# Smart-park-with-Astro

**Why Astro?**

Astro was created to solve problems like:

* Time and fuel wastage while looking for a parking spotInefficient use of parking spaces where it is not requiredOver paying to save the parking spot

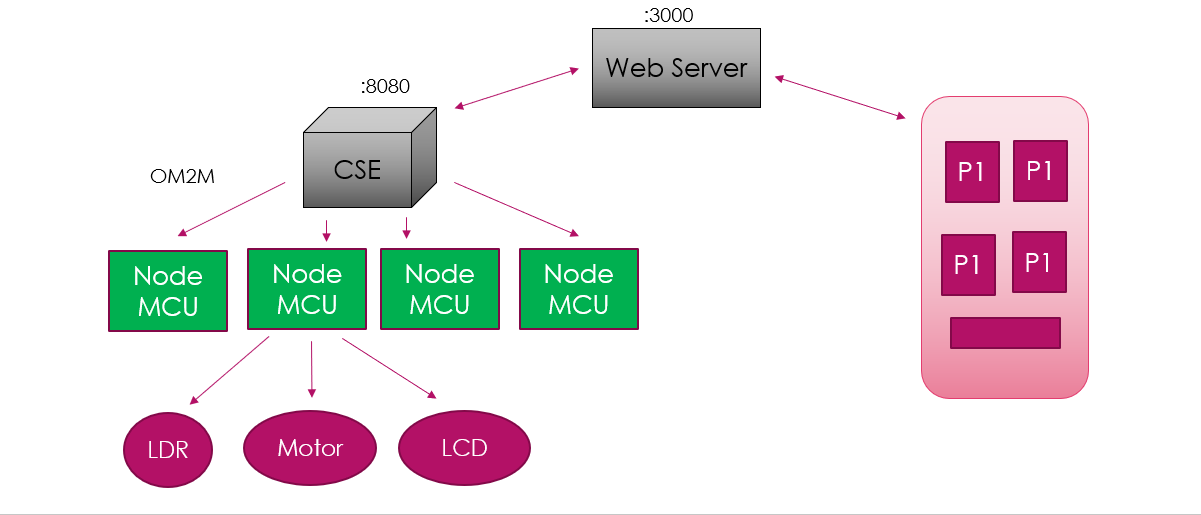
**Issues Astro can help us with**

* Relax and know your spot before] you get thereDon’t worry about forgetting allotted time, Astro will remind youSave time in finding a parking spotPlan better for parking places, better utilization of space available with the data collectedAnalyze parking patterns over a period of time over specific regionsPredict the parking availability using historical data

**Follow this to create what we created.**

**1. High level architecture**

Smart Parking application using OM2M Platform



**2. Prerequisites**

The following tools are required to run this demonstration.

* JAVA 1.8Arduino IDE 1.8.5 or Eclipse IDE (Version Neon along with "Eclipse C++ IDE for Arduino 2.0" plug-in)NodeJSYour favorite IDE**3. Download the code attached**

**4. Start oneM2M platform instance**

The oneM2M platform is available on folder “oneM2M Platform”

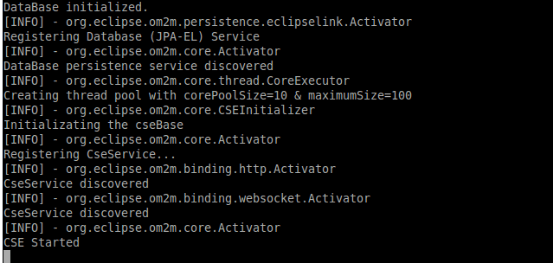
**4.1. Configure the platform**

You can keep the default configuration for a local demonstration. The platform will listen on ports 8080 and 8443. The database is reset after each restart.

If needed, you can change the configuration of the platform by editing the file “configuration/config.ini”.

**4.2. Start the platform**

Start the platform by executing the "start.bat" script on Windows or "start.sh" on Linux and Mac OS.



**4.3. Login to oneM2M web interface**

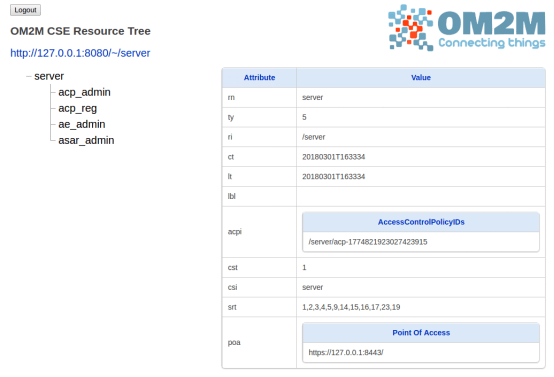
Open the following URL in your browser to access the oneM2M platform web interface: http://127.0.0.1:8080/webpage

Enter "Cae-admin" as originator then click on “connect” .



**5. Schematics**

Connect to your IoT devices, the arduino source code and required libraries are available on the folder onem2m-device.



**5.1. Components for Astro**

* Light sensor to determine if a parking spot is availableLED and LCD to indicate availability statusServo motor to lock and unlock the parking spotNodeMCU to send sensor data to CSEArduino UNO to provide 5 volts to power LCD and servomotor**5.2. Download and install arduino IDE**

Download and install the Arduino IDE v1.8.5 from the following link:

https://www.arduino.cc/en/Main/Software

NB: We don’t recommend to use the Arduino Web Editor for this demonstration because the nodemcu board is not supported.

If the NodeMCU port is not detected, then you need to download and install the USB driver manually using the following link:

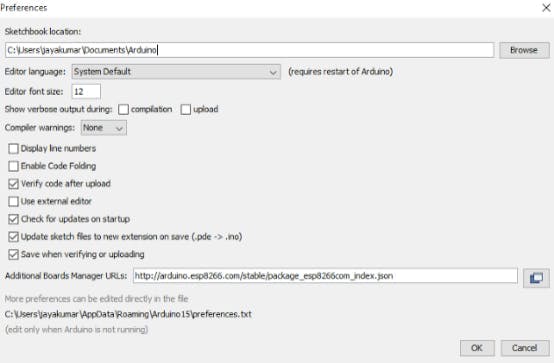
https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers

**5.3. Add Nodemcu board to Arduino IDE**

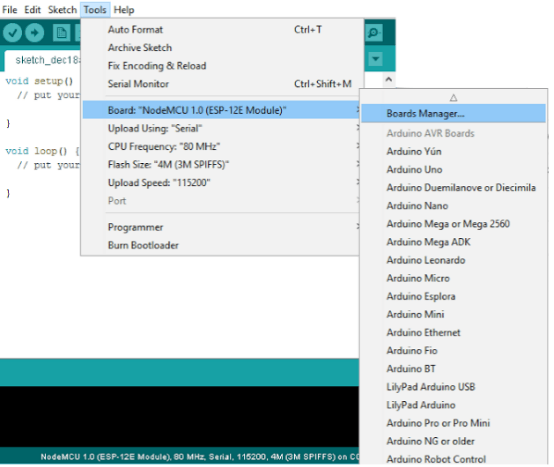
Firstly open the Arduino IDEGo to files and click on the preference in the Arduino IDE

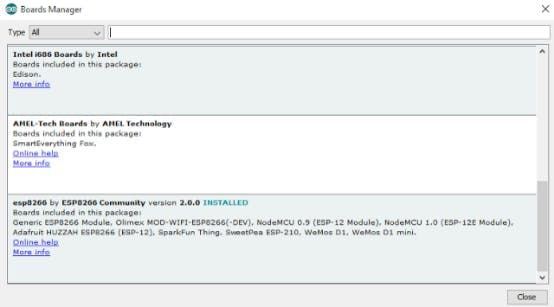
Copy the below URL in the Additional boards Manager URLS: http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

Click OK to close the preference Tab.



After completing the above steps, go to Tools and board, and then select Board Manager.

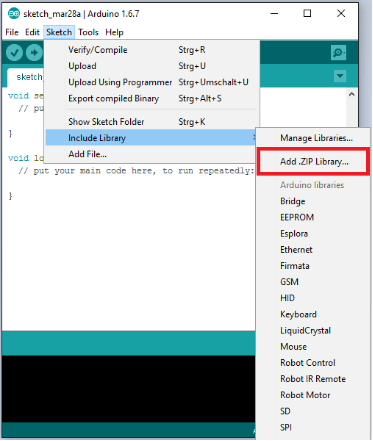
Navigate to esp8266 by esp8266 community and install the software for Arduino.

Go to Tools and board, and then select "NodeMCU 1.0" board.

Once all the above process been completed you are ready to program the nodemcu board with Arduino IDE.

**5.4. Add Arduino “Timer.h” library**

The “Timer.h” library is available on the zip folder “Timer-master.zip”. Go to Sketch, and then select “include library”. Chose the “Add .ZIP library” option:



Select the zip folder “Timer-master.zip” then confirm.NB: You don’t have to extract the zip content.

**5.5. Configure the oneM2M sketch**

Open the sketch “onem2m-adn.ino” in your Arduino IDE.

Set your WIFI parameters:

const char\* ssid = "XXXXXXXX";

const char\* password = "XXXXXXXX";

Set the IP address of the oneM2M platform:

const char\* host = "XXXXXXXX";

**5.6. Compile the oneM2M sketch**

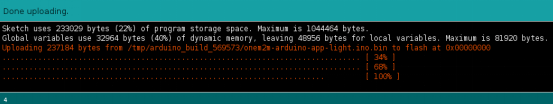
Click on verify button to compile the sketch.

Compilation output:



**5.7. Upload the oneM2M sketch to Nodemcu board**

Click on upload button to upload the sketch to the nodemcu board.

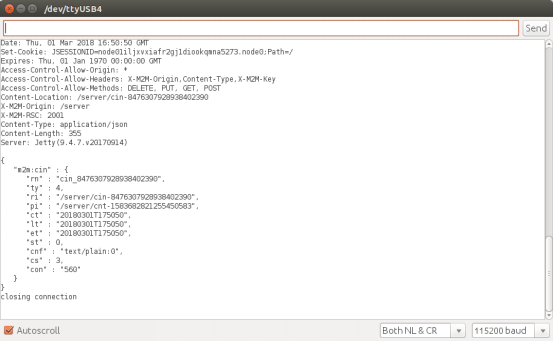


**Upload output:**

**5.8. Debug the code execution using serial monitor**

Click on button to open the Arduino Serial Monitor to display the Nodemcu console.

Then after opening the Serial Monitor select 115200 from the drop-down list.



**5.9. Check IoT device resources on oneM2M web interface**

You should see “mydevice1/2/3/4” Application Entity resource with “luminosity” and “led” containers created on the oneM2M web interface.

**6. Connect your IoT application**

The luminosity monitoring application source code is available on the folder onem2m-app.

**6.1. Configure application**

Open the file onem2m-monitor.js with your favorite javascript or text editor. Set the IP address of the oneM2M platform. You can keep localhost if you are running the oneM2M platform and the application in the same machine.

**6.2. Start the application**

Install the following node modules using the npm tool:

npm install express

npm install request

Start the nodejs oneM2M application using the following command:

node onem2m-monitor.js

**7. Demonstration**

If you hide the luminosity sensor with your hand for few seconds that means the car was parked, you should see the status of the spot as available in the app and if it is not covered that means the spot is available and it will show as green in the app. Also, the same will be displayed on the LCD screen.

**Code**

* Main JavaOneM2m-Monitor.jsonem2m-adn.ino

**Code for app:**

* MainActivity\_java
* activity\_main
* activity\_unlock
* Unlock