

SOLUCIONARIO DE ACTIVIDADES PROPUESTAS

Actividad b. del capítulo 3.4

```
// DHT sensor library for ESPx - Version: Latest
#include <DHTesp.h>
/*
    float dht11_humedad;
    float dht11_temperatura;
    bool estado;
*/
#include "thingProperties.h"
//Crea objeto
DHTesp dht;
void setup() {
    // Initialize serial and wait for port to open:
    Serial.begin(9600);
    // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
    delay(1500);
    // Defined in thingProperties.h
    initProperties();
    // Connect to Arduino IoT Cloud
    ArduinoCloud.begin(ArduinoIoTPreferredConnection);
    setDebugMessageLevel(2);
    ArduinoCloud.printDebugInfo();
    dht.setup(21,DHTesp::DHT11);
}

void loop() {
    ArduinoCloud.update();
    // Your code here
    dht11_temperatura=dht.getTemperature();
    dht11_humedad=dht.getHumidity();
    delay(4000);
    if (dht11_temperatura>29){
        estado=true;
    }
    if (dht11_temperatura<=29){
        estado=false;
    }
}
```

Actividad a. del capítulo 4.4

$R_b = 2,8K\Omega$

Ecuación Ley LKV

$$3.3 \text{ Voltios} - V_{Rb} - V_{be} = 0$$

$$3.3 - (I_b * 2,8K\Omega) - 0,7 = 0$$

Despejando la resistencia de base, se tiene:

$$\frac{3.3 - 0,7}{2,8K\Omega} = I_b$$

$$928\mu A = I_b$$

Ecuación para Corriente de saturación del transistor " I_c "

$$I_{\text{saturación}} = I_c = 100 * 928\mu A$$

$$I_{\text{saturación}} = I_c = 92,8mA$$

Esta intensidad de corriente que fluirá por el embobinado y que se encuentra entre 72mA y 120mA, garantizará la saturación del transistor y la conmutación del contacto en relé.

Actividad b. del capítulo 4.4

```
/*  
Variable de cosa "Actuador"  
bool interruptor1;  
bool interruptor2;  
bool interruptor3;  
bool interruptor4;  
*/  
#include "thingProperties.h"  
#define led1 // Pin GPIO 2 para bombillo 1  
#define led2 // Pin GPIO 3 para bombillo 2  
#define led3 // Pin GPIO 4 para bombillo 3  
#define led4 // Pin GPIO 5 para bombillo 4  
  
void setup() {  
  Serial.begin(9600);  
  delay(1500);  
  initProperties();  
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);  
  setDebugMessageLevel(2);  
  ArduinoCloud.printDebugInfo();  
  pinMode(led1, OUTPUT);
```

```

pinMode(led2, OUTPUT);
pinMode(led3, OUTPUT);
pinMode(led4, OUTPUT);
}
void loop() {
  ArduinoCloud.update();
  //Se puede colocar esta decisión y/o dejar la decisión en la función
  //onInterruptorChange()
  if (interruptor1==true){
    digitalWrite(led,HIGH);
  }
  if (interruptor1==false){
    digitalWrite(led,LOW);
  }
  if (interruptor2==true){
    digitalWrite(led2,HIGH);
  }
  if (interruptor2==false){
    digitalWrite(led2,LOW);
  }
  if (interruptor3==true){
    digitalWrite(led3,HIGH);
  }
  if (interruptor3==false){
    digitalWrite(led3,LOW);
  }
  if (interruptor4==true){
    digitalWrite(led4,HIGH);
  }
  if (interruptor4==false){
    digitalWrite(led4,LOW);
  }
}

```

Actividad a. del capítulo 5.4

Ecuación 5. Valor de temperatura medida

$$V_{temperatura} = (V_{decimal} * 1023) / 4096$$

$$730 = (V_{decimal} * 1023) / 4096$$

$$(730 * 4096) / 1023 = V_{decimal}$$

$$2922,8 = 2923 = V_{decimal}$$

Ecuación 6. Valor decimal a la salida del ADC

$$V_{decimal} = V_{entrada} / LSB$$

$$2923 = V_{entrada} / 805.6 \mu V$$

$$2923 * 805.6 \mu V = V_{entrada}$$

$$2,35V = V_{entrada}$$

Actividad b. del capítulo 5.4

Se modifican los valores en las estructuras de decisión existentes, se agrega una decisión y se modifican los mensajes de las variables tipo string.

```
// MAX6675 library - Version: Latest
#include <max6675.h>
/*
  Arduino IoT Cloud Variables description
  String alarma;
  int temperatura;
  bool alta;
  bool baja;
  bool media;
*/
//Pines de conexión a ESP32
int termoSO = 19;
int termoCS = 23;
int termoSCK = 5;
MAX6675 thermocouple(termoSCK, termoCS, termoSO);
#include "thingProperties.h"
void setup() {
  Serial.begin(9600);
  Serial.println("MAX6675 test");
  delay(1500);
  initProperties();
  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugLogLevel(2);
```

```

    ArduinoCloud.printDebugInfo();
}

void loop() {
    ArduinoCloud.update();
    // Your code here
    Serial.print("C = ");
    Serial.println(thermocouple.readCelsius());
    Serial.print("F = ");
    Serial.println(thermocouple.readFahrenheit());
    temperatura=thermocouple.readCelsius();
    // For the MAX6675 to update, you must delay AT LEAST 250ms between reads!
    delay(2000);
    // Estructuras de decisi3n para evaluar rangos de temperatura
    if (temperatura<=60){
        baja=true;
        media=false;
        alta=false;
        alarma="Baja temperatura";
    }
    if (temperatura>60 && temperatura<=120 ){
        baja=false;
        media=true;
        alta=false;
        alarma="Temperatura media";
    }
    if (temperatura>120 && temperatura<=250 ){
        baja=false;
        media=true;
        alta=false;
        alarma="Alta Temperatura";
    }

    if (temperatura>250 ){
        baja=true;
        media=true;
        alta=true;
        alarma="Peligro por muy Alta temperatura";
    }
}

void onBajaChange() {

}

void onMediaChange() {

}
}

```

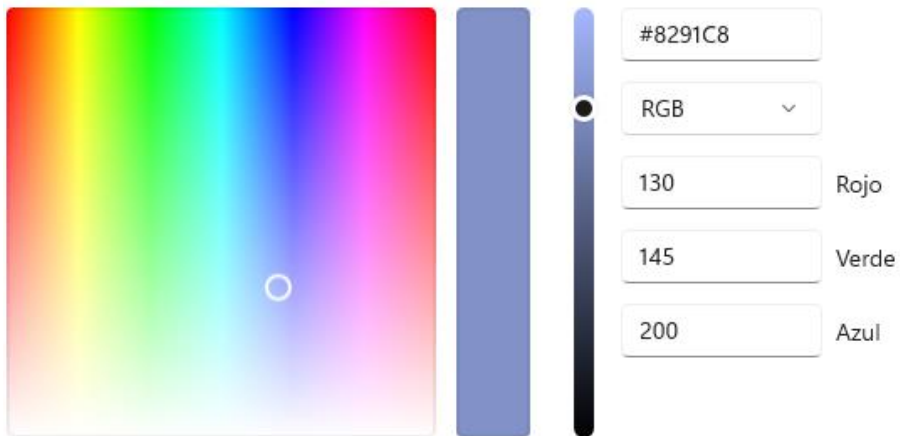
```

void onAltaChange() {
}

void onAlarmaChange() {
}

```

Actividad a. del capítulo 6.4



Para Led RGB de cátodo común los valores decimales de cada color deben ser:

Rojo= 130=10000010

Verde=145=10010001

Azul=200=11001000

a. Para Led RGB de ánodo común los valores decimales de cada color deben ser:

Rojo=01111101=125

Verde=01101110=110

Azul=00110111=55

b. Ancho del pulso del color verde

$$X_{\text{Ciclo Útil}} (\%) = (145 * 100\%) / 255$$

$$X_{\text{Ciclo Útil}} (\%) = 56,86\%$$

Intensidad de Corriente en led verde

$$I_{\text{Led}}(mA) = \frac{20mA * 56,86\%}{100\%}$$

$$I_{\text{Led}}(mA) = 11,37 \text{ mA}$$

Actividad b. del capítulo 6.4

```
//Arduino IoT Cloud Variables description
//CloudColoredLight led; bool max_r; bool max_g; bool max_b;
#include "thingProperties.h"
const int LED_Rojo_PIN = 4;
const int LED_Verde_PIN = 5;
const int LED_Azul_PIN = 18;
//Variables de tipo entero a 8 bits
uint8_t r, g, b;
//variables que tomaran dato invertido r,g,b por tener led tricolor de ánodo común
uint8_t mr, mg, mb;
void setup() {
  Serial.begin(9600);
  delay(1500);
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  pinMode(LED_Rojo_PIN, OUTPUT);
  pinMode(LED_Verde_PIN, OUTPUT);
  pinMode(LED_Azul_PIN, OUTPUT);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
}
void loop() {
  ArduinoCloud.update();
}
void onLedChange() {
  //Método para obtener los valores a 8 bits de rojo, verde y azul
  led.getValue().getRGB(r, g, b);
  //Invertir el valor decimal entre 0 y 255
  mr=~r;
  mg=~g;
  mb=~b;
  //Decisión cuando estan los valores en 0 y se invierten quedando en 255 por ser
  //Led tricolor de ánodo común
  if(mr == 255 && mg == 255 && mb == 255){
    analogWrite(LED_Rojo_PIN, 255);
    analogWrite(LED_Verde_PIN, 255);
    analogWrite(LED_Azul_PIN, 255);
  }
  else{
    analogWrite(LED_Rojo_PIN, mr);
    analogWrite(LED_Verde_PIN, mg);
    analogWrite(LED_Azul_PIN, mb);
  }
  if(mr == 0 && mg != 0 && mb != 0){
    max_r=true;
    max_g=false;
```

```

max_b=false;
}
if(mr != 0 && mg == 0 && mb != 0){
max_r=false;
max_g=true;
max_b=false;
}
if(mr != 0 && mg != 0 && mb == 0){
max_r=false;
max_g=false;
max_b=true;
}
if(mr == 0 && mg == 0 && mb == 0){
//niveles máximos de rojo, verde y azul, siendo producida la luz blanca
max_r=true;
max_g=true;
max_b=true;
}
}
}

```

Actividad a. del capítulo 7.4

Ecuación 10

$$V_{decimal} = V_{entrada}/LSB$$

$$V_{decimal} * 805.6 \text{ uV} = V_{entrada}$$

Ecuación 12. Ecuación general de recta línea de acuerdo a función map

$$Y = 0,0246X - 1,1174$$

$$Y(\%) = 0,0246X(V_{decimal}) - 1,1174$$

$$(31 + 1,1174)/0,0246 = X(V_{decimal})$$

$$1306 = X(V_{decimal})$$

Reemplazando Valor decimal en ecuación 10

$$1306 * 805.6 \text{ uV} = V_{entrada}$$

$$1,052 \text{ Voltios} = V_{entrada}$$

Actividad b. del capítulo 7.4

```
/*
  int humedad;
  bool alta_humedad;
  bool media_humedad;
  bool seco;
*/
#include "thingProperties.h"
#define Sensor_pin 34 //Canal ADC1_6
uint16_t humedadint;
#define led1 4 // Pin GPIO 4 para led1
#define led2 5 // Pin GPIO 5 para led2
#define led3 18 // Pin GPIO 18 para led3

void setup() {
  Serial.begin(9600);
  delay(1500);
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
  pinMode(led3, OUTPUT);
}

void loop() {
  ArduinoCloud.update();
  humedadint=analogRead(Sensor_pin); //lectura analógica
  Serial.println("Humedad (Conversion decimal)= ");
  Serial.println(humedadint);
  humedad= map(humedadint,0,4096,0,100); //amplificación por 100 para mantener
decimales
  Serial.println("Humedad (Ajuste porcentual)= ");
  Serial.println(humedad);
  delay(2000);
  if (humedad<30){
    Serial.println("Suelo con alta humedad");
    alta_humedad=true;
    media_humedad=false;
    seco=false;
    digitalWrite(led1,HIGH);
    digitalWrite(led2,LOW);
    digitalWrite(led3,LOW);
  }
  if (humedad>=30 && humedad<42){
    Serial.println("Suelo con humedad media");
```

```
    alta_humedad=false;
    media_humedad=true;
    seco=false;
    digitalWrite(led1,LOW);
    digitalWrite(led2, HIGH);
    digitalWrite(led3,LOW);
  }
  if (humedad>=42){
    Serial.println("Suelo seco");
    alta_humedad=false;
    media_humedad=false;
    seco=true;
    digitalWrite(led1,LOW);
    digitalWrite(led2,LOW);
    digitalWrite(led3,HIGH);
  }
}
void onAltaHumedadChange() {}
void onMediaHumedadChange() {}
void onSecoChange() {}
```

Actividad a. del capítulo 8.4

Ecuación 13. Pasos completos según grados de rotación

$$\text{Número de Pasos Completos} = \frac{30^\circ * 2048}{360} = 171$$

Ecuación 14. Tiempo transcurrido en la rotación

$$\text{Tiempo transcurrido (Seg)} = \frac{30^\circ * 60}{2 * 360^\circ} = 2,5 \text{ seg}$$

Actividad a. del capítulo 9.4

Peso de objeto=3500 gr

Escala=-423

Ecuación 15. Escala de calibración para bascula de peso

$$\text{Escala} = \frac{\text{Valor de lectura de Tara}}{\text{Peso real conocido (Kg o gramos)}}$$

$$-423 = \frac{\text{Valor de lectura de Tara}}{3500}$$

$$-423 * 3500 = \text{Valor de lectura de Tara}$$

$$-1.480.500 = \text{Valor de lectura de Tara}$$

Conversión de número decimal positivo a número binario

$$1480500_{10} = x_2$$

$$1480500_{10} = 000101101001011100110100_2$$

Conversión complemento a 2 de número binario

$$000101101001011100110100_2$$

$$111010010110100011001100_2$$

La trama de datos de 24 bits transmitida por la interfaz digital es:

$$\begin{array}{c} \nearrow \text{MSB} \quad 111010010110100011001100 \quad \nwarrow \text{LSB} \end{array}$$

El primer bit transmitido es el MSB=1 que corresponde al signo por ser número negativo y el último bit transmitido es LSB=0.

Esta trama de datos de 24 bits de complemento a 2, corresponde al número decimal negativo -1.480.500

Actividad b. del capítulo 9.4

```
#include <HX711.h>
const int DOUT_PIN = 19;
const int CLK_PIN = 18;
HX711 balanza;
int pesogr;
int tiempo=0;
/*
  Arduino IoT Cloud Variables description
  int peso;
  bool alarma;
  */

#include "thingProperties.h"

void setup() {
  // Initialize serial and wait for port to open:
  Serial.begin(9600);
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found
  delay(1500);

  // Defined in thingProperties.h
  initProperties();

  // Connect to Arduino IoT Cloud
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);

  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  balanza.begin(DOUT_PIN,CLK_PIN);
  Serial.print("Lectura del valor del ADC: ");
  Serial.println(balanza.read());
  Serial.println("No ponga ningun objeto sobre la balanza");
  Serial.println("Destarando...");
  Serial.println("...");
  balanza.set_scale(-423); // Establecemos la escala
  balanza.tare(20); //El peso actual es considerado Tara.

  Serial.println("Listo para pesar");

}

void loop() {
  ArduinoCloud.update();
  // Your code here
  Serial.print("Valor de lectura: t");
```

```

Serial.println(balanza.get_value(10),0);
pesogr=(balanza.get_value(10))/-423;
//Mostrar en dashboard
peso=pesogr;
//Mostrar en Monitor de plataforma
Serial.println(pesogr);
Serial.print("Peso: ");
Serial.print(balanza.get_units(20),3);
Serial.println(" gr");
if(pesogr>=500 && pesogr <=900){
Serial.println("Se encuentra celular en celda de carga ");
tiempo=0;
alarma=false;
delay(1000);
}
if(pesogr<500 && tiempo<=1800){
Serial.println("No se encuentra el celular en celda de carga y alarma apagada");
tiempo=tiempo+1;
Serial.println(tiempo);
alarma=false;
delay(1000);
}
if(pesogr<500 && tiempo>1800){
Serial.println("No se encuentra el celular en celda de carga y alarma encendida ");
tiempo=tiempo+1;
Serial.println(tiempo);
alarma=true;
delay(1000);
}
}
}

```

Actividad c. del capítulo 9.4

En el circuito solamente se cambia la celda de carga YZC-131A que mide hasta 5 Kg de peso, por la celda de carga de referencia YZC-1B con capacidad de soportar hasta máximo 50Kg.

<https://www.digikey.com/htmldatasheets/production/2104094/0/0/1/114990100.html>

Actividad a. del capítulo 10.4

Velocidad de la luz expresada en cm/uS:

343 m/s= 0,0343 cm/uS

$$11,9 \text{ cm} = \frac{\text{Techo} \times 0,0343 \frac{\text{cm}}{\text{uS}}}{2}$$

$$\text{Techo} = \frac{11,9 \times 2}{0,0343} = 693,8 \text{ uS}$$

Actividad b. del capítulo 10.4

```
/*  
  float dista_cm;  
  float dista_pulgadas;  
  bool estado;  
*/  
#include "thingProperties.h"  
const int trig_Pin = 5;  
const int echo_Pin = 18;  
//define velocidad de sonido y equivalente a de 1 cm en pulgadas  
#define vel_sonido 0.0343  
#define cm_a_pulgada 0.393701  
long duracion;  
float distanciaCm;  
float distanciaPulgadas;  
void setup() {  
  // Initialize serial and wait for port to open:  
  Serial.begin(9600);  
  // This delay gives the chance to wait for a Serial Monitor without blocking if none is found  
  delay(1500);  
  pinMode(trig_Pin, OUTPUT); // Configuración de pin trigger como salida  
  pinMode(echo_Pin, INPUT); // Configuración de pin echo como entrada  
  // Defined in thingProperties.h  
  initProperties();  
  // Connect to Arduino IoT Cloud  
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);  
  setDebugMessageLevel(2);  
  ArduinoCloud.printDebugInfo();  
}  
void loop() {  
  ArduinoCloud.update();  
  // Your code here  
  // En bajo Trigger  
  digitalWrite(trig_Pin, LOW);  
  delayMicroseconds(2);  
  // Estado alto del pulso por trigger en 30 uS  
  digitalWrite(trig_Pin, HIGH);
```

```

delayMicroseconds(30);
digitalWrite(trig_Pin, LOW);
// Lectura del echo, tiempo en uS de retorno de la onda de sonido
duracion = pulseIn(echo_Pin, HIGH);
// Calculo de distancia en cm
distanciaCm = duracion * vel_sonido/2;
// Conversion a pulgadas
distanciaPulgadas = distanciaCm * cm_a_pulgada;
// Mostrar distancia en monitor serial
Serial.print("Distancia (cm): ");
Serial.println(distanciaCm);
Serial.print("Distancia (pulgadas): ");
Serial.println(distanciaPulgadas);
//Mostrar en dashboard
dista_cm=distanciaCm;
dista_pulgadas=distanciaPulgadas;
//Llamado a función parqueo()
parqueo();
delay(1000);
}
//Función de decisión para ocupación de espacio de parqueo
void parqueo(){
  if(distanciaCm<=20){
    Serial.print("Ocupacion de espacio");
    //Mostrar en dashboard
    estado=true;
  }
  else{
    Serial.print("No ocupacion de espacio");
    //Mostrar en dashboard
    estado=false;
  }
}
void onDistaCmChange() {}
void onDistaPulgadasChange() {}
void onEstadoChange() {}

```

Actividad a. del capítulo 12.4

Ecuación 19. Conversión de Valor ADC a Valor de Voltaje

$$Valor_{Voltaje} = Valor_{ADC} \times \left(\frac{3,7}{4095} \right)$$
$$Valor_{Voltaje} = 2300 \times \left(\frac{3,7}{4095} \right) = 2,07$$

Reemplazando $Valor_{Voltaje}$ en $Vout$ de la ecuación 17, se hallará la “Rs” así:

$$Rs = 5k\Omega \times \frac{5 - Vout}{Vout}$$
$$Rs = 5k\Omega \times \frac{5 - 2,07}{2,07} = 7077 \Omega$$

Como último proceso, el resultado de “Rs” se reemplaza en la ecuación 18 para calcular el valor real PPM de la corriente de gas propano detectada por el sensor MQ2.

$$Y(PPM) = 634,98 \times \left(\frac{7077}{8783,78} \right)^{-2,169} = 1016,55$$

Para una concentración de gas propano de 1016 PPM el valor de voltaje a la salida del sensor MQ2 corresponde a 2,07 Voltios DC.

Actividad b. del capítulo 12.4

```
/*  
  Arduino IoT Cloud Variables description  
  float valores_gas;  
  int condicion;  
  bool presencia;  
*/  
#include "thingProperties.h"  
int Sensor_gas = 34;  
int LED = 2;  
void setup() {  
  Serial.begin(9600);  
  delay(1500);  
  pinMode(LED, OUTPUT);  
  // Defined in thingProperties.h  
  initProperties();  
  // Connect to Arduino IoT Cloud  
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);  
  setDebugMessageLevel(2);  
  ArduinoCloud.printDebugInfo();  
}  
void loop() {
```



```

ArduinoCloud.update();
float sensor_dato = analogRead(Sensor_gas); // Lectura ADC de la salida
//analógica del sensor
float voltaje = sensor_dato * (3.7 / 4095.0); //Conversión de lectura ADC a valor
//de voltaje
float Rs = 5000 * ((5-voltaje)/voltaje); //Cálculo de Rs con una RL=5000 Ohms
float gasppm = 634,98*pow(Rs/8783.78,-2.169); //Cálculo de PPM de la ecuación
//potencial obtenida
Serial.print("Sensor de Gas: ");
Serial.print(sensor_dato);
Serial.print("\t");
if(condicion==0) {
    if (sensor_dato > 2500) {
        Serial.println("Gas");
        digitalWrite (LED, HIGH) ;
        presencia=true;
        valores_gas =sensor_dato;
    }
    if (sensor_dato <= 2500) {
        Serial.println("No Gas");
        digitalWrite (LED, LOW) ;
        presencia=false;
        valores_gas =sensor_dato;
    }
}
if(condicion==1) {
    if (sensor_dato > 2500) {
        Serial.println("Gas");
        digitalWrite (LED, HIGH) ;
        presencia=true;
        valores_gas=voltaje;
    }
    if (sensor_dato <= 2500) {
        Serial.println("No Gas");
        digitalWrite (LED, LOW) ;
        presencia=false;
        valores_gas=voltaje;
    }
}
if(condicion==2) {
    if (sensor_dato > 2500) {
        Serial.println("Gas");
        digitalWrite (LED, HIGH) ;
        presencia=true;
        valores_gas=gasppm;
    }
    if (sensor_dato <= 2500) {

```

```

    Serial.println("No Gas");
    digitalWrite (LED, LOW) ;
    presencia=false;
    valores_gas=gasppm;
}
}
delay(1000);
}
void onCondicionChange() {}
void onPresenciaChange() {}
void onValoresGasChange() {}

```

Actividad b. del capítulo 13.4

```

// SparkFun MAX3010x Pulse and Proximity Sensor Library - Version: Latest
#include <MAX30105.h>
#include <heartRate.h>
#include <spo2_algorithm.h>
MAX30105 particleSensor;
/*
  Arduino IoT Cloud Variables description
  String mensaje;
  float infrarojo;
  float latidos;
  CloudLocation ubicacion;
*/
const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred
float beatsPerMinute;
int beatAvg;
#include "thingProperties.h"
void setup() {
  Serial.begin(115200);
  delay(1500);
  initProperties();
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);
  setDebugMessageLevel(2);
  ArduinoCloud.printDebugInfo();
  Serial.println("Initializing...");
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST))
  {
    Serial.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady pressure.");

```

```

particleSensor.setup();
particleSensor.setPulseAmplitudeRed(0x0A);
particleSensor.setPulseAmplitudeGreen(0);
}
void loop() {
  ArduinoCloud.update();
  long irValue = particleSensor.getIR();
  if (checkForBeat(irValue) == true)
  {
    long delta = millis() - lastBeat;
    lastBeat = millis();
    beatsPerMinute = 60 / (delta / 1000.0);
  }
  Serial.print("IR=");
  Serial.print(irValue);
  infrarojo=irValue;
  Serial.print(", BPM=");
  latidos=beatsPerMinute;
  Serial.print(beatsPerMinute);
  ubicacion = Location(2.930515, -75.270526);
  if (irValue < 1000){
    Serial.print(" Sin Dedo?");
    mensaje="Sin Dedo";
    Serial.println();
  }
  else{
    mensaje="Presencia de Dedo";
    Serial.println();
  }
}
void onLatidosChange() {}
void onUbicacionChange() {}
void onInfrarojoChange() {}
void onMensajeChange() {}

```

Actividad c. del capítulo 12.4

$$122588_{10} = 01 \ 1101 \ 1110 \ 1101 \ 1100_2$$

El Segundo Byte transmitido por interfaz I2C:

$$11011110_2$$

Actividad a. del capítulo 14.4

3 paneles en serie = 15v y 300mA

3 paneles en serie= 15v y 300 mA

3 paneles en serie= 15v y 300 mA

Las anteriores configuraciones se conectan en paralelo y se tendrá la capacidad máxima de 15v y 900mA.

En total se requieren 9 paneles solares en configuración mixta (3 series y un paralelo)

Actividad b. del capítulo 14.4

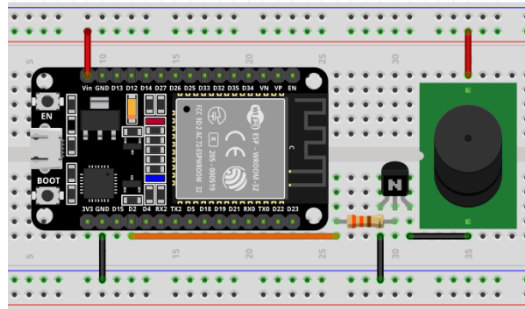
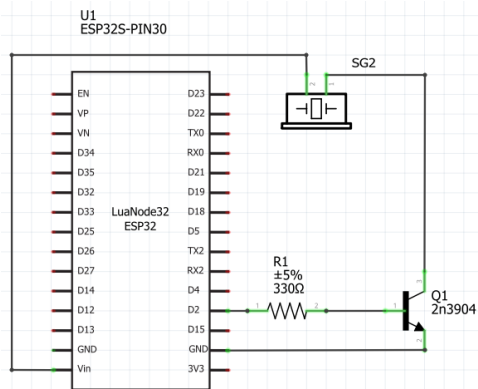
```
/*  
  Arduino IoT Cloud Variables description  
  float bate_vol;  
  int bate_decimal;  
  int bate_milivol;  
  int bate_porcen;  
*/  
#include "thingProperties.h"  
#define Sensor_pin 34 //Canal ADC1_6  
#define sonido 2 // Pin GPIO 2 para indicador sonoro  
void setup() {  
  Serial.begin(115200);  
  // Defined in thingProperties.h  
  initProperties();  
  // Connect to Arduino IoT Cloud  
  ArduinoCloud.begin(ArduinoIoTPreferredConnection);  
  setDebugMessageLevel(2);  
  ArduinoCloud.printDebugInfo();  
  pinMode(sonido, OUTPUT);  
}  
void loop() {  
  ArduinoCloud.update();  
  bate_decimal=analogRead(Sensor_pin); //lectura analógica  
  Serial.println("Nivel de bateria (Conversion decimal)= ");  
  Serial.println(bate_decimal);  
  bate_porcen= map(bate_decimal,0,4095,0,100); //Escala a 100  
  Serial.println("Nivel de bateria porcentual (Escala hasta el 100%)= ");  
  Serial.println(bate_porcen);  
  bate_milivol= map(bate_decimal,0,4095,0,3900); //Escala a 3900 mv  
  Serial.println("Nivel de bateria en milivoltios (Escala hasta el 3300 mv)= ");  
  Serial.println(bate_milivol);  
  bate_vol=bate_milivol/1000.0;  
  if (bate_milivol <=2500){  
    digitalWrite(sonido, HIGH);  
  } else{  
    digitalWrite(sonido, LOW);  
  }  
  delay(2000);  
}
```

```

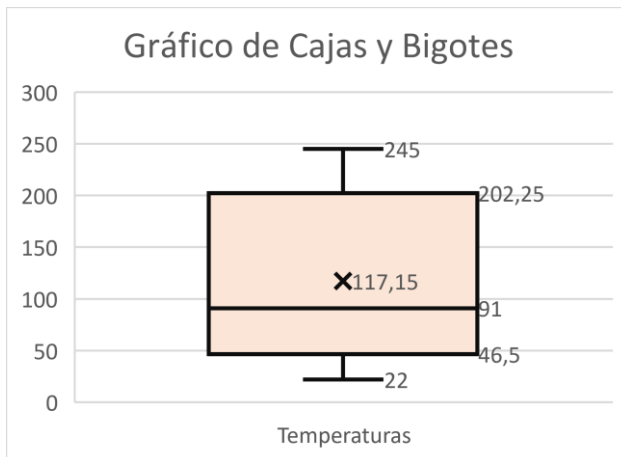
}
void onBateDecimalChange() {}
void onBatePorcenChange() {}
void onBateMilivolChange() {}
void onBateVolChange() {}

```

Actividad c. del capítulo 14.4



Actividad a. del capítulo 15.5



Dato máximo: 245
 Dato mínimo: 22
 Q₁: 46,5
 Q₂ (mediana): 91
 Q₃: 202,25
 Media o promedio: 117,15