Actividad 3.2: Uso de MQTT y Node-Red

Dispositivos y Sistemas Inalámbricos

Francisco M. Delicado Martínez*

Depto. Sistemas Informáticos

Escuela Superior de Ingeniería Informática de Albacete

Universidad de Castilla-La Mancha

Índice

*francisco.delicado@uclm.es

1. Objetivos

El objetivo de esta actividad es que el alumno sea capaz de comunicar las lecturas de los sensores realizadas en un "*mote*"LoPy4©a un bróker implementado en `Node-Red` , mediante el uso del protocolo MQTT. Además se deberá implementar un panel de control en `Node-Red` para mostrar los datos capturados por los sensores, además de interactuar con el LoPy4©.

El objetivo general de la actividad se puede dividir en los siguientes sub-objetivos a alcanzar:

1. Ser capaz de leer los datos proporcionados por un sensor.
2. Conectar un micro-controlador LoPy4©a una red WiFi.
3. Enviar y recibir notificaciones a un bróker utilizando el protocolo MQTT.
4. Diseñar un bróker en `Node-Red` que disponga de un panel de control.

2. Requerimientos Hardware y Software

La realización de la práctica requiere de la utilización de un dispositivo autónomo dotado de sensores, lo que comúnmente se conoce como "*mote*". Además de un entorno de programación que nos permita el desarrollar el programa e implantar dicho código en el "*mote*". A continuación se da una lista de los requerimientos tanto hardware como software que son necesarios.

2.1. Requisitos Hardware

Los dispositivos físicos necesarios de los que deberemos disponer para la realización de la práctica son:

Un PC: en el que se desarrollará la programación de la aplicación a instalar en el "*mote*". El SO a ejecutar en el PC podrá ser Windows, Linux o MacOS. Aunque se recomienda el uso de Linux.

Mote LoPy4©: es un micro-controlador compatible con MicroPython [5]. Se puede obtener información sobre este dispositivo en [8], y la Figura 1 muestra un esquema del dispositivo.

Figura 1: Esquema de un micro-controlador LoPy4©.

Placa de extensión Pysense©: es una placa de extensión para el micro-controlador LoPy4©que entre otras cosas incorpora: un puerto serie USB para su comunicación con un PC, un acelerómetro para los tres ejes, un barómetro, un sensor de luminosidad ambiental, y un sensor de humedad y temperatura. Además también incorpora una bahía para una tarjeta MicroSD, que puede ser utilizada como memoria adicional para el micro-controlador, a la manera de disco duro. La Figura 2 muestra una imagen de este dispositivo, una información detallada sobre el mismo se puede obtener en [9].

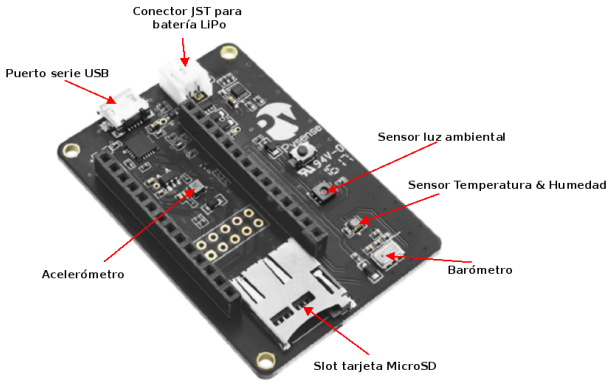

Figura 2: Esquema de una placa de extensión Pysense©.

Un AP WiFi: se necesitará un punto de acceso (AP) WiFi, para configurar una red inalámbrica 802.11 a la que se conectará el LoPy4©. A dicho

AP también se deberá conectar el PC, o bien a través de su enlace inalámbrico o de su interfaz Ethernet.

Una imagen de la conexión de un LoPy4©en una placa de expansión Pysense©se puede ver en la Figura 3.

Figura 3: Imagen de un LoPy4©conectado a un placa Pysense©.

2.2. Requisitos Software

Los requisitos software son los siguientes:

Git: Software de control de versión muy popular. Es utilizado para la obtención del código inicial de la práctica. Además se aconseja su utilización al alumno para llevar un control de los cambios que vaya incluyendo durante la realización de la práctica.

`Node-Red` **:** Es una herramienta de programación basada en flujo de datos. Con ella programaremos el bróker al que enviaremos los datos de los sensores y que proporcionará el panel de control al usuario.

mosquitto: Eclipse Mosquitto™ es un bróker de mensajes que implementa el protocolo MQTT v3.1 y v3.1.1, de código abierto. *Mosquito* es necesario para el envío de mensajes entre el LoPy4©y el bróker.

Atom: (*Opcional*) Este es un editor de texto de código abierto escrito en Node.js [3]. El uso de este editor de texto es opcional, aunque se recomienda su uso ya que existe un "***plug-in***"para él llamado `Pymakr` que permite la conexión directa, a través de una consola del propio editor, con un dispositivo LoPy4©conectado a una tarjeta de expansión Pysense©.

2.2.1. ¿Qué es `Git`?

`Git`[10] es un software de control de versiones que permite mantener un historial de los cambios realizados durante el desarrollo de un programa, o proyecto de programación. El software permite la recuperación de estados anteriores del proyecto, revocando cambios o recuperándolos. f Además el software tiene la posibilidad de crear *branches* o ramas. Un *branch* no es más que una versión nueva del proyecto, que aunque pertenece a él, se puede desarrollar de manera independiente a las demás *branches* o ramas. El software permite la unión de *branches* dando la opción de elegir, en caso de conflicto, los cambios que perduran en la rama final procedentes de las ramas a unir.

`Git` realiza el control de versiones en modo local, el usuario tiene una copia de todo el historial del proyecto en su máquina local. Pero la aplicación puede tener para cada proyecto uno o varios repositorios remotos donde guardar una copia del proyecto y su historia. De este modo la utilización de repositorios remotos es muy útil como copia de seguridad del proyecto desarrollado; o como mecanismo para la divulgación del código del proyecto entre otros usuario y desarrolladores.

En nuestro caso utilizaremos `Git` solo para la obtención, desde un repositorio remoto, del código necesario para el inicio de la práctica. Aunque se anima al alumno a que utilice la aplicación para llevar un historial de todos los cambios que vaya haciendo a lo largo de la práctica en su proyecto. Un buen manual de `Git` se puede encontrar en [1].

Instalación de `Git`

Para comprobar si `Git` está instalado en el sistema, se pude utilizar el comando:

```
$ git --version
```

que devuelve la versión de `Git` instalada. La última versión disponible se puede consultar en `https://git-scm.com/downloads`

Para la instalación de `Git` tan solo hay que seguir los pasos, según el SO a utilizar, indicados en la sección 1.5 (“Installing Git”) de [1].

2.2.2. ¿Qué es `Node-Red` ?

`Node-Red` [2] es una herramienta de programación basada en flujo de datos. Inicialmente desarrollada por IBM’s Emerging Technolgy Services, en la actualidad es un proyecto de la JS Foundation (`https://js.foundation/`).

Node-Red está consiste en un motor basado en Node.js y en una interfaz web con la que se accede al editor de flujos. Con este editor se crea la aplicación simplemente seleccionando nodos, o cajas negras, que realizan una determinada acción y conectando unos nodos con otros. Cada nodo toma el dato de entrada lo procesa y puede o no proporcionar un dato de salida. La conexión de un nodo con otro, lo que implementa es la comunicación del dato de salida de uno de los nodos con la entrada del otro nodo, implementado un flujo de datos entre nodos.

Instalación de Node-Red

Dado que Node-Red depende de Node.js es necesario que éste esté instalado previamente antes de instalar Node-Red . En https://nodered.org/docs/getting-started/installation se pude obtener una guía rápida de instalación tanto de Node-Red como de Node.js.

2.2.3. ¿Qué es Mosquitto ?

Eclipse Mosquitto™[4] es un bróker de mensajes de código abierto, el cual implementa el protocolo de envío de mensajes MQTT v3.1 y v3.1.1. Mosquitto es un proyecto de The Eclipse Foundation (https://iot.eclipse.org/).

MQTT es un protocolo de envío de mensajes muy ligero y adecuado para el caso de dispositivos de bajo consumo, mínima capacidad de procesamiento y redes de bajo ancho de banda, como el caso de las redes inalámbricas de sensores. MQTT está basado en un modelo de publicación/subscripción de mensajes.

Instalación de Mosquitto

Dependiendo del sistema operativo donde se instalará Mosquitto , el proceso de instalación varía, de ahí que se remite al lector a la URL: https://mosquitto.org/download/. En ella se indica que pasos ha de seguir el usuario según el sistema operativo en el que instalar la aplicación.

2.2.4. ¿Qué es Atom?

Atom [6] es un editor de texto de código abierto para MacOS, Linux y Windows, desarrollado por GitHub, y basado en Electron, entorno de desarrollo que permite la creación de aplicaciones de escritorio usando HTML, JavaScript, CSS y Node.js.

Una de las principales ventajas de Atom es que existen cientos de paquetes que añaden nuevas funcionalidades al editor de texto. Uno de ellos es el paquete `Pymakr` el cual añade una consola al editor Atom a través de la cual se puede conectar con un dispositivo LoPy4© conectado a una tarjeta de expansión Pysense©. Esta es la razón por la que se recomienda su uso. Más información sobre este paquete se puede encontrar en [7].

Instalación de `Atom`

La instalación de este editor de texto es muy sencilla. Tan solo hay que acceder a la URl `https://atom.io` y pulsar sobre el botón de "*Download*" para descargar según el SO de que se disponga el ejecutable de instalación o, como en el caso de Linux, el paquete a instalar.

En el caso de Linux el editor solo está disponible para arquitecturas de 64-bits. Una vez descargado el paquete a instalar se deberán de realizar las siguientes acciones según la distribución Linux que se posea:

- **Debian, Ubuntu o Linux Mint**:

```
$ sudo dpkg --install <downloaded_package>.deb
```

- **Red Hat, openSUSE, Fedora, CentOS**:

```
$ sudo rpm -i <downloaded_package>.rpm
```

Una vez instalado correctamente la ejecución se realiza mediante el comando en consola:

```
$ atom
```

Instalación del paquete de Atom `Pymark`

Para instalar el plugin Pymark en Atom tan solo hay que seguir las instrucciones siguientes, una vez instalado correctamente el editor Atom:

1. Elige la opción de la barra de menús `Edit >Preference >Install`.
2. Una vez en el menú **Install Packages**, buscar el paquete `Pymakr`.
3. Seleccionar el paquete oficial **Pymakr** y pulsar en el botón de instalación.

Más información sobre la instalación de este plug-in y su configuración puede ser encontrada en `https://docs.pycom.io/chapter/pymakr/installation/atom.html`.

3. Descripción de la actividad.

La actividad consistirá básicamente en la programación de una aplicación que permita la lectura de los sensores de un LoPy4©, su envío mediante MQTT a un bróker, el cual deberá procesar dichos datos para representarlos en un panel de control ("*dashboard*"). Además en el propio panel de control habrá elementos con los que el usuario podrá: modificar el color RGB del LED cuando este se encienda, el modo de encendido del LED, que será automático o manual; y en el caso de que el modo de encendido del LED sea manual, existirá un botón para encender y apagar dicho LED.

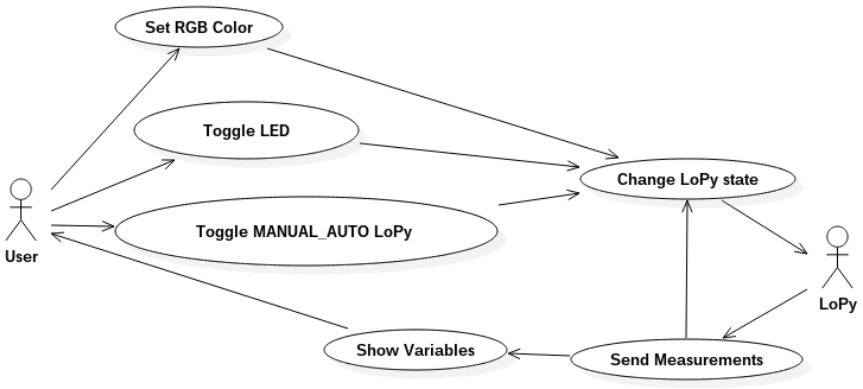

Figura 4: Casos de uso del sistema.

El diagrama de casos de uso del sistema se muestra en la Figura 4. Como se observa, el usuario podrá modificar el estado del LED (`Toggle LED`), que podrá estar encendido o apagado, y el estado del LoPy4© (`Toggle MANUAL_AUTO LoPY`) que podra ser `MANUAL/AUTO`. Además se podrá seleccionar el color RGB del LED (`Set RGB Color`). Estas tres acciones cambiarán el estado del LoPy4©, y dicho cambio se comunicará al LoPy4©. Por otro lado el LoPy4©realizará medidas de sus sensores y enviará dichas medidas al broker (`Send Measurements`). Provocando la representación de las mismas en el panel de control (`Show Variables`), para que puedan ser visualizadas por el usuario.

Así pues el alumno deberá de realizar básicamente tres acciones:

1. Configuración de una red WiFi con seguridad WPA Personal. La cual será utilizada para que se conecte el dispositivo LoPy4© al bróker situado en el PC.

2. Programación del LoPy4©.

3. Programación del bróker.

A continuación se detallan cada una de estas acciones.

3.1. Configuración de una WiFi

Lo primero que deberá de configurar el alumno es una red WiFi con configuración WPA2-Personal, utilizando el AP (*Access Point*) que se le proporcionara. Esta configuración se realizará para dar cobertura inalámbrica al dispositivo LoPy4©.

El PC donde situaremos el bróker deberá estar conectado al AP proporcionado a través de un enlace Ethernet o de la propia red WiFi creada.

Se aconseja que el AP proporcione, si fuera posible, direcciones IP a todos los dispositivos de la interfaz WiFi. De esta manera simplificaremos la configuración IP de los dispositivos LoPy4©.

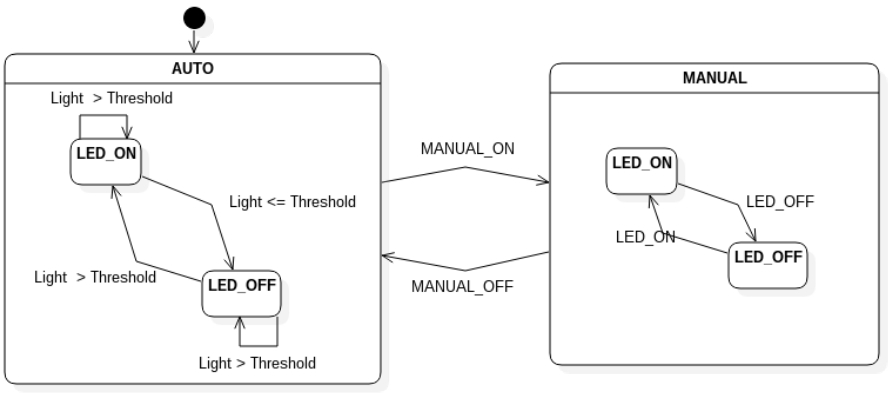

Figura 5: Diagrama de estados del LoPy4©.

3.2. Programación del LoPy4©

El LoPy4© realizará un muestreo constante de la temperatura y la luminosidad de la placa de expansión Pysense©. Dichos valores son agrupados

en un único mensaje JSON y enviados, utilizando MQTT, al bróker cada un periodo de tiempo determinado por una constante de programación.

Como se observa en la Figura 5, fundamentalmente el LoPy4© puede estar en estado `AUTO` o en estado `MANUAL`. Cada uno de estos dos estados tiene dos sub-estados: `LED_ON` estado en el que el LoPy4© enciende el LED RGB; y el `LED_OFF` estado en el que el LED RGB del LoPy4© estará apagado. El paso del estado `AUTO` al estado `MANUAL` se realizará en función de la interacción del usuario con el panel de control. Dónde habrá un selector `AUTO/MANUAL` para elegir en que estados estará el LoPy4©.

En el caso del estado `AUTO`, pasaremos al sub-estado `LED_ON` si el nivel de luminosidad es igual o inferior a un determinado umbral (definido como constante de programación). Y estaremos en sub-estado `LED_OFF` cuando dicho nivel de luminosidad sea superior al umbral indicado.

Sin embargo, en el estado `MANUAL`, la permanencia en el sub-estado `LED_ON` o `LED_OFF`, dependerá de la interacción del usuario con el panel de control, donde deberá existir un botón de encendido/apagado del LED.

El color de encendido del LED RGB inicialmente se puede establecer aun valor determinado, p.e. `0xff0000`: rojo intenso, pero podrá ser elegido por el usuario mediante un selector RGB de color en el panel de control. Una descripción del panel de control se verán en la sección 3.3.1.

Como ya se ha comentado el LoPy4© debe de estar continuamente leyendo sus sensores y enviado dichos datos mediante MQTT al bróker. La Figura 6 muestra un diagrama de secuencia del los mensajes generados por el LoPy4©.

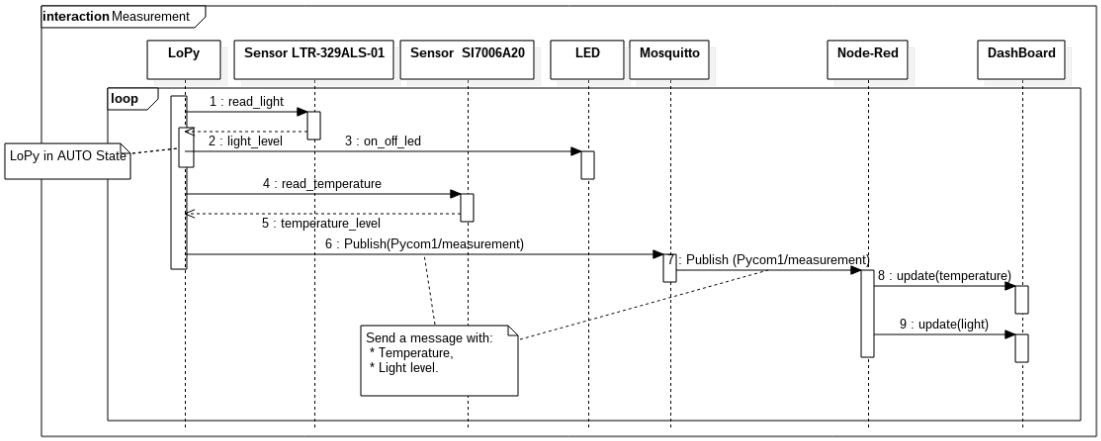

Figura 6: Diagrama de secuencia para la lectura de sensores y la comunicación de dichos datos.

3.3. Programación del bróker

El bróker deberá de realizar dos funciones fundamentalmente:

1. Obtener las medidas enviadas mediante MQTT por el LoPy4©y actualizar con ellas el panel de control.

2. Obtener del panel de control los parámetros del funcionamiento, introducidos por el usuario, y enviárselos al LoPy4©mediante MQTT.

La programación del bróker se realizará utilizando `Node-Red` lo cual nos permitirá también la implementación directa de un panel de control. Además de poder interactuar de manera muy simple con el servidor de `Mosquitto` .

3.3.1. Panel de control ("*dashboard*").

El Panel de control o, como se conoce comúnmente por su término en inglés, "*dashboard*", es una interfaz de usuario en la que se pueden representar estadísticas sobre los datos obtenidos por los sensores/actuadores de los dispositivos IoT, así como elementos de interacción con el usuario que pueden servir para: enviar acciones a realizar por los actuadores de nuestra red de dispositivos, realizar consultas a la BB.DD. donde tenemos almacenados los datos medidos, etc.

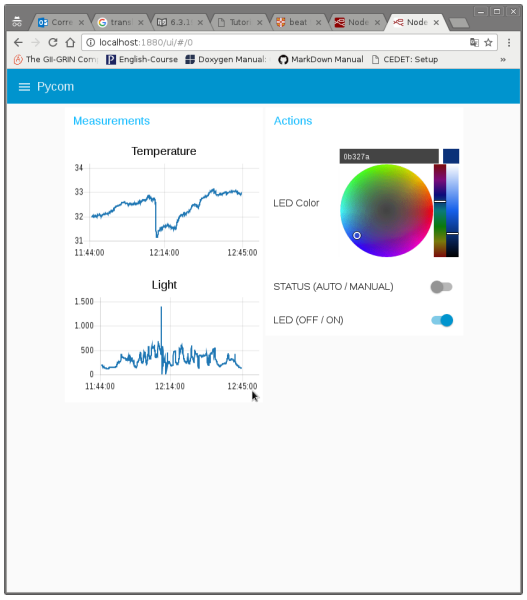

Figura 7: Ejemplo de Panel de Control.

En nuestro caso el panel de control podría tener un aspecto similar al mostrado en la Figura 7. En ella se puede ver como hay dos grupos de elementos bien diferenciados: por un lado está el grupo ***Measurement*** que muestra las medidas obtenidas mediante MQTT del LoPy4©, a saber, ***Temperature*** y ***Light***. Y el otro grupo será el de ***Actions*** que agrupa a todos los elementos que utilizará el usuario para interactuar directamente con el LoPy4©, éstos son: ***LED Color*** que es una paleta RGB para la selección del color del LED RGB, ***STATUS (AUTO/MANUAL)*** interruptor que determina el estado del LoPy4©(ver diagrama de estados en Figura 5), y el interruptor ***LED (OFF/ON)*** que en el caso de que el LoPy4©esté en estado `MANUAL` determinará si el LED RGB está encendio o no.

Un diagrama de secuencia de los mensajes generados por el panel de control, y su relación con el reto de componentes del sistema se puede ver en la Figura 8. En ella se puede ver como por cada acción del usuario sobre uno de los elementos del grupo ***Actions*** se genera el envío de un mensaje distinto a través de MQTT al LoPy4©.

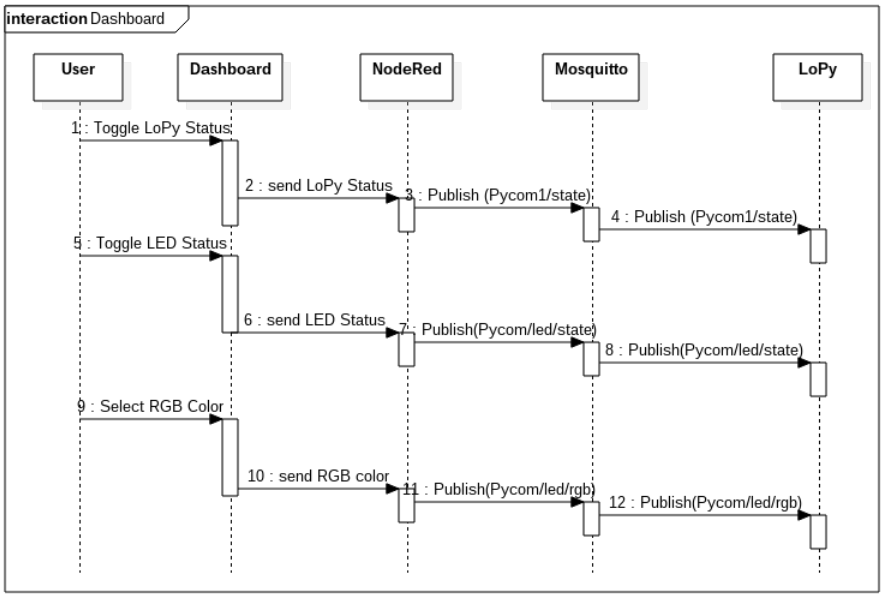

Figura 8: Diagrama de secuencia para las acciones sobre el panel de control.

3.4. Obtención del código fuente de la aplicación.

La aplicación está formada por parte de código que deberá desarrollar el alumno y parte de código que le es proporcionado. El código proporcionado

aunque necesario para que la aplicación final funcione, no es de interés para el desarrollo de los objetivo y competencias tratadas en esta práctica.

El código de partida para la actividad se obtendrá del repositorio git `https://gitraap.i3a.info/francisco.delicado/LoPy-NodeRed-student.git`

```
$ git clone https://gitraap.i3a.info/francisco.delicado
   /LoPy-NodeRed-student.git
```

El comando anterior creará un directorio `./Lopy-NodeRed-student/` con el siguiente contenido:

- `README.md`: fichero que describe el contenido del repositorio asociado a esta práctica. El documento está escrito en **Markdown**, lenguaje de marcado interpretado por aplicaciones como GitLab o GitHub para la presentación de documentos en su interfaz web.
- `pymakr.conf`: fichero de configuración del *plug-in* `Pymakr` de `Atom`.
- `doc`: directorio que contiene el presente documento.
- `code/Lopy`: directorio que contiene los ficheros necesarios para la implementación de la aplicación que deberá ser ejecutada en el microcontrolador LoPy4©. En concreto, el contenido del directorio será el siguiente:
  - `code/Lopy/lib`: es un directorio que contendrá los módulos en MicroPython necesarios para la implementación de la aplicación, y puede que sea necesario importar en el código de nuestro programa.
  - `code/Lopy/boot.py`: programa en MicroPython que lanza el microcontrolador LoPy4©cuando arranca. Su contenido es el siguiente:

```
1 from machine import UART
2 import machine
3 import os
4
5 uart = UART(0, baudrate=115200)
6 os.dupterm(uart)
7
8 machine.main('main.py')
```

Las líneas 5 y 6 configuran la interfaz USB serial para conectarnos a ella mediante una consola, y la última línea indica cual es

el programa principal que se ejecutará en el micro-controlador. En este caso el programa `main.py`. Que es el que deberá implementar el alumno.

- `code/Lopy/main.py`: fichero sin contenido alguno, y que deberá de contener el código desarrollado por el alumno.

- `code/Node-Red`: es el directorio, vacío, donde se guardarán los ficheros `JSON` que describen el código de cada uno de los flujos de datos desarrollados para `Node-Red` .

3.5. ¿Qué debe hacer el alumnos?

Una vez adquirido el código según lo indicado en la Sección 3.4, el alumno deberá realizar las siguientes acciones.

3.5.1. Abrir el proyecto en Atom

Lo primero será abrir el proyecto obtenido en el editor Atom. Se recuerda que el uso del editor Atom es opcional, aunque recomendable ya que existe un plug-in llamado Pymakr que permite la interacción directa con el LoPy4©a través de su puerto USB.

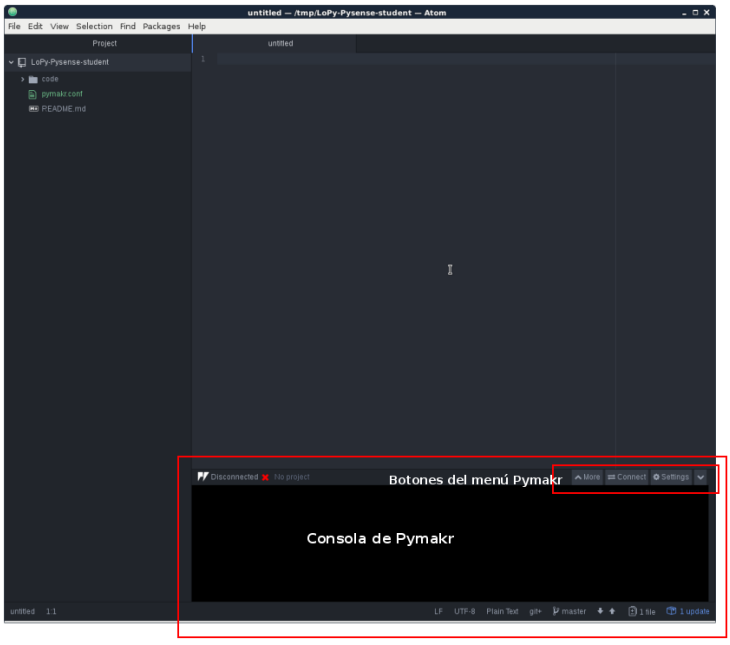

Figura 9: Editor Atom con consola Pymakr antes de conectarnos a un LoPy4©.

Para abrir el proyecto en Atom deberemos seleccionar la opción del menú "`File >Add Project Folder.`º pulsar (`Ctrl+Shift+A`), seleccionando en directorio creado tras la adquisición del código.

3.5.2. Conectar LoPy4© a la consola Pymakr

Lo primero que abrá que hacer es conectar el LoPy4© al PC mediante su puerto USB. Cuando el LoPy4© esté conectado y activo empezará a parpadear en color azul el LED RGB del micro-controlador. Ahora es el momento de conectarnos al LoPy4© pulsando el boton "`Connect`" de la consola del Pymakr (ver Figura 9).

Cuando la conexión ha sido realizada con éxito entre el PC y el LoPy4© se obtiene la siguiente salida:

```
Connecting on /dev/ttyACM0...
>>>
```

Además tras la conexión con el LoPy4© la consola Pymakr habrá cambiado su apariencia como se muestra en la Figura 10

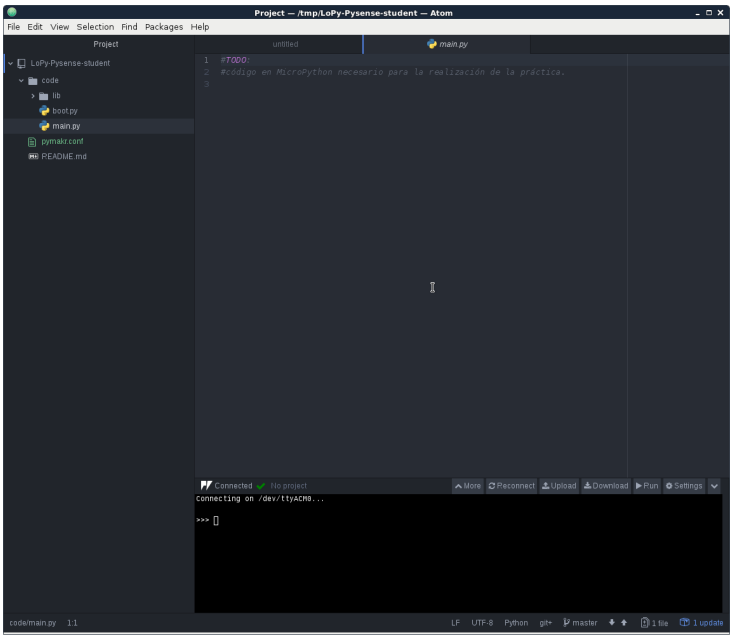

Figura 10: Editor Atom con consola Pymakr después de conectarnos a un LoPy4©.

3.5.3. Configurar Pymakr

Este paso de configuración tiene como objetivo el determinar qué ficheros y de qué tipo se sincronizarán con el LoPy4©. Es decir, le indicamos que ficheros vamos a cargar en el LoPy4©cuando pulsemos el botón "`Upload`" del menú de Pymakr. La configuración se realiza seleccionando en el menú de Pymakr la opción "`Settings >Proyect settings`". Esto abrirá un fichero llamado "pymakr.conf" que si no existe en el directorio del proyecto es creado y que tendrá el siguiente aspecto:

```
1 {
2     "address": "/dev/ttyACM0",
3     "username": "micro",
4     "password": "python",
5     "sync_folder": "LoPy",
6     "open_on_start": true,
7     "sync_file_types": "py,txt,log,json,xml,crt",
8     "ctrl_c_on_connect": false
9 }
```

En este fichero cabe destacar dos líneas:

- **línea 5: "`sync_folder`".** Indica el directorio dentro del proyecto que se va a sincronizar con el LoPy4©. Es decir, cada vez que pulsemos el botón de "`Upload`", el contenido de dicho directorio será volcado en su totalidad al LoPy4©. En nuestro caso este parámetro debería tener el siguiente valor:

```
5     "sync_folder": "LoPy",
```

- **línea 7:** "`sync_file_types`"**.** Indica la extensión de los ficheros que podrán ser sincronizados con el LoPy4©, de los incluidos en el "`sync_folder`". Solo se envían al micro-controlador los ficheros que acaben con alguna de las extensiones indicadas en la lista. En nuestro caso nos deberemos de asegurar que está incluida la extensión "`py`" de los ficheros de MicroPython.

```
7     "sync_file_types": "py,txt,log,json,xml,crt",
```

3.5.4. Inicialización del LoPy4©

Dado que se desconoce el contenido del LoPy4©, lo primero que deberemos hacer es **inicializar** el contenido del mismo borrando cualquier programa que pudiera tener. Esto se consigue inicializando su **memoria**

flash. Para ello en la consola de Pymakr deberemos incluir lo siguientes comandos:

```
>>> import os
>>> os.mkfs("/flash")
```

Esto habrá borrado el contenido de la memoria flash del LoPy4©, tras lo cual lo reinicializaremos con las instrucciones siguientes:

```
>>> import machine
>>> machine.reset()
```

Una vez inicializado el micro-controlador habrá que cargar en él el código inicial del proyecto. El cual fue adquirido por el alumno. Si se configuró correctamente la consola Pymakr (sección 3.5.3), tan solo hay que pulsar el botón "`Upload`" del menú de Pymakr.

3.5.5. Desarrollo del programa para el LoPy4©

El desarrollo del programa constará de dos acciones por parte del alumno.

1. Por un lado la **edición del fichero** `main.py` del proyecto, para programar utilizando MicroPython las acciones que el micro-controlador debe realizar.

2. El **subir** las modificaciones que se realicen en el fichero `main.py` **al LoPy4©**, para ver si el comportamiento del programa es el deseado. La subida del fichero `main.py` al micro-controlador se puede realizar de dos formas.

   - Mediante el **botón** "`Upload`" de Pymakr. Esto provoca que Pymakr compare el contenido del LoPy4©con el del directorio a sincronizar, y copie en la flash del micro-controlador aquellos ficheros del directorio que han modificado su contenido desde el último "`Upload`" realizado. Es decir, este procedimiento supone una modificación del contenido de la flash, seguido de un reinicio del micro-controlador. Tras el cual se comenzará a ejecutar el programa que estamos editanto.
   - Mediante el **botón** "`Run`" de Pymark. Este mecanismo lo que hace es la ejecución del fichero que estamos editando directamente en el LoPy4©. Pero sin su volcado en flash. Es decir. El contenido del fichero que estamos editando no se guarda en la flash

del LoPy4©, pero sí que se ejecutan sus instrucciones en él. Esta última es la manera más rápida de probar un cambio en el fichero `main.py` en el LoPy4©, pero hay que tener en cuenta que dicho cambio no se guardará en la flash del micro-controlador, por lo que al ser reiniciado no ejecutará esta última versión del programa, sino la última que tiene en flash.

Así pues, nos debemos de asegurar de realizar un "`Upload`" cuando queramos que la versión del fichero `main.py` esté permanentemente en el LoPy4©.

3.5.6. Desarrollo del flujo de datos para `Node-Red`

Paralelamente al desarrollo del código para el LoPy4©, el alumno deberá desarrollar la programación para los flujos de datos generados por LoPy4©en `Node-Red` . Para ello lo primero que se deberá hacer es, inicializar `Node-Red` , lo cual se realiza con el siguiente comando (desde un terminal en Linux):

```
$ node-red
19 Dec 16:02:27 - [info]

Welcome to Node-RED
===================

19 Dec 16:02:27 - [info] Node-RED version: v0.17.5
19 Dec 16:02:27 - [info] Node.js  version: v8.9.1
19 Dec 16:02:27 - [info] Linux 3.16.0-4-amd64 x64 LE
19 Dec 16:02:27 - [info] Loading palette nodes
19 Dec 16:02:28 - [info] Dashboard version 2.6.2
   started at /ui
19 Dec 16:02:29 - [warn]
   ------------------------------------------------------

19 Dec 16:02:29 - [warn] [rpi-gpio] Info : Ignoring
   Raspberry Pi specific node
19 Dec 16:02:29 - [warn]
   ------------------------------------------------------

19 Dec 16:02:29 - [info] Settings file  : /home/franman
   /.node-red/settings.js
19 Dec 16:02:29 - [info] User directory : /home/franman
   /.node-red
```

```
19 Dec 16:02:29 - [info] Flows file     : /home/franman
   /.node-red/flows_pelijes-laptop.json
19 Dec 16:02:29 - [info] Server now running at http
   ://127.0.0.1:1880/
19 Dec 16:02:29 - [info] Starting flows
19 Dec 16:02:29 - [info] Started flows
19 Dec 16:02:29 - [info] [mqtt-broker:6c638ac5.c73634]
   Connected to broker: mqtt://localhost:1883
19 Dec 16:02:29 - [info] [mqtt-broker:30f8adcc.9c28f2]
   Connected to broker: mqtt://localhost:1883
```

Como se puede ver las dos últimas líneas nos comunican que el `Node-Red` se ha conectado a un servidor `Mosquitto` , que es accesible en la URL `mqtt://localhos:1883`. Ahora solo nos queda acceder a la GUI de `Node-Red` : para ello debemos acceder, mediante un "*navegador web*.ª la URL `http://<IP_node_red>:1880`, donde `<IP_node_red>` es la dirección IP de la máquina donde hemos arrancado `Node-Red` , en nuestro caso `localhost`.

Una vez hemos accedido a la GUI de `Node-Red` , solo hay que crear mediante la utilización de los componentes existentes en `Node-Red` , y la interconexión entre ellos, el/los flujo/s de datos necesarios para realizar las acciones del bróker. Finalizado el diseño y programación de cada flujo de datos se pulsará sobre el botón "*Deploy*" situado en la parte superior derecha de la GUI de `Node-Red` para que los cambios introducidos tengan efecto.

3.6. Problemas con la programación del LoPy4©

En esta sección se dan solución a algunos de los problemas más comunes que surgen cuando se está trabajando con un LoPy4©.

3.6.1. Problema de volcado del código al LoPy4©

Hay veces que al intentar volcar el código al LoPy4©se obtiene el siguente mensaje de error:

```
An error occurred: Not enough memory available on the
   board.
Upload failed. Please reboot your device manually.
```

SOLUCIÓN: esta pasa por,

1. Reiniciar el LoPy4©pulsando su botón de `RESET` (ver Figura 1).

2. Volver a intentar el volcado del código una vez reiniciado el micro-controlador.

Si el problema persistiera tras el reinicio habrá que:

1. Reiniciar el LoPy4© pulsando su botón de `RESET` (ver Figura 1).

2. Inicializar la flash del micro-controlador, según se indica en la sección 3.5.4, una vez reiniciado el LoPy4©.

3. Tras lo cual volveremos a intentar el volcado del código.

3.6.2. El LoPy4© no responde

Si el micro-controlador entra en un estado de no respuesta, es decir, en la consola Pymakr no se detecta ninguna actividad, y no tenemos activo el *promp* "`>>>`".

SOLUCIÓN:

1. Pulsar las teclas "`Ctrl+C`", para forzar la salida del programa que se esté ejecutando en el micro-controlador. Esto nos tendría que devolver el *promp* "`>>>`". Si no fuera así pasaríamos al paso 2.

2. Realizar un reinicio forzado del dispositivo pulsando el botón de `RESET` (Figura 1). Tras su reinicialización sería conveniente realizar un borrado de la flash del LoPy4© según se indica en la sección 3.5.4.

3.7. ¿Cómo entregar la actividad?

Una vez finalizada la actividad, el alumno deberá de **comprimir todo el directorio** `code` que contendrá los fichero `boot.py`, `main.py`, el directorio `./LoPy/` y el directorio `./Node-Red/`. Y **subirlo a la plataforma moodle de la asignatura** , utilizando la tarea de entrega correspondiente.

Referencias

[1] Scott Chancon and Ben Straub. Pro git. `https://git-scm.com/book/en/v2`. Accedido: 09-11-2017.

[2] JS Foundation. Node-red. `https://nodered.org`. Accedido 15-12-2017.

[3] Node.js Foundation. Node.js. https://nodejs.org/es/. Accedido 14-12-2017.

[4] The Eclipse Foundation. Mosquitto. https://mosquitto.org. Accedido 15-12-2017.

[5] Damien George. Micropython. https://micropython.org/. Accedido 14-12-2017.

[6] GitHub Inc. Atom. a hackable text editor. https://atom.io/. Accedido 14-12-2017.

[7] GitHub Inc. Pymakr. https://atom.io/packages/pymakr. Accedido 14-12-2014.

[8] Pycom LTD. Lopy-4. https://pycom.io/hardware/lopy4-specs. Accedido 14-12-2017.

[9] Pycom LTD. Pysense. https://pycom.io/hardware/pysense-specs/. Accedido 14-12-2017.

[10] Linus Torvalds. Git. https://git-scm.com/. Accedido: 09-11-2017.