ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

TECNOLOGÍAS DE SOFTWARE PARA ELECTRÓNICA

SISTEMA DE CONTROL DE SENSORES, ACTUADORES Y PERIFÉRICOS USANDO EL LENGUAJE DE PROGRAMACIÓN JAVA

DOCENTE: ING. DARWIN ALULEMA

ESTUDIANTES:

SALAZAR ISABEL

MORALES STEVE

REINOSO SANTIAGO

NRC: 4463

SANGOLQUÍ, 10 DE MAYO DEL 2019

.

Índice general

Índice general **2**

0.1. PLANTEAMIENTO DEL PROBLEMA 3
0.2. OBJETIVOS 4
0.2.1. OBJETIVO GENERAL 4
0.2.2. OBJETIVOS ESPECIFICOS 4
0.3. ESTADO DEL ARTE 5
0.4. MARCO TEÓRICO 7
0.4.1. Java y Programacion Orientada a Objetos 7
0.4.2. Interfaz GUI en Java 8
0.4.3. Sistemas de Control 8
0.4.4. Sensores 10
0.5. DIAGRAMAS 11
0.6. LISTA DE COMPONENTES 11
0.7. MAPA DE VARIABLES 12
0.8. EXPLICACIÓN DE CÓDIGO FUENTE 13
0.8.1. Emisor 13
0.8.2. Receptor 17
0.9. CONCLUSIONES 23
0.10. RECOMENDACIONES 23
0.11. CRONOGRAMA 24
0.12. BIBLIOGRAFIA 25
0.13. MANUAL DE USUARIO 26

0.1. PLANTEAMIENTO DEL PROBLEMA

El desarrolllo de sistemas de control que permitan el manejo de perifericos, sensores y actuadores es una amplia rama del uso adecuado de la tecnología actual que permite mantener datos actuales de cualquier tipo de información ambiental solicitada en este caso se solicita sensores que controlen CO2, Humedad, Temperatura y de proximidad. Se tiene que tener en cuenta que la interaccion humano-sistema se ve siempre permitido por una interfaz HMI la cual permite que el usuario controle el sistema de mejor forma y de igual manera permite un mejor entendimiento e interpretacion de los datos copilados.

0.2. OBJETIVOS

0.2.1. OBJETIVO GENERAL

Diseñar, desarrollar e implementar un sistema de control de sensores, actuadores y periféricos usando el lenguaje de programación java.

0.2.2. OBJETIVOS ESPECIFICOS

- Diseñar un sistema de control con sensores de CO2, Humedad, Temperatura y de proximidad.
- Desarrollar la programacion necesaria y la implementación de los instrumentos para controlar un ventilador, una niquelina, un foco y un LCD para el despliegue de información del sistema.
- Comprobar el funcionamiento de los sensores utilizados con el sistema ARduino.

0.3. ESTADO DEL ARTE

En el campo del monitoreo del medio ambiente existen varios trabajos previos que muestran no solo un entorno de programación, también muestran usos más específicos; el uso de sensores para medir parámetros referenciales de calidad de aire en la ciudad de Quito haciendo referencia al artículo científico "Desarrollo de un prototipo de aplicación web en combinación con la plataforma Arduino para controlar la calidad de aire de la ciudad de Quito.", realizado por Carrera Arízaga, Diego Mauricio en el año 2015 en el cual se plantea un sistema con tres sensores y capaz de conectarse al internet utilizando un servicio web de tipo RESTful. Se puede apreciar que este artículo es mucho más específico la aplicación sistema Arduino en conjunto con un entorno de programación diferente en comparación al trabajo realizado el cual trabaja en Java y usa varios sensores. Como se puede apreciar el presente trabajo no ocupa solo un tipo de sensor, sino, se hace uso de varios con diferentes objetivos. De la misma manera se puede encontrar en el trabajo de grado "Diseño e Implementación de un Sistema de Monitorización y Alerta Temprana para la Escuela de Ingeniería Electrónica", realizado y ejecutado en la Escuela Superior Politécnica de Chimborazo por Tacuri Fernández, Mauricio Fernando en el año 2011; en este trabajo se usan micro controladores, sensores de temperatura, sensores de gas y de humo, al implementar el sistema, se logra dar alerta en caso de fugas de gas, minutos después de haberse iniciado, en el caso de la elevación de temperatura, se comparan con lecturas anteriores y finalmente el sensor de humo tiene características similares al sensor de gas, es decir que con la implementación del sistema se puede prevenir con un 90 porciento incendios causados por fugas de gas, con un mínimo de falsas alarmas.

Y en un trabajo aún más cercano a nuestro entorno "Diseño e implementación de un sistema de monitorización ambiental en el hogar para dispositivos Android" realizado por Mantilla Leonardo, Játiva Damián en el año 2016 el cual está enfocado al monitoreo ambiental de variables como temperatura, humedad y gases tóxicos como Gas Licuado de Petróleo (GLP), además permite varias opciones de respuesta ante un evento producido, como desactivar el suministro de gas mediante el cierre automático de la válvula de gas, así como la activación de un extractor para disminuir la cantidad de GLP en el aire, en caso de sobrepasar los rangos establecidos como perjudiciales para la salud el sistema se activa automáticamente y envía un mensaje de texto SMS a un contacto definido por el usuario. Durante el proyecto se desarrolló una aplicación para dispositivos con sistema operativo Android, que utiliza una interfaz para mostrar los niveles de temperatura, humedad y GLP, siendo recopilados a través de sensores, el procesamiento de datos se realiza a través de un dispositivo arduino MEGA utilizado como tarjeta de adquisición y como servidor.

En ambos proyectos de investigación se observa diferentes enfoques de control, como lo es un sistema de alarmas o un sistema de monitoreo en el hogar, utilizando una gran variedad de sensores para diferentes parámetros con cantidades específicas de medición. Es importante también reconocer los entornos de programación. Todos estos se ligan al presente trabajo por el uso del sistema Arduino, aplicación y uso de sensores, y visualización del resultados en consola.

0.4. MARCO TEÓRICO

0.4.1. Java y Programacion Orientada a Objetos

El lenguaje Java fue creado por Sun Microsystems Inc., Aparece en el año 1995 y debe, en gran medida, su popularidad al éxito del servicio WWW. Se creó en su origen para que fuese un lenguaje multiplataforma. Para ello se compila en un código intermedio: bytecodey necesita de una máquina virtual que lo ejecute. Normalmente, no utiliza código nativo, es decir, no se puede ejecutar directamente por el procesador.

Se disponen de varias plataformas Java para el desarrollo. Una plataforma es una combinación de hardware y software, usada para desarrollar y/o ejecutar programas.

Se va a empezar con el editor. Uno sencillo de libre distribución es el Notepad. Es un editor básico que reconoce la gramática del lenguaje; es recomendable para empezar con aplicaciones pequeñas, pero no para producir con efectividad. Para el desarrollo y compilación de aplicaciones Java, utilizaremos: StandardEdition(Java SE) o JavaDevelopmentKit(JDK) de Sun.

La programación orientada a objetos establece un equilibrio entre la importancia de los procesos y los datos, mostrando un enfoque más cercano al pensamiento del ser humano. Se introduce un aspecto novedoso respecto al anterior paradigma: la herencia, facilitando el crecimiento y la mantenibilidad. Las bases de la programación orientada a objetos son: abstracción, encapsulación, modularidad y jerarquización. La abstracciones un proceso mental de extracción de las características esenciales, ignorando los detalles superfluos. Resulta ser muy subjetiva dependiendo del interés del observador, permitiendo abstracciones muy diferentes de la misma realidad La encapsulación es ocultar los detalles que dan soporte a un conjunto de características esenciales de una abstracción. Existirán dos partes, una visible que todos tienen acceso y se aporta la funcionalidad, y una oculta que implementa los detalles internos. La modularidades descomponer un sistema en un conjunto de partes. Aparecen dos conceptos muy importantes: acoplamiento y cohesión.

El acoplamiento entre dos módulos mide el nivel de asociación entre ellos; nos interesa buscar módulos poco acoplados.

La cohesión de un módulo mide el grado de conectividad entre los elementos que los forman; nos interesa buscar una cohesión alta.

La jerarquía es un proceso de estructuración de varios elementos por niveles. La programación orientada a objetos implementa estos cuatro conceptos con los siguientes elementos: clases y objetos, atributos y

estado, métodos y mensajes, herencia y polimorfismo.

0.4.2. Interfaz GUI en Java

Llamamos Interfaz Gráfica GUI (Graphical User Interface) al conjunto de componentes gráficos que posibilitan la interacción entre el usuario y la aplicación. Es decir ventnas, botones, combos, listas, cajas de diálogo, campos de texto, etc.
Primero tenemos que diseñar la aplicación,programarla y por último los eventos que se generan a medida que el usuario interactua con la Interfaz.
Los componentes son objetos de las clases que heredan de la clase base componente como Button, List, TextField, TextArea, Label, etc.
En una GUI los componentes son contenidos en Contenedores o containers. Un Containes es un objeto cuya clase hereda de Container(clase que a su vez es subclase de Component) y tiene la responsabilidad de contener Componentes.
Generalmente una GUI se monta sobre un Frame. Esté sera el Container principal que contendrá a los componentes de la Interfaz Gráfica, un Container podría contener a otros containers.

0.4.3. Sistemas de Control

Un sistema dinámico puede definirse conceptualmente como un ente que recibe unas acciones externas o variables de entrada, y cuya respuesta a estas acciones externas son las denominadas variables de salida.
Las acciones externas al sistema se dividen en dos grupos, variables de control, que se pueden manipular, y perturbaciones sobre las que no es posible ningún tipo de control. Dentro de los sistemas se encuentra el concepto de sistema de control. Un sistema de control es un tipo de sistema que se caracteriza por la presencia de una serie de elementos que permiten influir en el funcionamiento del sistema. La finalidad de un sistema de control es conseguir, mediante la manipulación de las variables de control, un dominio sobre las variables de salida, de modo que estas alcancen unos valores prefijados (consigna).

Un sistema de control ideal debe ser capaz de conseguir su objetivo cumpliendo los siguientes requisitos:

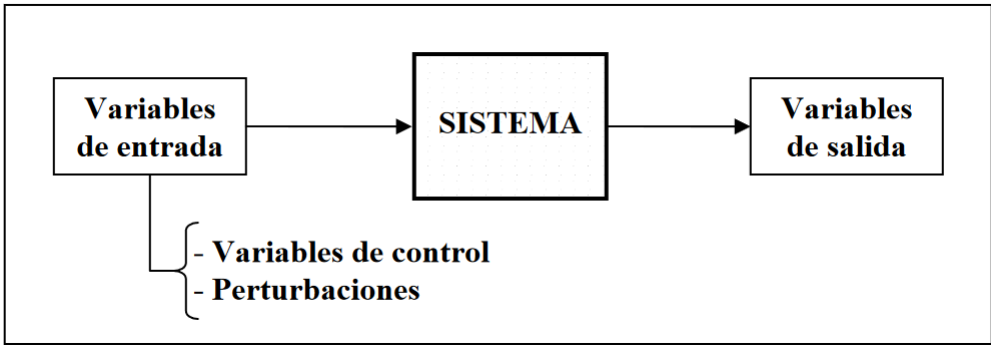


Figura 1: Esquema general de un sistema

1. Garantizar la estabilidad y, particularmente, ser robusto frente a perturbaciones y errores en los modelos.

2. Ser tan eficiente como sea posible, según un criterio preestablecido. Normalmente este criterio consiste en que la acción de control sobre las variables de entrada sea realizable, evitando comportamientos bruscos e irreales.

3. Ser fácilmente implementable y cómodo de operar en tiempo real con ayuda de un ordenador.

Los elementos básicos que forman parte de un sistema de control y permiten su manipulación son los siguientes:

- Sensores. Permiten conocer los valores de las variables medidas del sistema.

- Controlador. Utilizando los valores determinados por los sensores y la consigna impuesta, calcula la acción que debe aplicarse para modificar las variables de control en base a cierta estrategia.

- Actuador. Es el mecanismo que ejecuta la acción calculada por el controlador y que modifica las variables de control.

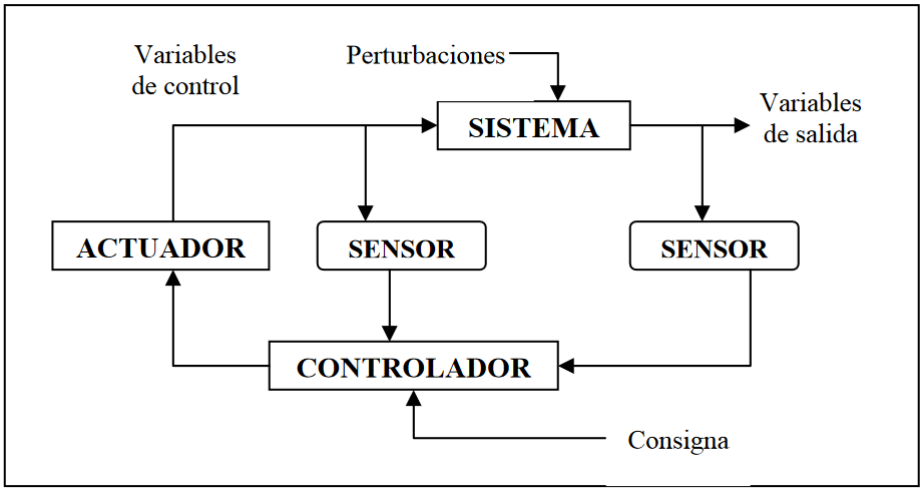


Figura 2: Esquema General de un Sistema de Control

0.4.4. Sensores

Los sensores son dispositivos quepueden ser eléctricos, mecánicos o una combinación de estos, que se utilizan para transformar magnitudes químicas o físicas como: luz, magnetismo, presión, humedad, Ph, entre otros, en valores medibles de dicha magnitud (normalmente magnitud eléctrica).

Los sensores básicamente, transforman un determinado fenómeno físico en una señal, este proceso se conoce en el ámbito de las ciencias como transducción, lo que constituye en la conversión de un dato en una información en un “lenguaje” diferente (Ávila & Jaramillo, 2010)

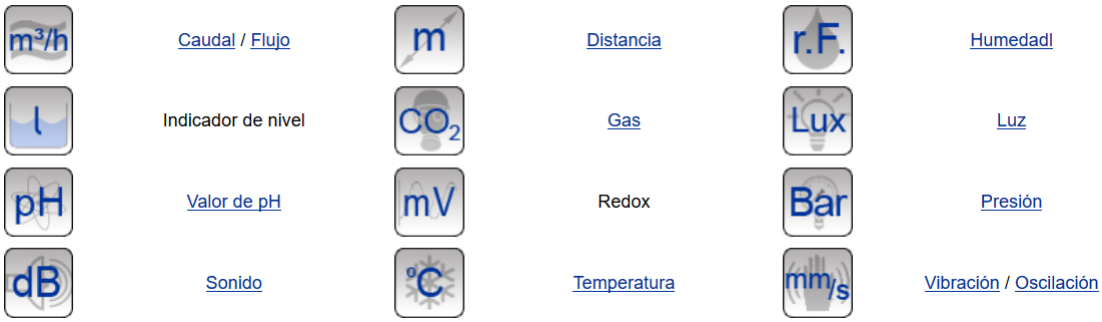


Figura 3: Clases de Sensores y su Magnitud

0.5. DIAGRAMAS

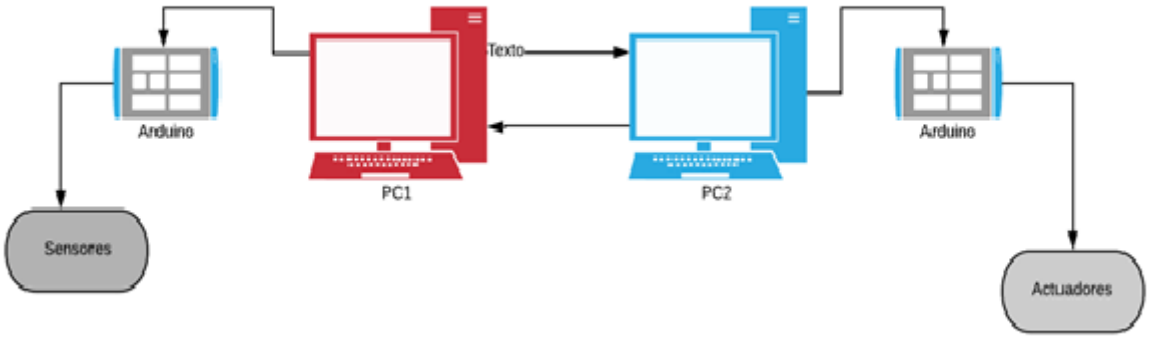


Figura 4: Diagrama de Bloques

0.6. LISTA DE COMPONENTES

1. Sensores
    - Sensor de Húmedad
    - Sensor de CO2
    - Sensor de Temperatura
    - Sensor de Proximidad
2. Relé
3. Ventilador
4. Niquelina
5. Foco
6. LCD
7. Arduino MEGA

0.7. MAPA DE VARIABLES

Puerto Serie	i	j	valor 0	valor 1	valor 2	valor 3	valor 4	tiempo	lcd
12-3-2-1-	0	0	12	0	0	0		200	distancia 12 cm
	1	0	12	3	0	0		400	distancia 12 cm
	2	0	12	3	2			600	distancia 12 cm
	3	0	12	3	2	1		800	distancia 12 cm
	4	0	12	3	2	1		1000	distancia 12 cm
13-2-4-5-	0	0	13	3	2	1		1200	distancia 13 cm
	1	0	13	2	2	1		1400	distancia 13 cm
	2	0	13	2	4	1		1600	distancia 13 cm
	3	0	13	2	4	5		1800	distancia 13 cm
	4	0	13	2	4	5		2000	distancia 13 cm
10-3-1-4	0	1	10	2	4	5		200	temperatura 2 C
	1	1	10	3	4	5		400	temperatura 3 C
	2	1	10	3	1	5		600	temperatura 3 C
	3	1	10	3	1	4		800	temperatura 3 C
	4	1	10	3	1	4		1000	temperatura 3 C
5-4-2-6	0	1	5	3	1	4		1200	temperatura 3 C
	1	1	5	4	1	4		1400	temperatura 4 C
	2	1	5	4	2	4		1600	temperatura 4 C
	3	1	5	4	2	6		1800	temperatura 4 C
	4	1	5	4	2	6		2000	temperatura 4 C
3-6-3-5	0	2	3	4	2	6		200	CO2 2%
	1	2	3	6	2	6		400	CO2 2%
	2	2	3	6	3	6		600	CO2 3%
	3	2	3	6	3	5		800	CO2 3%
	4	2	3	6	3	5		1000	CO2 3%
6-5-3-3-	0	2	6	6	3	5		1200	CO2 3%
	1	2	6	5	3	5		1400	CO2 3%
	2	2	6	5	3	5		1600	CO2 3%
	3	2	6	5	3	3		1800	CO2 3%
	4	2	6	5	3	3		2000	CO2 3%
3-3-3-3-	0	3	3	5	3	3		200	Humedad 3%
	1	3	3	3	3	3		400	Humedad 3%
	2	3	3	3	3	3		600	Humedad 3%
	3	3	3	3	3	3		800	Humedad 3%
	4	3	3	3	3	3		1000	Humedad 3%
6-4-3-4-	0	3	6	3	3	3		1200	Humedad 3%
	1	3	6	4	3	3		1400	Humedad 3%
	2	3	6	4	3	3		1600	Humedad 3%
	3	3	6	4	3	4		1800	Humedad 4%
	4	3	6	4	3	4		2000	Humedad 4%

Figura 5: Mapa de Variables

0.8. EXPLICACIÓN DE CÓDIGO FUENTE

0.8.1. Emisor

Arduino

Al inicio del programa se define los pines necesarios.

```
#define PIN_TRIGGER 8
#define PIN_ECHO 7
#define ESPERA_ENTRE_LECTURAS 1000 // tiempo entre lecturas consecutivas en milisegundos
#define TIMEOUT_PULSO 25000 // la espera máxima de es 30 ms o 30000 µs
#define MEDIA_VELOCIDAD_SONIDO 0.017175 // Mitad de la velocidad del sonido a 20 °C expresada en cm/µs
#define CO2 A1

// Incluimos librería
#include <DHT.h>

// Definimos el pin digital donde se conecta el sensor
#define DHTPIN 2
// Dependiendo del tipo de sensor
#define DHTTYPE DHT11

// Inicializamos el sensor DHT11
DHT dht(DHTPIN, DHTTYPE);
```

Se definen las constantes.

```
float distancia;
unsigned long tiempo;
unsigned long cronometro;
unsigned long reloj=0;
unsigned short deley=350;
```

```
void setup()
{
  Serial.begin(9600);
  dht.begin();
  pinMode(PIN_ECHO,INPUT);
  pinMode(PIN_TRIGGER,OUTPUT);
   pinMode(CO2,INPUT);
  digitalWrite(PIN_TRIGGER,LOW); // Para «limpiar» el pulso del pin
  delayMicroseconds(2);
}
```

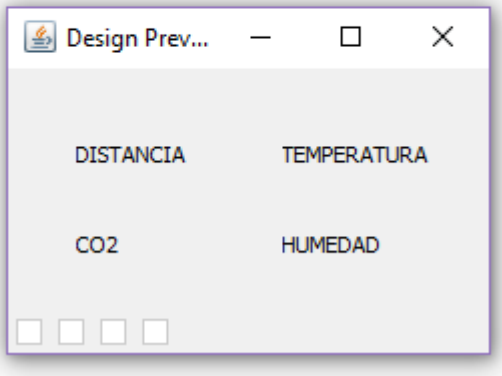
En el void set up se configura los pines.

En el void loop se realiza el desarrollo del programa el cual realiza la recepcion de datos obtenidos de los sensores.

```
void loop()
{
  cronometro=millis()-reloj;
  if(cronometro>ESPERA_ENTRE_LECTURAS)
  {
    digitalWrite(PIN_TRIGGER,HIGH); // Un pulso a nivel alto...
    delayMicroseconds(10); // ...durante 10 µs y
    digitalWrite(PIN_TRIGGER,LOW); // ...volver al nivel bajo
    tiempo=pulseIn(PIN_ECHO,HIGH,TIMEOUT_PULSO); // Medir el tiempo que tarda en llegar un pu
    distancia=MEDIA_VELOCIDAD_SONIDO*tiempo;
    reloj=millis();
  }

 double Co2=analogRead(CO2)/20.13;  //leo la entrada analogica y convierto en un porcentaje
 float tem=dht.readTemperature();   //leo temperatura
 float Hum=dht.readHumidity();      //leo humedad

  // doy una etiquetao identifiacion a  cada valor y los envio
 Serial.println("d"+(String)distancia);
 delay(deley);
 Serial.println("t"+(String)tem);
 delay(deley);
 Serial.println("c"+(String)Co2);
 delay(deley);
 Serial.println("h"+(String)Hum);
 delay(deley);
}
```

Java

Se muestra el frame con el que se trabaja



Paquete usados

```
package pue.vista;

import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.comm.NoSuchPortException;
import javax.comm.PortInUseException;
import javax.comm.UnsupportedCommOperationException;
import pue.controlador.PuertoSerie;
```

Se ocultan los checks y se usa el puerto en serie.

```
public class InterfazEmisor extends javax.swing.JFrame {

    //creamos 2 puertos uno para el cpu 2(receptor) y para el arduino con
    //los sensores
    private PuertoSerie arduino,cpu;

    public InterfazEmisor() {
        initComponents();
        //ocultamos todos los checkboxs
        jcbCo2.setVisible(false);
        jcbDis.setVisible(false);
        jcbHum.setVisible(false);
        jcbTem.setVisible(false);

        //inicializamos los puertos con sus respectivos COM
        try {
            arduino=new PuertoSerie("COM9");
            cpu=new PuertoSerie("COM14");
        } catch (NoSuchPortException ex) {
            Logger.getLogger(InterfazEmisor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (PortInUseException ex) {
            Logger.getLogger(InterfazEmisor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedCommOperationException ex) {
            Logger.getLogger(InterfazEmisor.class.getName()).log(Level.SEVERE, null, ex);
        } catch (IOException ex) {
            Logger.getLogger(InterfazEmisor.class.getName()).log(Level.SEVERE, null, ex);
        }


    }
```

Se eliminan las unidades y se añade una etiqueta al texto en el texBox.

```
    @SuppressWarnings("unchecked")
    Generated Code

    ///cada vez que el estado de un checkbox cambie se enviara el dato correspondiente al cpu2
    private void jcbDisStateChanged(javax.swing.event.ChangeEvent evt) {
        //elimino las unidades y aumento una etiqueta o identifiacion al principio
        cpu.tx('d'+jLdis.getText().replace(" cm",""));
    }

    private void jcbTemStateChanged(javax.swing.event.ChangeEvent evt) {
        cpu.tx('t'+jLtem.getText().replace(" °C",""));
    }

    private void jcbCo2StateChanged(javax.swing.event.ChangeEvent evt) {
        cpu.tx('c'+jLCo2.getText().replace(" %",""));
    }

    private void jcbHumStateChanged(javax.swing.event.ChangeEvent evt) {
        cpu.tx('h'+jLhum.getText().replace(" %",""));
    }
```

En la Clase Controlador, se muestra los paquetes usados, el inicio del main y la inicialización de las variables a usarse.

```
package pue.controlador;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.comm.CommPortIdentifier;
import javax.comm.NoSuchPortException;
import javax.comm.PortInUseException;
import javax.comm.SerialPort;
import javax.comm.UnsupportedCommOperationException;
import pue.vista.InterfazEmisor;


public class PuertoSerie extends Thread{

    private CommPortIdentifier idPuerto;
    private SerialPort puertoSerie;
    private InputStream flujoEntrada;
    private OutputStream flujoSalida;

    private int lon;
    private byte [] buffer = new byte[1024]; // tamaño maximo de caracteres que se puede recibir
    private String aux;
```

Se configura el puerto en serie.

```
public PuertoSerie (String puerto) throws NoSuchPortException, PortInUseException, UnsupportedCommOperationException, IOException{

        idPuerto = CommPortIdentifier.getPortIdentifier(puerto);
        System.out.println("Puerto "+puerto+" encontrado");
    if(idPuerto.isCurrentlyOwned()){
        System.out.println("Puerto Ocupado");
    }
    else{
        puertoSerie= (SerialPort) idPuerto.open("tx/rx",1000);
        System.out.println("Puerto abierto");
        puertoSerie.setSerialPortParams(9600, SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
        System.out.println("Puerto Configurado");
        flujoSalida = puertoSerie.getOutputStream();
        System.out.println("Flujo de salida configurado");
        flujoEntrada = puertoSerie.getInputStream();
        System.out.println("Flujo de entrada configurado");
        this.start();
    }
    }

    public void tx (String mensaje) {
        try {
            flujoSalida.write(mensaje.getBytes());
        } catch (IOException ex) {
            Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
        }
                System.out.println("Mensaje enviado. "+mensaje);
    }
```

Se Desarrolla el run en el cual se encuentra un bucle que se realiza para cada uno de los 4 sensores.

```
public void run(){
    while(true){
        try {
            if ((lon=flujoEntrada.read(buffer)) >- 1){
                aux = new String (buffer,0,lon);
                //Sacamos el primer caracter de la cadena
                char inicial=aux.charAt(0);
                //y eliminamos el primer caracter
                aux=aux.replace(String.valueOf(inicial),"");
                //identificamos el primer caracter y clasificamos los datos
                if(inicial=='d'){
                    //limpiamos el label
                    InterfazEmisor.jLdis.setText("");
                    //ponemos el valor del sensor con sus unidades
                    InterfazEmisor.jLdis.setText(aux+" cm");
                    //cambiamos el estado del checkbox
                    InterfazEmisor.jcbDis.setSelected(!InterfazEmisor.jcbDis.isSelected());
                 }
                if(inicial=='t'){
                    InterfazEmisor.jLtem.setText("");
                    InterfazEmisor.jLtem.setText(aux+" °C");
                    InterfazEmisor.jcbTem.setSelected(!InterfazEmisor.jcbTem.isSelected());
                 }
                if(inicial=='c'){
                    InterfazEmisor.jLCo2.setText("");
                    InterfazEmisor.jLCo2.setText(aux+" %");
                    InterfazEmisor.jcbCo2.setSelected(!InterfazEmisor.jcbCo2.isSelected());
                 }
                if(inicial=='h'){
                    InterfazEmisor.jLhum.setText("");
                    InterfazEmisor.jLhum.setText(aux+" %");
```

Culmina la recepción de datos y se cierra el flujo.

```
                    InterfazEmisor.jcbTem.setSelected(!InterfazEmisor.jcbTem.isSelected());
                }
                if(inicial=='c'){
                    InterfazEmisor.jLCo2.setText("");
                    InterfazEmisor.jLCo2.setText(aux+" %");
                    InterfazEmisor.jcbCo2.setSelected(!InterfazEmisor.jcbCo2.isSelected());
                }
                if(inicial=='h'){
                    InterfazEmisor.jLhum.setText("");
                    InterfazEmisor.jLhum.setText(aux+" %");
                    InterfazEmisor.jcbHum.setSelected(!InterfazEmisor.jcbHum.isSelected());
                }
            }

        } catch (IOExceptcn ex) {
            Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

public void cerrar() throws IOException{
    flujoSalida.close();
    flujoEntrada.close();
    puertoSerie.close();

}
```

0.8.2. Receptor

Arduino

Se define los pines, las librerias se incluyen y se crea estrings para la recpción de los datos

```
//incluyo las librerias para el lcd
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
//defino los pines necesarios
#define foco 9
#define niquelina 6
#define ventilador 7
//creo arreglos para presentarlos en el lcd
String etiqueta[4]={"DISTANCIA: ","TEMPERATURA: ","CO2: ","HUMEDAD: "
String unidad[4]={" cm"," C"," %"," %"};
String valor[5]={"","","","","",""};
int i=0,j=0;
float tiempo;
//defino el lcd a usar
LiquidCrystal_I2C lcd(0x27,16,2);
```

Se configura cada pin como salida en el set up.

```
void setup() {
  pinMode(foco,OUTPUT);         //defino el pin como salida
  pinMode(niquelina,OUTPUT);    //defino el pin como salida
  pinMode(ventilador,OUTPUT);   //defino el pin como salida
  lcd.init();                   //inicializo el lcd
  lcd.backlight();              //coloco contraste en el lcd
  Serial.begin(9600);           //inicio comunicaion serial a 96000 baudios
  delay(500);                   //retraso de 500ms
  lcd.print("listo");           //notifico
  delay(500);                   //otro retraso
  tiempo=millis();              //leo tiempo de arranque
}
```

En en loop se desarrolla el programa de recepción de información y se identifica para que clase de datos es, es decir, de que sensor es.

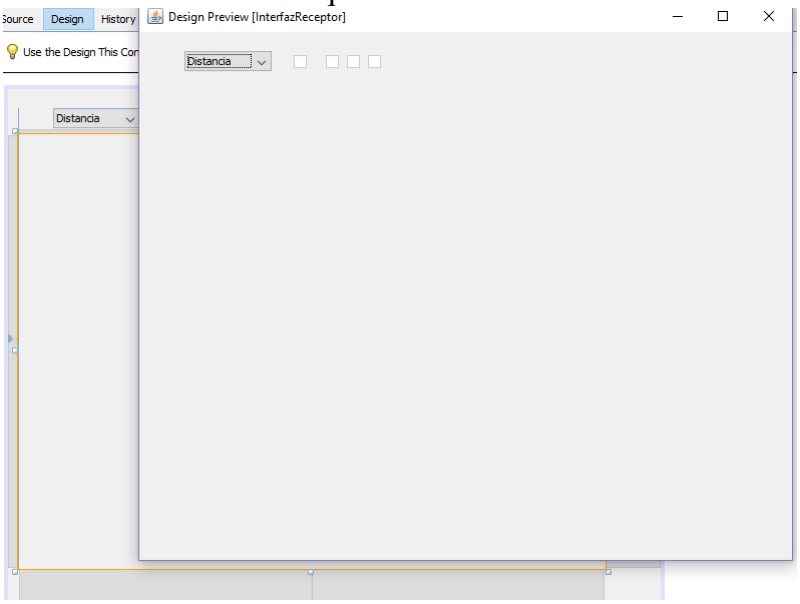
```
void loop() {
  if(Serial.available()>0){     ///si hay datos de llegada
     String data = Serial.readStringUntil('-');    //separo los datos cada vez que llegue un -
      valor[i]=data;                               // asigno el dato al valor de turno
      //voy ingrementando 0 1 2 3 4 y regreso a 0
      if(i!=4){
        i++;
      }else{
        i=0;
      }
  }
  if(millis()-tiempo>2000){       // si el timepo transcurrido es mayor a 2
   mostrar(j);                    // muestro los datos de turno
   tiempo=millis();               //mido tiempo
    //voy variando entre 0 1 2 3 0 1 2 3 ....
    if(j!=3){
        j++;
      }else{
        j=0;
      }
  }

 if(!valor[1].toFloat()>12){    //si la temperatua es mayor a 12 se prendera la niquelina
  digitalWrite(niquelina,HIGH);
 }else{
  digitalWrite(niquelina,LOW);
 }

 if(!valor[2].toFloat()>75){   // si  el co2 es mayor al 75% prende el ventilador
  digitalWrite(ventilador,LOW);

 }else{
  digitalWrite(ventilador,HIGH);
 }

if(!valor[3].toFloat()>12){     // si la humedad es mayor a 12% prende el foco
  digitalWrite(foco,HIGH);
 }else{
  digitalWrite(foco,LOW);
 }

}
```

El método mostrar es el cual permite visualizar en el LCD los datos.

```
void mostrar(int a){  ///metodo para mostrar en el lcd
  lcd.clear();           ///limpio el lcd
  lcd.setCursor(0,0);     ///ubico el cursor en el primer espacio de la primera fila
  lcd.print(etiqueta[a]);  ///muestro la etiqueta
  lcd.setCursor(0,1);       ///me ubico en la fila inferior
  lcd.print(valor[a]+unidad[a]);  /////escribo el valor
}
```

Java

Se muestra el Frame que se va a utilizar.




```
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.comm.NoSuchPortException;
import javax.comm.PortInUseException;
import javax.comm.UnsupportedCommOperationException;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYSeriesCollection;
import pue.controlador.PuertoSerie;


public class InterfazReceptor extends javax.swing.JFrame {

    //declaro 2 puertos uno para el cpu (emisor) arduino con los actuadores
    private PuertoSerie cpu,arduino;
    //creo un arreglo con el nombre de las graficas
    private String[] nombre={"DISTANCIA","TEMPERATURA","CO2","HUMEDAD"};

    public InterfazReceptor() {
        initComponents();
        //oculto todos los checkbox
        jcbCo2.setVisible(false);
        jcbDis.setVisible(false);
        jcbHum.setVisible(false);
        jcbTem.setVisible(false);

        //inicializo los puertos
```

```
 //inicializo los puertos
 try {
     arduino=new PuertoSerie("COM14");
     cpu=new PuertoSerie("COM9");
 } catch (NoSuchPortException ex) {
     Logger.getLogger(InterfazReceptor.class.getName()).log(Level.SEVERE, null, ex);
 } catch (PortInUseException ex) {
     Logger.getLogger(InterfazReceptor.class.getName()).log(Level.SEVERE, null, ex);
 } catch (UnsupportedCommOperationException ex) {
     Logger.getLogger(InterfazReceptor.class.getName()).log(Level.SEVERE, null, ex);
 } catch (IOException ex) {
     Logger.getLogger(InterfazReceptor.class.getName()).log(Level.SEVERE, null, ex);
 }

 //doy formato a la grafica usando la coleccion
     JFreeChart disChart=ChartFactory.createXYLineChart("DISTANCIA VS TIEMPO",
             "TIEMPO","DISTANCIA", cpu.getDis(), PlotOrientation.VERTICAL,
     true, true, false);

     //creo un panel
ChartPanel dispPanel = new ChartPanel(disChart);
//asigno el panel creado al ya existente en la interfaz
dispPanel.setBounds(jPdistancia.getBounds());
this.jPdistancia.add(dispPanel);
```

```
    //cuando el estado del checkbox cambie este enviara los datos separados por
    // y eliminara posibles tabluaciones
    private void jcbDisStateChanged(javax.swing.event.ChangeEvent evt) {
        arduino.tx(jcbDis.getText().replace(String.valueOf((char)11),""));
        arduino.tx('-'+jcbTem.getText().replace(String.valueOf((char)11),""));
        arduino.tx('-'+jcbCo2.getText().replace("-",""));
        arduino.tx('-'+jcbHum.getText().replace(String.valueOf((char)11),""));
    }

    //cuando el seleccionador del combobox cambie
    private void jCBSelectorItemStateChanged(java.awt.event.ItemEvent evt) {
        //extraigo el indice del combobox
        int aux=jCBSelector.getSelectedIndex();
        //creo una coleccion auxiliar
        XYSeriesCollection collection;
        //defino a que coleccion fue seleccionada
        switch (aux){
            case 0:
                collection=cpu.getDis();
            break;
            case 1:
                collection=cpu.getTem();
            break;
            case 2:
                collection=cpu.getCo2();
            break;
            default:
                collection=cpu.getHum();
        }
```

```
package pue.controlador;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.comm.CommPortIdentifier;
import javax.comm.NoSuchPortException;
import javax.comm.PortInUseException;
import javax.comm.SerialPort;
import javax.comm.UnsupportedCommOperationException;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
import pue.vista.InterfazReceptor;
 public class PuertoSerie extends Thread{

    private final CommPortIdentifier idPuerto;
    private SerialPort puertoSerie;
    private InputStream flujoEntrada;
    private OutputStream flujoSalida;

    private int lon;
    //defino un contador para la serie
    private long [] i={0,0,0,0};
    private byte [] buffer = new byte[1024]; // tamaño maximo de caracteres que se puede recibir
    private String aux;

    //creo las series que serian equivalentes a los pares ordenados
    private XYSeries distancia= new XYSeries("Distancia");
    private XYSeries temperatura= new XYSeries("Temperatura");
    private XYSeries co2= new XYSeries("CO2");
    private XYSeries humedad= new XYSeries("Humedad");

    //creo las colecciones que son similares a las funciones
    private XYSeriesCollection dis=new XYSeriesCollection(distancia);
    private XYSeriesCollection tem=new XYSeriesCollection(temperatura);
    private XYSeriesCollection Co2=new XYSeriesCollection(co2);
    private XYSeriesCollection hum=new XYSeriesCollection(humedad);

    public XYSeriesCollection getDis() {
        return dis;
    }

    public XYSeriesCollection getTem() {
        return tem;
```

```
    public XYSeriesCollection getDis() {
        return dis;
    }

    public XYSeriesCollection getTem() {
        return tem;
    }

    public XYSeriesCollection getCo2() {
        return Co2;
    }

    public XYSeriesCollection getHum() {
        return hum;
    }



    public String getAux() {
        return aux;
    }
public PuertoSerie (String puerto) throws NoSuchPortException, PortInUseException, UnsupportedCommOperationException, IOException{

        idPuerto = CommPortIdentifier.getPortIdentifier(puerto);
        System.out.println("Puerto "+puerto+" encontrado");
     if(idPuerto.isCurrentlyOwned()){
         System.out.println("Puerto Ocupado");
     }
     else{
         puertoSerie= (SerialPort) idPuerto.open("tx/rx",1000);
```

En el código siguiente del programa se realiza la recepcion del los datos, se elimina la etiqueta inicial y se procede a la gráfica de los mismos.

```
    public String getAux() {
        return aux;
    }
public PuertoSerie (String puerto) throws NoSuchPortException, PortInUseException, UnsupportedCommOperationException, IOException{
            idPuerto = CommPortIdentifier.getPortIdentifier(puerto);
        System.out.println("Puerto "+puerto+" encontrado");
     if(idPuerto.isCurrentlyOwned()){
         System.out.println("Puerto Ocupado");
     }
     else{
         puertoSerie= (SerialPort) idPuerto.open("tx/rx",1000);
         System.out.println("Puerto abierto");
         puertoSerie.setSerialPortParams(9600, SerialPort.DATABITS_8,
                 SerialPort.STOPBITS_1,SerialPort.PARITY_NONE);
         System.out.println("Puerto Configurado");
         flujoSalida = puertoSerie.getOutputStream();
         System.out.println("Flujo de salida configurado");
         flujoEntrada = puertoSerie.getInputStream();
         System.out.println("Flujo de entrada configurado");
         this.start();
     }
    }
       public void tx (String mensaje) {
        try {
            flujoSalida.write(mensaje.getBytes());
        } catch (IOException ex) {
           Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
        }
                System.out.println("Mensaje enviado. "+mensaje);
   }
public void run(){
    while(true){
        try {
            if ((lon=flujoEntrada.read(buffer)) >= 1){
                aux = new String (buffer,0,lon);
                //guardo el primer caracter de la cadena
                char inicial=aux.charAt(0);
                //elimino el primer caracter y un posible enter
                aux=aux.replace(String.valueOf(inicial),"").replace(String.valueOf((char)10),"");
                //compara cual fue el primer caracter
                if(inicial=='d'){
                //añado a la serie un par formado por el contador y el dato recibido
                    distancia.add(i[0],Float.parseFloat(aux));
                    i[0]++;    //incremeto el contador para el siguente dato
                    InterfazReceptor.jcbDis.setText(aux); //guardo el dato en la interfaz(checkbox)
                    //cambio el estado el checkbox
                    InterfazReceptor.jcbDis.setSelected(!InterfazReceptor.jcbDis.isSelected());
                 }
                if(inicial=='t'){
                    temperatura.add(i[1],Float.parseFloat(aux));
                    i[1]++;
                    InterfazReceptor.jcbTem.setText(aux);
                    InterfazReceptor.jcbTem.setSelected(!InterfazReceptor.jcbTem.isSelected());
                 }
                if(inicial=='c'){
                    co2.add(i[2],Float.parseFloat(aux.replace("-","")));
                    i[2]++;
                    InterfazReceptor.jcbCo2.setText(aux.replace("-",""));
                    InterfazReceptor.jcbCo2.setSelected(!InterfazReceptor.jcbCo2.isSelected());
                 }
```

```
                if(inicial=='h'){
                    humedad.add(i[3],Float.parseFloat(aux));
                    i[3]++;
                    InterfazReceptor.jcbHum.setText(aux);
                    InterfazReceptor.jcbHum.setSelected(!InterfazReceptor.jcbHum.isSelected());
                 }
            }
        } catch (IOException ex) {
            Logger.getLogger(PuertoSerie.class.getName()).log(Level.SEVERE, null, ex);
        }catch(java.lang.NumberFormatException e){
            //exepcion que nos permite continuar si un valor no se pudo convertir
        }
    }
}

public void cerrar() throws IOException{
    flujoSalida.close();
    flujoEntrada.close();
    puertoSerie.close();

}
```

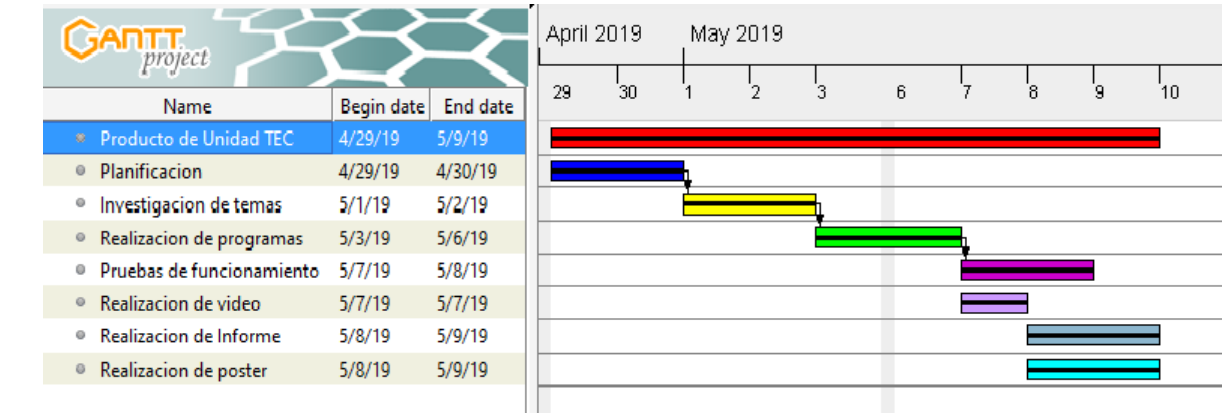
0.9. CONCLUSIONES

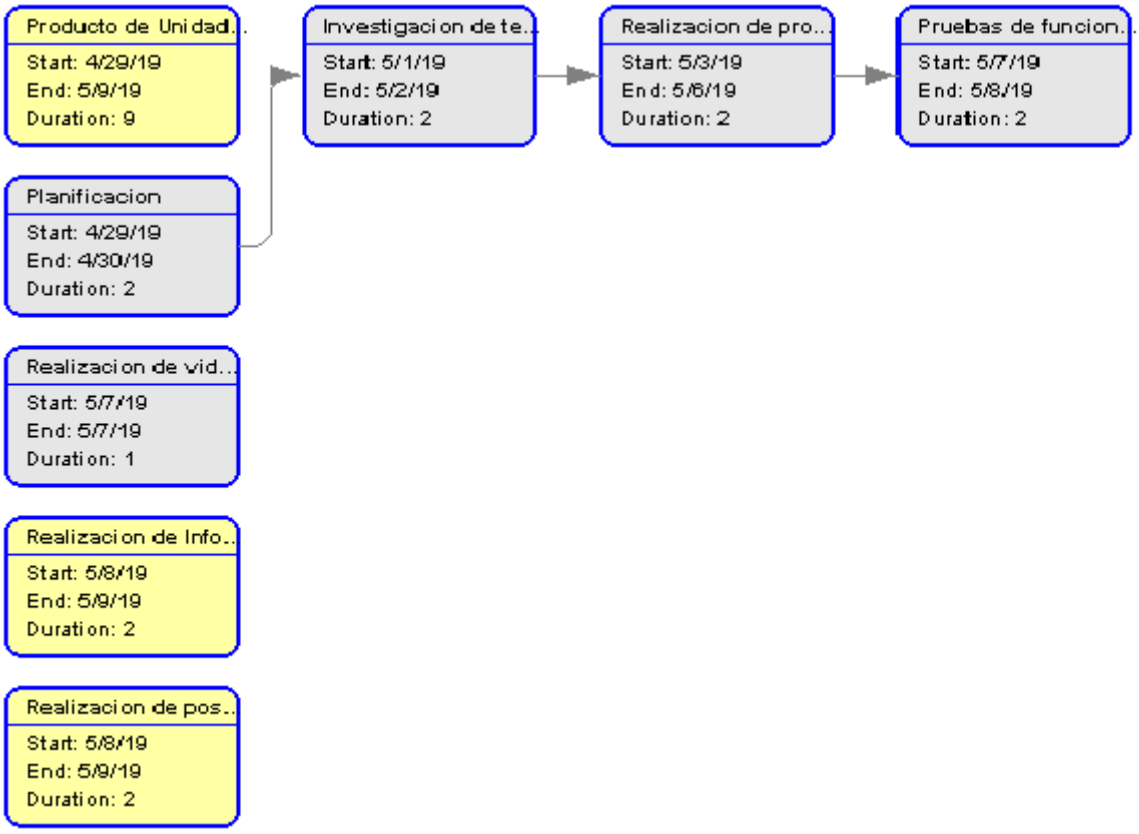
- Al implementar un sistema electrónico utilizando una gran variedad de sensores, se obtuvo una medicion variada que se muestra en pantalla por medio de graficas.
- Se desarrollo los respectivos programas receptor y emisor tanto en Java como para Arduino, los cuales permiten la conección de ambos PCs.
- Para el desarrollo del presente proyecto se utilizó los programas previamente aprendidos en clase, se los acoplo para que cumplan con las caracteristicas solicitadas.

0.10. RECOMENDACIONES

- Para controlar un dispositivo se necesita tener conocimiento de los pequeños cambios que se tienen y a las pequeñas atenuaciones que son sujetos y pueden afectar los resultados de las mediciones.
- Se recomienda el uso de dos arduinos MEGA, uno como receptor y otro como emisor para asi tener las programaciones de cada uno de ellos bien dirigidos y poder asi tener una independencia.
- Con la implementacion de los sensores y los demnas elementos tomar en cuenta que cada uno de ellos tendra su alimentación.

0.11. CRONOGRAMA





0.12. BIBLIOGRAFIA

- Carrera Arízaga, Diego Mauricio (2015). Desarrollo de un prototipo de aplicación web en combinación con la plataforma Arduino para controlar la calidad de aire de la ciudad de Quito. Carrera de Ingeniería en Sistemas e Informática. Universidad de las Fuerzas Armadas ESPE. Matriz Sangolquí.

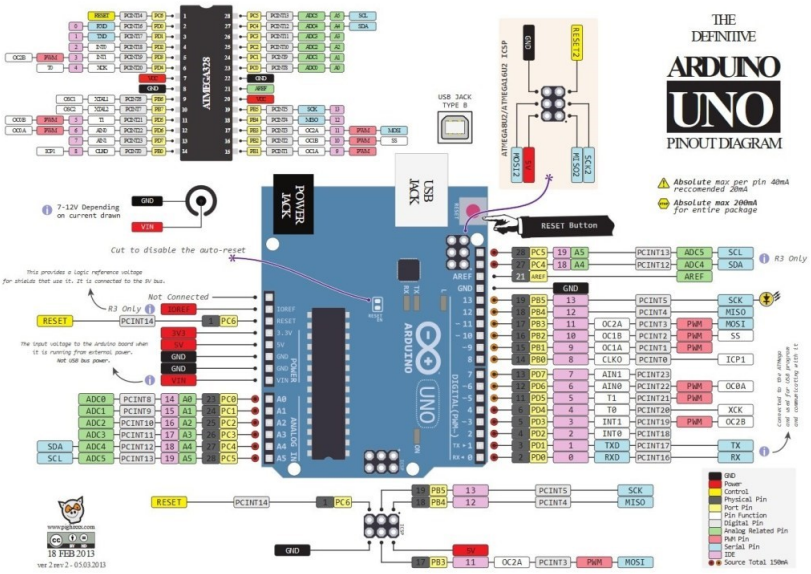
- Tacuri Fernández, Mauricio Fernando (2011). Diseño e Implementación de un Sistema de Monitorización y Alerta Temprana para la Escuela de Ingeniería Electrónica. Carrera de Ingenieria en Electrónica y Computación. Escuela Superior Politecnica de Chimborazo.

- Mantilla Torres, Leonardo Paúl y Játiva López, Damián Fernando (2016). Diseño e implementación de un sistemas de monitorización ambiental en el hogar para dispositivos Android. Carrera de Ingeniería en Electrónica y Telecomunicaciones. Universidad de las Fuerzas Armadas ESPE. Matriz Sangolquí.

- Ávila, B. P., & Jaramillo, R. A. (2010). Desarrollo de aplicación con sensores de temperatura usando una versión del Lenguaje JAVA llamada JAVELIN adecuada para el uso en Microcontroladores que admiten esta tecnología.Guayaquil: Escuela Superior Politecnica del Litoral. Recuperado el 23 de Agosto de 2015

0.13. MANUAL DE USUARIO

Tener en cuenta que se debe conectar correctamente los pines al Arduino a los que fue programado.



Emisor

- Sensor de CO2 conectado al Pin A1
- Sensor de humedad conectamos al pin 8
- Sensor de proximidad conectamos al pin 7

Receptor

- Niquelina conectamos al puerto 6
- Foco conectamos al puerto 9
- Ventilador al puerto 7

Para Java solo tener en cuenta que COMM se está utilizando.