Informe de Proyecto Final

Actualización del Laboratorio CIM

Alumno: Jorge Nicolas Franco

DNI: 30353371

Nombre de la Institución: Laboratorio CIM. UNLZ-FI

Tutor institucional y académico: Martin González

Año: 2025

Índice

Tabla de contenido

Resumen

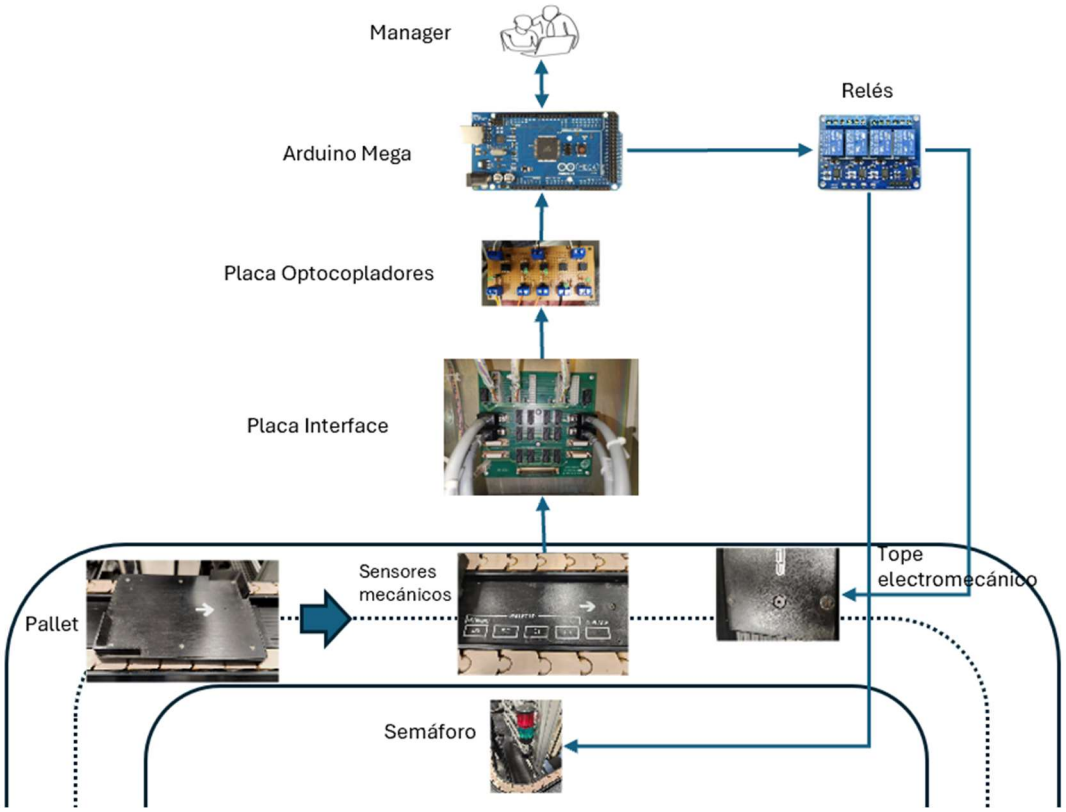
El propósito de este proyecto surgió ante la necesidad de reemplazar el PLC Sysmac CQM1 de Omron, que había quedado obsoleto y ya no garantizaba el funcionamiento

adecuado de la cinta transportadora de cuatro estaciones. Para abordar esta actualización, se optó por implementar un Arduino Mega como controlador principal del sistema.

Un paso fundamental fue realizar un análisis detallado del equipo existente mediante ingeniería inversa. Esto incluyó el estudio de los sensores presentes en cada estación y el sistema de cableado eléctrico. Esta evaluación permitió adaptar y optimizar el nuevo controlador, asegurando una operación eficiente con tecnologías más modernas.

El sistema utiliza sensores magnéticos para identificar los pallets y topes electromecánicos para detenerlos. Como parte de las pruebas iniciales, el Arduino fue programado para detectar los pallets y detenerlos en una de las cuatro estaciones, donde se realiza la lectura de su identificación.

Esquema general del Proyecto



Estado Inicial del CIM

Ilustración 1 Cinta transportadora

Ilustración 2 Gabinete 1

Ilustración 3 Gabinete 2

Ilustración 4 PLC

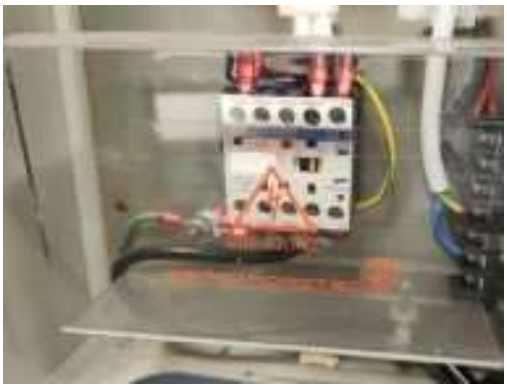


Ilustración 5 PLC input-output

Ilustración 6 Contactor

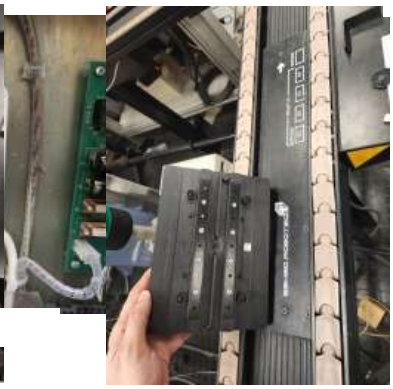


Ilustración 7 Llave start

Ilustración 8 placa de sensores

Ilustración 9 pallet lado inferior

Ilustración 10 válvula electroneumática

Ilustración 11 placa interfaz PLC

Componentes instalados durante la actualización:

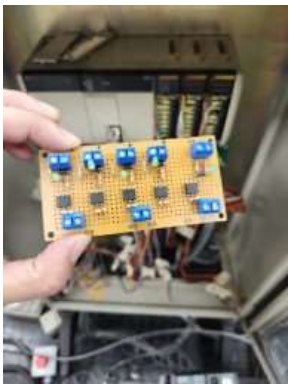


Ilustración 12 prototipo optoacopladores

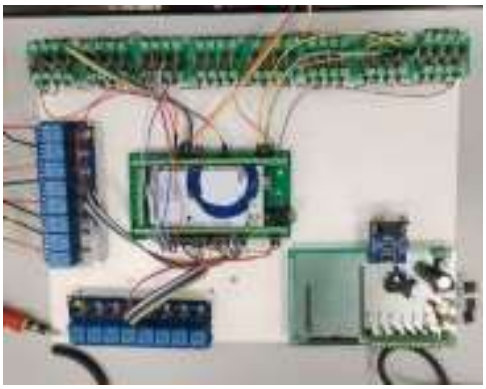


Ilustración 13 Arduino Mega, reles, RS-232

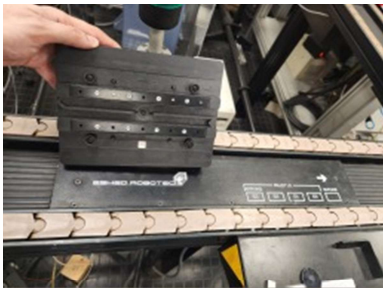
Ilustración 14 Montaje

Componentes

Cinta Transportadora:

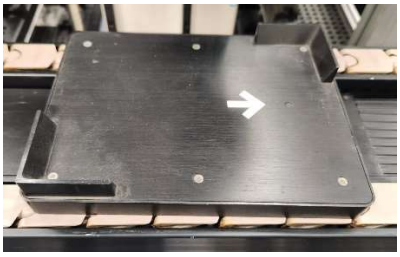
- Posee 4 estaciones de trabajo.

- Cada estación cuenta con 5 sensores mecánicos, 2 topes electromecánicos y un semáforo de señalización (verde: start, rojo: stop).



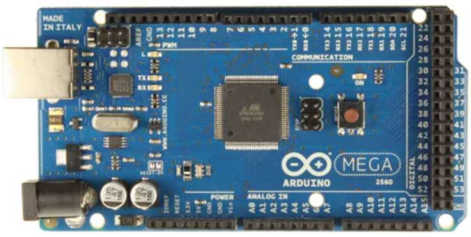
Pallets:

- Los pallets transportan los materiales, cada uno posee 5 imanes.
- 4 imanes indican la identificación o número de pallet con notación binaria.
- El 5to imán es el “In place”, indica que el Pallet está posicionado en la estación.



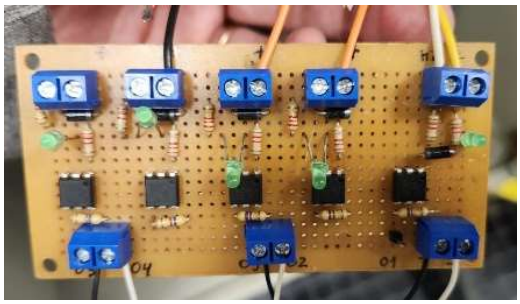
Arduino Mega:

- Reemplaza el antiguo PLC.
- Lee las señales de los sensores de la cinta.
- Acciona los topes electromecánicos para leer el número de Pallet.
- Acciona las señales luminosas, rojo “stop” y verde “run”.
- Se comunica con el Manager (PC) a través del protocolo RS-232.



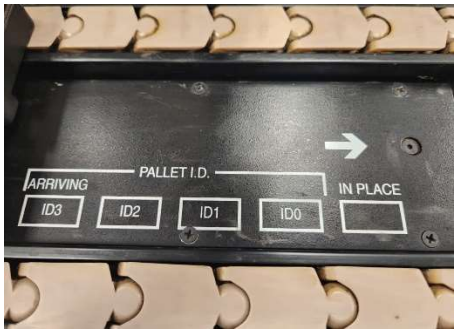
Placa de Optoacopladores:

- Los optoacopladores adaptan las señales de los sensores magnéticos al nivel de voltaje requerido por el Arduino Mega (de 24V a 5V) y protegen al sistema contra picos y fluctuaciones eléctricas.



Sensores Magnéticos:

- Detectan los imanes que poseen pallets y envían la señal amplificada a la placa de optoacopladores, a su vez esta última se comunica con el Arduino mega.



Topes Electromecánicos:

- Cada estación posee 2 topes que actúan en simultaneo, el primero detiene el Pallet a leer y el segundo asegura un gap con el Pallet siguiente para evitar colisiones entre estos.
- Se activan cuando el primer imán del Pallet pasa por el primer sensor magnético (ID3 “Irriving”) de una estación.
- Detienen el pallet para leer su identificación.

Comunicación con la PC:

- El Arduino envía el número de pallet a la PC vía RS-232
- El operador de la PC decide si se realiza una operación o si el Pallet continúa avanzando hacia la siguiente estación.

Relés:

- Energizan y des energizan los topes electromecánicos para detener los Pallets en cada estación.}

ingeniería Inversa

Identificación de terminales del PLC

Antes de avanzar con el diseño del circuito electrónico se realizó un relevamiento de las conexiones eléctricas del PLC a fin de identificar los sensores magnéticos y demás componentes de entrada y salida del sistema.

Dicho PLC posee 3 módulos de conexiones donde cada módulo tiene 16 puntos de conexión.

1. Modulo 1 – “OCH” (En general salidas de los sensores magnéticos)
2. Modulo 2 – “ID212” (“INPLACE”)
3. Modulo 3 – “OC222” (“Salidas Topes Electromecánicos y semáforos)

Identificación de los pines de las 4 estaciones

Modulos del PLC					
OCH		ID212		OC222	
	B0		B0		B0
A0	B1	A0	B1	A0	B1
A1	B2	A1	B2	A1	B2
A2	B3	A2	B3	A2	B3
A3	B4	A3	B4	A3	B4
A4	B5	A4	B5	A4	B5
A5	B6	A5	B6	A5	B6
A6	B7	A6	B7	A6	B7
A7	B8	A7	B8	A7	B8
A8		A8		A8	

Estación 1			
	Modulos del PLC		
Sensores Magneticos	OCH (IN)	ID212 (IN)	OC222 (OUT)
ID-3 (Arriving) / Tope Emec.	A1	-	
ID-2	B1		
ID-1	A0		
ID-0	NO FUNCIONA		
IN PLACE	A7		
Actuador			B0 (-)
Semaforo rojo			B2 (-)

Estación 2			
	Modulos del PLC		
Sensores Magneticos	OCH (IN)	ID212 (IN)	OC222 (OUT)
ID-3 (Arriving) / Tope Emec.	B4		
ID-2	A3		
ID-1	B3		
ID-0	A2		
IN PLACE		B5	
Actuador			B3 (-)
Semaforo rojo			B5 (-)

Estación 3			
	Modulos del PLC		
Sensores Magneticos	OCH (IN)	ID212 (IN)	OC222 (OUT)
ID-3 (Arriving) / Tope Emec.	A6		
ID-2	B6		
ID-1	A5		
ID-0	B5		
IN PLACE		A5	
Actuador			B6 (-)
Semaforo rojo			A0 (-)

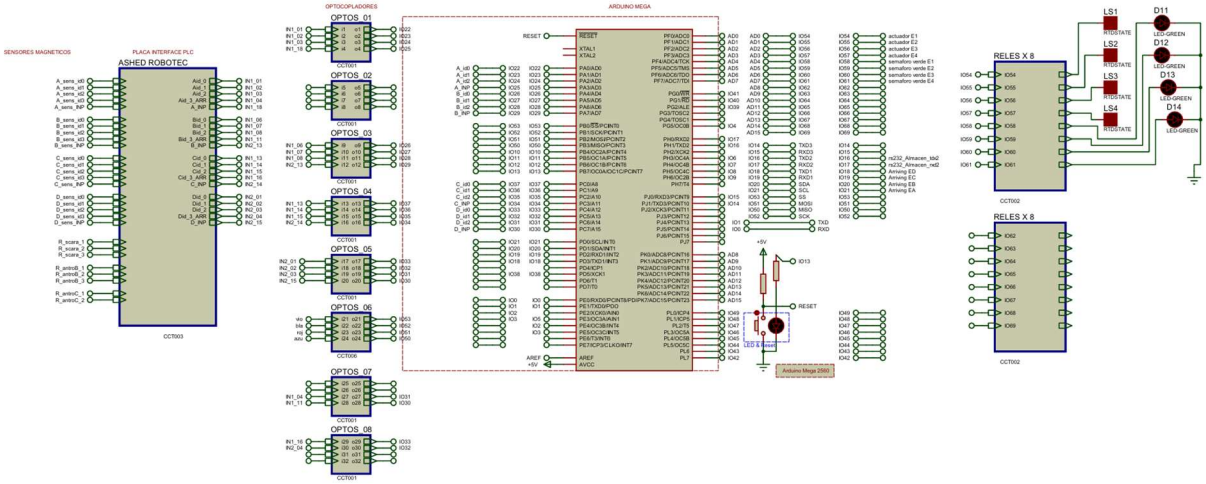
Estación 4			
	Modulos del PLC		
Sensores Magneticos	OCH (IN)	ID212 (IN)	OC222 (OUT)
ID-3 (Arriving) / Tope Emec.		A1	
ID-2		B1	
ID-1		A0	
ID-0		B0	
IN PLACE		B6	
Actuador			A1 (-)
Semaforo rojo			A3 (-)

Esquema eléctrico en Proteus

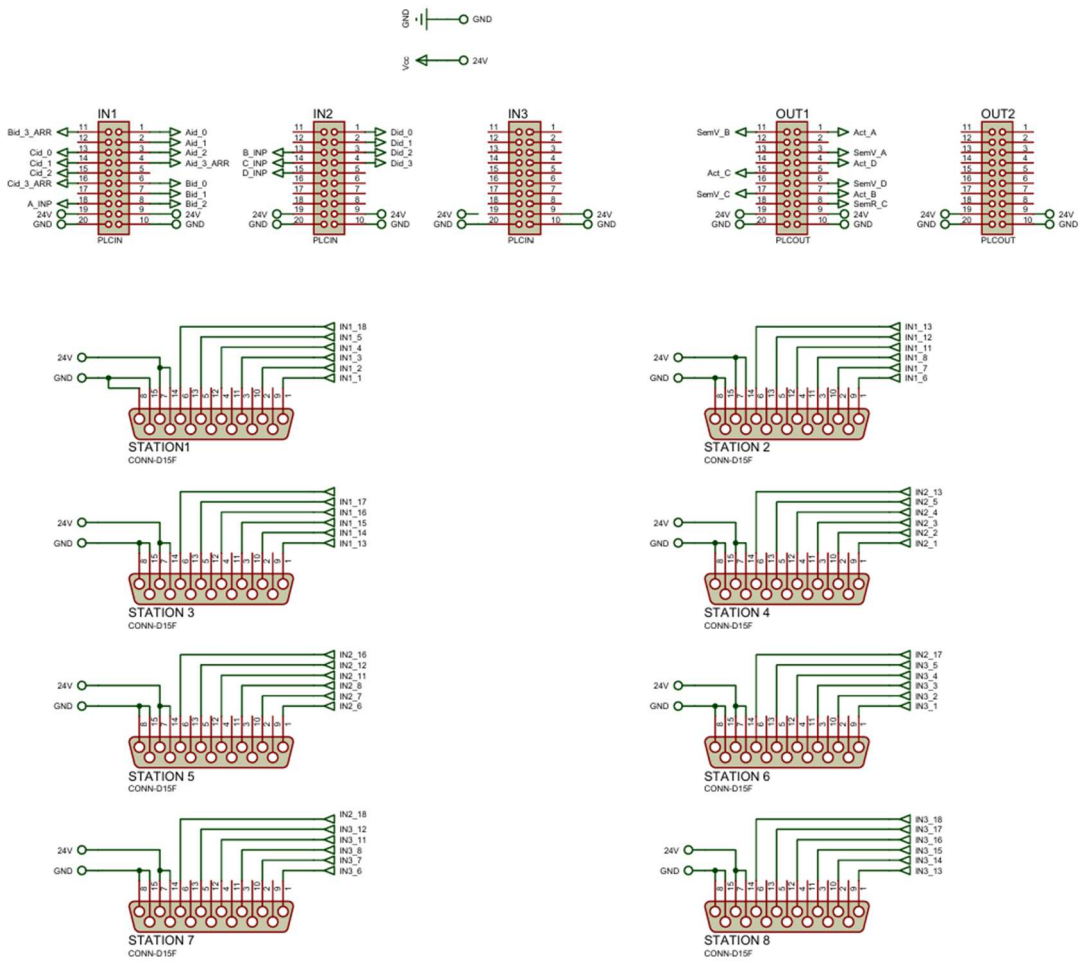
Diagrama eléctrico general

- **Subcircuito: Placa interface PLC**
- **Subcircuitos: Optoacopladores x 4**

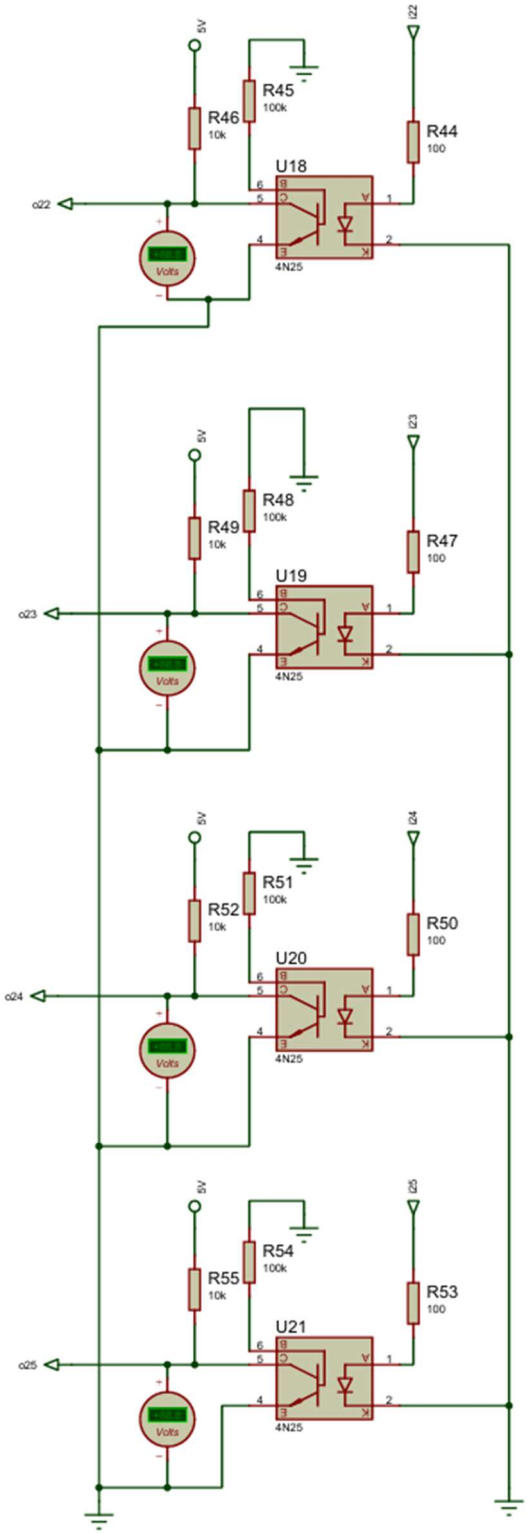
- **Arduino Mega**
- **Subcircuito: Relés**



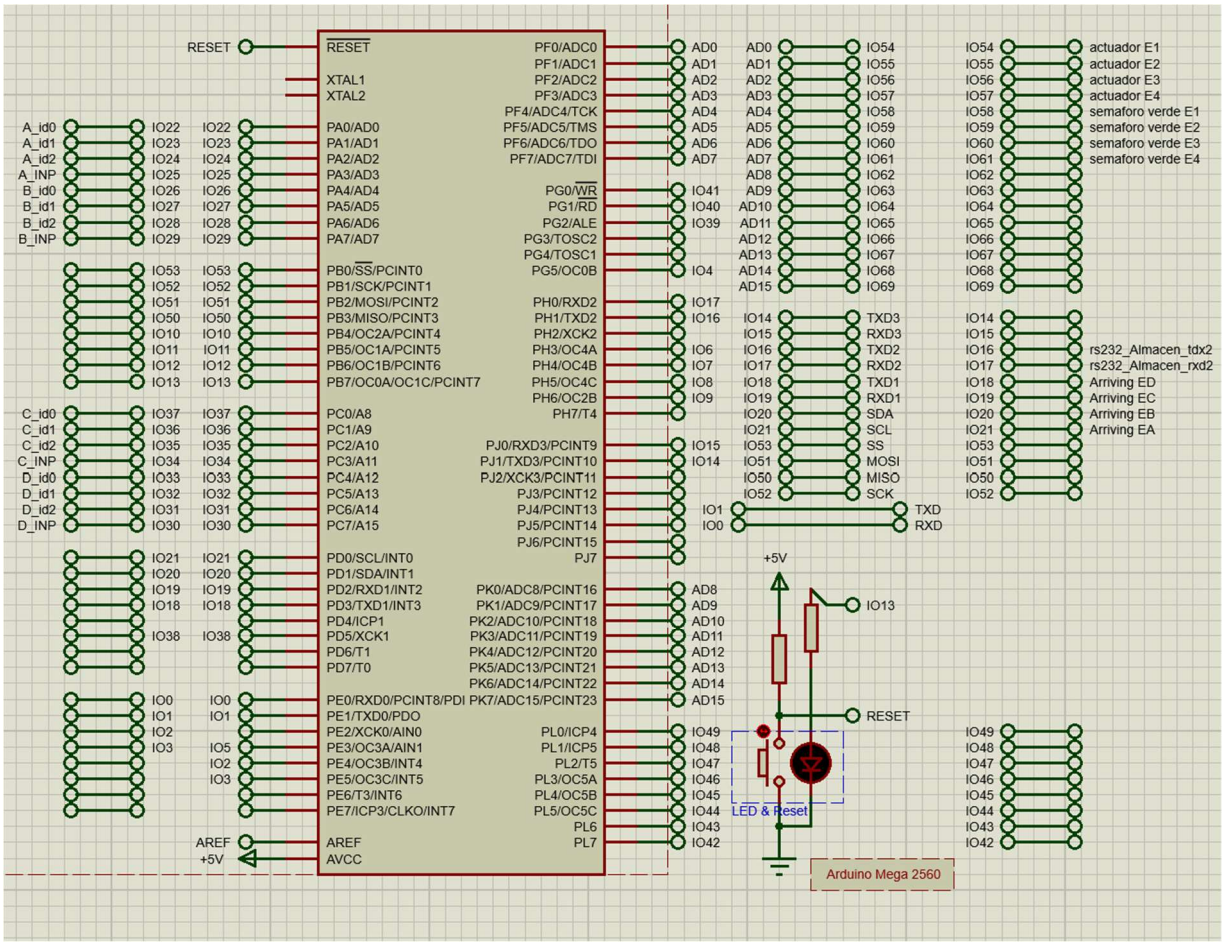
Subcircuito: Placa interfaz PLC



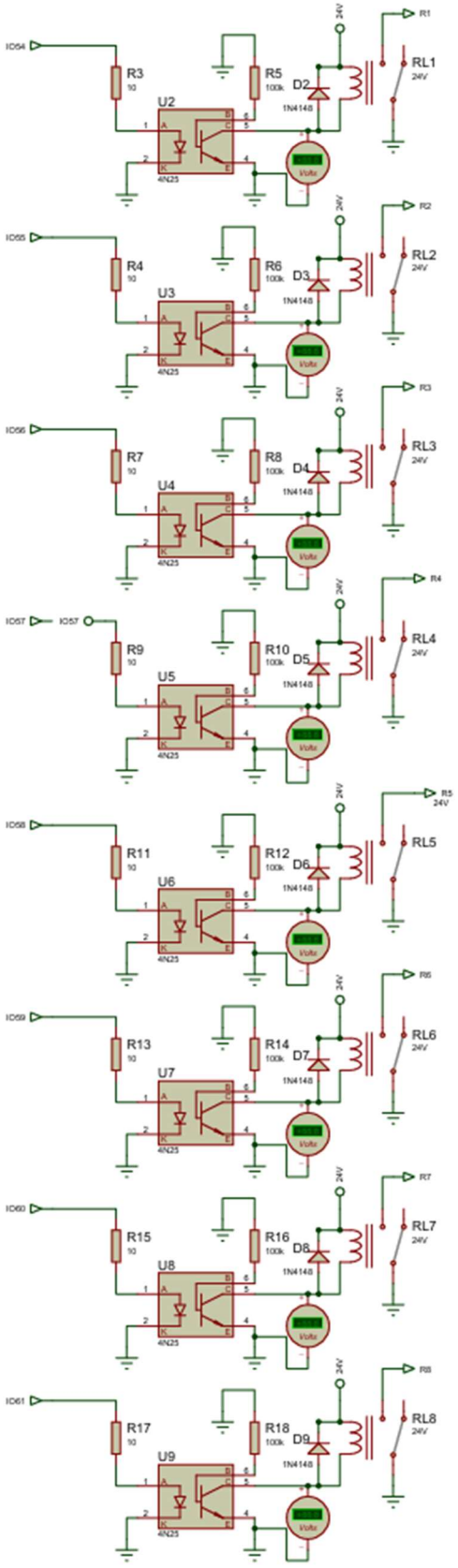
Subcircuito: Optoacopladores x 4


5V
R45
100k
R46
10k
R44
100
U18
o22
i22
4N25
5V
R48
100k
R49
10k
R47
100
U19
o23
i23
4N25
5V
R51
100k
R52
10k
R50
100
U20
o24
i24
4N25
5V
R54
100k
R55
10k
R53
100
U21
o25
i25
4N25
Volts

Arduino Mega



Subcircuito: Relés x 8



Programación

El código del Arduino debe:

a) Leer señales de los sensores magnéticos.
   i) Cuando el primer imán de un pallet es detectado por el sensor ID3 (arriving) se debe accionar el tope electromecánico de la estación correcta.
   ii) Cuando el sensor IN PLACE detecte un imán, se deberá tomar lectura del pallet para identificarlo.
   iii) Se debe transmitir el número de Pallet y esperar la instrucción del Manager para continuar.
   iv) Si el Manager indica que el pallet debe continuar, poner en 0 la salida del relé que energiza el tope (freno del pallet).
   v) Tope hacia arriba = semáforo en rojo, tope desactivado = semáforo en verde.
   vi) Se utilizarán interrupciones para detener el pallet y luego leer el nro. de identificación.

Primer código. Permite leer la identificación de los Pallets que pasan por cualquiera de las 4 estaciones de la cinta transportadora. Imprime el número de estación e identificación del Pallet.

```
#include <LiquidCrystal.h>

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


void setup() {
  lcd.begin(16, 2);

  pinMode(26, INPUT);
  pinMode(33, INPUT);
  pinMode(A4, INPUT);
  pinMode(A12, INPUT);
  pinMode(13, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(8, OUTPUT);

  Serial.begin(9600);

}
void loop()
{

  int i = digitalRead(26);  // estacion a
```

```
  int j = digitalRead(33);  // estacion b
  int k = digitalRead(A4);  // estacion c
  int l = digitalRead(A12);  // estacion d

    if(digitalRead(26)==1)
     {
      i=PINC;
      i=PINC&0b00001111;
      digitalWrite(13, HIGH);
      Serial.println("Estacion 1 ON");
      Serial.print("carro nro "); Serial.println(i,HEX);
     }
     else{
      digitalWrite(13,LOW);
      Serial.println("Estacion 1 OFF");
     }

if(digitalRead(33)==1)
     {
      j=PINC;
      j=PINC&0b00001111;
      digitalWrite(9, HIGH);
      Serial.println("Estacion 2 ON");
      Serial.print("carro nro "); Serial.println(j,HEX);

     }
     else{
      digitalWrite(9,LOW);
      Serial.println("Estacion 2 OFF");
     }

if(digitalRead(A4)==1)
     {
      k=PINF;
      k=PINF&0b00001111;
      digitalWrite(10, HIGH);
      Serial.println("Estacion 3 ON");
      Serial.print("carro nro "); Serial.println(k,HEX);
     }
     else{
      digitalWrite(10,LOW);
      Serial.println("Estacion 3 OFF");
     }

if(digitalRead(A12)==1)
     {
      l=PINK;
      l=PINK&0b00001111;
      digitalWrite(8, HIGH);
```

```
     Serial.println("Estacion 4 ON");
     Serial.print("carro nro "); Serial.println(l,HEX);
    }
    else{
     digitalWrite(8,LOW);
     Serial.println("Estacion 4 OFF");
    }
      //lcd.noDisplay();
      //delay(500);
      lcd.display();
      delay(300);
      lcd.print("a");
      lcd.print(i);
      //if(i==0b00000001)
    //{
     //lcd.print("p1_c1");
    //}
      delay(500);
      lcd.print("b");
      lcd.print(j);
      delay(500);
      lcd.print("c");
      lcd.print(k);
      delay(500);
      lcd.print("d");
      lcd.print(l);
}
```

Código Arduino mejorado para las 4 Estaciónes:

```
#include <avr/interrupt.h>
#include <Arduino.h>

// ================================================================
// 1. DEFINICIONES DE HARDWARE (ASIGNACIÓN DE PINES)
// ================================================================
// --- Estación A ---
#define SENSOR_A_ID0      PA0   // Pin 22
#define SENSOR_A_ID1      PA1   // Pin 23
#define SENSOR_A_ID2      PA2   // Pin 24
#define SENSOR_A_ID3      PD0   // Pin 21 (INT0) Arriving
#define SENSOR_A_INPLACE  PA3   // Pin 25
#define ACTUADOR_A_PIN    PF0   // Asignado al Puerto F

// --- Estación B ---
```

```
#define SENSOR_B_ID0      PA4   // Pin 26
#define SENSOR_B_ID1      PA5   // Pin 27
#define SENSOR_B_ID2      PA6   // Pin 28
#define SENSOR_B_ID3      PD1   // Pin 20 (INT1) Arriving
#define SENSOR_B_INPLACE  PA7   // Pin 29
#define ACTUADOR_B_PIN    PF1   // Asignado al Puerto F

// --- Estación C ---
#define SENSOR_C_ID0      PC0   // Pin 37
#define SENSOR_C_ID1      PC1   // Pin 36
#define SENSOR_C_ID2      PC2   // Pin 35
#define SENSOR_C_ID3      PD2   // Pin 19 (INT2) Arriving
#define SENSOR_C_INPLACE  PC3   // Pin 34
#define ACTUADOR_C_PIN    PF2   // Asignado al Puerto F

// --- Estación D ---
#define SENSOR_D_ID0      PC4   // Pin 33
#define SENSOR_D_ID1      PC5   // Pin 32
#define SENSOR_D_ID2      PC6   // Pin 31
#define SENSOR_D_ID3      PD3   // Pin 18 (INT3) Arriving
#define SENSOR_D_INPLACE  PC7   // Pin 30
#define ACTUADOR_D_PIN    PF3   // Asignado al Puerto F

// ================================================================
// 2. PARÁMETROS DE TIEMPO (AJUSTABLES EN MILISEGUNDOS)
// ================================================================
#define TIEMPO_FRENADO_MS 500  // Tiempo que el actuador permanece activado antes
de esperar la señal "inplace".
#define TIEMPO_LECTURA_MS 20  // Pausa adicional después de leer el ID y antes de
liberar el pallet.

// ================================================================
// 3. ESTRUCTURAS Y VARIABLES GLOBALES
// ================================================================

#define NUM_ESTACIONES 4
const char NOMBRES_ESTACION[NUM_ESTACIONES] = {'A', 'B', 'C', 'D'};

// El desbordamiento del Timer2 con preescalador 1024 ocurre cada ~16 ms.
// Calculamos cuántos desbordamientos se necesitan para alcanzar el tiempo de
frenado.
const int OVERFLOWS_PARA_FRENADO = TIEMPO_FRENADO_MS / 16;

struct EstadoEstacion {
  volatile bool palletDetectado = false;
  volatile bool tiempoDeFrenadoCumplido = false;
  volatile int contadorOverflows = 0;
};
```

```
EstadoEstacion estados[NUM_ESTACIONES];

// Prototipos de funciones auxiliares
void gestionarEstacion(int i);
uint8_t leerID(int estacionIndex);
void imprimirID(int estacionIndex, uint8_t id);
bool leerSensorInplace(int estacionIndex);
void controlarActuador(int estacionIndex, bool activar);
void configurarInterrupcionesExternas();
void configurarTimer2();

// ================================================================
// 4. CONFIGURACIÓN INICIAL (setup)
// ================================================================
void setup() {
  Serial.begin(9600);
  Serial.println("Sistema de estaciones iniciado.");
  Serial.print("Tiempo de frenado configurado a ");
  Serial.print(TIEMPO_FRENADO_MS);
  Serial.println(" ms.");
  Serial.print("Tiempo de lectura configurado a ");
  Serial.print(TIEMPO_LECTURA_MS);
  Serial.println(" ms.");

  // --- Configuración de Pines de Entrada (Sensores) ---
  DDRA &= ~((1 << SENSOR_A_ID0) | (1 << SENSOR_A_ID1) | (1 << SENSOR_A_ID2) | (1
<< SENSOR_A_INPLACE) | (1 << SENSOR_B_ID0) | (1 << SENSOR_B_ID1) | (1 <<
SENSOR_B_ID2) | (1 << SENSOR_B_INPLACE));
  PORTA |= (1 << SENSOR_A_ID0) | (1 << SENSOR_A_ID1) | (1 << SENSOR_A_ID2) | (1
<< SENSOR_A_INPLACE) | (1 << SENSOR_B_ID0) | (1 << SENSOR_B_ID1) | (1 <<
SENSOR_B_ID2) | (1 << SENSOR_B_INPLACE);

  DDRC &= ~((1 << SENSOR_C_ID0) | (1 << SENSOR_C_ID1) | (1 << SENSOR_C_ID2) | (1
<< SENSOR_C_INPLACE) | (1 << SENSOR_D_ID0) | (1 << SENSOR_D_ID1) | (1 <<
SENSOR_D_ID2) | (1 << SENSOR_D_INPLACE));
  PORTC |= (1 << SENSOR_C_ID0) | (1 << SENSOR_C_ID1) | (1 << SENSOR_C_ID2) | (1
<< SENSOR_C_INPLACE) | (1 << SENSOR_D_ID0) | (1 << SENSOR_D_ID1) | (1 <<
SENSOR_D_ID2) | (1 << SENSOR_D_INPLACE);

  DDRD &= ~((1 << SENSOR_A_ID3) | (1 << SENSOR_B_ID3) | (1 << SENSOR_C_ID3) | (1
<< SENSOR_D_ID3));
  PORTD |= (1 << SENSOR_A_ID3) | (1 << SENSOR_B_ID3) | (1 << SENSOR_C_ID3) | (1
<< SENSOR_D_ID3);

  // --- Configuración de Pines de Salida (Actuadores) ---
  DDRF |= (1 << ACTUADOR_A_PIN) | (1 << ACTUADOR_B_PIN) | (1 << ACTUADOR_C_PIN) |
(1 << ACTUADOR_D_PIN);
  PORTF |= (1 << ACTUADOR_A_PIN) | (1 << ACTUADOR_B_PIN) | (1 << ACTUADOR_C_PIN)
| (1 << ACTUADOR_D_PIN);
```

```
  // --- Configuración de Interrupciones ---
  configurarInterrupcionesExternas();
  configurarTimer2();

  sei();
}

// ================================================================
// 5. BUCLE PRINCIPAL (loop)
// ================================================================
void loop() {
  for (int i = 0; i < NUM_ESTACIONES; i++) {
    gestionarEstacion(i);
  }
}

// ================================================================
// 6. LÓGICA PRINCIPAL POR ESTACIÓN (CORREGIDA)
// ================================================================
void gestionarEstacion(int i) {
  if (!estados[i].palletDetectado) {
    return;
  }

  // --- INICIO DE LA CORRECCIÓN PARA EL PALLET 8 ---
  // Si el sensor 'inplace' está activo justo cuando llega el pallet,
  // omitimos el temporizador de frenado para evitar la condición de carrera.
  if (leerSensorInplace(i) && !estados[i].tiempoDeFrenadoCumplido) {
    estados[i].tiempoDeFrenadoCumplido = true;
  }
  // --- FIN DE LA CORRECCIÓN ---

  // PASO 1: Pallet detectado, se activa el freno hasta que se cumpla el tiempo.
  if (estados[i].tiempoDeFrenadoCumplido == false) {
    controlarActuador(i, true);  // Activa el actuador
  }
  // PASO 2: Tiempo cumplido (o saltado), ahora se espera la posición final.
  else {
    if (leerSensorInplace(i)) {  // Si el sensor de posición está activado...

      // --- PROCESO COMPLETADO PARA ESTA ESTACIÓN ---
      uint8_t id = leerID(i);
      imprimirID(i, id);

      delay(TIEMPO_LECTURA_MS);

      controlarActuador(i, false);  // Desactiva el actuador
```

```
      // --- REINICIO DE ESTADO PARA LA PRÓXIMA LECTURA ---
      estados[i].palletDetectado = false;
      estados[i].tiempoDeFrenadoCumplido = false;
      estados[i].contadorOverflows = 0;

      EIFR |= (1 << (INTF0 + i));
      EIMSK |= (1 << (INT0 + i));
    }
  }
}

// ================================================================
// 7. RUTINAS DE SERVICIO DE INTERRUPCIÓN (ISR)
// ================================================================

ISR(INT0_vect) {
  estados[0].palletDetectado = true;
  EIMSK &= ~(1 << INT0);
}
ISR(INT1_vect) {
  estados[1].palletDetectado = true;
  EIMSK &= ~(1 << INT1);
}
ISR(INT2_vect) {
  estados[2].palletDetectado = true;
  EIMSK &= ~(1 << INT2);
}
ISR(INT3_vect) {
  estados[3].palletDetectado = true;
  EIMSK &= ~(1 << INT3);
}

// --- ISR para el Desbordamiento del Timer2 ---
ISR(TIMER2_OVF_vect) {
  for (int i = 0; i < NUM_ESTACIONES; i++) {
    if (estados[i].palletDetectado && !estados[i].tiempoDeFrenadoCumplido) {
      estados[i].contadorOverflows++;
      if (estados[i].contadorOverflows >= OVERFLOWS_PARA_FRENADO) {
        estados[i].tiempoDeFrenadoCumplido = true;
        estados[i].contadorOverflows = 0;
      }
    }
  }
}

// ================================================================
// 8. FUNCIONES AUXILIARES
// ================================================================
```

```
void configurarInterrupcionesExternas() {
  EICRA = (1 << ISC01) | (1 << ISC11);
  EICRB = (1 << ISC21) | (1 << ISC31);
  EIMSK |= (1 << INT0) | (1 << INT1) | (1 << INT2) | (1 << INT3);
}

void configurarTimer2() {
  TCCR2B = (1 << CS22) | (1 << CS21) | (1 << CS20);
  TIMSK2 |= (1 << TOIE2);
}

uint8_t leerID(int estacionIndex) {
  uint8_t pina = PINA;
  uint8_t pinc = PINC;
  uint8_t pind = PIND;

  switch (estacionIndex) {
    case 0:  // Estación A
      return ((!(pina & (1 << SENSOR_A_ID2))) ? 0b0100 : 0) | ((!(pina & (1 <<
SENSOR_A_ID1))) ? 0b0010 : 0) | ((!(pina & (1 << SENSOR_A_ID0))) ? 0b0001 : 0) |
((!(pind & (1 << SENSOR_A_ID3))) ? 0b1000 : 0);
    case 1:  // Estación B
      return ((!(pina & (1 << SENSOR_B_ID2))) ? 0b0100 : 0) | ((!(pina & (1 <<
SENSOR_B_ID1))) ? 0b0010 : 0) | ((!(pina & (1 << SENSOR_B_ID0))) ? 0b0001 : 0) |
((!(pind & (1 << SENSOR_B_ID3))) ? 0b1000 : 0);
    case 2:  // Estación C
      return ((!(pinc & (1 << SENSOR_C_ID2))) ? 0b0100 : 0) | ((!(pinc & (1 <<
SENSOR_C_ID1))) ? 0b0010 : 0) | ((!(pinc & (1 << SENSOR_C_ID0))) ? 0b0001 : 0) |
((!(pind & (1 << SENSOR_C_ID3))) ? 0b1000 : 0);
    case 3:  // Estación D
      return ((!(pinc & (1 << SENSOR_D_ID2))) ? 0b0100 : 0) | ((!(pinc & (1 <<
SENSOR_D_ID1))) ? 0b0010 : 0) | ((!(pinc & (1 << SENSOR_D_ID0))) ? 0b0001 : 0) |
((!(pind & (1 << SENSOR_D_ID3))) ? 0b1000 : 0);
  }
  return 0;
}

bool leerSensorInplace(int estacionIndex) {
  switch (estacionIndex) {
    case 0: return (PINA & (1 << SENSOR_A_INPLACE)) == 0;
    case 1: return (PINA & (1 << SENSOR_B_INPLACE)) == 0;
    case 2: return (PINC & (1 << SENSOR_C_INPLACE)) == 0;
    case 3: return (PINC & (1 << SENSOR_D_INPLACE)) == 0;
  }
  return false;
}

void controlarActuador(int estacionIndex, bool activar) {
  uint8_t pinActuador;
```

```
  switch (estacionIndex) {
    case 0: pinActuador = ACTUADOR_A_PIN; break;
    case 1: pinActuador = ACTUADOR_B_PIN; break;
    case 2: pinActuador = ACTUADOR_C_PIN; break;
    case 3: pinActuador = ACTUADOR_D_PIN; break;
    default: return;
  }

  if (activar) {
    PORTF &= ~(1 << pinActuador);
  } else {
    PORTF |= (1 << pinActuador);
  }
}

void imprimirID(int estacionIndex, uint8_t id) {
  Serial.print("Estacion ");
  Serial.print(NOMBRES_ESTACION[estacionIndex]);
  Serial.print(". ID leido: ");
  for (int8_t b = 3; b >= 0; b--) {
    Serial.print((id >> b) & 1);
  }
  Serial.print(" (Decimal: ");
  Serial.print(id);
  Serial.println(")");
}
```

Conclusiones

- La implementación del Arduino Mega como reemplazo del PLC en el sistema de control de la cinta transportadora demostró ser una solución eficiente y versátil. A través del diseño de esquemas eléctricos, la incorporación de componentes óptimos como optoacopladores y sensores magnéticos, y el desarrollo de un código funcional para la comunicación y control, se logró cumplir con los objetivos del proyecto: modernizar el sistema y optimizar su funcionamiento.
- Este enfoque no solo permitió la automatización precisa de las cuatro estaciones de trabajo, sino que también facilitó una interacción efectiva entre el Arduino y la PC mediante comunicación USB - RS232. Las pruebas realizadas garantizaron la fiabilidad del sistema, ajustando tiempos y lógica para evitar colisiones entre pallets y mejorar la productividad.

ANEXO 1

Placa Optoacopladores – diseño y calculo

La corriente típica del LED del PC817A es de 10 mA, pero puede funcionar correctamente con corrientes más bajas, como 5 mA. de esta forma optimizamos el consumo y temperaturas elevadas en la resistencia de entrada.

Datos conocidos:

Tensión de la fuente: Vfuente= Vin=24V

Tensión directa del led del optoacoplador: Vled=1.2V

Corriente deseada a través del led: iled=5mA=0.005ª

Paso 1: Calculo de la Resistencia de Entrada

Corriente de 5 mA para disminuir la disipación de potencia y el calor generado:

$$If = 5mA = 0.005\ A$$

La nueva resistencia se calcula como:

$$R1 = \frac{Vin - Vled}{iled}$$

Sustituimos los valores:

$$R1 = \frac{24V - 1.2V}{0.005A}$$

$$R1 = 4560\Omega$$

Utilizamos la resistencia estándar de 4.7k Ω

Paso 2: Verificar la Potencia Disipada

La potencia disipada por la resistencia se calcula como:

$$Pr1 = iled^2\ x\ R1$$

Donde:

iled=0.005A

R1=4700 Ω

Calculamos:

$$Pr1 = (0.005A)^2\ x\ 4700\Omega$$

$$Pr1 = 0.1175W$$

Una resistencia de 1/4 de vatio (0.25 W) debería ser suficiente, ya que la potencia disipada es mucho menor que la capacidad de la resistencia.

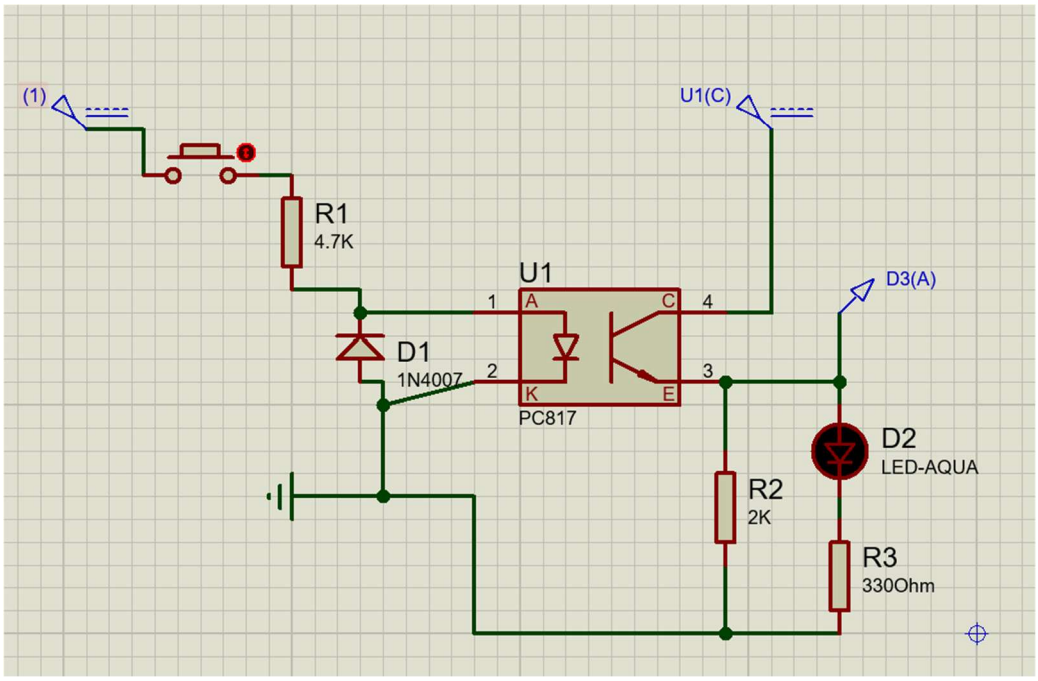
1. Diodo de protección en la entrada:

Colocamos un diodo en polaridad inversa en la entrada para proteger el optoacoplador en caso de conexión inversa de la fuente de 24V.

2. LED en la salida:

Colocamos un LED con una resistencia en la salida del optoacoplador para visualizar el estado de la señal de 5V.

Esquema del Circuito:



Lista de Componentes:

- R1: 4.7kΩ (resistor de entrada)
- R2: 2kΩ (resistor de pull-up)
- R3: 330Ω (resistor limitador de corriente para el LED de salida)
- D1: Diodo de protección en polaridad inversa (1N4007 o similar)
- LED (OUT): LED para indicar la salida de 5V

Detalles del Circuito:

1. Resistor de entrada (R1):
    - Valor: 4.7kΩ
    - Función: Limitar la corriente a través del LED del optoacoplador.

2. Diodo de protección (D1):
    - Tipo: 1N4007 o similar.
    - Conexión: En polaridad inversa entre la entrada de 24V y GND.
    - Función: Proteger el optoacoplador en caso de conexión inversa de la fuente.
3. Optoacoplador (PC817A):
    - LED interno: Conectado en serie con R1.
    - Transistor interno: Colector a GND a través de D1, emisor a +5V a través de R2.
4. Resistor de salida (R2):
    - Valor: 2kΩ
    - Función: Actuar como un resistor de pull-up para la salida del optoacoplador.
5. Resistor limitador de corriente para el LED de salida (R3):
    - Valor: 330Ω (calculado para una corriente de aproximadamente 15 mA para el LED).
    - Función: Limitar la corriente a través del LED de salida.
6. LED de salida (LED OUT):
    - Función: Indicar visualmente la presencia de la señal de 5V.