PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: DISEÑO Y CONSTRUCCIÓN DE UN MÓDULO DIDÁCTICO DE UN SISTEMA DE AUTOMATIZACIÓN DE LLENADO Y ENVASADO DE SÓLIDOS, UTILIZANDO SENSORES FOTOELÉCTRICOS, ULTRASÓNICOS, CAPACITIVOS, ENCODERS, GALGAS EXTENSIOMÉTRICAS Y BRAZOS ROBÓTICOS, PARA EL LABORATORIO DE REDES INDUSTRIALES Y CONTROL DE PROCESOS DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE EXTENSIÓN LATACUNGA

Autores
- Washington Herrera
- Sebastián Panchi

Director
- Ing. Edwin Pruna

Codirector
- Ing. Galo Ávila

Agenda

- Objetivo general
- Objetivos específicos
- Resumen
- Automatización de un sistema
- Mecanismos industriales
- Instrumentación industrial
- Software libre
- Diseño y construcción del módulo didáctico
- Automatización del módulo didáctico
- Pruebas y resultados
- Conclusiones y recomendaciones

Objetivo General:

- **DISEÑAR Y CONSTRUIR UN MÓDULO DIDÁCTICO DE UN SISTEMA DE AUTOMATIZACIÓN DE LLENADO Y ENVASADO DE SÓLIDOS, UTILIZANDO SENSORES FOTOELÉCTRICOS, ULTRASÓNICOS, CAPACITIVOS, ENCODERS, GALGAS EXTENSIOMÉTRICAS Y BRAZOS ROBÓTICOS, PARA EL LABORATORIO DE REDES INDUSTRIALES Y CONTROL DE PROCESOS DE LA UNIVERSIDAD DE LAS FUERZAS ARMADAS ESPE EXTENSIÓN LATACUNGA**

Objetivos específicos

- Recopilar información sobre la Automatización de Procesos.
- Analizar el funcionamiento de tarjetas de adquisición de datos para procesar las señales de los distintos sensores.
- Recopilar información sobre sensores fotoeléctricos, ultrasónicos, capacitivos y encoders a nivel industrial.
- Investigar los tipos de controles utilizados en la industria alimenticia .
- Implementar mecanismos industriales apropiados al proceso de llenado y envasado de sólidos.
- Desarrollar un interfaz máquina-humano (HMI) mediante el software Monitoriza for Arduino de Acimut SCADA para el monitoreo del proceso de llenado y envasado de sólidos, que sea amigable con el usuario.
- Realizar pruebas experimentales sobre el módulo didáctico para obtener un correcto funcionamiento.
- Analizar resultados del comportamiento de las etapas que forman parte del módulo didáctico.

Resumen

El módulo didáctico de un sistema de llenado y envasado de sólidos se encuentra conformado principalmente por cinco etapas controladas por tarjetas Arduino. La primera etapa del módulo didáctico dispensa recipientes plásticos y los distribuye en una mesa giratoria dividida en 8 partes iguales. La segunda etapa del módulo didáctico se encarga del dispensado sobre el recipiente plástico del material sólido contenido en la tolva. La tercera etapa del módulo didáctico realiza el envasado de los recipientes anteriormente llenados, denominada también como etapa de envasado. La cuarta etapa se encarga de organizar los recipientes envasados con la ayuda del movimiento de un brazo robótico en cajas con 4 compartimentos movilizadas a través de una banda transportadora. Y por último la quinta etapa o etapa de inspección se encarga de verificar y clasificar el producto final en la respectiva ruta final de transporte, donde el operador tiene que retirar el producto final para ser enviado a su destino o rechazar los productos que no hayan pasado dicha inspección (rechazo).

Las etapas anteriormente mencionadas son monitoreadas mediante un HMI (Human Machine Interface) desarrollado en un software libre (Monitoriza for Arduino-Scada Acimut), para observar y verificar el correcto funcionamiento de cada una de las etapas del módulo didáctico.

Automatización de un sistema

Se lo define como un sistema capaz de ejecutar acciones previamente establecidas en un espacio y tiempo determinado sin la necesidad de la intervención humana frente a ambientes agresivos y hostiles, mejorando la cadencia y control de la producción.

La automatización se hace posible mediante los Sistemas de Control, mejorando el rendimiento en los procesos repetitivos (clasificación de objetos, semáforos, apertura y cierre de puertas, etc.), realizando tareas que implican desgaste físico importante en el ser humano y controlando procesos difícilmente controlables de forma manual. Un sistema automatizado consta de dos partes principales:

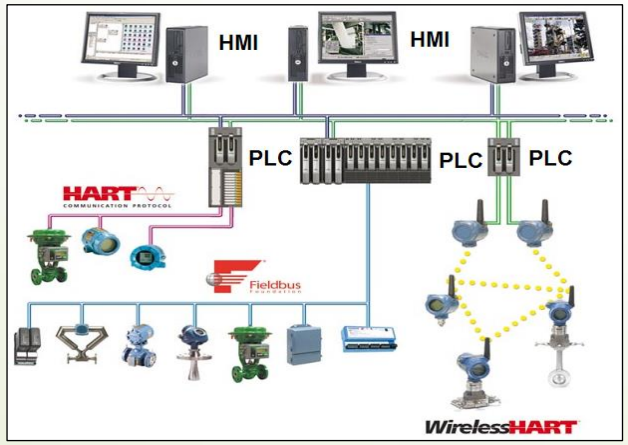
- Parte de Mando
- Parte Operativa

Parte operativa

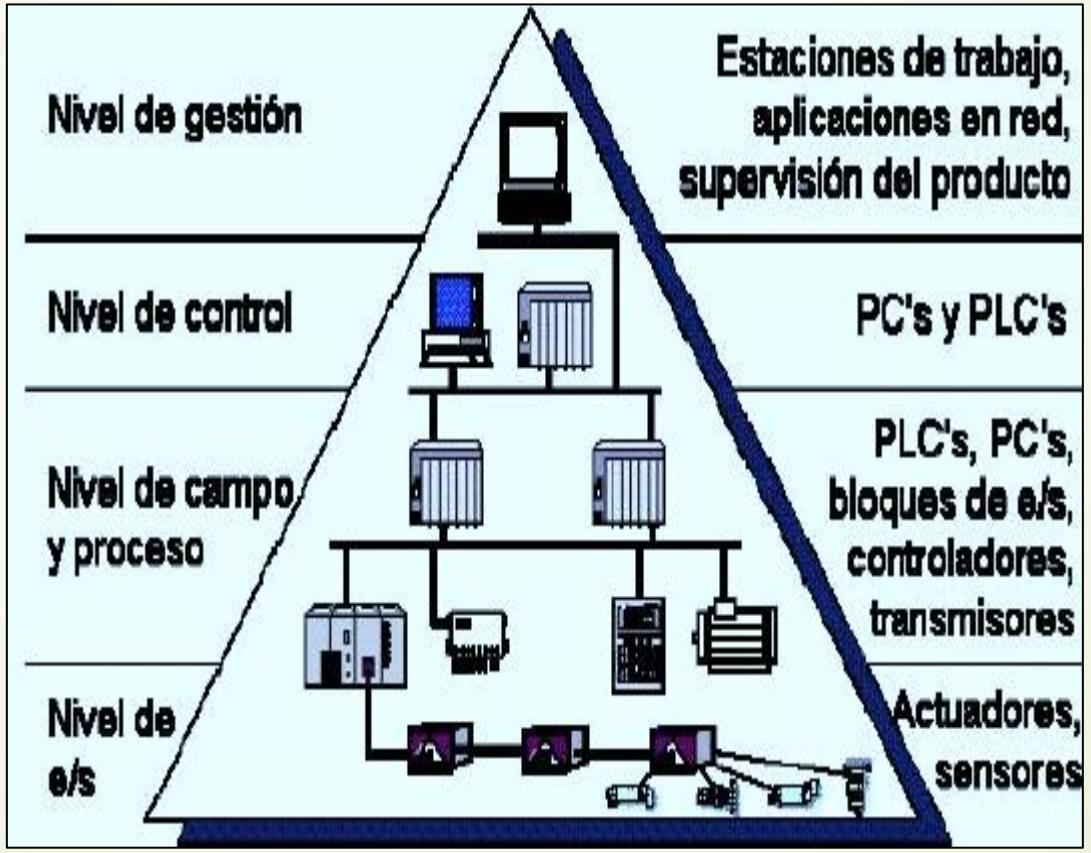
- La parte operativa es la parte que actúa directamente sobre la máquina. Son los elementos que hacen que la máquina se mueva y realice la operación deseada. Los elementos que forman la parte operativa son los accionadores de las máquinas como motores, cilindros de accionamiento, etc., y los captadores como fotodiodos, finales de carrera, entre otros.

Parte de mando

- En su mayoría suele ser un autómata programable (tecnología programada), aunque hasta hace poco se utilizaban relés electromagnéticos, tarjetas electrónicas o módulos lógicos (tecnología cableada). En un sistema de fabricación automatizado el autómata programable está en el centro del sistema, siendo capaz de comunicarse con todos los constituyentes del sistema realizando operaciones específicas de forma secuencial.



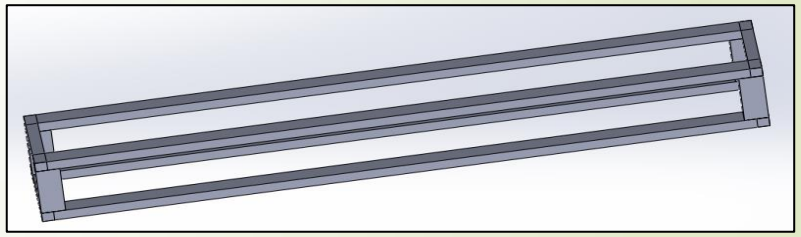
Pirámide de automatización



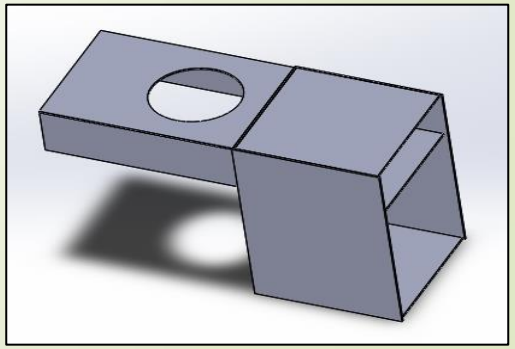
Mecanismos industriales

Los mecanismos industriales que conforman el módulo didáctico de un sistema de automatización de llenado y envasado de sólidos son los siguientes:

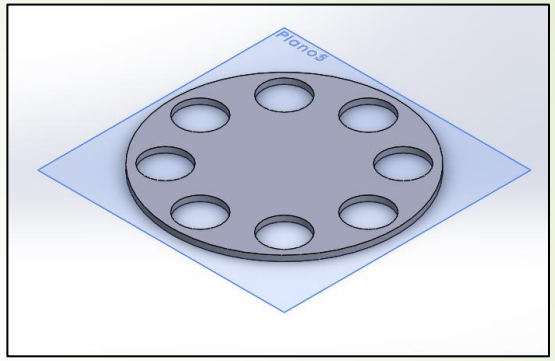
- Banda transportadora



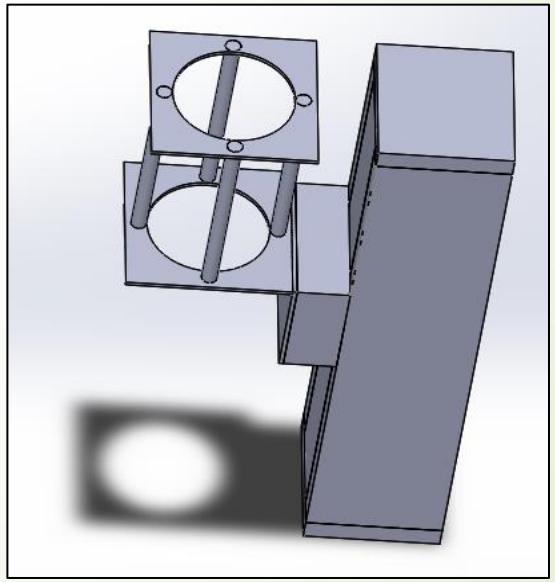
- Estructura base del dispensador de recipientes



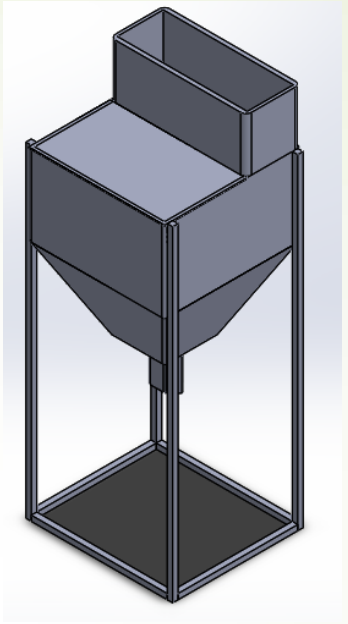
- Mesa giratoria



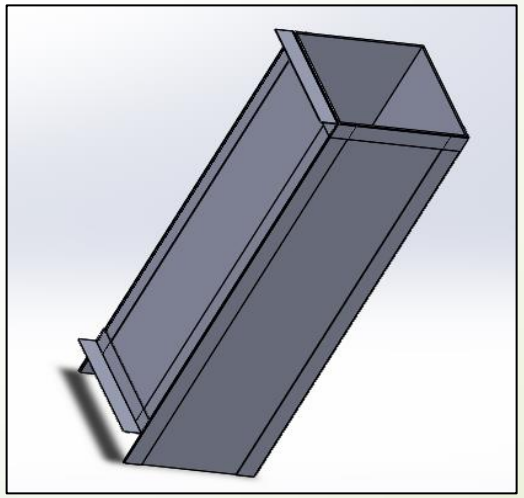
- Máquina de envasado



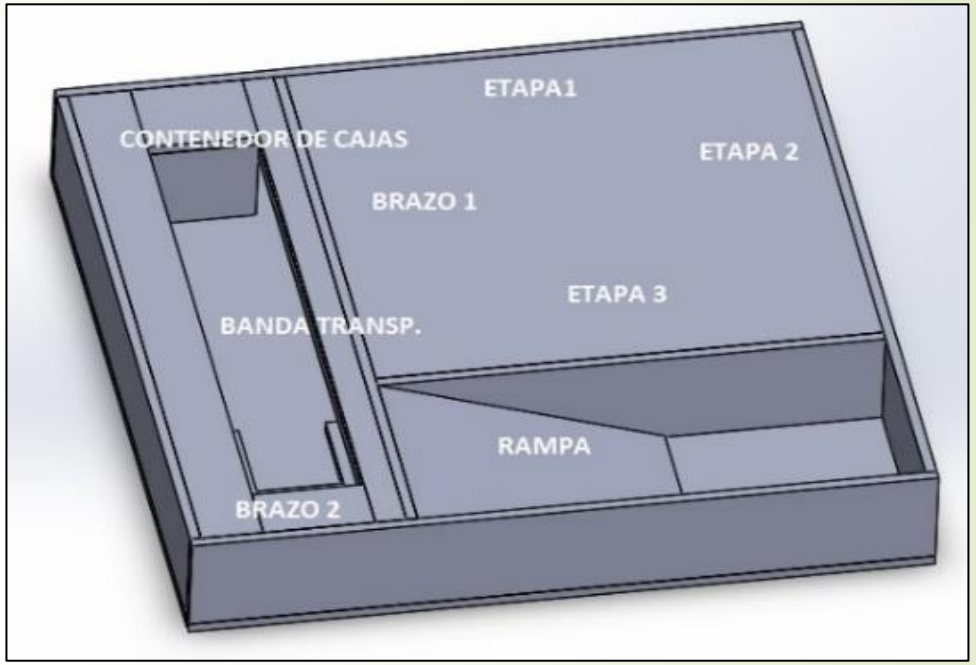
- Tolva



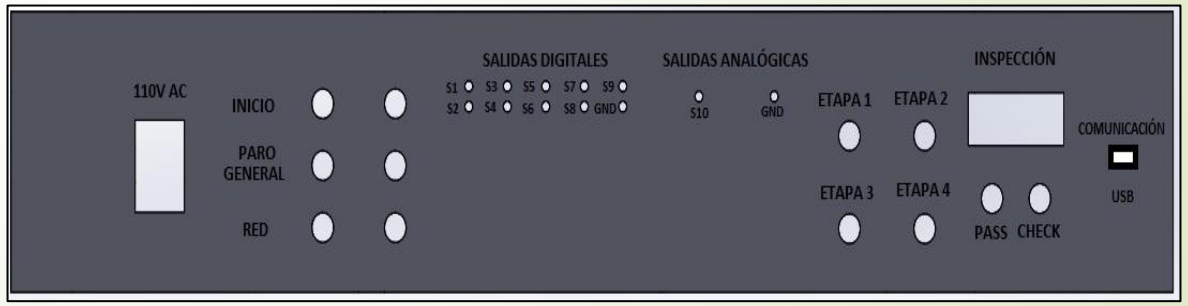
- Contenedor de cajas



- Estructura base del módulo didáctico



- Panel de control



Instrumentación industrial

Dispositivos de medida o sensores que conforman el módulo didáctico de un sistema de llenado y envasado de sólidos.

- **Sensor fotoeléctrico:** Es un dispositivo electrónico que responde al cambio de luz. Estos sensores requieren de un componente emisor que genera la luz, y un componente receptor que percibe la luz generada por el emisor.

Sensores fotoeléctricos utilizados en el módulo didáctico:

- MINI-BEAM SME312D

Especificaciones

Sensing Mode (General)	**Proximity**
Sensing Mode (Specific)	Diffuse
Sensing Beam	Infrared LED
Max Sensing Range [mm]	380
Supply Voltage	10-30 V dc
Output Type	Bipolar (NPN & PNP)
Operation	Light/Dark Operate
Output response time [ms]	0.5
Delay at Power-up [ms]	1000
Repeatability (µs)	100
Connection	2 m Cable
Number of Pins	5
IP Rating	IP67
NEMA Rating	NEMA 1, 2, 3, 3S, 4, 4X, 6, 12, 13
Min Op. Temperature [°C]	-20
Max Op. Temperature [°C]	70
Max Op. Relative Humidity [%]	90% @ 50°C
Basic Housing Material	Thermoplastic
Housing Style	Rectangular with M18 Barrel
Lens Material	Acrylic
Adjustments	Teach Button
Barrel Diameter [mm]	18

- MINI-BEAM SMU315D

Especificaciones

Sensing Mode (General)	Proximity
Sensing Mode (Specific)	Diffuse
Sensing Beam	Infrared LED
Max Sensing Range [mm]	380
Supply Voltage	24-240 V ac/dc
Output Type	SPDT e/m Relay
Operation	Light/Dark Operate
Output response time [ms]	20
Connection	2 m Cable
Number of Pins	5
IP Rating	IP67
NEMA Rating	NEMA 1, 2, 3, 3S, 4, 4X, 6, 12, 13
Min Op. Temperature [°C]	-20
Max Op. Temperature [°C]	55
Max Op. Relative Humidity [%]	90% @ 50°C
Basic Housing Material	Thermoplastic
Housing Style	Rectangular with M18 Barrel
Adjustments	Potentiometer
Barrel Diameter [mm]	18

QS18VP6R (RX) Y QS18SE (TX)

Especificaciones

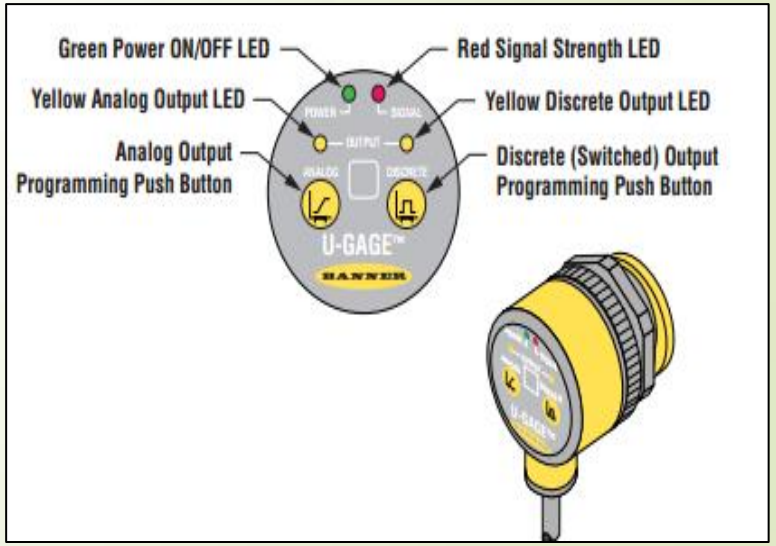
Sensing Mode (General)	Opposed
Sensing Beam	Infrared LED
Opposed Mode Emitter/Receiver	Receiver
Max Sensing Range [m]	20
Supply Voltage	10-30 V dc
Output Type	PNP
Operation	Light/Dark Operate
Output response time [ms]	0.75 On/0.375 Off
Delay at Power-up [ms]	100
Repeatability (µs)	100
Connection	2 m Cable
Number of Pins	4
IP Rating	IP67
NEMA Rating	NEMA 6
Min Op. Temperature [°C]	-20
Max Op. Temperature [°C]	70
Max Op. Relative Humidity [%]	95% @ 50°C
Basic Housing Material	Thermoplastic
Housing Style	Rectangular with M18 Barrel
Lens Material	Acrylic
Barrel Diameter [mm]	18

Sensor ultrasónico: Su función es la de medición de la distancia de objetos mediante la detección de ecos de ultrasonidos. Las ondas ultrasónicas tienen la capacidad de que cuando viajan por un medio cualquiera son reflejadas si encuentran en su camino una discontinuidad o algún elemento extraño. El tiempo de espera entre el envío de la onda ultrasónica hasta su recepción se denomina tiempo de eco, y es utilizado para determinar la distancia al objeto.

El sensor ultrasónico utilizado en el módulo didáctico es el U-GAGE T30UIPAQ.

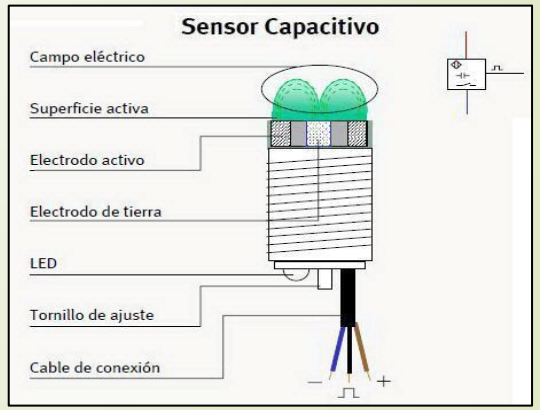
Excelente para aplicaciones de medición, tales como la detección de niveles de líquido de un tanque o materiales sólidos, por ejemplo, la determinación de alturas de caja con fines de clasificación.

Modelo	Rango y frecuencia	Cable	Voltaje de alimentación	Salida discreta	Salida Analogica	Tiempo de respuesta
T30UIPAQ	150mm a 1m 228 kHz	2 m 5 pin Euro DQ	15 a 24V dc	PNP (	4 a 20 mA	48ms



- **Sensores capacitivos:** Están diseñados para detectar materiales aislantes tales como el plástico, el papel, la madera, entre otros, no obstante también cuentan con la capacidad de detectar metales.

En un principio éstos constan de una sonda que se encuentra situada en la cara posterior en donde se encuentra colocada una placa condensadora, y al aplicar una corriente al sensor por más mínima que sea, se produce una especie de campo electroestático cuya reacción se produce frente a los cambios de la capacitancia provocados por la presencia de un objeto cualquiera. En el caso de que el objeto se encuentre fuera del campo electroestático entonces el oscilador de los sensores capacitivos se encontrará inactivo pero a medida que el objeto se va acercando al sensor, este se activa.

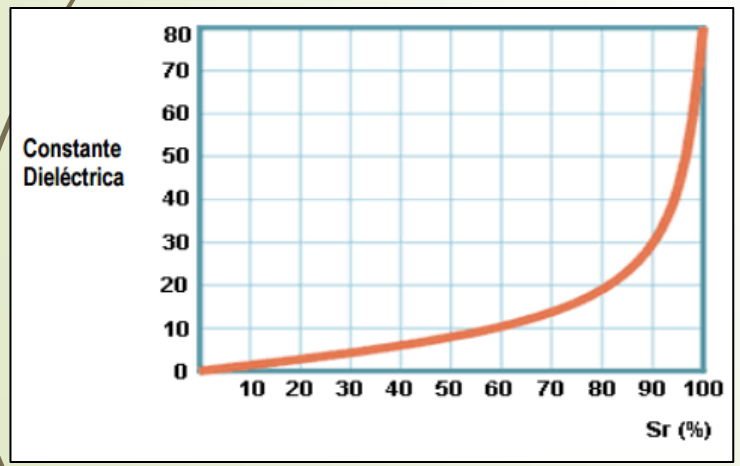


Sensor capacitivo AMICO LJC30A3-H-Z/AY

El sensor capacitivo utilizado detecta objetos ubicados a 25 mm como máximo.

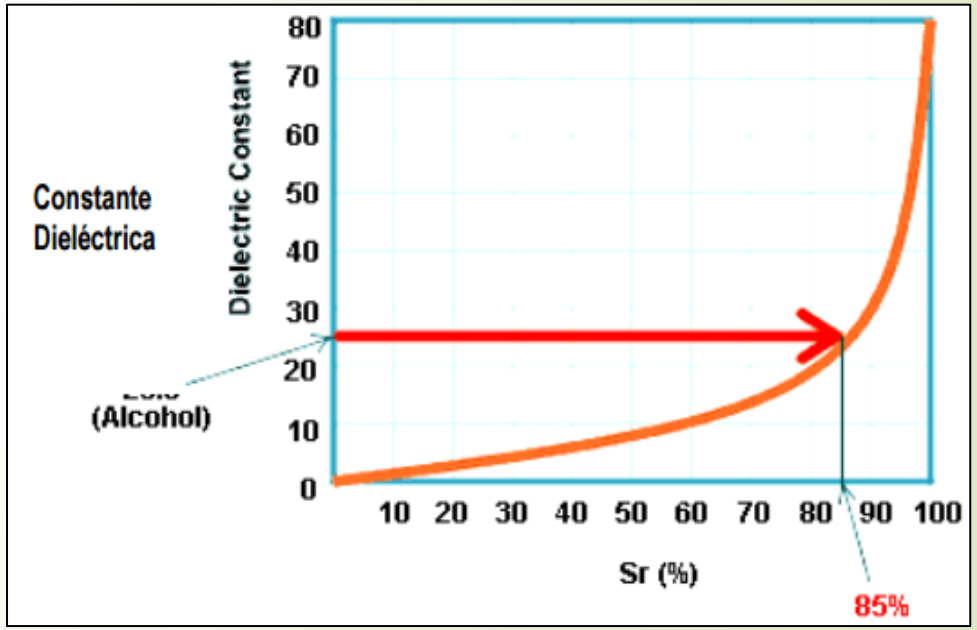
Los sensores capacitivos dependen de la **constante dieléctrica** del objetivo. Mientras más grande sea la constante dieléctrica de un material es más fácil detectar.

La grafica siguiente muestra la relación de las constantes dieléctricas de un objetivo y habilidad del sensor de detectar el material basado en la distancia nominal de sensado (Sr).



Constantes dieléctricas de algunos materiales

Material	k	Material	k
Acetona	19.5	Plexiglás	3.2 - 3.5
Resina Acrílica	2.7-4.5	Petróleo	2.0 - 2.2
Aire	1.000264	Resina de Fenol	4 - 12
Alcohol	25.8	Poliacetato	3.6 - 3.7
Amoníaco	15-25	Poliamidas	5
Anilina	6.9	Resina de Poliéster	2.8 - 8.1
Soluciones Acuosas	50-80	Polietileno	2.3
Baquelita	3.6	Polipropileno	2.0 - 2.3
Bencina	2.3	Poliestireno	3
Dióxido de carbono	1.000985	Porcelana	4.4 - 7
Celuloide	3	Leche en polvo	3.5 – 4
Cemento en polvo	4	Papel de diario	2.5

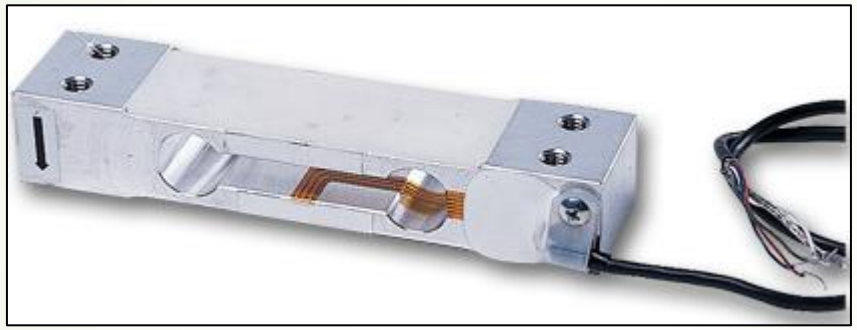


- **Sensor láser QS30LDL:** Los sensores diodo láser combinan las ventajas de la alineación de un haz de detección visible con el mayor alcance de detección de un láser. Funcionan con cc o ca/cc (voltaje universal).

Especificaciones

Sensing Mode (General)	Proximity
Sensing Mode (Specific)	Diffuse
Sensing Beam Wavelength	658
Laser Classification	Class 2
Max Sensing Range [mm]	800
Supply Voltage	10-30 V dc
Output Type	Bipolar (NPN & PNP)
Operation	Light/Dark Operate
Output response time [ms]	0.5
Delay at Power-up [ms]	1000
Repeatability (µs)	70
Connection	2 m Cable
Number of Pins	5
IP Rating	IP67
NEMA Rating	NEMA 6
Min Op. Temperature [°C]	-10
Max Op. Temperature [°C]	50
Max Op. Relative Humidity [%]	95% @ 50°C
Basic Housing Material	Thermoplastic
Housing Style	Rectangular with M30 Barrel
Lens Material	Acrylic
Adjustments	Teach Button
Barrel Diameter [mm]	30

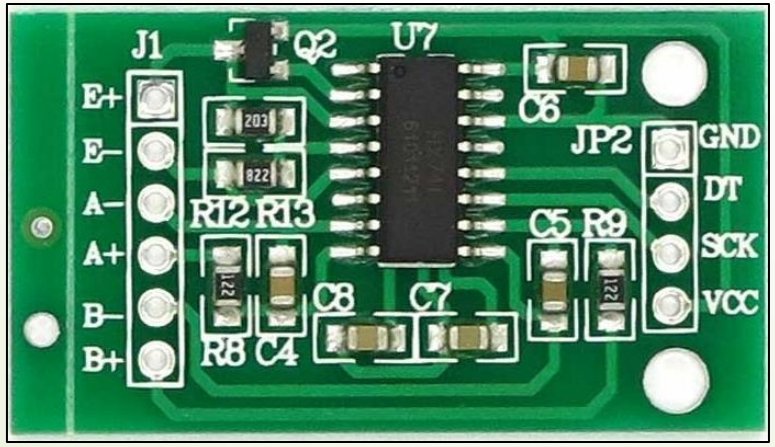
Galgas extensiométricas: La galga extensiométrica es una herramienta importante en la técnica aplicada de medición eléctrica de magnitudes mecánicas.



Para acondicionar la señal de las galgas utilizadas en el módulo didáctico se utilizo un módulo conversor análogo/digital.

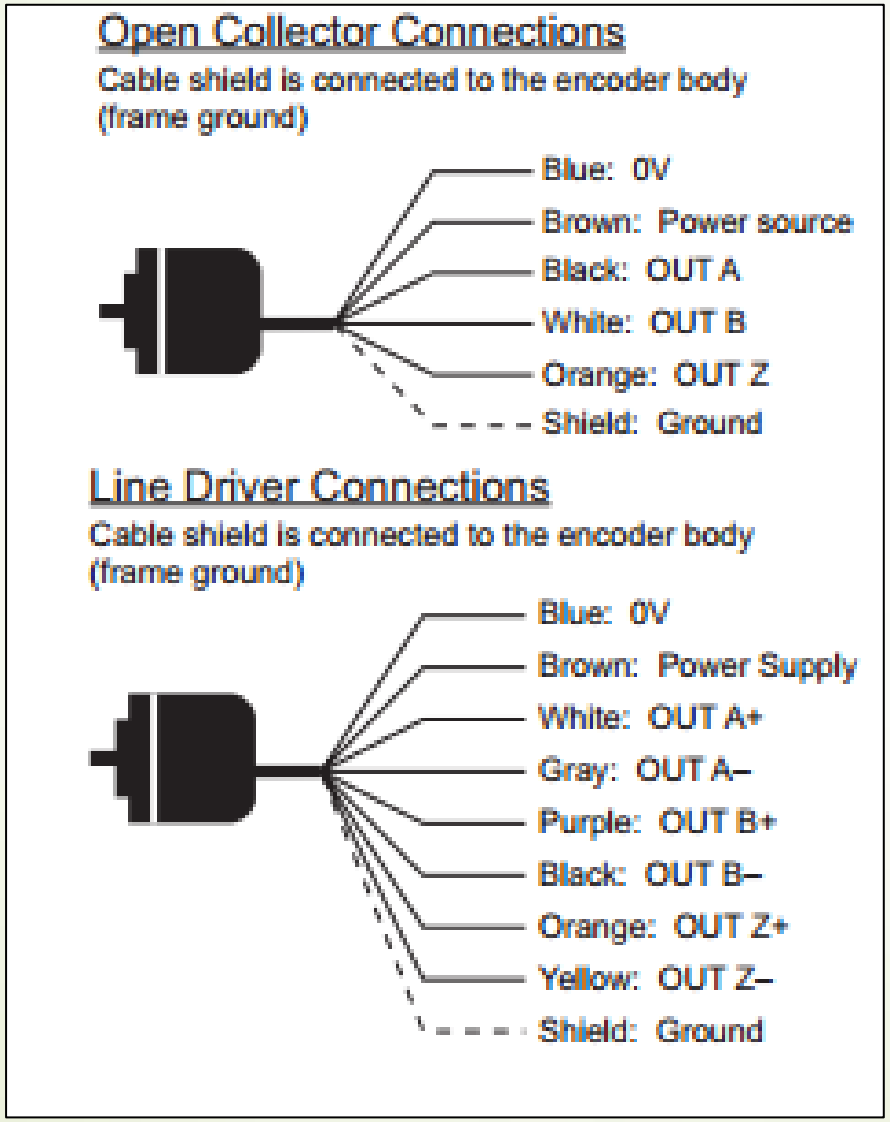
HX711

- BasadO en la tecnología patentada por Avia Semiconductor's, el módulo HX711 es un conversor ADC (Analog Digital Converter por sus siglas en inglés) de 24 bits, diseñado para escalas de peso y aplicaciones de control industrial para interconectarse directamente con un puente weathstone.



Encoder: El encoder incremental TRD - S360BD es un transductor rotativo que provee un número específico de pulsos equitativamente espaciados por revolución (PPR) o por pulgada o milímetro de movimiento lineal. Se utiliza un solo canal de salida para aplicaciones donde el sentido de la dirección de movimiento no es importante (unidireccional).

Conexiones


Open Collector Connections
Cable shield is connected to the encoder body
(frame ground)
Blue: 0V
Brown: Power source
Black: OUT A
White: OUT B
Orange: OUT Z
Shield: Ground
Line Driver Connections
Cable shield is connected to the encoder body
(frame ground)
Blue: 0V
Brown: Power Supply
White: OUT A+
Gray: OUT A–
Purple: OUT B+
Black: OUT B–
Orange: OUT Z+
Yellow: OUT Z–
Shield: Ground

Software libre

Software libre es el software que respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el software. Es decir, el software libre es una cuestión de libertad, no de precio. Para entender el concepto, piense en libre como en libre expresión, no como en barra libre. En inglés a veces decimos libre software, en lugar de free software, para mostrar que no queremos decir que es gratuito.

Un programa es software libre si los usuarios tienen las cuatro libertades esenciales:

- La libertad de ejecutar el programa como se desea, con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para ayudar a su prójimo (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (libertad 3). Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

Plataforma Ard

¿QUÉ ES ARDUINO?

- Arduino es una plataforma electrónica de código abierto basado en hardware y software fácil de usar. Está dirigido a cualquier persona que hace proyectos interactivos.

TARJETA ARDUINO

- Arduino detecta el medio ambiente mediante la recepción de las aportaciones de muchos sensores, y afecta a su entorno por las luces de control, motores y otros actuadores.

SOFTWARE ARDUINO

- Usted puede decirle a su Arduino qué hacer escribiendo código en el lenguaje de programación de Arduino y utilizando el entorno de desarrollo Arduino.

Diseño y construcción del módulo didáctico

Etapa 1

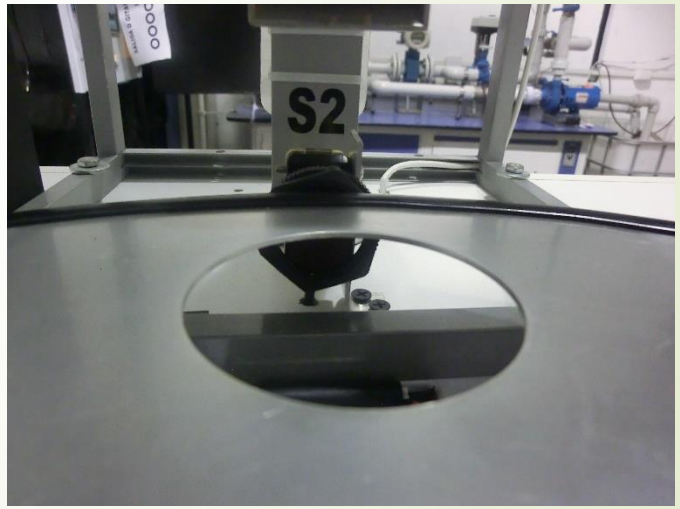
Elementos:

- Pistón Eléctrico
- Motor de alto torque
- Tarjeta Arduino MEGA 2560 R3
- Driver L298N
- Tarjeta de control motores DC
- Sensor láser QS3OLDL
- Sensor capacitivo LJC30A3-H-Z/AY

Etapa 2

Elementos:

- Motor vibrador
- Sensor ultrasónico U-GAGE T30UIPAQ.
- Tarjeta Arduino UNO R3
- Driver L298N
- Sensor láser QS3OLDL
- Galga extensiométrica
- Módulo conversor análogo/digital HX711



Etapa 3

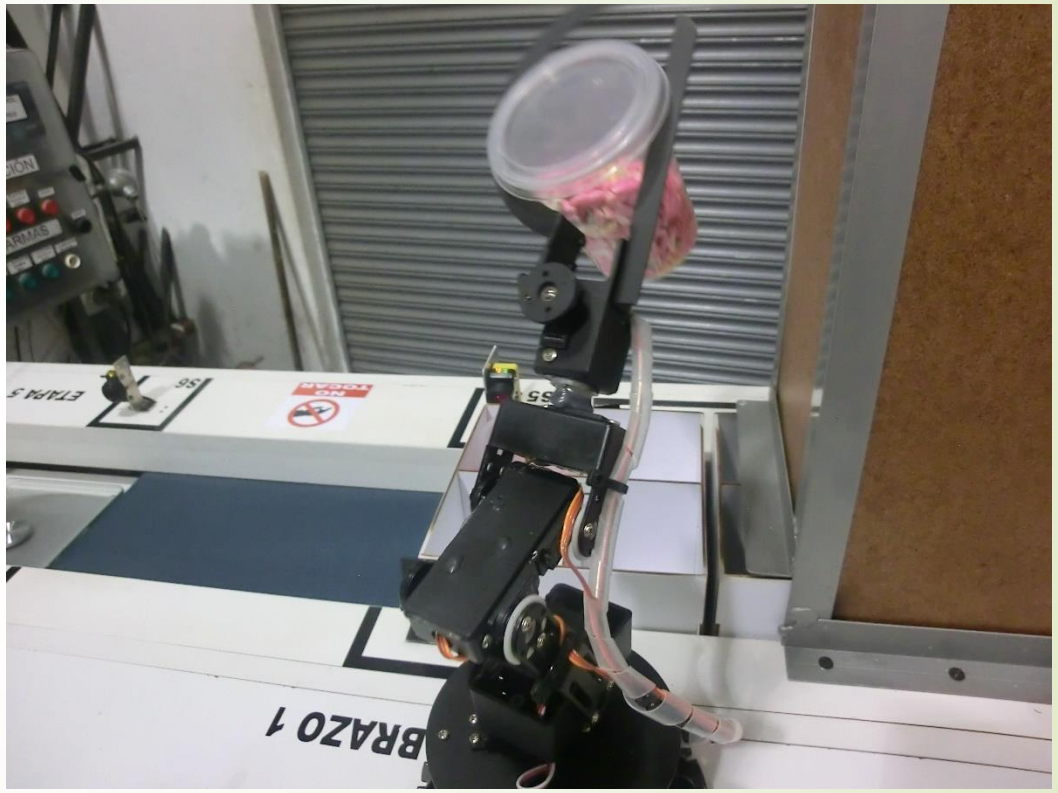
Elementos:

- Pistón Eléctrico
- Servomotor engranes metálicos 12kg/cm
- Tarjeta Arduino UNO R3
- AdafruitMotorShield v2.3
- Bomba de vacío
- Ventosa o copa de succión
- Sensor capacitivo LJC30A3-H-Z/AY
- Sensor fotoeléctrico SME312D

Etapa 4

Elementos:

- Motor de alto torque
- Tarjeta Arduino MEGA 2560 R3
- Tarjeta de control motores DC
- Sensor láser QS3OLDL
- Sensor fotoeléctrico SMU315D
- Sensor fotoeléctrico G30-3A70NA
- Brazo robótico banshi



Etapa 5

Elementos:

- brazo robótico Saint Smart 4 axis
- Galga extensiométrica
- Módulo conversor análogo/digital HX711
- Tarjeta Arduino UNO R3
- Sensor fotoeléctrico QS18VP6R (RX) Y QS18SE (TX)
- LCD 20X4 con comunicación I2C

Panel de control

- Luces piloto
- Pulsadores
- LCD 20X4 con comunicación I2C
- Conexión USB

Automatización del módulo didáctico

- Para automatizar el módulo didáctico se unifican todos los elementos de control, sensores y tarjetas de control pertenecientes a cada una de las etapas, para que realicen una función en específico en el proceso de llenado y envasado de sólidos.
- El código utilizado para automatizar cada una de las etapas se muestra a continuación.

ETAPA 1

Declaración de variables e importación de librerías

```
#define MOTOR1_CTL1  8  // I1
#define MOTOR1_CTL2  9  // I2
#define MOTOR1_PWM   10 // EA
#define MOTOR2_CTL1  12  // I3
#define MOTOR2_CTL2  13 // I4
#define MOTOR2_PWM   11 // EB
#define MOTOR_DIR_FORWARD  0
#define MOTOR_DIR_BACKWARD   1

#include <Encoder.h> // incluye la libreria encoder para poder leer un encoder
Encoder mesa_giratoria(2, 14);// defino la funcion mesa_giratoria en Encoder con los pines 2 y 3 de la arduino MEGA
// Se puede realizar el cambio de estos pines de acuerdo a la necesidad del usuario
//   Mejor Rendimiento: pines que disponen de capacidad de interrupciones
//   Buen rendimiento: solo un pin que disponga de capacidad de interrpciones
//   Bajo rendimiento: ningun pin que tenga capacidad de interrupciones
// Declaracion de variables
volatile int inicio;
int motor = 6;
int i = 0;
volatile long x;
long anterior;
long actual;
int reset = 50;
int reset1 = 48;
int reset2 = 46;
int alarma_etapa1;
int sc_vasos = 24;
int luz_piloto_inicio = 53;
int luz_piloto_paro = 52;
int luz_piloto_etapa1 = 51;
volatile int s1;
int enviar_etapa2 = 4;
int enviar_etapa3 = 5;
int enviar_etapa4 = 26;
volatile int esperar2;
volatile int esperar3;
volatile int esperar4;
volatile int continuar;
const int boton_inicio = 7;
unsigned long wdog = 0;          /* watchdog */
const int tiempoAntirebote = 10;
int estadoBoton;
int estadoBotonAnterior = 1;
int dispensar;
int botones;
long int c;
int a;
```

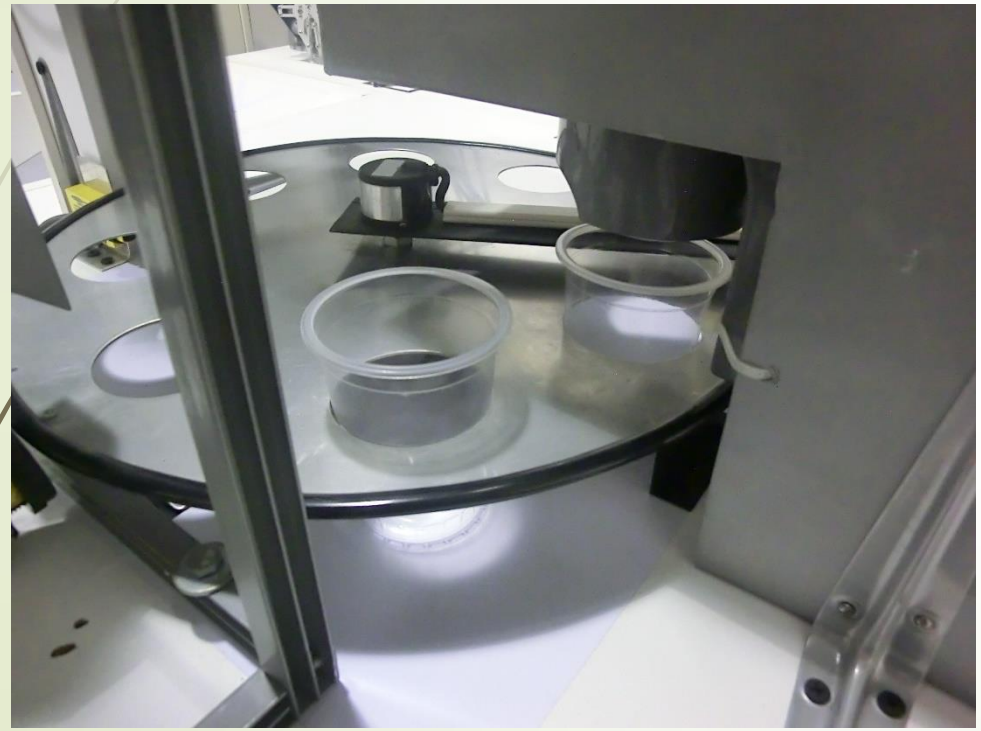
Configuración de pines digitales y enceramiento de variables

```
void setup() {

  Serial.begin(9600);
  pinMode(enviar_etapa2, OUTPUT);
  pinMode(enviar_etapa3, OUTPUT);
  pinMode(enviar_etapa4, OUTPUT);
  pinMode(MOTOR1_CTL1, OUTPUT);
  pinMode(MOTOR1_CTL2, OUTPUT);
  pinMode(MOTOR1_PWM, OUTPUT);
  pinMode(MOTOR2_CTL1, OUTPUT);
  pinMode(MOTOR2_CTL2, OUTPUT);
  pinMode(MOTOR2_PWM, OUTPUT);
  pinMode(boton_inicio, INPUT);
  pinMode(luz_piloto_inicio, OUTPUT);
  pinMode(luz_piloto_paro, OUTPUT);
  pinMode(luz_piloto_etapa1, OUTPUT);
  pinMode(sc_vasos, INPUT_PULLUP);
  pinMode(reset, OUTPUT);
  pinMode(reset1, OUTPUT);
  pinMode(reset2, OUTPUT);
  digitalWrite(luz_piloto_inicio, HIGH);
  digitalWrite(luz_piloto_paro, HIGH);
  digitalWrite(luz_piloto_etapa1, HIGH);
  digitalWrite(enviar_etapa2, HIGH);
  digitalWrite(enviar_etapa3, HIGH);
  digitalWrite(enviar_etapa4, HIGH);
  digitalWrite(reset, HIGH);
  digitalWrite(reset1, HIGH);
  digitalWrite(reset2, HIGH);
  analogWrite(motor, 0);
  inicio = 3;
  continuar = 0;
  esperar2 = 0;
  esperar3 = 0;
  esperar4 = 0;
  actual = 0;
  anterior = -999;
  x = 0;
  c = 0;
  i = 0;
  dispensar = 0;
  botones = 1;
  motorStart(1, MOTOR_DIR_BACKWARD);
  setSpeed(1, 160 );
  delay(200);
  motorStop(1);
  alarma1();
  a = 0;
  Serial.println("PULSE INICIO PARA INICIALIZAR EL PROCESO.....");
  attachInterrupt(1, paro_general, CHANGE);

}
```

Condición para poder iniciar el proceso

```
void loop() {

  alarma1();
  if (((botones == 1)) && alarma_etapa1 == 0 ) {
    estadoBoton = digitalRead (boton_inicio);
    if (estadoBoton  != estadoBotonAnterior) {
      if (antirebote (boton_inicio)) {
        digitalWrite(luz_piloto_inicio, LOW);
        digitalWrite(luz_piloto_paro, HIGH);
        Serial.println ("PROCESO INICIADO...");
        Serial.println ("PLATO GIRATORIO EN MOVIMIENTO");
        attachInterrupt(2, ajuste, FALLING);
        digitalWrite(enviar_etapa4, LOW);
        delay(10);
        digitalWrite(enviar_etapa4, HIGH);
        inicio = 3;
        botones = 0;
        analogWrite(motor, 72);
      }
    }
    estadoBotonAnterior = estadoBoton;
  }
```

Dispensado del recipiente, lectura del encoder y envío de solicitudes a las etapas 2, 3 y 4.



```
switch (inicio) {

    case 1:

      x = actual % 180;
      if (x == 0 && continuar == 0) {
        analogWrite(motor, 0);
        mesa_giratoria.write(0);
        alarma1();
        dispensar = 0;
        Serial.println("MOTOR DETENIDO...");
        Serial.println("INTERRUPCIONES HABILITADAS....ESPERANDO
RESPUESTA DE LAS ETAPAS 2,3 Y 4--->");
        attachInterrupt(5, respuesta_etapa3, LOW);
        attachInterrupt(4, respuesta_etapa2, LOW);
        attachInterrupt(3, respuesta_etapa4, LOW);
        //solicitud etapa 3
        digitalWrite(enviar_etapa3, LOW);
        delay(10);
        digitalWrite(enviar_etapa3, HIGH);
        delay(100);
        //solicitud etapa 2
        digitalWrite(enviar_etapa2, LOW);
        delay(10);
        digitalWrite(enviar_etapa2, HIGH);
        delay(100);
        //solicitud etapa 4
        digitalWrite(enviar_etapa4, LOW);
        delay(10);
        digitalWrite(enviar_etapa4, HIGH);
        delay(100);
        // Dispensa el recipiente
        Serial.println("Dispensar recipiente ---> ");
        motorStart(1, MOTOR_DIR_FORWARD);
        setSpeed(1, 235 );
        delay(400);
        motorStop(1);
        delay(300);
        motorStart(1, MOTOR_DIR_BACKWARD);
        setSpeed(1, 180 );
        delay(350);
        motorStop(1);
        inicio = 2;
      }
      else {
        analogWrite(motor, 78);
        actual = mesa_giratoria.read();
        if (actual != anterior) {
          anterior = actual;
          Serial.println(actual);
          continuar = 0;
        }
      }
      break;
```

Respuestas de etapas 2, 3, y 4
Funciones de tipo interrupción

```
    case 2:

      for (int i = 0; i < 101; i++) {
        alarma_etapa1 = digitalRead(sc_vasos);
        delay(10);
      }
      if ( esperar2 == 1 && esperar3 == 1 && esperar4 == 1 &&
alarma_etapa1 == 0 ) {
        Serial.println(" CONTINUAR --->");
        inicio = 1;
        esperar2 = 0;
        esperar3 = 0;
        esperar4 = 0;
        continuar = 1;
        dispensar = 1;
      }
      else {
        inicio = 2;
      }
      break;
    case 3:
      break;
  }
}

void ajuste() {

  inicio = 1;
  mesa_giratoria.write(1);
  Serial.println(" POSICION INICIAL AJUSTADA ");
  continuar = 1;
}

void respuesta_etapa2() {

  detachInterrupt(4);
  Serial.println(" RESPUESTA DESDE ETAPA 2 --->");
  esperar2 = 1;
  continuar = 1;
}

void respuesta_etapa3() {

  detachInterrupt(5);
  Serial.println(" RESPUESTA DESDE ETAPA 3 --->");
  esperar3 = 1;
  continuar = 1;
}

void respuesta_etapa4() {

  detachInterrupt(3);
  Serial.println(" RESPUESTA DESDE ETAPA 4 --->");
  esperar4 = 1;
  continuar = 1;
}
```

Funciones para el manejo del driver L298N

```
void setSpeed(char motor_num, char motor_speed) {

  if (motor_num == 1)
  {
    analogWrite(MOTOR1_PWM, motor_speed);
  }
  else
  {
    analogWrite(MOTOR2_PWM, motor_speed);
  }
}

void motorStart(char motor_num, byte direction) {

  char pin_ctl1;
  char pin_ctl2;
  if (motor_num == 1)
  {
    pin_ctl1 = MOTOR1_CTL1;
    pin_ctl2 = MOTOR1_CTL2;
  }
  else
  {
    pin_ctl1 = MOTOR2_CTL1;
    pin_ctl2 = MOTOR2_CTL2;
  }
  switch (direction) {
    case MOTOR_DIR_FORWARD:
      {
        digitalWrite(pin_ctl1, LOW);
        digitalWrite(pin_ctl2, HIGH);
      }
      break;
    case MOTOR_DIR_BACKWARD:
      {
        digitalWrite(pin_ctl1, HIGH);
        digitalWrite(pin_ctl2, LOW);
      }
      break;
  }
}

void motorStop(char motor_num) {
  setSpeed(motor_num, 0);
  if (motor_num == 1)
  {
    digitalWrite(MOTOR1_CTL1, HIGH);
    digitalWrite(MOTOR1_CTL2, HIGH);
  }
  else
  {
    digitalWrite(MOTOR2_CTL1, HIGH);
    digitalWrite(MOTOR2_CTL2, HIGH);
  }
}
```

Función de antirebote
Función de la alarma 1


```
boolean antirebote  (int pin ) {
  int  contador = 0;
  boolean estado;            // guarda el estado del botón
  boolean estadoAnterior;    // guarda el ultimo estado del botón

  do {
    estado = digitalRead (pin);
    if (estado != estadoAnterior ) { // comparamos el estado actual
      contador = 0;                  // reiniciamos el contador
      estadoAnterior = estado;
    }
    else {
      contador = contador + 1;      // aumentamos el contador en 1
    }
    delay (1);
  }
  while (contador < tiempoAntirebote);
  return estado;
}

void alarma1() {

  alarma_etapa1 = digitalRead(sc_vasos);
  if (alarma_etapa1 == HIGH) {

    digitalWrite(luz_piloto_etapa1, LOW);
  }
  else {

    digitalWrite(luz_piloto_etapa1, HIGH);
  }
}
```

Función tipo interrupción del paro general


```
void paro_general() {
  detachInterrupt(1);
  detachInterrupt(2);
  detachInterrupt(3);
  detachInterrupt(4);
  detachInterrupt(5);
  digitalWrite(reset, LOW);
  delay(10);
  digitalWrite(reset, HIGH);
  delay(10);
  digitalWrite(reset1, LOW);
  delay(10);
  digitalWrite(reset1, HIGH);
  delay(10);
  digitalWrite(reset2, LOW);
  delay(10);
  digitalWrite(reset2, HIGH);
  delay(10);
  analogWrite(motor, 0);
  motorStop(1);
  Serial.println ("PROCESO DETENIDO...");
  digitalWrite(luz_piloto_inicio, HIGH);
  digitalWrite(luz_piloto_paro, LOW);
  digitalWrite(luz_piloto_etapa1, HIGH);
  digitalWrite(enviar_etapa2, HIGH);
  digitalWrite(enviar_etapa3, HIGH);
  digitalWrite(enviar_etapa4, HIGH);
  botones = 1;
  inicio = 3;
  continuar = 0;
  esperar2 = 0;
  esperar3 = 0;
  esperar4 = 0;
  actual = 0;
  anterior = -999;
  x = 0;
  alarma1();
  Serial.println("PULSE INICIO PARA INICIALIZAR EL PROCESO.....");
  attachInterrupt(1, paro_general, CHANGE);
}
```

ETAPA 2

Declaración de variables
e importación de librerías

```
#define MOTOR1_CTL1  8  // I1
#define MOTOR1_CTL2  9  // I2
#define MOTOR1_PWM   10 // EA
#define MOTOR2_CTL1  12  // I3
#define MOTOR2_CTL2  13 // I4
#define MOTOR2_PWM   11 // EB
#define MOTOR_DIR_FORWARD  0
#define MOTOR_DIR_BACKWARD   1

#include <FuzzyRule.h>
#include <FuzzyComposition.h>
#include <Fuzzy.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzyOutput.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzySet.h>
#include <FuzzyRuleAntecedent.h>
#include "HX711.h"
HX711 scale(A1, A0);// Pin DOUT Módulo HX711 conectar A1 (Arduino),
Pin PD_SCK Módulo HX711 conectar A0 (Arduino)
// El parametro ganancia es omitido debido a que la ganancia preestabl
ecida es 128
volatile int inicio;
int v_analogico = A2;
int luz_piloto_etapa2 = 5;
const int sensor_etapa2 = 6;
int nivel_anterior;
int i;
int nivel;
int lectura[30];
int total;
int promedio;
volatile int acceso;
int s2;
float pesopromedio;
int respuesta_etapa2 = 4;
float salida, vel;
float sp;
Fuzzy* fuzzy = new Fuzzy();// Petición de la creación de un nuevo
control Fuzzy
```

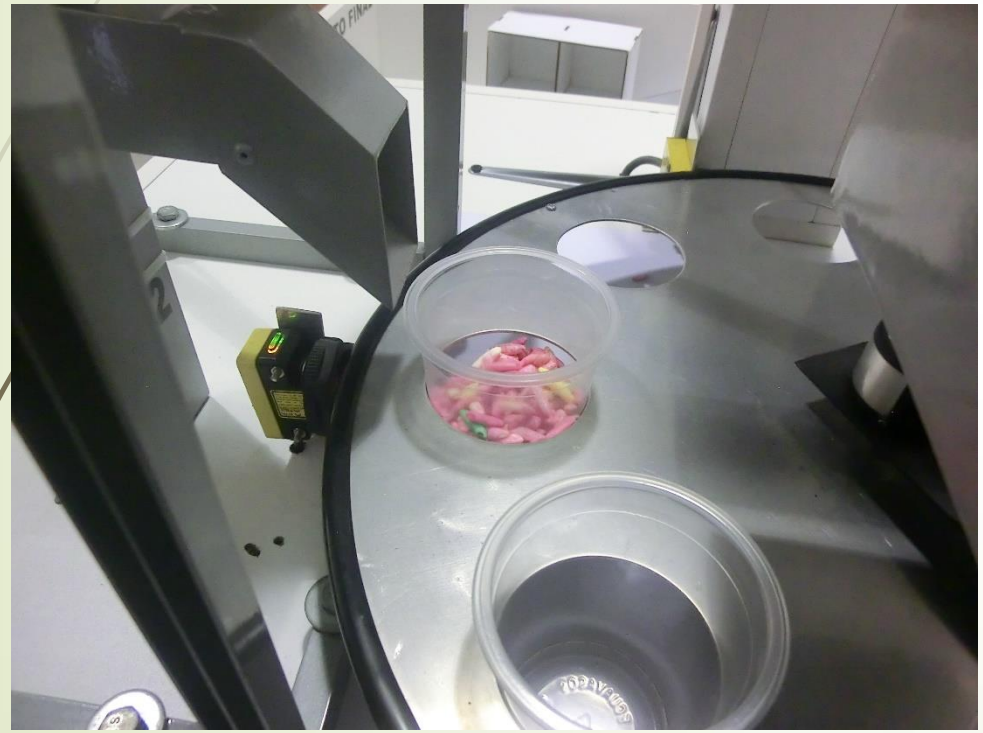
Configuración de pines digitales, enceramiento de variables y declaración De parámetros para el controlador difuso

```
void setup() {

  Serial.begin(9600);
  pinMode(respuesta_etapa2, OUTPUT);
  pinMode(luz_piloto_etapa2, OUTPUT);
  pinMode(sensor_etapa2, INPUT_PULLUP);
  digitalWrite(respuesta_etapa2, HIGH);
  digitalWrite(luz_piloto_etapa2, HIGH);
  inicio =  2;
  acceso = 0;
  s2 = 0;
  nivel_anterior = 0;
  total = 0;
  promedio = 0;
  nivel_material();
  attachInterrupt(0, recibir_etapa1, LOW);
  alarma2();

  for (i = 0; i < 30; i++) {

    lectura[i] = 0;
  }
  i = 0;

  FuzzyInput* peso = new FuzzyInput(1);  // Crear FuzzyInput peso
  //Creando objetos FuzzySet que conforman el FuzzyInput peso
  FuzzySet* pesomuybajo = new FuzzySet(0, 0, 0, 30); // Parámetros
para pesos muy pequeños (funcion trapezoidal)
  peso->addFuzzySet(pesomuybajo); // Adicionar a FuzzySet pesomuybajo
en peso
  FuzzySet* pesobajo = new FuzzySet(20, 30, 30, 35); // Parámetros
para pesos bajos (funcion triangular)
  peso->addFuzzySet(pesobajo); // Adicionar a FuzzySet pesomuybajo en
peso
  FuzzySet* pesonormal = new FuzzySet(32, 35, 35, 40); // Parámetros
para pesos normales (funcion triangular)
  peso->addFuzzySet(pesonormal); // Adicionar a FuzzySet pesonormal en
peso
  FuzzySet* pesoalto = new FuzzySet(37, 40, 40, 50); // Parámetros
para pesos altos (funcion triangular)
  peso->addFuzzySet(pesoalto); // Adicionar a FuzzySet pesoalto en
peso
  FuzzySet* pesomuyalto = new FuzzySet(45, 50, 50, 50); // Parámetros
para pesos muy altos (funcion trapezoidal)
  peso->addFuzzySet(pesomuyalto); // Adicionar a FuzzySet pesomuyalto
en peso
  fuzzy->addFuzzyInput(peso); // Adicionando FuzzyInput (peso) como
objeto Fuzzy

  // Creando la salida FuzzyOutput velocidad
  FuzzyOutput* velocidad = new FuzzyOutput(1);
  //Crear objetos FuzzySet que conforman el FuzzyOutput velocidad
  FuzzySet* velocidadmuybaja = new FuzzySet(150, 150, 150, 180); //
Parámetros para velocidades muy bajas (funcion trapezoidal)
  velocidad->addFuzzySet(velocidadmuybaja); // Adicionar a FuzzySet
velocidadmuybaja en velocidad
  FuzzySet* velocidadbaja = new FuzzySet(170, 190, 190, 210); //
Parámetros para velocidades bajas(funcion triangular)
  velocidad->addFuzzySet(velocidadbaja); // Adicionar a FuzzySet
velocidadbaja en velocidad
```

Conjunto de reglas para el Controlador difuso

```
//DESCRIPCION  DE REGLAS PARA EL CONTROLADOR FUZZY
  // FuzzyRule "Si peso= pesomuybajo entonces velocidad = velocidadmuybaja"
  FuzzyRuleAntecedent* ifPmuybajo = new FuzzyRuleAntecedent(); // Creando un antecedente
  ifPmuybajo->joinSingle(pesomuybajo); // Adicionando a FuzzySet un objeto del tipo antecedente
  FuzzyRuleConsequent* thenVmuybaja = new FuzzyRuleConsequent(); // Creando una expresión consecuente
  thenVmuybaja->addOutput(velocidadmuybaja);// Adicionando a FuzzySet un objeto de tipo consecuente
  // Creando la regla Fuzzy
  FuzzyRule* fuzzyRule01 = new FuzzyRule(1, ifPmuybajo, thenVmuybaja); // Pasando el antecedente en una expresión consecuente
  // Adicionando la regla Fuzzy como objeto Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule01);

  // FuzzyRule "Si peso= pesobajo entonces velocidad = velocidadbaja"
  FuzzyRuleAntecedent* ifPbajo = new FuzzyRuleAntecedent(); // Creando un antecedente
  ifPbajo->joinSingle(pesobajo); // Adicionando a FuzzySet un objeto del tipo antecedente
  FuzzyRuleConsequent* thenVbaja = new FuzzyRuleConsequent(); // Creando una expresión consecuente
  thenVbaja->addOutput(velocidadbaja);// Adicionando a FuzzySet un objeto de tipo consecuente
  // Creando la regla Fuzzy
  FuzzyRule* fuzzyRule02 = new FuzzyRule(2, ifPbajo, thenVbaja); // Pasando el antecedente en una expresión consecuente
  // Adicionando la regla Fuzzy como objeto Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule02);

  // FuzzyRule "Si peso = normal entonces velocidad = normal"
  FuzzyRuleAntecedent* ifPnormal = new FuzzyRuleAntecedent(); // Creando un antecedente
  ifPnormal->joinSingle(pesonormal); // Adicionando a FuzzySet un objeto del tipo antecedente
  FuzzyRuleConsequent* thenVnormal = new FuzzyRuleConsequent(); // Creando una expresión consecuente
  thenVnormal->addOutput(velocidadnormal); // Adicionando a FuzzySet un objeto de tipo consecuente
  // Creando la regla Fuzzy
  FuzzyRule* fuzzyRule03 = new FuzzyRule(3, ifPnormal, thenVnormal); // Pasando el antecedente en una expresión consecuente
  // Adicionando la regla Fuzzy como objeto Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule03);

  // FuzzyRule "Si peso = alto entonces velocidad = baja"
  FuzzyRuleAntecedent* ifPalto = new FuzzyRuleAntecedent(); // Creando un antecedente
  ifPalto->joinSingle(pesoalto); // Adicionando a FuzzySet un objeto del tipo antecedente
  FuzzyRuleConsequent* thenValta = new FuzzyRuleConsequent(); // Creando una expresión consecuente
  thenValta->addOutput(velocidadalta); // Adicionando a FuzzySet un objeto de tipo consecuente
  // Creando la regla Fuzzy
  FuzzyRule* fuzzyRule04 = new FuzzyRule(4, ifPalto, thenValta); // Pasando el antecedente en una expresión consecuente
  // Adicionando la regla Fuzzy como objeto Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule04);
```

Dispensado del material, fusificación y defusificación.



```
void loop() {

  nivel_material();
  Serial.print("Nivel de material:   ");
  Serial.print(nivel);
  Serial.println("\t");
  switch (inicio) {
    case 1:
      if ( s2 == 1 ) {
        pesopromedio = scale.get_units(1);
        Serial.print("Lectura:\t");
        Serial.print(pesopromedio, 2);
        Serial.print("\t| Promedio:\t");
        Serial.println(scale.get_units(1), 3);
        float e = sp - pesopromedio;
        Serial.print("ERROR: ");
        Serial.println(e);
        fuzzy->setInput(1, e);
        fuzzy->fuzzify();
        salida = fuzzy->defuzzify(1);
        vel = map(salida , 0, 6405, 0, 255);
        setSpeed(1, salida);
        motorStart(1, MOTOR_DIR_FORWARD);
        Serial.print("VELOCIDAD:  ");
        Serial.println(salida);
        if ( e <= 0) {
          nivel_material();
          motorStop(1);
          alarma2();
          Serial.println("DISPENSADO LISTO --->");
          delay(500);
          inicio = 2;
          s2 = 0;
          acceso = 0;
          attachInterrupt(0, recibir_etapa1, LOW);
          // Envia una respuesta a la etapa1 para que pueda continuar
          digitalWrite(respuesta_etapa2, LOW);
          delay(10);
          digitalWrite(respuesta_etapa2, HIGH);
        }
      }
      else {

        Serial.println("CONTINUAR ---> ");
        inicio = 2;
        acceso = 0;
        attachInterrupt(0, recibir_etapa1, LOW);
        // Envia una respuesta a la etapa1 para que pueda continuar
        digitalWrite(respuesta_etapa2, LOW);
        delay(10);
        digitalWrite(respuesta_etapa2, HIGH);
      }
      break;
```

Encerado de la balanza

```
case 2:
      if (acceso == 1) {
        s2 = digitalRead(sensor_etapa2);
        if (s2 == HIGH) {
          Serial.println("RECIPIENTE DETECTADO --->");
          //Encerado de la balanza
          scale.read();                      // lectura en crudo del
ADC
          scale.read_average(2);      // promedio de 10 muestras del
ADC
          scale.get_value(1);         // promedio de 5 lecturas del
ADC menos el peso de la tara (todavía sin definir)
          scale.get_units(1);          // promedio de 5 lecturas del
ADC menos el peso de la tara (sin definir ) dividido
          // para el parámetro de escala ( aún no establecido )
          scale.set_scale(2681.f);                    // Este valor
se obtiene mediante la calibración de la escala con pesos conocidos
          scale.tare();                                          //
Resetea la escala a 0
          scale.read();                 // lectura en crudo del ADC
          scale.read_average(2);       // promedio de 5 muestras del
ADC
          scale.get_value(1);         // promedio de 5 lecturas del
ADC menos el peso de la tara , obtenido con la tara ( )
          scale.get_units(1);           // promedio de 5 lecturas de
la ADC menos el peso de la tara, dividido
          // Para el parámetro de escala establecido con set_scale
          Serial.println("ENCERADO DE LA BALANZA LISTO --->");
        }
        else {
          Serial.println("RECIPIENTE NO DETECTADO --->");
        }
        inicio = 1;
        attachInterrupt(0, recibir_etapa1, LOW);
      }
      break;
  }
}
```

Funciones de alarma 2
Función de tipo interrupción
Cálculo del nivel de material


```
void recibir_etapa1() {
  detachInterrupt(0);
  acceso = 1;
  Serial.println("ACCESO PERMITIDO --->");
}
void alarma2() {

  if (nivel <= 20 ) {
    digitalWrite(luz_piloto_etapa2, LOW);
  }
  else {

    digitalWrite(luz_piloto_etapa2, HIGH);
  }
}

void nivel_material() {

  for (i = 0; i < 30; i++) {

    total = total - lectura[i];
    lectura[i] = analogRead(v_analogico);
    total = total + lectura[i];
  }
  promedio = total / i;
  nivel = map(promedio, 285, 1023, 200, 0);
  i = 0;
}
```

Funciones para el manejo del driver L298N

```
void setSpeed(char motor_num, char motor_speed) {

  if (motor_num == 1)
  {
    analogWrite(MOTOR1_PWM, motor_speed);
  }
  else
  {
    analogWrite(MOTOR2_PWM, motor_speed);
  }
}

void motorStart(char motor_num, byte direction) {

  char pin_ctl1;
  char pin_ctl2;
  if (motor_num == 1)
  {
    pin_ctl1 = MOTOR1_CTL1;
    pin_ctl2 = MOTOR1_CTL2;
  }
  else
  {
    pin_ctl1 = MOTOR2_CTL1;
    pin_ctl2 = MOTOR2_CTL2;
  }
  switch (direction) {
    case MOTOR_DIR_FORWARD:
      {
        digitalWrite(pin_ctl1, LOW);
        digitalWrite(pin_ctl2, HIGH);
      }
      break;
    case MOTOR_DIR_BACKWARD:
      {
        digitalWrite(pin_ctl1, HIGH);
        digitalWrite(pin_ctl2, LOW);
      }
      break;
  }
}

void motorStop(char motor_num) {
  setSpeed(motor_num, 0);
  if (motor_num == 1)
  {
    digitalWrite(MOTOR1_CTL1, HIGH);
    digitalWrite(MOTOR1_CTL2, HIGH);
  }
  else
  {
    digitalWrite(MOTOR2_CTL1, HIGH);
    digitalWrite(MOTOR2_CTL2, HIGH);
  }
}
```

ETAPA 3

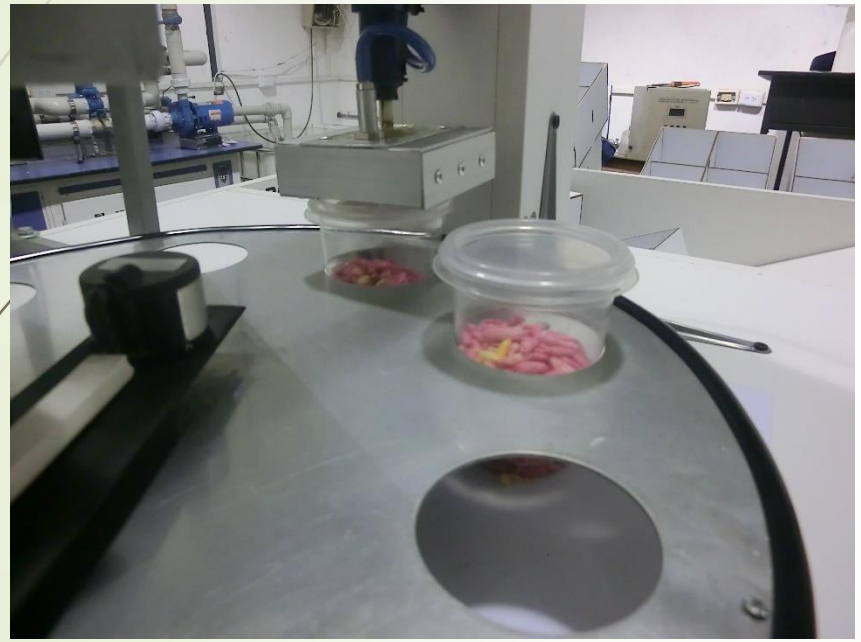
Declaración de variables e importación de librerías

```
#include <Wire.h>// incluye libreria de comunicacion I2C
#include <Adafruit_MotorShield.h>// incluye libreria de tarjeta controladora de motores
#include "utility/Adafruit_PWMServoDriver.h" // incluye utilididad PWM para servos correspondiente a la libreria de la tarjeta controladora de motores
#include <Servo.h> // incluye libreria manejo de servos
Adafruit_MotorShield AFMS = Adafruit_MotorShield(); // crea un objeto con una direccion por defecto para la comunicacion I2C
Adafruit_DCMotor *bomba = AFMS.getMotor(3); // crea un objeto perteneciente a M3
Adafruit_DCMotor *piston = AFMS.getMotor(4); // crea un objeto perteneciente a M2
Servo servo1; // crea un objeto tipo servo
int i;
int respuesta_etapa3 = 4;
volatile int inicio;
int luz_piloto_etapa3 = 5;
volatile int acceso;
volatile int s3;
int sc_tapas = 8;
int estado;
```

Configuración de pines digitales, enceramiento de variables

```
void setup() {

  Serial.begin(9600);
  pinMode(respuesta_etapa3, OUTPUT);
  pinMode(luz_piloto_etapa3, OUTPUT);
  pinMode(sc_tapas, INPUT_PULLUP);
  digitalWrite(respuesta_etapa3, HIGH);
  digitalWrite(luz_piloto_etapa3, HIGH);
  inicio = 2;
  acceso = 0;
  s3 = 0;
  Serial.begin(9600);// inicia una comunicacion serial a 9600 bps
  AFMS.begin();  // asigno una frecuencia de trabajo de 1.6KHz
  piston-> setSpeed(50);
  piston-> run(FORWARD);
  for (i = 0; i < 101; i++) {
    delay(10);
  }
  servo1.attach(9); // asigno al objeto servo1 un pin de control #9
  for (i = 110; i >= 90; i -= 1) {
    servo1.write(i);
    delay(30);
  }
  //prueba de funcionamiento de la bomba de vacio
  bomba->setSpeed(0);// accionamiento de la bomba
  bomba->run(RELEASE);// deteniene la bomba
  piston-> run(RELEASE);
  attachInterrupt(0, recibir_etapa1, FALLING);
  for (i = 0; i < 11; i++) {
    delay(10);
  }
  alarma3();
}
```

Procedimiento para el sellado del recipiente



```
void loop() {

  servo1.detach();

  switch (inicio) {
    case 1:

      for (i = 0; i <= 101; i += 1) {
        estado = digitalRead(sc_tapas);
        delay(10);
      }

      if ( s3 == 1 && estado == LOW ) {
        servo1.attach(9);
        piston->setSpeed(50);
        piston->run(FORWARD);
        delay(1000);
        for (i = 90; i <= 164; i += 1) {
          servo1.write(i);
          delay(5);
        }
        delay(250);
        bomba->setSpeed(255); // accionamiento de la bomba
        bomba->run(FORWARD);
        delay(250);
        piston->setSpeed(120);
        piston->run(BACKWARD);
        delay(100);
        piston->setSpeed(150);
        piston->run(FORWARD);
        delay(1000);
        for (i = 164; i >= 16; i -= 1) {
          servo1.write(i);
          delay(5);
        }
        delay(1000);
        bomba->run(RELEASE);
        piston->setSpeed(150);
        piston->run(BACKWARD);
        delay(1500);
        piston->run(RELEASE);

        delay(9000);
        piston->setSpeed(130);
        piston->run(FORWARD);
        delay(1000);
        piston->setSpeed(150);
        piston->run(BACKWARD);
        delay(1200);
        piston->setSpeed(50);
        piston->run(FORWARD);
        delay(500);
        for (i = 16; i <= 90; i += 1) {
          servo1.write(i);
          delay(5);
        }
        piston-> run(RELEASE);
        Serial.println("SELLADO --->");
        // Envia una respuesta para que continue la etapa 1
        digitalWrite(respuesta_etapa3, LOW);
        delay(10);
```

Si no se detecta recipiente alguno
Función para recibir solicitud de Etapa1

```
else {

        if (estado == LOW) {

          alarma3();
          Serial.println(" CONTINUAR --->");
          inicio = 2;
          acceso = 0;
          servo1.detach();
          // Envia una respuesta para que continue la etapa 1
          digitalWrite(respuesta_etapa3, LOW);
          delay(10);
          digitalWrite(respuesta_etapa3, HIGH);
          attachInterrupt(0, recibir_etapa1, FALLING);
        }
        else {

          inicio = 1;
          alarma3();
        }
      }
      break;

    case 2:
      alarma3();
      if (acceso == 1) {
        inicio = 1;
        servo1.attach(9);
        attachInterrupt(1, sensor, FALLING);
      }
      break;
  }
}
```

Funciones de tipo interrupción y función de la alarma 3


```
      else {

        if (estado == LOW) {

          alarma3();
          Serial.println(" CONTINUAR --->");
          inicio = 2;
          acceso = 0;
          servo1.detach();
          // Envia una respuesta para que continue la etapa 1
          digitalWrite(respuesta_etapa3, LOW);
          delay(10);
          digitalWrite(respuesta_etapa3, HIGH);
          attachInterrupt(0, recibir_etapa1, FALLING);
        }
        else {

          inicio = 1;
          alarma3();
        }
      }
      break;

    case 2:
      alarma3();
      if (acceso == 1) {
        inicio = 1;
        servo1.attach(9);
        attachInterrupt(1, sensor, FALLING);
      }
      break;
  }
}
void sensor() {

  detachInterrupt(1);
  Serial.println("RECIPIENTE DETECTADO --->");
  s3 = 1;
}

void recibir_etapa1() {

  detachInterrupt(0);
  acceso = 1;
  Serial.println("ACCESO PERMITIDO --->");
}

void alarma3() {

  estado = digitalRead(sc_tapas);
  if ( estado == HIGH ) {
    digitalWrite(luz_piloto_etapa3, LOW);
  }
  else {
    digitalWrite(luz_piloto_etapa3, HIGH);
  }
}
```

ETAPA 4

Declaración de variables
e importación de librerías

```
#include <Servo.h>
#include <Wire.h>
Servo s1;
Servo s2;
Servo s3;
Servo s4;
Servo s5;
int i;
int j;
int pos = 0;
int alarma_etapa4;
int lectura_s4;
int inicio_banda;
int avanzar;
volatile int inicio;
volatile int s5_1;
const int s4_1 = 24;
const int motor2 = 6;
const int sc_cajas = 22;
const int respuesta_etapa4 = 8;
const int luz_piloto_etapa4 = 12;
const int enviar_etapa5 = 26;
const int reset_etapa5 = 28;
```

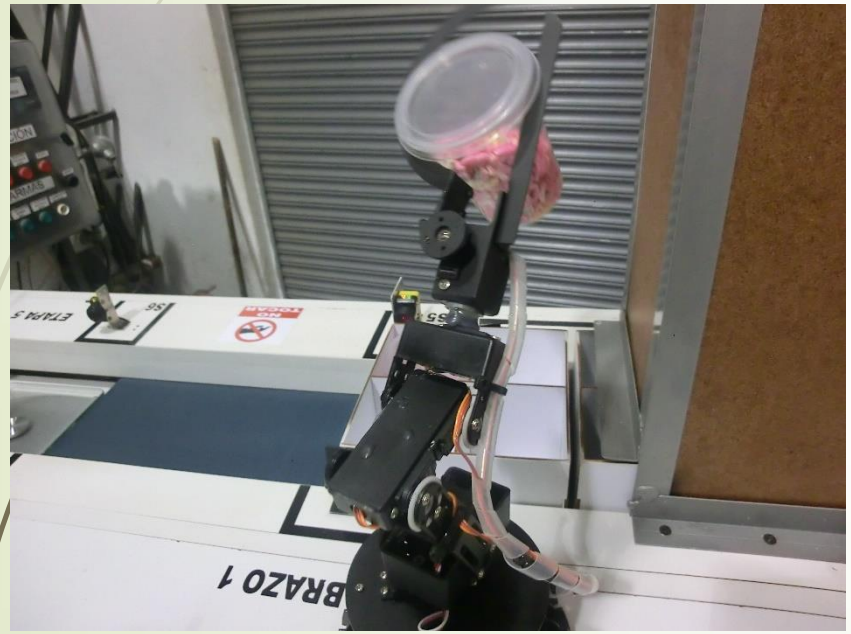
Configurar pines digitales como entradas o salidas y encerar variables

```
void setup() {

  Serial.begin(9600);
  pinMode(sc_cajas, INPUT_PULLUP);
  pinMode(s4_1, INPUT_PULLUP);
  pinMode(respuesta_etapa4, OUTPUT);
  pinMode(enviar_etapa5, OUTPUT);
  pinMode(reset_etapa5, OUTPUT);
  pinMode(luz_piloto_etapa4, OUTPUT);
  digitalWrite(luz_piloto_etapa4, HIGH);
  digitalWrite(respuesta_etapa4, HIGH);
  digitalWrite(enviar_etapa5, HIGH);
  digitalWrite(reset_etapa5, HIGH);
  delay(10);
  digitalWrite(reset_etapa5, LOW);
  delay(10);
  digitalWrite(reset_etapa5, HIGH);
  analogWrite(motor2, 0);
  inicio = 7;
  j = 0;
  inicio_banda = 0;
  s5_1 = 0;
  avanzar = 0;
  alarma4();
  Serial.println("ESPERANDO POSICION INCIAL DE LA CAJA");
  attachInterrupt(1, recibir_etapa1, FALLING);
}
```

Funciones de detención de la banda transportadora, interrupciones por parte de la etapa 1 y movimiento Inicial de las cajas


```
void detener_banda() {
  detachInterrupt(0);
  analogWrite(motor2, 0);
  Serial.println("banda detenida --->");
  inicio = 6;
}
void alarma4() {
  alarma_etapa4 = digitalRead(sc_cajas);
  if (alarma_etapa4 == HIGH) {
    digitalWrite(luz_piloto_etapa4, HIGH);
  }
  else {
    digitalWrite(luz_piloto_etapa4, LOW);
  }
}
void recibir_etapa1() {
  detachInterrupt(1);
  Serial.println("ACCESO PERMITIDO --->");
  movimiento_inicial();
}
void movimiento_inicial() {
  alarma4();
  attachInterrupt(0, detener_banda, RISING);
  if (inicio_banda == 0 && alarma_etapa4 == HIGH) {
    inicio_banda ++;
    Serial.println("EXISTEN CAJAS EL PROCESO CONTINUA --->");
    analogWrite(motor2, 254);
    s1.attach(4);
    s2.attach(5);
    s3.attach(7);
    s4.attach(9);
    s5.attach(10);
    //posicion inicial
    movimiento(4, 15, 1, 80, 90);
    movimiento(1, 15, 1, 85, 52);
    movimiento(2, 15, 1, 120, 147);
    movimiento(3, 15, 1, 120, 138);
    movimiento(5, 15, 1, 90, 15);
    inicio = 7;
    attachInterrupt(1, recibir_etapa1, FALLING);
    digitalWrite(enviar_etapa5, LOW);
    delay(10);
    digitalWrite(enviar_etapa5, HIGH);
    delay(10);
    digitalWrite(respuesta_etapa4, LOW);
    delay(10);
    digitalWrite(respuesta_etapa4, HIGH);
  }
  else {

    if (alarma_etapa4 == LOW) {
      inicio = 8;
      Serial.println("AGREGAR CAJAS AL DISPENSADOR --->");
    }
    else {
      inicio = 5;
    }
  }
}
```

Primer movimiento del brazo robótico




```
  switch (inicio) {
    case 1:
      //movimiento 1
      Serial.println("REALIZANDO EL MOVIMIENTO DEL RECIPIENTE 1 ---
>");
      movimiento(1, 15, 1, 52, 85);
      movimiento(4, 15, 1, 90, 20);
      movimiento(5, 15, 1, 15, 11);
      delay(500);
      movimiento(1, 15, 1, 85, 52);
      delay(500);
      movimiento(4, 15, 1, 20, 90);
      movimiento(2, 15, 1, 147, 90);
      movimiento(5, 25, 1, 11, 167);
      movimiento(3, 15, 1, 138, 110);
      movimiento(2, 15, 1, 90, 122);
      delay(1000);
      movimiento(4, 25, 1, 90, 18);
      movimiento(1, 15, 1, 52, 85);
      //posicion inicial
      movimiento(4, 15, 2, 18, 90);
      movimiento(1, 15, 2, 85, 52);
      movimiento(2, 15, 2, 122, 147);
      movimiento(3, 15, 2, 110, 138);
      movimiento(5, 15, 2, 167, 15);
      delay(1000);
      digitalWrite(respuesta_etapa4, LOW);
      delay(10);
      digitalWrite(respuesta_etapa4, HIGH);
      attachInterrupt(1, recibir_etapa1, FALLING);
      inicio = 7;
      break;
```

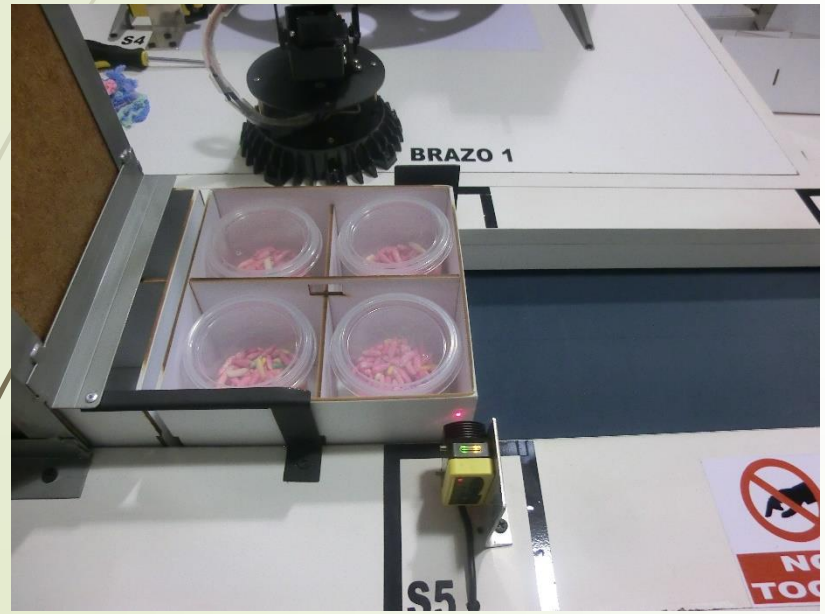
Segundo movimiento del brazo robótico

```
case 2:
      //movimiento
      Serial.println("REALIZANDO EL MOVIMIENTO DEL RECIPIENTE 2 --->");
      movimiento(1, 15, 1, 52, 85);
      movimiento(4, 15, 1, 90, 20);
      movimiento(5, 15, 1, 15, 11);
      delay(500);
      movimiento(1, 15, 1, 85, 52);
      delay(500);
      movimiento(4, 15, 1, 20, 90);
      movimiento(2, 15, 1, 147, 90);
      movimiento(5, 25, 1, 11, 149);
      movimiento(4, 25, 1, 90, 80);
      movimiento(3, 15, 1, 138, 105);
      movimiento(2, 15, 1, 90, 107);
      delay(1000);
      movimiento(4, 25, 1, 80, 19);
      movimiento(1, 15, 1, 52, 85);
      //posicion inicial
      movimiento(4, 15, 2, 19, 90);
      movimiento(1, 15, 2, 85, 52);
      movimiento(2, 15, 2, 107, 147);
      movimiento(3, 15, 2, 105, 138);
      movimiento(5, 15, 2, 149, 15);
      delay(1000);
      digitalWrite(respuesta_etapa4, LOW);
      delay(10);
      digitalWrite(respuesta_etapa4, HIGH);
      attachInterrupt(1, recibir_etapa1, FALLING);
      inicio = 7;
      break;
```

Tercer movimiento del brazo robótico

```
case 3:
      //movimiento
      Serial.println("REALIZANDO EL MOVIMIENTO DEL RECIPIENTE 3 --->");
      movimiento(1, 15, 1, 52, 85);
      movimiento(4, 15, 1, 90, 20);
      movimiento(5, 15, 1, 15, 11);
      delay(500);
      movimiento(1, 15, 1, 85, 52);
      delay(500);
      movimiento(4, 15, 1, 20, 90);
      movimiento(2, 15, 1, 147, 90);
      movimiento(4, 15, 1, 90, 110);
      movimiento(5, 25, 1, 11, 176);
      movimiento(3, 15, 1, 138, 130);
      delay(1000);
      movimiento(2, 15, 1, 90, 20);
      movimiento(1, 15, 1, 52, 85);
      //posicion inicial
      movimiento(2, 15, 2, 20, 147);
      movimiento(4, 15, 2, 110, 90);
      movimiento(5, 15, 2, 176, 15);
      movimiento(3, 15, 2, 130, 138);
      movimiento(1, 15, 2, 85, 52);
      delay(1000);
      digitalWrite(respuesta_etapa4, LOW);
      delay(10);
      digitalWrite(respuesta_etapa4, HIGH);
      attachInterrupt(1, recibir_etapa1, FALLING);
      inicio = 7;
      break;
```

Cuarto movimiento del brazo robótico




```
case 4:
      //movimiento
      Serial.println("REALIZANDO EL MOVIMIENTO DEL RECIPIENTE 4 --->");
      movimiento(1, 15, 1, 52, 85);
      movimiento(4, 15, 1, 90, 20);
      movimiento(5, 15, 1, 15, 11);
      delay(500);
      movimiento(1, 15, 1, 85, 52);
      delay(500);
      movimiento(4, 15, 1, 20, 90);
      movimiento(2, 15, 1, 147, 90);
      movimiento(4, 25, 1, 90, 113);
      movimiento(5, 25, 1, 11, 150);
      movimiento(3, 15, 1, 138, 140);
      delay(1000);
      movimiento(2, 15, 1, 90, 35);
      movimiento(4, 25, 1, 113, 103);
      movimiento(1, 15, 1, 52, 85);
      //posicion inicial
      movimiento(2, 15, 2, 35, 147);
      movimiento(4, 15, 2, 103, 90);
      movimiento(5, 15, 2, 150, 15);
      movimiento(3, 15, 2, 140, 138);
      movimiento(1, 15, 2, 85, 52);
      delay(1000);
      j = 0;
      inicio = 7;
      attachInterrupt(0, detener_banda, RISING);
      delay(1000);
      analogWrite(motor2, 255);
      digitalWrite(enviar_etapa5, LOW);
      delay(10);
      digitalWrite(enviar_etapa5, HIGH);
      break;
```

Condiciones para la organización de los recipientes envasados


```
case 5:
      lectura_s4 = digitalRead(s4_1);
      if (lectura_s4 == LOW) {
        j++;
        Serial.print("MOVIMIENTO #: ");
        Serial.println(j);
        inicio = j;
      }
      else {
        Serial.println("CONTINUAR --->");
        digitalWrite(respuesta_etapa4, LOW);
        delay(10);
        digitalWrite(respuesta_etapa4, HIGH);
        inicio = 7;
        attachInterrupt(1, recibir_etapa1, FALLING);
      }
      break;
    case 6:
      analogWrite(motor2, 0);
      for (i = 0; i < 201; i++) {
        alarma_etapa4 = digitalRead(sc_cajas);
        delay(10);
      }
      if (alarma_etapa4 == HIGH) {
        digitalWrite(respuesta_etapa4, LOW);
        delay(10);
        digitalWrite(respuesta_etapa4, HIGH);
        attachInterrupt(1, recibir_etapa1, FALLING);
        inicio = 7;
      }
      else {
        inicio = 8;
      }
      break;
    case 7:
      break;
    case 8:
      alarma4();
      for (i = 0; i < 101; i++) {
        alarma_etapa4 = digitalRead(sc_cajas);
        delay(10);
      }
      if (alarma_etapa4 == HIGH) {
        Serial.println("CAJAS AGREGADAS--->");
        Serial.println("CONTINUAR --->");
        inicio = 5;
      }
      else {
        Serial.println("ESPERANDO CAJAS --->");
      }
      break;
  }
}
```

ETAPA 5

Declaración de variables
e importación de librerías

```
#include "HX711.h"
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#if defined(ARDUINO) && ARDUINO >= 100
#define printByte(args)  write(args);
#else
#define printByte(args)  print(args,BYTE);
#endif
uint8_t bell[8]  = {0x4, 0xe, 0xe, 0xe, 0x1f, 0x0, 0x4};
uint8_t note[8]  = {0x2, 0x3, 0x2, 0xe, 0x1e, 0xc, 0x0};
uint8_t clock[8] = {0x0, 0xe, 0x15, 0x17, 0x11, 0xe, 0x0};
uint8_t heart[8] = {0x0, 0xa, 0x1f, 0x1f, 0xe, 0x4, 0x0};
uint8_t duck[8]  = {0x0, 0xc, 0x1d, 0xf, 0xf, 0x6, 0x0};
uint8_t check1[8] = {0x0, 0x1, 0x3, 0x16, 0x1c, 0x8, 0x0};
uint8_t cross[8] = {0x0, 0x1b, 0xe, 0x4, 0xe, 0x1b, 0x0};
uint8_t retarrow[8] = {        0x1, 0x1, 0x5, 0x9, 0x1f, 0x8, 0x4};
LiquidCrystal_I2C lcd(0x27, 20, 4);
HX711 scale(A1, A0);           // Pin DOUT Módulo HX711 conectar A1
(Arduino), Pin PD_SCK Módulo HX711 conectar A0 (Arduino)
// El parametro ganancia es omitido debido a que la ganancia preestabl
ecida es 128
Servo s1;
Servo s2;
Servo s3;
Servo s4;
int i;
int j;
int pos = 0;
int correctas;
int incorrectas;
const int s6 = 11;
int estado_sensor6;
volatile int inicio;
float peso;
float pesopromedio;
int pass = 7 ;
int check = 8;
char n1[20] = "UNIVERSIDAD DE LAS ";
char n2[21] = "FUERZAS ARMADAS ESPE";
char n3[20] = "EXTENSION LATACUNGA";
char n4[17] = "SEBASTIAN PANCHI";
char n5[20] = "WASHINGTON HERRERA";
char n6[20] = "MODULO DIDACTICO DE";
char n7[18] = "AUTOMATIZACION DE";
char n8[19] = "LLENADO Y ENVASADO";
char n9[11] = "DE SOLIDOS";
float peso_bruto;
```

Configuración de los pines Digitales y enceramiento de variables

```
void setup() {
  Serial.begin(9600);
  lcd.init();
  lcd.backlight();
  lcd.createChar(0, bell);
  lcd.createChar(1, note);
  lcd.createChar(2, clock);
  lcd.createChar(3, heart);
  lcd.createChar(4, duck);
  lcd.createChar(5, check1);
  lcd.createChar(6, cross);
  lcd.createChar(7, retarrow);
  lcd.home();
  pinMode(s6, INPUT_PULLUP);
  pinMode(pass, OUTPUT);
  pinMode(check, OUTPUT);
  digitalWrite(pass, HIGH);
  digitalWrite(check, HIGH);
  attachInterrupt(0, recibir_etapa4, LOW);
  inicio = 2;
  i = 0;
  correctas = 0;
  incorrectas = 0;
  estado_sensor6 = 0;
  peso_bruto = ((6.7 * 4) + 158.7 + 4 * (11));
```

Mensaje inicial (demo)

```
  lcd.setCursor(20, 0);
  for (int pos = 0; pos < 19; pos++) {
    lcd.autoscroll();
    lcd.print(n1[pos]);
    delay(50);
  }

  lcd.noAutoscroll();
  lcd.setCursor(19, 1);
  for (int pos = 0; pos < 20; pos++) {
    lcd.print(n2[pos]);
    delay(50);
  }
  lcd.setCursor(20, 3);
  for (int pos = 0; pos < 19; pos++) {
    lcd.print(n3[pos]);
    delay(50);
  }
  delay(1000);
  lcd.clear();
  lcd.setCursor(19, 0);
  for (int pos = 0; pos < 19; pos++) {
    lcd.autoscroll();
    lcd.print(n6[pos]);
    delay(50);
  }
  lcd.noAutoscroll();
  lcd.setCursor(20, 1);
  for (int pos = 0; pos < 17; pos++) {
    lcd.print(n7[pos]);
    delay(50);
  }
  lcd.setCursor(20, 3);
  for (int pos = 0; pos < 18; pos++) {
    lcd.print(n8[pos]);
    delay(50);
  }
  lcd.setCursor(4, 1);
  for (int pos = 0; pos < 10; pos++) {
    lcd.print(n9[pos]);
    delay(50);
  }
  delay(1000);
  lcd.clear();
  lcd.setCursor(4, 0);
  lcd.print("INTEGRANTES:");
  lcd.setCursor(2, 1);
  for (int pos = 0; pos < 16; pos++) {
    lcd.print(n4[pos]);
    delay(50);
  }
  lcd.setCursor(1, 3);
  for (int pos = 0; pos < 18 ; pos++) {
    lcd.print(n5[pos]);
    delay(50);
  }
  delay(1000);
  lcd.clear();
}
```

Enceramiento de la balanza y pesaje del producto final


```
void loop() {
  estado_sensor6 = digitalRead(s6);
  switch (inicio) {
    case 1:
      for (j = 0; j < 101; j++) {
        estado_sensor6 = digitalRead(s6);
        delay(10);
      }
      if (estado_sensor6 == LOW) {
        lcd.clear();
        lcd.setCursor(0, 2);
        lcd.print("ENCERANDO LA BALANZA");
        lcd.setCursor(10, 3);
        lcd.printByte(2);
        s1.attach(5);
        s2.attach(9);
        s3.attach(6);
        s4.attach(10);
        s1.write(15);
        s2.write(25);
        s3.write(106);
        s4.write(133);
        // Encerado de la balanza
        scale.read();                  // lectura en crudo del ADC
        scale.read_average(2);         // promedio de 10 muestras del
ADC
        scale.get_value(1);            // promedio de 5 lecturas del
ADC menos el peso de la tara (todavía sin definir)
        scale.get_units(1);            // promedio de 5 lecturas del
ADC menos el peso de la tara (sin definir ) dividido
        // para el parámetro de escala ( aún no establecido )
        scale.set_scale(445.f);                      // Este valor se
obtiene mediante la calibración de la escala con pesos conocidos
        scale.tare();                                // Resetea la
escala a 0
        scale.read();                  // lectura en crudo del ADC
        scale.read_average(2);        // promedio de 5 muestras del ADC
        scale.get_value(1);            // promedio de 5 lecturas del
ADC menos el peso de la tara , obtenido con la tara ( )
        // Para el parámetro de escala establecido con set_scale
        lcd.setCursor(10, 3);
        lcd.printByte(5);
        lcd.clear();
        Serial.println("INICIANDO LA MEDIDA...");
        for (i = 0; i < 5; i++) {

          pesopromedio = (scale.get_units(20));
          Serial.print("\t| Lectura pesopromedio:\t");
          Serial.println(pesopromedio, 2);
          lcd.setCursor(0, 0);
          lcd.print("PESO BRUTO: ");
          lcd.setCursor(12, 0);
          lcd.print(pesopromedio, 1);
          lcd.setCursor(18, 0);
          lcd.print("gr");
          lcd.setCursor(3, 1);
          lcd.print("TARAS +/-");
          lcd.setCursor(12, 1);
          lcd.print(5);
          lcd.setCursor(14, 1);
          lcd.print("GR");
```

Clasificación del producto final


```
if (pesopromedio <= (peso_bruto + 5) && pesopromedio >= (peso_bruto -
5)) {
          Serial.println("LA CAJA PASA LA INSPECCION --->");
          digitalWrite(pass, LOW);
          digitalWrite(check, HIGH);
          lcd.clear();
          lcd.setCursor(2, 0);
          lcd.print("EL PRODUCTO FINAL");
          lcd.setCursor(8, 1);
          lcd.print("PASA");
          lcd.setCursor(3, 2);
          lcd.print("LA INSPECCION");
          lcd.setCursor(10, 3);
          lcd.printByte(5);
          brazo_robotico();
          i = 0;
          inicio = 2;
          correctas ++;
          lcd.clear();
          attachInterrupt(0, recibir_etapa4, LOW);
        }
        else {
          Serial.println("LA CAJA NO PASA LA INSPECCION X");
          digitalWrite(pass, HIGH);
          digitalWrite(check, LOW);
          lcd.clear();
          lcd.setCursor(2, 0);
          lcd.print("EL PRODUCTO FINAL");
          lcd.setCursor(7, 1);
          lcd.print("NO PASA");
          lcd.setCursor(4, 2);
          lcd.print("LA INSPECCION");
          lcd.setCursor(10, 3);
          lcd.printByte(6);
          brazo_robotico_e();
          i = 0;
          inicio = 2;
          incorrectas ++;
          lcd.clear();
          attachInterrupt(0, recibir_etapa4, LOW);
        }
      }
      else {
        Serial.println("NO SE DETECTO NINGUNA CAJA...");
        inicio = 2;
        attachInterrupt(0, recibir_etapa4, LOW);
      }
      break;
```

Lazo de espera y visualización de la contabilidad del producto final

```
case 2:
      attachInterrupt(0, recibir_etapa4, LOW);

      lcd.setCursor(0, 0);
      lcd.print("CAJAS CONTABILIZADAS");
      lcd.setCursor(0, 1);
      lcd.print("____________________");
      lcd.setCursor(0, 2);
      lcd.print("|  PASS  |");
      lcd.setCursor(12, 2);
      lcd.print("CHECK  |");
      lcd.setCursor(4, 3);
      lcd.print(correctas);
      lcd.setCursor(0, 3);
      lcd.print("|");
      lcd.setCursor(9, 3);
      lcd.print("|");
      lcd.setCursor(19, 3);
      lcd.print("|");
      lcd.setCursor(14, 3);
      lcd.print(incorrectas);
      break;
```

Funciones para movimientos del brazo robótico

```
void brazo_robotico() {

  movimiento(1, 15, 1, 15, 80);
  movimiento(2, 30, 1, 25, 57);
  movimiento(1, 15, 1, 80, 0);
  movimiento(3, 15, 1, 106, 125);
  delay(500);
  movimiento(2, 15, 1, 57, 0);
  delay(1000);
  movimiento(4, 15, 1, 133, 75);
  movimiento(3, 15, 1, 125, 101);
  movimiento(2, 30, 1, 0, 40);
  delay(1000);
  movimiento(1, 15, 5, 0, 80);
  movimiento(2, 15, 1, 40, 10);
  movimiento(4, 15, 1, 75, 133);
  movimiento(3, 15, 1, 101, 135);
  movimiento(2, 15, 1, 10, 65);
  digitalWrite(pass, HIGH);
  delay(10);
  digitalWrite(check, HIGH);
  s1.detach();
  s2.detach();
  s3.detach();
  s4.detach();
}
void brazo_robotico_e() {
  movimiento(1, 15, 1, 15, 80);
  movimiento(2, 30, 1, 25, 57);
  movimiento(1, 15, 1, 80, 0);
  movimiento(3, 15, 1, 106, 125);
  delay(500);
  movimiento(2, 20, 1, 57, 0);
  delay(1000);
  movimiento(4, 15, 1, 133, 0);
  movimiento(3, 15, 1, 125, 101);
  movimiento(2, 30, 1, 0, 55);
  delay(1000);
  movimiento(1, 15, 5, 0, 80);
  movimiento(2, 15, 1, 55, 10);
  movimiento(4, 15, 1, 0, 133);
  movimiento(3, 15, 1, 101, 135);
  movimiento(2, 15, 1, 10, 65);
  digitalWrite(pass, HIGH);
  delay(10);
  digitalWrite(check, HIGH);
  s1.detach();
  s2.detach();
  s3.detach();
  s4.detach();
}
```

```
void movimiento(int servo, int vel_servo, int pasos, int pos_ini, int
pos_fin) {
  switch (servo) {
    case 1:
      if (pos_ini < pos_fin) {
        for (pos = pos_ini; pos <= pos_fin; pos += pasos) {
          s1.write(pos);
          delay(vel_servo);
        }
      }
      else {
        for (pos = pos_ini; pos >= pos_fin; pos -= pasos) {
          s1.write(pos);
          delay(vel_servo);
        }
      }
      break;
    case 2:
      if (pos_ini < pos_fin) {
        for (pos = pos_ini; pos <= pos_fin; pos += pasos) {
          s2.write(pos);
          delay(vel_servo);
        }
      }
      else {
        for (pos = pos_ini; pos >= pos_fin; pos -= pasos) {
          s2.write(pos);
          delay(vel_servo);
        }
      }
  break;
```

```
    case 3:
      if (pos_ini < pos_fin) {
        for (pos = pos_ini; pos <= pos_fin; pos += pasos) {
          s3.write(pos);
          delay(vel_servo);
        }
      }
      else {
        for (pos = pos_ini; pos >= pos_fin; pos -= pasos) {
          s3.write(pos);
          delay(vel_servo);
        }
      }
      break;
    case 4:
      if (pos_ini < pos_fin) {
        for (pos = pos_ini; pos <= pos_fin; pos += pasos) {
          s4.write(pos);
          delay(vel_servo);
        }
      }
      else {
        for (pos = pos_ini; pos >= pos_fin; pos -= pasos) {
          s4.write(pos);
          delay(vel_servo);
        }
      }
      break;
  }
}
void recibir_etapa4() {
  detachInterrupt(0);
  Serial.println("ACCESO -->");
  inicio = 1;
}
```

Implementación del protocolo ModBus RTU para el monitoreo de módulo didáctico

- Mediante las siguientes funciones se configura la comunicación:

```
void configure_mb_slave(long baud, char parity, char txenpin);
int update_mb_slave(unsigned char slave, int *regs,
                    unsigned int regs_size);
enum {
  COMM_BPS = 19200, /* baud rate */
  MB_SLAVE = 2,/* modbus slave id */
  PARITY = 'n' /* even parity */
};

/* slave registers */
enum {
  MB_S2,
  MB_NIVEL,
  MB_SP,
  MB_CV,
  MB_PV,
  MB_REGS               /* número total de registros */
};
```

- La actualización de los registros de escritura como le lectura se realiza mediante la siguiente función.

- El la función Void Setup del código se configura los parámetros anteriormente establecidos:

```
void setup() {

  //Serial.begin(9600);
  configure_mb_slave(COMM_BPS, PARITY, 0);
  pinMode(respuesta_etapa2, OUTPUT);
  pinMode(luz_piloto_etapa2, OUTPUT);
  pinMode(sensor_etapa2, INPUT_PULLUP);
  digitalWrite(respuesta_etapa2, HIGH);
  digitalWrite(luz_piloto_etapa2, HIGH);
  inicio =  2;
  acceso = 0;
  s2 = 0;
  nivel_anterior = 0;
  total = 0;
  promedio = 0;
  regs[MB_SP] = 11;
  update_mb_slave(MB_SLAVE, regs, MB_REGS);
```

- Y mediante la función update_mb_slave(MB_SLAVE, regs, MB_REGS)

HMI construido en el Software Monitoriza for Arduino de Acimut-Scada

Gracias al proyecto de Juan Pablo Zometa de implementación del protocolo ModBus para la placa Arduino nos ha sido posible diseñar una versión especifica de nuestro SCADA Acimut Monitoriza.

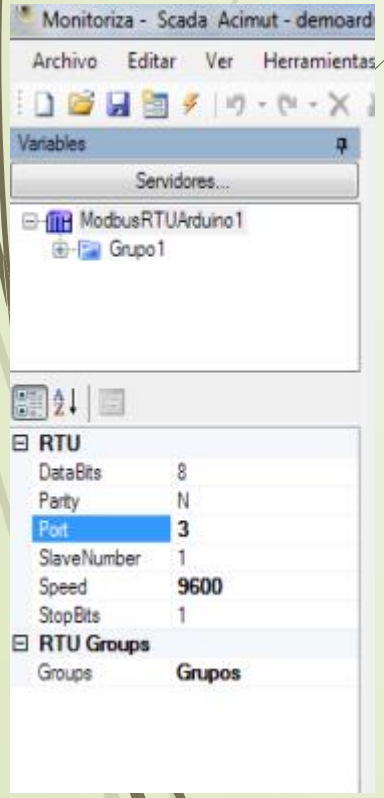
Esta versión de "Acimut Monitoriza for Arduino" es totalmente funcional y libre de todo tipo de restricciones de uso, tanto en cuanto número de variables a controlar como de clientes-puestos de monitorización que la versión comercial si que tiene. La única limitación que tiene es que solo se puede conectar a dispositivos Arduino.

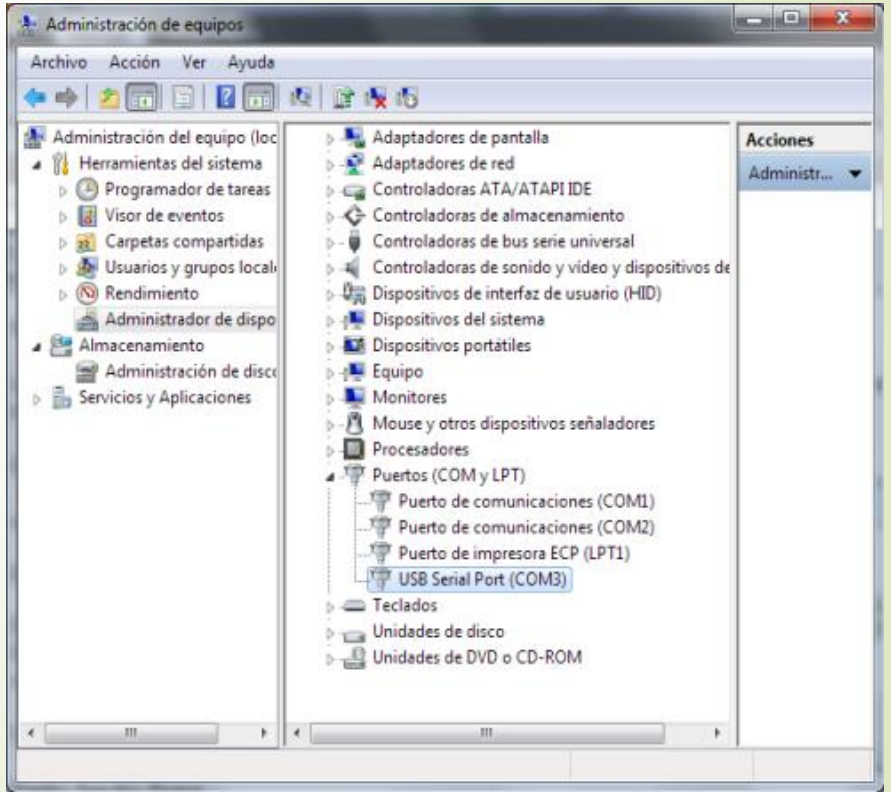
Características

- Instalación sencilla e inmediata del producto.
- Fácil configuración, incluso cuando se trata de una instalación con puestos remotos (WAN) ya que las comunicaciones entre los equipos cliente y el servidor se basan en los estándares de internet (protocolo HTTP).
- No precisa programación para la creación de proyectos completamente funcionales, basta "pinchar y arrastrar" los objetos SCADA sobre la superficie de los formularios y establecer las propiedades correspondientes para obtener una solución operativa.
- Si se requiere una funcionalidad avanzada que no esté contemplada en los objetos SCADA definidos en Monitoriza no hay problema ya que Monitoriza es extensible mediante programación en C# o VB.Net. También es posible la utilización de librerías de terceros desarrolladas para el .NET Framework de Windows.
- La creación de la interfaz gráfica de usuarios está basada en la tecnología de Windows Forms Designer de Microsoft© lo que facilita de forma notable el diseño.
- Fácil creación de gráficas para representar la tendencia de las variables
- A nivel de proyecto podemos definir usuarios y los permisos asignados a cada uno ellos. Por ejemplo si solo se tiene permiso de lectura en un determinado formulario o si se tiene acceso total a este.
- Definición inmediata de alarmas.
- Fácil seguimiento de variables.
- Datos en formatos accesibles. Monitoriza permite almacenar las variables que se monitorizan en bases de datos estándar del mercado (Microsoft© SQL Server™, Microsoft© Access™, Oracle®, etc.)

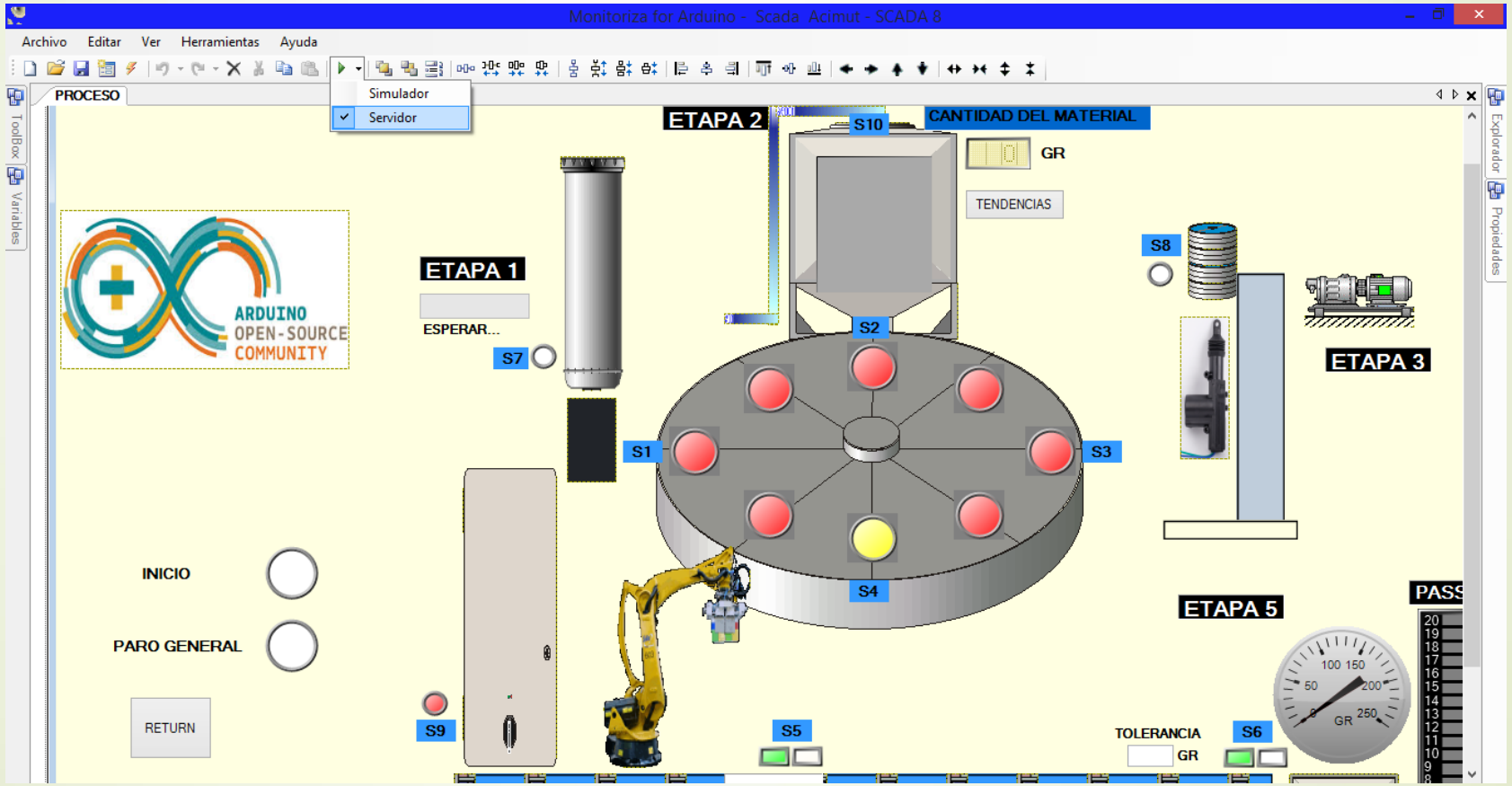
Comprobaremos el puerto de comunicaciones COMM asignado al cable USB del Arduino mediante el Administrador de equipos de Windows. Y en el proyecto Scada estableceremos en la propiedad port el número de puerto que tengamos asignado para el cable USB. Tambien es importante tener en cuenta la velocidad de comunicaciones (propiedad Speed) está deberá ser la misma que la establecida en el sketchbook

```
/* Modbus RTU common parameters, the Master MUST use the same parameters */

enum {
        COMM_BPS = 9600, /* baud rate */
        MB_SLAVE = 1,     /* modbus slave id */
        PARITY = 'N' /* even parity */
};
```



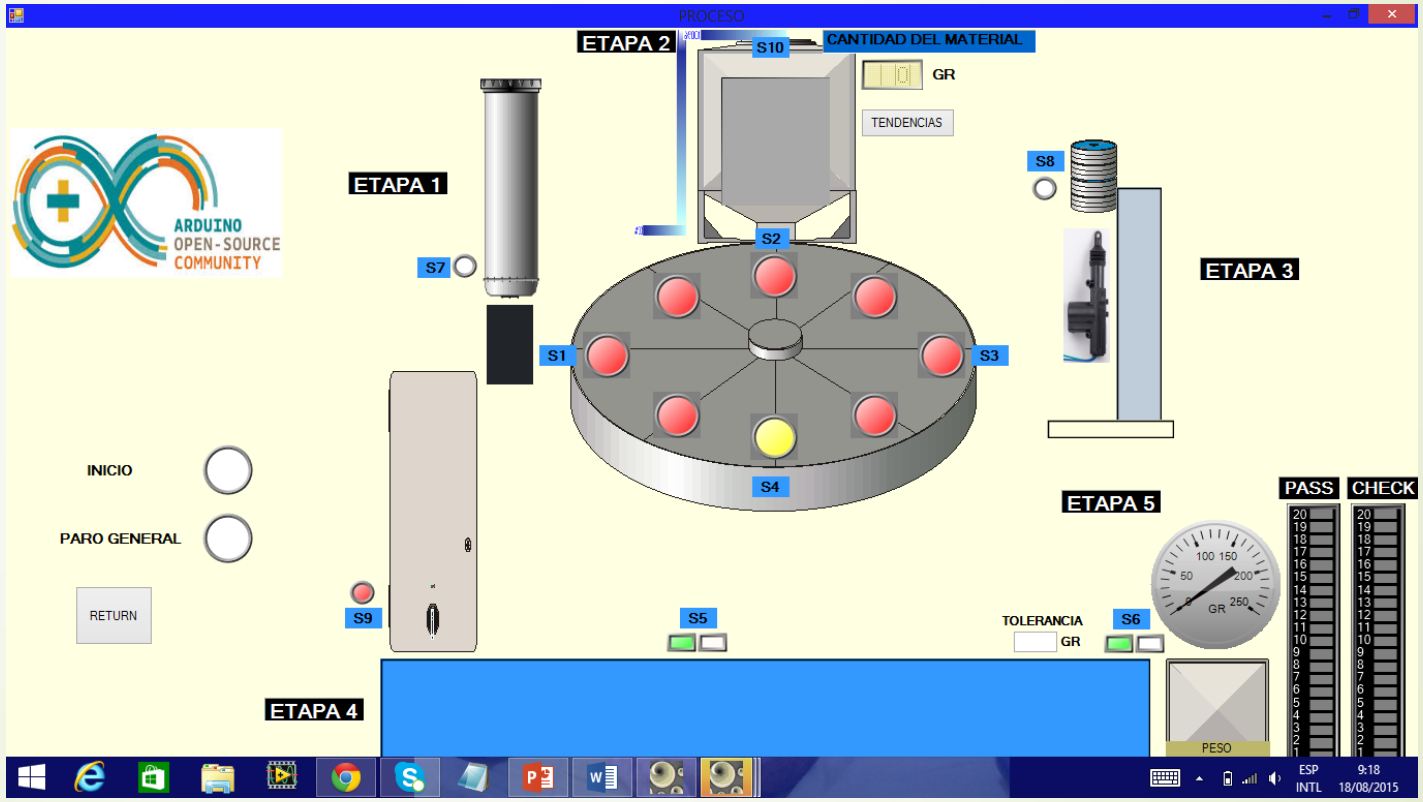


- A continuación simplemente tendremos que pulsar sobre el botón Play (con la opción de Servidor activada) para ejecutar el proyecto Scada y poder encender o apagar el led desde el scada.

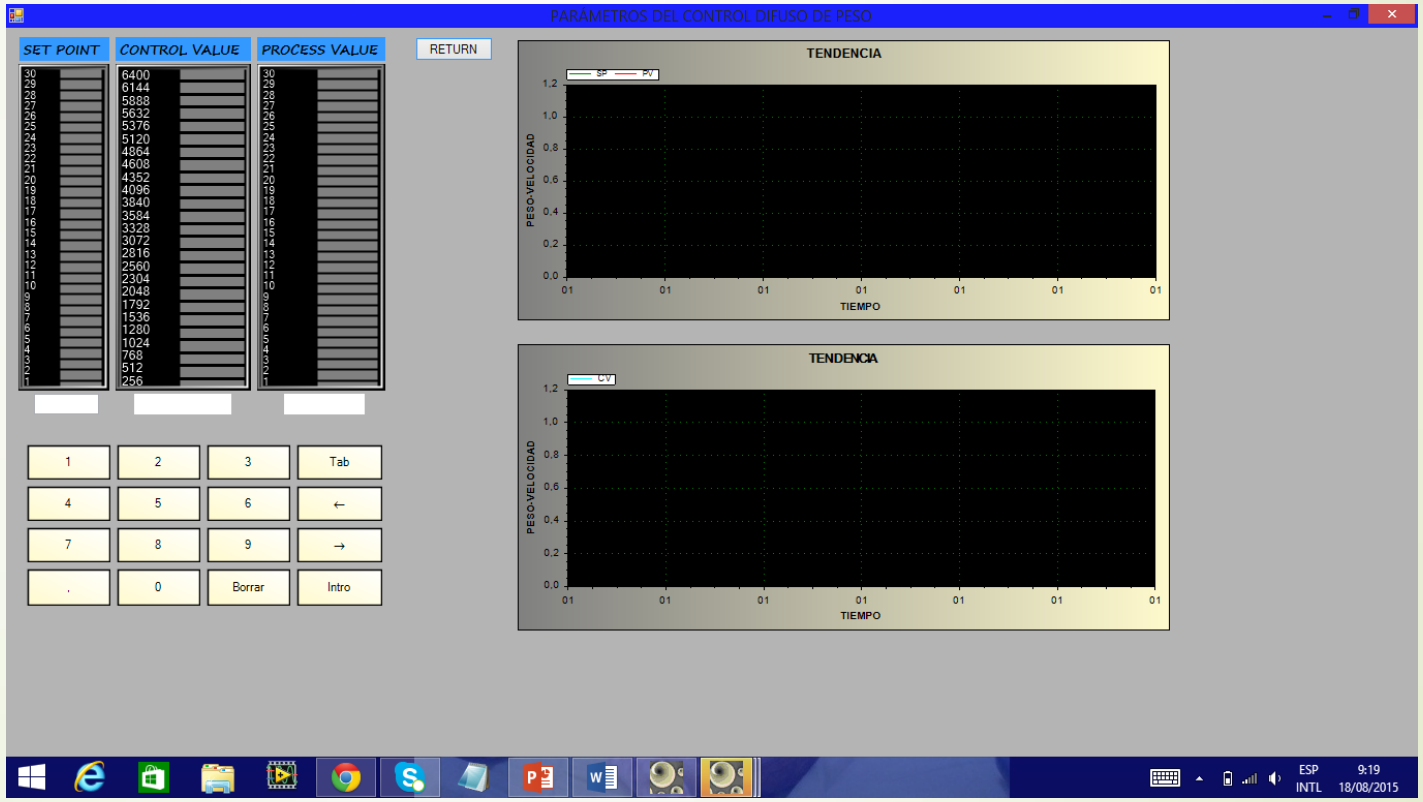


Pantalla principal del HMI

Proceso


PROCESO
ETAPA 2
S10
CANTIDAD DEL MATERIAL
GR
TENDENCIAS
ETAPA 1
S8
S7
S2
S1
S3
ETAPA 3
INICIO
PARO GENERAL
RETURN
S4
ETAPA 5
PASS
CHECK
S9
S5
TOLERANCIA
S6
GR
ETAPA 4
PESO
ESP
INTL
9:18
18/08/2015

Tendencias


PARÁMETROS DEL CONTROL DIFUSO DE PESO
SET POINT
CONTROL VALUE
PROCESS VALUE
RETURN
TENDENCIA
PESO-VELOCIDAD
TIEMPO
TENDENCIA
PESO-VELOCIDAD
TIEMPO
1
2
3
Tab
4
5
6
←
7
8
9
→
.
0
Borrar
Intro
ESP
INTL
9:19
18/08/2015

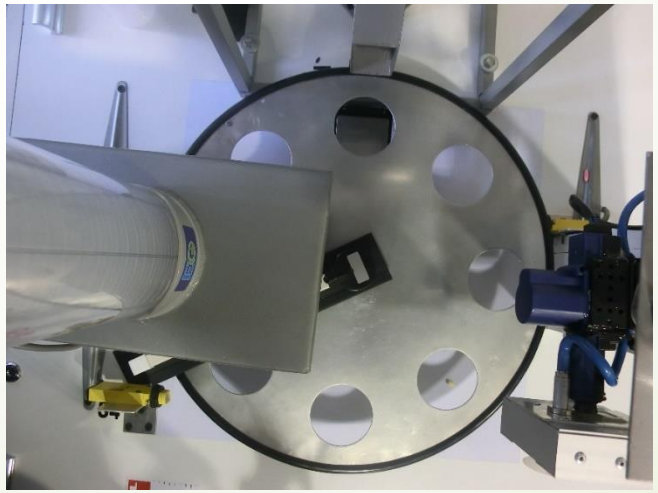
Pruebas y resultados

El correcto desempeño del módulo didáctico se verifica en la sincronización de los elementos como sensores, actuadores y tarjetas de control que conforman cada una de las etapas del módulo didáctico de automatización de llenado y envasado de sólidos, optimizando así los recursos y minimizando el tiempo de producción.

Para ello se determinó la importancia de realizar diferentes pruebas sobre los dispositivos de medida, comprobando algunos de los conceptos frecuentemente utilizados en la instrumentación.

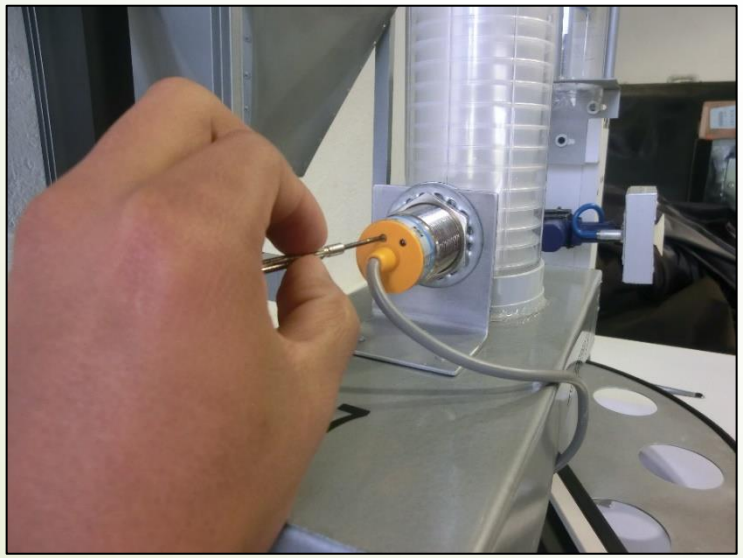
Funcionamiento y ajuste de la mesa giratoria

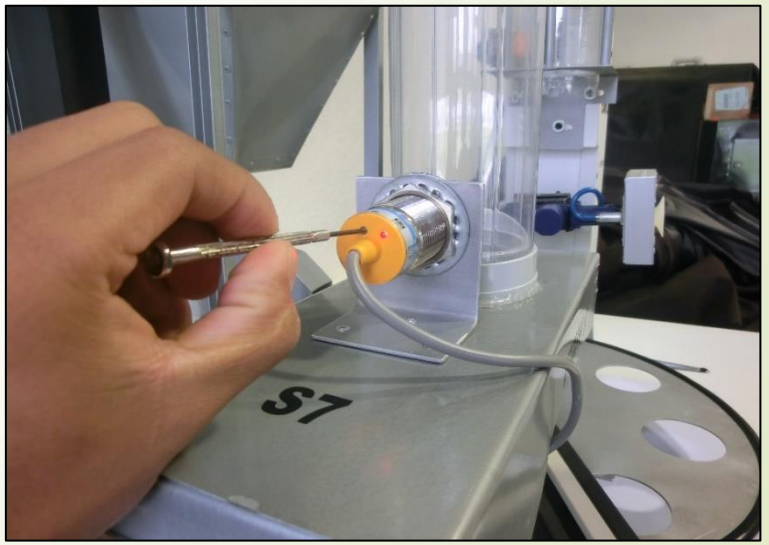
- Esta prueba se realiza a partir del sensor laser de la etapa 1 que detecta un cejillo ubicado en la mitad de una de las divisiones de la mesa giratoria, enviando una señal digital “FALLING” (Flanco de bajada), hacia la tarjeta de control de la Etapa 1, de tal manera que posiciona dicha mesa giratoria de forma que coincida con la ubicación de las etapas restantes.



Ajuste del sensor capacitivo ljc30a3-h-z/ay

- El sensor capacitivo fue sometido a pruebas de ajuste verificando que la distancia con respecto al material se encuentre en los límites establecidos en la hoja de datos del sensor, mediante el giro de un potenciómetro se ajusta la sensibilidad con respecto al material que se requiere detectar, hasta que la luz indicadora se apague.





Ajuste sensor láser QS30LDL

- El sensor laser QS30LDL fue sometido a pruebas de ajuste, acorde a los procedimientos detallados en las hojas de datos de dicho sensor, de la siguiente manera:
- Presionar el botón (-) ubicado en el sensor laser durante 2 segundos aproximadamente.
- Verificar que las luces indicadoras 5 y 6 se enciendan de forma alternada.
- Presionar el botón (-) durante un segundo y fijamos el primer límite de la ventana.
- Presionar el botón (-) durante un segundo y fijamos el segundo límite de la ventana.
- El sensor se dirigirá automáticamente a modo Run y estará ajustado para detectar el material que va a interceptar dicha ventana.

BOTÓN (-)

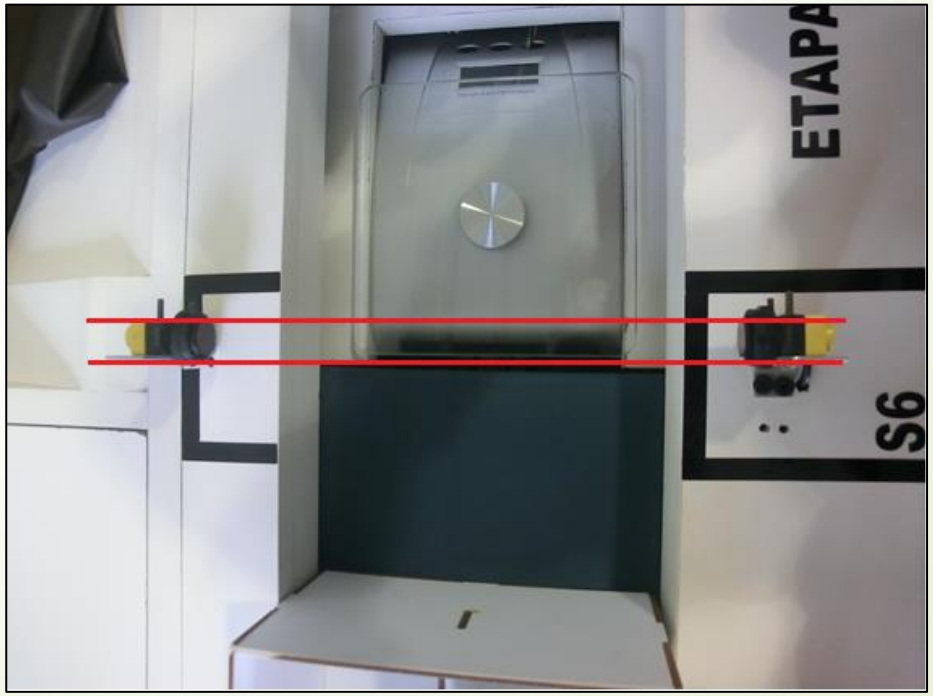
Ajuste del sensor MINI-BEAM SMU315D

- El sensor fotoeléctrico SMU315D fue sometido a pruebas de ajuste con respecto al material a detectar, acorde a los procedimientos detallados en las hojas de datos de dicho sensor, de la siguiente manera:
- El sensor consta de un potenciómetro ubicado en la parte posterior mediante el cual ajusta la sensibilidad de dicho sensor.
- Colocar el objeto frente al sensor y a la distancia establecida por el usuario para poder detectar dicho objeto y verificar que el LED de estado se encienda.
- Retirar el objeto a detectar verificar que el LED de estado se apague.
- Además consta de un selector ubicado en la parte posterior del sensor para cambiar el tipo de salida del sensor, normalmente cerrado (NC) o normalmente abierto (NO).

Punto de ajuste

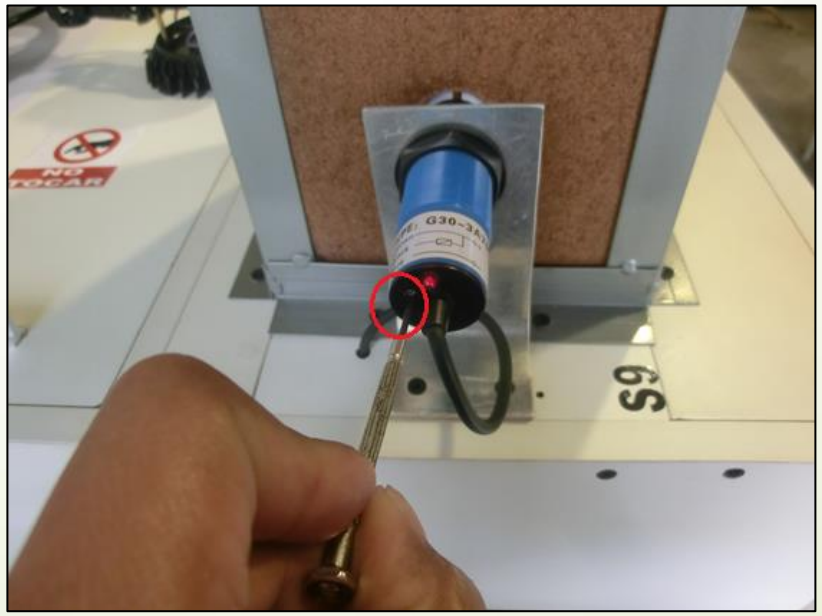
Ajuste del sensor QS18VP6R (RX) y el sensor QS18SE (TX)

- El par de sensores fotoeléctricos (RX Y TX) fueron sometidos a pruebas de ajuste colocándolos de tal forma que los dos puntos de la ventana queden en línea de vista para obtener un rendimiento óptimo.



Ajuste sensor fotoeléctrico G30-3A70NA

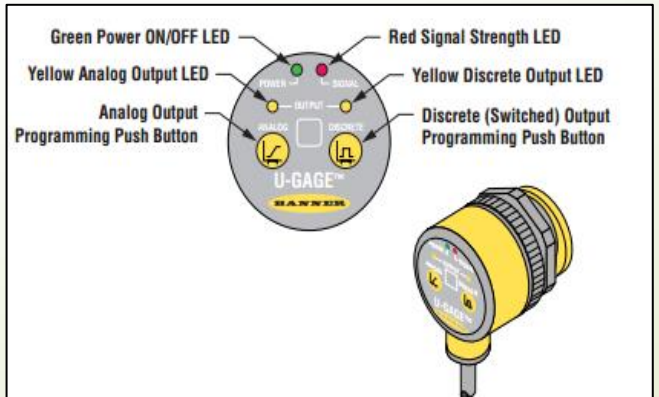
- El sensor fotoeléctrico G30-3A70NA fue sometido a pruebas de ajuste con respecto a la detección de la presencia o ausencia de cajas en el contenedor, mediante la variación de un potenciómetro ubicado en la parte posterior del sensor de acuerdo a la distancia que haya sido ubicado el sensor de las cajas.



Sensor ultrasónico U-GAGE U30

El sensor ultrasónico U-GAGE U30 fue sometido a pruebas de ajuste de acuerdo a los procedimientos detallados en la hoja de datos de dicho sensor, de la siguiente manera:

- Pulsar sobre el botón de programación de la salida analógica para establecer la primera condición, si no se establece la primera condición el sensor volverá al modo RUN dentro de 120 segundos.
- Pulsar nuevamente sobre el botón de programación de la salida analógica después de establecer el primer límite, el sensor permanecerá en modo PROGRAM hasta que finalice la secuencia TEACH.
- Mantener pulsado el botón de programación por 2 segundos aproximadamente para salir del modo PROGRAM



Accionamiento de alarmas

- Para verificar el correcto funcionamiento de las alarmas que existen en el módulo didáctico de un sistema de llenado y envasado de solidos se procede a vaciar los contenedores de recipientes, cubiertas, cajas y el material contenido en la tolva, observando la activación de las luces pilotos en el panel de control.

Etapa de inspección

- La etapa de inspección dispone de luces indicadoras para visualizar si acepta o rechaza el producto final, el cual se encuentra conformado por los pesos de los 4 recipientes con sus cubiertas, el peso de la caja y el peso neto vertido en cada recipiente.

HMI

- Se verifica el funcionamiento de la interfaz HMI en el software Monitoriza for Arduino Acimut–Scada, variando parámetros de configuración para la comunicación como el rango de actualización en milisegundos y se comprobó que los datos a visualizar no contenían errores pero al contrario la visualización de eventos, alarmas y tendencias se tornó lenta.
- Se identificó que al no actualizar los datos desde la tarjeta Arduino el software APCAutomation de Acimut Monitoriza for Arduino durante un tiempo de 5 segundos genera un error de comunicación, para ello se procedió a actualizar los datos de los registros enviados desde la tarjeta Arduino mediante el protocolo de comunicación ModBus RTU después de cada línea de código para garantizar la comunicación.
- Se realizan pruebas modificando valores de escritura en la etapa 2, como el set point de la cantidad a dispensar en gramos, de la misma forma trasladar este dato a la etapa 5 para poder actualizar el peso del producto final y verificar si es aprobada o no la inspección, la cual fue satisfactoria y se constató la correcta interacción de la interfaz HMI con los esclavos de la red ModBus RTU.

Conclusiones

- El diseño y construcción del módulo didáctico de un sistema de llenado y envasado de sólidos, permitió verificar el funcionamiento de un proceso industrial a pequeña escala, el cual será de gran ayuda para los estudiantes de la carrera de Ing. Electrónica e Instrumentación, incrementando los conocimientos prácticos y reforzando los conocimientos teóricos.
- La operación en paralelo de las etapas del módulo didáctico optimiza los recursos y aumenta la velocidad de producción, ya que estos son los principales parámetros a tomar en consideración en la automatización de un proceso industrial.
- El manejo de tarjetas Arduino en la automatización del proceso de llenado y envasado de sólidos no requiere de licencia alguna ya que está basado en una plataforma libre, pero para implementarlo a nivel industrial se necesita de acondicionamiento de hardware para evitar el mal funcionamiento de los dispositivos.
- El ruido inductivo generado por el accionamiento de relés provocó un mal funcionamiento en las distintas etapas del módulo didáctico, reseteando contadores y generando señales falsas en las entradas digitales de las tarjetas Arduino, por lo que se vio necesario implementar filtros RC paso bajos para eliminar las frecuencias altas generadas por dichos ruidos.
- El uso del protocolo de comunicación ModBus RTU en las tarjetas Arduino permitió recolectar información en tiempo real del proceso en funcionamiento del módulo didáctico, lo que fue de gran ayuda para monitorear el proceso y administrar eventos externos que suscitan, que serán manejados por el operador .

- La velocidad de producción alta, media o baja, está limitada por los actuadores que se encuentran en el proceso de llenado y envasado de sólidos.
- El módulo didáctico de llenado y envasado de sólidos se limita a envasar sólidos como los cereales, debido a la composición densidad y peso de dicho material, lo que conlleva a un óptimo desempeño en la etapa de transporte.
- El uso de dispositivos de medida y el correcto acondicionamiento de los mismos, acorde a las condiciones presentes en el módulo didáctico fue de gran importancia, ya que de las señales o respuestas a cambios producidas por los mismos, depende el sincronismo de cada etapa en el módulo didáctico.
- El uso de interrupciones generadas por eventos externos en las tarjetas Arduino, facilito e incremento la velocidad del proceso obteniendo la independencia de cada etapa en el módulo didáctico.
- El uso de brazos robóticos fue una limitante para utilizar diversos materiales sólidos para el llenado, debido a la inestabilidad de los mismos y falencia de torque en los servomotores que conforman los mismos.

Recomendaciones

- Se recomienda implementar y diseñar módulos didácticos de procesos industriales para incrementar los conocimientos de los estudiantes de la carrera Ing. Electrónica e Instrumentación, y así en momento de buscar una fuente laboral ganar experiencia en el manejo de dichos procesos.
- El implementar tarjetas electrónicas mucho más robustas aumenta la fiabilidad en el proceso, pero conlleva un gasto mayor, por esta razón se recomienda aislar la parte electrónica de la parte de potencia para obtener un rendimiento óptimo.
- El aprovechar la comunicación serie que dispone la tarjeta Arduino para utilizar el protocolo ModBus RTU fue de gran ayuda para la recolección de información, pero se pudo haber utilizado "shields ethernet" para obtener la información del proceso más rápido y poder añadir aplicaciones web para hacer más amigable el entorno con el operador, esto gracias a las diversos complementos que dispone la plataforma Arduino.

- La importancia de la implementación en el módulo didáctico de una etapa de inspección es de gran relevancia, debido a que se asemeja a la parte de comprobación de la mercadería antes de salir a la venta en una empresa, por lo que se clasifica los productos aceptados y rechazados para su expendio.
- El sobredimensionar parámetros como torque en servomotores es de gran importancia para procesos que manejan diversos materiales que van a ser procesados, en este caso el torque de los servomotores que constituyen los brazos robóticos no generaron el torque suficiente para transportar el producto, lo cual se solucionó sustituyendo dichos servomotores por unos de mayor torque.
- El realizar este tipo de proyectos en software libre abre una gran puerta de ideas a ser implementadas, se recomienda aportar a esta comunidad para que muchas más ideas como está se vean plasmadas a precios módicos y sean de gran ayuda para los estudiantes acorde al tema.