

## Capítulo 4

# Integracion Mbot en Kibotics

En este capítulo se aborda el proceso de integración que va a permitir programar al robot físico Mbot sin necesidad de una instalación previa ni descargas adicionales.

### 4.1. Características del Mbot

Mbot es un robot fácil de montar y con una estructura robusta, orientado para empezar el aprendizaje de la robótica y la programación desde la educación primaria. Está diseñado por la empresa MakeBlock, la cual dispone de una gran variedad de recursos, robots y kits de robotica.

El robot pesa 400 gramos y sus dimensiones son de 17x13x9 cm. Respecto a las especificaciones técnicas, posee una placa llamada “mCore” que está basada en ArduinoUno, dispone de una microcontroladora ATmega238, cuatro puertos Rj25, un interruptor de encendido, dos leds RGB, un botón, zumbador, un sensor de infrarrojos y otro de luminosidad. Dispone de una batería de litio de 3,7V, pero también funciona con cuatro pilas de tipo AA. En cuanto sus módulos externos, presenta dos motes, ultrasonido y un seguidor en línea, aunque también pueden utilizarse otras conexiones adicionales. Se comunica por vía puerto serie o por Bluetooth 4.0 (Robótica Educativa con Mbot, 2020).

MackeBlock distribuye un software gratuito, llamador mBlock (Figura 4.2) que permite programar a este robot. Este software puede ser usado vía web, aunque para utilizarlo de esta manera es necesario que el usuario instale un driver, también puede utilizarse de forma local, descargándolo en el ordenador. Con él se puede programar al Mbot utilizando los lenguajes de programación Scratch o Python.

### 4.2. Diseño

En la Figura 4.3 se muestra el diseño seguido para la integración del Mbot en la plataforma de Kibotics.

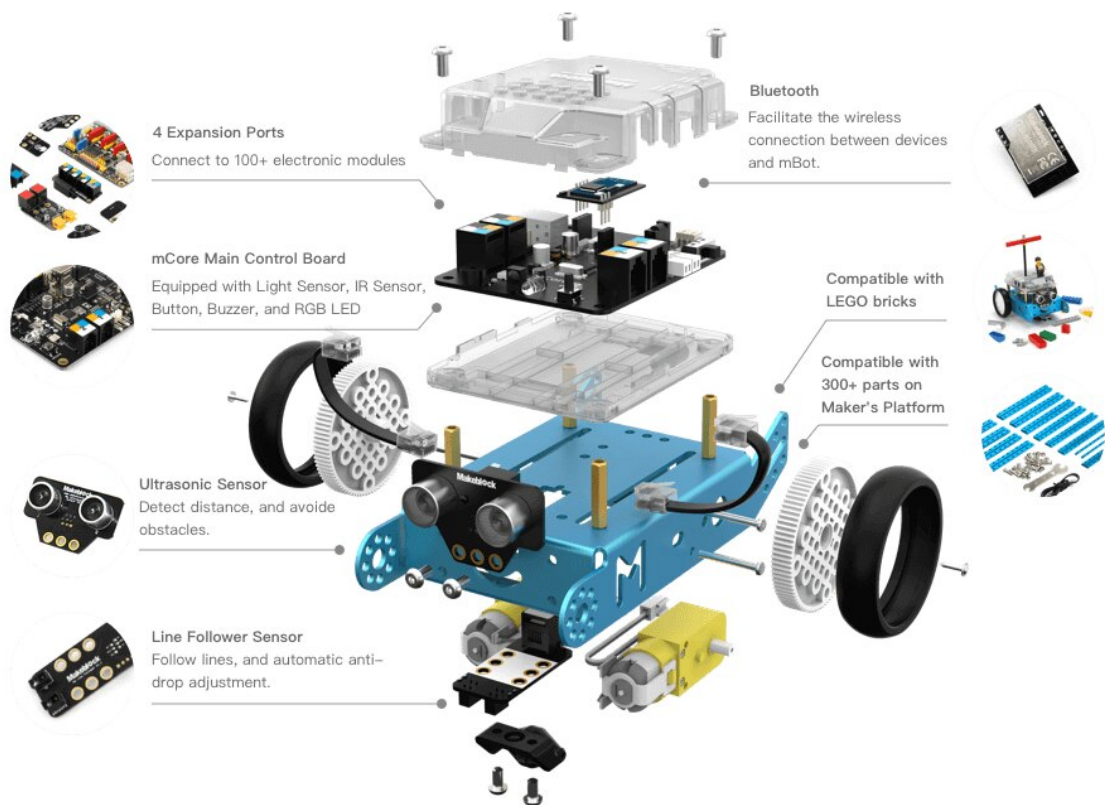


Figura 4.1: Partes robot Mbot

En las secciones siguiente se detallarán cuales son las interacciones necesarias entre el cliente y el servidor para permitir la carga del programa al Mbot

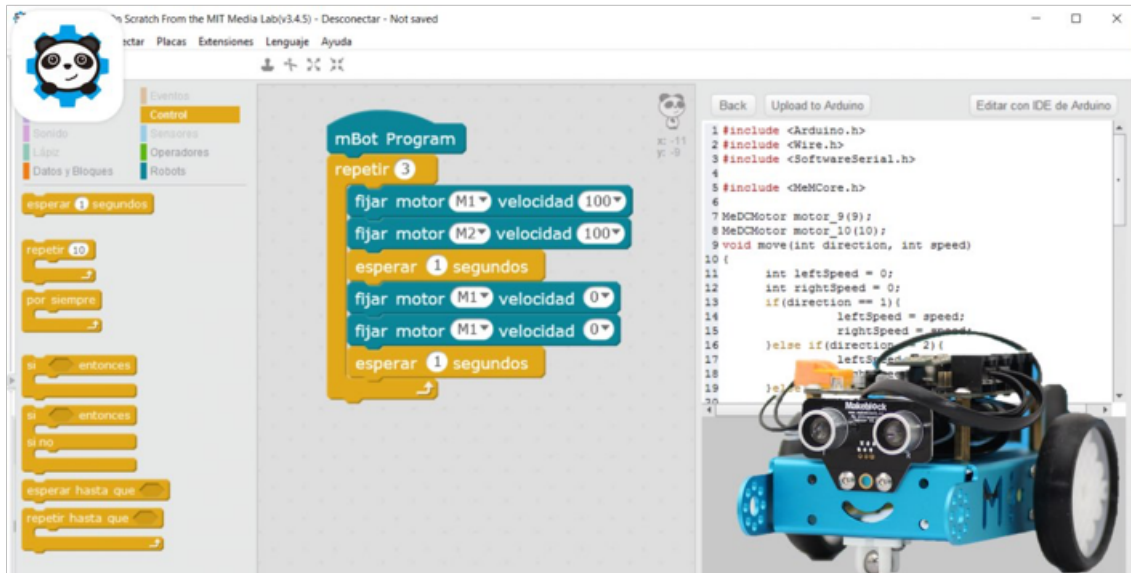


Figura 4.2: Software mBlock

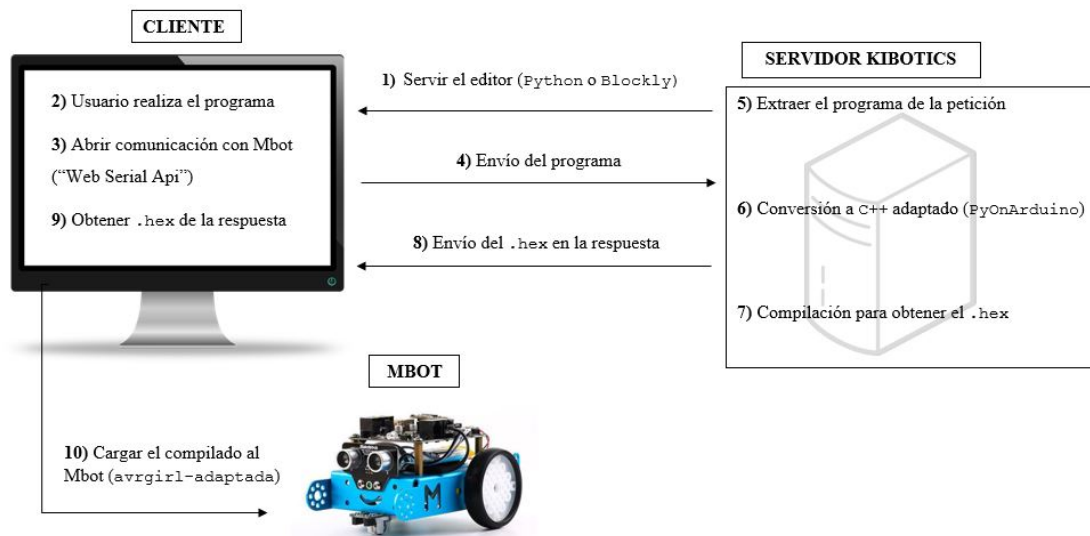


Figura 4.3: Infraestructura seguida para la integración del Mbot

### 4.3. Lado Cliente

Desde la parrilla principal de Kibotics se puede entrar a la unidad dedicada al Mbot, pudiendo elegir qué lenguaje de programación usar (Python o Blockly). Dentro de dicha unidad encontra-

mos un apartado dedicado a proporcionar las instrucciones necesarias para realizar un programa y enviárselo al Mbot.

Gracias al diseño utilizado, no es necesario que el usuario instale nada en su ordenador ni en el robot para permitir su uso, por lo que el proceso de programación del robot se limitará a que una vez que el usuario ha realizado el programa se envíe al robot directamente.

### 4.3.1. Editor

En la unidad del robot se proporciona un editor que le permitirá al usuario escribir el programa que será enviado al robot. Según la unidad elegida el editor estará orientado a ese lenguaje.

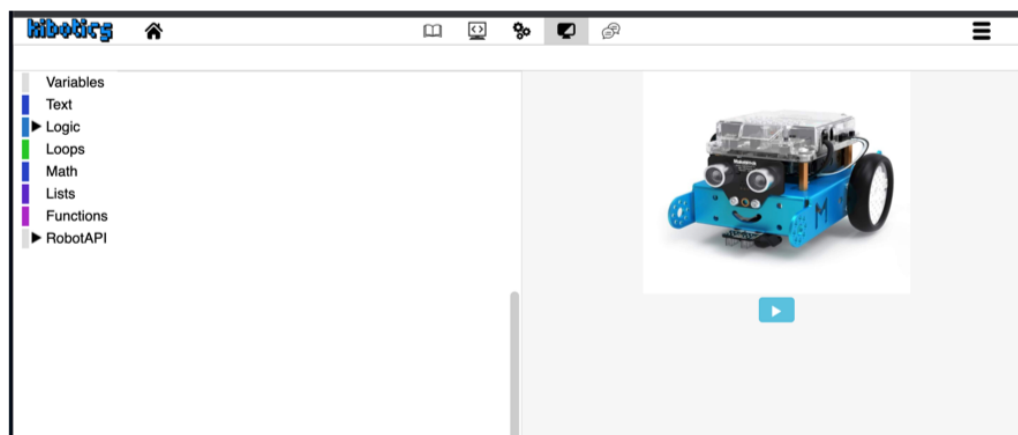


Figura 4.4: Editor Blockly para el Mbot

### 4.3.2. Conexión con el Mbot (USB)

Una vez que el usuario ha realizado el programa deseado y ha conectado el Mbot por USB a su ordenador, al pulsar el botón azul que se muestra en la Figura 4.4, se iniciará el proceso de envío.

En primer lugar, se abrirá una ventana que muestra los puertos del ordenador disponibles. Se debe seleccionar en el que se encuentra conectado el robot, que aparece nombrado como USB2.0Serial. Esta acción permitirá abrir una comunicación entre el navegador y el robot (Fragmento 4.1). Esto es posible gracias a “Web Serial Api”, que proporciona una Api al navegador capaz de interactuar con el puerto serial de la computadora.

```
document.getElementById('uploadMbot').addEventListener('click',  
  clickConnect);
```

```

});

async function clickConnect() {
    progress.style.width = '0%';
    alertError.style.visibility = 'hidden';
    //— Si el puerto serie estaba ya abierto, cerrarlo
    if (portPrev) {
        await disconnect();
    }
    await connect();
    activeProgress.style.visibility = 'visible';
    progress.style.width = '25%';
    if (document.getElementById('uploadMbot').value == 'python') {
        send_code_to_mbot();
    } else {
        convert_and_send_code_to_mbot();
    }
}

async function connect() {
    portPrev = await navigator.serial.requestPort();

    await portPrev.open({ baudrate: 115200 });
}

async function disconnect() {
    // — Cerrar el puerto serie
    await portPrev.close();
    portPrev = null;
}

```

Fragmento 4.1: Abrir conexión con el Mbot por el puerto serie

### 4.3.3. Envío del código fuente al servidor

Una vez abierta la comunicación de forma correcta, se procede a enviar el programa desde el navegador a el servidor de Kibotics, para su manipulación, como un query parameter de una petición GET. Para este envío se utiliza la función `fetch()` que está proporcionada por JavaScript (Fragmento 4.2).

```

function send_code_to_mbot() {
    var editor = ace.edit("ace");
    let code = editor.getValue();

```

```

        console.log(code);
        var enc = new TextEncoder();
        const message = {
            method: "GET"
        };
        url = '/get-python-to-arduino-binary?python_code=' + JSON.
            stringify(code);
        fetch(url, message)
            . . . . .
    }

```

Fragmento 4.2: Envío del programa desde el navegador al servidor

Hay que tener en cuenta que si el programa está escrito en Blockly, debe convertirse previamente al lenguaje Python antes de enviarse, como puede verse en el fragmento 4.3.

```
var pythoncode = Blockly.Python.workspaceToCode(editor.ui);
```

Fragmento 4.3: Conversión de Blockly a Python

#### 4.3.4. Recepción del ejecutable

De la respuesta enviada por el navegador con el programa realizado al servidor, se recibe una respuesta que contiene el programa compilado para poder enviárselo a la microcontroladora.

La función `fetch()`, mencionada anteriormente, espera hasta recibir la respuesta, pudiendo extraer así el cuerpo de la respuesta, donde se encuentra la compilación del programa.

```

fetch(url, message)
    .then(function(response) {
        if(response.ok){
            responseOk = true
        }else{
            responseOk = false
        }
        return response.text();
    })
    .then(function(data) {

        var dataBuffer = enc.encode(data);

    })
    .catch(function(err) {

```

```

        console.error(err);
    });

```

Fragmento 4.4: Extracción del dato enviado en la respuesta

#### 4.3.5. Carga en el robot y ejecución

Una vez extraído el código compilado, se procede a cargarlo en el robot. Avrgirl-arduino, una librería nodejs de código abierto, proporciona esa posibilidad de quemar el programa compilado a la microcontroladora, y es compatible con un gran número de placas (entre ellas ArduinoUno). Hemos utilizado esta librería adaptándola a nuestras necesidades para el proceso de carga.

```

. . .

.then(function(data) {

    if(responseOk){
        var dataBuffer = enc.encode(data);
        upload_to_mbot(dataBuffer)
        progress.style.width = '99%';
        alertError.style.visibility = 'hidden';
    }else{
        console.log("Fallo al compilar el programa")
        infoError.innerHTML = "Error"
        alertError.style.visibility = 'visible';
        activeProgress.style.visibility = 'hidden';
    }

})

. . .

function upload_to_mbot(dataBuffer){
    let avrgirl = new AvrgirlArduino({
        board: "uno"
    });

    avrgirl.flash(dataBuffer,(error) => {
        if (error) {
            console.error(error);
            portPrev = null;
            infoError.innerHTML = "Error!"
            alertError.style.visibility = 'visible';
            activeProgress.style.visibility = 'hidden';
        } else {

```

```

        console.info('done correctly.');
```

```

        portPrev = null
        alertError.style.visibility = 'hidden';
        activeProgress.style.visibility = 'hidden';
    }
});
}
```

Fragmento 4.5: Carga del compilado al Mbot

Tras “quemar” el programa en el Mbot, este empezará a ejecutar las instrucciones que le fueron programadas.

## 4.4. Lado Cliente

El servidor de Kibotics es el encargado de servir al navegador las páginas que le permiten ir navegando por la web, y por lo tanto dirigirse a la unidad del Mbot. Pero también toma otro papel fundamental para el desarrollo del envío, ya que será el responsable de adaptar el código que el usuario ha escrito, para que pueda ser entendido por la controladora del robot a la hora de realizarse la carga.

### 4.4.1. Recepción del código fuente

Una vez que el usuario realiza su programa y pulsa en enviar, el navegador le envía al servidor el código escrito para que este lo adapte. Por lo que el servidor tendrá que extraer el código (Fragmento 4.6), que se encuentra en un *query parameter* de la URL.

```

def get_python_to_arduino_code_binary(request):
    . . . .
    python_code = json.loads(request.GET.get('python_code', None))
    . . . .
```

Fragmento 4.6: Extracción programa en el servidor

### 4.4.2. Conversión

El programa extraído en el servidor es lenguaje Python, pero el Mbot, basado en Arduino, no entiende este lenguaje. El lenguaje de alto nivel que usa Arduino, para posteriormente generar el compilado que entiende, es un C++ adaptado, por lo que se debe traducir desde Python. Esto es posible gracias a la librería PyOnArduino, la cual proporciona los mecanismos necesarios para esa traducción y conversión de un fichero.py (extensión para Python) a .ino (extensión para Arduino).

```

. . .
parsed_file = ast.parse(code)
translator.robot = 'MBot'
```



```

translator.robot_architecture = ''
translator.vars.Variables()
translator.vars.halduino_directory = py_to_arduino_path + 'HALduino
/halduino'
translator.TranslatorVisitor().visit(parsed_file)
translator.create_setup()
translator.variables_manager = translator.create_variables_manager(
    file=py_to_arduino_path + 'HALduino/variablesManager.ino')
translator.create_output('output-file.ino')
translator.create_makefile('MBot', py_to_arduino_path + 'makefiles
/')
. . .

```

Fragmento 4.7: Traducción de lenguaje Python a Arduino

#### 4.4.3. Compilación cruzada

Una vez extraído el código, se debe compilar para obtener el fichero .hex que contiene las instrucciones traducidas a hexadecimal entendidas por la microcontroladora. Esta compilación se lleva a cabo utilizando avrdude, una herramienta que permite la programación de chips AVR y que puede ser llamada por la línea de comandos.

```

def create_mbot_binary():
    shutil.copyfile(os.path.join(settings.BASE_DIR, 'kibotics-drivers/
    mbot/avrdude'), './avrdude')
    shutil.copyfile(os.path.join(settings.BASE_DIR, 'kibotics-drivers/
    mbot/libftdi1.so'), './libftdi1.so')
    shutil.copyfile(os.path.join(settings.BASE_DIR, 'kibotics-drivers/
    mbot/libreadline.so'), './libreadline.so.6')
    shutil.copyfile(os.path.join(settings.BASE_DIR, 'kibotics-drivers/
    mbot/avrdude.conf'), './avrdude.conf')
    shutil.copytree(os.path.join(settings.BASE_DIR, 'kibotics-drivers/
    mbot/arduino-1.8.10/'), 'arduino/')
    call(['make'])
    exercise_dir = './'
    # Extraemos el binario
    f_binary = open(exercise_dir + 'build-uno/output.hex', 'r')
    binary = f_binary.read()
    f_binary.close()
    return binary

```

Fragmento 4.8: Compilado y obtención del .hex

#### 4.4.4. Envío del ejecutable

Tras la compilación del programa, obtenemos el hex, que será enviado como respuesta al navegador, para que pueda ser cargado al robot.

```
...
binary = create_mbot_binary()

response = HttpResponse(binary, content_type='text/plain')
response['Content-Length'] = len(response.content)
return response
```

Fragmento 4.9: Respuesta a la petición que envió el navegador

### 4.5. Validación experimental

En las secciones anteriores no se ha mencionado en ningún momento para qué sistema operativo era compatible todo el proceso de envío, y es aquí donde recae una de las principales ventajas de este desarrollo, pues al tratarse de una tecnología de lado del navegador, proporciona compatibilidad para cualquier sistema operativo que disponga de un navegador basado en Chromium (Google Chrome, Opera, Edge), ya que actualmente Web Serial solo esta disponible para ellos. Con este diseño se permite conseguir que esta herramienta sea multiplataforma.

Además, al no necesitar una instalación previa de dependencias o programas por parte del usuario, su usabilidad es más sencilla. Esto es muy importante de cara al uso de la plataforma, ya que al estar orientada a alumnos de baja edad, es necesario que el proceso sea lo más simple posible.

La principal desventaja es que, actualmente, Web Serial Api está en fase de experimentación, por lo que necesita ser activada con un proceso previo. Para ello, para activarla por ejemplo en Chrome, debemos escribir en el navegador Chrome “*chrome://flags*” y habilitar el flag “*Experimental web Platform features*”.

La integración del Mbot ha sido probada en diferentes sistemas operativos, verificando el correcto funcionamiento (*Ubuntu 18.04*, *Ubuntu 16.04*, *Windows 10*, *Windows 8*, *MacOs Mojave Versión 10.14.6*).

Por ultimo, se han creado una serie de vídeos, donde se puede visualizar un ejemplo de realización de un programa y envío al Mbot desde la plataforma Kibotics ,para los principales sistemas operativos:

- **En Linux:** <https://youtu.be/1jyvoN5ZRxQ>
- **En Windows:** <https://youtu.be/4Wq4kMRUeIc>
- **En MacOS:** <https://www.youtube.com/watch?v=UyNa9R-L0Ps>

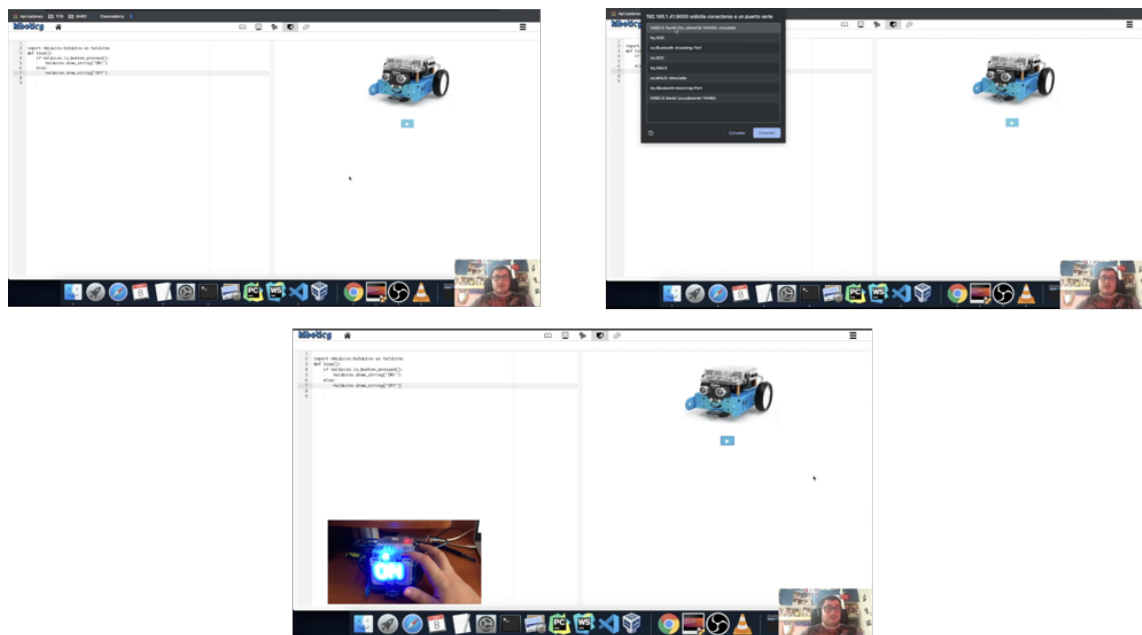


Figura 4.5: Fotogramas del vídeo de ejemplo de envío para el Mbot en MacOS