Actividad 3.1: Lectura de sensores y encendido LEDs en un LoPy4©

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 realizar lecturas de los sensores incorporados en un "*mote*", realizar un pequeño análisis de dichos datos, y realizar una acción sobre algún dispositivo conectado al propio "*mote*".

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. Ser capaz de procesar en el propio "*mote*" los datos obtenidos por un sensor y actuar en consecuencia.
3. Escribir datos en un determinado dispositivo conectado al "*mote*".

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 [3]. Se puede obtener información sobre este dispositivo en [6], y la Figura 1 muestra un esquema del dispositivo.

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

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

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 [7].

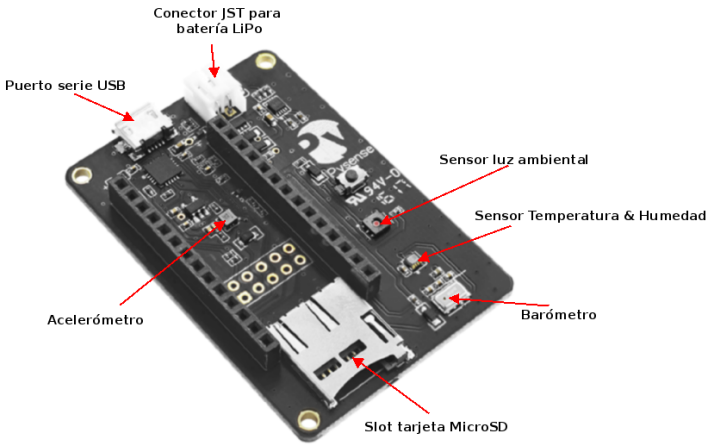

Figura 2: Esquema de una placa de extensión Pysense©.

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 mínimos y se limitan a las siguientes aplicaciones:

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.

Atom: (*Opcional*) Este es un editor de texto de código abierto escrito en Node.js [2]. 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`[8] 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 apli-

cación 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 `Atom`?

Atom [4] 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 [5].

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 tarea a realizar en esta actividad consiste en la realización de un programa en MicroPython para un micro-controlador LoPy4©conectado a una placa Pysense©que deberá de ejecutar la siguientes acciones:

- Cada segundo deberemos leer el valor de la temperatura y luminosidad exterior proporcionada por los sensores de la placa Pysense©. Dichos valores deberán ser mostrados en la consola con la que estaremos conectados al LoPy4© (se recomienda utilizar la consola del "*plug-in*"Pymakr de Atom).

- En el caso de que la luminosidad esté por debajo de un valor de 50 unidades, se deberá encender el LED RGB que posee el LoPy4© con un valor RGB igual a `#ff0000` (valor correspondiente al rojo intenso).

3.1. 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-Pysense-student.git`

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

El comando anterior creará un directorio `Lopy-Pysense-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.

- `doc`: directorio que contiene el presente documento.

- `code`: 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/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/boot.py`: programa en MicroPython que lanza el micro-controlador 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/main.py`: fichero sin contenido alguno, y que deberá de contener el código desarrollado por el alumno.

3.2. ¿Qué debe hacer el alumnos?

Una vez adquirido el código según lo indicado en la Sección 3.1, el alumno deberá realizar las siguientes acciones.

3.2.1. 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.

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.

3.2.2. 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 4).

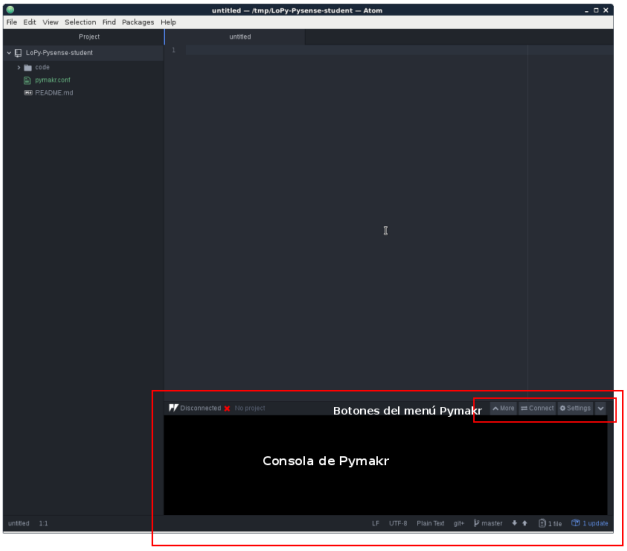

Figura 4: Editor Atom con consola Pymakr antes de conectarnos a un LoPy4©.

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 5

3.2.3. 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:

```
{
    "address": "/dev/ttyACM0",
    "username": "micro",
    "password": "python",
    "sync_folder": "LoPy",
    "open_on_start": true,
    "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",
```

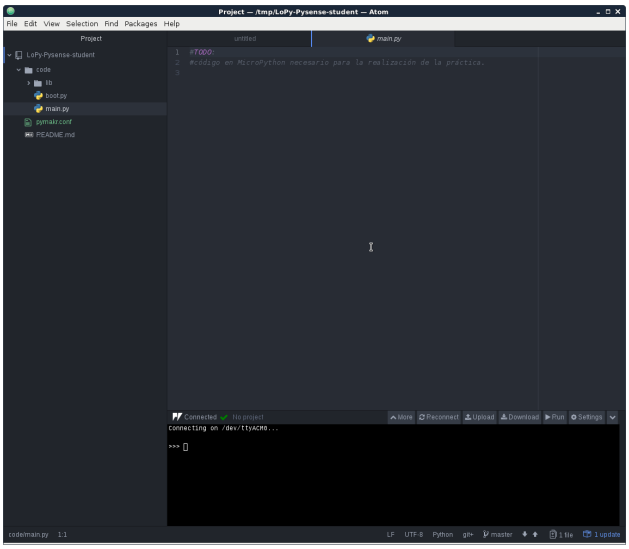

Figura 5: Editor Atom con consola Pymakr después de conectarnos a un LoPy4©.

3.2.4. 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 3.2.3), tan solo hay que pulsar el botón "`Upload`" del menú de Pymakr.

3.2.5. Desarrollo del programa

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©.

3.3. 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©.

3.3.1. 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 3.2.4, una vez reiniciado el LoPy4©.
3. Tras lo cual volveremos a intentar el volcado del código.

3.3.2. 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 3.2.4.

3.4. ¿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` y el directorio `lib/`, 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] Node.js Foundation. Node.js. `https://nodejs.org/es/`. Accedido 14-12-2017.

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

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

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

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

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

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