Escuela Técnica Superior de Ingeniería de Telecomunicación

Universidad Politécnica de Cartagena

Desarrollo de un teclado accesible con MicroPython

Autor: Pablo Martínez Bernal

Director: Jose Alfonso

Máster Universitario en Ingeniería de Telecomunicación

Curso 2022-23

Universidad Politécnica de Cartagena

ÍNDICE GENERAL

SECCIÓN 1

IMPLEMENTACIÓN DEL FIRMWARE

1.1. Instalar MicroPython

Python es un lenguaje interpretado, por lo que existen distintas implementaciones del mismo, por nombrar algunas:

- Cython (C)
- IronPython (.NET)
- Jython (Java)
- PyPy (Python)

Para este projecto voy a utilizar MicroPython, que también está escrito en C pero su objetivo es el de minimizar el uso de recursos (mayormente RAM) para que sea viable emplearlo en microcontroladores.

1.1.1. Preparar el compilador

Tras clonar el repositorio de MicroPython (`git clone https://github.com/miropython/micropython`), hacemos `make -C mpy-cross` para compilar el compilador cruzado de MicroPython que nos permitirá compilar nuestro código para ser ejecutado en diferentes arquitecturas

1.1.2. Compilar para Linux (Opcional)

Si queremos usar MicroPython en nuestro ordenador para hacer pruebas, en vez de CPython(que es la versión más común), usaremos el compilador que acabamos de construir para compilar el código fuente del intérprete y usarlo en nuestra máquina `cd ports/unix && make submodules && make` Y ahora podemos ejecutarlo con `cd build-standard`

```
$ ./micropython
MicroPython 13dceaa4e on 2022-08-24; linux [GCC 12.2.0] version
Use Ctrl-D to exit, Ctrl-E for paste mode
```

1.1.3. Compilar para RP2040

Primero instalamos un compilador necesario para la arquitectura del procesador, en mi caso (Arch Linux), el comando es `sudo pacman -S arm-none-eabi-gcc`.
Ahora,añadimos TinyUSB a la configuración del compilador, para que añada dicha libreria siguiendo la información de noobee [1]

</> ports/rp2/CMakeLists.txt

```
    | set(MICROPY_SOURCE_PORT
(+) |   modusb_hid.c
    |   fatfs_port.c
          .
          .
          .
    | set(MICROPY_SOURCE_QSTR
(+) |   ${PROJECT_SOURCE_DIR}/modusb_hid.c
    |   ${MICROPY_SOURCE_PY}
```

Y añadimos/editamos los archivos necesarios (ver commits en su repositorio de GitHub). Por último, de forma análoga al paso anterior, hacemos `cd ../../rp2 && make submodules && make`

SECCIÓN 2

REFERENCIAS

1. noobee, *Add TinyUSB to RP2040*, `https://github.com/noobee/micropython/tree/usb-hid`.