Manual de Programación

Trabajo de Tesis – Ingeniería Mecatrónica

Autores: Neil Sebastian Castro Caicedo – Hubert Armando Delgado Maestre

1. Introducción

En este manual se describe el procedimiento básico, paso a paso, para desarrollar un programa exitoso utilizando la plataforma Ilyzaelle. Además, se presentan ejemplos de códigos fundamentales para gestionar diferentes entradas y salidas del controlador Xelorium, aplicados a pequeños ejemplos.

2. Configurar paso a paso con Xelorium.

Lo primero que tenemos que tener en cuenta es que el controlador xelorium, tiene salidas y entradas.

Al estar basado en Arduino, las diferentes salidas y entradas, están relacionadas a un pin del Arduino mega, para esto dependiendo al entrada o salida, que se usara se debe relacionar con los pines de la tabla:

ENTRADAS DIGITALES	
PIN ARDUINO	USO PLC
22	X1.0
24	X1.1
26	X1.2
28	X1.3
30	X1.4
32	X1.5

ENTRADAS ANALOGICAS	
PIN ARDUINO	USO PLC
A0	I0.0
A1	I0.1
A2	I0.2
A3	I0.3
A4	I0.4
A5	I0.5
A6	I0.6
A7	I0.7

SALIDAS MOTOR	
PIN ARDUINO	USO PLC
2	QW0
23	QX1.0
25	QX1.1
3	QW1
27	QX1.2
29	QX1.3
5	QW3
31	QX1.4
33	QX1.5
6	QW4
35	QX1.6
37	QX1.7

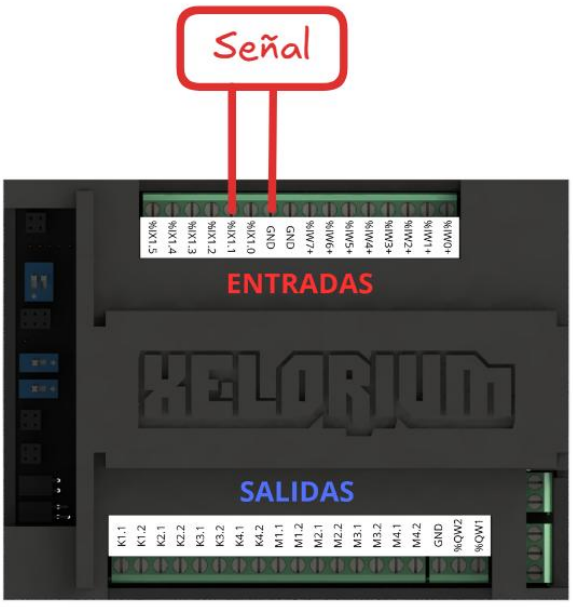
SALIDA ANALOGICAS	
PIN ARDUINO	USO PLC
4	QW2
13	QW11
SALIDA RELÉ	
PIN ARDUINO	USO PLC
39	QX2.0
41	QX2.1
43	QX2.2
45	QX2.3

3. Programación Básica:

Entrada digital

Montaje y configuración:

Decidimos que entrada digital usaremos, en este caso X1.1 buscamos por medio de la tabla su igual en el pin Arduino.



ENTRADAS DIGITALES	
PIN ARDUINO	USO PLC
22	X1.0
24	X1.1
26	X1.2
28	X1.3
30	X1.4
32	X1.5

Código en el editor de la plataforma:

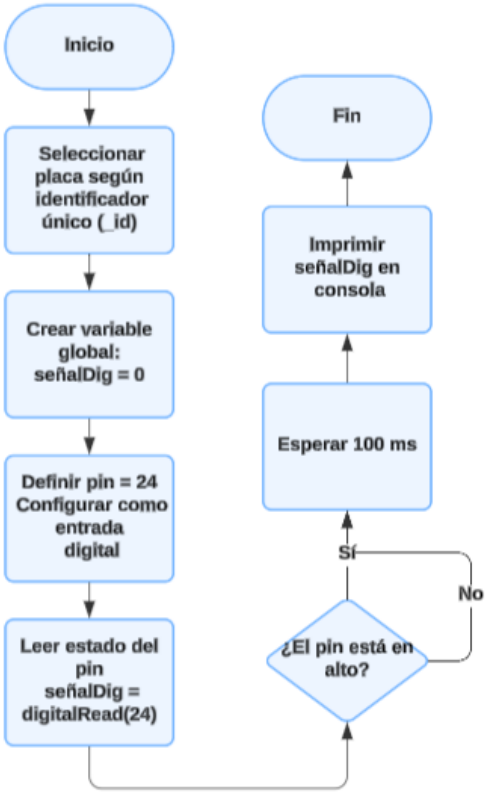
Al tener su igual en el pin Arduino en este caso X1.1 es 24, procedemos a configurarlo en nuestro código.

```
const board = boards[_id];
// DECLARACION VARIABLE GLOBAL
gVar[project].señalDig = 0
// DECLARACION PINES ARDUINO
const señalDigital = 24;
// DECLARACION TIPO DE PIN
//Sensores
board.pinMode(señalDigital, board.MODES.INPUT);    //Entrada Digital
//LECTURA DEL PIN Y PONER EL VALOR EN VARIABLE GLOBAL
board.digitalRead(señalDigital, (value) => {
  gVar[project].señalDig = value
});
// CODIGO GENERAL
setInterval(() => {
  console.log(gVar[project].señalDig)
}, 100, { _id });
```

Resumen:

El código primero selecciona la placa que se va a usar a partir de un identificador único (_id). Luego crea una variable global llamada señalDig dentro del proyecto, inicializada en 0, la cual servirá para guardar el valor leído de un pin. Después se define que el pin digital que se va a usar es el número 24 y se configura como una entrada digital, es decir, solo podrá leer señales externas (0 o 1). A continuación, se establece la lectura del pin con digitalRead: cada vez que el pin cambie de estado, el valor leído se guarda automáticamente en la variable global señalDig (0 si está en bajo o 1 si está en alto). Finalmente, mediante un intervalo de 100 milisegundos, se imprime en la consola el valor de esa variable, lo que permite monitorear en tiempo real el estado del pin 32 desde la aplicación.

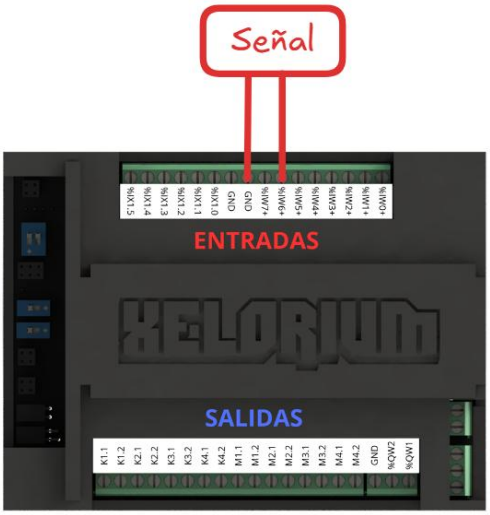
Diagrama Flujo:



Entrada Análoga

Montaje y configuración:

Decidimos que entrada análoga usaremos, en este caso IW6 buscamos por medio de la tabla su igual en el pin Arduino.



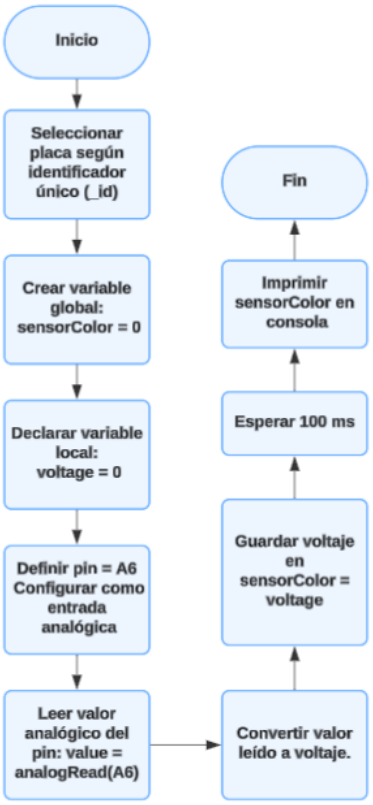
ENTRADAS ANALOGICAS		
PIN ARDUINO	USO PLC	PIN OPEN PLC
A0	I0.0	IW0
A1	I0.1	IW1
A2	I0.2	IW2
A3	I0.3	IW3
A4	I0.4	IW4
A5	I0.5	IW5
A6	I0.6	IW6
A7	I0.7	IW7

Al tener su igual en el pin Arduino en este caso IW6 es A6, procedemos a configurarlo en nuestro código.

```
const board = boards[_id];
//VARIABLES GLOBALES
gVar[project].sensorColor = 0
// VARIABLES LOCALES
let voltage = 0
// DECLARACION PINES ARDUINO
const sensorColor = A6;
// DECLARACION TIPO DE PIN
//Sensores
board.pinMode(sensorColor, board.MODES.ANALOG);    //Entrada Analoga
// CODIGO LECTURA PINES
//sensores
board.analogRead(sensorColor, (value) => {
  voltage = (value.toFixed(2) / 1023) * 10; // Conversión a rango 0-5V
  gVar[project].sensorColor = voltage;
});
setInterval(() => {
  console.log(gVar[project].sensorColor)
}, 100, { _id });
```

El código empieza seleccionando la placa correspondiente usando el identificador _id. Luego crea una variable global sensorColor inicializada en 0, que servirá para guardar la lectura de un sensor. También define una variable local llamada voltage, usada para almacenar temporalmente el valor convertido de la señal analógica. A continuación, se declara el pin analógico A6 como el que se va a usar y se configura como entrada analógica, lo que permite leer valores entre 0 y 1023 en lugar de solo 0 o 1. Después, mediante analogRead, cada vez que el pin recibe una lectura, ese valor se convierte a voltaje en un rango aproximado de 0 a 10 V con la fórmula (value.toFixed(2) / 1023) * 10, y el resultado se guarda en la variable global sensorColor. Finalmente, cada 100 ms se imprime en la consola el valor actualizado de esa variable, lo que permite visualizar en tiempo real el voltaje leído del sensor conectado al pin A6.

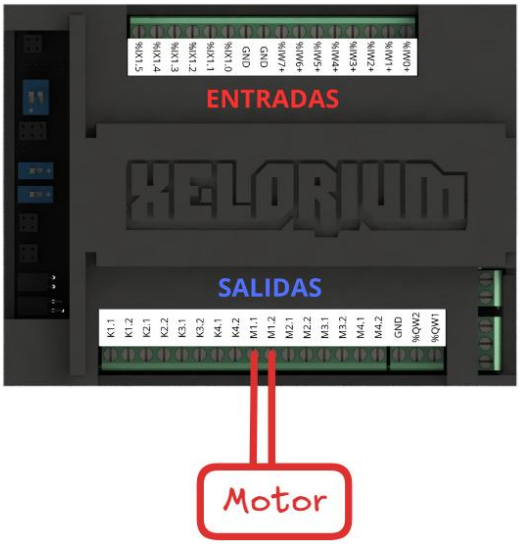
Diagrama de flujo:



Salida PWM

Montaje y configuración:

Decidimos que entrada de motores M usaremos, en este caso M1.1 y M1.2 buscamos por medio de la tabla su igual en el pin Arduino.



SALIDAS MOTOR		
PIN ARDUINO	USO PLC	
2	QW0	
23	QX1.0	
25	QX1.1	M1
3	QW1	
27	QX1.2	
29	QX1.3	M2
5	QW3	
31	QX1.4	
33	QX1.5	M3
6	QW4	
35	QX1.6	
37	QX1.7	M4

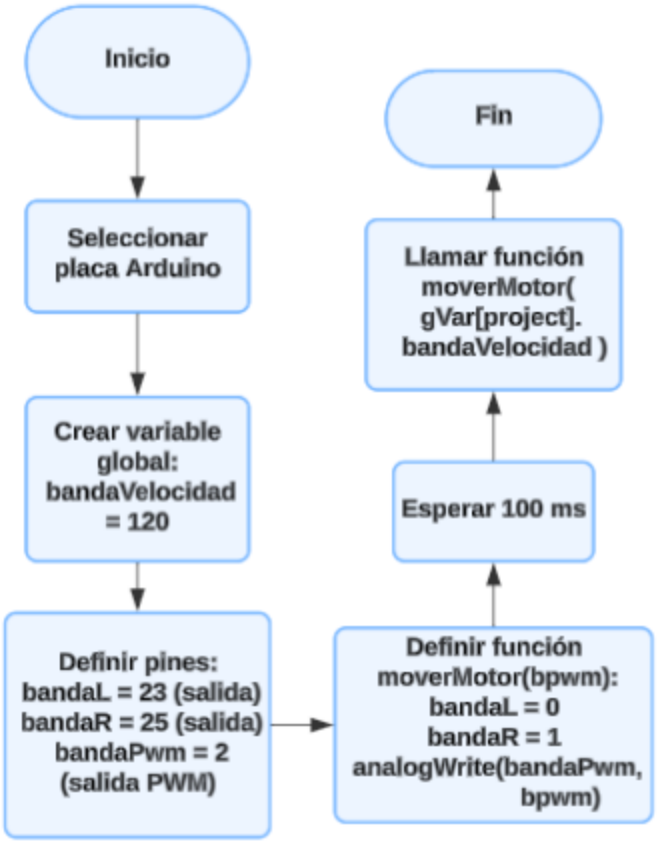
Al tener su igual en el pin Arduino en este caso M1.1 es 23, M1.2 es 25 y PWM es 2, procedemos a configurarlo en nuestro código.

```
const board = boards[_id];
//VARIABLES GLOBALES
gVar[project].bandaVelocidad = 120

// DECLARACION PINES ARDUINO
const bandaL = 23;
const bandaR = 25;
const bandaPwm = 2;

// DECLARACION TIPO DE PIN
//motor1
board.pinMode(bandaL, board.MODES.OUTPUT);   //Salida Digital
board.pinMode(bandaR, board.MODES.OUTPUT);
board.pinMode(bandaPwm, board.MODES.PWM);    //Salida Analoga PWM
// CODIGO LECTURA PINES
function moverMotor(bpwm) {
  board.digitalWrite(bandaL, 0);
  board.digitalWrite(bandaR, 1);
  board.analogWrite(bandaPwm, bpwm);
  // Código para mover el motor
}

//CODIGO PRINCIPAL
setInterval(() => {
  moverMotor(gVar[project].bandaVelocidad) //por medio de la variable global controlamos la velocidad el pwm va de 0-255
}, 100, { _id });
```

Resumen:

El código comienza seleccionando la placa Arduino y declarando una variable global llamada bandaVelocidad, inicializada en 120, que controlará la rapidez del motor. Luego se definen tres pines: bandaL (23) y bandaR (25) como salidas digitales para manejar la dirección del motor, y bandaPwm (2) como salida analógica PWM para regular la velocidad. Después se crea la función moverMotor(bpwm), que fija la dirección del motor poniendo bandaL en 0 y bandaR en 1, y al mismo tiempo aplica un valor PWM al pin bandaPwm, lo que determina cuán rápido gira el motor. Finalmente, en el bloque principal, cada 100 ms se llama a moverMotor usando el valor de gVar[project].bandaVelocidad, de manera que modificando esta variable global se puede controlar dinámicamente la velocidad del motor en un rango de 0 a 255.

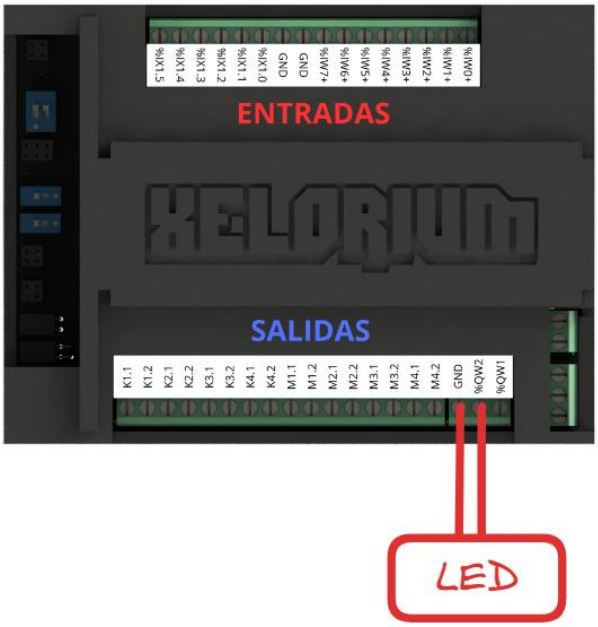
Diagrama de flujo:



Salida Análoga

Montaje y configuración:

Decidimos que salida análoga usaremos, en este caso QW2 buscamos por medio de la tabla su igual en el pin Arduino.



SALIDA ANALOGICAS	
PIN ARDUINO	USO PLC
4	QW2
13	QW11

Al tener su igual en el pin Arduino en este caso QW2 es 4, procedemos a configurarlo en nuestro código.

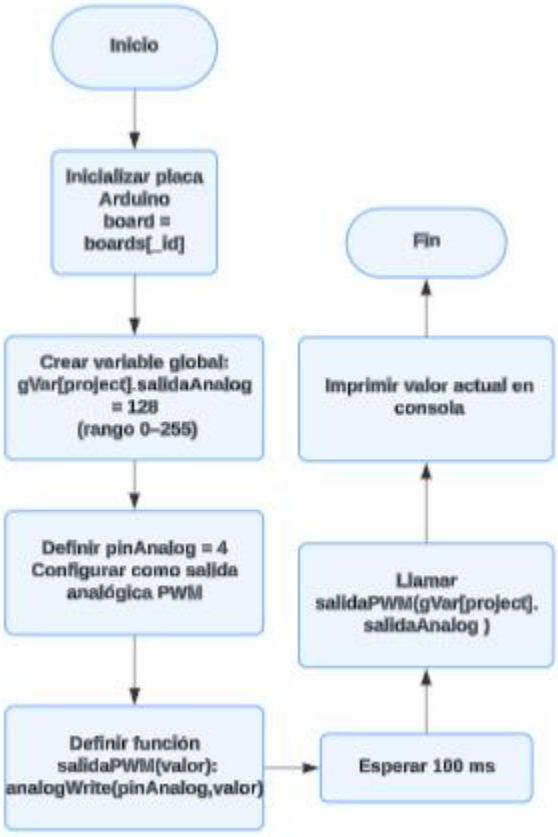
```
const board = boards[_id];
// VARIABLE GLOBAL
gVar[project].salidaAnalog = 128   // Valor inicial (0-255)

// DECLARACION PIN ARDUINO
const pinAnalog = 4;   // Pin PWM (ejemplo)

// DECLARACION TIPO DE PIN
board.pinMode(pinAnalog, board.MODES.PWM);   // Salida Analógica (PWM)

// FUNCION PARA ENVIAR VALOR PWM
function salidaPWM(valor) {
  board.analogWrite(pinAnalog, valor);
}

// CODIGO PRINCIPAL
setInterval(() => {
  salidaPWM(gVar[project].salidaAnalog);
  console.log("Valor PWM:", gVar[project].salidaAnalog);
}, 100, { _id });
```

Resumen:

El código comienza inicializando la placa Arduino con const board = boards[_id];. Después se declara una variable global gVar[project].salidaAnalog con un valor inicial de 128, que

corresponde a la intensidad de la señal PWM en un rango de 0 a 255. Luego se define el pin 4 como la salida analógica (PWM) mediante board.pinMode(pinAnalog, board.MODES.PWM);. A continuación, se crea la función salidaPWM(valor), que se encarga de enviar el valor de PWM al pin configurado. En la parte principal del programa, mediante un setInterval que se ejecuta cada 100 milisegundos, se llama a la función para escribir en el pin el valor guardado en la variable global y, además, se imprime en consola el valor actual de la salida. Con esto, el sistema permite controlar dinámicamente la intensidad de una salida analógica simulada en Arduino (PWM), por ejemplo, para regular la velocidad de un motor o la intensidad de un LED, simplemente modificando el valor de la variable global.

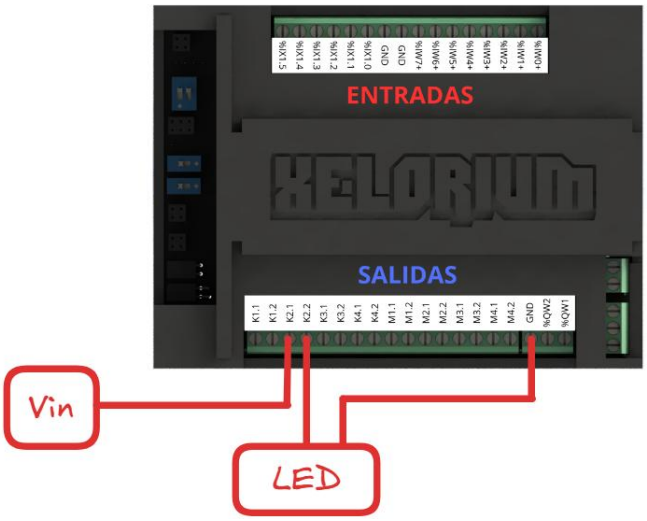
Diagrama de flujo:



Salida Digitales o Relé

Montaje y configuración:

Decidimos que salida digital o relé usaremos, en este caso K2.2 buscamos por medio de la tabla su igual en el pin Arduino.



SALIDA RELÉ		
PIN ARDUINO	USO PLC	
39	QX2.0	
41	QX2.1	
43	QX2.2	
45	QX2.3	

Al tener su igual en el pin Arduino en este caso K2.2 es 41, procedemos a configurarlo en nuestro código.

```
const board = boards[_id];
//VARIABLES GLOBALES
gVar[project].releAspirarBrazo = false

// DECLARACION PINES ARDUINO
//Reles
const releAspirar = 41;
// DECLARACION TIPO DE PIN
//Reles
board.pinMode(releAspirar, board.MODES.OUTPUT);
// CODIGO LECTURA PINES
function relesA(rAspira) {
  board.digitalWrite(releAspirar, rAspira);
}

//CODIGO PRINCIPAL
  if (gVar[project].releAspirarBrazo === true){
    relesA(1)
  }else {
    relesA(0)
  }
}, 100, { _id });
```

Resumen:

El código empieza inicializando la placa Arduino con const board = boards[_id];. Después se declara una variable global gVar[project].releAspirarBrazo, inicialmente en false, que servirá para controlar el estado de un relé. Luego se define el pin 39 como salida digital mediante board.pinMode(releAspirar, board.MODES.OUTPUT);. A continuación, se crea la función relesA(rAspira), que recibe un valor (0 o 1) y lo envía al pin configurado, activando o desactivando el relé. En la parte principal del código se comprueba el valor de la variable global: si está en true, la función activa el relé (relesA(1)), y si está en false, lo desactiva (relesA(0)). Finalmente, el bloque se ejecuta cada 100 ms gracias al setInterval, permitiendo controlar de forma continua el encendido o apagado del relé en función de la variable global.

Diagrama de flujo: