Capítulo 11

# Proyectos de aplicación ANDROID

## Primera parte

### ☼ Contenido



### ☑ Objetivo

Se pondrá en práctica lo aprendido en los capítulos anteriores, además de conocer qué tipo de aplicaciones Android se pueden realizar con las plataformas actuales de Android Studio y Arduino. Se desarrollarán aplicaciones para control de sensores del teléfono Android y su integración con Arduino; además, en cada proyecto se podrá interactuar con cada elemento del celular. Por ejemplo, el uso del dispositivo de comunicación NFC y el control por voz.

## 11.1 Introducción

Hoy en día el uso de los teléfonos celulares con Android tiene un enorme impacto en la actualidad y en el mundo real, el control desde el teléfono móvil de dispositivos del mundo real y la integración de ambos, haciendo uso de sensores del mismo dispositivo móvil, esto permite realizar el monitoreo de dispositivos electrónicos de la vida real desde la palma de tu mano. En el siguiente capítulo se abordarán algunos proyectos enfocados al manejo del dispositivo móvil y sus aplicaciones como sensores para la integración con la tarjeta Arduino para lograr integrar ambas tecnologías en proyectos del mundo real.

### Proyecto: Sensores Android para control

En este proyecto se conectará un servomotor a una placa Arduino para poder controlar desde el teléfono Android. Un servomotor es básicamente un motor cuya posición angular puede ser controlado con precisión por un microcontrolador. Se usará BLE una vez más para recibir comandos desde el dispositivo Android.

#### *Requerimientos de software y hardware*

Se requiere de un módulo BLE, se elige el chip Adafruit nRF8001 porque contiene una librería Arduino y ya tiene ejemplos de Android aplicaciones para controlar el módulo. Se utilizará un servomotor de 5v de cualquier marca, como el que se muestra a continuación:

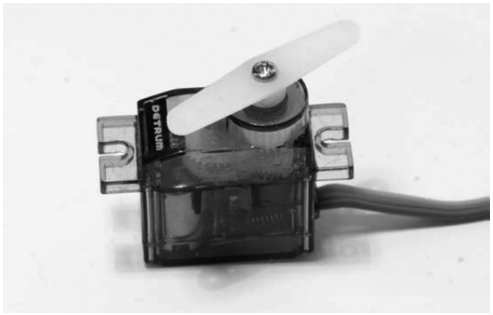



**Figura 11.1** Servomotor

La lista de componentes para este proyecto es la siguiente:

- Arduino Uno (*https://www.adafruit.com/product/50*)
- La placa bluetooth Low energy NRF8001 BLE (*https://www.adafruit.com/product/1697*)
- Un servomotor de 5v (*https://www.adafruit.com/product/1143*)
- La placa de prueba (*https://www.adafruit.com/products/64*)
- Cables para conectar macho-macho (*https://www.adafruit.com/products/1957*)

En la parte del software se requiere lo siguiente:

- La biblioteca para la placa nRF8001 se encuentra en *https://github.com/adafruit/Adafruit_nRF8001/archive/master.zip*
- La biblioteca aREST se encuentra en *https://github.com/marcoschwartz/aREST*

### *Diagrama de conexiones*

A continuación se muestra el diagrama de las conexiones del proyecto:

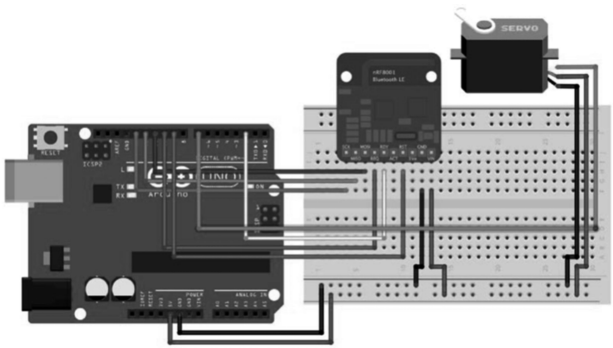



**Figura 11.2** Diagrama de conexiones

### *Diagrama del proyecto ya ensamblado*

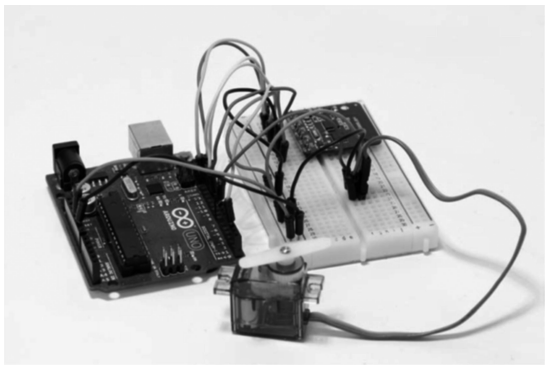


**Figura 11.3** Ensamble del proyecto

### *Prueba del servo:*

```
// Required libraries
#include <Servo.h>
```

```
// Create servo object
Servo myservo;

// Servo position
int pos = 0;

void setup( )
{
  // Attaches the servo on pin 7 to the servo object
  myservo.attach(7);
}

void loop( )
{
  // Goes from 0 degrees to 180 degrees
  for(pos = 0; pos < 180; pos += 1)
  {
    myservo.write(pos);
    delay(15);
  }

  // Goes from 180 degrees to 0 degrees
  for(pos = 180; pos>=1; pos-=1)
  {
    myservo.write(pos);
    delay(15);
  }
}
```

***El sketch de la aplicación de Arduino con el módulo Bluetooth Low Energy es:***

```
// Required libraries
#include <SPI.h>
#include “Adafruit_BLE_UART.h”
#include <aREST.h>
#include <Servo.h>

// Pins
#define ADAFRUITBLE_REQ 10
#define ADAFRUITBLE_RDY 2     // This should be an interrupt pin, on Uno
thats #2 or #3
#define ADAFRUITBLE_RST 9
```

```
// Create servo object
Servo myservo;

// Create aREST instance
aREST rest = aREST( );

// Servo position
int pos = 0;

// BLE instance
Adafruit_BLE_UART BTLEserial = Adafruit_BLE_UART(ADAFRUITBLE_REQ,
ADAFRUITBLE_RDY, ADAFRUITBLE_RST);

void setup( )
{
   // Start Serial
  Serial.begin(115200);

  // Attaches the servo on pin 7 to the servo object
  myservo.attach(7);

  // Start BLE
  BTLEserial.begin( );

  // Give name and ID to device
  rest.set_id("001");
  rest.set_name("servo_control");

  // Expose function to API
  rest.function("servo",servoControl);
}


void loop( )
{
  // Tell the nRF8001 to do whatever it should be working on.
  BTLEserial.pollACI( );

  // Ask what is our current status
  aci_evt_opcode_t status = BTLEserial.getState( );

  // Handle REST calls
  if (status == ACI_EVT_CONNECTED) {
    rest.handle(BTLEserial);
```

```
  }
}

// Control servo from REST API
int servoControl(String command) {

  // Get position from command
  int pos = command.toInt( );
  Serial.println(pos);

  myservo.write(pos);

  return 1;
}
```

## Configuración de la aplicación en Android Studio

En este proyecto se diseñará una aplicación de Android muy simple que mostrará la devolución de llamada Bluetooth en una línea de texto y la salida del sensor en otro texto. Esta vez se implementará también un botón Actualizar, que se reiniciará la devolución de llamada de Bluetooth si hay una necesidad de una actualización.

La parte del proyecto que será más sofisticada será el acceso al hardware de sensores disponibles para enviar comandos al servo y girar el eje según la orientación del eje X del dispositivo Android, determinado por el hardware del giroscopio que está incluido en el dispositivo.

Es importante considerar que las lecturas y los datos del sensor pueden variar entre diferentes dispositivos Android debido a diversas configuraciones de hardware. Entonces, de nuevo, se podría usar este proyecto como una línea de base para promover los emprendimientos.

Para hacer la configuración se realiza primero lo siguiente:

### *Configuración de permisos en la interfaz:*

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.arduinoandroid.androidarduinosensserv" >
      <uses-permission android:name="android.hardware.sensor.
gyroscope"/>
            <uses-permission android:name="android.permission.BLUETOOTH"/>
            <uses-permission android:name="android.permission.BLUETOOTH_
ADMIN"/>
<application
android:allowBackup="true"
```

```
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/AppTheme" >
<activity
android:name=".MainScreen"
android:label="@string/app_name" >
<intent-filter>
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.
LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

A continuación se muestra la interfaz de la aplicación:


**Figura 11.4** Interfaz de la aplicación

### *Proyectos futuros*

Las lecturas de orientación de la aplicación de Android se pueden visualizar en la aplicación con gráficos en tiempo real, y este proyecto podría ser más avanzado e integrado en una apli-

cación remota de control de objetos donde el usuario del teléfono inteligente Android pueda controlar un objeto que esté conectado al servomotor desde una distancia específica.

La aplicación simple pero útil de tal acción sería abrir una puerta o control a un robot móvil a través de un giroscopio. Los teléfonos inteligentes Android también tienen una serie de sensores disponibles como el acelerómetro y el magnetómetro, que podrían usarse efectivamente para controlar diferentes componentes conectados al microcontrolador Arduino. Todo esto en conjunto permite crear aplicaciones a través de bluetooth para resolver situaciones del mundo real en las áreas de Internet de las Cosas, aplicaciones wearable.

## Proyecto: Control de activación por voz

En este proyecto se utilizará otra característica de los dispositivos Android para controlar un Arduino sistema: reconocimiento de voz. Se controlará un relevador conectado a una placa de Arduino que envíe comandos de voz desde el teléfono.

El relevador se puede conectar a muchas cosas; por ejemplo, a una cerradura eléctrica para que se pueda abrir y cerrar una puerta simplemente hablando en el teléfono. También se puede conectar el relé a una lámpara para encenderla y apagarla con el envío de un comando de voz por la misma vía.

### *Hardware y software requerido*

Se requiere de un módulo BLE. Se elige el chip Adafruit nRF8001 porque contiene una buena librería Arduino y ya tiene ejemplos de Android aplicaciones para controlar el módulo. Para realizar la activación de un dispositivo se utilizará un relevador como el que se muestra a continuación, o de cualquier marca.


**Figura 11.5** Relevador

### *Diagrama de conexión del módulo bluetooth*

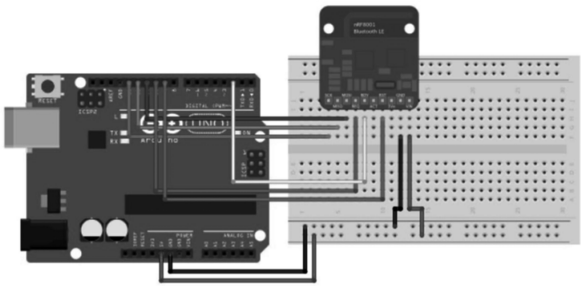


**Figura 11.6** Diagrama de conexión

La conexión de todos los elementos se podrá observar como sigue:

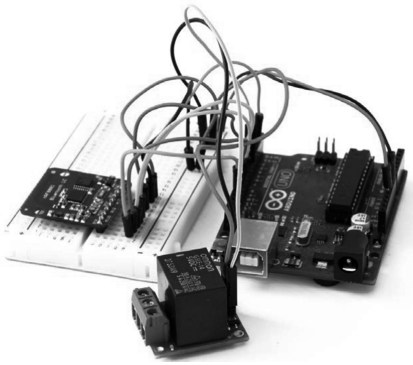


**Figura 11.7** Conexión de elementos

A continuación se presenta una imagen de las conexiones del módulo de comunicación y el relevador:

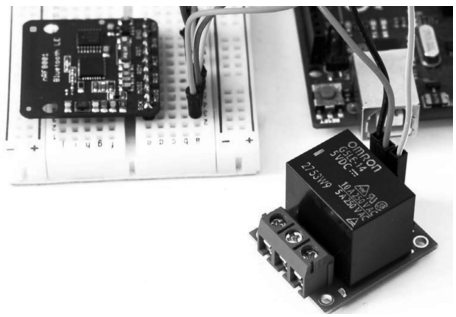


**Figura 11.8** Conexiones del módulo y relevador

### *Sketch del Arduino*

```
// Control Arduino board from BLE

// Libraries
#include <SPI.h>
#include “Adafruit_BLE_UART.h”
#include <aREST.h>

// Pins
#define ADAFRUITBLE_REQ 10
#define ADAFRUITBLE_RDY 2    // This should be an interrupt pin, on Uno thats
#2 or #3
#define ADAFRUITBLE_RST 9

// Relay pin
const int relay_pin = 7;

// Create aREST instance
aREST rest = aREST( );

// BLE instance
Adafruit_BLE_UART BTLEserial = Adafruit_BLE_UART(ADAFRUITBLE_REQ,
ADAFRUITBLE_RDY, ADAFRUITBLE_RST);

void setup(void)
{
  // Start Serial
  Serial.begin(115200);

  // Start BLE
  BTLEserial.begin( );

  // Give name and ID to device
  rest.set_id(“001”);
  rest.set_name(“relay_control”);

   // Init relay pin
  pinMode(relay_pin,OUTPUT);
}

void loop( ) {
```

```
  // Tell the nRF8001 to do whatever it should be working on.
  BTLEserial.pollACI( );

  // Ask what is our current status
  aci_evt_opcode_t status = BTLEserial.getState( );

  // Handle REST calls
  if (status == ACI_EVT_CONNECTED) {
    rest.handle(BTLEserial);
  }
}
```

## Configuración de la aplicación en Android Studio

En este proyecto se implementará una aplicación de Android que aproveche el uso de la API de reconocimiento de voz; se saldrá de ese texto en un campo EditText. En el fondo, también se incluirán los servicios BLE para conectar a este módulo y enviar mensajes al mismo.

Una vez que se tiene el BLE y la configuración de la API de reconocimiento de voz, se pueden conectar a ambos al configurar condiciones donde, si la voz se reconoce como encender, se prenderá el relevador; mientras que si se reconoce el apagado, el relevador se desconectará.

La interfaz se muestra como sigue:

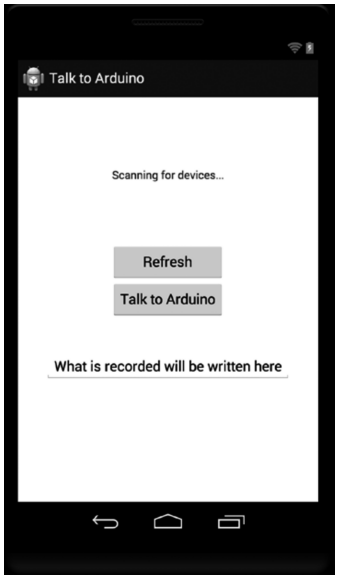



**Figura 11.9** Interfaz de la aplicación

### *Aplicaciones futuras*

Este proyecto base ofrece infinitas posibilidades y quizás incluya comandos que se reconozcan y conecten a otros componentes y sensores para mejorar las capacidades de la aplicación activada por voz. Se espera que este trabajo básico aliente a mejorar los proyectos del lector.

## Proyecto: Control de acceso vía NFC desde el dispositivo móvil Android

Se observarán las capacidades de integrar el Arduino Near Field Communication (NFC) de SeeedStudio con un Android habilitado para NFC y una aplicación que utiliza la tecnología Android Beam para enviar un mensaje desde la aplicación de Android para la antena del shield NFC.

NFC permite la comunicación instantánea entre dos dispositivos que están cerca uno del otro, hecho que lo convierte en la tecnología perfecta para abrir cerraduras de puerta o servicios de pago.

Se realizará una aplicación de domótica: El shield NFC se conectará a la placa Arduino Uno junto con el relé. Por lo tanto, se podrá encender o apagar el relé dependiendo del mensaje enviado por la aplicación de Android.

### *Requerimientos de hardware y software*

Lo primero que se necesitará es una placa Arduino Uno y por ende un shield NFC. Existen diversos escudos NFC disponibles en el mercado pero, para este proyecto, se elige el NFC V2.0 de SeeedStudio ya que posee documentación y algunos códigos de ejemplo disponibles.

A continuación se muestra la lista del material que se requerirá:

- La placa Arduino Uno (*http://www.adafruit.com/product/50*)
- El módulo de relé de 5 V (*http://www.pololu.com/product/2480*)
- El shield Arduino NFC (*http://www.seeedstudio.com/depot/nfcshield-v20-p-1370.html*)
- Un protoboard (*https://www.adafruit.com/product/64*)
- Cables macho-macho (*https://www.adafruit.com/product/758*)

También se requiere el siguiente software:

1. Primero, descargar la librería PN532 (*https://github.com/Seeed-Studio/PN532*) y colocar todas las carpetas en la de bibliotecas de Arduino.

2. Luego, descargar la biblioteca NDEF (*https://github.com/don/NDEF*) y colocarla en la carpeta de bibliotecas de Arduino; cambiar el nombre a NDEF.

## Configuración del hardware

A continuación se presenta el shield NFC para Arduino:

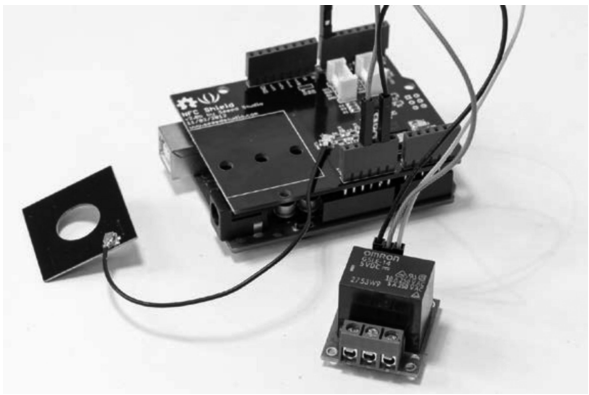


**Figura 11.10** Shield NFC

### *Prueba del código del shield NFC para Arduino*

```
// Required libraries
#include <SPI.h>
#include <PN532_SPI.h>
#include <PN532.h>
#include <NfcAdapter.h>

// NFC instances
PN532_SPI pn532spi(SPI, 10);
NfcAdapter nfc = NfcAdapter(pn532spi);

void setup(void) {

  // Start Serial
  Serial.begin(9600);

  // Start NFC chip
  nfc.begin( );
  Serial.println("NFC shield started");
}

void loop(void) {

  // Start scan
```

```
  Serial.println("\nScan a NFC tag\n");
  if (nfc.tagPresent( ))
  {
    NfcTag tag = nfc.read( );
    tag.print( );
  }
  delay(5000);
}
```

### *Código de la aplicación de Arduino*

Ahora se escribirá el código que recibirá los comandos de la aplicación NFC de Android cuyo objetivo será encender o apagar el relé cuando el shield NFC reciba uno dado desde el dispositivo Android. Como el código para esta parte es bastante largo, se dividirá en varias partes que se detallarán individualmente.

```
// NFC relay controller
// This code is based on the code from Don Coleman

// Include libraries
#include "SPI.h"
#include "PN532_SPI.h"
#include "snep.h"
#include "NdefMessage.h"

// Relay pin
#define RELAY_PIN 8

// Code for the On states
#define RELAY_ON "oWnHV6uXre"

// Create NFC object instance
PN532_SPI pn532spi(SPI, 10);
SNEP nfc(pn532spi);

// NFC buffer
uint8_t ndefBuf[128];

void setup( ) {

    // Start Serial communications
    Serial.begin(9600);
    Serial.println("NFC Peer to Peer Light Switch");
```

```
    // Declare relay pin as output
    pinMode(RELAY_PIN, OUTPUT);
}

void loop(void) {

  // Wait for NFC message
  Serial.println("Waiting for message from Peer");
  int msgSize = nfc.read(ndefBuf, sizeof(ndefBuf));

  if (msgSize > 0) {

    // Read message
    NdefMessage message  = NdefMessage(ndefBuf, msgSize);

    // Make sure there is at least one NDEF Record
    if (message.getRecordCount( ) > 0) {

      NdefRecord record = message.getRecord(0);
      Serial.println("Got first record");

      // Check the TNF and Record Type
      if (record.getTnf( ) == TNF_MIME_MEDIA && record.getType( ) ==
"application/com.arduinoandroid.arduinonfc") {
        Serial.println("Type is OK");

        // Get the bytes from the payload
        int payloadLength = record.getPayloadLength( );
        byte payload[payloadLength];
        record.getPayload(payload);

        // Convert the payload to a String
        String payloadAsString = "";
        for (int c = 0; c < payloadLength; c++) {
          payloadAsString += (char)payload[c];
        }

        // Print out the data on the Serial monitor
        Serial.print("Payload is ");Serial.println(payloadAsString);

        // Modify the state of the light, based on the tag contents
        if (payloadAsString == RELAY_ON) {
            digitalWrite(RELAY_PIN, HIGH);
        } else {
```

```
            digitalWrite(RELAY_PIN, LOW);
          }
        } else {
          Serial.print("Expecting TNF 'Mime Media' (0x02) with type
'application/com.arduinoandroid.arduinonfc' but found TNF ");
          Serial.print(record.getTnf( ), HEX);
          Serial.print(" type ");
          Serial.println(record.getType( ));
        }
      }
    }
}
```

### *Configuración de la aplicación en Android Studio*

En este proyecto se implementará una aplicación Android que aprovecha el uso de NFC API y el hardware que permite enviar un mensaje de tipo MIME para encender y apagar el relevador.

### *Permisos en la aplicación Android*

```
<uses-permission android:name="android.permission.NFC" />
      <uses-feature android:name="android.hardware.nfc"
android:required="true" />
```

La interfaz de la aplicación se muestra como sigue:


**Figura 11.11** Interfaz de la aplicación

Una vez incluidos todos los métodos se debería de poder construir la aplicación y ejecutarse en el dispositivo físico Android con capacidades NFC, con la ejecución de Android 4.3 o superior y con Android Beam activado dentro de la configuración.

Se puede encender el relevador tocando el botón Conectar y sosteniendo el teléfono frente al shield NFC durante al menos 5 o 10 segundos. En este punto se debe volver a tocar en la interfaz de usuario para enviar el mensaje.

### *Futuros proyectos*

Este proyecto se enfocó principalmente en el uso de la tecnología NFC para transmitir un mensaje y hacer que lo lea el shield de Arduino NFC. La experiencia de usuario ideal sería que simplemente se acerque el teléfono frente al shield NFC y se encienda la luz. Esto podría lograrse a través de la emulación de Host-Card o con modificaciones adicionales de este proyecto.

## RESUMEN

En este capítulo se desarrollaron diversos proyectos relacionados con las plataformas Android y Arduino y la integración de ambas tecnologías; el objetivo fue tener una base tecnológica para que el lector pueda hacer sus propios desarrollos y crear con ello nuevas aplicaciones para la resolución de problemas de la vida real. En el siguiente capítulo se desarrollará otra serie de proyectos de aplicación con Android.

Capítulo 12

# Proyectos de aplicación ANDROID

## Segunda parte

### ☼ Contenido



### ☑ Objetivo

En este capítulo se abordará una serie de proyectos por medio de la plataforma de Android con aplicaciones, enfocada al desarrollo de dispositivos de tecnología aplicada con la integración de Arduino y Android.

## 12.1 Introducción

Las tecnologías de desarrollo que existen en la actualidad permiten interactuar con el mundo físico para llevar a cabo proyectos de aplicación real; con la integración de estas tecnologías de hardware y software de programación es posible el continuo avance de la tecnología aplicada en la vida cotidiana.

## 12.2 Particle Core

Particle Core es una placa compatible con Arduino pero con algunas reservas. De hecho, el núcleo de partículas se basa en procesadores ARM Cortex-M3, que utilizan una arquitectura diferente de los microprocesadores Atmega utilizados en la mayoría de los Arduino compatibles tableros. Todavía es posible considerar que el núcleo de partícula es Arduino-familiar, ya que se puede programar mediante los mismos comandos y estructura que se utilizó para todas las placas de Arduino. Las librerías están específicamente desarrolladas para Particle Core con el fin de utilizar el Conectividad Wifi.

En la siguiente dirección se puede encontrar el kit de desarrollo: *https://www.particle.io/*

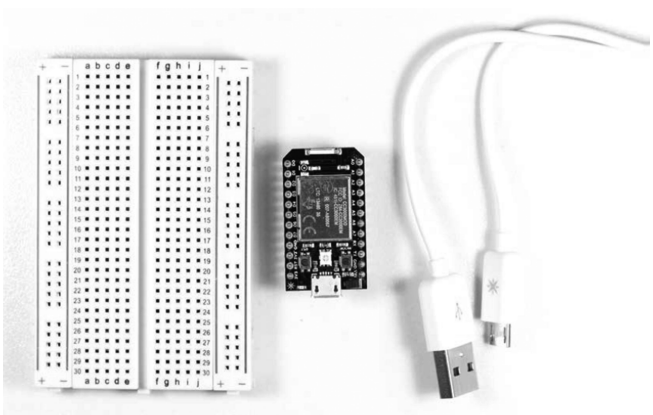


**Figura 12.1** Kit de desarrollo

### 12.2.1 Programación de Particle Core

Como se mencionó anteriormente, no es posible usar el Arduino IDE para programar la tarjeta Particle Core desde Arduino; IDE no tiene soporte para este tipo de placa. Dado que estas placas implementan la tecnología Wifi se les puede programar de forma inalámbrica, lo que podría ser un gran beneficio en algunos casos; especialmente cuando se trata de proyectos usables. Esto permite hacer la depuración en tiempo real. Así que la programación inalámbrica

sola es una razón para considerar Particle Core para un proyecto portátil. Entonces, también se genera una conexión con Internet, lo que hace a algunos proyectos realmente interesantes.

Para poder acceder a la plataforma en línea y para realizar la programación en el IDE para Particle Core, se ingresa al siguiente link: *https://build.particle.io/login*


**Figura 12.2** Imagen de la plataforma en línea

Ejemplo de un sketch en el entorno Web:

```
int led = D7;
void setup( ){
pinMode(led,OUTPUT);
}
void loop( ){
digitalWrite(led,HIGH);
delay(1000);
digitalWrite(led,LOW);
delay(1000);
}
```

Existe parecido al entorno de Arduino IDE, sólo que hay a diferencia una serie de menús y algunas funciones adicionales. A continuación se muestra el menú de herramientas:

**Figura 12.3** Imagen del menú de herramientas

El cual permite verificar el sketch; o bien, con el botón de Flash se puede descargar en la placa.

## Proyecto: Reloj inteligente con tarjeta Particle Core

En este proyecto de utilizará la placa Particle Core que interactuará con diferentes servicios en línea; la misma actuará como la base de hardware en todo este capítulo, donde se creará un propio reloj inteligente. Este proyecto se basa en la idea de futuro, donde todas las áreas de la ciudad más pequeña estén cubiertas por Wifi y donde los relojes inteligentes, entre otras cosas, se conecten directamente a Internet de manera fluida. Puede que no lo vivamos todavía, de cualquier modo el reloj inteligente de este capítulo funcionará en el presente. La mejor parte de este proyecto es cuando alguien pregunte ¿Qué es eso en tu brazo? Por lo cual el usuario puede responder: es un reloj inteligente que construí yo mismo.

Siempre ha habido un problema con los relojes inteligentes. Antes que nada, todos los relojes que es posible comprar son dependientes de un teléfono móvil. Luego, estos dispositivos nunca parecen hacer exactamente lo que se pretende que hagan.

La solución entonces es construir uno propio y con componentes comerciales, lo cual puede acercarse mucho a la realidad. Incluso puede ser mejor que la realidad, ya que este reloj puede personalizarse como mejor parezca al usuario.

Al final de este capítulo se tendrá una buena idea de cómo construir proyectos portátiles propios usando las herramientas propuestas. Como se ha dicho antes, este libro es una introducción pero la meta es que el lector desarrolle este conocimiento para realmente hacer los proyectos suyos. Hacer modificaciones en los diseños y todo tipo de cambios es el objetivo de los ejercicios aquí propuestos.

### *Componentes*

- Tarjeta Particle Core
- Una batería de litio de 3.7V
- Una pantalla OLED de Adafruit de 128 pulgadas x 64 pulgadas
- Un circuito de carga de la batería Adafruit
- Un material de lámina de plástico de 1 mm
- Material de cuero o de alta resistencia, de al menos 30 cm x 10 cm
- Soldadura
- Un cuchillo o exacto
- Punzones de costura de cuero e hilo resistente
- Ojales y una herramienta ojeteador
- Un protoboard pequeño

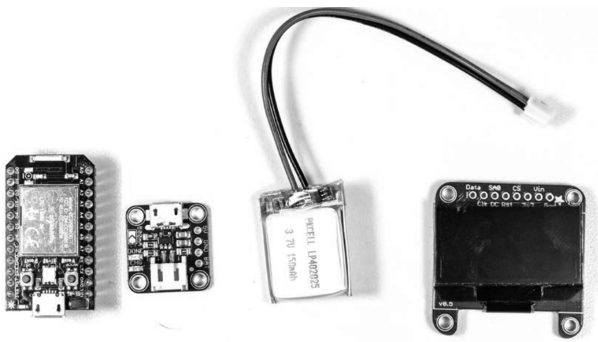


**Figura 12.4** Elementos que conforman el proyecto

### *Conexiones de la pantalla OLED*

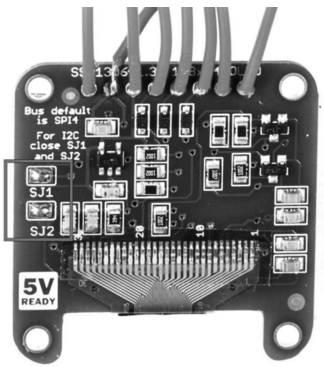



**Figura 12.5** Imagen de las conexiones

- Data conecta a D0
- CLK conecta a D1
- RST conecta a D4
- SA0 conecta a GND
- 3V3 conecta a 3V3
- GND conecta a GND

### *Diseño del reloj y soldado*

A continuación, se muestra el dibujo de la plantilla del reloj:

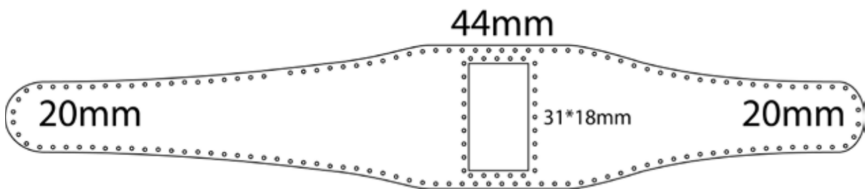



**Figura 12.6** Plantilla del proyecto

Aquí se puede ver el corte de la correa de piel:

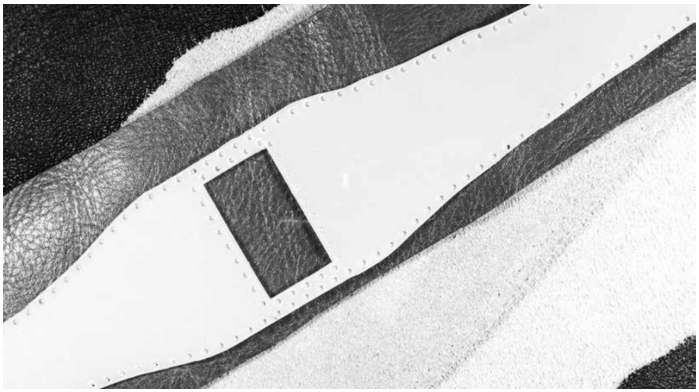


**Figura 12.7** Correa de piel para el reloj

Las partes que conforman cada una de las piezas de hardware y que se van a colocar en la correa del reloj, son las siguientes:

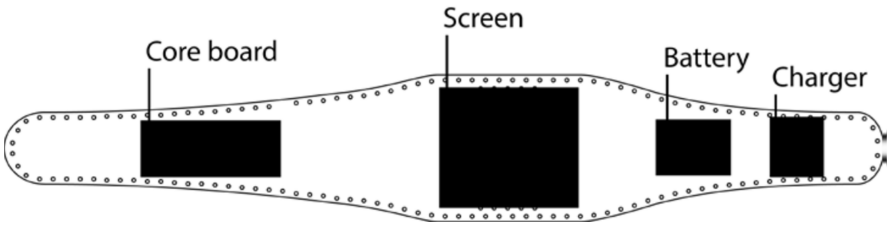



**Figura 12.8** Partes de las piezas del hardware

A continuación, se muestran las partes a soldar:

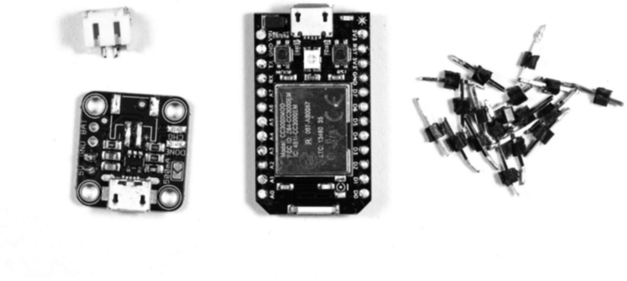


**Figura 12.9** Elementos a soldar

Se conectan las piezas:

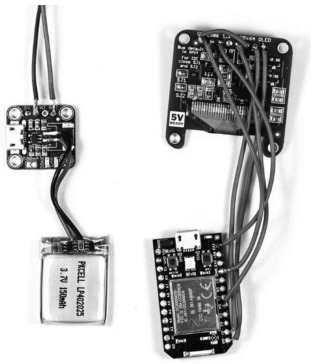


**Figura 12.10** Conexión de elementos

Luego, se muestran las uniones entre las partes de la plantilla y la piel:

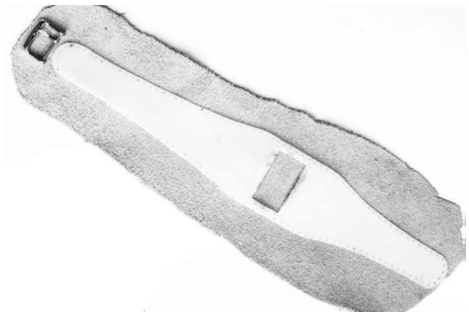


**Figura 12.11** Uniones de las partes

Se puede observar el corte requerido para la pantalla OLED:

**Figura 12.12** Corte de la pantalla

Se presenta primeramente la parte de piel que se debe cortar, ello para que se vea totalmente el cuadro:

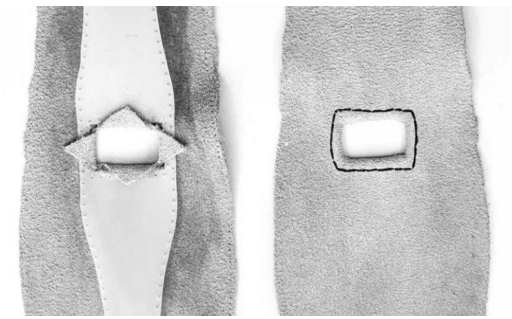


**Figura 12.13** Superficie de piel a cortar

A continuación, se muestran las piezas pegadas sobre la plantilla:

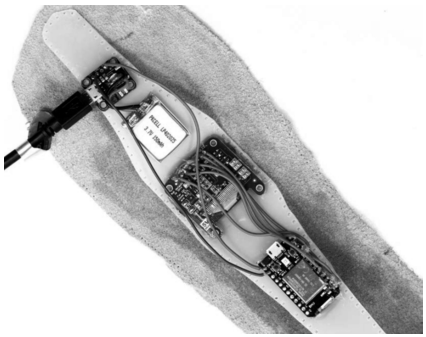


**Figura 12.14** Piezas a pegar

En la parte final se agrega una hebilla para sujetar el reloj:

**Figura 12.15** Hebilla que sujetará el reloj

Enseguida se muestra cómo la segunda parte cubre a la plantilla:

**Figura 12.16** Cubierta de la plantilla

Se muestran las dos cubiertas ya unidas:

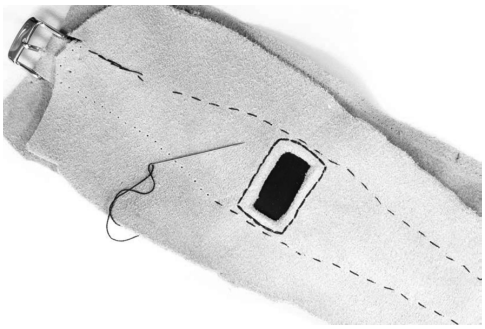


**Figura 12.17** Unión de ambas cubiertas

A continuación, se realizan los orificios para sujetar y apretar:

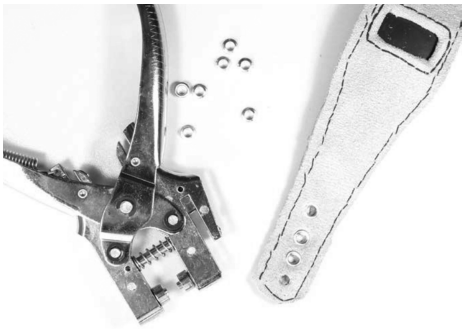


**Figura 12.18** Orificios de la correa

### *Funcionalidad del proyecto*

Como se mencionó anteriormente, la versión del reloj inteligente que se está construyendo está diseñada principalmente para recibir notificaciones. La idea es basar la mayoría de las notificaciones en IFTTT, ya que tiene una buena infraestructura para conectar diferentes servicios y ahorra mucho código y tiempo. Se podrían conectar la mayoría de los servicios disponibles a través de IFTTT sin usarla, pero sí las API abiertas disponibles de los diferentes servicios. Sin embargo, IFTTT hace que sea mucho más fácil la conexión sin la necesidad de aprender sobre todas las API de cada servicio de forma individual.

Volviendo a la parte del código, el siguiente ejemplo muestra el principio básico que se utilizará para conectar el reloj a diferentes servicios y mostrar notificaciones en la pantalla. El primer firmware para el reloj utiliza la misma función IFTTT; una vez que el siguiente se actualiza a la placa base es preciso dirigirse al sitio web de IFTTT y crear una receta desde la cuenta pro-

pia, donde cualquier actualización hecha en el estado de Facebook desencadena nuevamente el "getMail" del canal de partículas con el fin de refrescar la memoria. Para que funcione el siguiente ejemplo, es necesario incluir las bibliotecas Adafruit_GFX y Adafruit_SSD1306.

### *Código de la aplicación:*

```
#include "Adafruit_GFX.h"
#include "Adafruit_SSD1306.h"
#define OLED_RESET D4
Adafruit_SSD1306 display(OLED_RESET);
#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2
#define LOGO16_GLCD_HEIGHT 16
        #define LOGO16_GLCD_WIDTH 16

#if (SSD1306_LCDHEIGHT != 64)
#error("Height incorrect, please fix Adafruit_SSD1306.h!");
#endif
//Declare the functions
int fbStatus(String topic);
int displayTime( );
int instaGram(String insta);
int bestBuy(String pebble);
int missedCall(String pNbr);
void setup( ) {
Serial.begin(9600);
//Initialize the functions
Spark.function("getfbStatus", fbStatus);
Spark.function("getBSinsta", instaGram);
Spark.function("getBestBuy", bestBuy);
Spark.function("getMissCall", missedCall);

// Init display
         display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the
I2C addr 0x3D (for the 128x64)
              //Clear the memory
              }
void loop( ) {
              //Show the time and date
              displayTime( );
              }
              //If your Facebook status is updated show the status
```

```
        int fbStatus(String topic){
        if(topic !=“”){
        display.clearDisplay( );
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(10,11);
        display.println(topic);
        display.display( );
        delay(2000);
        }
  else return -1;

        }
        //If Brittany Spears updates her Instagram let me know
        int instaGram(String insta){
        if(insta !=“”){
        display.clearDisplay( );
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(10,11);
        display.println(“New Briteny”);
        display.display( );
        delay(2000);
        }
        else return -1;
  }
        //If the price on the latest Pebble smart watch changes let me know
        int bestBuy(String pebble){
        if(pebble !=“”){
        display.clearDisplay( );
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(10,11);
        display.println(“Pebble price”);
        display.println(pebble);
        display.display( );
        delay(2000);
        }
  else return -1;
        }
        //If I have a missed phone call on my Android device show the number
        int missedCall(String pNbr){
        if(pNbr !=“”){
        display.clearDisplay( );
```

```
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(10,11);
        display.println("Missed call");
        display.println(pNbr);
        display.display( );
        delay(2000);
        }
        else return -1;
        }
        int displayTime( ){
/*Set the courser and display the time in hours, minutes and seconds. Remem-
ber this is standard time so you need to add or decrease the hours depending
on your time zone*/
        display.clearDisplay( );
        display.setTextSize(2);
        display.setTextColor(WHITE);
        display.setCursor(15,15);
        display.print(Time.hour( ));
        display.print(":");
        display.print(Time.minute( ));
        display.print(":");
        display.print(Time.second( ));
        /*Set the cursor and display the date in days, months and year*/
        display.setCursor(15,40);
        display.print(Time.day( ));
        display.print("/");
        display.print(Time.month( ));
        display.print("/");
        display.println(Time.year( ));
        display.display( );
        delay(1000);
}
```

Esta es la aplicación ya en funcionamiento:

**Figura 12.19** Muestra de la aplicación terminada

## Proyecto: Grabadora de paseo de bicicleta

Las computadoras de bicicleta informan sobre el desempeño cuando el usuario corre en la carretera. Se almacena información sobre la velocidad de las ruedas, la distancia del viaje o el tiempo en que se ha estado montando. Hay computadoras para bicicletas que se conectan al teléfono y permiten iniciar sesión; dan cuenta del rendimiento para luego compararlo con otros viajes a través de gráficos. El proyecto se acerca un poco más a la fabricación de una computadora propia para bicicletas usando el dispositivo Android y un poco de ayuda de componentes electrónicos integrados. Es interesante comparar la vuelta de la rueda con el giro de los pedales: si se está corriendo cuesta abajo, la cantidad de esfuerzo que se hace en la bicicleta es mucho menor que cuando se está en una carretera plana. Es posible medir ese esfuerzo comparando los datos de dos sensores idénticos: uno en una rueda y uno en un pedal.

Este es un proyecto desafiante para el cual se usará el código de bajo nivel en el lado de Arduino —en forma de interrupciones de hardware—, mientras que los datos se representan en la parte superior del video, en el lado de Android.

La esencia de este proyecto es crear un dispositivo para ayudar a mejorar el tipo de información que se obtiene de computadoras que operan con bicicletas contemporáneas. Usualmente se recibe información sobre velocidad y distancia. Las computadoras más recientes permiten ingresar información personal, como el peso o cuerpo masa, para estimar el esfuerzo (medido en kilocalorías) que se hizo durante el ejercicio.

Al mismo tiempo, se pretende ofrecer herramientas para explorar formas alternativas de representar la información obtenida de la bicicleta. El uso de la cámara, cuando se instala correctamente en el mango de la bicicleta, ofrece una vista del viaje desde el propio punto de vista y permite agregar más información como una capa en la parte superior del video. La mayoría de los ciclistas tienen rutas preferidas para su entrenamiento. Esta herramienta puede ayudar a aprender sobre la forma en que se hace el ejercicio; por ejemplo, puede mostrar cuándo podría llegarse al límite.

Resumen del diseño:

- Diseñar un circuito que lea información de dos sensores, uno conectado a una rueda y otro unido a los pedales de la bicicleta.
- Contener el circuito dentro de una caja montada en el mango de la bicicleta.
- Construir la caja para tener espacio suficiente que lleve el teléfono (o tableta, pero se proporcionan planos para una caja con espacio para un teléfono solamente).
- Diseñar una caja para transportar un paquete de baterías.
- Dado que se experimentará con la mejora del software, dejar los conectores en la placa Arduino visible para mejorar el firmware sobre la marcha.
- Usar algunos LED para proporcionar retroalimentación visual sin valerse de la pantalla.
- Usar botones de comando individuales para interactuar con el dispositivo Android sin tocar la pantalla.

- Crear un software para grabar una secuencia de video del viaje, así como las velocidades de rotación instantáneas de los pedales y las ruedas.
- Crear un software que reproduzca el video con una capa de información en la parte superior.
- Hacer que todo el mecanismo funcione con pilas.

Este es el proyecto armado con los elementos requeridos:

**Figura 12.20** Vista del proyecto

A continuación se muestran las partes del proyecto:

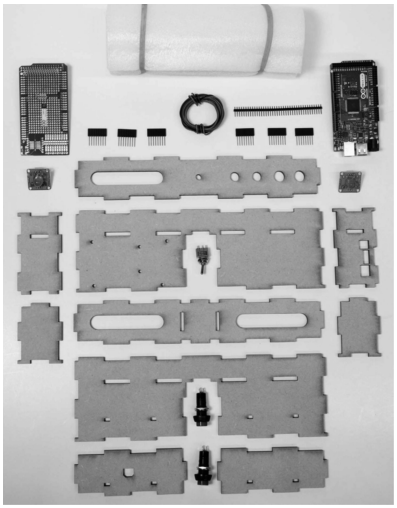


**Figura 12.21** Elementos del proyecto

Partes del proyecto:

- Arduino MEGA ADK
- Shield para prototipo para Arduino MEGA
- Pines macho-hembra y macho-macho
- Botones pulsadores para montarse sobre la base
- Leds y resistencias

## *Esquema del prototipo y partes de la caja*

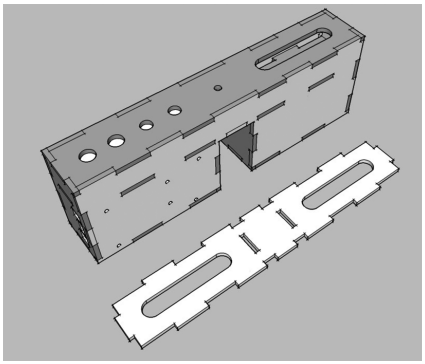


**Figura 12.22** Imagen del prototipo

Conexiones del hardware:

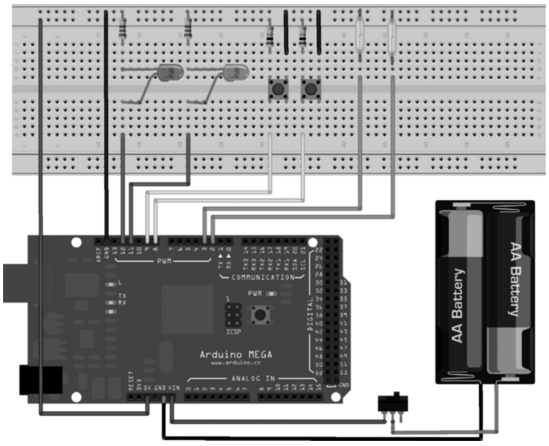



**Figura 12.23** Imagen de las conexiones

Conexión y soldado del hardware en el shield protoboard:

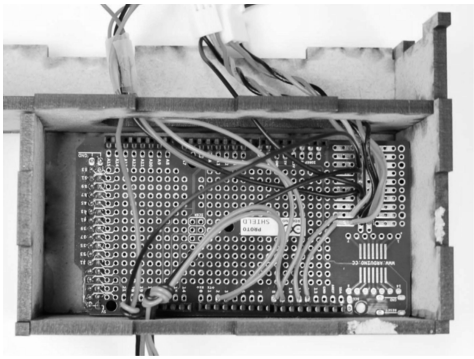


**Figura 12.24** Imagen de las soldaduras y conexiones

Conexión de la pila y ajuste:

**Figura 12.25** Pilas y ajuste

### *Programación de la computadora de bicicleta:*

- Esperar hasta que un dispositivo Android se conecte.
- Al conectarse, escuchar los botones. Cuando el usuario presiona el botón Inicio, se informa al teléfono y éste envía la velocidad de la rueda y del pedal por intervalos en forma periódica.
- Continuar hasta que el usuario presione el botón Detener o hasta que desconecte el teléfono.

### *Esqueleto de la aplicación para Arduino:*

```
#include <libraries_Communication>
int declareSomeVariables;
```

```
libComm mqtt = mqttConstructor( );
long timer = 0; // used to count time
void setup( ) {
initSerialComm( ); // to debug that things are going ok
mqtt.initMqttComm( ); // to establish the communication towards the Android
device
// determine which function will act for each interrupt arrival
attachInterrupt(Wheel, callBackWheel);

attachInterrupt(Pedal, callBackPedal);
}
void loop( ) {
if(!connected) turnOffGreenLED( );
if(connected && !subscribed) {
subscribed = mqtt.checkSubscription( );
turnOnGreenLED( );
}
if(connected && subscribed) {
if(buttonStart( )) {
mqtt.publish(start);
turnOnRedLED( );
}
if(buttonStop( )) {
mqtt.publish(start);
turnOffRedLED( );
}
}
if(aSecondPassed(timer)) {
disableInterrupts( );
mqtt.publish(counterWheel);
reset(counterWheel);
mqtt.publish(counterPedal);
reset(counterPedal);
reset(timer);
enableInterrupts( );
}
}
void callBackWheel( ) {
counterWheel++;
}
void callBackPedal( ) {
counterPedal++;
```

***Construcción de la aplicación en Android:***

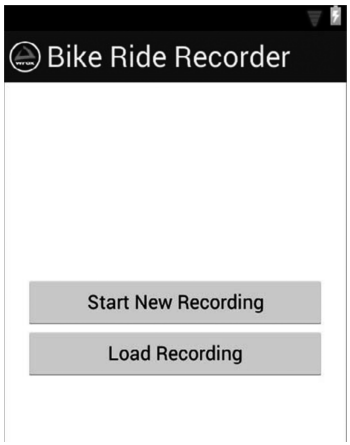



**Figura 12.26** Pantalla de inicio para el recorrido

Pantalla para captura de la cámara:

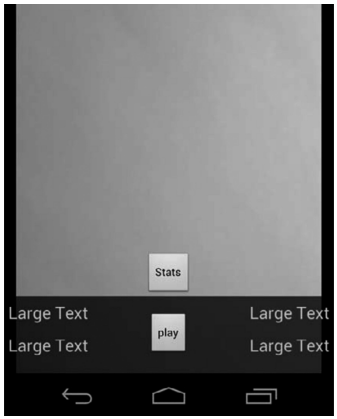



**Figura 12.27** Pantalla de captura

**Mejoras al proyecto:**

***Mecánica***

El cuadro actual para el proyecto es sólo la primera iteración de lo que podría llegar a ser. Una vez que el código está suficientemente limpio, lo primero que hay que hacer es ocultar todos

los contactos metálicos. Esto significa que la caja tomará una forma completamente diferente. También sería inteligente hacer un diseño que cubra por completo el teléfono —excepto la cámara— para protegerlo de condiciones climáticas adversas. Para esto, es preciso considerar el uso de acrílico transparente al menos para una parte de la caja.

### *Sensores*

Como se mencionó al principio del capítulo, este proyecto es una invitación a mejorar las medidas del esfuerzo realizado durante el ejercicio. Otros sensores interesantes que podrían proporcionar la información sobre la condición física del usuario podrían ser:

- Un sensor de pulso implementado en el mango al exponer un par de contactos metálicos, pues ayudará a medir la condición cardíaca mientras se hace ejercicio.
- Un giroscopio para medir el ángulo de la bicicleta con respecto al suelo.
- Un sensor de presión debajo del asiento, pues en el caso del ciclismo mientras se está parado se requiere un mayor esfuerzo que cuando el usuario está sentado.

La cuestión no es tanto mapear todos estos datos, sino cómo se puede traducir en esfuerzo o energía-consumo. En otras palabras, es preciso encontrar una forma de mapear toda la información en kilocalorías para saber si se puede o no comer pizza para la cena, por ejemplo.

## *Desarrollo de proyectos con el shield 1Sheeld o shields virtuales en el mismo teléfono*

**1Sheeld** integra conectividad Bluetooth Low Energy al Arduino y se interactúa de forma automática con una aplicación en el teléfono móvil con Android. De forma directa el usuario tendrá a través de shields virtuales en su mismo teléfono una seria de funciones que a su vez podrán interactuar con el hardware y software del dispositivo y del control de la tarjeta Arduino. Acelerómetro del teléfono, sensores, conectividad, servicios, etc., podrán tener control directo de lo que esté conectado en la tarjeta Arduino.


**Figura 12.28** Tarjeta de desarrollo One Shield y aplicación

El 1Sheeld es una placa con un microcontrolador y un módulo bluetooth para transmitir datos entre Arduino y una aplicación móvil para la plataforma Android, es decir, el teléfono inteligente. La plataforma y la aplicación en Android abren los sensores y capacidades del teléfono para ser usados como shields virtuales para Arduino. Se puede encontrar como referencia más información al respecto en: *https://1sheeld.com/*

En la siguiente página también se encuentra la descripción de todos los shields que se pueden configurar en el teléfono: *https://1sheeld.com/shields/*

### *Tipos de shields*

- Shields para entrada/salida básicos
- Shields de sensores
- Shields especiales
- Shields de comunicación
- Shields para redes sociales

A continuación se presenta una guía de uso del Shield 1Sheeld para Arduino:

- 1Sheeld Library funciona en Arduino IDE 1.0.5+

- 1Sheeld está basado en una tecnología BLE "Bluetooth Low Energy" que sólo es compatible con dispositivos que poseen bluetooth 4; así que es preciso comprobar si el dispositivo cuenta con dicha tecnología, especialmente en aquellos antiguos de Android.

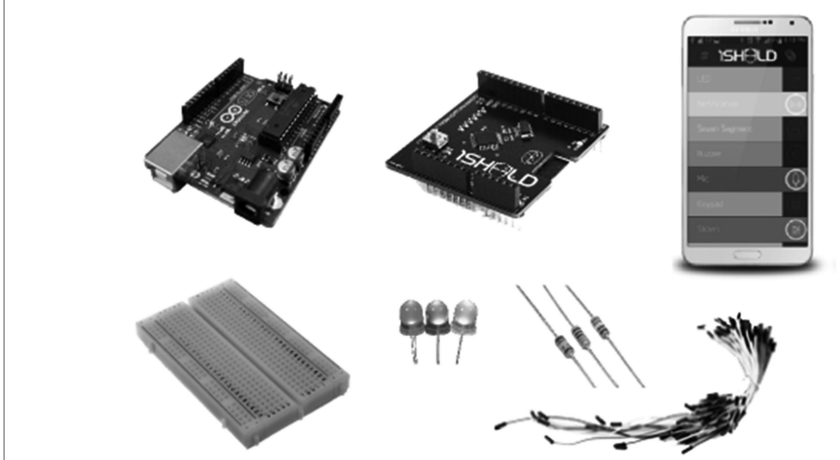


**Figura 12.29** Elementos que conforman la plataforma

Ajuste de voltaje del 1Sheeld:

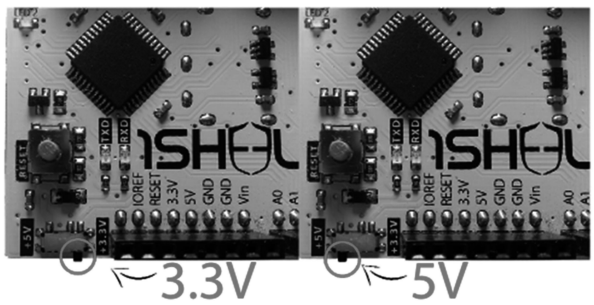



**Figura 12.30** Ajuste en el voltaje

### *Colocar el Shield al Arduino:*

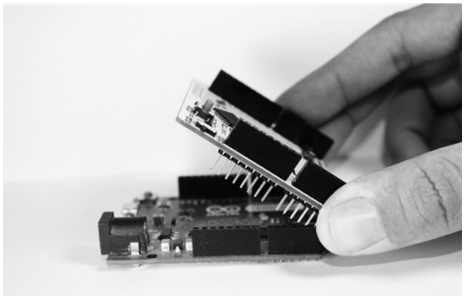


**Figura 12.31** Shield y Arduino

### *Para un Arduino Mega:*

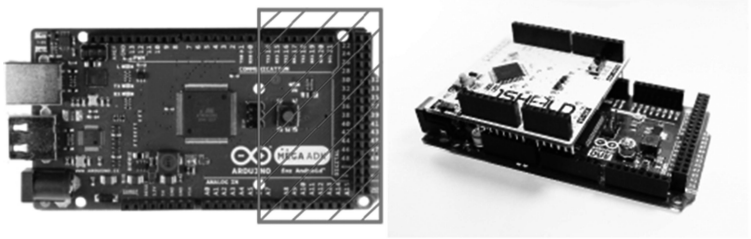


**Figura 12.32** Conexión a un Arduino Mega

### *Conexión a la computadora:*


**Figura 12.33** Conexión a computadora portátil

Descargar aplicación Android:


**Figura 12.34** Íconos de descarga

Descargar la librería para Arduino en: *https://1sheeld.com/downloads/*

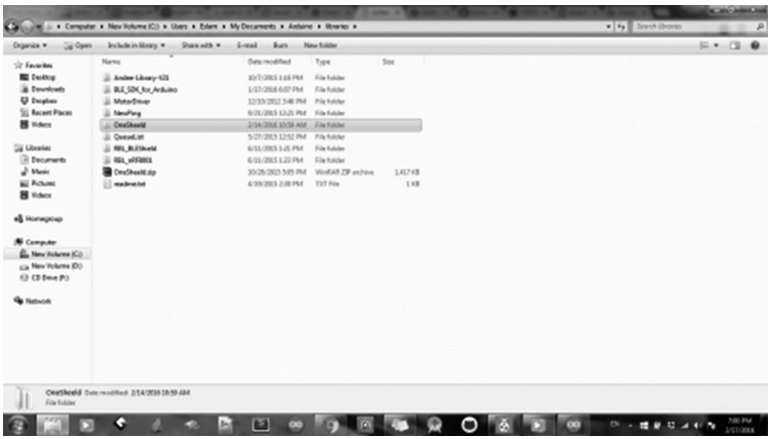


**Figura 12.35** Librería Arduino

Descargar el ejemplo de prueba:

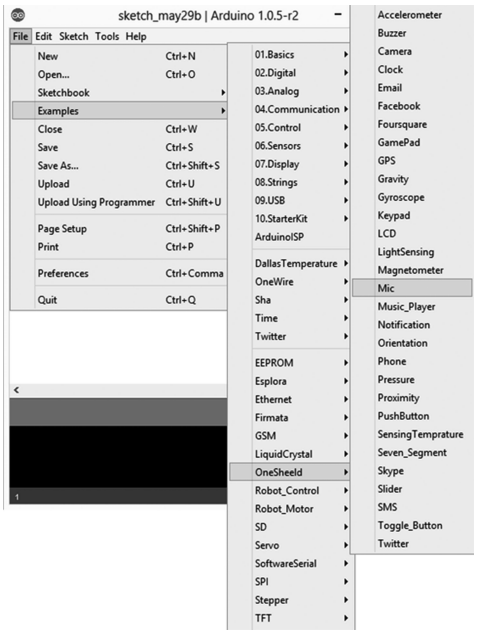



**Figura 12.36** Imagen de la pantalla de prueba

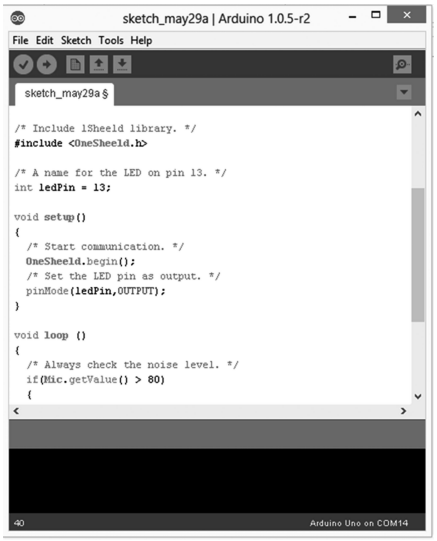



**Figura 12.37** Ejemplo de un Sketch base

### *Descarga del sketch*

Aquí viene la parte más importante de esta guía; es preciso cambiar 1Sheeld al modo Uploading —este es el interruptor etiquetado UART Switch en la placa— antes de subir el boceto a la placa Arduino para evitar conflictos seriales entre 1Sheeld y Arduino; luego, dar clic en el botón Cargar en el IDE.

El modo de carga se activa cuando el interruptor UART se aleja del logotipo de 1Sheeld, o si se tiene 1Sheeld + se debe cambiar al signo SWS "Software serial" que aparece en la placa.

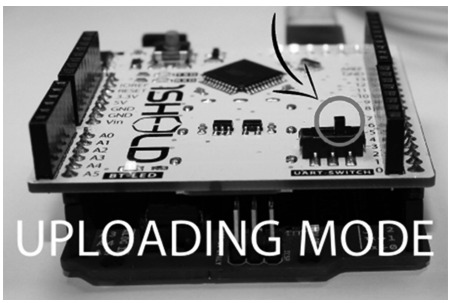



**Figura 12.38** Activación de modo de carga

### *Modo de operación*

No se debe olvidar que si no se cambia el switch UART de nuevo al modo operativo, el proyecto no funcionará correctamente pues no tendrá comunicación entre 1Sheeld y la placa Arduino.

El modo de funcionamiento se enciende cuando el interruptor UART se empuja más cerca del logotipo 1Sheeld, o si se tiene 1Sheeld + cambiar al signo "Hardware Serial" de HWS que aparece en el tablero.

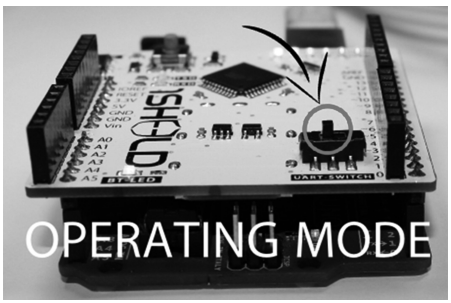



**Figura 12.39** Activación de modo de funcionamiento

### *Uso de la aplicación Android para 1Sheeld*

Abrir la aplicación 1Sheeld en el teléfono inteligente iOS o Android. La aplicación escaneará primero a través de bluetooth para el 1Sheeld; esto llevará unos segundos y el teléfono lo encontrará. Una vez que aparezca en la pantalla como 1Sheeld #xxxx, se le pedirá que ingrese el código de sincronización —el predeterminado es 1234— y se conectará a 1Sheeld a través de bluetooth.

**Nota**: Si hay problemas, es preciso asegurarse de que el Bluetooth esté encendido para el teléfono Android y que éste se encuentre cerca del 1Sheeld.


**Figura 12.40** Aplicación One shield

Si se recibe un mensaje de error como se muestra a continuación, el teléfono no puede encontrar 1Sheeld; entonces presionar el botón de reinicio en 1Sheeld e intentar de nuevo.

**Nota:** Es necesario asegurarse de que el interruptor UART está en el modo operativo.

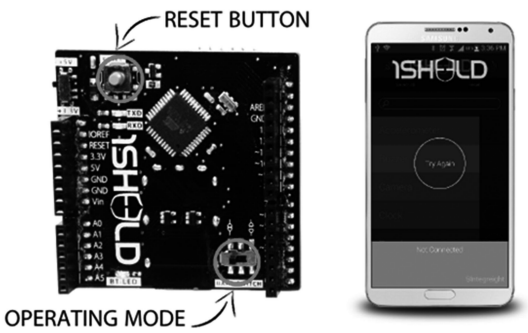



**Figura 12.41** Reinicio de 1Sheeld

### *Acceso a los shields*

Seleccionar los shields que se desea usar en el sketch Arduino (proyecto) y presionar el ícono de Shields Múltiples en la parte superior derecha de la aplicación.

En este caso, usar el escudo de Mic (Micrófono) y ya se pueden crear prototipos.


**Figura 12.42** Pantalla que indica la creación de prototipos

### *Algunos shields y su aplicación*

#### *NFC shield*

NFC Shield ofrece la potencia de las etiquetas NFC para proyectos, por lo que ahora con una pequeña etiqueta se pueden controlar las cerraduras de la puerta de casa sin llevar las llaves, o también luces, objetos físicos o cualquier cosa conectada a Arduino a través del módulo NFC en el teléfono inteligente.

Éste es uno de los escudos de comunicación, los cuales permiten usar el teléfono inteligente como una puerta de enlace entre Arduino y otros teléfonos; hacen que sea más fácil comunicarse con el tablero Arduino al enviar y recibir datos hacia o desde otro teléfono.


**Figura 12.43** Escudo NFC shield

**Ejemplo en Arduino:**

```
#define CUSTOM_SETTINGS
#define INCLUDE_NFC_SHIELD
#define INCLUDE_TERMINAL_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

void setup( )
{
  /* Start Communication. */
  OneSheeld.begin( );
}

void loop( )
{
  /* Check if there's a new tag scanned. */
  if(NFC.isNewTagScanned( ))
  {
    /* Get that last scanned tag. */
    NFCTag & myTag = NFC.getLastTag( );
    /* Print all the data about the scanned tag. */
    Terminal.print("Number of Records in Tag = ");
    /* Print number of records. */
    Terminal.println(myTag.getNumberOfRecords( ));
    /* Print tag used size. */
    Terminal.print("Used Tag Size = ");
    Terminal.println(myTag.getSize( ));
    /* Print tag maximum size. */
    Terminal.print("Maximum Tag Size = ");
    Terminal.println(myTag.getMaxSize( ));
    Terminal.print("Tag id length = ");
    Terminal.println(myTag.getIdLength( ));
    switch(myTag.getRecord(0).getTypeCategory( ))
    {
      case TNF_UNKNOWN : Terminal.println("TNF_UNKNOWN");break;
      case TNF_EMPTY : Terminal.println("TNF_EMPTY");break;
      case TNF_EXTERNAL_TYPE : Terminal.println("TNF_EXTERNAL_TYPE");break;
      case TNF_MIME_MEDIA : Terminal.println("TNF_MIME_MEDIA");break;
      case TNF_UNCHANGED : Terminal.println("TNF_UNCHANGED");break;
      case TNF_ABSOLUTE_URI : Terminal.println("TNF_ABSOLUTE_URI");break;
      case RTD_TEXT : Terminal.println("RTD_TEXT");break;
      case RTD_URI : Terminal.println("RTD_URI");break;
```

```
        case RTD_UNSUPPORTED : Terminal.println("RTD_UNSUPPORTED");break;
      }
    }
  }
```

### *GPS shield*

Obtener la ubicación de un objeto en movimiento mediante las coordenadas longitud y latitud de manera fácil desde el teléfono, usando dos líneas de código en el boceto; se puede tomar una determinada acción.

Éste es uno de los escudos del sensor, los cuales permiten acceder a los sensores del teléfono inteligente y usarlos en la creación de prototipos con Arduino; se puede obtener información del entorno y uso.


**Figura 12.44** Escudo GPS shield

**Ejemplo en Arduino:**

```
#define CUSTOM_SETTINGS
#define INCLUDE_GPS_SHIELD
#define INCLUDE_SMS_SHIELD

/* Include 1Sheeld library.*/
#include <OneSheeld.h>

/* Define a boolean flag. */
boolean isInRange = false;
```

```
void setup( )
{
  /* Start communication.*/
  OneSheeld.begin( );
}

void loop( )
{
   /* Always check if the smartphone's GPS and a given latitude and
longitude are in a range of 100 meters. */
  if(GPS.isInRange(30.0831008,31.3242943,100))
  {
    if(!isInRange)
    {
      /* Send SMS. */
      SMS.send("+1234567890","Smartphone is In Range.");
      isInRange = true;
    }

  }
  else
  {
    if(isInRange)
    {
      /* Send SMS. */
      SMS.send("1234567890","Smartphone is not In Range.");
      isInRange = false;
    }

  }
}
```

Cámara shield Permite a la placa Arduino capturar ciertos momentos utilizando la cámara del teléfono inteligente.

Los shields especiales permiten a la placa Arduino utilizar algunas capacidades que el teléfono inteligente puede hacer, como reproducir música, tomar una foto, recibir notificaciones, acceder a los datos del reloj y usar la pantalla táctil para varias funciones.

**Ejemplo en Arduino:**


**Figura 12.45** Cámara shield

```
#define CUSTOM_SETTINGS
#define INCLUDE_CAMERA_SHIELD
#define INCLUDE_TWITTER_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* A name for the button on pin 12. */
int buttonPin = 12;
/* A name for the LED on pin 13. */
int ledPin = 13;

void setup( )
{
  /* Start communication. */
  OneSheeld.begin( );
  /* Set the button pin as input. */
  pinMode(buttonPin,INPUT);
  /* Set the LED pin as output. */
  pinMode(ledPin,OUTPUT);
}
void loop( )
{
  /* Always check the button state. */
  if(digitalRead(buttonPin) == HIGH)
  {
```

```
    /* Turn on the LED. */
    digitalWrite(ledPin,HIGH);
    /* Turn on the camera flash. */
    Camera.setFlash(ON);
    /* Take the picture. */
    Camera.rearCapture( );
    /* Wait for 10 seconds. */
    OneSheeld.delay(10000);
    /* Post the picture on Twitter. */
    Twitter.tweetLastPicture("Posted by @1Sheeld and @Arduino");
  }
  else
  {
    /* Turn off the LED. */
    digitalWrite(ledPin,LOW);
  }

}
```

### *Voice recognition shield*

Arduino Voice Recognition Shield brinda la capacidad de controlar el Arduino utilizando comandos de voz con sólo una línea de código. Ahora se pueden controlar las luces por voz; cuenta con domótica controlada por la misma vía o incluso se puede manejar un robot.

**Ejemplo en Arduino:**


**Figura 12.46** Reconocimiento de voz

```
/*
This is 1Sheeld's Arduino Voice Recognition shield Tutorial to make a voice
controlled lights project.
So turn on light with just saying "on" and turn it off when saying "off"
*/

//Call the custom settings to optimize the flash memory usage
#define CUSTOM_SETTINGS
//Calling for Arduino Voice Recognize Shield
#define INCLUDE_VOICE_RECOGNIZER_SHIELD

//Including 1Sheeld library
#include <OneSheeld.h>

//Set commands to be recognized by the Arduino Voice Recognition Shield
const char on[ ] = "on";
const char off[ ] = "off";

//Set the lamp to pin 13
int lamp = 13;

void setup ( )
{
  //Initialize the communication between 1Sheeld's Arduino Voice Recognition
Shield and Arduino
  OneSheeld.begin( );
  //Set the lamp to be an Output
  pinMode(lamp,OUTPUT);
}
void loop ( )
{
  //check if 1Sheeld's Arduino Voice Recognition Shield received a new com-
mand
  if(VoiceRecognition.isNewCommandReceived( ))
  {
    //Compare the last command received by the Arduino Voice Recognition
Shield with the command "on"
    if(!strcmp(on,VoiceRecognition.getLastCommand( )))
    {
      //Then turn the light on
      digitalWrite(lamp,HIGH);
    }
    //Compare the last command received by the Arduino Voice Recognition
Shield with the command "off"
```

```
    else if(!strcmp(off,VoiceRecognition.getLastCommand( )))
    {
      //Then turn the light off
      digitalWrite(lamp,LOW);
    }
  }
}
```

### *Pressure shield*

Obtiene valores de presión del sensor de presión en el teléfono inteligente para usarlo en el boceto.

Éste es uno de los shield del sensor, los cuales permiten acceder a los sensores del teléfono inteligente y usarlos en la creación de prototipos con Arduino; se puede obtener información del entorno y usarla para desencadenar una determinada acción.

**Ejemplo en Arduino:**


**Figura 12.47** Pressure shield

```
#define CUSTOM_SETTINGS
#define INCLUDE_PRESSURE_SENSOR_SHIELD
#define INCLUDE_SMS_SHIELD


/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* Define a boolean flag. */
```

```
boolean isMessageSent = false;

void setup( )
{
  /* Start communication. */
  OneSheeld.begin( );
}

void loop( )
{
  /* Always read the pressure value and check if it exceeds a certain value.
*/
  if (PressureSensor.getValue( ) > 1008)
  {
    /* Check that we haven't sent the SMS already. */
    if (!isMessageSent)
    {
      /* Send the SMS. */
      SMS.send("1234567890","Pressure is getting high in here!");
      /* Set the flag. */
      isMessageSent = true;
    }
  }
  else
  {
    /* Reset the flag. */
    isMessageSent = false;
  }
}
```

### *Arduino IoT shield*

Convierte el teléfono inteligente en un cliente MQTT para placas Arduino. Abre muchas aplicaciones conectadas a Internet en las placas Arduino, lo cual permite actualizar y obtener datos a través de la nube preferida —Broker—.

**Ejemplo en Arduino:**


**Figura 12.48** Arduino IoT shield

```
#define CUSTOM_SETTINGS
#define INCLUDE_IOT_SHIELD
#define INCLUDE_TERMINAL_SHIELD

/* Set your host name */
#define HOST_NAME “test.mosquitto.org”

/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* Set an LED on pin 13. */
int ledPin = 13;
/* Subscribe to topic 1Sheeld/MyArduino/led . */
const char * myTopic = “1Sheeld/MyArduino/led”;

void setup( )
{
  /* Start communication. */
  OneSheeld.begin( );
  /* Disconnect from broker. */
  IOT.disconnect( );
  /* Reset all connection variables to default */
  IOT.resetConnectionParametersToDefaults( );
  /* Connect to mosquitto’s public broker. */
  IOT.connect(HOST_NAME);
  /* Subscribe to new messages. */
  IOT.setOnNewMessage(&newMessage);
  /* Subscribe to connnection status callback. */
```

```
  IOT.setOnConnectionStatusChange(&connectionStatus);
  /* Subscribe to error callback. */
  IOT.setOnError(&error);
  /* Some time for app to connect. */
  delay(3000);
  /* Subscribe to led topic. */
  IOT.subscribe(myTopic);
  /* LED pin mode output. */
  pinMode(ledPin,OUTPUT);
}

void loop( )
{}

void newMessage(char * incomingTopic, char * payload, byte qos, bool
retained)
{
  /* Check on incomingTopic. */
  if(!strcmp(myTopic,incomingTopic))
  {
    /* If payload states ON. */
    if(!strcmp("ON",payload))
    {
      /* Turn on the led. */
      digitalWrite(ledPin,HIGH);
    }
    /* If payload states OFF. */
    else if(!strcmp("OFF",payload))
    {
      /* Turn off the led. */
      digitalWrite(ledPin,LOW);
    }
  }
}

void connectionStatus(byte statusCode)
{
  /* Check connection code and display. */
  switch(statusCode)
  {
    case CONNECTION_SUCCESSFUL: Terminal.println("CONNECTION_
SUCCESSFUL");break;
    case CONNECTION_FAILED: Terminal.println("CONNECTION_FAILED");break;
    case CONNECTION_LOST: Terminal.println("CONNECTION_LOST");break;
```

```
    case CONNECTION_LOST_RECONNECTING: Terminal.println("CONNECTION_LOST_
RECONNECTING");break;
    case NOT_CONNECTED_YET:  Terminal.println("NOT_CONNECTED_YET");break;
    case MISSING_HOST: Terminal.println("MISSING_HOST");break;
  }
}

void error(byte errorCode)
{
  /* Check error code and display. */
  switch(errorCode)
  {

    case CONNECTION_REFUSED                    : Terminal.println("CONNECTION_
REFUSED");break;
    case ILLEGAL_MESSAGE_RECEIVED              : Terminal.println("ILLEGAL_
MESSAGE_RECEIVED");break;
    case DROPPING_OUT_GOING_MESSAGE            : Terminal.println("DROPPING_
OUT_GOING_MESSAGE");break;
    case ENCODER_NOT_READY                     : Terminal.println("ENCODER_
NOT_READY");break;
    case INVALID_CONNACK_RECEIVED              : Terminal.println("INVALID_
CONNACK_RECEIVED");break;
    case NO_CONNACK_RECEIVED                   : Terminal.println("NO_CONNACK_
RECEIVED");break;
    case CONNACK_UNACCEPTABLEP_ROTOCOLVERSION  : Terminal.println("CONNACK_
UNACCEPTABLEP_ROTOCOLVERSION");break;
    case CONNACK_IDENTIFIER_REJECTED           : Terminal.println("CONNACK_
IDENTIFIER_REJECTED");break;
    case CONNACK_SERVER_UNAVAILABLE            : Terminal.println("CONNACK_
SERVER_UNAVAILABLE");break;
    case CONNACK_AUTHENTICATION_FAILED         : Terminal.println("CONNACK_
AUTHENTICATION_FAILED");break;
    case CONNACK_NOT_AUTHORIZED                : Terminal.println("CONNACK_
NOT_AUTHORIZED");break;
    case CONNACK_RESERVED                      : Terminal.println("CONNACK_
RESERVED");break;
  }
}
```

### *Chart shield*

Convierte el teléfono inteligente en un protector de trazador de gráficos para Arduino. Chart shield permite al Arduino trazar valores —sensores, variables, etc.— y mostrarlos a lo largo del tiempo.


**Figura 12.49** Chart shield

**Ejemplo en Arduino:**

```
#define CUSTOM_SETTINGS
#define INCLUDE_MIC_SHIELD
#define INCLUDE_CHART_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

void setup( )
{
  /* Start communication. */
  OneSheeld.begin( );
  /* Save a screenshot for chart 0. */
  Chart.saveScreenshot(CHART_0);
  /* Save a csv file for chart 0. */
  Chart.saveCsv("MicValues",CHART_0);
  /* Clear Chart 0. */
  Chart.clear(CHART_0);
}

void loop( ) {
  /* Add mic values to be ploted over chart. */
```

```
  Chart.add("Mic/db",Mic.getValue( ));
  /* Plot the values. */
  Chart.plot( );
  /* Delay for 1 second. */
  OneSheeld.delay(1000);
}
```

### *Data logger shield*

Usar la memoria del teléfono inteligente para registrar los datos; por ejemplo, un sensor ultrasónico. Con Arduino, luego se pueden exportar los datos en formato CSV.


**Figura 12.50** Data logger shield

**Ejemplo en Arduino:**

```
#define CUSTOM_SETTINGS
#define INCLUDE_DATA_LOGGER_SHIELD
#define INCLUDE_MIC_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* Reserve a counter. */
int counter = 0;
/* Button on pin 12. */
int buttonPin = 12;
/* Boolean to start logging. */
bool startFlag = false;
/* Float to store sensor value in. */
```

```
float sensorvalue = 0;

void setup( ) {
    /* Start communication. */
  OneSheeld.begin( );
  /* Save any previous logged values. */
  Logger.stop( );
  /* Set buttonPin as input. */
  pinMode(buttonPin,INPUT);
}

void loop( )
{
  /* Check if button pressed. */
  if(digitalRead(buttonPin) == HIGH)
  {
    /* First insure to save previous logged values. */
    Logger.stop( );
    /* Set a delay. */
    OneSheeld.delay(500);
    /* Start logging in a new CSV file. */
    Logger.start("Mic values");
    /* Set startFlag. */
    startFlag = true;
  }

  /* Check logging started. */
  if(startFlag)
  {
    /* Add noise level values as a column in the CSV file. */
    Logger.add("Decibles",Mic.getValue( ));
    /* Store the sensor value in the sensor value variable. */
    sensorvalue = (analogRead(A0) * 5.00 *100.00 / 1024.00);
    /* Add sensor values as a column in the CSV file. */
    Logger.add("Temprature",sensorvalue);
    /* Log the row in the file. */
    Logger.log( );
    /* Delay for 1 second. */
    OneSheeld.delay(1000);
    /* Increment counter. */
    counter++;
    /* Stop logging after 20 readings and save the CSV file. */
    if(counter==10)
    {
```

```
        /* Save the logging CSV file. */
        Logger.stop( );
        /* Reset counter. */
        counter=0;
        /* Start Logging again. */
        Logger.start("Mic values");
      }
    }
}
```

### *Keyboard shield*

Usar la pantalla táctil del teléfono inteligente como un teclado. Este escudo brinda la ventaja de enviar caracteres ASCII al Arduino y desencadenar ciertas acciones escritas en el boceto.


**Figura 12.51** Keyboard shield

**Ejemplo en Arduino:**

```
#define CUSTOM_SETTINGS
#define INCLUDE_KEYBOARD_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* LED on pin 13. */
int ledPin = 13;

void setup( )
{
```

```
  /* Start communication. */
  OneSheeld.begin( );
  /* Set the LED as output. */
  pinMode(ledPin,OUTPUT);
  /* Keyboard callBack function. */
  AsciiKeyboard.setOnButtonChange(&keyboardFunction);
}

void loop( )
{}

/* Function to be invoked once a new character is pressed. */
void  keyboardFunction(char data)
{
  /* Check on the incoming character. */
  if(data == 'A')
  {
    /* Turn on the LED. */
    digitalWrite(ledPin,HIGH);
  }
  else
  {
    /* Turn off the LED.*/
    digitalWrite(ledPin,LOW);
  }
}
```

### *Pattern shield*

El patrón de protección convierte el teléfono inteligente en un protector de armario de patrón secreto, esto para que Arduino pueda tomar medidas según ciertos patrones definidos por el usuario.

Éste es uno de los escudos especiales, los cuales permiten al tablero Arduino usar algunas capacidades que el teléfono inteligente puede hacer, como reproducir música, tomar una foto, recibir notificaciones, acceder a los datos del reloj y usar la pantalla táctil para varias funciones.

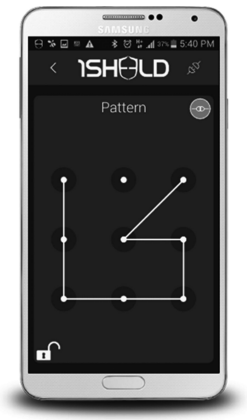



**Figura 12.52** Pattern shield

**Ejemplo en Arduino:**

```
#define CUSTOM_SETTINGS
#define INCLUDE_PATTERN_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* A name for the LED on pin 13. */
int ledPin = 13;

/* A variable to store the right pattern which is “L” character. */
/* The first number is the row, and the second is the column of the
PatternNode. */

PatternNode patternStored[ ] = { {0,0}, {1,0}, {2,0}, {2,1}, {2,2} };

/* A variable used to check wether the pattern is right or wrong. */
int counter = 0;

/* A variable used to store the length of the pattern. */
int length = 0;

void setup( )
{
  /* Start Communication. */
  OneSheeld.begin( );
  /* Set the LED as output. */
```

```
  pinMode(ledPin,OUTPUT);
}

void loop( )
{
  /* Always check if the user entered any pattern. */
  if (Pattern.isNewPatternReceived( ))
  {
    /* This pointer act as array to store the last pattern entered by user.
*/
    PatternNode * patternEntered = Pattern.getLastPattern( );

    /* Get the length of the pattern to loop over it. */
    length = Pattern.getLastPatternLength( );

    /* Check if the pattern length is equal to 5. */
    if (length == 5)
      {
        /* A for loop to check the entered pattern to the stored pattern. */
        for (int i = 0; i < length ; i++)
        {
          /* A check for the pattern by checking the row and the column of
each node entered by the user. */
          if (patternEntered[i].row == patternStored[i].row &&
patternEntered[i].col == patternStored[i].col)
            {
              counter++;
            }
        }

        /* Check if all the nodes entered by the user are right and equal to
the correct pattern. */
        if (counter == 5)
          {
            digitalWrite(ledPin,HIGH);
            counter = 0;
          }
        else
          {
            digitalWrite(ledPin,LOW);
            counter = 0;
          }
      }
    /* If pattern length is different turn off the LED.  */
```

```
    else
      {
        digitalWrite(ledPin,LOW);
      }
  }
}
```

## Proyecto: Control de acceso por detección de sonido y keypad

### *Arduino shield keypad*

En este proyecto se utiliza uno de los shields virtuales en el mismo dispositivo Android, el cual sirve para capturar los códigos de acceso y el shield del micrófono para detectar los toquidos.


**Figura 12.53** Arduino shield keypad

**Prueba del Keypad**

```
#define CUSTOM_SETTINGS
#define INCLUDE_KEYPAD_SHIELD

/* Include 1Sheeld library. */
#include <OneSheeld.h>

/* A name for the LED on pin 13. */
int ledPin1 = 13;
/* A name for the LED on pin 12. */
int ledPin2 = 12;
/* A name for the LED on pin 11. */
```

```
int ledPin3 = 11;
/* A name for the LED on pin 10. */
int ledPin4 = 10;

void setup( )
{
  /* Start communication. */
  OneSheeld.begin( );
  /* Set LEDs 1, 2, 3 and 4 as output. */
  pinMode(ledPin1,OUTPUT);
  pinMode(ledPin2,OUTPUT);
  pinMode(ledPin3,OUTPUT);
  pinMode(ledPin4,OUTPUT);
}

void loop( )
{
  /* If keypad's button 1 is pressed. */
  if(Keypad.isRowPressed(0) && Keypad.isColumnPressed(0))
  {
    /* Turn on the LED 1. */
    digitalWrite(ledPin1,HIGH);
    /* Turn off the other LEDs. */
    digitalWrite(ledPin2,LOW);
    digitalWrite(ledPin3,LOW);
    digitalWrite(ledPin4,LOW);
  }
  /* If keypad's button 2 is pressed. */
  else if(Keypad.isRowPressed(0) && Keypad.isColumnPressed(1))
  {
    /* Turn on the LED 2. */
    digitalWrite(ledPin2,HIGH);
    /* Turn off the other LEDs. */
    digitalWrite(ledPin1,LOW);
    digitalWrite(ledPin3,LOW);
    digitalWrite(ledPin4,LOW);
  }
  /* If keypad's button 3 is pressed. */
   else if(Keypad.isRowPressed(0) && Keypad.isColumnPressed(2))
  {
    /* Turn on the LED 3. */
    digitalWrite(ledPin3,HIGH);
    /* Turn off the other LEDs. */
    digitalWrite(ledPin1,LOW);
```

```
      digitalWrite(ledPin2,LOW);
      digitalWrite(ledPin4,LOW);
    }
    /* If keypad's button 4 is pressed. */
     else if(Keypad.isRowPressed(1) && Keypad.isColumnPressed(0))
    {
      /* Turn on the LED 4. */
      digitalWrite(ledPin4,HIGH);
      /* Turn off the other LEDs. */
      digitalWrite(ledPin1,LOW);
      digitalWrite(ledPin2,LOW);
      digitalWrite(ledPin3,LOW);
    }
    else
    {
      /* Turn off all of LEDs. */
      digitalWrite(ledPin1,LOW);
      digitalWrite(ledPin2,LOW);
      digitalWrite(ledPin3,LOW);
      digitalWrite(ledPin4,LOW);
    }
  }
```

### *Conexiones del proyecto*

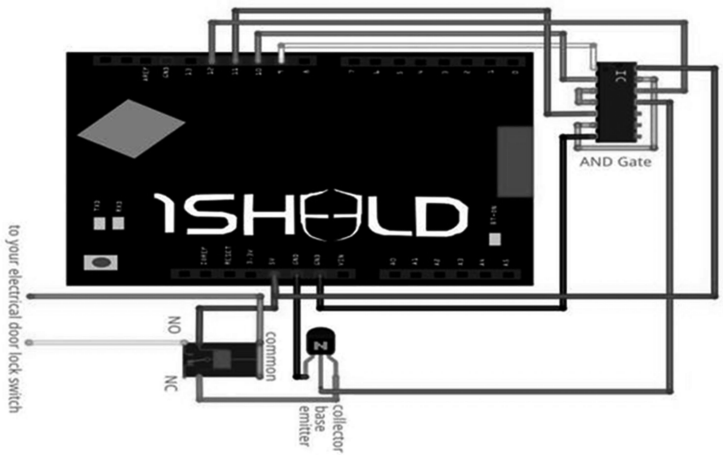



**Figura 12.54** Muestra de las conexiones

Para realizar las pruebas se utilizan tres botones en el keypad (A, B y C):

- Presionar la tecla A en el keypad para grabar el toque especial
- Presionar el botón B, dejarlo presionado mientras se toca la puerta
- Finalmente, presionar el botón C para abrir la puerta

**Código del proyecto:**

```
// Basically the code hear and save the knock and compare it to any other
knock to open the door

#include <OneSheeld.h>
int pin1 = 9;
int pin2 = 10;
int pin3 = 11;
int pin4 = 12;
int led4 = 13;
int counter = 0;
int sec = 0;
int hear[20] = {};
int saved[20] = {};
int flagc = 0;
int flags = 0;
int flagf = 0;
int i = 0;
int ii = 0;

void setup( )
{
  OneSheeld.begin( );
  pinMode(pin1,OUTPUT);
  pinMode(pin2,OUTPUT);
  pinMode(pin3,OUTPUT);
  pinMode(pin4,OUTPUT);
  pinMode(led4,OUTPUT);
}
void loop( )
{
  if (Keypad.isRowPressed(0) && Keypad.isColumnPressed(3))
  {
    set( );
  }
  else if (Keypad.isRowPressed(1) && Keypad.isColumnPressed(3))
  {
```

```
    record( );
  }
  else if (Keypad.isRowPressed(2) && Keypad.isColumnPressed(3))
  {
    check( );
  }
  else {
    flagf = 0;
    digitalWrite(pin1,LOW);
    digitalWrite(pin2,LOW);
    digitalWrite(pin3,LOW);
    digitalWrite(pin4,LOW);
    digitalWrite(led4,LOW);
  }
}

void clc ( )
{
  counter = 0;
  for (int i=0; i<=20;i++)
  {
    hear[i] = 0;
    saved[i] = 0;
  }
  i = 0;
  ii = 0;
  digitalWrite(led4,LOW);
}

void set( )
{
  if (flagf == 0)
  {
  clc( );
  }
  if(Mic.getValue( ) > 80 && flags == 0)
  {
    digitalWrite(led4,HIGH);
    saved[ii] = millis( ) - sec;
    sec= millis( );
    ii++;
    flags = 1;
  }
  if (Mic.getValue( ) < 80)
```

```
  {
    digitalWrite(led4,LOW);
    flags = 0;
  }
  flagf = 1;
}

void record ( )
{

  if(Mic.getValue( ) > 80 && flagc == 0)
  {
    digitalWrite(led4,HIGH);
    hear[i] = millis( ) - sec;
    sec= millis( );
    i++;
    flagc = 1;
  }
  if (Mic.getValue( ) < 80)
  {
    digitalWrite(led4,LOW);
    flagc = 0;
  }
}
void check ( )
{ if (i == ii)
  {
    for (int j=1;j<=i;j++)
    {
      int x = hear[j];
      int y =saved[j];
      if (x+400 >= y && x-400 <=y )
      {
        counter++;
      }
    }
    if (counter == i)
    {
      digitalWrite(pin1,HIGH);
      digitalWrite(pin2,HIGH);
      digitalWrite(pin3,HIGH);
      digitalWrite(pin4,HIGH);
    }
  }
```

```
    counter = 0;
    for (int i=0; i<=20;i++)
    {
      hear[i] = 0;
    }
    i = 0;
  }
```

**Prueba de la aplicación:**

**Figura 12.55** Muestra de la prueba

## RESUMEN

En este capítulo se realizó una serie de proyectos que se enfocaron en resolver problemas del mundo actual; se utilizaron las herramientas de Android y Arduino, al hacer una integración entre sí de ambas plataformas. Estos proyectos son para aplicaciones de uso cotidiano a través de medios de comunicación como bluetooth, Internet de las Cosas, conectividad por Wifi. En el siguiente capítulo, se verán algunos proyectos enfocados a la integración de todas las tecnologías vistas en los apartados anteriores.

Capítulo 13

# Proyectos de aplicación ANDROID

## Tercera parte

### ☼ Contenido



### ☑ Objetivo

En este capítulo se verá la última parte de proyectos Android, donde se integrarán las herramientas Android con Arduino; se mencionarán algunas tecnologías que marcarán el futuro de las aplicaciones móviles con el entorno de desarrollo, dirigidas hacia los diferentes tipos de aplicaciones.

## Proyecto: Robot móvil por Bluetooth Low Energy

En este proyecto se realizará el control de un robot móvil por BLE (Bluetooth Low Energy) desde una aplicación en Android. El robot también tendrá un chip BLE para que pueda recibir comandos desde la aplicación de Android. La aplicación tendrá los siguientes comandos básicos, necesarios para controlar el robot:

- Adelante
- Retroceder
- Girar a la izquierda
- Doblar a la derecha
- Mostrar el estado de la conexión del robot

### *Requerimientos de software y hardware*

La base de este proyecto es, por supuesto, el robot en sí. Se requiere un chasis de robot de DFRobot miniQ de dos ruedas, que viene con un chasis de robot redondo, dos motores de CD, dos ruedas y algunos tornillos y pernos para que pueda montar múltiples tarjetas de Arduino.

Básicamente, se puede usar cualquier chasis de robot equivalente con dos ruedas acopladas y con motores de CD en las que puede montar tarjetas compatibles con Arduino.

Para controlar el robot, se utilizarán tres tarjetas Arduino diferentes. El "cerebro" del robot será una simple placa Arduino Uno. Además de eso, se usará un shield de motores DFRobot para controlar los dos motores de CD del robot. Encima de estas dos tarjetas se pondrá un shield de prototipos, con ello se podrán conectar diferentes módulos para el robot. Para controlarlo de forma remota, nuevamente se usará el módulo bluetooth BLE; una placa de Adafruit nRF8001 para darle la capacidad de detectar lo que está delante de él, y además un sensor ultrasónico URM37.

Finalmente, también se requieren cables macho-macho para hacer las diferentes conexiones entre el robot, el sensor y el módulo bluetooth. La siguiente es una lista de todo el hardware necesario para este proyecto, junto con enlaces a las partes en la web:

- Una placa Arduino Uno (*http://www.dfrobot.com/index.php?route=producto/producto&search=uno&description=true&product_id=838*)
- Shield de motores para Arduino (*http://www.dfrobot.com/index.php?route=product/product&path=35_39yproduct_id=59*)
- Un Shield de prototipos (*http://www.dfrobot.com/index.php?route=producto/productoyproduct_id=55*)
- Un módulo BLE nRF8001 (*https://www.adafruit.com/products/1697*)
- Un sensor ultrasónico (*http://www.dfrobot.com/index.php?route=producto/producto&search=ultrasónico&description=true&page=1&product_id=53*)

- Un kit de montaje de sensor ultrasónico (*http://www.dfrobot.com/index.php?route=producto/productoyproduct_id=322*)
- Un chasis DFQbot miniQ (*http://www.dfrobot.com/index.php?route=producto/producto&search =miniq&description=true&product_id=367*)
- Una batería de 9 V (*https://www.adafruit.com/product/67*)
- Cables macho-macho (*https://www.adafruit.com/products/1957*)

En la parte del software se requiere lo siguiente:

- Una librería para el chip nRF8001 (*https://github.com/adafruit/Adafruit_nRF8001*)
- La librería aREST para enviar comandos al robot (*https://github.com/marcoschwartz/aREST*)

### *Configuración del hardware*

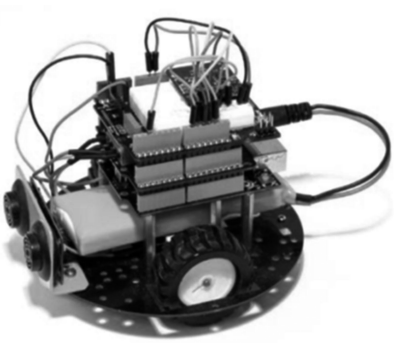


**Figura 13.1** Adaptación del hardware

### *Se muestra la parte trasera del robot:*

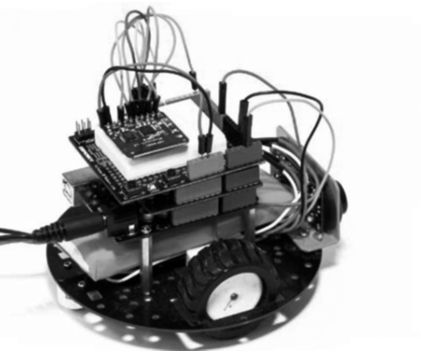


**Figura 13.2** Vista trasera

### *Módulo shield de motores de corriente directa:*

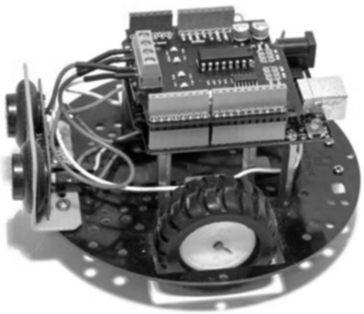


**Figura 13.3** Módulo shield

### *Conexiones entre los elementos del proyecto:*

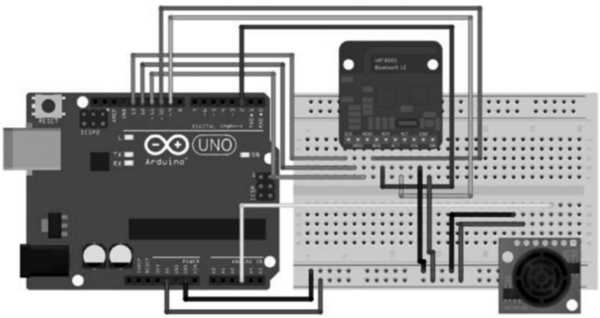



**Figura 13.4** Conexiones

### *Módulo BLE para comunicación:*

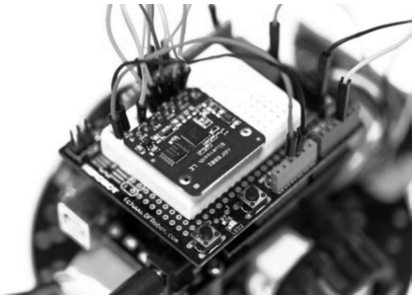


**Figura 13.5** Módulo BLE

***Prueba del sketch del robot:***

```
// Robot test via aREST + Serial

// Libraries
#include <aREST.h>

// Motor pins
int speed_motor1 = 6;
int speed_motor2 = 5;
int direction_motor1 = 7;
int direction_motor2 = 4;

// Sensor pins
int distance_sensor = A3;

// Create aREST instance
aREST rest = aREST( );

// Variable to be exposed to the API
int distance;

void setup(void)
{
  // Start Serial
  Serial.begin(115200);

  // Expose variables to REST API
  rest.variable(“distance”,&distance);

  // Expose functions
  rest.function(“forward”,forward);
  rest.function(“backward”,backward);
  rest.function(“left”,left);
  rest.function(“right”,right);
  rest.function(“stop”,stop);

  // Give name and ID to device
  rest.set_id(“001”);
  rest.set_name(“mobile_robot”);
}

void loop( ) {
```

```
  // Measure distance
  distance = measure_distance(distance_sensor);

  // Handle REST calls
  rest.handle(Serial);

}

// Forward
int forward(String command) {

  send_motor_command(speed_motor1,direction_motor1,100,1);
  send_motor_command(speed_motor2,direction_motor2,100,1);
  return 1;
}

// Backward
int backward(String command) {

  send_motor_command(speed_motor1,direction_motor1,100,0);
  send_motor_command(speed_motor2,direction_motor2,100,0);
  return 1;
}

// Left
int left(String command) {

  send_motor_command(speed_motor1,direction_motor1,75,0);
  send_motor_command(speed_motor2,direction_motor2,75,1);
  return 1;
}

// Right
int right(String command) {

  send_motor_command(speed_motor1,direction_motor1,75,1);
  send_motor_command(speed_motor2,direction_motor2,75,0);
  return 1;
}

// Stop
int stop(String command) {

  send_motor_command(speed_motor1,direction_motor1,0,1);
```

```
  send_motor_command(speed_motor2,direction_motor2,0,1);
  return 1;
}

// Function to command a given motor of the robot
void send_motor_command(int speed_pin, int direction_pin, int pwm, boolean dir)
{
  analogWrite(speed_pin,pwm); // Set PWM control, 0 for stop, and 255 for
maximum speed
  digitalWrite(direction_pin,dir);
}

// Measure distance from the ultrasonic sensor
int measure_distance(int pin){

  unsigned int Distance=0;
  unsigned long DistanceMeasured=pulseIn(pin,LOW);

  if(DistanceMeasured==50000){              // the reading is invalid.
      Serial.print("Invalid");
   }
    else{
      Distance=DistanceMeasured/50;      // every 50us low level stands for 1cm
   }

  return Distance;
}
```

### *Escritura del sketch completo para Arduino:*

```
// Robot test via aREST + Serial

// Libraries
#include <SPI.h>
#include "Adafruit_BLE_UART.h"
#include <aREST.h>

// BLE pins
#define ADAFRUITBLE_REQ 10
#define ADAFRUITBLE_RDY 2     // This should be an interrupt pin, on Uno
thats #2 or #3
#define ADAFRUITBLE_RST 9
```

```
// Motor pins
int speed_motor1 = 6;
int speed_motor2 = 5;
int direction_motor1 = 7;
int direction_motor2 = 4;

// Sensor pins
int distance_sensor = A3;

// BLE instance
Adafruit_BLE_UART BTLEserial = Adafruit_BLE_UART(ADAFRUITBLE_REQ,
ADAFRUITBLE_RDY, ADAFRUITBLE_RST);

// Create aREST instance
aREST rest = aREST( );

// Variable to be exposed to the API
int distance;

void setup(void)
{
  // Start Serial
  Serial.begin(115200);
  BTLEserial.begin( );

  // Expose variables to REST API
  rest.variable("distance",&distance);

  // Expose functions
  rest.function("forward",forward);
  rest.function("backward",backward);
  rest.function("left",left);
  rest.function("right",right);
  rest.function("stop",stop);

  // Give name and ID to device
  rest.set_id("001");
  rest.set_name("mobile_robot");
}

aci_evt_opcode_t laststatus = ACI_EVT_DISCONNECTED;

void loop( ) {
```

```
  // Measure distance
  distance = measure_distance(distance_sensor);

  // Tell the nRF8001 to do whatever it should be working on.
  BTLEserial.pollACI( );

  // Ask what is our current status
  aci_evt_opcode_t status = BTLEserial.getState( );
  // If the status changed....
  if (status != laststatus) {
    // print it out!
    if (status == ACI_EVT_DEVICE_STARTED) {
        Serial.println(F("* Advertising started"));
    }
    if (status == ACI_EVT_CONNECTED) {
        Serial.println(F("* Connected!"));
    }
    if (status == ACI_EVT_DISCONNECTED) {
        Serial.println(F("* Disconnected or advertising timed out"));
    }
    // OK set the last status change to this one
    laststatus = status;
  }
    // Handle REST calls
  if (status == ACI_EVT_CONNECTED) {
    rest.handle(BTLEserial);
  }

}

// Forward
int forward(String command) {

  send_motor_command(speed_motor1,direction_motor1,200,1);
  send_motor_command(speed_motor2,direction_motor2,200,1);
  return 1;
}

// Backward
int backward(String command) {

  send_motor_command(speed_motor1,direction_motor1,200,0);
  send_motor_command(speed_motor2,direction_motor2,200,0);
```

```
  return 1;
}

// Left
int left(String command) {

  send_motor_command(speed_motor1,direction_motor1,150,0);
  send_motor_command(speed_motor2,direction_motor2,150,1);
  return 1;
}

// Right
int right(String command) {

  send_motor_command(speed_motor1,direction_motor1,150,1);
  send_motor_command(speed_motor2,direction_motor2,150,0);
  return 1;
}

// Stop
int stop(String command) {

  send_motor_command(speed_motor1,direction_motor1,0,1);
  send_motor_command(speed_motor2,direction_motor2,0,1);
  return 1;
}

// Function to command a given motor of the robot
void send_motor_command(int speed_pin, int direction_pin, int pwm, boolean
dir)
{
  analogWrite(speed_pin,pwm); // Set PWM control, 0 for stop, and 255 for
maximum speed
  digitalWrite(direction_pin,dir);
}

// Measure distance from the ultrasonic sensor
int measure_distance(int pin){

  unsigned int Distance=0;
  unsigned long DistanceMeasured=pulseIn(pin,LOW);

  if(DistanceMeasured==50000){              // the reading is invalid.
      Serial.print("Invalid");
```

```
   }
    else{
      Distance=DistanceMeasured/50;      // every 50us low level stands for
1cm
   }

  return Distance;
}
```

### *Código de funciones:*

```
//User Interface Elements
Button fwdBtn;
Button leftBtn;
Button rightBtn;
Button backBtn;
Button stopBtn;
Button connectBtn;
TextView connectionSts;
//Logging Variables
private final String LOG_TAG = RobotControlActivity.class.
getSimpleName( );


fwdBtn = (Button) findViewById(R.id.fwdBtn);
leftBtn = (Button) findViewById(R.id.leftBtn);
rightBtn = (Button) findViewById(R.id.rightBtn);
backBtn = (Button) findViewById(R.id.backwardBtn);
stopBtn = (Button) findViewById(R.id.stopBtn);
connectBtn = (Button) findViewById(R.id.connectBtn);
connectionSts = (TextView)findViewById(R.
id.connectionStsView);
```

### *Acciones de los botones:*

```
fwdBtn.setOnClickListener(new View.OnClickListener( ) {
@Override
public void onClick(View view) {
String setOutputMessage = "/forward /";
tx.setValue(setOutputMessage.getBytes(Charset.
forName("UTF-8")));
if (gatt.writeCharacteristic(tx)) {
writeConnectionData("Sent: " + setOutputMessage);
} else {
```

```
writeConnectionData("Couldn't write TX
characteristic!");
}
}
});

leftBtn.setOnClickListener(new View.OnClickListener( ) {
@Override
public void onClick(View view) {
String setOutputMessage = "/left /";
tx.setValue(setOutputMessage.getBytes(Charset.
forName("UTF-8")));
if (gatt.writeCharacteristic(tx)) {
writeConnectionData("Sent: " + setOutputMessage);
} else {
writeConnectionData("Couldn't write TX
characteristic!");
}
}
});
rightBtn.setOnClickListener(new View.OnClickListener( ) {
@Override
public void onClick(View view) {
String setOutputMessage = "/right /";
tx.setValue(setOutputMessage.getBytes(Charset.
forName("UTF-8")));
if (gatt.writeCharacteristic(tx)) {
writeConnectionData("Sent: " + setOutputMessage);
} else {
writeConnectionData("Couldn't write TX
characteristic!");
}
}
});
backBtn.setOnClickListener(new View.OnClickListener( ) {
@Override
public void onClick(View view) {
String setOutputMessage = "/backward /";
tx.setValue(setOutputMessage.getBytes(Charset.
forName("UTF-8")));
if (gatt.writeCharacteristic(tx)) {
writeConnectionData("Sent: " + setOutputMessage);
} else {
writeConnectionData("Couldn't write TX
```

```
characteristic!”);
}
}
});
stopBtn.setOnClickListener(new View.OnClickListener( ) {
@Override
public void onClick(View view) {
String setOutputMessage = “/stop /”;
tx.setValue(setOutputMessage.getBytes(Charset.
forName(“UTF-8”)));
if (gatt.writeCharacteristic(tx)) {
writeConnectionData(“Sent: ” + setOutputMessage);
} else {
writeConnectionData(“Couldn’t write TX
characteristic!”);
}
}
});
connectBtn.setOnClickListener(new View.OnClickListener( ) {
@Override
public void onClick(View view) {
restartScan( );
}
});
```

### *Aplicación en Android Studio:*

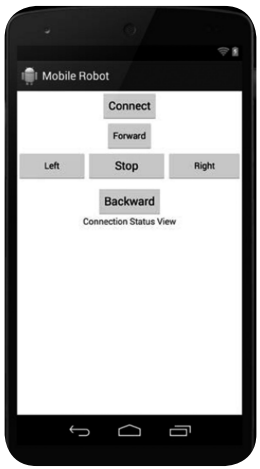



**Figura 13.6** Vista en Android

***Mejoras en la interfaz:***


**Figura 13.7** Vista en la interfaz

## 13.1 FUTURAS APLICACIONES

La aplicación de Android se puede mejorar aún más con controles refinados que podrían cuantificar el ángulo exacto por el que se desea que el robot gire a la izquierda o a la derecha. También se pueden extraer datos del sensor ultrasónico y mostrarlos dentro de la aplicación de Android para saber en qué momento detectará los obstáculos.

Además, la aplicación de Android definitivamente podría desplegar un listado de los dispositivos bluetooth disponibles y mostrarse a los usuarios para elegir el chip adecuado. Esto mejorará la experiencia del usuario y, al mismo tiempo, proporcionará una conexión más estable con el robot; especialmente si se está trabajando en un entorno rodeado de otros módulos BLE. Por último, el lector puede hacer otras modificaciones a la interfaz de usuario, en el diseño, para hacer que la aplicación sea aún más atractiva y presentable. Puede consultar el link *http://developer.android.com*.

### Proyecto: Robot Autónomo Arduino OpenCV

En este proyecto se construirá un robot con Arduino; con el uso de una cámara desde el dispositivo móvil se podrá capturar la imagen para que a través de la librería OpenCV, el robot pueda tomar decisiones con base en la detección de imágenes.

### *OpenCV*

Es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere reconocimiento de objetos. Esto se debe a que su publicación se dio bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma y existen versiones para GNU/Linux, Mac OS X y Windows. Contiene más de quinientas funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estérea y robótica. El proyecto pretende proporcionar un entorno de desarrollo fácil de utilizar y altamente eficiente. Esto se ha logrado realizando su programación en código C y C++ optimizados, además de aprovechar las capacidades que proveen los procesadores multinúcleo. OpenCV puede además utilizar el sistema de primitivas de rendimiento integradas de Intel, un conjunto de rutinas de bajo nivel específicas para procesadores Intel.

### *Plataforma Arduino Robot*

El Arduino robot es el primer Arduino oficial sobre ruedas. Tiene dos procesadores, uno en cada tablero de los dos que posee. La placa del motor controla los motores, y la placa de control lee sensores y decide cómo operar. Cada una de las placas es una Arduino completa programable que usa el Arduino IDE. Se puede localizar información en el siguiente link: *https://store.arduino.cc/usa/arduino-robot*


**Figura 13.8** Plataforma de Arduino robot

Sus especificaciones son:

**Tabla 13.1** Especificaciones de Arduino robot

| | |
|---|---|
| Microcontroller | ATmega32u4 |
| Operating Voltage | 5V |
| Input Voltage | 5V through flat cable |
| Digital I/O Pins | 5 |
| PWM Channels | 6 |
| Analog Input Channels | 4 (of the Digital I/O pins) |
| Analog Input Channels (multiplexed) | 8 |
| DC Current per I/O Pin | 40 mA |
| Flash Memory | 32 KB (ATmega32u4)<br>of which 4 KB used by bootloader |
| SRAM | 2.5 KB (ATmega32u4) |
| EEPROM (internal) | 1 KB (ATmega32u4) |
| EEPROM (external) | 512 Kbit (I2C) |
| Clock Speed | 16 MHz |
| Keypad | 5 keys |
| Knob | potentiomenter attached to analog pin |
| Full color LCD | over SPI communication |
| SD card reader | for FAT16 formatted cards |
| Speaker | 8 Ohm |
| Digital Compass | provides deviation from the<br>geographical north in degrees |
| I2C soldering ports | 3 |
| Prototyping áreas | 4 |
| Radius | 185 mm |
| Heigth | 85 mm |

**Tabla 13.2** Especificaciones del motor Board Summary

| | |
|---|---|
| Microcontroller | ATmega32u4 |
| Operating Voltage | 5V |
| Input Voltage | 9V to battery charger |

| | |
|---|---|
| AA battery slot | 4 alkaline or NiMh rechargeable batteries |
| Digital I/O Pins | 4 |
| PWM Channels | 1 |
| Analog Input Channles | 4 (same as the Digital I/O pins) |
| DC Current per I/O Pin | 40 mA |
| DC-DC converter | generates 5V to power up the whole robot |
| Flash Memory | 32 KB (ATmega32u4) of which 4 KB used by bootloader |
| SRAM | 2.5 KB (ATmega32u4) |
| EEPROM | 1 KB (ATmega32u4) |
| Clock Speed | 16 MHz |
| Trimmer | for movement calibration |
| IR line following sensors | 5 |
| I2C soldering ports | 1 |
| Prototyping areas | 2 |

## *Elementos del hardware:*

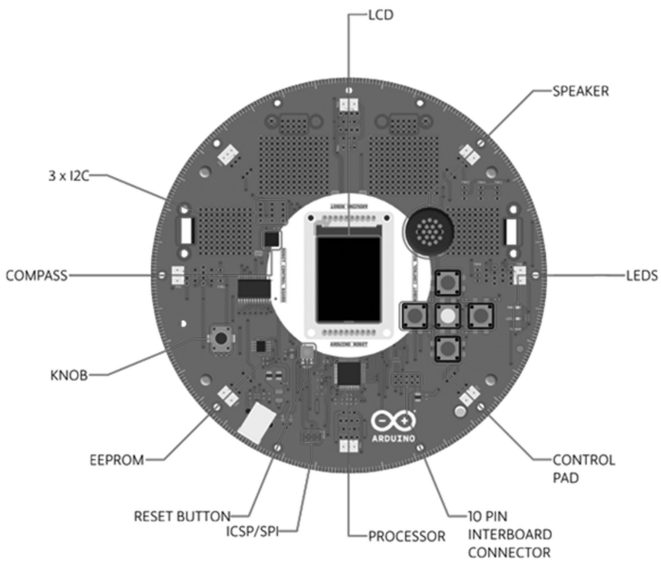



**Figura 13.9** Vista del hardware

## *Componentes del hardware*

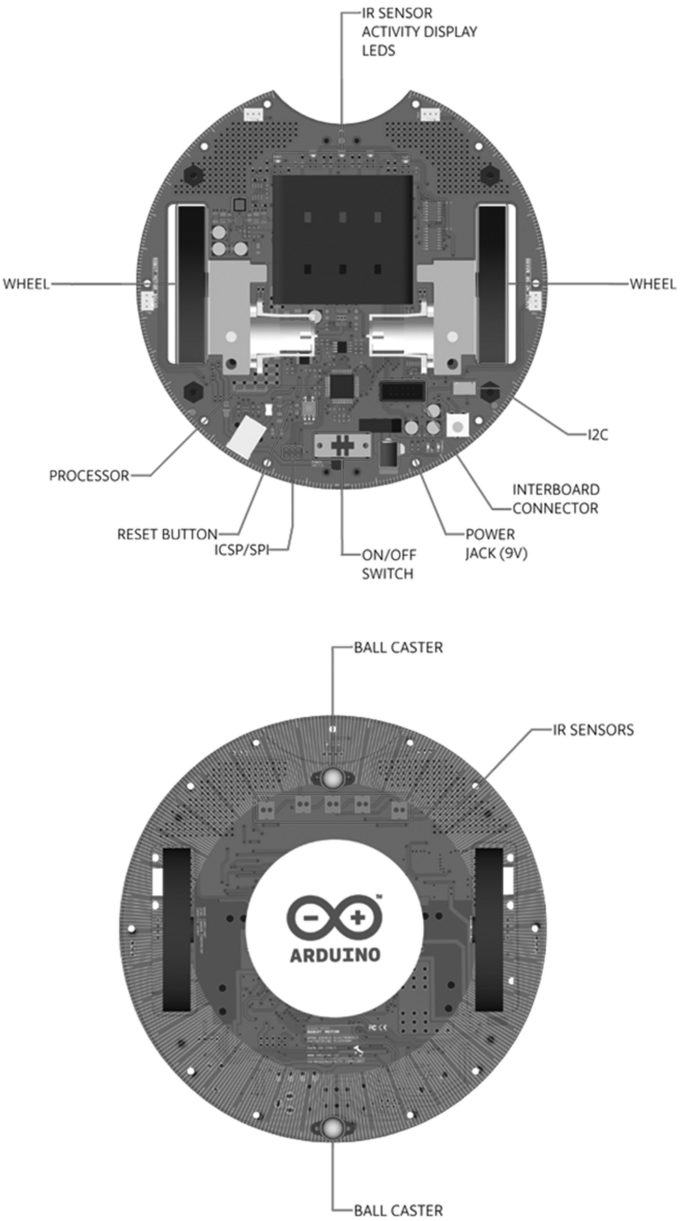



**Figura 13.10** Componentes del hardware

### *Esquema de todos los elementos*

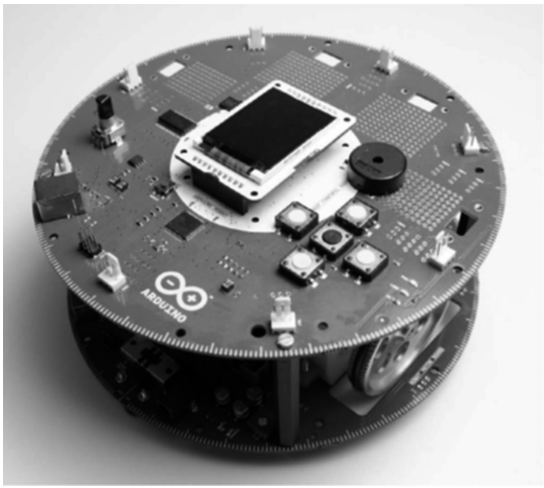


**Figura 13.11** Representación de los elementos

Representación del diseño:

- Firmware: el software que se ejecuta en el robot.
- Software (integrado): este es el software que se ejecuta en Arduino Mega ADK. Puede leer Mandatos MQTT publicados por el teléfono y extraer la carga útil para usarla en el robot.
- Aplicación: la aplicación ejecuta los algoritmos de visión por computadora en el dispositivo Android y envía comandos para Arduino Mega ADK.

### *Características del proyecto:*

- Crear un robot capaz de girar hacia un objeto, en este caso una pelota de tenis.
- Construir la inteligencia del robot en un dispositivo Android.
- Usar un marco de visión por computadora (CV) como OpenCV, para simplificar operaciones relacionadas con CV.
- Hacer que el robot sea autónomo; esto significa que sin interacción con los humanos debe seguir la pelota y detenerse cuando esté lo suficientemente cerca del objeto.
- Diseñar un soporte para teléfono/tableta, ya que es necesario para asegurarse de que el dispositivo se encuentra encima del robot con la cámara apuntando a la frente. Se sugiere una construcción como la de la figura 13.11. Para ello, se pueden encontrar los planos en la sección de descargas en el sitio web para este capítulo.


**Figura 13.12** Soporte para teléfono/tableta

### *Hardware requerido:*

Materiales que se requieren para realizar el proyecto:

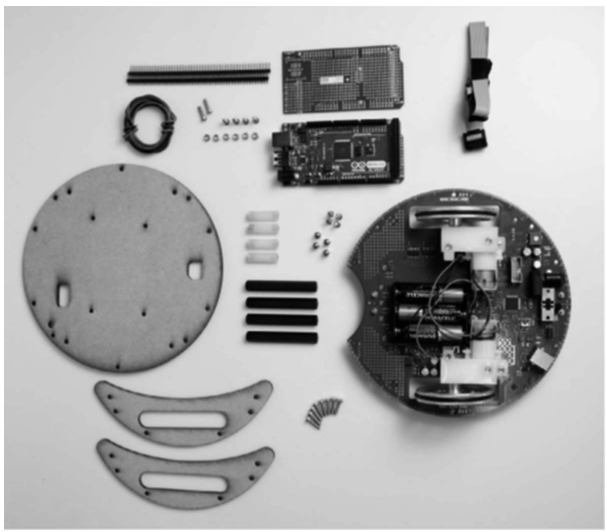


**Figura 13.13** Elementos requeridos de hardware

### *Lista de partes:*

- Cable Arduino Mega ADK + micro USB para procesador.
- Arduino prototyping shield para Arduino Mega Hea.
- Terminales de clavijas, tanto para hombres como para hembra con pasadores largos.
- El tablero del motor para el Arduino Robot.
- Pilas AA recargables (2450mAh o más preferido).
- Plataforma de madera para llevar el teléfono y el Arduino Mega ADK en la parte superior de la placa del motor.

### *Diagrama de conexiones:*

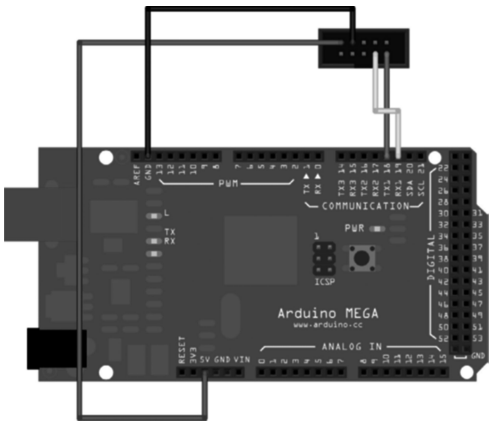



**Figura 13.14** Vista de conexiones

### *Shield de conexiones tipo protoboard*

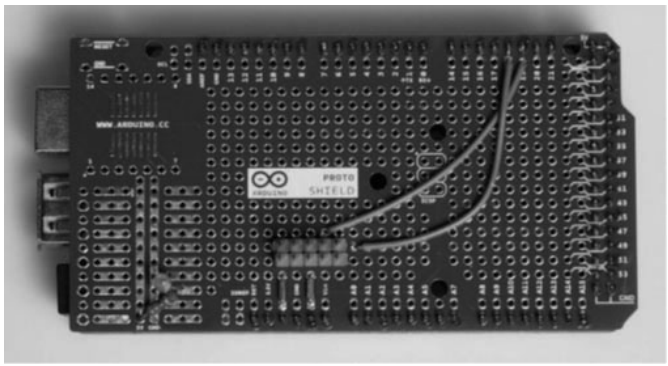



**Figura 13.15** Shield de conexiones

### *Aplicación en Android Studio*

**Figura 13.16** Vista de la aplicación para Android

## Proyecto: Lámpara de cocina controlada desde Android

En el siguiente proyecto se creará un prototipo de control de iluminación para una cocina, utilizando como medio de control un teléfono Android; a través del envío de mensajes de texto SMS y llamadas se activarán los colores de la lámpara RGB.

**Figura 13.17** Prototipo de control de iluminación

### *Partes del diseño del proyecto*

A continuación se presentan los elementos de la construcción:

- Crear una lámpara para iluminar el área de la cocina y el fregadero.
- Asegurarse de que la lámpara funcione como tal (usando luz blanca) de manera predeterminada, sin necesidad de que el usuario presione el interruptor de encendido/apagado.
- Hacer que la lámpara actúe como un accesorio de Android, ofreciendo la posibilidad de aumentar partes del sistema de notificación del teléfono.
- Hacer que la lámpara responda a la llegada de SMS y llamadas telefónicas con diferente luz de animación.
- Activar la lámpara para usarla como temporizador de cocina, mostrando la cantidad de tiempo restante como un medidor de lúmenes (todas las luces encendidas son cien por ciento del tiempo, la mitad de las luces es cincuenta, y así sucesivamente); como se muestra a continuación.

**Figura 13.18** Uso de lámpara como temporizador

## *Hardware*

A continuación se muestran los elementos de hardware requeridos:

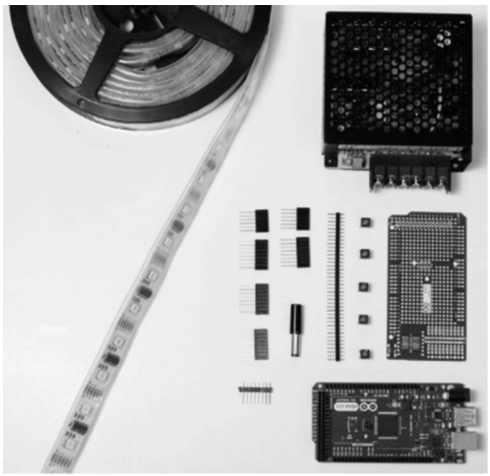


**Figura 13.19** Partes del hardware

## *Esquema de conexiones:*

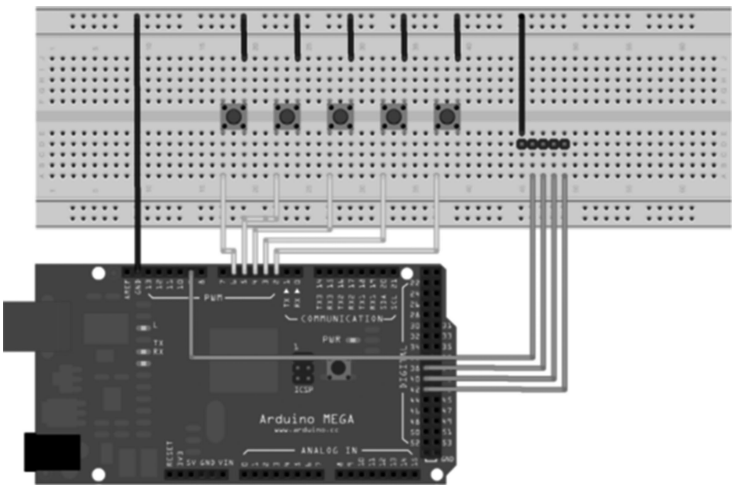



**Figura 13.20** Vista de las conexiones

Para este proyecto se utilizará una tira de LED; en la siguiente imagen se muestra un esquema del circuito del LED.

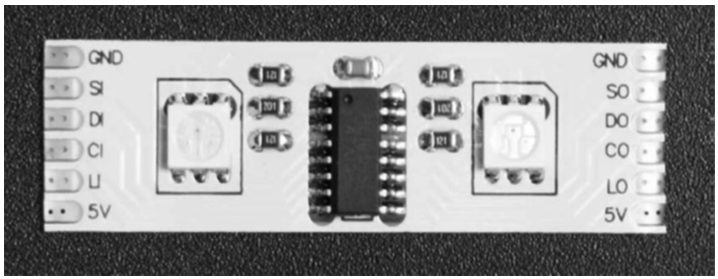



**Figura 13.21** Circuito de LED

### *Conexiones en el shield de protoboard:*

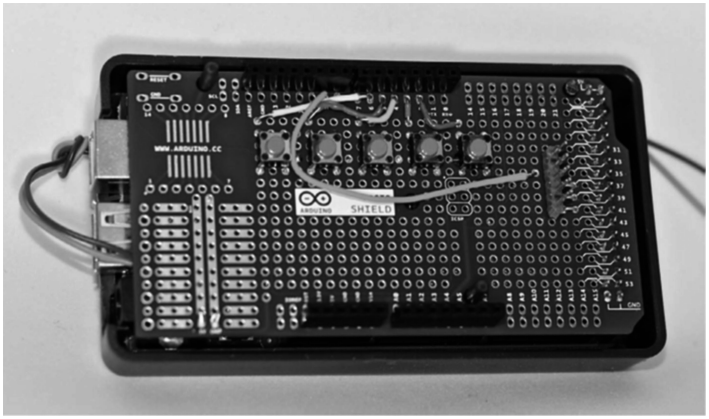



**Figura 13.22** Conexiones del shield

### *Aplicación en Android Studio*

En esta parte de la aplicación se muestra el control de tiempos para manipular el encendido:


**Figura 13.23** Control de tiempos

Diferentes tipos de eventos:

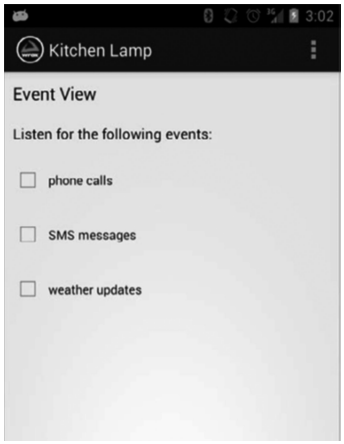



**Figura 13.24** Diferencia de eventos

### *Futuras mejoras*

A la llegada de una llamada o mensaje de texto, se responderá al encendido y apagado. Algunas mejoras que se le podrían hacer al proyecto serían las siguientes:

**Figura 13.25** Mejoras de la aplicación

- Sería interesante hacer más funciones con diversos efectos; por ejemplo, atenuación de la luz.
- También se podría agregar una función de atenuación usando un solo botón, el cual se podría presionar continuamente para regular la intensidad de la luz.
- Más importante es que, una vez que ha habido una notificación, por ejemplo, una llamada telefónica, sería bueno tener un recordatorio visual en la lámpara de las llamadas perdidas o mensajes de texto recibidos.
- Se puede hacer que un LED se prenda en color rojo para cada llamada y usar otro color para informar sobre cada llegada de SMS. Esto implicaría usar un botón para desactivar las notificaciones en la lámpara.

## Proyectos Android Things

Recientemente, Google lanzó su primer sistema operativo creado para Internet de las Cosas (IoT) llamado Android Things. En esta sección se verán algunos proyectos utilizando esta herramienta de desarrollo; a través de este sistema operativo con tarjetas de desarrollo y periféricos compatibles, como sensores, LED, servos, etc.

Las placas de prototipos juegan un papel importante en Internet de las Cosas y ayudan a desarrollar la cantidad de objetos conectados por medio de tarjetas de prototipos; existen varias tarjetas que soportan la instalación del sistema operativo Android Things:

- Arduino
- Raspberry Pi
- Intel Edison
- ESP8266
- NXP

En este caso, por cuestiones prácticas se usarán Arduino y Raspberry Pi

### *Sistema Operativo Android Things*

Android Things permite crear productos profesionales de mercado masivo en una plataforma confiable, sin conocimientos previos del diseño de sistemas integrados. Reduce grandes costos de desarrollo inicial y los riesgos inherentes a la puesta en marcha de la idea. Cuando se está listo para enviar grandes cantidades de dispositivos, los costos también se escalarán linealmente y aquellos constantes de ingeniería y pruebas se minimizarán con las actualizaciones provistas por Google. Android Things facilita el desarrollo de dispositivos integrados conectados, al proporcionar las mismas herramientas de desarrollo de Android, el mejor marco de Android de su clase y las API de Google que hacen que los desarrolladores tengan éxito en los dispositivos móviles.

Arquitectura de una aplicación con Android Things:

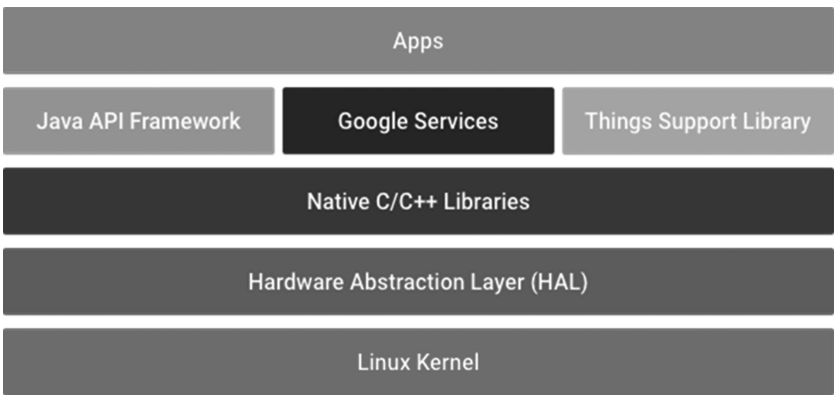



**Figura 13.26** Arquitectura de la aplicación

### *Instalación de Android Things en Raspberry Pi*

Raspberry Pi 3 es la última placa desarrollada por Raspberry. Se trata de una actualización de Raspberry Pi 2 Modelo B y, al igual que su predecesor, tiene algunas características excelentes:

- CPU ARMv8 de cuatro núcleos a 1.2 Ghz.
- Lan inalámbrico 802.11n Bluetooth 4.0.

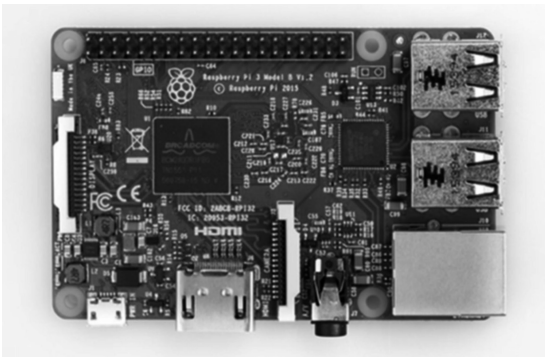


**Figura 13.27** Placa Raspberry Pi 3

### *Instalación de Android Things en Raspberry Pi*

A continuación, se presentan los pasos para instalar Android Things en la tarjeta de desarrollo Raspberry Pi. Primero es necesario entrar al siguiente link: *https://developer.android.com/things/index.html* y dar clic en Consola:


**Figura 13.28** Consola de Android Things

Firmar con la cuenta de Google ID:

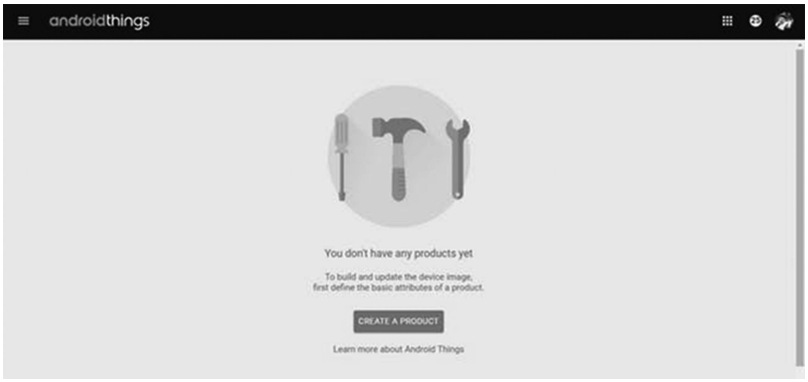



**Figura 13.29** Firma con cuenta de Google

Luego, seleccionar Crear producto:

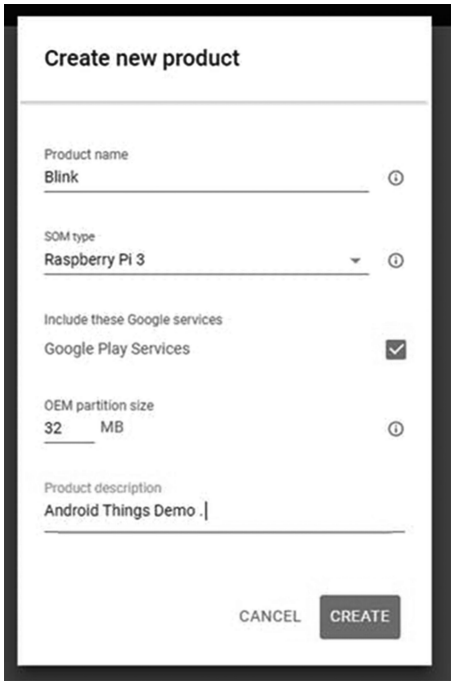



**Figura 13.30** Creación de nuevo producto

Enseguida, escribir el nombre de producto (SOM=System On Module), y dar una descripción del mismo:

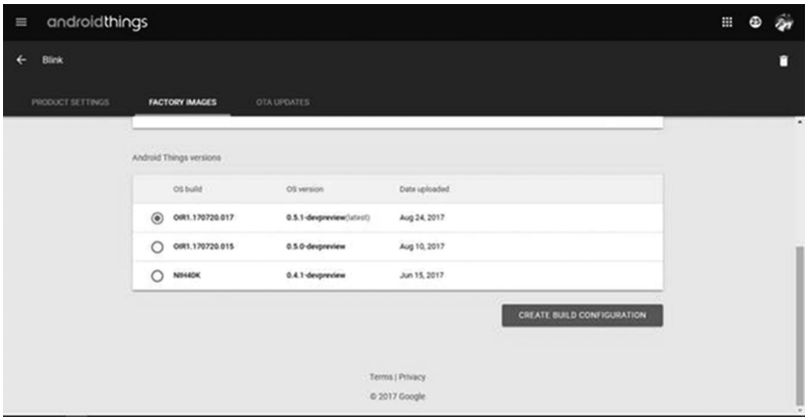



**Figura 13.31** Nombre y descripción del producto

Hacer clic en Create Build Configuration:

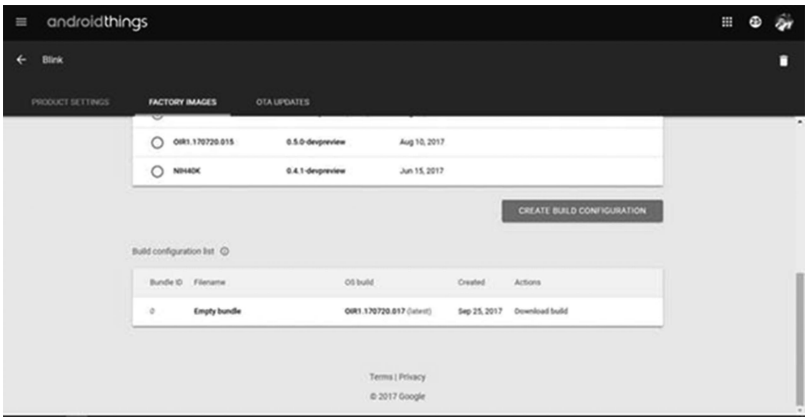



**Figura 13.32** Create Build Configuration

Luego, dar clic en Download Field:


**Figura 13.33** Descarga de archivo

Se obtendrá el archivo .Zip, como se muestra a continuación:

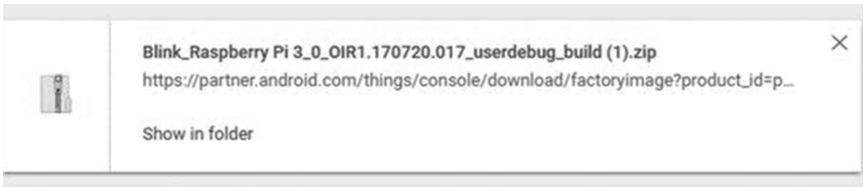



**Figura 13.34** Archivo .Zip

Extraer los archivos comprimidos:

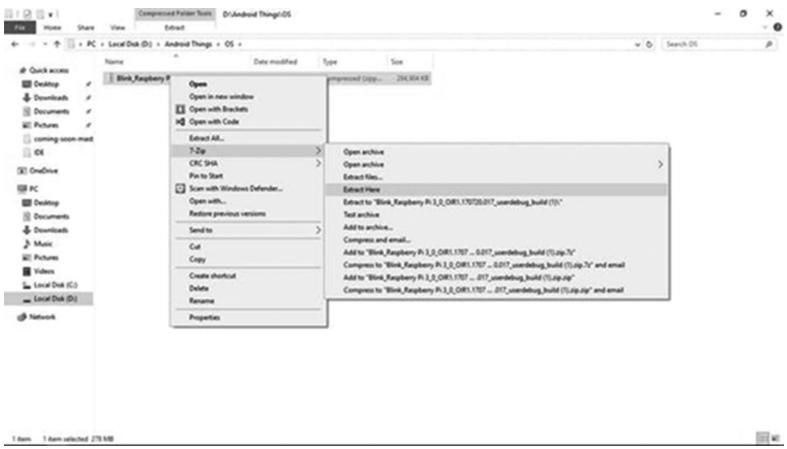


**Figura 13.35** Extracción de archivos comprimidos

Al final, se muestra el proceso de la descarga:

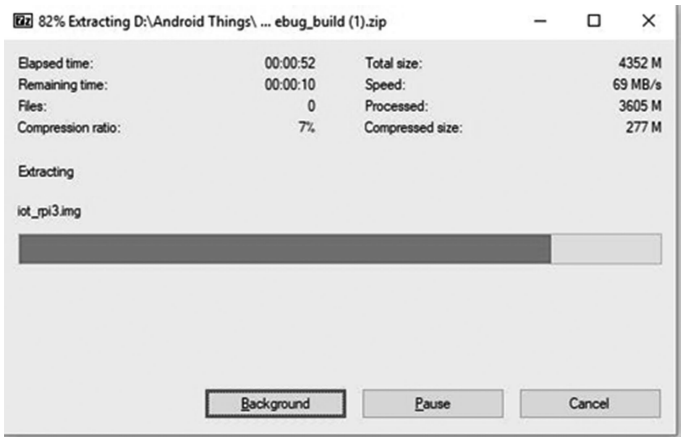



**Figura 13.36** Proceso de descarga

Por último, se muestra la imagen creada. Luego se graba la misma en una memoria microSD:


**Figura 13.37** Grabación de imagen

Después del paso anterior, se muestra una pantalla de comunicación con el sistema operativo Android Things:


**Figura 13.38** Pantalla de comunicación con Android

## Primer proyecto en Android Things

Considerando que Android Things se deriva de Android, el proceso de desarrollo y la estructura de la aplicación son los mismos utilizados en una aplicación común de Android. Por esta razón, la herramienta de desarrollo para Android Things es Android Studio. Si ya se ha usado en el pasado, con esta sección el lector podrá descubrir las principales diferencias entre una aplicación para Android Things y una de Android. De lo contrario, si es la primera vez en el desarrollo de Android, esta sección guiará paso a paso para crear la primera aplicación.

Android Studio es el entorno de desarrollo oficial para aplicaciones Android Things; por tanto, antes de comenzar es necesario que se haya instalado. Dirigirse a la siguiente dirección: *https://developer.android.com/studio/index.html* para descarga e instalación; el entorno de desarrollo debe cumplir con estos requisitos previos:

- SDK tools versión 24 o superior.
- Actualizar el SDK con Android 7 (API nivel 24).
- Android Studio 2.2 o superior.

Si el entorno no cumple con las condiciones anteriores, se debe actualizar Android Studio por medio del administrador de actualizaciones.


**Figura 13.39** Administrador de actualizaciones

### *Diferencias entre Android y Android Things*

Como se puede observar, un proyecto de Android Things es muy similar a uno de Android aunque existen algunas diferencias:

- Android Things no usa diseños múltiples para admitir diferentes tamaños de pantalla.
- Entonces, cuando se desarrolla una aplicación Android Things se crea un solo diseño.

- Android Things no es compatible con temas y estilos.
- Las bibliotecas de soporte de Android no están disponibles en Android Things.

### *Creación de una primera APP en Android Things*

Este ejemplo consiste en conectar tres botones y controlar el encendido del led RGB:

**Figura 13.40** Conexión de tres botones

### *Led RGB:*

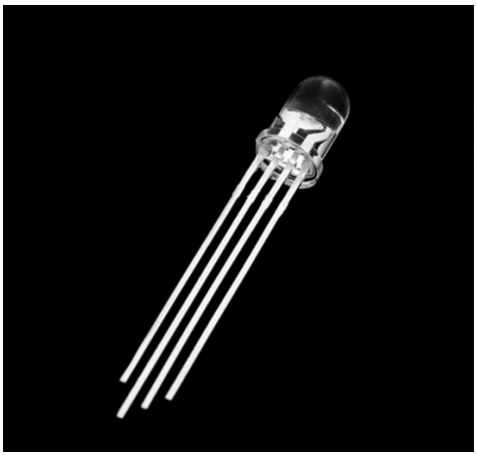


**Figura 13.41** Imagen del led

### *Conexiones del hardware:*

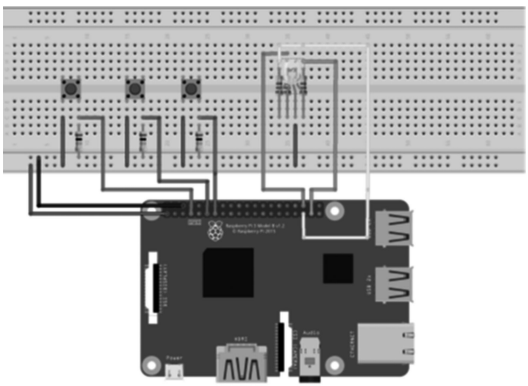


**Figura 13.42** Conexiones de hardware

### *Diagrama físico:*

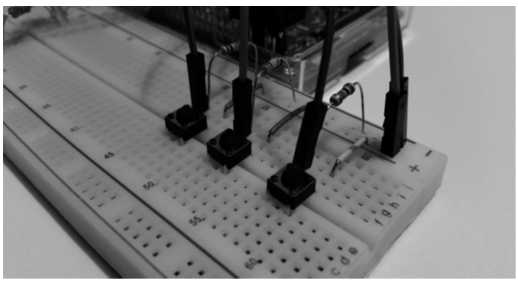


**Figura 13.43** Vista del diagrama

### *Conexión de los botones y las resistencias de 10k de tipo pull-down:*

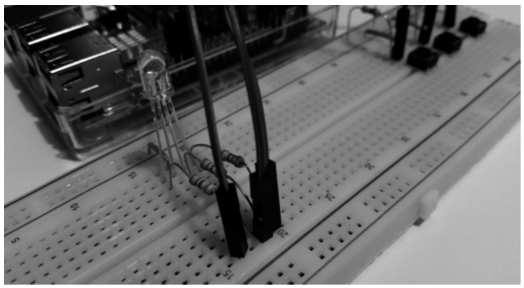


**Figura 13.44** Conexiones de botones y resistencias

### *Código de la aplicación:*

```
button1.setOnButtonEventListener(
new Button.OnButtonEventListener( ) {
@Override
public void onButtonEvent(Button button, boolean
pressed) { if (pressed) {
redPressed = !redPressed;
try {
redIO.setValue(redPressed);}
catch (IOException e1) { }
}
}
});
button2.setOnButtonEventListener(new Button.OnButtonEventListener( )
{
@Override
public void onButtonEvent(Button button,
boolean pressed) {
if (pressed) {
greenPressed = !greenPressed; try {
greenIO.setValue(greenPressed);
}
catch (IOException e1) {}
}
}
});
```

Después de haber visto lo anterior, en necesario descargar la aplicación en el dispositivo dando clic en el icono Play, de color verde:


**Figura 13.45** Descarga de la aplicación

A continuación, se muestra la pantalla de ejecución de la aplicación:

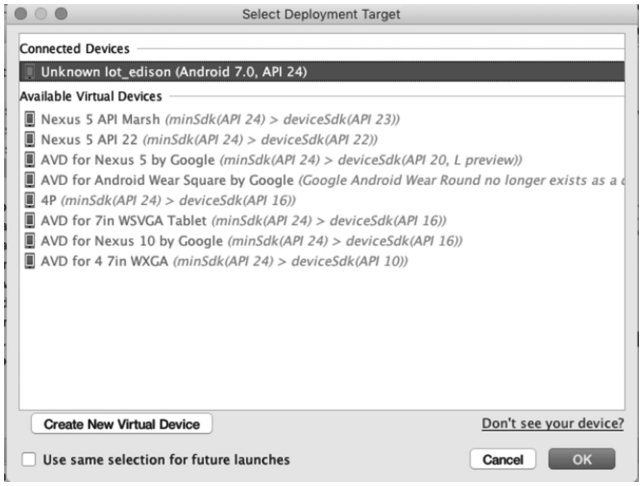



**Figura 13.46** Pantalla de ejecución

Ahora es posible ejecutar la aplicación; el proceso de instalación es el mismo que el utilizado para Android. Cuando el proceso finalice, se puede usar la aplicación presionando cada botón; entonces e deberá ver el led cambiando de color.

## Proyecto: Sistema de alarma

Un sistema de alarma es uno complejo que usa varios sensores para mantener el hogar a salvo. El corazón de este tipo de sistemas son los sensores que pueden detectar movimiento. En otras palabras, estos sensores pueden saber si un objeto se está moviendo en su área de detección; cuando esto sucede, se notifica. En esta sección se creará un proyecto de la vida real que utilice estos sensores para percibir  movimiento y notificarlo a los teléfonos inteligentes. Una vez que se haya creado, se puede expandir y agregar más sensores para controlar varias habitaciones.

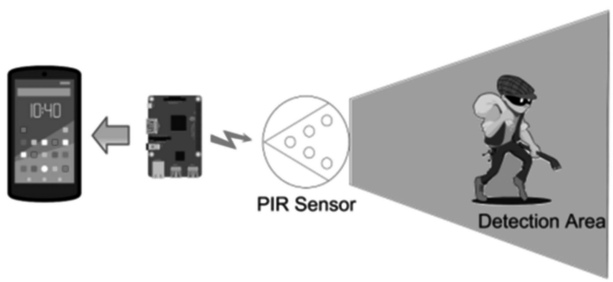



**Figura 13.47** Cadena de procesos del sistema

***Conexiones del hardware:***

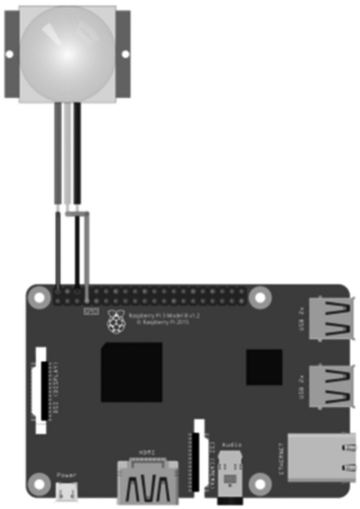


**Figura 13.48** Conexiones

## Proyecto: Sistema de monitoreo ambiental

En este proyecto se realizará un prototipo de sistema para el monitoreo de variables ambientales, como temperatura y presión.

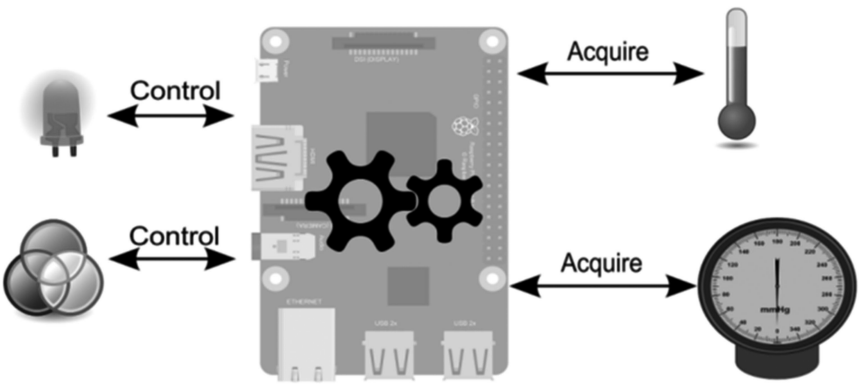



**Figura 13.49** Sistema de monitoreo ambiental

### *Sensor de presión barométrica:*

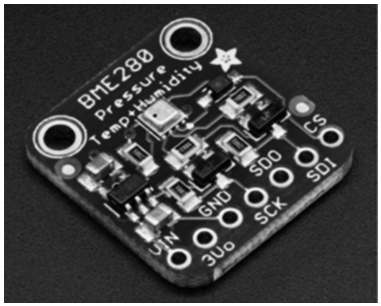



**Figura 13.50** Sensor requerido

### *Esquema del proyecto:*

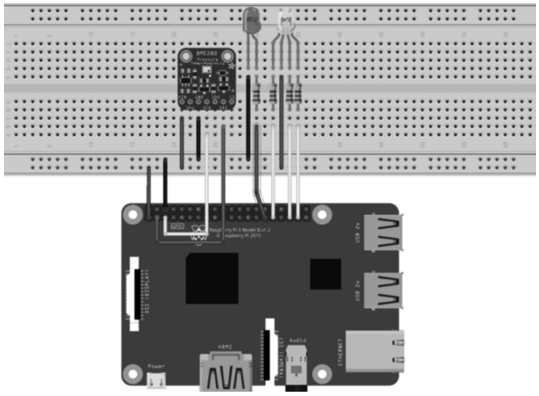
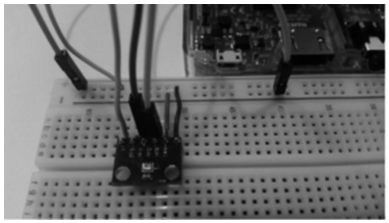


**Figura 13.51** Conexiones del proyecto

### *Conexiones del hardware:*



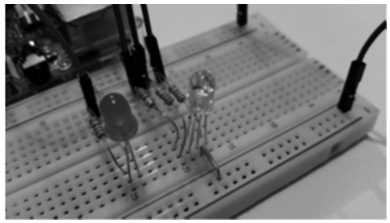


**Figura 13.52** Conexiones del hardware

## Proyecto: Sistema inteligente de luz ambiental

En este proyecto se integrará el Arduino como puntos de conexión; es decir, como diferentes nodos a través de comunicación Ethernet y, desde la placa Raspberry Pi cargada al sistema Android Things, se controlarán las tarjetas Arduino. Antes de profundizar en los detalles de implementación del proyecto, es útil tener una visión general de lo que se pretende construir: un sistema que tiene un centro de control único representado por Android Things, y varios IoT remotos con tarjetas Arduino que están conectadas a tiras de led RGB. Estas tarjetas Arduino IoT reciben los comandos de la aplicación Android Things y de acuerdo con ellos, se configuraron varias tiras de led RGB o se aplican varios efectos de luz.

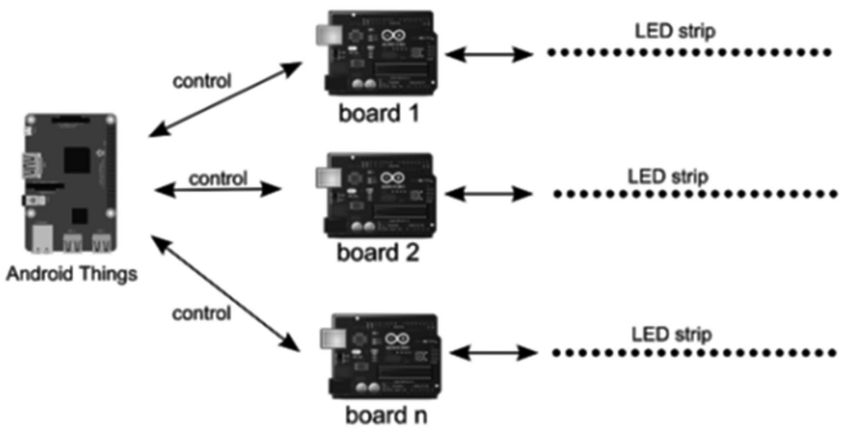



**Figura 13.53** Sistema inteligente de luz ambiental

### *Componentes del proyecto: Tira de led*

**Figura 13.54** Tira de led

### *Fuente de energía:*

**Figura 13.55** Vista de la fuente de energía

### *Arquitectura de conexión del Arduino:*

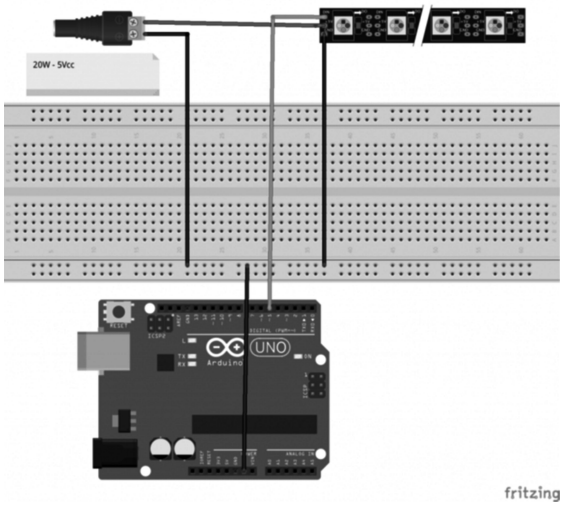



**Figura 13.56** Conexiones Arduino

### *Código del sketch del Arduino:*

```
#include <Adafruit_NeoPixel.h>
#include <Ethernet.h>
#include “aREST.h”

#define PIN 5
#define LED_NUMBER 60
```

```
#define FORWARD 1
#define REVERSE 2

// Server definition
#define SERVER_PORT 80


byte mac[ ] = { 0xDD, 0xAB, 0xCE, 0xFF, 0xEE, 0xED };
IPAddress ip(192,168,1,14);

struct data {
  String color;
  int wait;
  int dir;
  int func;

  int r;
  int g;
  int b;
} ;

Adafruit_NeoPixel strip =
          Adafruit_NeoPixel(LED_NUMBER, PIN, NEO_GRB + NEO_KHZ800);

// create the server
EthernetServer server(SERVER_PORT);

// Create aREST
aREST rest = aREST( );

void setup( ) {
  Serial.begin(9600);
  Serial.println("Arduino Strip RGB Led");
  strip.begin( );
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    Ethernet.begin(mac, ip);
  }
  Serial.println("Connection ok");
  server.begin( );
  rest.function("fill", setStripColor);
  rest.function("clear", setClearStrip);
  rest.function("rainbow", setRainbow);
}
```

```
void loop( ) {
  EthernetClient client = server.available( );
  rest.handle(client);
  //clearStrip(5, REVERSE);
 // rainbow(5, FORWARD);
 // fillStrip(strip.Color(40,190,49), 10, FORWARD);
}

void rainbow(int wait, int direction) {
   int first, last;
   setDirection(&first, &last, direction);
   byte color[3];
   byte count, a0, a1, a2;
   color[count]=random(256);
   a0=count+random(1)+1;
   color[a0%3]=random(256-color[count]);
   color[(a0+1)%3]=255-color[a0%3]-color[count];

  count+=random(15); // to avoid repeating patterns
  count%=3;
  fillStrip(strip.Color(color[0], color[1], color[2]), wait, direction);

}
void fillStrip(uint32_t color, int wait, int direction) {
   int first, last;
   setDirection(&first, &last, direction);

  for (int p = first; p <= last; p++) {
    strip.setPixelColor(abs(p), color);
    strip.show( );
    delay(wait);
  }
}

void clearStrip(int wait, int direction) {
  int first, last;
  setDirection(&first, &last, direction);

  for (int p = first; p <= last; p++) {
    strip.setPixelColor(abs(p), 0);
    strip.show( );
    delay(wait);
  }
```

```
}
void setDirection(int *first, int *last,int direction) {

  if (direction == FORWARD) {
    *first = 1;
    *last = LED_NUMBER;
  }
  else {
    *first = -LED_NUMBER;
    *last = -1;
  }
 }

// AABBDD Dir(1) Wait(2)
struct data parseCommand(String command) {
  struct data parsedData;

  parsedData.color = command.substring(1,7);
  String p = command.substring(0,1);
  parsedData.dir = p.toInt( );
  p = command.substring(7,9);
  parsedData.wait = p.toInt( );

  parsedData.func = command.substring(9).toInt( );

  long tmpColor = strtol( &("#" + parsedData.color)[1], NULL, 16);
  parsedData.r = tmpColor >> 16;
  parsedData.g = tmpColor >> 8 & 0xFF;
  parsedData.b = tmpColor & 0xFF;

  return parsedData;
}

// Function exposed
int setStripColor(String command) {
  Serial.println("Color strip function...");
  struct data value = parseCommand(command);
  debugData(value);
  fillStrip(strip.Color(value.r,value.g,value.b), value.wait, value.dir);
  return 1;
}
int setClearStrip(String command) {
  Serial.println("Clear strip function...");
  struct data value = parseCommand(command);
```

```
    debugData(value);
    clearStrip(value.wait, value.dir);
    return 1;
  }
  int setRainbow(String command) {
    Serial.println("Rainbow strip function...");
    struct data value = parseCommand(command);
    rainbow(value.wait, value.dir);
    debugData(value);
  }
  void debugData(struct data value) {
    Serial.println("=========================");
    Serial.println("Color ["+value.color+"]");
    Serial.println("Direction ["+String(value.dir)+"]");
    Serial.println("Wait ["+String(value.wait)+"]");
    Serial.println("Func ["+String(value.func)+"]");
    Serial.println("R ["+String(value.r)+"]");
    Serial.println("G ["+String(value.g)+"]");
    Serial.println("B ["+String(value.b)+"]");
    Serial.println("=========================");

  }
```

### *Aplicación en Android Things:*

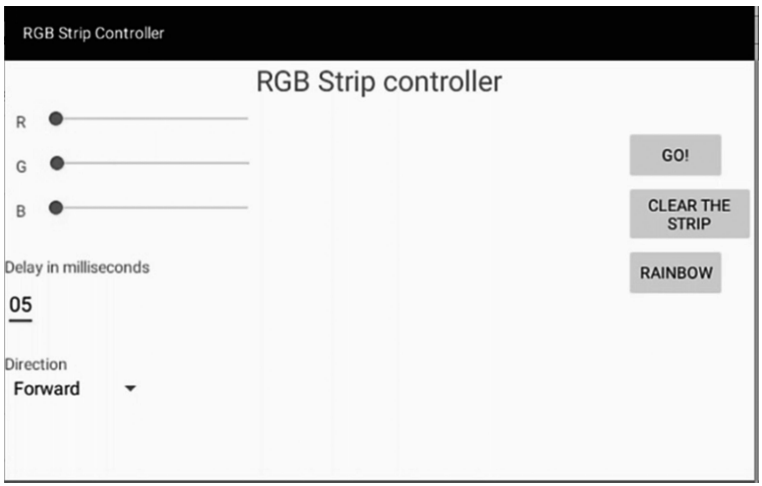



**Figura 13.57** Vista en Android Things

### *Interfaz web*

Ahora, es posible realizar una interfaz para el control:


**Figura 13.58** Interfaz de control

## Proyecto: Ojo espía

En este proyecto se utilizará Android Things para controlar una cámara y un servomotor parar rotarla. Se usará el manejo de la modulación por ancho de pulso (PWM). Se construirá un sistema completamente funcional de Android Things que sea capaz de adquirir imágenes usando la cámara y, al mismo tiempo, controlar la dirección de la misma con el servomotor.

### *Esquema del proyecto:*

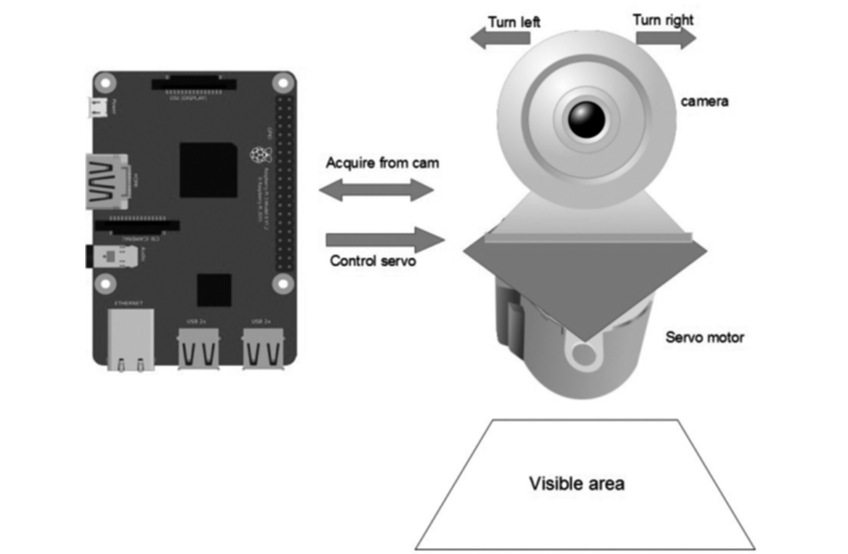



**Figura 13.59** Esquema del proyecto

## *Componentes del proyecto*

Cámara para Raspberry Pi 3:

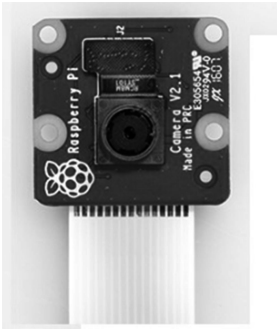



**Figura 13.60** Cámara

## *Servomotor:*

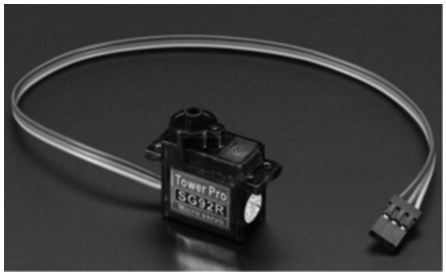



**Figura 13.61** Vista del servomotor

## *Sujetador para la cámara:*

**Figura 13.62** Sujetador

### *Desarrollo del proyecto*

En la primera parte se describió cómo controlar un servomotor; en la segunda se verá cómo adquirir imágenes de Android usando una cámara. Como se recordará, el servomotor se usa en este proyecto para rotar la cámara y hacer una exploración más amplia. Además, se tiene una IU que se puede usar para controlar el servo y tomar la fotografía. La interfaz de usuario es muy simple e intuitiva; el resultado se muestra en la siguiente figura:


**Figura 13.63** Interfaz del proyecto

### *Control de servomotor en Android Things:*

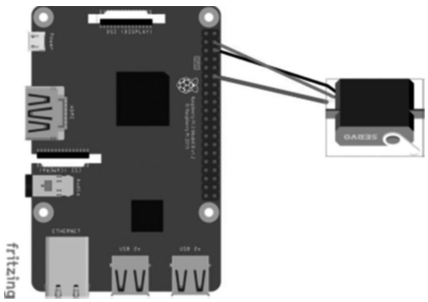



**Figura 13.64** Control de servomotor

### *Imagen de la conexión:*

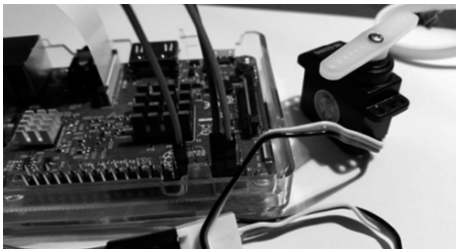


**Figura 13.65** Conexión del proyecto

### *Código de la aplicación para el control del servo:*

```
btnLeft.setOnClickListener(new
View.OnClickListener( ) {
@Override
public void onClick(View v) {
angle += STEP;
setServoAngle(angle);
}
});
btnRight.setOnClickListener(new
View.OnClickListener( ) {
@Override
public void onClick(View v) {
angle -= STEP;
setServoAngle(angle);
}
});
```

Rotación del servomotor:

```
private void setServoAngle(int angle) {
if (angle > mServo.getMaximumAngle( ))
angle = (int) mServo.getMinimumAngle( );
if (angle < mServo.getMinimumAngle( ))
angle = (int) mServo.getMinimumAngle( );
try {
mServo.setAngle(angle);
}
catch (IOException e) {
e.printStackTrace( );
} }
```

### *Uso de la cámara en Android Things*

La cámara está conectada; debe usarse usa una interfaz serie común (CSI-2), además de una cámara compatible como se especifica al principio de este capítulo. Para manejarla, se empleará android.hardware.camera2 (agregado desde API nivel 21). Este paquete proporciona todas las clases e interfaces necesarias para manejar una cámara conectada a un dispositivo Android. Como se verá más adelante, el proceso para tomar una foto en Android Things es el mismo que para Android.

### *Código para el uso de la cámara:*

```
cManager = (CameraManager)
ctx.getSystemService(Context.CAMERA_SERVICE);
try {
String[ ] idCams = cManager.getCameraIdList( );
camId = idCams[0];
}
catch (CameraAccessException e) {
e.printStackTrace( );
}

imgHandler.start( );
iReader = ImageReader.newInstance(320, 240,
ImageFormat.JPEG, 1);
iReader.setOnImageAvailableListener(new
ImageReader.OnImageAvailableListener( ) {
@Override
public void onImageAvailable(ImageReader
reader) {
listener.onImageReady(reader);
}
}, new Handler(imgHandler.getLooper( )));

public void openCamera( ) {
try {
cManager.openCamera(camId, stateCallback, null);
}
catch (CameraAccessException e) {
e.printStackTrace( );
}
catch (SecurityException se) {
se.printStackTrace( );
}
}

private final CameraDevice.StateCallback
stateCallback = new
CameraDevice.StateCallback( ) {
@Override
public void onOpened(@NonNull CameraDevice
camera) { Log.d(TAG, “Camera opened”);
AndroidCamera.this.camera = camera;
listener.onCameraAvailable( );
```

```
}
@Override
public void onDisconnected(@NonNull CameraDevice
camera) { Log.d(TAG, “Camera disconnected”);
}
@Override
public void onError(@NonNull CameraDevice camera,
int error) { Log.d(TAG, “Camera Error” + error);
}
};
```

### *Interacción entre Android y Android Things*

En este proyecto se descubrirá cómo integrar Android Things y Android. Hoy en día existen varios productos de mercado que tienen una aplicación móvil complementaria para interactuar con sistemas inteligentes. Aquí hay algunos de ellos:

- Sistemas de monitoreo remoto.
- Alarmas.
- Control de dispositivos a distancia.

En términos generales, hay tres escenarios diferentes:

- Un teléfono controla un objeto inteligente (como la placa Android Things) (maestro-esclavo).
- Un teléfono recibe transmisión de datos a través de la placa Android Things.
- Un teléfono inteligente recibe notificaciones del sistema Android Things cuando un evento ocurre.

### *Esquemas de conectividad:*


**Figura 13.66** Vista de la conectividad

## Proyecto: Control de una tira de led desde una aplicación Android

A continuación, se presenta el esquema de conectividad del proyecto. Se puede observar en el esquema el uso de una aplicación desde el dispositivo móvil; se accede de forma remota al sistema Raspberry para controlarse a través de conexión Ethernet, Wifi o bluetooth desde el dispositivo remoto.

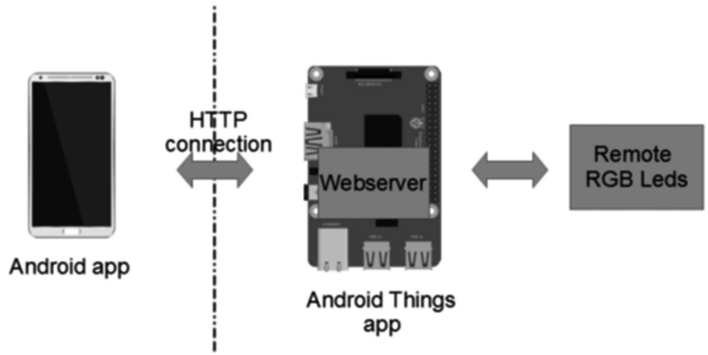



**Figura 13.67** Esquema de uso de aplicación

### *Desarrollo de la interfaz en el dispositivo móvil:*


**Figura 13.68** Interfaz del dispositivo

### *Interfaz del control de los colores*


**Figura 13.69** Interfaz de control de colores

## 13.2 Ir hacia adelante: El futuro del Internet de las Cosas y de Bluetooth Low Energy

En esta sección se presentarán varios casos de uso práctico de IoT construidos mediante Bluetooth Low Energy (BLE), que ya aplican muchas personas en la vida real. Estos casos darán una vista panorámica de las aplicaciones BLE respecto de IoT en diferentes áreas. Eventualmente, se discutirá qué depara el futuro para BLE e IoT, y se concluirá consolidando lo aprendido hasta ahora.

### 13.2.1 Bluetooth 5

Bluetooth 5 fue anunciado por Bluetooth SIG el 16 de junio de 2016; finalmente se dio a conocer oficialmente el 7 de diciembre de 2016. Según la publicidad inicial, se dijo que tenía "Cuádruple el rango de su predecesor, duplicaba la velocidad y [tenía] 8 veces la capacidad de transmisión para conexiones de baja energía".

| | BLE 4.2 | BLE 5 | BLE 5 Long Range (S=2) | BLE 5 Long Range (S=8) |
|---|---|---|---|---|
| Connection speed | 1 Mbps | 2 Mbps | 1 Mbps | 1 Mbps |
| Network data rate | 1 Mbps | 2 Mbps | 500 Kbps | 125 Kbps |
| Data throughput | 800 Kbps | 1400 Kbps | 380 Kbps | 109 Kbps |
| Error Correction | None | None | FEC | FEC |
| Bluetooth 5 Requirement | Mandatory | Optional | Optional | Optional |

**Figura 13.70** Características de Bluetooth 5

## *Casos prácticos*

En esta sección se verán algunas aplicaciones y casos de uso del Internet de las Cosas, utilizando Bluetooth Low Energy.

### Philips Sonicare Toothbrush

**Figura 13.71** Cepillo dental: *http://www.usa.philips.com/c-m-pe/electric-toothbrushes*

### Wahoo Balance Smartphone Scale

**Figura 13.72** Báscula: *http://eu.wahoofitness.com/devices/accessories/wahoo-balance-smartphonescale*

### iLumi Smart Bulb


**Figura 13.73** Iluminación: *https://ilumi.co/*

### Eve Smart Light Switch


**Figura 13.74** Interruptor de luz: *https://www.youtube.com/watch?v=sQfc4bjqM6I*

### Danalock

**Figura 13.75** Para seguridad en el hogar: *https://danalock.com/*

### Fobo

**Figura 13.76** Medidor de presión y temperatura: *https://my-fobo.com/*

### Tile


**Figura 13.77** Búsqueda de llaves: *https://www.thetileapp.com/*

### HAPIfork


**Figura 13.78** Control de alimentación: *https://www.ncbi.nlm.nih.gov/pubmed/26100137*

## 13.3 El futuro de BLE (Bluetooth Low Energy)

Se han mencionado las diversas características de Bluetooth Low Energy 5, que agrega una mejora significativa sobre Bluetooth Low Energy 4.2, especialmente en términos de rango y velocidad. Con estas nuevas mejoras y promesas, se verá cómo Bluetooth Low Energy penetra cada vez más en la vida cotidiana.

En esta sección se discutirán algunos de los productos y tecnologías basados en BLE 5, que se encuentran en desarrollo activo y se espera que lleguen pronto al consumidor.

### Dreem

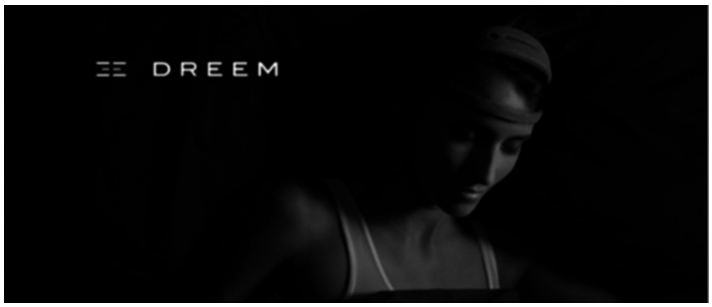



**Figura 13.79** Ayuda para conciliar el sueño: *https://dreem.com/*

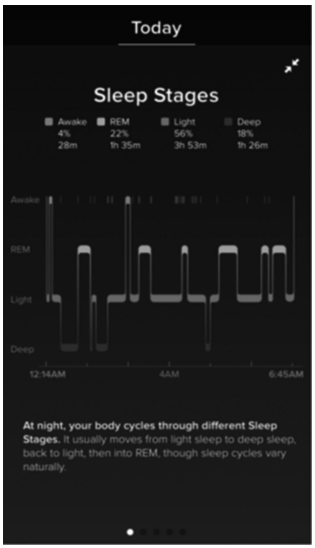



**Figura 13.80** Medidor de profundidad del sueño

**BabyGigl**


**Figura 13.81** Ingesta del bebé: *http://www.slowcontrol.com/en/baby-gigl/*

## RESUMEN

Cada que exista la pregunta del porqué existen las computadoras, se podrá decir que sirven para traducir la experiencia humana del mundo físico a una experiencia virtual en el mundo digital. Si los algoritmos de visión por computadora son similares cuando se hace que una computadora "vea", entonces BLE proporciona una manera de hablar y de sentir entre dispositivos.

Si el lector voltea a su alrededor, a ver los objetos inanimados en el hogar, se podría imaginar: ¿Qué me diría ese objeto si pudiera hablar? Una planta de interior podría decir que el suelo está seco, el plato de comida para perros podría decir que está vacío, y así con infinidad de cosas.

En este libro se dieron las bases para desarrollar aplicaciones del mundo real aplicando la creatividad e imaginación. Se inició con una introducción formal a Bluetooth Low Energy a la luz de IoT, continuando con el diseño de aplicaciones BLE con IoT para varios sensores, beacons y rastreadores de fitness.

Finalmente, se presentaron varias aplicaciones prácticas y ejemplos para determinar el impacto que Bluetooth Low Energy tiene en el mundo de IoT. Con estas bases se pretende que el lector explore nuevos escenarios y pueda emprender sus propios proyectos y desarrollos en el ámbito de esta tecnología en su vida cotidiana.

# Índice analítico