

ENVÍO DE DATOS DE ARDUINO A MIT APP INVENTOR: SENSOR DE TEMPERATURA

CONTROLANDO ARDUINO DESDE DISPOSITIVOS
MÓVILES

Jose Luis Núñez
José Pujol

CEP Sevilla 2019



ÍNDICE



1. Finalidad del sistema
2. Arduino
3. Aplicación MIT APP Inventor
4. Mejorando la app con CloudDB
5. Propuestas de mejora



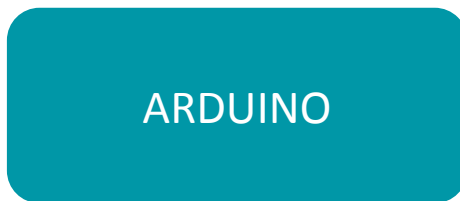
FINALIDAD DEL SISTEMA



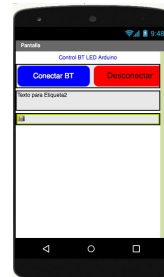
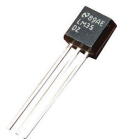
Obtener la temperatura mediante un sensor analógico y recibirla de manera inalámbrica con MIT App Inventor.

Mejora: Si abrimos la app de manera remota, aunque no tengamos conectividad BT, obtendremos el último valor de temperatura monitorizado por el sistema

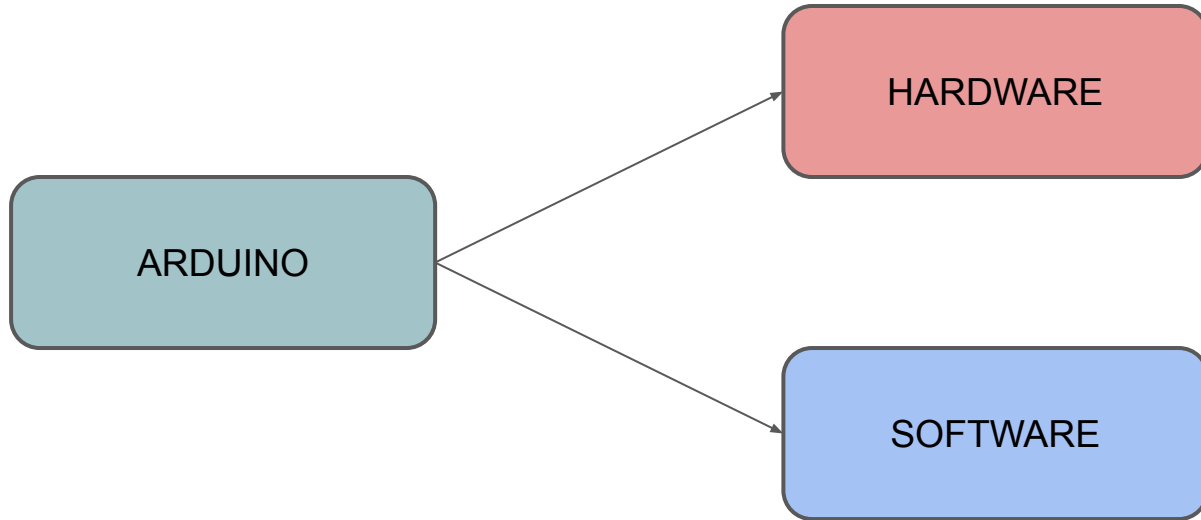
ENTRADAS
→
Sensor de temperatura
Carácter



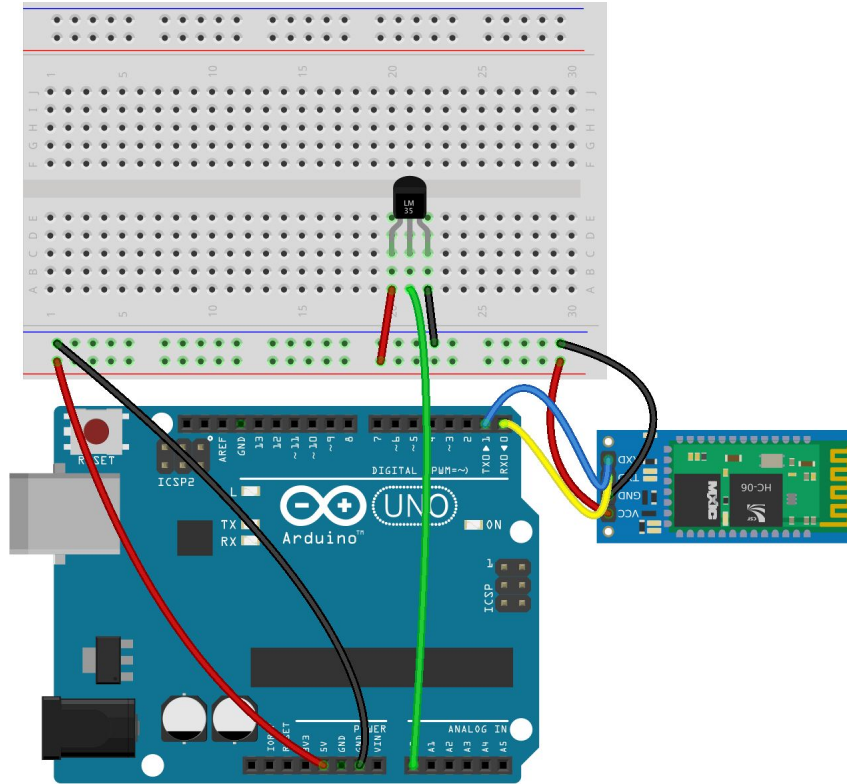
SALIDAS
→
Display Dispositivo móvil
Temperatura (°C)



ARDUINO



Hardware



fritzing

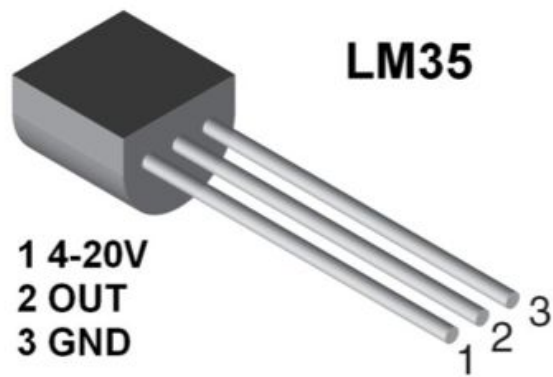
| HC06 | ARDUINO |
|----------|---------|
| GND | GND |
| 5V | 5V |
| Tx(entr) | 10 |
| Rx(sal) | 11 |

Nota:

No podemos tener el Bluetooth conectado a los pines Rx, Tx en el momento que cargamos el programa

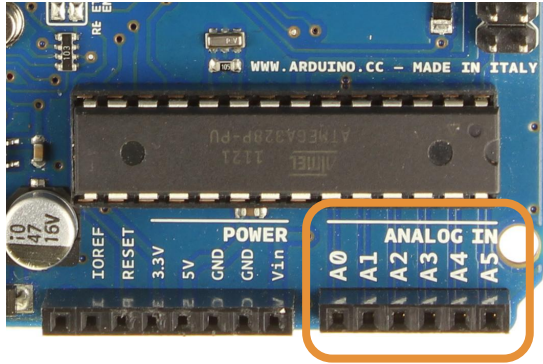
Sensor de temperatura LM35

1. Tensión de salida proporcional a la temperatura
2. Cada grado centígrado equivale a 10mv



1°C=10mv
5v=1024 pasos

analogRead



Entradas analógicas: A0-A5

10 bits = 2^{10} = 1024 niveles

No necesitan configuración

```
int valor=analogRead(pin);
```

```
valor= voltaje*1024/5
```

| Valor | Voltaje |
|-------|---------|
| 0 | 0v |
| 1023 | 5v |

Variables: alcance

El alcance (ámbito, scope) de la variable nos indica aquella zona del programa donde esta propia variable tiene existencia y se puede utilizar.

```
int a=10;
```

a | Variable global. Está definida fuera de todas las funciones y tiene existencia en todo el programa.

```
void setup(){  
  int b=2*a;  
}
```

b y c | Variables locales. Solo tienen ámbito donde están definidas. Podremos utilizar la variable “b” dentro de la función setup y la variable “c” dentro de la función loop.

```
void loop(){  
  float c=134;  
}
```

Se recomienda utilizar variables locales pues son más eficientes en la gestión de memoria. Sin embargo, en Arduino se utilizan frecuentemente las variables globales para lectura de datos por sencillez de programa.

Variables tipos y tamaños

| Tipo | Nº de bits | Rango |
|---------|------------|--|
| boolean | 1 | 0 o 1 |
| byte | 8 | 0 a 255 |
| int | 16 | -32767 a 32678 |
| long | 32 | -2×10^6 a 2×10^6 |
| float | 32 | -3.4×10^{34} a 3.4×10^{34} |

Operaciones

Las operaciones se hacen con el tipo de la variable. Es decir, si realizamos una operación en la que utilizamos variables de tipo entero, el resultado será entero.

A continuación se muestra un ejemplo:

```
float temperatura= (lectura * 5.0 * 100.0) / 1024.0;
```

```
float temperatura= (lectura * 5 * 100) / 1024;
```

Obtenemos valores diferentes

Software

```
int incomingByte; // variable para leer los bytes de entrada
// variables temperatura
const int LM35Pin = A0; // pin conexion
float temperature = 0; // variable almacenar lectura

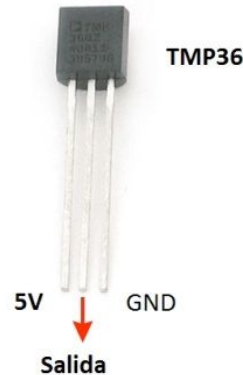
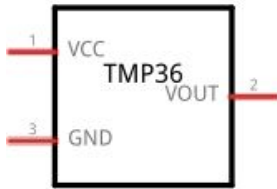
void setup() {
    // inicializamos la comunicacion serie
    Serial.begin(9600);
}

void loop() {
    // leemos el sensor
    int lectura = analogRead(LM35Pin); // Valor entre 0 y 1023
    // convertimos el sensor a °C
    temperature = (5.0 * lectura * 100.0) / 1024.0;
    // comprobamos si hay datos de entrada
    if (Serial.available() > 0) {
        // lectura del byte mas antiguo del buffer serial
        incomingByte = Serial.read();
        // si el byte es T envia dato de temperatura
        if (incomingByte == 'T') {
            Serial.print(temperature);
        }
    }
    delay(100);
}
```

[Código](#)

Sensor de temperatura TMP36

1. Tensión de salida proporcional a la temperatura
2. Relación Voltaje temperatura según la fórmula



$$\text{Temp in } ^\circ\text{C} = [(\text{Vout in V}) - 5] * 100$$

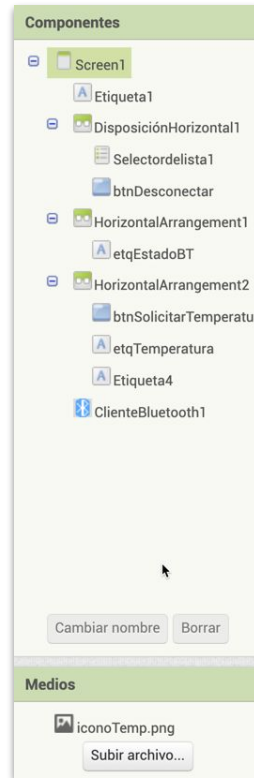
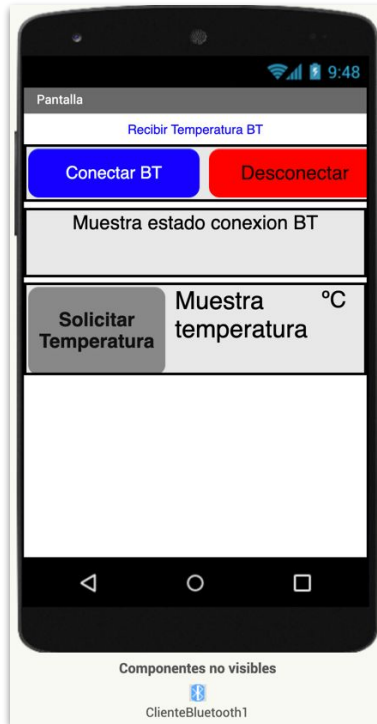
```
float voltaje = valor * 5.0 / 1024.0; // voltios
```

```
float temperaturaC = (voltaje - 0.5) * 100 ; // ° Celsius
```

Propuesta de actividades

- Visualizar la temperatura desde el monitor serie del PC al enviar una T

APLICACIÓN MIT APP INVENTOR



1. Conectamos el BT
2. Solicitamos la temperatura
3. Mostramos el resultado en °C en una etiqueta



Configuración del selector bluetooth

inicializar global `datos_entradaBT` como `" "`

cuando `Screen1` .Iniciar

ejecutar

poner `etqEstadoBT` . Texto como `" Estado desconectado "`

poner `etqTemperatura` . Texto como `" No detectada "`

cuando `Selectordelista1` .AntesDeSelección

ejecutar

poner `Selectordelista1` . Elementos como `ClienteBluetooth1` . DireccionesYNombres

cuando `Selectordelista1` .DespuésDeSelección

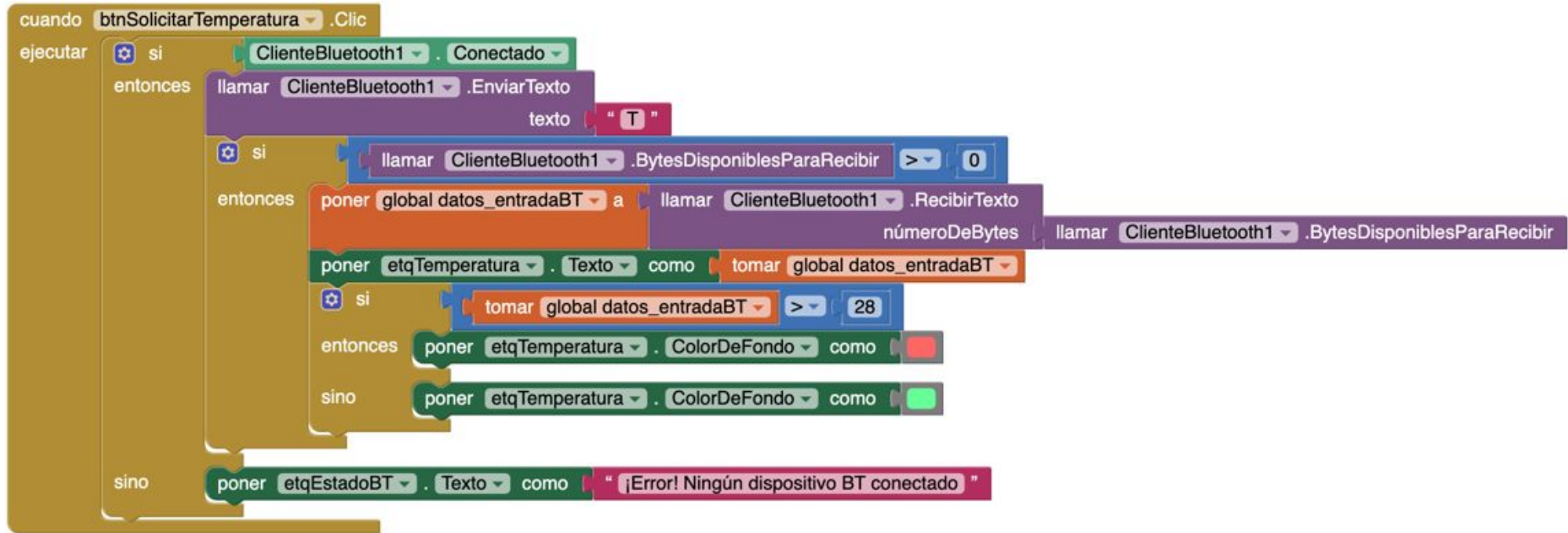
ejecutar

si `llamar` `ClienteBluetooth1` .Conectar
dirección `Selectordelista1` . Selección

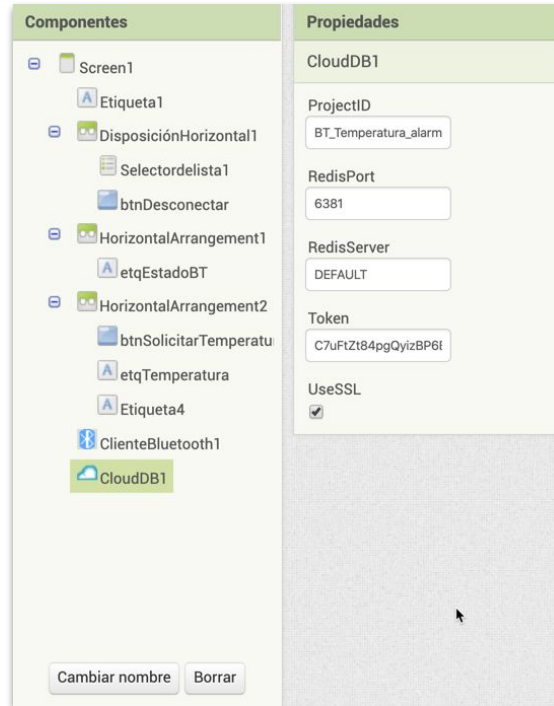
entonces poner `etqEstadoBT` . Texto como `" Estado conectado "`

sino poner `etqEstadoBT` . Texto como `" Error de conexión "`

Envío de información por BT



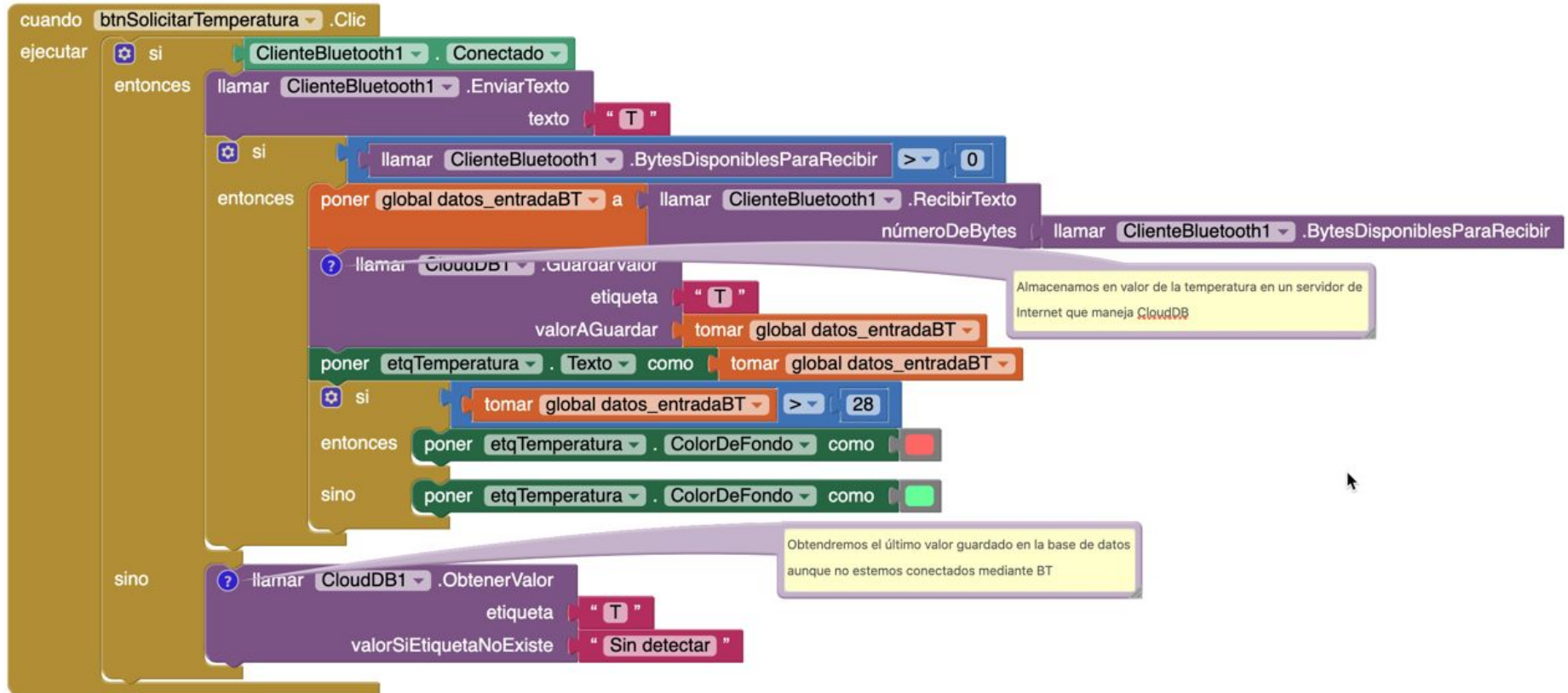
MEJORANDO NUESTRA APP CON CLOUDDB



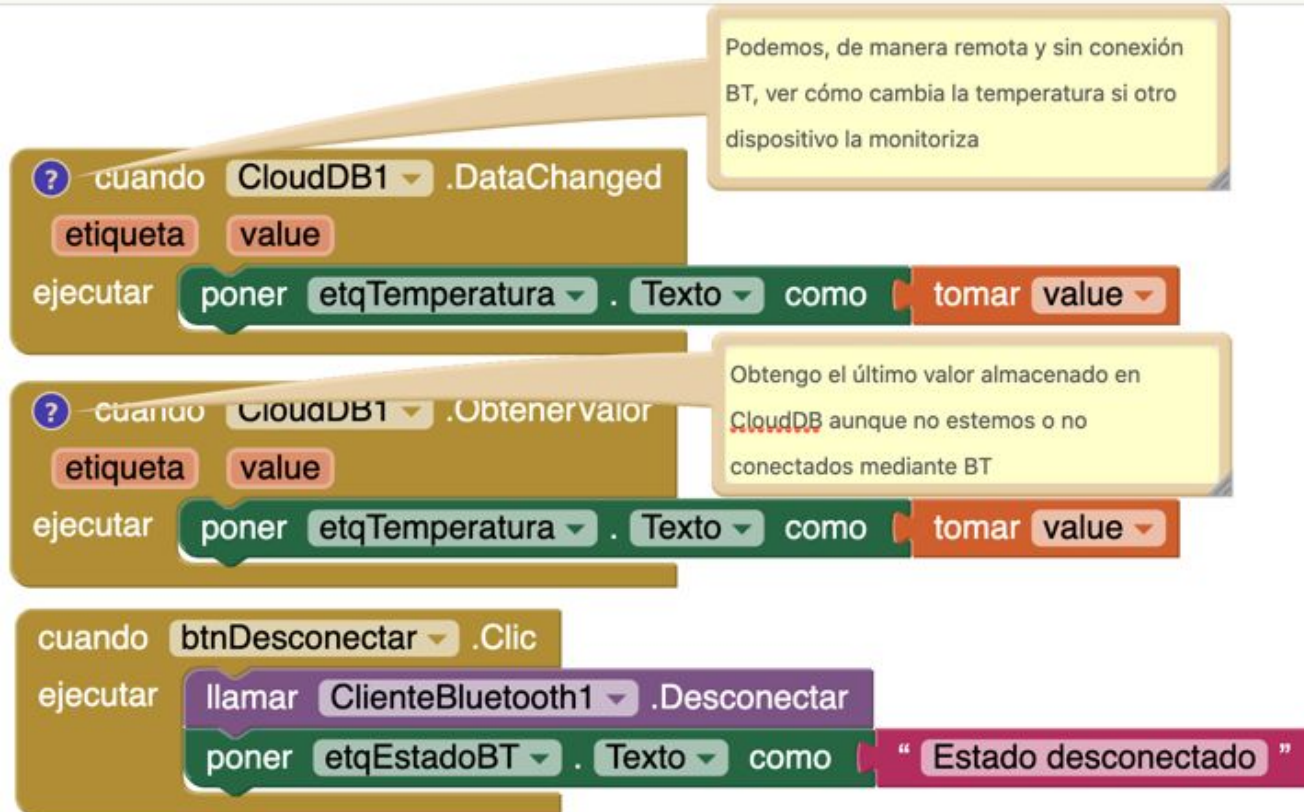
Ahora además de obtener la salida por pantalla podremos ver la última temperatura obtenida por el sensor ¡sin estar conectados por BT!



Trabajando con datos en la nube



Gestión de datos en CloudDB



Propuesta de actividades

- ¿Cómo podríamos conseguir que nuestra app mostrara la temperatura también en grados Kelvin?

LICENCIA



Esta guía se distribuye bajo licencia Reconocimiento-CompartirIgual Creative commons 4.0

Las diapositivas son obra de Jose Pujol y Jose Luis Núñez creadas para el curso “Controlando Arduino desde el teléfono móvil” para el CEP de Sevilla y han sido elaboradas usando parte del material elaborado para el curso “Tech Project: Arduino en el aula” que fue realizado por Jose Antonio Vacas y Jose Pujol en colaboración con Avante s.l.

