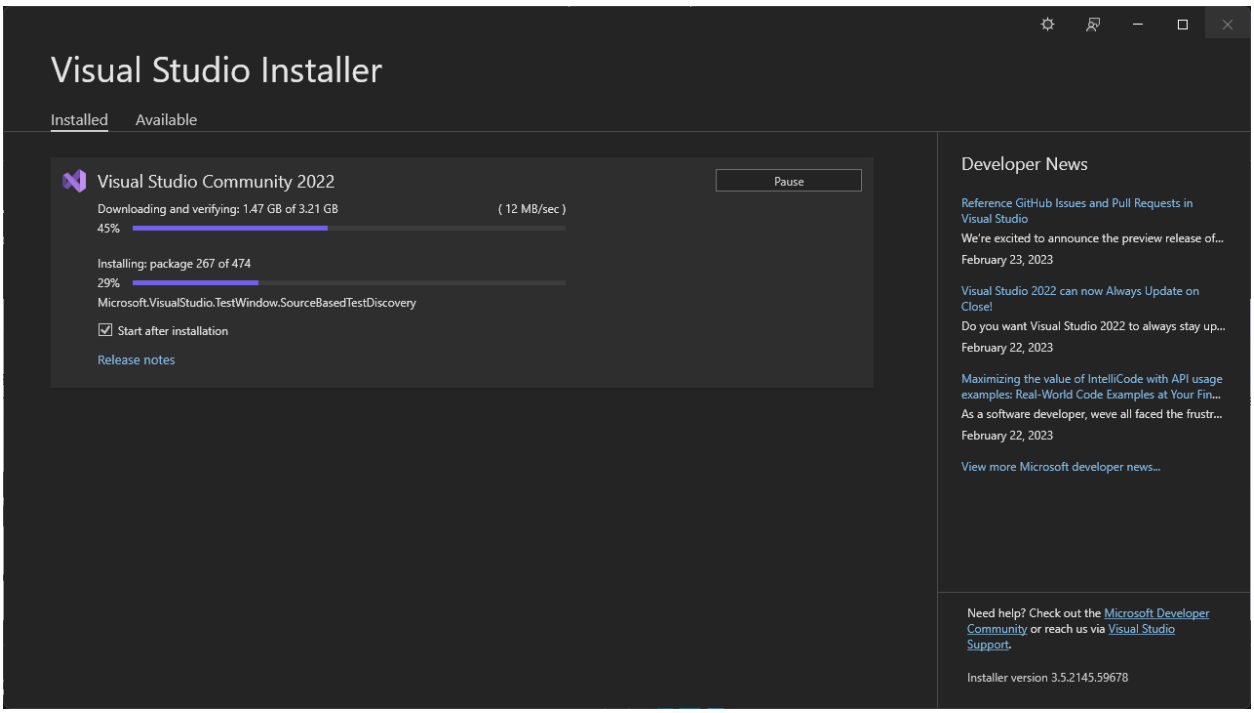


Click **Free download** for the **Community** edition and run the **installer**.

Ensure that **.NET desktop development** is checked on the **Workloads** menu. Press **Install**.



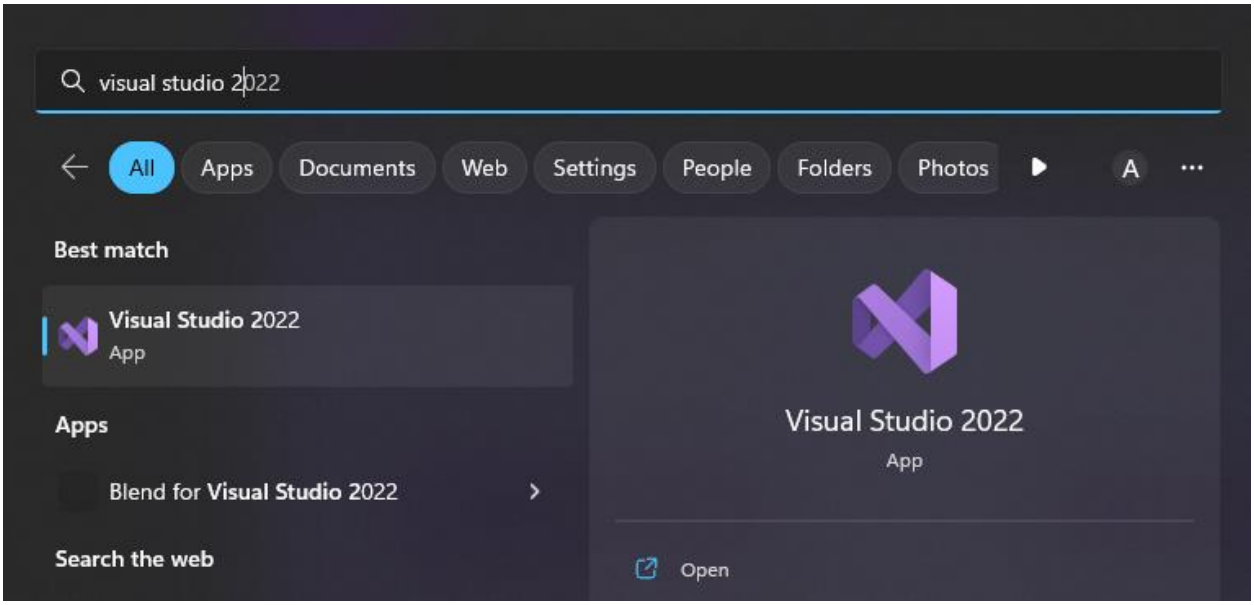
The install process will take 10 minutes.



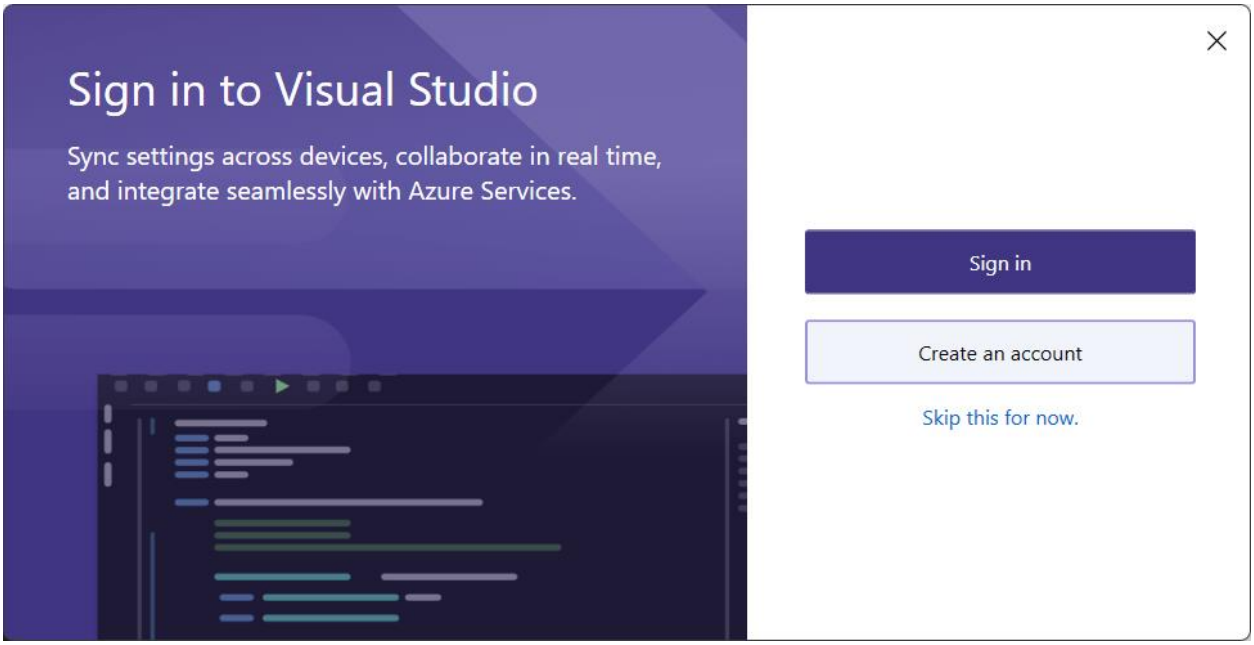
Creating a WinForms Desktop Application

Setup

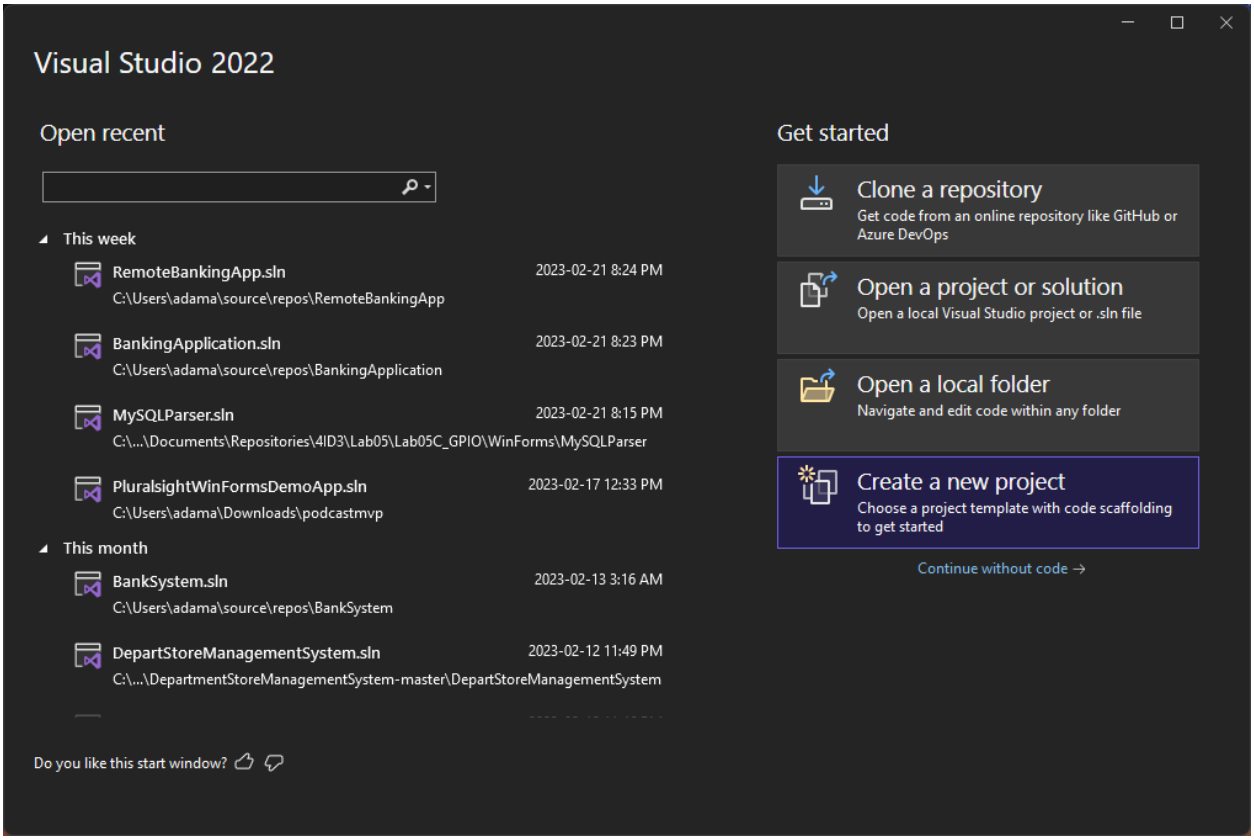
Once the install is completed, **restart** your PC and launch **Visual Studio 2022**.

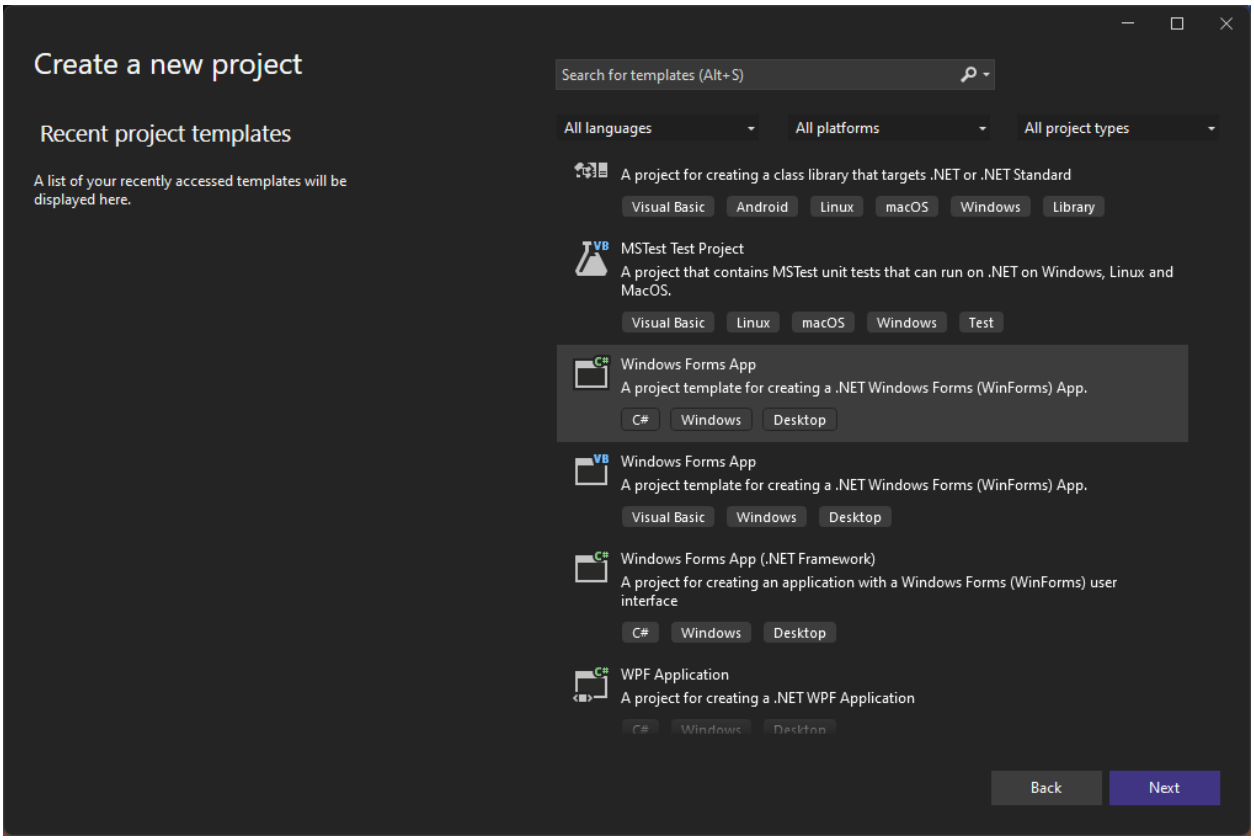


Skip creating an account or use your McMaster email.

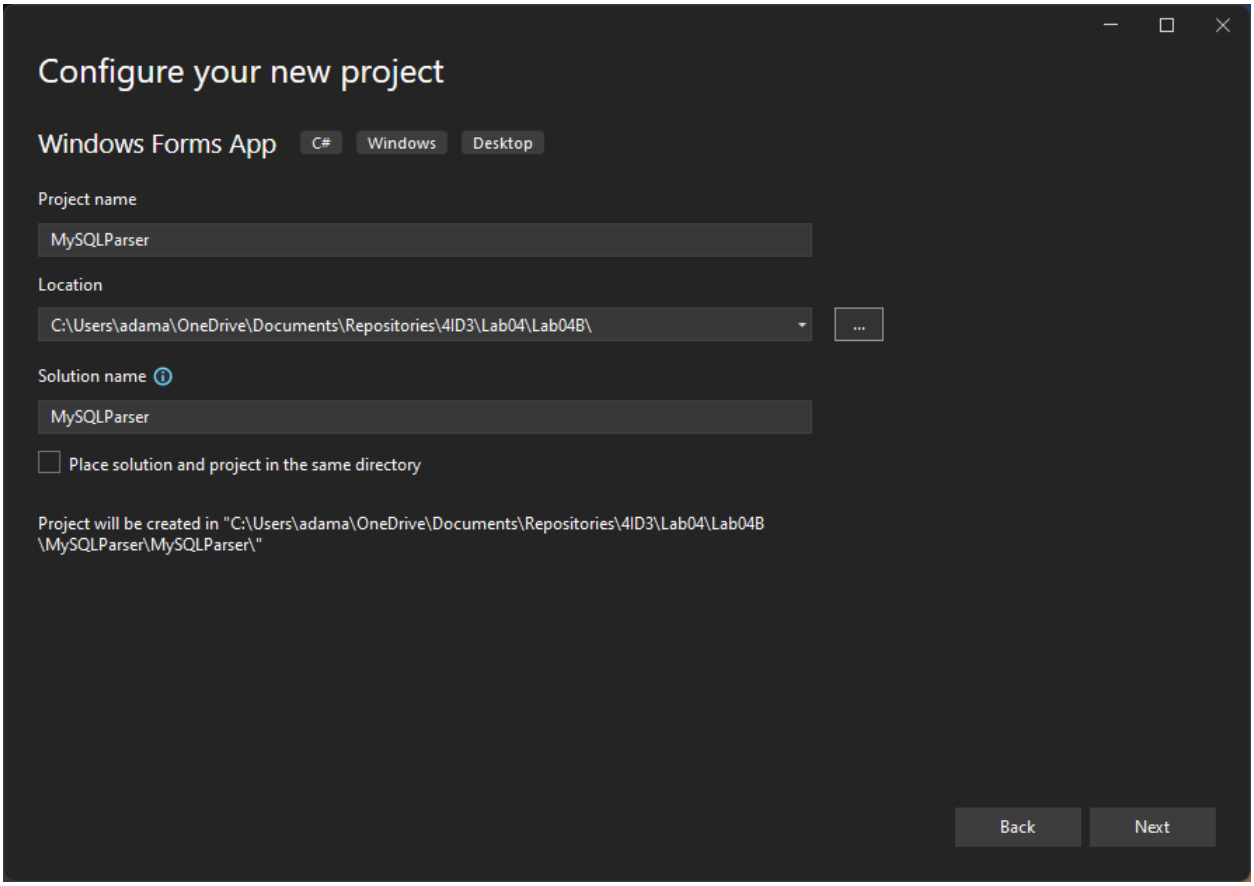


Select **Create New Project > Windows Forms App**.

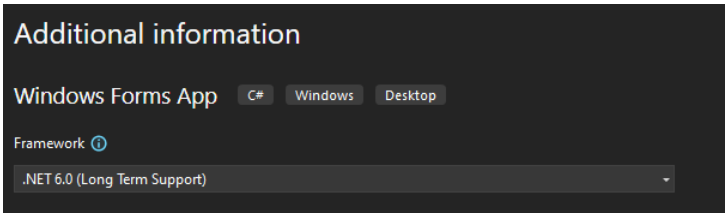

Visual Studio 2022
Open recent
This week
RemoteBankingApp.sln
C:\Users\adama\source\repos\RemoteBankingApp
2023-02-21 8:24 PM
BankingApplication.sln
C:\Users\adama\source\repos\BankingApplication
2023-02-21 8:23 PM
MySQLParser.sln
C:\...\Documents\Repositories\4ID3\Lab05\Lab05C_GPIO\WinForms\MySQLParser
2023-02-21 8:15 PM
PluralsightWinFormsDemoApp.sln
C:\Users\adama\Downloads\podcastmvp
2023-02-17 12:33 PM
This month
BankSystem.sln
C:\Users\adama\source\repos\BankSystem
2023-02-13 3:16 AM
DepartStoreManagementSystem.sln
C:\...\DepartmentStoreManagementSystem-master\DepartStoreManagementSystem
2023-02-12 11:49 PM
Do you like this start window?
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →


Create a new project
Recent project templates
A list of your recently accessed templates will be displayed here.
Search for templates (Alt+S)
All languages
All platforms
All project types
A project for creating a class library that targets .NET or .NET Standard
Visual Basic Android Linux macOS Windows Library
MSTest Test Project
A project that contains MSTest unit tests that can run on .NET on Windows, Linux and MacOS.
Visual Basic Linux macOS Windows Test
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop
Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
Visual Basic Windows Desktop
Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop
WPF Application
A project for creating a .NET WPF Application
Back
Next

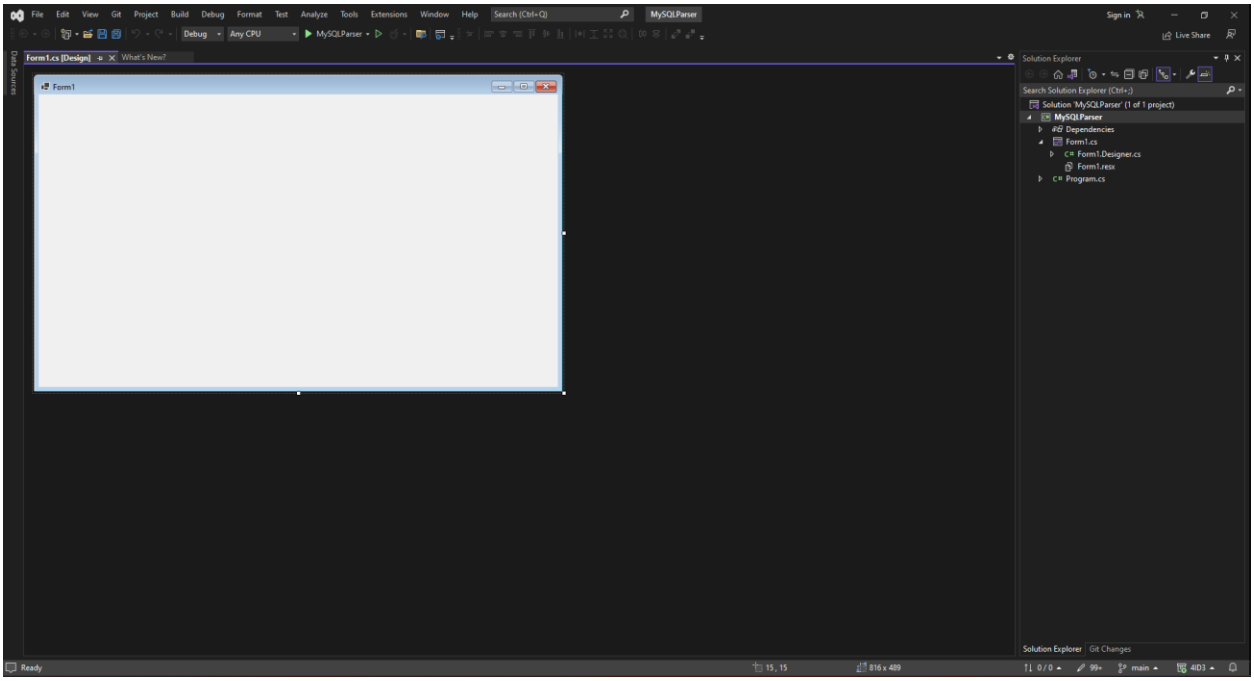
Save your **MySQLParser** desktop application inside of **Lab4B/**.



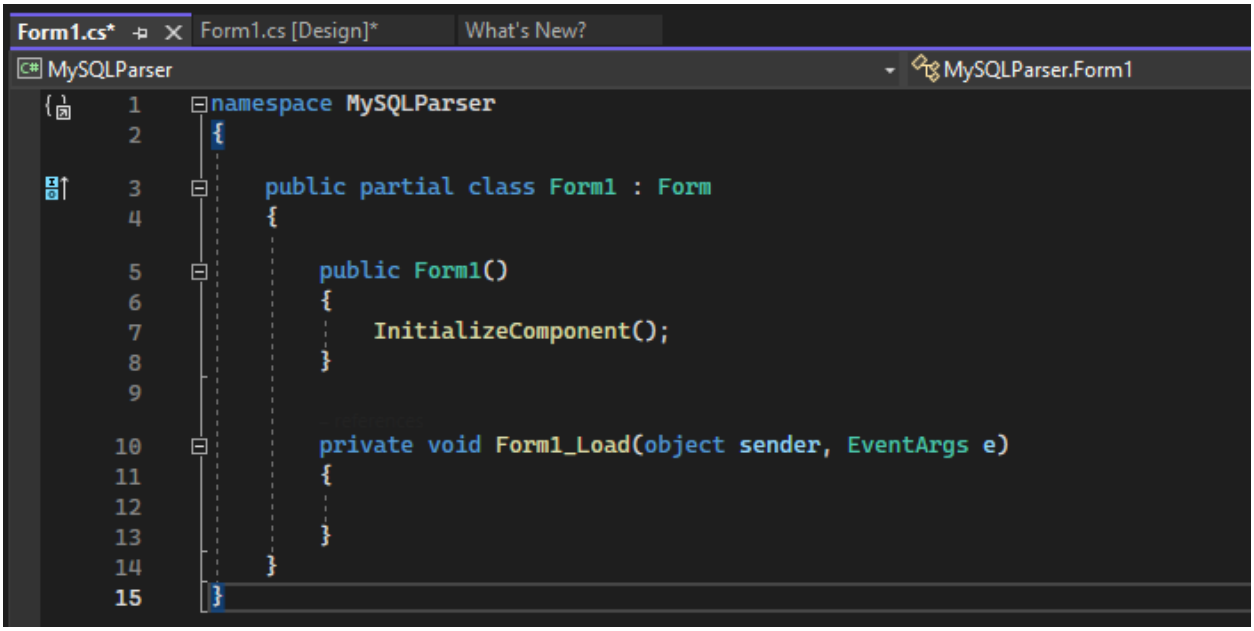
Use **.NET 6.0 LTS**. Press **Create**. It will load for **20 seconds**.



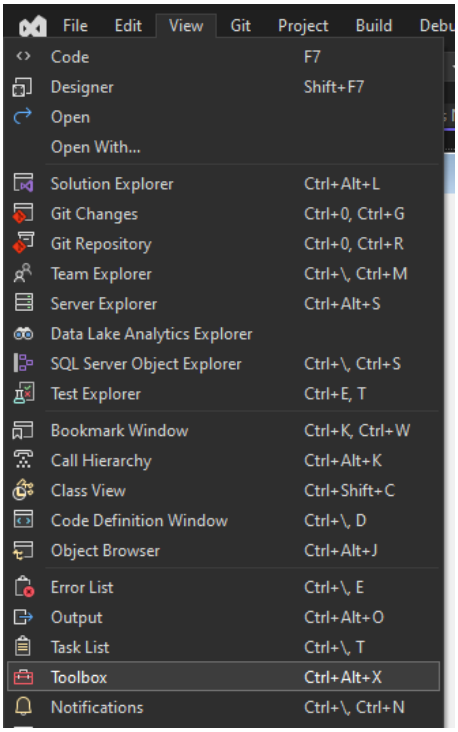
You will be visited by an empty project interface.



Firstly, **double-click Form1** to open the code editor.

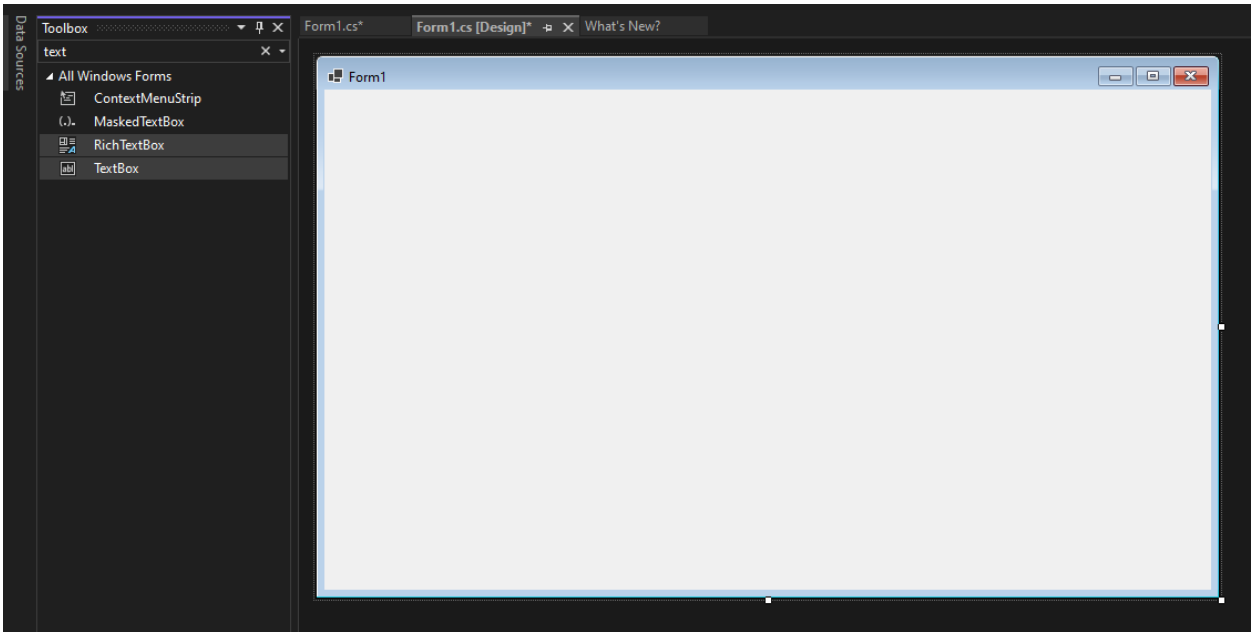


Navigate to **View > Toolbox** to open the toolbox.

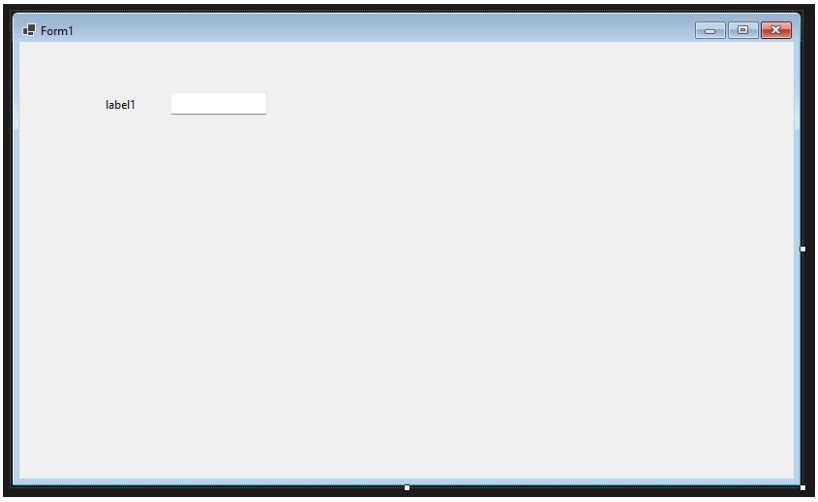


Adding Elements

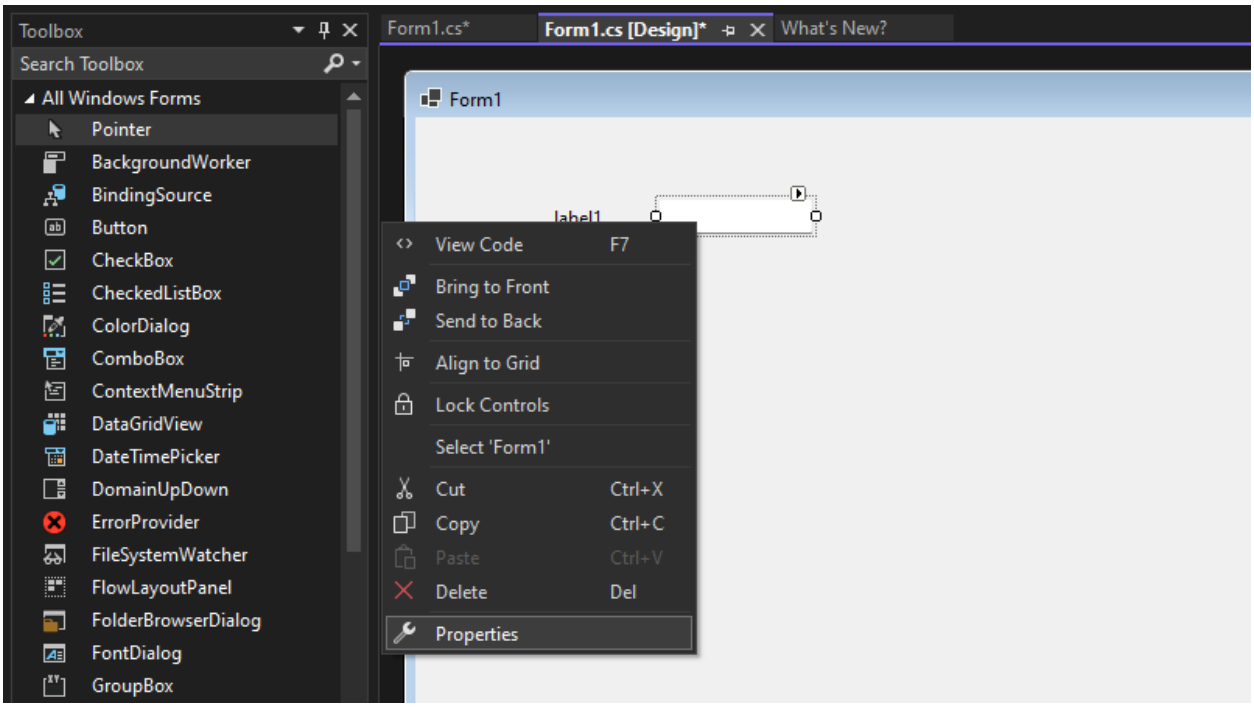
Search **text** and drag a **textbox** into your **Form1**.



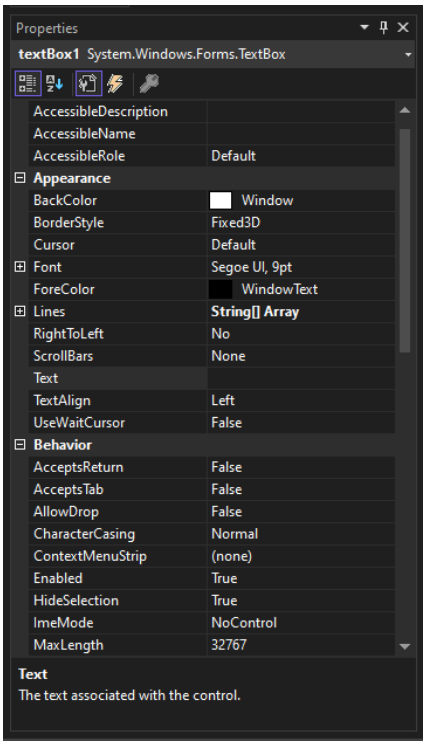
Next, we need a **label** to tell the user what this text box is for. Search and drag a **label** into your **Form1.**



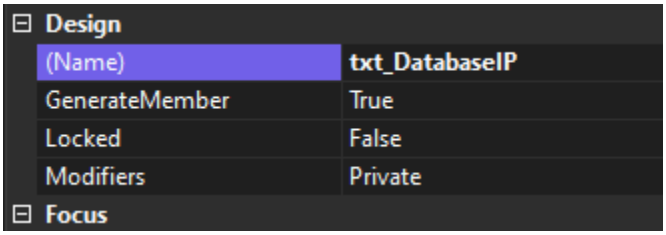
To open the **Properties editor**, **right click** on an element and select **Properties**.



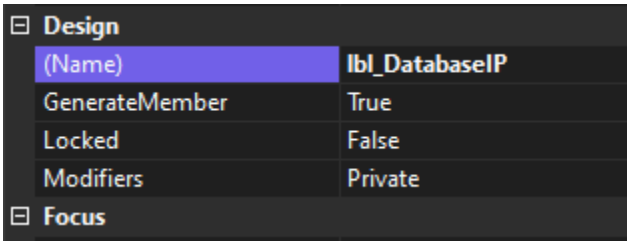
A new window should appear on the **right.**



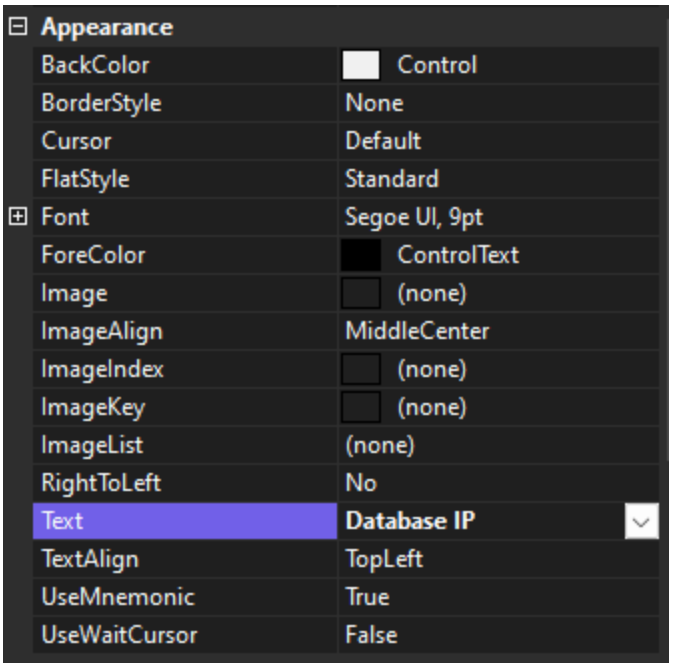
Modify the **Design > (Name)** property to be **txt_DatabaseIP**. This is the value that we will use to reference this object in the code.



Next, click on your **label** and set the **Design > (Name)** property to **lbl_DatabaseIP**.



Finally, the **Appearance > Text** property and set it to **Database IP**. This changes what the label says.



Search for a **button** and drag it into **Form1**.

Change its **Design > (Name)** property to **btn_QueryDatabase**.

Change its **Appearance > Text** property to **Query Database.**

Search for a **Data Grid View** and drag it into **Form1**.

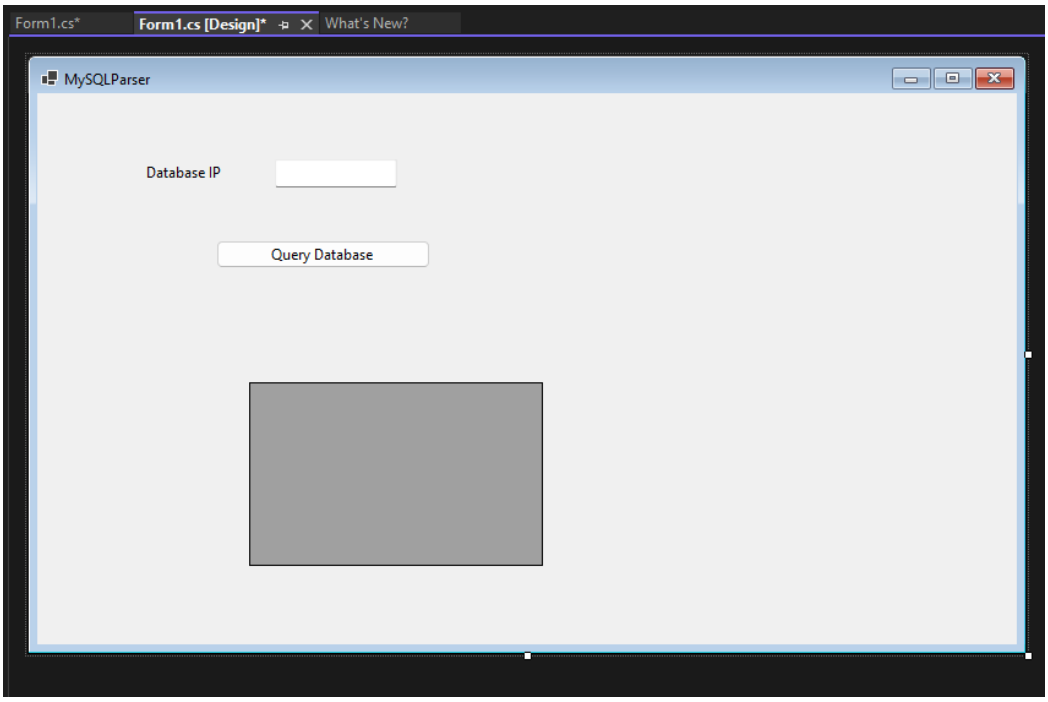
Change its **Design > (Name)** property to **grid_DataView**.

Click on the **Form1** blue window.

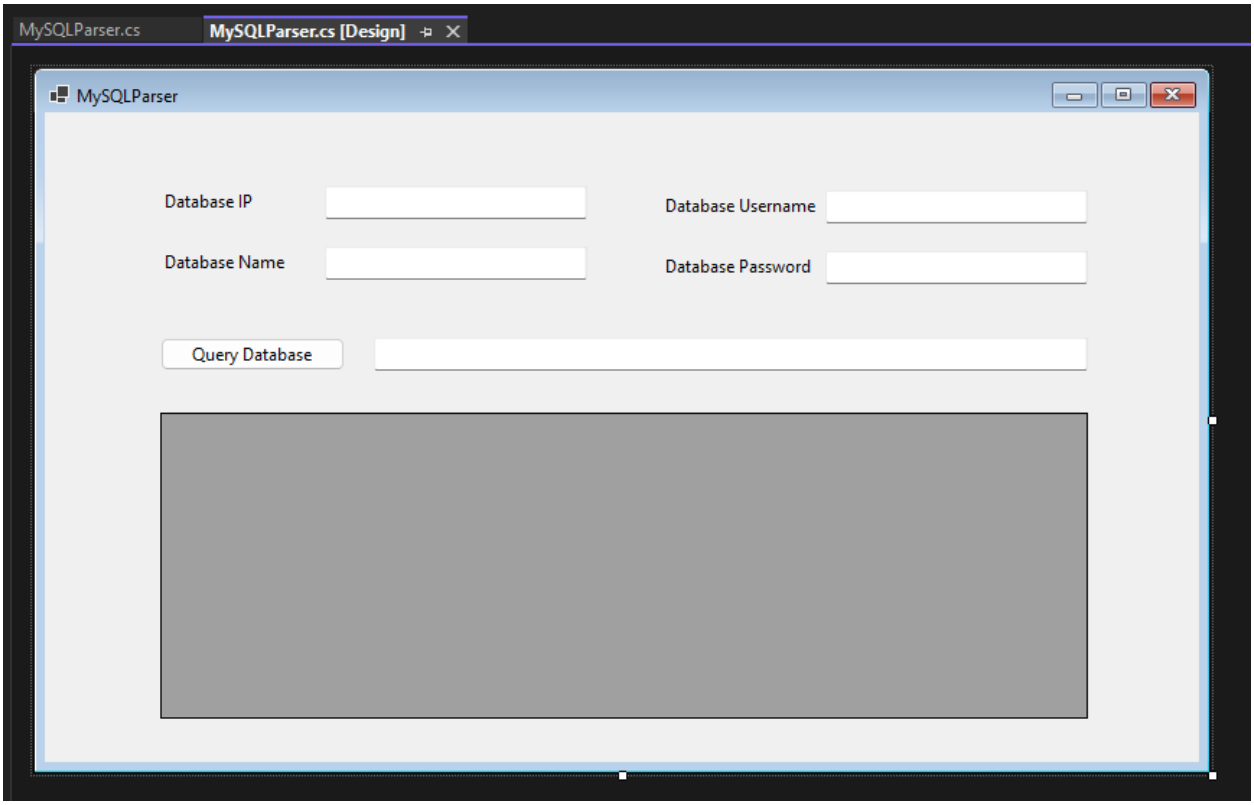
Change its **Design > (Name)** property to **MySQLParser**.

Change its **Appearance > Text** property to **MySQLParser**.

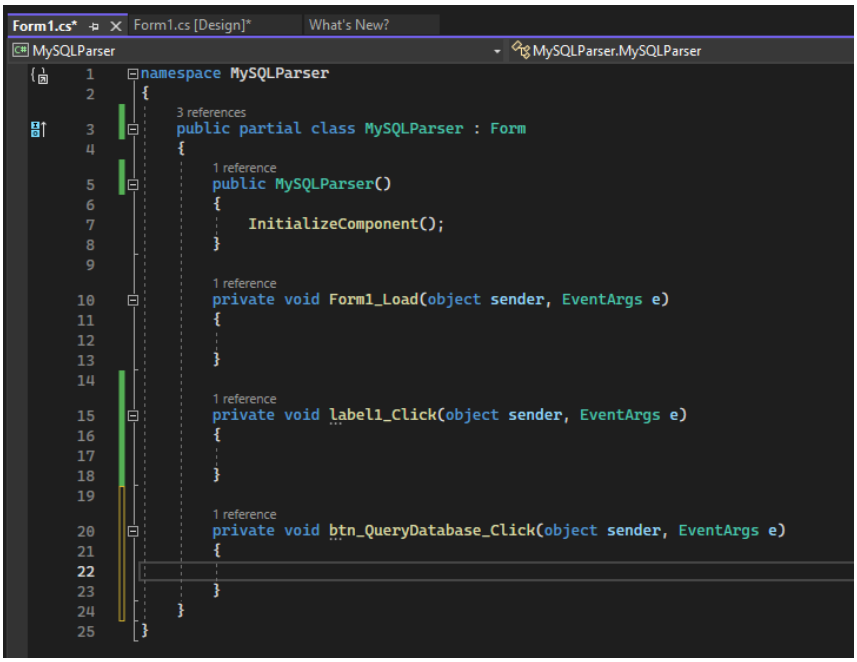
You should have this:



Following the previous steps, add more txt and lbl to create the following app:



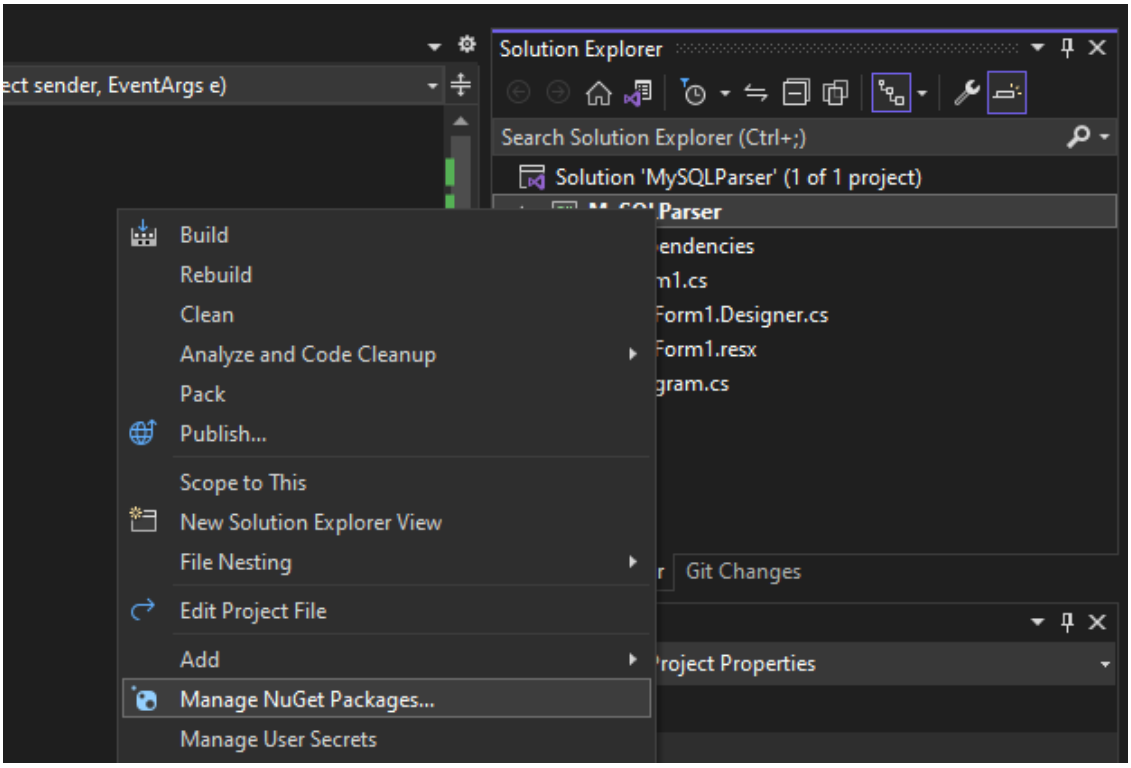
Double-click the **Query Database button** to auto-generate the **callback function**. You should be automatically taken to the **MySQLParser** class.



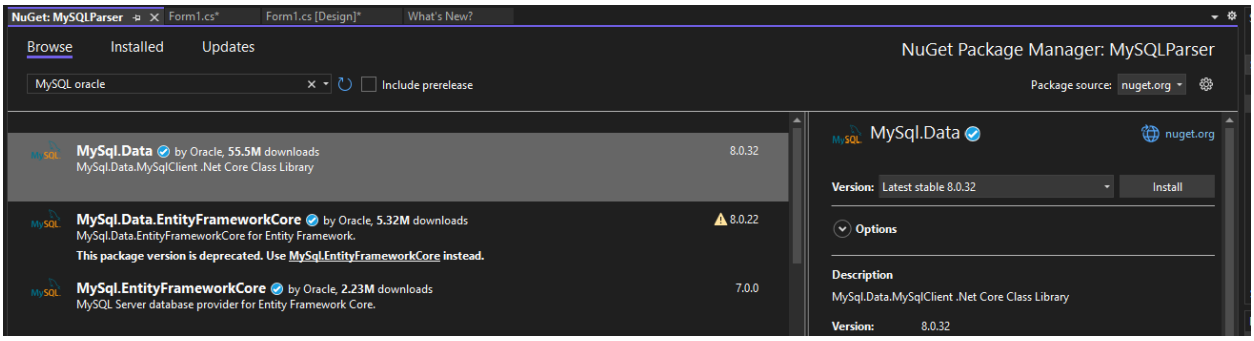
Installing NuGet Packages

In your **Solution Explorer**, **right click** on your project and select **Manage NuGet Packages**.

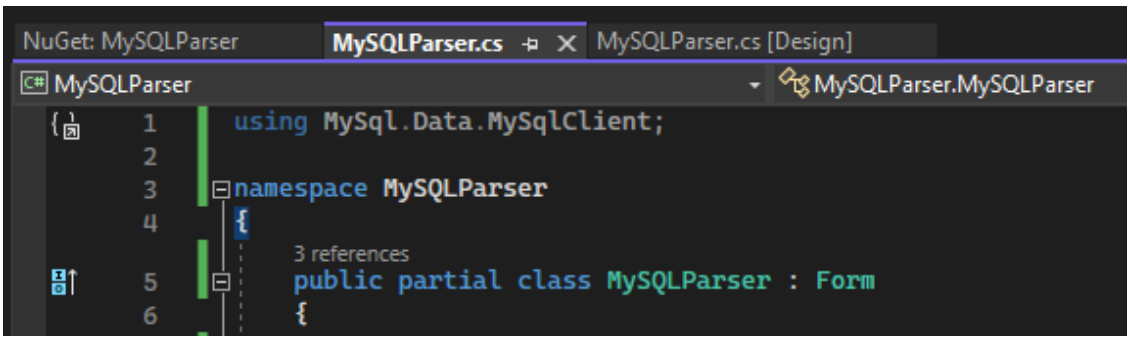
NuGet is a package manager, like pip is for Python.



In the **Browse** tab, search **MySQL Oracle** and select **MySql.Data** by Oracle. Click **Install**.



Back in your **code**, add the installed MySql package.



Creating a Database Model

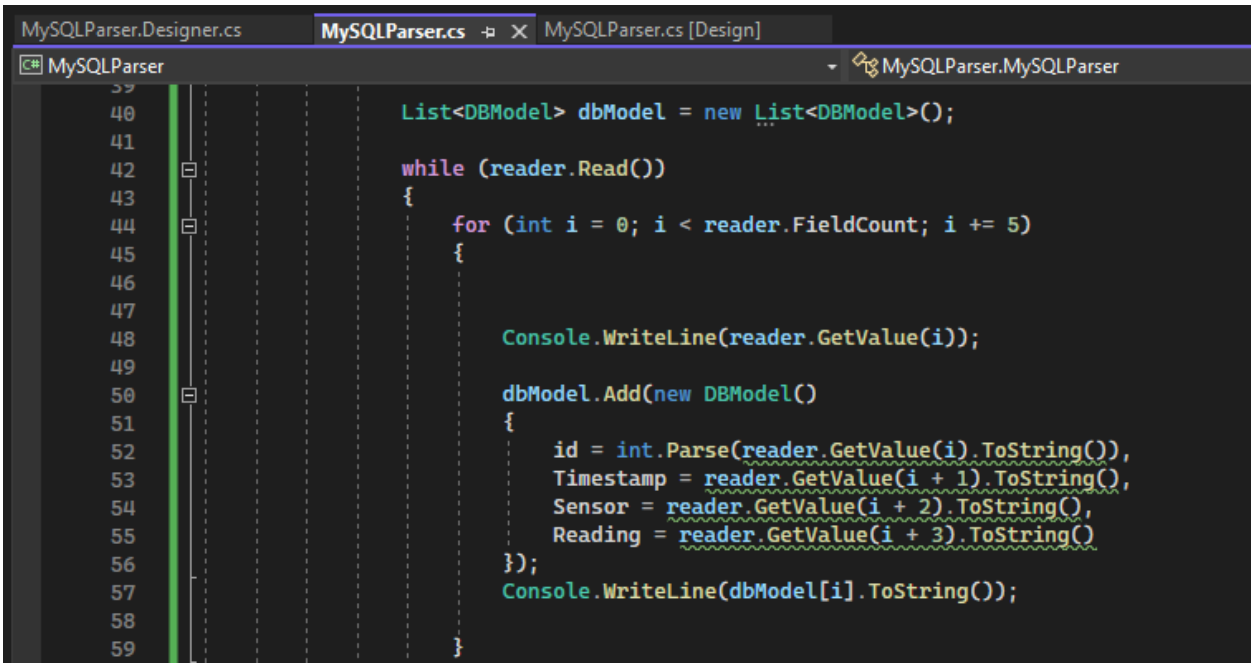
Inside of your **MySQLParser form class**, create a class named **DBModel**.

```
MySQLParser.Designer.cs | MySQLParser.cs* | MySQLParser.cs [Design]*
MySQLParser                                  MySQLParser.MySQLParser.DBModel
1   using MySql.Data.MySqlClient;
2
3   namespace MySQLParser
4   {
5
        3 references
6       public partial class MySQLParser : Form
7       {
            0 references
8           public partial class DBModel
9           {
                0 references
10              public int id { get; set; }
                0 references
11              public string Timestamp { get; set; }
                0 references
12              public string Sensor { get; set; }
                0 references
13              public string Reading { get; set; }
14          }
15
            1 reference
16          public MySQLParser()
17          {
```

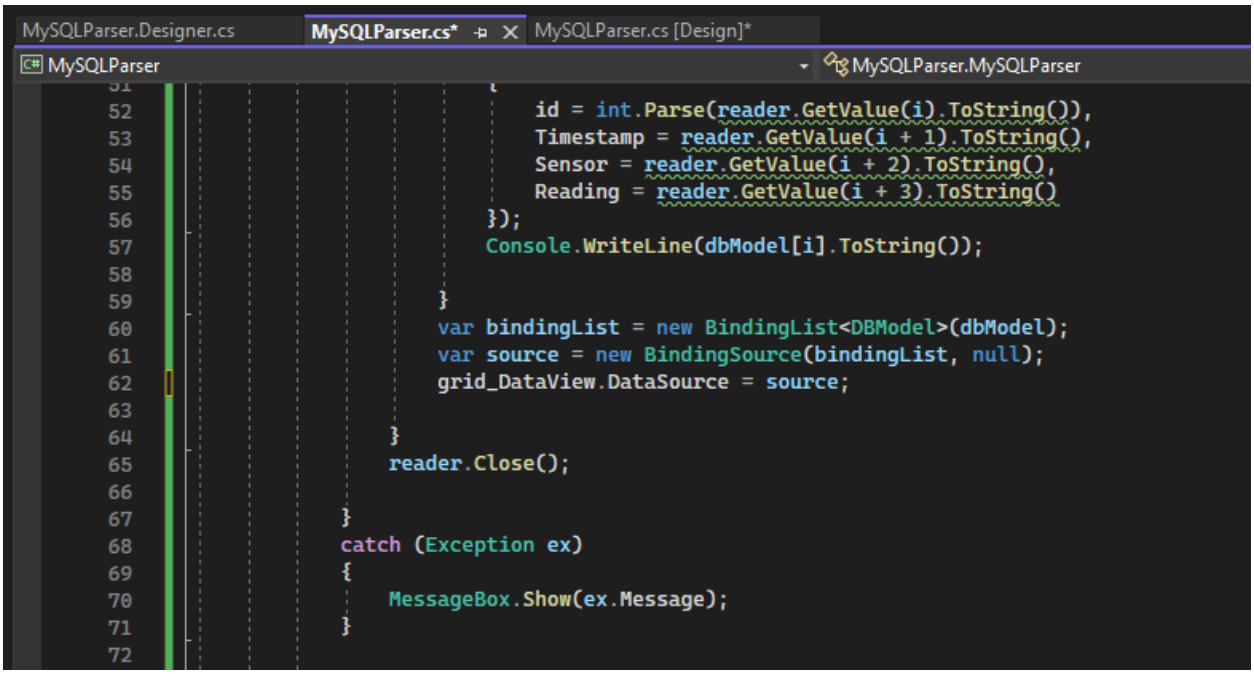
In the onClick callback method for your **Query Database button**, start off by using the MySql library to establish connection with your database by concatenating the provided data to a connection string.

```
MySQLParser.Designer.cs | MySQLParser.cs | MySQLParser.cs [Design]
MySQLParser                              MySQLParser.MySQLParser
25
        1 reference
26      private void btn_QueryDatabase_Click(object sender, EventArgs e)
27      {
28          try
29          {
30              MySqlConnection conn = new MySqlConnection("server=" + txt_DatabaseIP.Text
31                          + ";user id=" + txt_DatabaseUsername.Text + ";password="
32                          + txt_DatabasePassword.Text + ";database=" + txt_DatabaseName.Text);
33
34              conn.Open();
35              String query = "";
36              MySqlCommand cmd = new MySqlCommand(query, conn);
37              MySqlDataReader reader = cmd.ExecuteReader();
```

Next, create an instance of your database model and parse the data retrieved from the database.



Finally, bind the model to your **Data Grid View** and close your **try/catch** block.



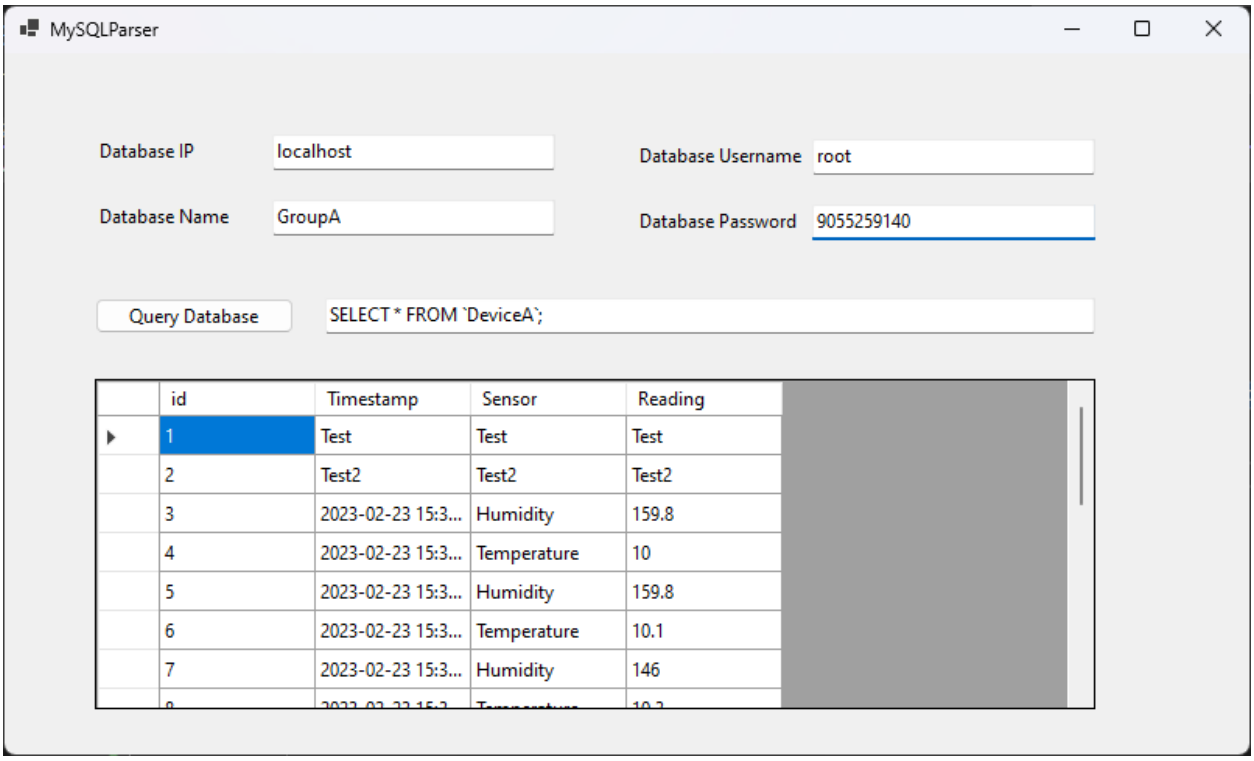
Full code:

```
using MySql.Data.MySqlClient;
using System.ComponentModel;

namespace MySQLParser
```

```
{

    public partial class MySQLParser : Form
    {
        public partial class DBModel
        {
            public int id { get; set; }
            public string Timestamp { get; set; }
            public string Sensor { get; set; }
            public string Reading { get; set; }
        }

        public MySQLParser()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void btn_QueryDatabase_Click(object sender, EventArgs e)
        {
            try
            {
                MySqlConnection conn = new MySqlConnection("server=" + txt_DatabaseIP.Text
                            + ";user id=" + txt_DatabaseUsername.Text + ";password="
                            + txt_DatabasePassword.Text + ";database=" + txt_DatabaseName.Text);

                conn.Open();
                String query = txt_QueryDatabase.Text;
                MySqlCommand cmd = new MySqlCommand(query, conn);
                MySqlDataReader reader = cmd.ExecuteReader();


                List<DBModel> dbModel = new List<DBModel>();

                while (reader.Read())
                {
                    for (int i = 0; i < reader.FieldCount; i += 5)
                    {


                        Console.WriteLine(reader.GetValue(i));

                        dbModel.Add(new DBModel()
                        {
                            id = int.Parse(reader.GetValue(i).ToString()),
                            Timestamp = reader.GetValue(i + 1).ToString(),
                            Sensor = reader.GetValue(i + 2).ToString(),
                            Reading = reader.GetValue(i + 3).ToString()
                        });
                        Console.WriteLine(dbModel[i].ToString());

                    }
                    var bindingList = new BindingList<DBModel>(dbModel);
                    var source = new BindingSource(bindingList, null);
                    grid_DataView.DataSource = source;

                }
                reader.Close();

            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }

        }
```

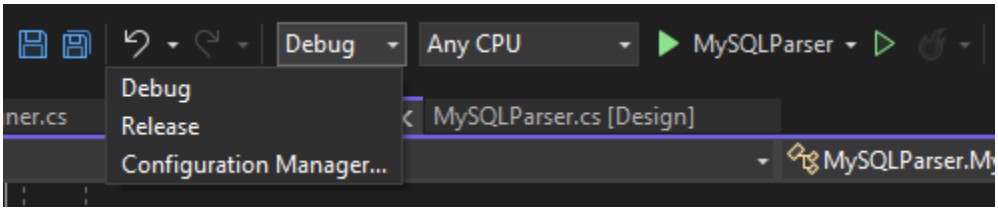
```
        }
    }
```

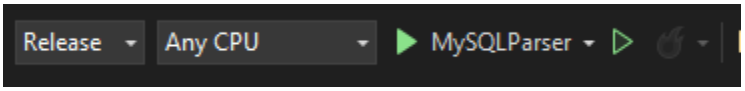
Run your program in debug mode and test it out.



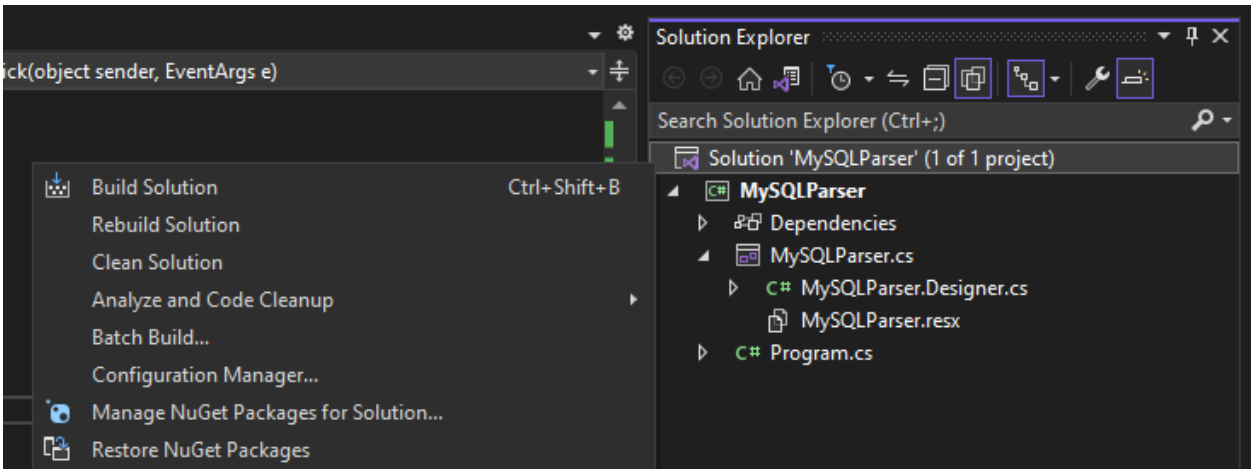
Lastly, lets build it in release mode.

Change your build configuration to **Release**.

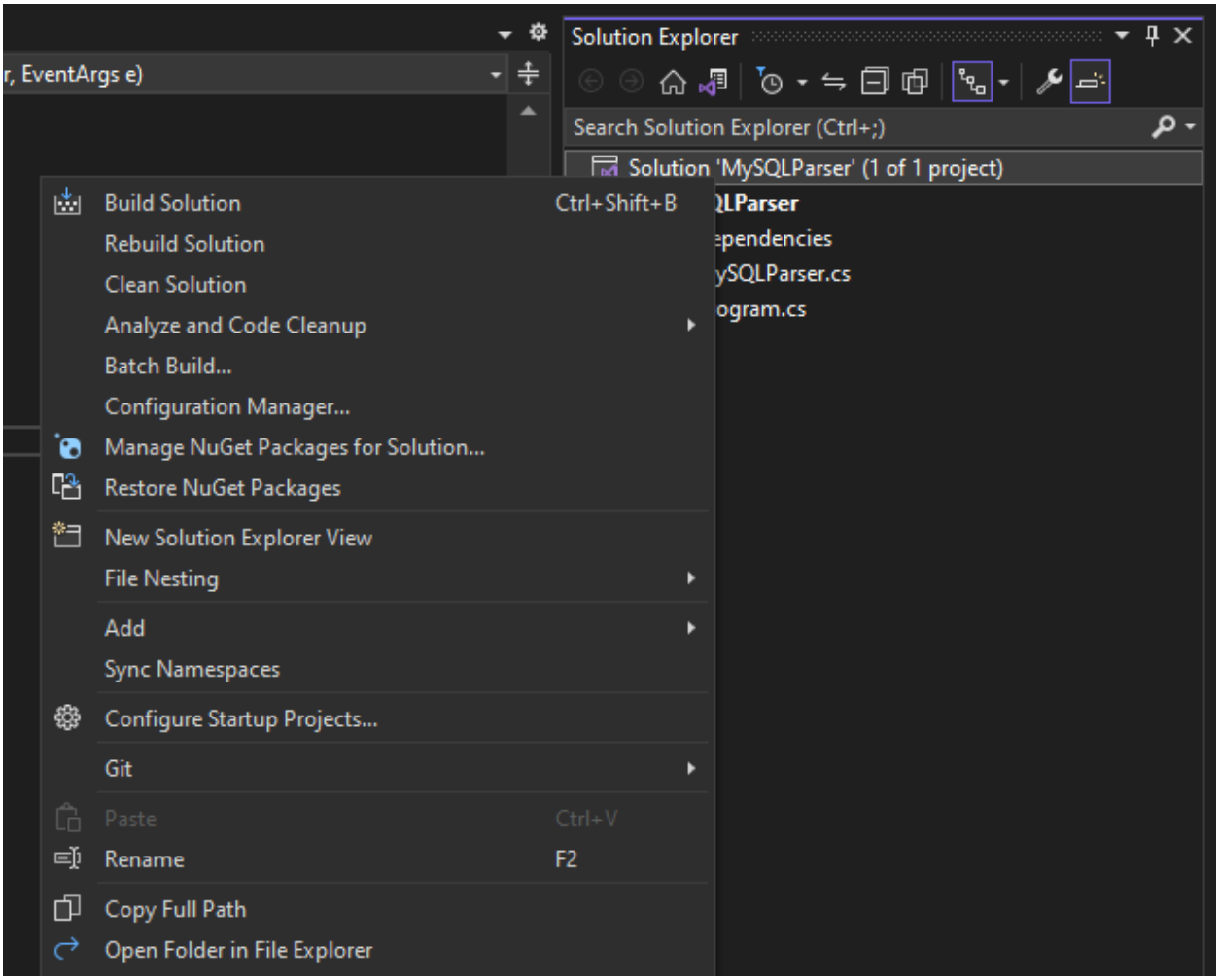


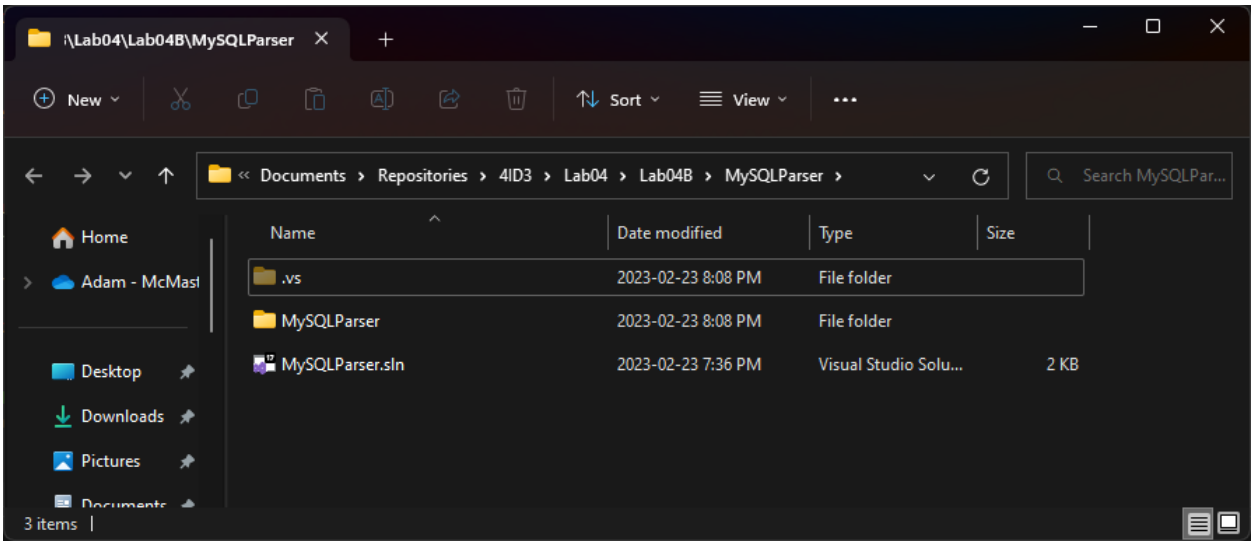


Right click on your project and select **Build Solution**.

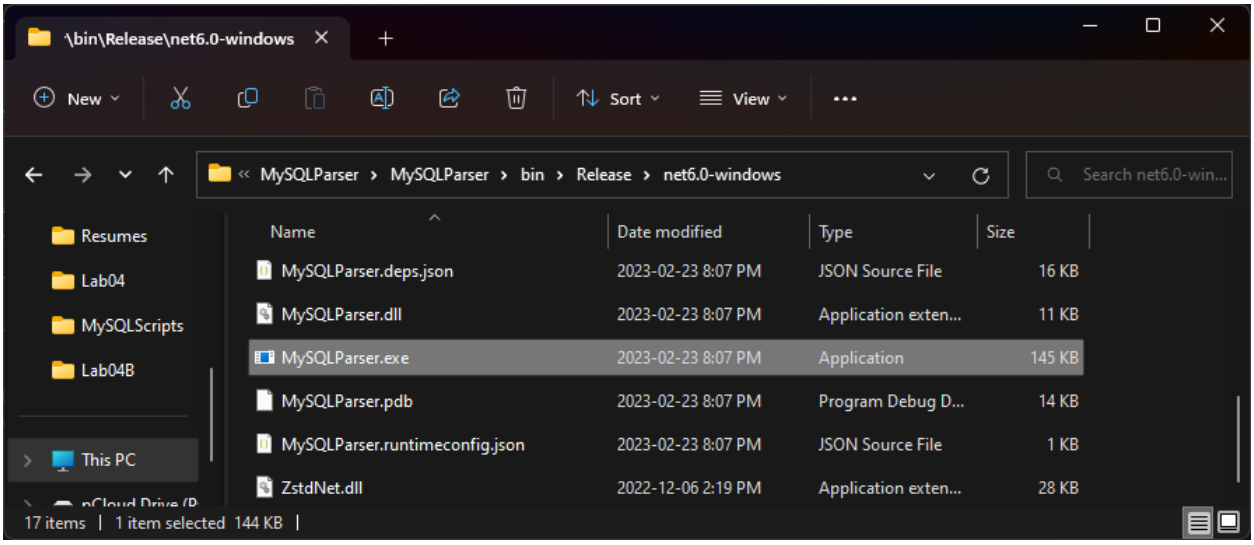


To find the built program, **right click** on your **Solution** in the **Solution Explorer** and click **Open Folder in File Explorer.**

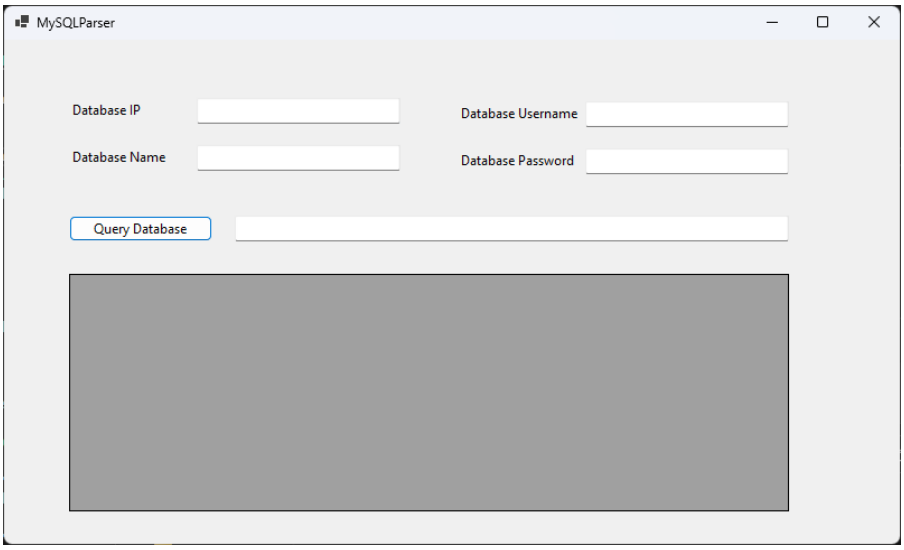




Navigate to **MySQLParser > bin > Release > net6.0-windows** and find **MySQLParser.exe**.



Double click to **run** the program.


MySQLParser
Database IP
Database Username
Database Name
Database Password
Query Database

Syncing with GitHub

Use the following commands to resync your Git repo with your GitHub remote repo.

git add .

git commit -m "Lab 04 completed"

git push origin main

git pull origin main

END