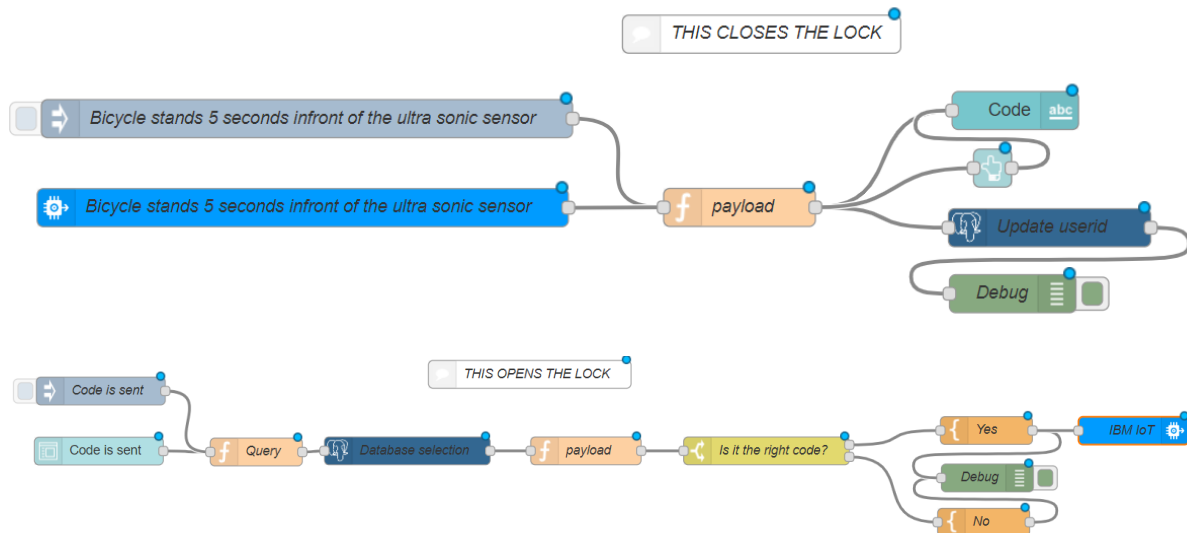


Node-RED

The node-RED part for this project consist of the two flows. We have a flow that closes the lock, gives a code to the user and saves this code in the databse. And we have a flow that opens the lock if the right code is entered in the system. The system is made for one lock, but it can be refactored to work on multiple locks.



First flow: closing the lock

We will start with making the first flow (the flow that closes the lock).

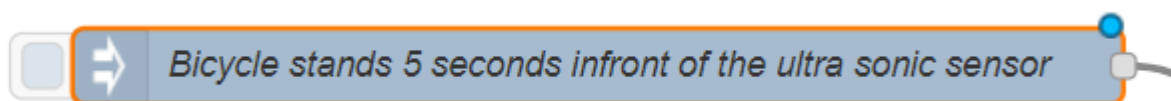
Setup

For this part we will be using a database. You can use any sort of database, we used a postgresql database in this project. Depending on your database download the right plugin in node-RED to manage your database. In our case the plugin is called *postgrestor*.

Our database only has one table parkingSpot with 3 columns, the id of the spot, a boolean isAvailabe and a code (in our case we called it userId).


Step 1

Start by making an injection node. This will be used to simulate the bicycle standing infornt of the sensor, it will also be used for debugging the project. We will later make the implementation for the actual device.



Step 2

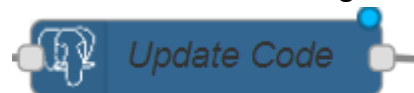
Make a function node with code to generate a four digit number. In this tutorial we will give you the code to do this.

 **Function** 

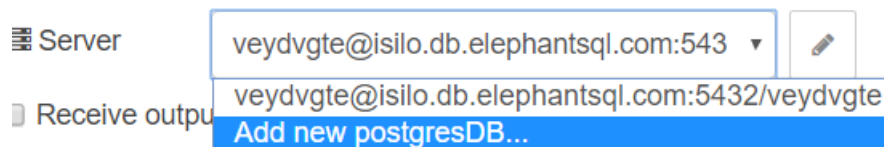
```
1 msg.topic= Math.floor(1000 + Math.random() *8999);
2 return msg;
```

Step 3


We will need to save the generated code to the database, so we need a database node.



You need a database configuration, this depends on what kind of database you are using.



In postgresql we can save our generated 4 digit number like so.

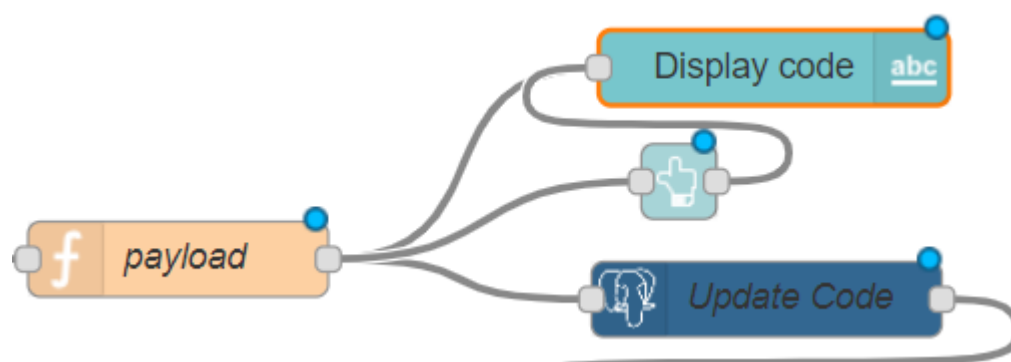
 **Query**

```
1 update parkingspot set "userId" = {{msg.topic}};
```

You can add a debug node to the database node, because it can give a lot of problems, but this is not necessary.

Step 4

We use the IBM Bluemix dashboard plugin (needs to be installed if you don't have it) to create an ui for the project. The ui is very basic and we are not going to go in any detail about it because you can add your own ui if you want to. You can connect your function with the generated code to a node that displays the code, like so.



Step 5

Your logic should work for now, we will show you how to connect a real device later.

Second flow: opening the lock

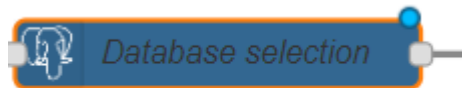
Step 1

Add a node that sends a code, I recommend to also make an injection for testing purposes. We used the form of the dashboard plugin, but this is entirely up to you. We are not going to go into any detail about ui in this tutorial.



Step 2

We need to make a query in the database to check if the code is that is entered is the right one. We used this query. Ofcourse you can make your own query if you have a different database model.



Query

```
1 select *
2 from parkingspot
3 where "userId" = {{msg.payload.code}};
```

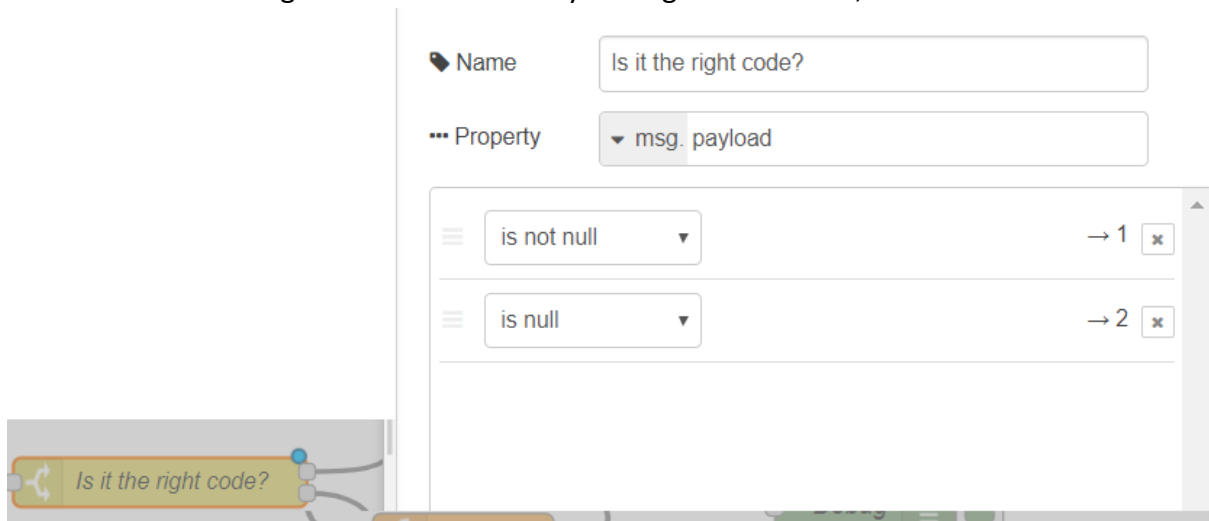
Step 3

After this we need to make a new function that returns the rows that got selected. You can check the image at the top where to place the flows exactly.

Function

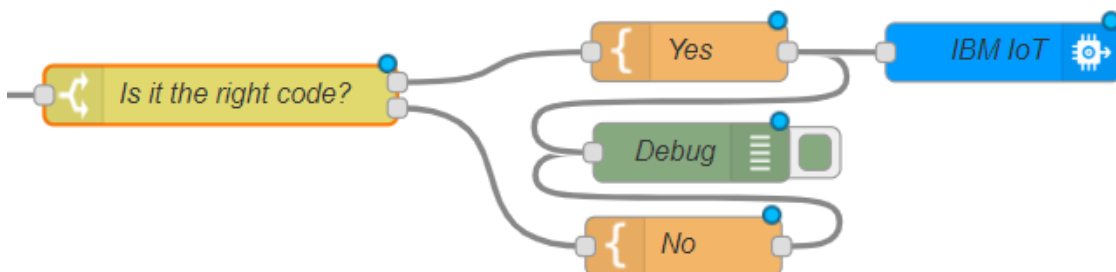
```
1 return {payload:msg.payload.rows[0]};
```

And we need to check if the result is null or not null. If it is null the code is wrong but if it is not null the code is right. You can do this by adding a switch case;



Step 4

After this we need to add logic to the different cases. You need to unlock the lock if the code is right (upper box). Add an ibmiot node, the setup will be in the next part. You can add a template between them, but this is optional. I would also recommend to add a ui element to send the result if the code is right or wrong (we did not do this in this project).



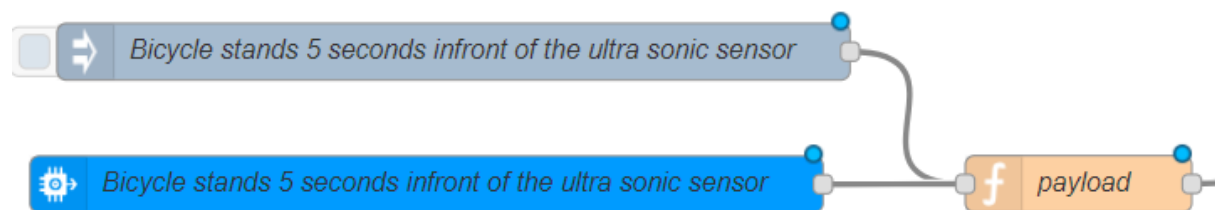
Adding the devices

Setup

You will need to register a new device on the platform, but I would recommend following a different tutorial to do that. We gave “sonic-sensor-arduino” as a device id and “sensor” as device type. The token that you get with creating your device is very important so make sure to save it. (you need it for the arduino)

Step 1

Add a ibmiot device to the **first flow**.



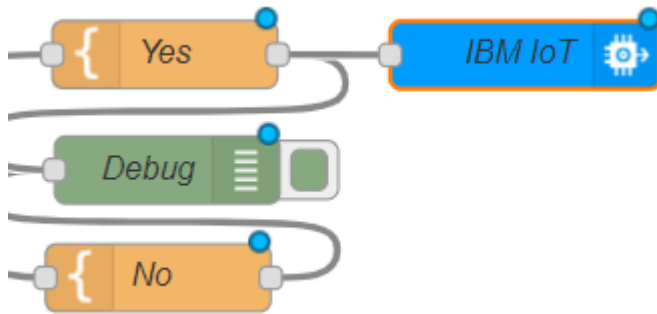
It’s important that the input type is on device event and that the event is status.

🔑 Authentication	Bluemix Service ▼
⚙️ Input Type	Device Event ▼
🚀 Device Type	<input type="checkbox"/> All or <input type="text" value="sensor"/>
👤 Device Id	<input type="checkbox"/> All or <input type="text" value="sonic-sensor-arduino"/>
📋 Event	<input type="checkbox"/> All or <input type="text" value="status"/>
📄 Format	<input type="checkbox"/> All or <input type="text" value="json"/>
⚙️ QoS	<input type="text" value="2"/> ▼
🏷️ Name	<input type="text" value="Bicycle stands 5 seconds infront of the ultra sc"/>

Your first flow is done.

Step 2

Now double click on the iotdevice that you previously created in the second flow.



The output type has to be a device command, and the command type (in this case) is update.

Edit ibmiot out node

Delete Cancel Done

Properties

Authentication	Bluemix Service
Output Type	Device Command
Device Type	sensor
Device Id	sonic-sensor-arduino
Command Type	update
Format	json
Data	json
QoS	0
Name	IBM IoT

The second flow is also done.

Step 3

Click on deploy in the upper right corner and you are done.

