UNIVERSIDAD TECNOLÓGICA DE LA SIERRA HIDALGUENSE

ORGANISMO DESCENTRALIZADO DE LA ADMINISTRACIÓN PÚBLICA ESTATAL

#SomosVenados #SomosGanadores #SomosUTSH

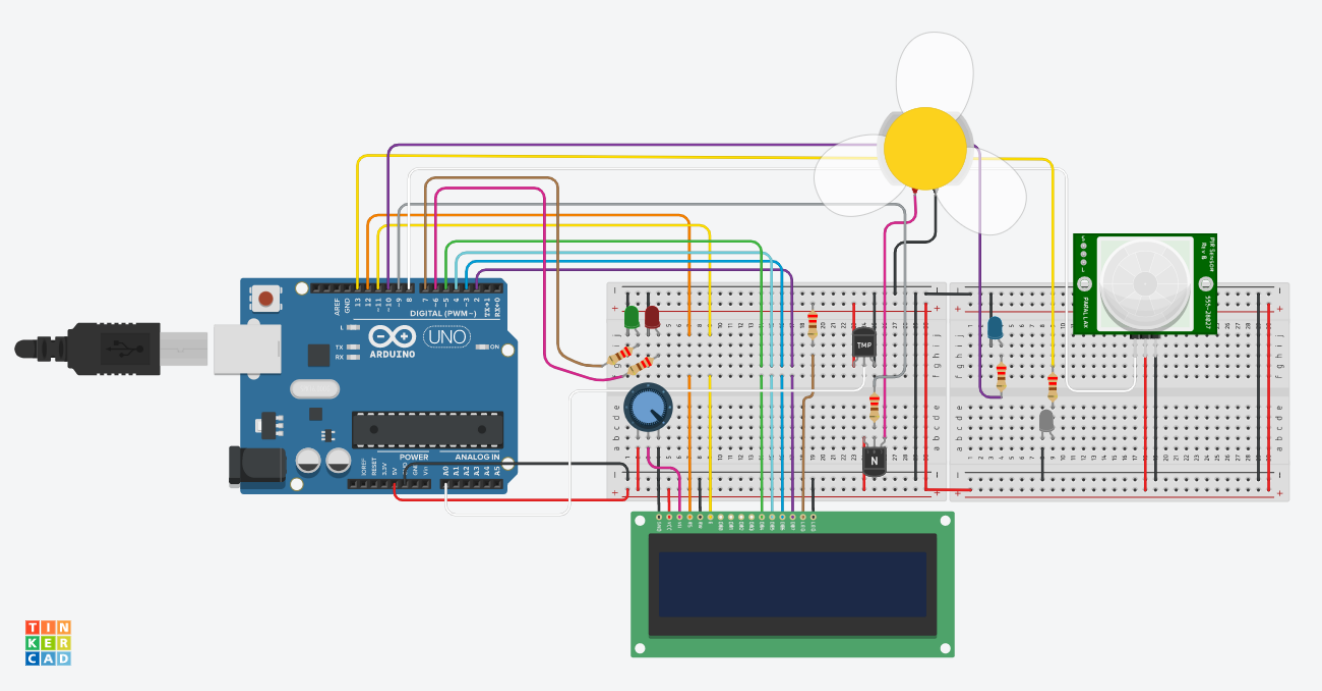
Alumnos: Jair Eddell Hernández Cabrera.
Jonatan Raúl Rodríguez Alarcón.

Profesor: Ing. Manuel Pelcastre Ibarra.

Asignatura: Principios IOT

Actividad: Reporte proyecto final.

DIAGRAMA DE CONEXIONES:



El diagrama de componentes fue elaborado en Tinkercad, un software gratuito de diseño y modelado 3D que ofrece un simulador online que permite crear diagramas con componentes electrónicos y poder probar el código con dichos componentes.

Para el diagrama se utilizaron los siguientes componentes:

- ✓ Arduino Uno R3
- ✓ 2 mini Protoboards
- ✓ 4 leds (Verde, Rojo, Azul, Blanco)
- ✓ 6 resistencias de 220Ω
- ✓ 1 potenciómetro 10k
- ✓ 1 transistor npm 2n2222
- ✓ 1 sensor de Temperatura TMP36
- ✓ 1 sensor Pir HC-sr501
- ✓ 1 ventilador 5v
- ✓ 1 pantalla LCD×216

CÓDIGO DEL PROGRAMA:

```ino
//Agrega la libreria para el LCD
#include <LiquidCrystal.h>

//Agregar la libreria para el sensor de Temperatura DHT11
#include <DHT.h>
#include <DHT_U.h>

//Pines para encender la Pantalla LCD
int rs = 12;
int e = 11;
int d4 = 5;
int d5 = 4;

//Tiempo de Actuzlización de información del LCD
int dTime = 1000;

//Envio de Informacion al LCD
int d6 = 3;
int d7 = 2;
int numero = 0;

//Configuración LCD
LiquidCrystal lcd(rs, e, d4, d5, d6, d7);   //Verificar que la Pantalla LCD este Conectada

//TEMPERATURA
int sensor; //variable que controla la informacion del sensor
float temperatura; //variable que calcula la Tenperatura Ambiente

//VENTILADOR
const int ventilador = 9 ; // Constante que maneja las instrucciones del Motor

//INDICADOR DEL SISTEMA DE TEMPERATURA
int ledApagado = 7;
int ledEncendido = 6;

//SISTEMA DE RIEGO POR TIEMPO
int sistemaRiego = 10;

//SISTEMA DE MOVIMIENTO
int pirState = LOW;
int val = 0;

//SISTEMA DE ILUMINACIÓN POR MOVIMIENTO
const int ledIluminacion = 13;
const int sensorPir = 8;

void setup() {
    Serial.begin(9600);     //Inicializar el Monitor Serial
    lcd.begin(16, 2);       //Inicializar la pantalla lcd 16x2
    pinMode(ledApagado, OUTPUT);        //Inicializar led indicador de Temperatura Estandar
    pinMode(ledEncendido, OUTPUT);      //Inicializar led indicador de Temperatura Alta
    pinMode(ledIluminacion, OUTPUT);    //Inicializar la iluminación por Movimiento
    pinMode(sistemaRiego, OUTPUT);      //Inicializar leds de Riego por Temperatura
    pinMode(sensorPir, INPUT);          //Inicializar el Sensor de Movimiento
}

void loop() {

    lcd.clear(); //Limpia el contenido de la pantalla LCD
    lcd.print("Temperatura: "); //Imprimimos la palabra 'Temperatura' en la pantalla LCD

    sensor = analogRead(A0);    //Leemos el valor del sensor de Temperatura TMP36 del pin análogo A0
    temperatura = ((400.00 * sensor) / 1024); //Formula para calcular la temperatura en funcion del voltaje del sensor
    lcd.setCursor(0,1);     //Situamos la pocisión de impresión en la pantalla LCD

    lcd.print(temperatura); //Imprimimos el Resultado de la Temperatura en la pantalla LCD
    lcd.print(" *C  "); //Imprimimos la letra 'C' de grados Celsius
```

```
68
69      delay(dTime); //Se genera un Retraso de 1 segundo para actualizar la Temperatura
70
71      if (temperatura>30){  //Se Realiza una Comprobación de Temperatura para accionar el Ventilador y el indicador
72          Serial.print(" TEMPERATURA ALTA   "); //Se muestra en el puerto Serial el Mensaje que indica la Temperatura
73          digitalWrite(ventilador, HIGH);     //Si se llega a los 30°C se enciende el Motor del Ventilador
74          digitalWrite(ledEncendido, HIGH);   //Se enciende el Led del indicador del sistema de Temperatura Alta
75          digitalWrite(ledApagado, LOW);      //Se mantiene apagado el Led indicador de Temperatura Estandar
76      } else{  //Si no sucede la condición
77          Serial.print(" TEMPERATURA ESTANDAR   "); //Se muestra en el puerto Serial el Mensaje que indica la Temperatura
78          digitalWrite(ventilador, LOW);      //Si NO se llega a los 30°C se Mantiene apagado el Motor del Ventilador
79          digitalWrite(ledEncendido, LOW);    //Se apaga el Led del indicador del sistema de Temperatura Alta
80          digitalWrite(ledApagado, HIGH);     //Se enciende el Led indicador de Temperatura Estandar
81      }
82
83      Serial.println(temperatura);    //Se muestra en el puerto Serial la Temperatura Actual
84
85      val = digitalRead(sensorPir);   //Se lee y almacena el valor del sensor PIR HC-SR501 del pin digital 8
86
87      if(val == HIGH) { //Se Verifica si esta activado el sensor de Movimiento
88          digitalWrite(ledIluminacion, HIGH); //Se enciende el Led del sistema de iluminación por Movimiento
89          delay(10000);   //Se genera un Retraso de 10 segundos para apagar la iluminación
90          if(sensorPir == LOW) { //Se verifica si previamente estaba inactivo el sensor de Movimiento
91              Serial.println(" Invernadero Activo "); //Se muestra en el puerto Serial el Estado del Invernadero
92              pirState = HIGH;    //Se cambia el estado del Sensor a Encendido
93          }
94      } else {    //Si no sucede la condición
95          digitalWrite(ledIluminacion, LOW); //Se apaga el Led del sistema de iluminación por movimiento
96          if(pirState == HIGH) { //Se verifica si previamente estaba Activo el Sensor de Movimiento
97              Serial.println(" Invernadero Inactivo   "); //Se muestra en el puerto Serial el Estado del Invernadero
98              pirState = LOW;     //Se cambia el estado del Sensor a Apagado
99          }
100     }
101
102     /*  Para el Sensor de Temperatura:
103     1 dia equivale a 1 minuto = 60000ms
104     1 minuto equivale a 1 segundo = 1000ms
105     */
106
107     digitalWrite(sistemaRiego, HIGH);   //El sistema de Riego se Enciende 25 minutos al día
108     delay(2500);    //Se genera un Retraso de 2.5 segundos para cerrar la valvula de riego
109     digitalWrite(sistemaRiego, LOW);    //El sistema de Riego se Activa cada 12 horas (Medio día)
110     delay(30000);   //Se genera un Retraso de 30 segundos para cerrar la valvula de riego
111 }
112
113
114 /* Compartir Simulación:
115 https://www.tinkercad.com/things/ia2G6CAeOA0-proyectoinvernadero/editel?sharecode=IuqKKV8s9QXr18JrUo4g4sVBMclpDohgXjDgTI
    -3nC0
116 */
117
118
119 /*
```

Paso 1: Agrega la librería para la pantalla LCD

```
1  //Agrega la libreria para el LCD
2  #include <LiquidCrystal.h>
```

Paso 2: Establecer los pines para encender la Pantalla LCD

```
 8  //Pines para encender la Pantalla LCD
 9  int rs = 12;
10  int e = 11;
11  int d4 = 5;
12  int d5 = 4;
```

Paso 3: Establecer el tiempo de Actualización del LCD

```
14  //Tiempo de Actuzlización de información del LCD
15  int dTime = 1000;
```

Paso 4: Variables que controlan la Información del LCD

```
17  //Envio de Informacion al LCD
18  int d6 = 3;
19  int d7 = 2;
20  int numero = 0;
```

Paso 5: Verificar la Conexión del LCD

```
22  //Configuración LCD
23  LiquidCrystal lcd(rs, e, d4, d5, d6, d7);   //Verificar que la Pantalla LCD este Conectada
```

Paso 6: Variables que controlan la temperatura del Sensor TMP36

```
25  //TEMPERATURA
26  int sensor; //variable que controla la informacion del sensor
27  float temperatura; //variable que calcula la Tenperatura Ambiente
```

Paso 7: Variable que controla la información del ventilador

```
29  //VENTILADOR
30  const int ventilador = 9 ; // Constante que maneja las instrucciones del Motor
```

Paso 8: Variables que controlan la iluminación por temperatura

```
32  //INDICADOR DEL SISTEMA DE TEMPERATURA
33  int ledApagado = 7;
34  int ledEncendido = 6;
```

Paso 9: Variable que controla el sistema de Riego (LED)

```
36  //SISTEMA DE RIEGO POR TIEMPO
37  int sistemaRiego = 10;
```

Paso 10: Variable que controla el sensor PIR

```
39 //SISTEMA DE MOVIMIENTO
40 int pirState = LOW;
41 int val = 0;
```

Paso 11: Variable que controla el sistema de iluminación por Movimiento

```
43 //SISTEMA DE ILUMINACIÓN POR MOVIMIENTO
44 const int ledIluminacion = 13;
45 const int sensorPir = 8;
```

Paso 12: Inicialización de Variables

```
47 void setup() {
48     Serial.begin(9600);     //Inicializar el Monitor Serial
49     lcd.begin(16, 2);       //Inicializar la pantalla lcd 16x2
50     pinMode(ledApagado, OUTPUT);        //Inicializar led indicador de Temperatura Estandar
51     pinMode(ledEncendido, OUTPUT);      //Inicializar led indicador de Temperatura Alta
52     pinMode(ledIluminacion, OUTPUT);    //Inicializar la iluminación por Movimiento
53     pinMode(sistemaRiego, OUTPUT);      //Inicializar leds de Riego por Temperatura
54     pinMode(sensorPir, INPUT);          //Inicializar el Sensor de Movimiento
55 }
```

Paso 13: Se limpia e imprime el contenido LCD

```
57 void loop() {
58
59     lcd.clear(); //Limpia el contenido de la pantalla LCD
60     lcd.print("Temperatura: "); //Imprimimos la palabra 'Temperatura' en la pantalla LCD
```

Paso 14: Se calcula la temperatura ambiente

```
62 sensor = analogRead(A0);    //Leemos el valor del sensor de Temperatura TMP36 del pin análogo A0
63 temperatura = ((400.00 * sensor) / 1024); //Formula para calcular la temperatura en funcion del voltaje del sensor
64 lcd.setCursor(0,1);     //Situamos la pocisión de impresión en la pantalla LCD
```

Paso 15: Se Imprime la temperatura actual del Sensor

```
66 lcd.print(temperatura); //Imprimimos el Resultado de la Temperatura en la pantalla LCD
67 lcd.print(" *C  "); //Imprimimos la letra 'C' de grados Celsius
```

Paso 16: Se Actualiza la pantalla cada Segundo

```
69 delay(dTime); //Se genera un Retraso de 1 segundo para actualizar la Temperatura
```

Paso 17: Se verifica si la temperatura es superior a 30 grados para encender el ventilador

```
71  if (temperatura>30){  //Se Realiza una Comprobación de Temperatura para accionar el Ventilador y el indicador
72      Serial.print(" TEMPERATURA ALTA   "); //Se muestra en el puerto Serial el Mensaje que indica la Temperatura
73      digitalWrite(ventilador, HIGH);      //Si se llega a los 30°C se enciende el Motor del Ventilador
74      digitalWrite(ledEncendido, HIGH);    //Se enciende el Led del indicador del sistema de Temperatura Alta
75      digitalWrite(ledApagado, LOW);       //Se mantiene apagado el Led indicador de Temperatura Estandar
76  } else{  //Si no sucede la condición
77      Serial.print(" TEMPERATURA ESTANDAR   "); //Se muestra en el puerto Serial el Mensaje que indica la Temperatura
78      digitalWrite(ventilador, LOW);       //Si NO se llega a los 30°C se Mantiene apagado el Motor del Ventilador
79      digitalWrite(ledEncendido, LOW);     //Se apaga el Led del indicador del sistema de Temperatura Alta
80      digitalWrite(ledApagado, HIGH);      //Se enciende el Led indicador de Temperatura Estandar
81  }
```

Paso 18: Se muestra en el puerto serial la temperatura

```
83  Serial.println(temperatura);    //Se muestra en el puerto Serial la Temperatura Actual
```

Paso 19: Se almacena en la variable la información del sensor

```
85  val = digitalRead(sensorPir);   //Se lee y almacena el valor del sensor PIR HC-SR501 del pin digital 8
```

Paso 20: Se verifica el estado del sensor de movimiento para encender la iluminación

```
87  if(val == HIGH) { //Se Verifica si esta activado el sensor de Movimiento
88      digitalWrite(ledIluminacion, HIGH); //Se enciende el Led del sistema de iluminación por Movimiento
89      delay(10000);   //Se genera un Retraso de 10 segundos para apagar la iluminación
90      if(sensorPir == LOW) { //Se verifica si previamente estaba inactivo el sensor de Movimiento
91          Serial.println(" Invernadero Activo "); //Se muestra en el puerto Serial el Estado del Invernadero
92          pirState = HIGH;    //Se cambia el estado del Sensor a Encendido
93      }
94  } else {    //Si no sucede la condición
95      digitalWrite(ledIluminacion, LOW); //Se apaga el Led del sistema de iluminación por movimiento
96      if(pirState == HIGH) { //Se verifica si previamente estaba Activo el Sensor de Movimiento
97          Serial.println(" Invernadero Inactivo   "); //Se muestra en el puerto Serial el Estado del Invernadero
98          pirState = LOW;     //Se cambia el estado del Sensor a Apagado
99      }
100 }
```

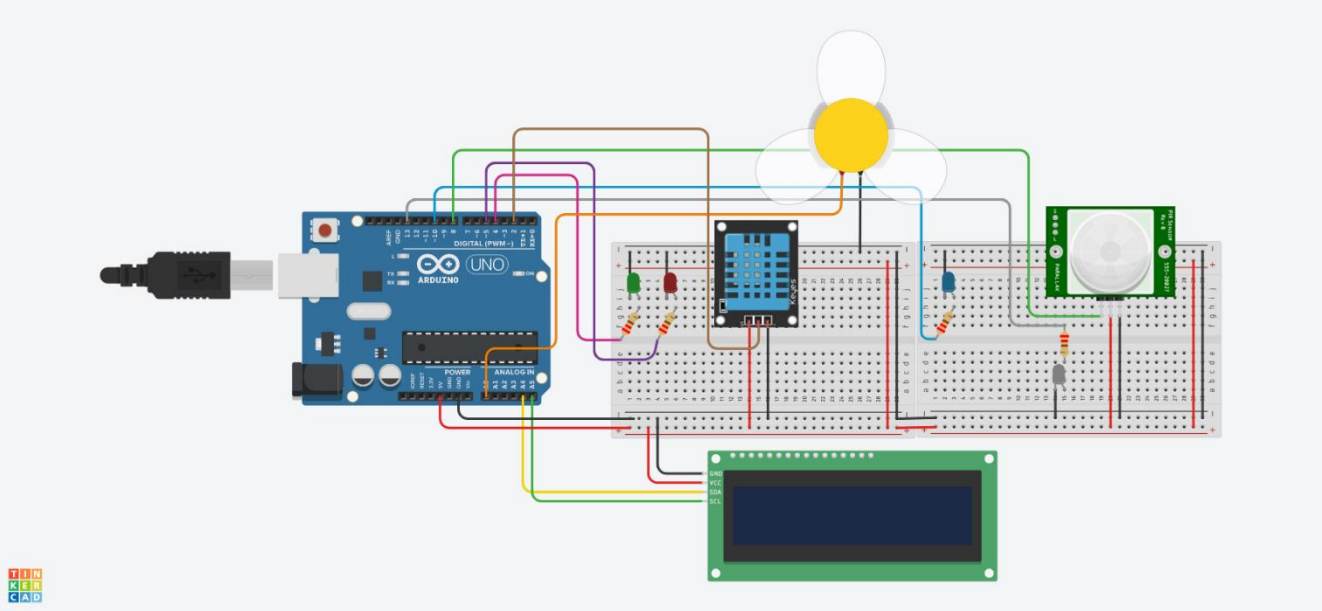
Paso 21: Se enciende el sistema de riego cada medio día, durante 25 minutos

```
102     /*  Para el Sensor de Temperatura:
103     1 dia equivale a 1 minuto = 60000ms
104     1 minuto equivale a 1 segundo = 1000ms
105     */
106
107     digitalWrite(sistemaRiego, HIGH);   //El sistema de Riego se Enciende 25 minutos al día
108     delay(2500);    //Se genera un Retraso de 2.5 segundos para cerrar la valvula de riego
109     digitalWrite(sistemaRiego, LOW);    //El sistema de Riego se Activa cada 12 horas (Medio día)
110     delay(30000);   //Se genera un Retraso de 30 segundos para cerrar la valvula de riego
111 }
```

PROBAR SIMULACIÓN:

https://www.tinkercad.com/things/ia2G6CAeOA0?sharecode=hc81ClTn5URA3pkA7YnT4ZGrVWeDfV8QDM7kmD6cZU

DIAGRAMA DE CONEXIONES V2:



El diagrama de componentes fue elaborado en Tinkercad, (un software gratuito de diseño y modelado 3D) que ofrece la posibilidad de montar, programar y simular circuitos electrónicos con Arduino. Para ello, se deberá crear una cuenta de usuario e iniciar sesión.

En este diagrama se utilizó el Módulo de interfaz serial I2C que permite controlar la pantalla LCD con una mayor facilidad, reduciendo en gran medida la cantidad de cables y simplificando la conexión de los componentes del circuito.

Para el diagrama se utilizaron los siguientes componentes:

- ✓ Arduino Uno R3
- ✓ 2 mini Protoboards
- ✓ 4 leds (Verde, Rojo, Azul, Blanco)
- ✓ 4 resistencias de 220Ω
- ✓ 1 sensor de Temperatura DHT 11
- ✓ 1 sensor Pir HC-sr501
- ✓ 1 ventilador 5v
- ✓ 1 módulo de Comunicaciones I2C
- ✓ 1 pantalla LCD×216

CÓDIGO DEL PROGRAMA:

```ino
//Se Añaden las Librerias para el LCD
#include <Wire.h> //Libreria para la Interfaz I2C
#include <LiquidCrystal_I2C.h>  //Libreria para la Pantalla LCD

//Se Añaden las Librerias para el Sensor de Temperatura
#include <DHT.h>
#include <DHT_U.h>

//Constante del Sistema de Riego
const int LEDpin = 10;   //Constante que indica el Pin de conexión del Led del Sistema de Riego

/*  Para el Sistema de Riego:
1 dia equivale a 1 minuto = 60000ms
1 minuto equivale a 1 segundo = 1000ms
*/
const long onDuration = 3500; //Constante de Tiempo de Encendido del LED en milisegundos
const long offDuration = 15000; //Constante de Tiempo de Apagado del LED en milisegundos
int LEDState = HIGH;  //Variable que almacena el estado Actual del LED
long rememberTime = 0;  //Variable que almacenará el Tiempo de Inicio en milisegundos

//Variables del Sensor de Movimiento
int pirState = LOW; //Variable que almacena el estado Actual del Sensor
const int sensorPir = 8; //Constante que indica el Pin de conexión del sensor
int val = 0; //Variable auxiliar que almacena la información del Sensor

//Sistema de Iluminacion activado por Movimiento
const int ledIluminacion = 13; //Constante que indica el Pin de conexión del Led de Iluminación

//Sistema de Alerta por Temperatura
const int tempAlta = 5; //Constante que indica el Pin de conexión del Led de Temperatura Alta
const int tempBaja = 4; //Constante que indica el Pin de conexión del Led de Temperatura Baja

//Variables del Sensor de Temperatura
const int sensor = 2; //Constante que indica el Pin de conexión del Sensor
int Temperatura;   //Variable que almacena la temperatura registrada por el sensor

//Varibles de Ventilador
const int cooler = A0; //Constante que indica el Pin de conexión del Ventilador

//Se crea un objeto llamado "dht" con el número de pin de conexión del sensor y se indica el modelo
DHT dht(sensor, DHT11);

//Se crea un objeto llamado "lcd" con la dirección del LCD y se indican las columnas y filas del mismo
LiquidCrystal_I2C lcd(0x27,16,2);


void setup() {
  //Se inicializan de las variables
  Serial.begin(9600);   //Se inicializa el Puerto Serial
  dht.begin();  //Se inicializa el sensor de Temperatura

  lcd.init(); //Se inicializa el LCD
  lcd.backlight();  //Se Enciende la luz de fondo del LCD.

  //Se inicializan sensor de Movimiento
  pinMode(ledIluminacion, OUTPUT);  //Se inicializa el Led activado por Movimiento
  pinMode(sensorPir, INPUT);

  //Se inicializan los Leds de Alerta de Temperatura
  pinMode(tempAlta, OUTPUT);
  pinMode(tempBaja, OUTPUT);

  //Se inicializan los Leds del Sistema de Riego
  pinMode(LEDpin,OUTPUT);
  digitalWrite(LEDpin,LEDState);  //Se inicializa según su estado
}

void loop() {

  lcd.clear(); //Se limpia el contenido del LCD
  Temperatura = dht.readTemperature();  //Se almacena la temperatura ambiente en la variable

  //Se Muestra La Temperatuar en el LCD.
  lcd.setCursor(0, 0);  //Se indica la posición de las letras
  lcd.print("Temperatura: "); //Se muestra la cadena de carácteres
  lcd.setCursor(0, 1);  //Se indica la nueva posición del Mensaje
  lcd.print(Temperatura); //Se imprime la temperatura almacenada en la variable
  lcd.println(" *C"); //Se concatenan los caracteres para indicar que se trata de grados Centigrados
  delay(500); //Se actualiza la Información de la Temperatura cada 500 milisegundos
```

```
80
81  if(Temperatura >= 25) { //Se Verifica si la Temperatura es Superior 24°C
82    Serial.print("Temperatura ALTA: "); //Se Muestra la Alerta en el Monitor Serial
83    Serial.print(Temperatura);  //Se Muestra la Temperatura actual en el Monitor Serial
84    Serial.println("°C"); //Se concatenan los caracteres para indicar que se trata de grados Centigrados
85    digitalWrite(tempBaja,LOW); //Se Apaga el Led indicador de temperatura Estandar
86    digitalWrite(tempAlta,HIGH);  //Se Enciende el Led de Alerta de Temperatura Alta
87    analogWrite(cooler, 500); //Si la temperatura alcanza los 25°C Se enciende el Ventilador
88  }
89
90  else {  //Si no sucede la condición
91    Serial.print("Temperatura NORMAL: "); //Se Muestra la Alerta en el Monitor Serial
92    Serial.print(Temperatura);  //Se Muestra la Temperatura actual en el Monitor Serial
93    Serial.println("°C"); //Se concatenan los caracteres para indicar que se trata de grados Centigrados
94    digitalWrite(tempAlta,LOW); //Se Apaga el Led de Alerta de Temperatura Alta
95    digitalWrite(tempBaja,HIGH);  //Se Enciende el Led indicador de Temperatura Estandar
96    analogWrite(cooler, 0); //Si no se superan los 24°C se mantiene apagado el Ventilador
97  }
98
99  val = digitalRead(sensorPir); //Se lee y almacena el valor del sensor PIR HC-SR501 del pin digital 8
100
101 if(val == HIGH) { //Se Verifica si esta activado el sensor de Movimiento
102   digitalWrite(ledIluminacion, HIGH); //Se enciende el Led del sistema de iluminación por Movimiento
103   if(sensorPir == LOW) { //Se verifica si previamente estaba inactivo el sensor de Movimiento
104     Serial.println(" Invernadero Activo "); //Se muestra en el puerto Serial el Estado del Invernadero
105     pirState = HIGH;  //Se Invierte el estado del Sensor a Encendido
106   }
107 }
108
109 else {  //Si no sucede la condición
110   digitalWrite(ledIluminacion, LOW); //Se apaga el Led del sistema de iluminación por movimiento
111   if(pirState == HIGH) { //Se verifica si previamente estaba Activo el Sensor de Movimiento
112     Serial.println(" Invernadero Inactivo "); //Se muestra en el puerto Serial el Estado del Invernadero
113     pirState = LOW;   //Se Invierte el estado del Sensor a Apagado
114   }
115 }
116
117 /*  Para el Sistema de Riego:
118 1 día equivale a 1 minuto = 60000ms
119 1 minuto equivale a 1 segundo = 1000ms
120 */
121
122 if( LEDState == HIGH ) {  //Se verifica si el estado del LED esta Activo
123   if( (millis() - rememberTime) >= onDuration){ //Se comprueba si el tiempo Actual MENOS el tiempo de Inicio es SUPERIOR al tiempo de ENCENDIDO del LED
124   LEDState = LOW; //Se Invierte el estado del LED
125   rememberTime = millis();  //Se actualiza el Tiempo Inicial mediante la función millis()
126   }
127 }
128
129 else  {   //Si no sucede la condición
130   if( (millis() - rememberTime) >= offDuration){  //Se comprueba si el tiempo Actual MENOS el tiempo de Inicio es SUPERIOR o IGUAL al tiempo de APAGADO del LED
131   LEDState = HIGH;  //Se Invierte el estado del LED
132   rememberTime = millis();  //Se actualiza el Tiempo Inicial mediante la función millis()
133   }
134 }
135
136 //LED encendido por medio de la función millis()
137 digitalWrite(LEDpin,LEDState); //Según el estado del LED, se Encenderá o se Apagará
138 }
```

Paso 1: Se Agrega la librería para la Interfaz I2C y la pantalla LCD

```
1  //Se Añaden las Librerias para el LCD
2  #include <Wire.h> //Librería para la Interfaz I2C
3  #include <LiquidCrystal_I2C.h>  //Libreria para la Pantalla LCD
```

Paso 2: Se Agrega la librería para el sensor de Temperatura DHT 11

```
5  //Se Añaden las Librerias para el Sensor de Temperatura
6  #include <DHT.h>
7  #include <DHT_U.h>
```

Paso 3: Se declaran las constantes y variables para encendido periódico del sistema de Riego

```
//Constante del Sistema de Riego
const int LEDpin = 10;   //Constante que indica el Pin de conexión del Led del Sistema de Riego

/*  Para el Sistema de Riego:
1 dia equivale a 1 minuto = 60000ms
1 minuto equivale a 1 segundo = 1000ms
*/
const long onDuration = 3500; //Constante de Tiempo de Encendido del LED en milisegundos
const long offDuration = 15000; //Constante de Tiempo de Apagado del LED en milisegundos
int LEDState = HIGH;  //Variable que almacena el estado Actual del LED
long rememberTime = 0;  //Variable que almacenará el Tiempo de Inicio en milisegundos
```

Paso 4: Se declaran las constantes y variables para el sensor de movimiento

```
//Variables del Sensor de Movimiento
int pirState = LOW; //Variable que almacena el estado Actual del Sensor
const int sensorPir = 8; //Constante que indica el Pin de conexión del sensor
int val = 0; //Variable auxiliar que almacena la información del Sensor
```

Paso 5: Se declara la constante para la iluminación del sensor de movimiento

```
//Sistema de Iluminacion activado por Movimiento
const int ledIluminacion = 13; //Constante que indica el Pin de conexión del Led de Iluminación
```

Paso 6: Se declaran las constantes de alerta de temperatura

```
//Sistema de Alerta por Temperatura
const int tempAlta = 5; //Constante que indica el Pin de conexión del Led de Temperatura Alta
const int tempBaja = 4; //Constante que indica el Pin de conexión del Led de Temperatura Baja
```

Paso 7: Se declaran la constante y variable que almacena la información del sensor de temperatura

```
//Variables del Sensor de Temperatura
const int sensor = 2; //Constante que indica el Pin de conexión del Sensor
int Temperatura;   //Variable que almacena la temperatura registrada por el sensor
```

Paso 8: Se declara la constante que controla el encendido del ventilador

```
37  //Varibles de Ventilador
38  const int cooler = A0; //Constante que indica el Pin de conexión del Ventilador
```

Paso 9: Se instancia un objeto con las propiedades del sensor de temperatura

```
40  //Se crea un objeto llamado "dht" con el número de pin de conexión del sensor y se indica el modelo
41  DHT dht(sensor, DHT11);
```

Paso 10: Se instancia un objeto con las propiedades de la pantalla LCD

```
43  //Se crea un objeto llamado "lcd" con la dirección del LCD y se indican las columnas y filas del mismo
44  LiquidCrystal_I2C lcd(0x27,16,2);
```

Paso 11: Se inicializan las variables que controlan el puerto serial, el sensor de temperatura, el sensor de movimiento, la pantalla LCD y los Leds de iluminación

```
47  void setup() {
48    //Se inicializan de las variables
49    Serial.begin(9600);   //Se inicializa el Puerto Serial
50    dht.begin();  //Se inicializa el sensor de Temperatura
51
52    lcd.init(); //Se inicializa el LCD
53    lcd.backlight();  //Se Enciende la luz de fondo del LCD.
54
55    //Se inicializan sensor de Movimiento
56    pinMode(ledIluminacion, OUTPUT);  //Se inicializa el Led activado por Movimiento
57    pinMode(sensorPir, INPUT);
58
59    //Se inicializan los Leds de Alerta de Temperatura
60    pinMode(tempAlta, OUTPUT);
61    pinMode(tempBaja, OUTPUT);
62
63    //Se inicializan los Leds del Sistema de Riego
64    pinMode(LEDpin,OUTPUT);
65    digitalWrite(LEDpin,LEDState);  //Se inicializa según su estado
66  }
```

Paso 12: Se Limpia el LCD y se almacena la temperatura ambiente en la variable

```
68  void loop() {
69
70    lcd.clear(); //Se limpia el contenido del LCD
71    Temperatura = dht.readTemperature();  //Se almacena la temperatura ambiente en la variable
```

Paso 13: Se Imprime la temperatura actual en la pantalla LCD

```
//Se Muestra La Temperatuar en el LCD.
lcd.setCursor(0, 0);  //Se indica la posición de las letras
lcd.print("Temperatura: "); //Se muestra la cadena de carácteres
lcd.setCursor(0, 1);  //Se indica la nueva posición del Mensaje
lcd.print(Temperatura); //Se imprime la temperatura almacenada en la variable
lcd.println(" *C"); //Se concatenan los caracteres para indicar que se trata de grados Centigrados
delay(500); //Se actualiza la Información de la Temperatura cada 500 milisegundos
```

Paso 14: Se verifica si la temperatura es superior a 24°C, para encender el led de alerta y el ventilador

```
if(Temperatura >= 25) { //Se Verifica si la Temperatura es Superior 24°C
    Serial.print("Temperatura ALTA: "); //Se Muestra la Alerta en el Monitor Serial
    Serial.print(Temperatura);  //Se Muestra la Temperatura actual en el Monitor Serial
    Serial.println("°C"); //Se concatenan los caracteres para indicar que se trata de grados Centigrad
os
    digitalWrite(tempBaja,LOW); //Se Apaga el Led indicador de temperatura Estandar
    digitalWrite(tempAlta,HIGH);  //Se Enciende el Led de Alerta de Temperatura Alta
    analogWrite(cooler, 500); //Si la temperatura alcanza los 25°C Se enciende el Ventilador
  }
```

Paso 15: Si la temperatura no es superior a 24°C, se apaga el led de alerta y el ventilador

```
else {  //Si no sucede la condición
  Serial.print("Temperatura NORMAL: "); //Se Muestra la Alerta en el Monitor Serial
  Serial.print(Temperatura);  //Se Muestra la Temperatura actual en el Monitor Serial
  Serial.println("°C"); //Se concatenan los caracteres para indicar que se trata de grados Centigrados
  digitalWrite(tempAlta,LOW); //Se Apaga el Led de Alerta de Temperatura Alta
  digitalWrite(tempBaja,HIGH);  //Se Enciende el Led indicador de Temperatura Estandar
  analogWrite(cooler, 0); //Si no se superan los 24°C se mantiene apagado el Ventilador
}
```

Paso 16: Se verifica si el sensor PIR está en movimiento para encender el led e invertir el estado del sensor

```
val = digitalRead(sensorPir); //Se lee y almacena el valor del sensor PIR HC-SR501 del pin digital 8

if(val == HIGH) { //Se Verifica si esta activado el sensor de Movimiento
  digitalWrite(ledIluminacion, HIGH); //Se enciende el Led del sistema de iluminación por Movimiento
  if(sensorPir == LOW) { //Se verifica si previamente estaba inactivo el sensor de Movimiento
    Serial.println(" Invernadero Activo "); //Se muestra en el puerto Serial el Estado del Invernadero
    pirState = HIGH;  //Se Invierte el estado del Sensor a Encendido
  }
}
```

Paso 17: Si el sensor PIR no está en movimiento se invierte el estado del sensor y se apaga el led

```
else {  //Si no sucede la condición
  digitalWrite(ledIluminacion, LOW); //Se apaga el Led del sistema de iluminación por movimiento
  if(pirState == HIGH) { //Se verifica si previamente estaba Activo el Sensor de Movimiento
    Serial.println(" Invernadero Inactivo "); //Se muestra en el puerto Serial el Estado del Invernadero
    pirState = LOW;   //Se Invierte el estado del Sensor a Apagado
  }
}
```

Paso 18: Se verifica el tiempo de ejecución es suficiente apagar el sistema de Riego

```
/*  Para el Sistema de Riego:
1 dia equivale a 1 minuto = 60000ms
1 minuto equivale a 1 segundo = 1000ms
*/

if( LEDState == HIGH ) {  //Se verifica si el estado del LED esta Activo
  if( (millis() - rememberTime) >= onDuration){ //Se comprueba si el tiempo Actual MENOS el tiempo de Inicio es SUPERIOR al tiempo de ENCENDIDO del LED
  LEDState = LOW; //Se Invierte el estado del LED
  rememberTime = millis();  //Se actualiza el Tiempo Inicial mediante la función millis()
  }
}
```

Paso 19: Se verifica el tiempo de ejecución es suficiente para encender el sistema de Riego

```
else  {   //Si no sucede la condición
  if( (millis() - rememberTime) >= offDuration){  //Se comprueba si el tiempo Actual MENOS el tiempo de Inicio es SUPERIOR o IGUAL al tiempo de APAGADO del LED
  LEDState = HIGH;  //Se Invierte el estado del LED
  rememberTime = millis();  //Se actualiza el Tiempo Inicial mediante la función millis()
  }
}
```

Paso 20: Se enciende o se apaga el sistema de Riego según el estado actual del LED

```
  //LED encendido por medio de la función millis()
  digitalWrite(LEDpin,LEDState); //Según el estado del LED, se Encenderá o se Apagará
}
```

JUSTIFICACIÓN DE COMPONENTES

Microcontrolador ATmega 328P-Pu:

- **Familia:** AVR ATmega
- **CPU:** 8- bits AVR
- **Voltaje:** 1.8v a 5.5v
- **Frecuencia Máxima:** 20Mhz
- **Comunicación:** 12C, SPI, UART
- **Temperatura:** -40°C - 85°C
- **Dimensiones:** 4.6 mm × 34.8 mm × 7.5mm
- **Peso:** 2.2g
- **Pines:** 28
- **I/O:** 23
- **Memoria del programa:** 32 Kb
- **Memoria Flash:** 32 kb
- **Memoria Ram:** 2 kb
- **Memoria Rom:** 1 kb
- **Memoria EEPROM:** 1024 bytes
- **ADC:** 6 canales

Display LCD 2 × 16:

- **Resolución:** 2 líneas × 16 caracteres
- **Modelo:** LC1622-BMDWH6
- **Controlador:** S6A0069
- **Voltaje de Operación:** 5v Cc
- **Dimensiones de la Pantalla:** 65mm × 16mm
- **Temperatura:** -20°C - 75°C
- **Peso:** 32g
- **Corriente Máxima:** 25mA
- **Interfaz de comunicación:** Paralelo 4 u 8 bits
- **Color Texto:** Negro
- **Backlight:** Verde amarillento

Modulo Bluetooth HC–05:

- **Voltaje:** 3.6V – 6V DC
- **Consumo de Corriente:** 50mA
- **Bluetooth:** v2.0 + EDR
- **Frecuencia:** IEC 2.4GHz
- **Alcance:** 10 m
- **Interfaz de Comunicación:** Serial, TTL
- **Seguridad:** Autenticación y Encriptación
- **Potencia de Transmisión:** 4dBm
- **Velocidad:** 1200bps -1.3mb
- **Compatibilidad:** Android
- **Temperatura:** -20°C – 75°C
- **Peso:** 3.6gr

Sensor de Temperatura y Humedad HDT11:

- **Voltaje de Operación:** 3V - 5V DC
- **Rango de medición de temperatura:** 0 a 50 °C
- **Precisión de medición de temperatura:** ±2.0 °C
- **Resolución Temperatura:** 0.1°C

- **Rango de medición de humedad:** 20% a 90% RH.
- **Precisión de medición de humedad:** 5% RH.
- **Resolución Humedad:** 1% RH
- **Tiempo de censado:** 1 seg.
- **Interface digital:** Single-bus (bidireccional)
- **Modelo:** DHT11
- **Dimensiones:** 16 × 12 5 mm
- **Peso:** 1 gr.

Ventilador de 5V:

- **Marca:** Creality
- **Voltaje:** 5V
- **Corriente Nominal:** 100mA
- **Velocidad:** 7000rpm ± 10%
- **Diámetro tornillo:** 4.5mm
- **Tiempo de vida útil:** 50000 horas
- **Temperatura de funcionamiento:** -10 °C hasta 70°C
- **Longitud de cable:** 18cm
- **Dimensiones:** 40x40x10mm

Sensor de temperatura TMP36:

- **Voltaje de funcionamiento:** 2.7 a 5.5 V
- **Salida:** Analógica
- **Consumo de corriente en reposo:** 0.5 uA máx
- **Factor de escala:** 10 mV / °C
- **Precisión:** ±2 °C sobre temperatura
- **Linealidad:** ±0.5 °C
- **Rango de temperatura:** -40 °C a +125 °C

Transistor 2N2222 NPN:

- **Voltaje colector emisor en corte:** 60V (Vceo)
- **Corriente de colector constante:** 800mA (Ic)
- **Potencia total disipada:** 500mW(Pd)
- Ganancia o hfe 35 mínima.
- **Frecuencia de trabajo:** 250 Mhz (Ft)
- **Encapsulado de metal:** TO-92.
- **Dimensiones:** 36 mm * 18 mm * 1.5 mm.
- Estructura NPN.

Potenciómetro de 10K:

- **Tipo:** Trimpot
- **Tipo de ajuste:** Ajuste superior
- **Número de pines:** 3
- **Dimensiones de la base:** 0.9 cm x 0.5 cm x 0.9 cm
- **Resistencia:** 10k ohm
- **Número de vueltas:** 25
- **Potencia:** 500mW
- **Tolerancia:** ±10%

- **Coeficiente de temperatura:** ±100 ppm/ °C

Resistencia 220 ohm:

- **Valor:** 220 Ω
- **Potencia máxima:** 250 mWatts ≈ ¼ Watt
- **Tolerancia:** ±5%
- **Material:** Carbón

Sensor PIR HC-SR501:

- **Modelo:** HC-SR501
- **Voltaje de alimentación:** 4.5V a 12V DC
- **Voltaje de salida (TTL):** 3.3V
- **Distancia de detección:** 3 a 7 Metros Apróx.
- **Dimensiones:** 32.7mm x 29mm
- **Angulo de detección:** 90° a 110°
- **Intervalo de tiempo de alarma:** 5 s a 5 m
- **Ajustes de rango:** De detección y tiempo de alarma activa
- **Consumo de corriente en reposo:** < 50 µA
- **Temperatura de trabajo:** -20°C a 70°C
- **Peso:** 6g

Módulo de Comunicación I2C:

- **Microcontrolador:** ATMEL ATMEGA328
- **Voltaje de entrada:** 5V~9V
- **Voltaje de salida:** 3.3V/5V
- **Pines digitales entradas/salidas:** 14
- **Pines analógicos entradas/salidas:** 6
- **Interfaz (protocolo):** I2C/TWI/SPI
- **Potenciómetro:** Ajustar contraste y luz de fondo
- **Líneas de salida:** 4
- **Dirección del dispositivo:** 0x20 / 0x27
- **Tamaño:** 5.4cm x 1.9cm
- **Peso:** 16 gramos

Leds:

- **Voltaje de Operación:** Rojo:1.8V / Verde:2.25V / Azul:3.0V
- **Luminosidad:** (R:800, G:4000, B:900) MCD
- **Ángulo de visión:** 20º
- **Diámetro:** 5mm
- **Temperatura de trabajo:** -40ºC hasta 70ºC

PORQUE MOTIVO SE ELIGIERON LOS SENSORES Y COMPONENTES

- Por la accesibilidad a las librerías.
- Por sus bajos costos.
- Por la documentación disponible.

Microcontrolador Atmega 328P-Pu:

Es un dispositivo basado en la arquitectura RISC estos son de muy alto rendimiento ya que combinan una memoria 32KB ISP flash con capacidades de lectura y escritura.

Es capaz de ejecutar instrucciones de gran alcance en un solo ciclo de reloj, permitiendo que el dispositivo logre rendimientos que se aproximan 1 MIPS por MHz mientras equilibra el consumo de energía y velocidad de procesamiento.

Display LCD 2 × 16:

Es una pantalla de cristal líquido, un dispositivo de salida que permite mostrar texto, mensajes, datos numéricos o caracteres especiales en una disposición de dos líneas de texto con 16 caracteres cada una.

Modulo Bluetooth HC–05:

Es un módulo Bluetooth SPP (Protocolo de puerto serie) fácil de usar, diseñado para una configuración de conexión serie inalámbrica transparente.

El módulo Bluetooth del puerto serie está totalmente calificado con Bluetooth V2.0 + EDR (Velocidad de datos mejorada) Modulación de 3 Mbps con transceptor de radio completo de 2.4GHz y banda base. Utiliza CSR Bluecore 04-Sistema Bluetooth de un solo chip externo con tecnología CMOS y con AFH (función de salto de frecuencia adaptable).

Sensor de Temperatura y Humedad HDT11:

Es un sensor digital de temperatura y humedad de bajo costo y fácil uso. Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos.

El sensor DHT11 se caracteriza por tener la señal digital calibrada, asegurando alta estabilidad y fiabilidad a lo largo del tiempo. El sensor integra sensores resistivos para temperatura (termistor) y otro para humedad.

Ventilador de 5V:

Permite disipar la temperatura de componentes electrónicos que son propensos a calentarse demasiado, permitiendo conservar su estado fresco. Su consumo ronda los 200mA.

Sensor de Temperatura TMP36:

Es un sensor de temperatura de precisión y bajo voltaje. Proporciona una salida de voltaje que es linealmente proporcional a la temperatura Celsius. Tampoco requiere ninguna calibración externa para proporcionar precisiones en el rango de temperatura de −40°C a + 125°C. Es muy fácil de usar, simplemente se conecta a fase y neutro y lea el voltaje en el pin Vout. El voltaje de salida se puede convertir a temperatura fácilmente usando el factor de escala de 10 mV/°C.

Potenciómetro de 10K:

El potenciómetro tiene una resistencia de variación lineal que va desde 0 Ohmios hasta 10K Ohmios, la cual es controlada por el giro de un eje estriado. Su operación es muy sencilla y viene diseñado para que, a pesar de su suavidad, no se gire solo o con alguna vibración, incrementado o disminuyendo la potencia de la señal.

Resistencia 220 ohm:

Las resistencias eléctricas son componentes semiconductores que permiten limitar el paso de la corriente en un circuito eléctrico. Tiene diferentes usos, puede usarse desde un componente que ayude a generar un pequeño retardo en el funcionamiento de un circuito hasta generar diferentes frecuencias y poder aplicar un control en ciertos aparatos. Esta es una resistencia de 220 Ω y soporta una potencia máxima de ¼ W.

Sensor Pir HC-SR501:

Dentro del HC-SR501 se integra toda la electrónica necesaria para el soporte del sensor PIR propiamente dicho y entrega un pulso configurable y compatible con TTL que puede ser aceptado por la mayoría de los microcontroladores.
Este módulo permite detectar personas y animales grandes a través de un sensor PIR (Passive Infrared). Los sensores PIR se utilizan ampliamente en aplicaciones como sistemas de alarma, puertas automáticas, luces automáticas, etc.

Transistor 2N2222 NPN:

Permite regular una señal de salida en función del voltaje de entrada, cortando el flujo de corriente a partir de una pequeña señal o amplificar la señal de entrada que será enviada al motor del ventilador, para que esta funcione.

Módulo de comunicación I2C:

Permite controlar la pantalla LCD de forma fácil, reduciendo en gran medida la cantidad de cables. El módulo optimiza los recursos del microcontrolador dado que estos son realmente limitados, y este no permite conectar diferentes cantidades de sensores o tarjetas SD.

La ventaja de utilizar este dispositivo es que se puede evitar los engorrosos cables que en ocasiones se dañan y causan más problemas que beneficios.