Laboratorio de microcontroladores.
Alejando Ossio Díaz, A01209122.
Gerardo Daniel Naranjo Gallegos, A01209499.
Roberto Figueroa Saavedra, A01209689.
05 de diciembre de 2016.
Práctica final.

Proyecto final.

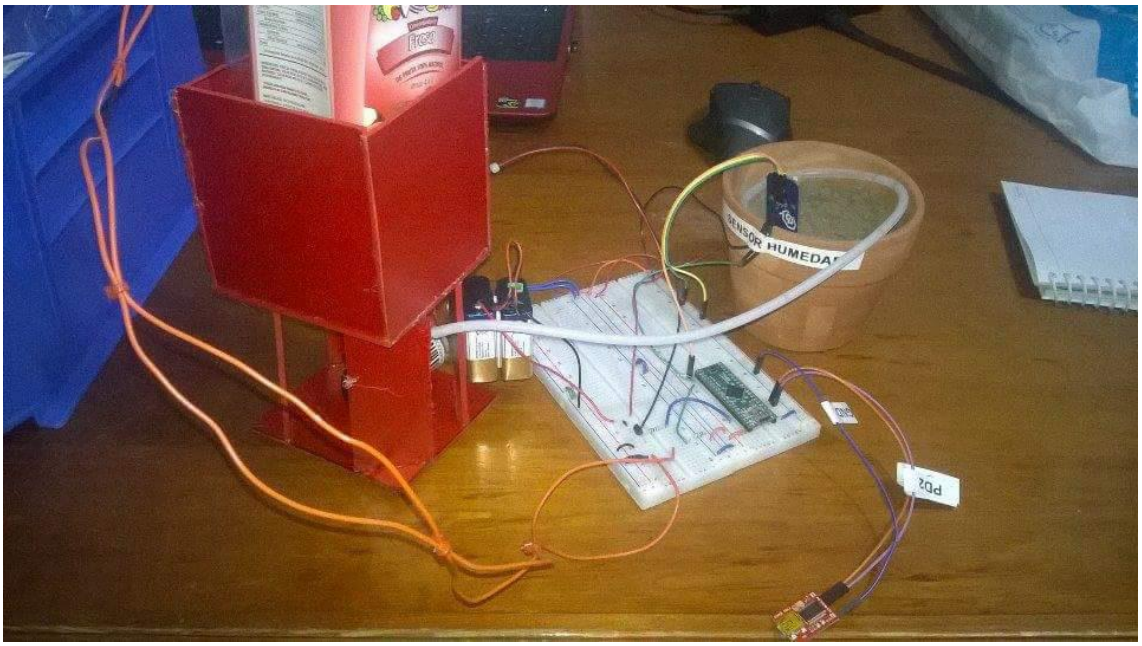
Objetivo:

Elaborar un sistema de riego, capaz de regular la humedad de una planta, así como llevar registro de la misma, y examinar la temperatura en tiempo real.

Descripción:

Proyecto:

El proyecto se compone de una planta, un circuito, descrito a continuación; una interfaz en la computadora, descrita más adelante; un depósito de agua (color rojo), que sirve para almacenar el agua con la que se regará la planta; una bomba (debajo del depósito de agua), que sirve para regar la planta; baterías, que alimentan a la bomba de agua; un sensor de humedad; un sensor de temperatura; y, finalmente, un puerto de comunicación serial. Adjuntamos una fotografía del proyecto.



Circuito:

El circuito del proyecto consiste en utilizar el microcontrolador, cuyo código fuente se explica más adelante, un sensor de temperatura, un sensor de humedad, dos placas de cobre, baterías y una bomba de agua.

El microcontrolador, como su nombre lo indica, es quien se encarga de controlar el circuito. Recibe todos los datos de los sensores y de la interfaz (mediante serial), para poder realizar acciones en base al código fuente.

El sensor de temperatura utilizado es un LM35, tiene una gran precisión de medición en un rango de -55°C hasta 150°C. El sensor se conecta a tierra y voltaje y le envía al microcontrolador un voltaje que varía en función de la temperatura. Se colocó en un extremo de la maceta. (Texas Instruments, 2016)

El sensor de humedad de tierra tiene un funcionamiento similar, cuenta con un comparador de voltaje interno, en donde dependiendo de la humedad medida envía un voltaje correspondiente. En este sensor en particular se puede seleccionar entre una salida de la señal analógica o digital. (AG Electrónica S.A. de C.V., 2014)

Las baterías son para alimentar a la bomba de agua. Cabe mencionar que se coloca un diodo de protección en el circuito.

La bomba de agua succiona agua del depósito y la manda a la maceta. Es activada por el microcontrolador y alimentada con las baterías; basta con conectar a voltaje y tierra.

Finalmente, se utilizan dos placas de cobre que se colocan dentro del depósito de agua. Su función, es determinar cuánta agua contiene.

Código fuente:

El código utilizado en el microcontrolador debe encargarse de varias cosas. Debe interpretar los datos mandados desde la interfaz en la computadora, mediante serial, en donde el microcontrolador recibe el nivel deseado de humedad en la planta.

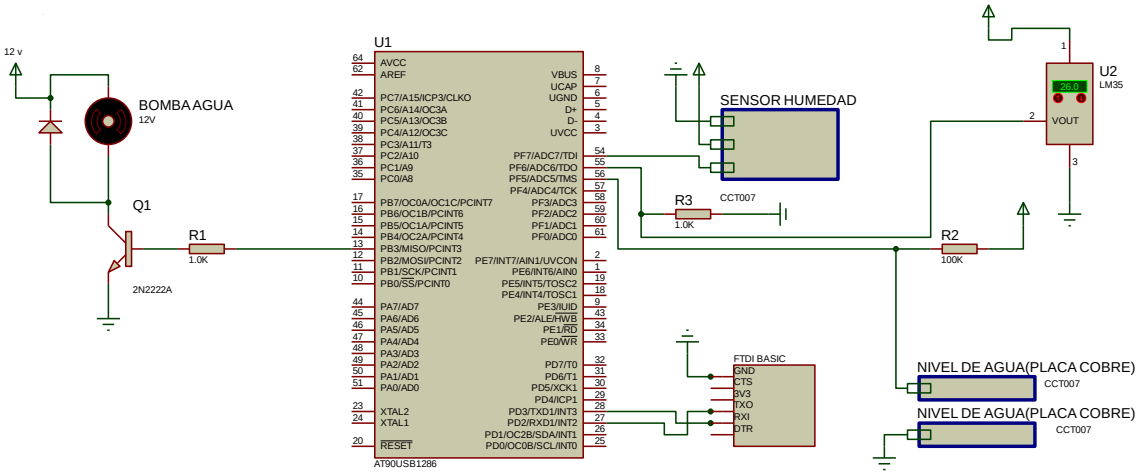
Con ayuda de un sensor de humedad, localizado en la planta, recibe también el nivel de humedad en tiempo real y, gracias a esto, se puede realizar la validación, si el nivel de humedad es correcto o no; en caso de necesitar más agua, el microcontrolador debe encender la bomba que envía agua desde el depósito hasta la planta y, al detectar que se alcanzó el nivel de humedad deseado, apagará la bomba.

Interfaz y código en Processing:

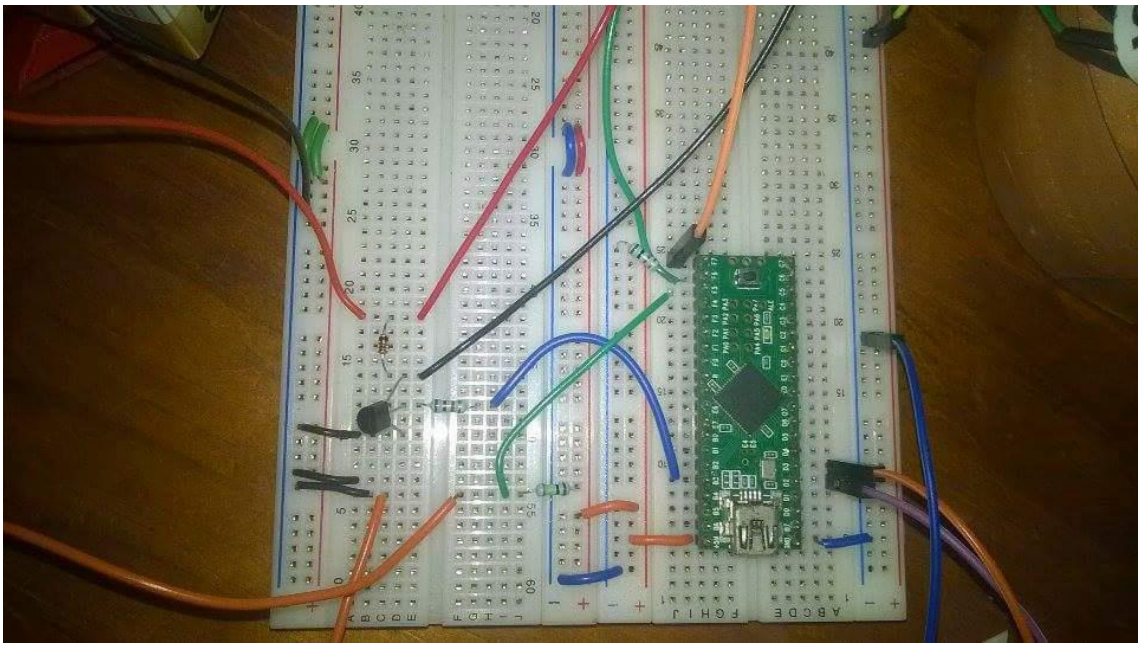
Respecto al código utilizado en Processing, se necesita generar una interfaz, en la cual, se puedan realizar las siguientes acciones:

a) Seleccionar el nivel de porcentaje de humedad necesario para la planta: esto con la intensión de dar al usuario la posibilidad de elegir la humedad porque cada planta requiere de un porcentaje distinto.
b) Mostrar en un cuadro de texto las condiciones del depósito de agua, es decir, mostrar si tiene la suficiente cantidad de agua para regar la planta o si requiere ser rellenado.
c) Graficar un registro de la humedad que ha tenido la planta, que se actualice cada cierto tiempo, por ejemplo, cada cinco minutos.
d) Observar en tiempo real la temperatura a la que está expuesta la planta.

Diagrama eléctrico:



Fotografía del circuito:



Código Processing:

El código implementado en Processing es el siguiente:

```
// Processing code
import processing.serial.*;
Serial port;
int moveY,n=0,temp=0,serialCount=0,tanque=0,porcentaje=0,i=0,g=0,x=10;
float k=0.0;
PFont f,a;
int[] porc = new int[11];  //Arreglo donde se guardan los porcentajes por cada 5 mins
int[] data = new int[5];  //Arreglo donde se guardaran los datos envidos por el microcontrolador
```

```
float[] graf = new float[11];

void setup() {
  size(1000, 600);
  f= createFont("Arial",16,true);
  a= createFont("Arial",32,true);
  println("Available serial ports:");
  println(Serial.list());
  port = new Serial(this, "COM4", 9600);
}

void serialEvent (Serial port) {
  // get the ASCII string:
  String T = port.readStringUntil(13);
  if (T != null) {
    // trim off any whitespace:
    T = trim(T);

    data[serialCount]=int(T);    //Procedimiento encargado de la comunicacion
serial
    serialCount++; //Se envia en orden desde el micro los siguientes datos
    temp=data[0]; // La temperatura dada por el sensor
    tanque=data[1]; //Que tan lleno se encuentra el tanque de agua
    porcentaje=data[2]; //El porcentaje del agua en ese momento
    g=data[3]; //Variable que determina cada que se debe imprimir la
grafica(cada 5 mins)
    i=data[4]; //Variable externa que controla el arreglo donde se guardan los
datos que se graficaran

    switch(g){
      case 0:
        break;
      case 1:
        porc[i]=porcentaje; //se asigna el porcentaje del agua cada 5 mins
    }

    if(serialCount>4)
      serialCount=0; // se reinician los datos
  }
}

void draw() {
  background(204,204,255);
  stroke(0);
  strokeWeight(1);

  //Barra indicadora de porcentaje de agua
  fill(0,0,255);
  rect(10,20,70,500);

  //Indicador de porcentaje
```

```
fill(255,0,0);
rect(10,moveY+20,70,20);

//Texto porcentaje
fill(255,255,255);
textFont(f,16);
text(n+" %",23,moveY+37);

//Texto temperatura
fill(223,29,29);
textFont(a,32);
text(temp+" °C",900,37);// Pendiente

//Marco gris
fill(96,96,96);
rect(100,20,780,500);

//Fondo grafica
fill(0);
rect(130,50,720,335);

//Lineas grafica
stroke(255);
line(160,70,160,365); //vertical
line(160,365,815,365); //horizontal

//Cuadro dialogo
fill(255,255,255);
rect(130,390,720,100);

//Porcentaje humedad
textFont(f,16);
text("%",155,67);
text("0",145,380);
text("10",136,350);
text("20",136,320);
text("30",136,290);
text("40",136,260);
text("50",136,230);
text("60",136,200);
text("70",136,170);
text("80",136,140);
text("90",136,110);
text("100",130,80);

//Tiempo
text("Min",820,370);
text("5",220,380);
text("10",283,380);
text("15",348,380);
text("20",413,380);
```

```
  text("25",478,380);
  text("30",543,380);
  text("35",608,380);
  text("40",673,380);
  text("45",738,380);
  text("50",800,380);

  //Texto cuadro de dialogo
  fill(0);
  textFont(f,32);
  if(tanque==0){
    text("Tanque encima de la mitad",132,422);

  }
  else{
    text("Tanque debajo de la mitad",132,422);
    text("Llenalo",132,456);
  }

  //Lineas
  stroke(0,255,0); //linea de grafica
  strokeWeight(2);
  switch(porc[i]){ //Variable que determina la altura de cada porcentaje
    case 0:
      k=0; // 0%
    case 10:
      k=-2.0;// 10%
      break;
    case 20:
      k=-2.5;// 20%
      break;
    case 30:
      k=-2.7;// 30%
      break;
    case 40:
      k=-2.75;// 40%
      break;
    case 50:
      k=-2.8;// 50%
      break;
    case 60:
      k=-2.85;// 60%
      break;
    case 70:
      k=-2.85;// 70%
      break;
    case 80:
      k=-2.88;// 80%
      break;
    case 90:
      k=-2.88;// 90%
```

```
      break;
    case 100:
      k=-2.9;// 100%
      break;
  }

    graf[i]=k; //se asigna a un arreglo para poder graficarlo posteriormente


  //Lineas graficadas
      line(160,graf[0]*porc[0]+365,224,graf[1]*porc[1]+365); //De 0 a 5
      line(224,graf[1]*porc[1]+365,290,graf[2]*porc[2]+365); //De 5 a 10
      line(290,graf[2]*porc[2]+365,356,graf[3]*porc[3]+365);//De 10 a 15
      line(356,graf[3]*porc[3]+365,422,graf[4]*porc[4]+365);//De 15 a 20
      line(422,graf[4]*porc[4]+365,486,graf[5]*porc[5]+365);//De 20 a 25
      line(486,graf[5]*porc[5]+365,552,graf[6]*porc[6]+365);//De 25 a 30
      line(552,graf[6]*porc[6]+365,618,graf[7]*porc[7]+365);//De 30 a 35
      line(618,graf[7]*porc[7]+365,682,graf[8]*porc[8]+365);//De 35 a 40
      line(682,graf[8]*porc[8]+365,746,graf[9]*porc[9]+365);//De 40 a 45
      line(746,graf[9]*porc[9]+365,809,graf[10]*porc[10]+365);//De 45 a 50

//Detección de nivel de porcentaje con cada if
  //Se le manda una letra al micro dependiendo del porcentaje deseado
  if(n==0)
    port.write('a');
  if(n>0 && n<=10)
    port.write('b');
  if(n>10 && n<=20)
    port.write('c');
  if(n>20 && n<=30)
    port.write('d');
  if(n>30 && n<=40)
    port.write('e');
  if(n>40 && n<=50)
    port.write('f');
  if(n>50 && n<=60)
    port.write('g');
  if(n>60 && n<=70)
    port.write('h');
  if(n>70 && n<=80)
    port.write('i');
  if(n>80 && n<=90)
    port.write('j');
  if(n>90 && n<=100)
    port.write('k');
}

void mouseDragged(){
  //Se limita la parte donde se movera el porcentaje de agua del usuario
  //Se calcula la variable n dependiendo de la posicion del maouse
  if((mouseX>=10 && mouseX<=80) && (mouseY>=-10 && mouseY<=480)){
```

```
   if(mouseY<0 && mouseY>=-10)
     n=((mouseY+10)*100/(500));
   else{
     n=((mouseY+20)*100/(500));
     moveY=mouseY;
   }
  }
}
```

Código fuente:

El código utilizado en el microcontrolador es el siguiente:

```
#include <90USB1286.h>
#include <delay.h>
#include <stdio.h>
#include <io.h>
#include "display.h"
#define ADC_VREF_TYPE 0x40

 float voltaje,temperatura,tinaco;
 unsigned int g=0,porcentaje,n=0,contador=0,tinacoN=0,d=0;
 char a=0;

// Timer1 output compare A interrupt service routine
interrupt [TIM1_COMPA] void timer1_compa_isr(void)
{

   printf("%d\n\r",(int)temperatura);              //MANDAR VALOR TEMPERATURA
   printf("%d\n\r",tinacoN);                      //porcentaje tinaco lleno
   printf("%d\n\r",(int)porcentaje);               //porcentaje humedad
   if(contador==10){
      a=1;
      printf("%d\n\r",a);
      printf("%d\n\r",g);                        //contador para graficar en terminal de
processing
      contador=0;
      g++;
   }
   else{
      a=0;
      printf("%d\n\r",a);
      printf("%d\n\r",g);
      contador++;
   }
   if(g>10){
      g=0;
   }
}

// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
```

```
{
ADMUX=adc_input | (ADC_VREF_TYPE & 0xff);
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40;
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0);
ADCSRA|=0x10;
return ADCW;
}

//calculo de porcentaje de humedad
int CalculaPorcentaje(float v){
int p;
if(voltaje>0.0 && voltaje<=0.7){          //porcentaje 100%
p=100;
}

if(v>0.7 && v<=1.0){          //porcentaje 90%
        p=90;
}

if(v>1.0 && v<=1.5){          //porcentaje 80%
        p=80;
}

if(v>1.5 && v<=2.0){          //porcentaje 70%
        p=70;
}

if(v>2.0 && v<=2.5){          //porcentaje 60%
        p=60;
}

if(v>2.5 && v<=3.0){          //porcentaje 50%
        p=50;
}

if(v>3.0 && v<=3.5){          //porcentaje 40%
        p=40;
}

if(v>3.5 && v<=4.0){          //porcentaje 30%
        p=30;
}

if(v>4.0 && v<=4.5){          //porcentaje 20%
        p=20;
}
```

```
if(v>4.5 && v<=4.9){        //porcentaje 10%
        p=10;
}

if(v>4.9 && v<=5.2){        //porcentaje 0%
        p=0;
}
return p;
}

void main(){
PORTF.7=1;
PORTF.6=1;
PORTF.5=1;
DDRB.3=1;

// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 31.250 kHz
// Mode: CTC top=OCR1A
// OC1A output: Discon.
// OC1B output: Discon.
// OC1C output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: On
// Compare B Match Interrupt: Off
// Compare C Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x0B;
OCR1AH=15624/256;
OCR1AL=15624%256;

// Timer/Counter 1 Interrupt(s) initialization
TIMSK1=0x02;

// Global enable interrupts
#asm("sei")

// USART1 initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART1 Receiver: On
// USART1 Transmitter: On
// USART1 Mode: Asynchronous
// USART1 Baud Rate: 9600
UCSR1A=0x00;
UCSR1B=0x18;
UCSR1C=0x06;
UBRR1H=0x00;
```

```
UBRR1L=0x0C;

// ADC initialization
// ADC Clock frequency: 125.000 kHz
// ADC Voltage Reference: AVCC pin
// ADC High Speed Mode: Off
// Digital input buffers on ADC0: On, ADC1: On, ADC2: On, ADC3: On
// ADC4: On, ADC5: Off, ADC6: Off, ADC7: Off
DIDR0=0xE0;
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x84;
ADCSRB&=0x7F;

while(1){

n=getchar();          //RECIBIR PORCENTAJE DE HUMEDAD

tinaco=read_adc(5);     //leer nivel de agua tinaco

if(tinaco>=512){
        tinacoN=0;              //Agua a la mitad
}
else{
        tinacoN=1;              //Mas de la mitad
}

voltaje=read_adc(7)/204.8;            //LECTURA VOLTAJE SENSOR DE HUMEDAD
porcentaje=CalculaPorcentaje(voltaje);  //PORCENTAJE CALCULADO

temperatura=((read_adc(6)/1024.0)*500.0)-6.6;         //OBTENER LECTURA DEL
SENSOR DE TEMP


if(n=='a'){         //10%
        PORTB.3=0;
}

if(n=='b'){         //10%
        if(porcentaje<10)
                PORTB.3=1;
        else
                PORTB.3=0;
}

if(n=='c'){         //20%
        if(porcentaje<20)
                PORTB.3=1;
        else
                PORTB.3=0;
}
```

```
if(n=='d'){       //30%
        if(porcentaje<30)
                PORTB.3=1;
        else
                PORTB.3=0;
}

if(n=='e'){       //40%
        if(porcentaje<40)
                PORTB.3=1;
        else
        PORTB.3=0;
}
                                //CASOS PORCENTAJES DE HUMEDAD RECIBIDOS POR
USUARIO
if(n=='f'){       //50%
if(porcentaje<50)
PORTB.3=1;
else
PORTB.3=0;
}

if(n=='g'){       //60%
if(porcentaje<60)
PORTB.3=1;
else
PORTB.3=0;
}

if(n=='h'){       //70%
if(porcentaje<70)
PORTB.3=1;
else
PORTB.3=0;
}

if(n=='i'){       //80%
if(porcentaje<80)
PORTB.3=1;
else
PORTB.3=0;
}

if(n=='j'){       //90%
if(porcentaje<90)
PORTB.3=1;
else
PORTB.3=0;
}

if(n=='k'){       //100%
```

```
if(porcentaje<100)
PORTB.3=1;
else
PORTB.3=0;
}
}
}
```

Conclusiones individuales:

Alejandro Ossio:

Con el proyecto pudimos implementar gran parte de los conocimientos adquiridas durante el curso y se aprendió a utilizar un sensor que fue el de humedad en tierra el cual complementamos con un sensor de temperatura y usando el principio del sensor de humedad elaboramos un medidor de nivel de agua. Todo esto mediante el manejo de una terminal en la computadora con Processing en la cual encontramos el único problema al tratar de graficar los datos enviados por el microprocesador por comunicación serial, fuera de este problema, el proyecto cumplió con lo que esperaba y fue interesante llevarlo a cabo debido a las diferentes mediciones de peso que se debieron realizar en la tierra tanto húmeda como seca para obtener lecturas del sensor de humedad.

Gerardo Naranjo:

El proyecto resultó como se esperaba, cumplió nuestros objetivos. El circuito fue relativamente fácil, de hecho, para medir el nivel de agua del contenedor se utilizó el mismo principio del sensor de humedad. En cuanto al código fuente, pusimos en práctica cosas del curso actual y anterior de microcontroladores; su elaboración y funcionamiento no tuvo mayor problema. En cuanto a la interfaz realizada en Processing, requirió de mucho tiempo debido a los dibujos y la gráfica nos dio batalla. Para finalizar, el proyecto ayudó a poner en práctica distintos temas del curso.

Roberto Figueroa:

Con nuestro proyecto pusimos en práctica lo aprendido tanto en micros como en su laboratorio, creando una planta de riego automático con una interfaz interactiva en Processing, esta iba conectada desde el micro a la computadora por medio de comunicación serial, tuvimos un único problema con dibujar la gráfica, que pudimos resolver al último mandando un dato desde el micro.

Referencias:

AG Electrónica S.A. de C.V. (05 de 05 de 2014). *Sensor de humedad con interfaz Grove*. Obtenido de Agspecinfo: http://www.agspecinfo.com/pdfs/I/IM117001.PDF

Texas Instruments. (Agosto de 2016). *LM35 Precision Centigrade Temperature Sensors*. Obtenido de Datasheet: http://www.ti.com/lit/ds/symlink/lm35.pdf