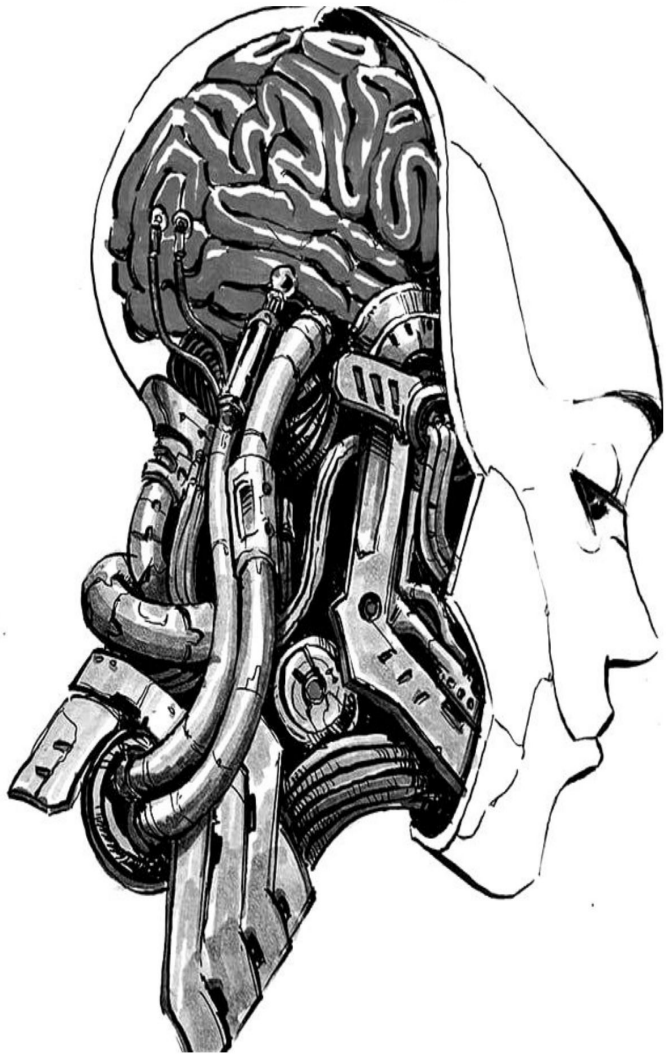
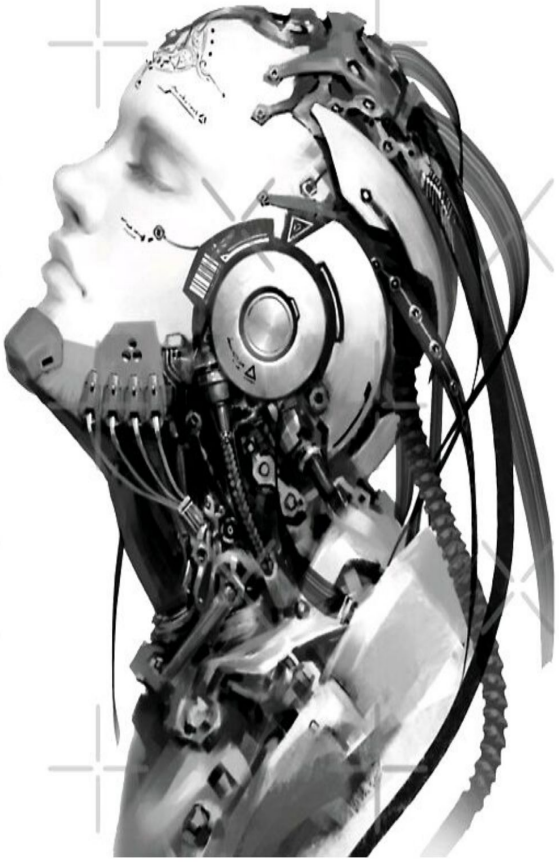
Automatización y control en el entorno Arduino I

Universidad Nacional de San Cristóbal de Huamanga

JOSE LUIS HUAYANAY VILLAR





To dedicado a

A mi familia y colegas investigadores.

Author's Note: Jose luis huayanay villar es un profesional e investigador egresado de la facultad de ciencias fisico y matematicas-UNSCH, con estudios de maestria en Automatización y control (UERJ-Brasil) y un Doctorado en Mecanica de control espacial (INPE-UNSCH). Con múltiples artículos científicos publicados.

Publisher: UNSCH
First edition, published in 2023, Volumen I.
Agradecer a todos los lectores.

ISBN: 1708417846

Conteúdo

Capítulo 1

Automatización en el entorno Arduino

La automatización consiste en usar la tecnología para realizar tareas con muy poca intervención humana. Se puede implementar en cualquier sector en el que se lleven a cabo tareas repetitivas o cíclicas, reduciendo la cantidad de trabajo manual. Existe componentes electrónicos de acceso libre que permiten automatizar un determinado sistema. un ejemplo de ellos es el Arduino, que permite automatizar cualquier objeto o sistema con ellos; conectar con otros dispositivos, interactuar con otros programas, desarrollar elementos autónomos, etc (figura 1.1). Las placas Arduino pueden dar vida a robots, mandos a distancia, móviles, consolas portátiles, cámaras de foto y mucho más.

1.1 Sistemas de control en el entorno Arduino

Un sistema de control es un conjunto de dispositivos encargados de administrar, ordenar, dirigir o regular el comportamiento de otro sistema, con el fin de obtener los resultados deseados. En un sistema de control pueden identificarse 4 partes:

- Medición, Acción, Control y Referencia.

Todo sistema de control mide y hace algo, y el proceso lo realiza el controlador que utiliza programas y referencias (Set-point). Las referencias o parámetros son la parte ajustable del Software; son intangibles, no se pueden tocar, y necesitan un soporte físico para memo-rizarse; y las Mediciones, Accionamientos y Controladores son el Hardware que se pueden ver y tocar. Por nuestra naturaleza humana es útil mencionar por un lado el control manual y por otro el control automático. Los problemas considerados en la ingeniería de los sistemas de control, básicamente se tratan mediante dos pasos

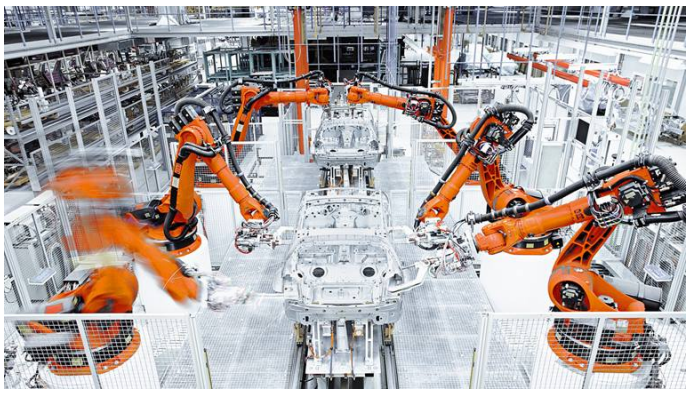


Figura 1.1: Sistemas de automatización en plantas industriales (fuente:www.kuka.com)

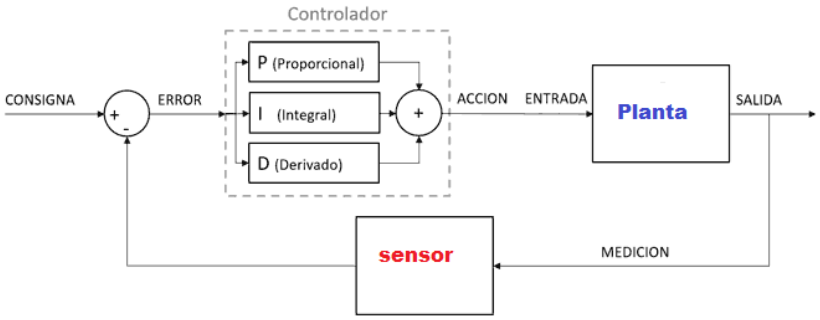


Figura 1.2: Ejemplo de un controlador proporcional, Integral y derivativo(PID) (fuente: Autor)

fundamentales como son: Análisis y diseño. La representación de los problemas en los sistemas de control se lleva a cabo mediante tres representaciones básicas o modelos:

- Ecuaciones diferenciales e integrales
- derivadas y otras relaciones matemáticas.

Los diagramas en bloque y las gráficas de flujo son representaciones gráficas que pretenden el acortamiento del proceso correctivo del sistema, sin importar si está caracterizado de manera esquemática o mediante ecuaciones matemáticas. Las ecuaciones diferenciales y otras relaciones matemáticas, se emplean cuando se requieren relaciones detalladas del sistema. Cada sistema de control se puede representar teóricamente por sus ecuaciones matemáticas. El uso de operaciones matemáticas es patente en todos los controladores de tipo Controlador proporcional (P), Controlador proporcional,Integral (PI) y Controlador proporcional,Integral y derivativo(PID) (figura 1.2), que debido a la combinación y superposición de cálculos matemáticos ayuda a controlar circuitos, montajes y sistemas industriales para así ayudar en el perfeccionamiento de los mismos.

Capítulo 2

Tipos de sistemas control Automático

2.1 Hechos por el hombre

Como los sistemas eléctricos o electrónicos que están permanentemente capturando señales del estado del sistema bajo su control y que al detectar una desviación de los parámetros preestablecidos del funcionamiento normal del sistema (figura 2.1), actúan mediante sensores (ejemplo, temperatura, presión, magnetismo etc.) y actuadores, para llevar al sistema de vuelta a sus condiciones operacionales normales de funcionamiento. Un claro ejemplo de este será un termostato, el cual capta consecutivamente señales de temperatura. En el momento en que la temperatura desciende o aumenta y sale del rango, este actúa encendiendo un sistema de refrigeración o de calefacción.

- Por su causalidad pueden ser: causales y no causales. Un sistema es causal si existe una relación de causalidad entre las salidas y las entradas del sistema, más explícitamente, entre la salida y los valores futuros de la entrada.
- Según el número de entradas y salidas del sistema, se denominan:por su comportamiento
  - De una entrada y una salida o SISO(single input, single output).
  - De una entrada y múltiples salidas o SIMO (single input, multiple output).
  - De múltiples entradas y una salida o MISO (multiple input, single output).
  - De múltiples entradas y múltiples salidas o MIMO (multiple input, multiple output).
  - Según la ecuación que define el sistema, se denomina:
  - Lineal, si la ecuación diferencial que lo define es lineal.

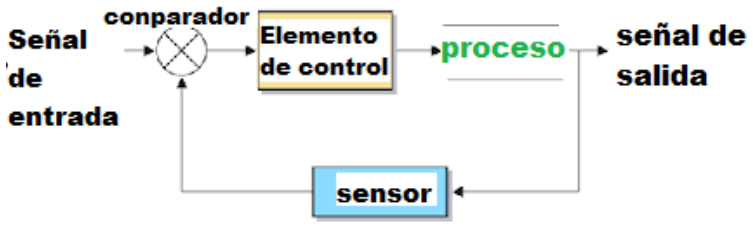


Figura 2.1: Sistema de control Automático (fuente: Autor)

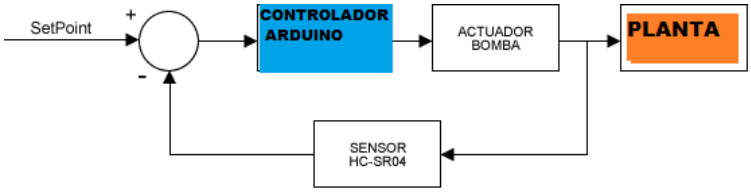


Figura 2.2: Ejemplo control tradicional en lazo cerrado en el entorno Arduino (fuente: Autor)

  - No lineal, si la ecuación diferencial que lo define es no lineal.

- Las señales o variables de los sistema dinámicos son función del tiempo. Y de acuerdo con ello estos sistemas son:
  - De tiempo continuo, si el modelo del sistema es una ecuación diferencial, y por tanto el tiempo se considera infinitamente divisible. Las variables de tiempo continuo se denominan también analógicas
  - De tiempo discreto, si el sistema está definido por una ecuación por diferencias. El tiempo se considera dividido en períodos de valor constante. Los valores de las variables son digitales (sistemas binario, hexadecimal, etc), y su valor solo se conoce en cada período.
  - De eventos discretos, si el sistema evoluciona de acuerdo con variables cuyo valor se conoce al producirse un determinado evento
- Según la relación entre las variables de los sistemas, diremos que:
  - Dos sistemas están acoplados, cuando las variables de uno de ellos están relacionadas con las del otro sistema.
  - Dos sistemas están desacoplados, si las variables de ambos sistemas no tienen ninguna relación.
- En función de la evolución de las variables de un sistema en el tiempo y el espacio, pueden ser:
  - Estacionarios, cuando sus variables son constantes en el tiempo y en el espacio
  - No estacionarios, cuando sus variables no son constantes en el tiempo o en el espacio.
  - Según sea la respuesta del sistema (valor de la salida) respecto a la variación de la entrada del sistema:

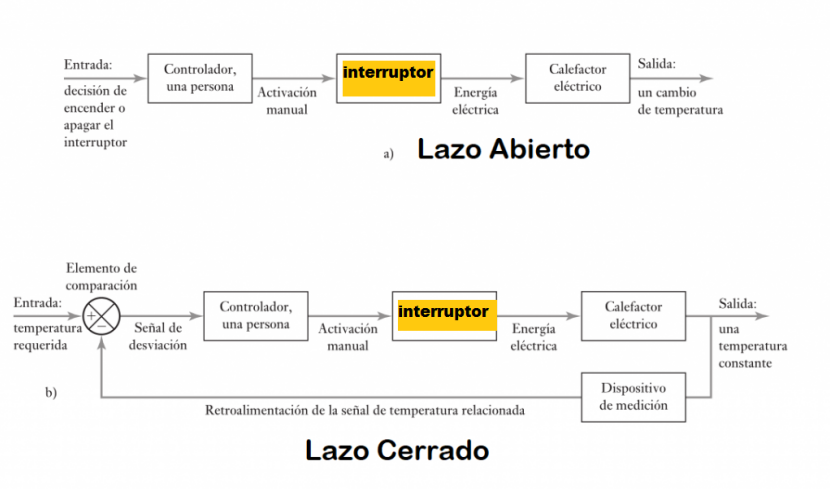


Figura 2.3: Ejemplo sistema en lazo abierto y sistemas en lazo cerrado(fuente: Autor)

    * El sistema se considera estable cuando ante cualquier señal de entrada acotada, se produce una respuesta acotada de la salida.
    * El sistema se considera inestable cuando existe por lo menos una entrada acotada que produzca una respuesta no acotada de la salida.
    * Según sea la respuesta del sistema (valor de la salida) respecto a la variación de la entrada del sistema:

- Si se comparan o no, la entrada y la salida de un sistema, para controlar esta última, el sistema se denomina:
    – Sistema en lazo abierto, cuando la salida para ser controlada, no se compara con el valor de la señal de entrada o señal de referencia.
    – Sistema en lazo cerrado, cuando la salida para ser controlada, se compara con la señal de referencia. La señal de salida que es llevada junto a la señal de entrada, para ser comparada, se denomina señal de feedback o de retroalimentación.

2.2 Hechos por la Naturalesa

Incluyendo sistemas biológicos. Por ejemplo, los movimientos corporales humanos como el acto de indicar un objeto, caminar o hablar. Incluye como componentes del sistema de control:

- Los sentidos (Mediciones), Los músculos (Accionamientos), y el cerebro en su

lóbulo frontal (Controlador).

El cerebro en sí mismo es un sistema de control completo.Las entradas, los sentidos, se procesan en su parte posterior y lateral ocupando el mayor parte de la masa encefálica.Las salidas, los movimientos musculares, se procesan en su parte central, la corteza motora,.El lóbulo frontal es el controlador de las acciones humanas y la mente es el software de todo el sistema.

Control predictivo

Son los sistemas de control que trabajan con un sistema predictivo, y no activo como el tradicional (ejecutan la solución al problema antes de que empiece a afectar al proceso). De esta manera, mejora la eficiencia del proceso contrarrestando rápidamente los efectos.

Otros ejemplos

- Sistema de control de señalización: control de semáforos.
- Sistema de control de temperatura: control de calefacción de una vivienda.
- Sistema de control de nivel de líquidos: control de bombas de agua en un edificio.
- Sistema de control de transporte vertical o de cargas: Control de ascensores o montacargas, cintas transportadoras.
- Sistema de control de traslado de componentes: Control de transferencia de archivos, documentos, componentes multimedia, etc.
- Sistema de control de posición: Control de movimiento de un servomotor.
- Sistema de control de Potencia eléctrica: Control de la Potencia eléctrica proporcionada.

Capítulo 3

Estrategias de control con Arduino

3.1 Estructura de programación en Arduino

La estructura básica del lenguaje de programación de Arduino es bastante simple y se compone de al menos tres partes.

- variables
  La función de configuración debe contener la declaración de las variables. Es la primera función a ejecutar en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar pinMode
- void $setup()$ { es la parte encargada de recoger la configuración. }
  La función $setup()$ se invoca una sola vez cuando el programa empieza. Se utiliza para inicializar los modos de trabajo de los pins, o el puerto serie. Debe ser incluido en un programa aunque no haya declaración que ejecutar.
- void $loop()$ { es la que contienen el programa que se ejecutará cíclicamente (de ahí el termino loop–bucle). }
  Después de llamar a $setup()$, la función $loop()$ hace precisamente lo que sugiere su nombre, se ejecuta de forma cíclica, lo que posibilita que el programa este respondiendo continuamente ante los eventos que se produzcan en la tarjeta.

Ambas funciones son de suma importancia para que el programa trabaje de manera correcta.

Ejemplo de juego de Led

- int pin=9; //declarar el led en el pin 9 (figura 3.1).
- void $setup()$ { pinMode(pin, OUTPUT); // configura el pin como salida }
- void $loop()$ { digitalWrite(pin, HIGH); // pone en uno (on, 5v) el pin

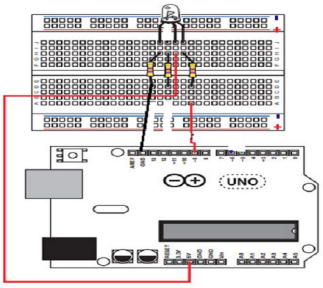


Figura 3.1: Ejemplo de conexión Arduino-Led (fuente: Autor)

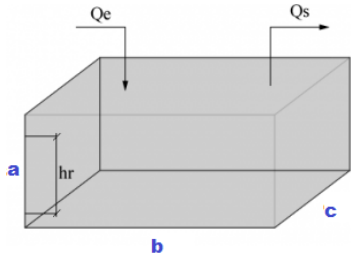


Figura 3.2: Esquema del depósito (fuente: Autor)

delay(1000); // espera un segundo (1000 ms)
digitalWrite(pin, LOW); // pone en cero (off, ov.) el pin delay(1000); }

3.2 Control depósitos de agua

Modelo matemático

A continuación se desarrollará un modelo matemático pragmático que nos servirá sobre todo para desarrollar el apartado de control por eventos.
Datos
$a = 0,18m, b = 0,4m, c = 0,18m$
$Q_e = 3,05 \times 10^{-5}\ m/s$ - $4,71 \times 10^{-5}\ m/s$
$Qs = 3,33 \times 10^{-5}\ m/s$
A partir de las medidas del prisma rectangular obtenemos el volumen

$$V = a.b.c \tag{3.1}$$

El volumen real lo podemos obtener en todo momento sabiendo la altura actual del nivel del depósito.

$$V_r = h_r.b.c \tag{3.2}$$

Conocido el volumen en cada instante podemos obtener el tiempo de llenado o vaciado.

$$t_v = \frac{V_r}{q_e - q_s} \tag{3.3}$$

$$t_{ll} = \frac{(a - h_r)bc}{q_e - q_s} \tag{3.4}$$

Para el tiempo de vaciado (t_v) usamos el volumen calculado previamente. Para el tiempo de llenado (t_{ll}) necesitamos el volumen que nos queda por llenar. El control por eventos implica que es un evento el que hace que muestreamos una señal y no se haga periódicamente como es más habitual. Obviamente el tiempo no desaparece nunca del cálculo, ya que necesitamos tener una referencia a la hora de detectar eventos

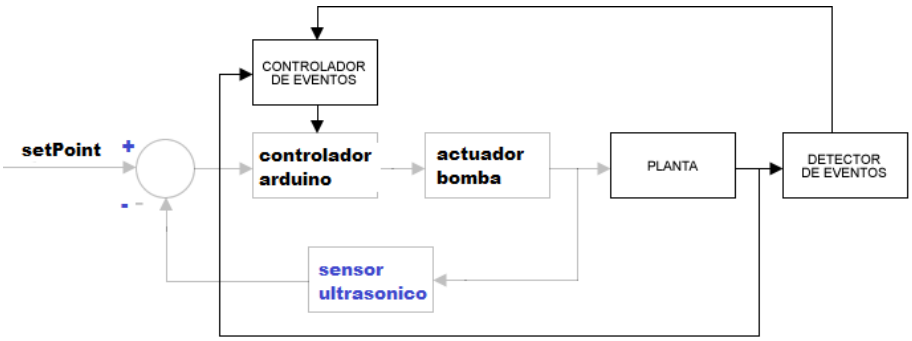


Figura 3.3: Diagrama de flujo del control por eventos en PID (fuente: Autor)

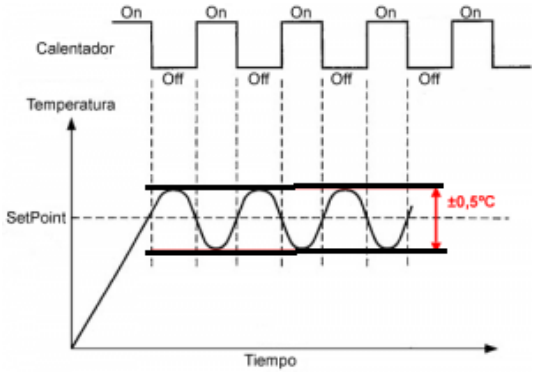


Figura 3.4: Gráfica del control ON/OFF con histéresis. (fuente: Autor)

y muestrear señales. Para ello generaremos una serie de funciones que nos ayudaran en que situación está nuestra planta y si debemos lanzar un evento o podemos esperar al siguiente.

Control ON/OFF con histéresis

Al tratarse de un calentador de 220V, se actuará mediante un relé, que para evitar que no esté entrando y saliendo de forma continua, se establecerá una banda muerta. Tras analizar el comportamiento del sistema, se establece una tolerancia de 0,5ºC como se puede observar en la (figura 3.4). El rango útil del calentador está entre 20 y 30ºC y leeremos la temperatura mediante un sensor de temperatura situado en el depósito 1. Para simplificar, el setpoint de la temperatura será un número entero.

programación de control depósitos de agua

Leemos el setpoint introducido mediante el potenciómetro y comprobamos que la diferencia con la temperatura real no es mayor o igual a medio grado centígrado. Con el

fin de obtener una respuesta más rápida en la temperatura, y puesto que el calentador y la sonda se encuentran en depósitos separados, el nivel está variando entre 65 y 100mm de forma continua, de este modo se garantiza el movimiento de flujo de agua entre depósitos. A continuación se muestra el código principal del control.

```
    Librerías #include < OneWire.h >
#include < DallasTemperature.h >
#include <Wire.h>
# include <LiquidCrystal-I2C.h>
# define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensores(oneWire);
LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);//Direccion de LCD
DeviceAddress S1 = {0x28, 0xFF, 0xAA, 0x58, 0x2D, 0x04, 0x00, 0xBD}; //T Tanque
unsigned long time = 0; // tiempo de ejecucion del ultimo ciclo
int period = 50; // Periodo de muestreo en ms
int periodo = 1000; // Periodo de muestreo en ms
boolean debug = false;//false=matlab
boolean modo = false;//false=auto
true=manual boolean bypassPID = false; //FALSE=PID normal TRUE=nivel 65mm-
100mm
String modos = ;
float temp;
int Trig_pi, Echo_pin; //Trigger = pin 9, ECHO = pin 8
int n=5; // valores para media
int v[5];
unsigned long mm;
int mediamm;
unsigned long lastmm;
int senal;
float Kp = 25;
float Kd = 0.1;
float Ki = 0.3;
float I; // Valor Integral float D; // Valor Derivativo
int Error;
int ErrorAnt;
float control;
unsigned long SetPoint;
int SPtemp;
float ErrorTemp;
```

void setup(void) { Serial.begin(9600); //Abrimos la comunicación por serial
sensores.begin(); //Iniciamos sensor T
lcd.begin(20,4); //Indicamos tipo de LCD
Ultrasonido(9,8);
mm = microsecondsToMeasure(duracion());
v[0] =mm;
v[1] =mm;
v[2] =mm;
v[3] =mm;
v[4] =mm; }
//================================
BUCLE PRINCIPAL
//====================================
void loop(void) {
mm = microsecondsToMeasure(duracion());
Calculo la media del NIVEL
for (int $i = 0; i < n - 1; i + +$){ // Utilizamos array
$v[i] = v[i + 1];$ }
$v[n - 1] = mm; //(mm - lastmm)$; mediamm=0;
for (int $i = 0; i < n; i + +$) // Calculamos la media
mediamm = mediamm + v[i];
mediamm = mediamm/n;
Nivel en parte superior
$mediamm = 170 - mediamm;$
Temperatura
sensores.setResolution(S1, 9);//0,5ºC resolución
sensores.requestTemperatures();
$Mostrar_Datos(S1)$;
if (digitalRead(12) == HIGH)
modo=true;
modos="MANUAL";
else modo=false;
modos="AUTOMATICO";
AUTOMÁTICO——
if (modo==false){
float set=analogRead(A1); //SetPoint regulable NIVEL
SetPoint=map(set,0,1023,50,120);
PID—–
Error———
ErrorAnt=Error;

Error=SetPoint-mediamm;
Derivada—-
$D = Error - ErrorAnt$;
int ok = 3; //damos por bueno banda muerta de 5mm
int nook = 10; //8mm de diferencia, dejo actuar
$if(abs(Error) > ok$ && abs(Error)<nook) { $I = I + mediamm.Ki$; }
else { $I = 0$; } $control = Kp.Error + Kd.D + I$; if(control $<= 0$){
control=0;
senal=0; }
if(control $>= 255$){
control=255;
senal=255;}
if(control $<= 165$ && control $>=50$){
senal=175;
control=175;
} else{
senal=control;}
Actuo sobre la bomba
if (bypassPID==true){ //No hago caso al setpoint
if (mediamm>=100){
analogWrite(3, 0); }
if (mediamm $<= 65$){
analogWrite(3, 255); }
}else{
analogWrite(3, control);}

3.3 Planificación de trayectoria para un vehículo de cuatro ruedas

Para llevar a cabo el modelado del robot se debe tener en cuenta ciertas hipótesis que generalizan el comportamiento del robot, por ejemplo, se supone que el robot se desplaza sobre una superficie plana idealmente sin fricción, los ejes de las ruedas perpendiculares al suelo por donde se desplaza. Finalmente, el robot debe moverse únicamente por las fuerzas que ejerce el movimiento de la rueda de rotación. El robot se considera un mecanismo sólido. rígido y sin partes flexibles, pero debe tener en cuenta las restricciones del sistema no holonómico.

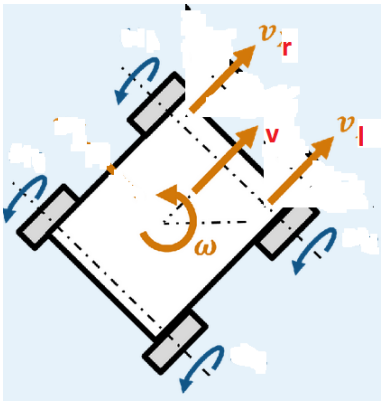


Figura 3.5: Variables de movimiento del movil (propias). (fuente: Autor)

Modelo cinemático

Para lograr un movimiento controlado, deben tener dominio sobre "Vr"la velocidad de la rueda derecha y "Vl"la velocidad de la rueda izquierda, ver figura 4. El propósito de modelar el robot es buscar una relación dirigir cómo las entradas "Vr y Vl"afectan el el sistema indica "x, y, ϕ" El robot puede moverse con una velocidad lineal "v"y rotar con una velocidad angular "ϕ", como se puede ver en la figura 4.
Para que el robot se mueva en línea recta, la velocidad de la rueda debe ser la misma por ello, la velocidad lineal del robot se puede definir como la velocidad media de las ruedas y proporcional a su radio.

$$v = R\frac{v_r + v_l}{2} \tag{3.5}$$

Para que el robot tenga un movimiento de rotación sobre el mismo centro de masa, las velocidades de sus ruedas deben tener la misma magnitud, pero diferente segno, por lo tanto, podemos definir la velocidad angular como la diferencia en la velocidad de sus ruedas de longitud entre a ellos.

$$w = R\frac{v_r - v_l}{2} \tag{3.6}$$

Además de las condiciones de velocidad, plante una matriz de rotación, ya que el robot no siempre está alineado con el eje global.:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} sen\theta & 0 \\ cos\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{3.7}$$

Respuesta del modelo de robot a velocidades desiguales en la misma dirección.

$$x_d = x_d(t), \tag{3.8}$$
$$y_d = y_d(t), \tag{3.9}$$
$$t \leq t_0.$$

Las rutas de referencia de entrada se pueden obtener derivando x_d, y_d e θ_d:

$$\dot{x}_d = v_d \cdot cos(\theta_d), \tag{3.10}$$
$$\dot{y}_d = v_d \cdot sen(\theta_d), \tag{3.11}$$
$$\dot{\theta}_d = v_d \cdot \frac{tan\phi_d}{l} \tag{3.12}$$

Donde v_d y ϕ_d son los comandos de entrada deseables. A partir de las ecuaciones 3.10 y 3.11, calculamos v_d, donde el signo indica si el movimiento es hacia delante o hacia atrás. La ecuación 3.14 presenta la velocidad angular ω_d.

$$v_d(t) = \pm\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \tag{3.13}$$

$$\omega_d(t) = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2 + \dot{y}_d^2} \tag{3.14}$$

La orientación deseada del carrito se calcula a partir de la ecuación 3.15. Cabe señalar que se tiene en cuenta el signo de v_d:

$$\theta_d(t) = Arctg\left(\frac{\dot{y}_d}{v_d}, \frac{\dot{x}_d}{v_d}\right) \tag{3.15}$$

Para obtener el ángulo de dirección deseado se deben derivar las ecuaciones 3.10 y 3.11, además de aislar v_d:

$$\dot{\theta}_d(t) = \frac{1}{1 + \left(\frac{\dot{y}_d}{\dot{x}_d}\right)^2} \cdot \left[\frac{d}{dt}\left(\frac{\dot{y}_d}{\dot{x}_d}\right)\right] \tag{3.16}$$

$$\dot{\theta}_d(t) = \frac{x_d^2}{x_d^2 + y_d^2} \cdot \frac{d}{dt}\left(\frac{\dot{y}_d}{\dot{x}_d}\right) \tag{3.17}$$

$$\dot{\theta}_d(t) = \frac{x_d^2}{v_d^2} \cdot \frac{d}{dt}\left(\frac{\dot{y}_d}{\dot{x}_d}\right) \tag{3.18}$$

$$\dot{\theta}_d(t) = \frac{x_d^2}{v_d^2}\left(\frac{\ddot{y}_d\dot{x} - \ddot{x}_d\dot{y}}{x_d^2}\right) \tag{3.19}$$

$$\dot{\theta}_d(t) = \left(\frac{\ddot{y}_d \dot{x} - \ddot{x}_d \dot{y}}{v_d^2} \right) \tag{3.20}$$

El ángulo de dirección deseado es:

$$\begin{gathered} \phi_d(t) = Arctg \left(l \left[\frac{\ddot{y}_d \dot{x}_d - \ddot{x}_d \dot{y}_d}{v_d^3} \right] \right) \\ \phi_d(t) \in \langle -\tfrac{\pi}{2}, \tfrac{\pi}{2} \rangle \end{gathered} \tag{3.21}$$

La trayectoria cartesiana viene dada por:

$$\begin{cases} x_d(t) = Rcos(\omega t) + x_{dc} \\ y_d(t) = Rsin(\omega t) + y_{dc} \end{cases} \tag{3.22}$$

3.4 La búsqueda de la trayectoria

En este trabajo vamos a implementar una odometría usando un *encoder*, el cual estará conectado al Arduino. Así, podremos calcular la velocidad angular de cada una de las ruedas.

En la Figura **??** sigue una de las partes del dispositivo que nos permite calcular la velocidad (y determinar la posición). Es un disco, con agujeros (*espacios*) en los extremos que está unido a un extremo del eje del motor. En el otro extremo está la rueda. Aún así, dispositivo compuesto por dos partes, donde por un extremo emite una señal (luz) y por el otro la recibe. En el paso de un agujero al otro, tenemos una variación en la señal del sensor y así, en consecuencia, es posible determinar el movimiento.

Figura 3.6: Disco *encoder1* [5].

En la siguiente Figura, el sensor *enconder* está preparado para ser utilizado con el Arduino.

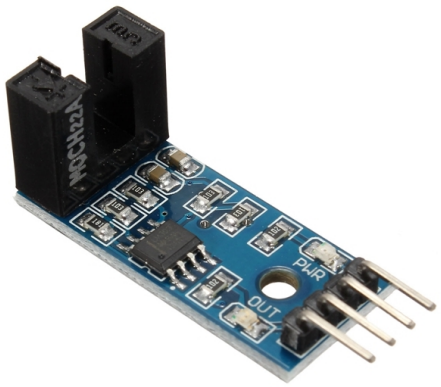


Figura 3.7: Sensor de velocidad *enconder* [6].

Para determinar la velocidad angular en cada una de las ruedas, se debe comenzar midiendo el número de agujeros en el disco que se necesitaban en un momento dado. Por lo tanto:

$$\delta(TK) = TK_f - TK_i \tag{3.23}$$

donde:

- TK: furo ou *gap*,
- TK_f: furo atual;
- Tk_i: furo anterior;

Determinando la velocidad:

$$\omega = \frac{2\pi/N_{Tk}}{\delta TK} \tag{3.24}$$

Donde N_{Tk} corresponde al número de ranuras en el *codificador*. Así, la velocidad lineal, v_l será:

$$v_l = \omega r \tag{3.25}$$

En el caso de un caso genérico, la distancia recorrida será:

$$D = 2r\pi\frac{\delta(TK)}{N_{Tk}} \tag{3.26}$$

Función Odometria

La odometría es un método relativamente simple y económico que se usa comúnmente para estimar la posición. El principio básico de la odometría se basa en la integración de la información de forma incremental a lo largo del tiempo. Lo que termina, en consecuencia, aumentando también el error. La distancia recorrida por la rueda derecha será:

$$D_1 = 2\pi r \frac{\delta(TK_1)}{N_{TK}} \tag{3.27}$$

La distancia recorrida por la rueda izquierda será:

$$D_2 = 2\pi r \frac{\delta(TK_2)}{N_{TK}} \tag{3.28}$$

Por tanto, para la distancia recorrida desde un punto central a un punto dado será:

$$D_c = \frac{D_1 + D_2}{2} \tag{3.29}$$

La actualización de la posición deseada, considerando una trayectoria circular, de radio R es:

$$\begin{cases} x_d(t) = D_c cos(\omega t) + x_{dc} \\ y_d(t) = D_c sin(\omega t) + y_{dc} \end{cases} \tag{3.30}$$

orientación del vehículo

Para que el robot se mueva entre estas posiciones, debe estar desarrollado dos movimientos, primero de orientación y luego de traslación. Para realizar el control de orientación, la velocidad angular del robot como la diferencia de la orientación del robot y el ángulo entre el punto cerca del eje de coordenadas. Como se muestra en la figura 14.

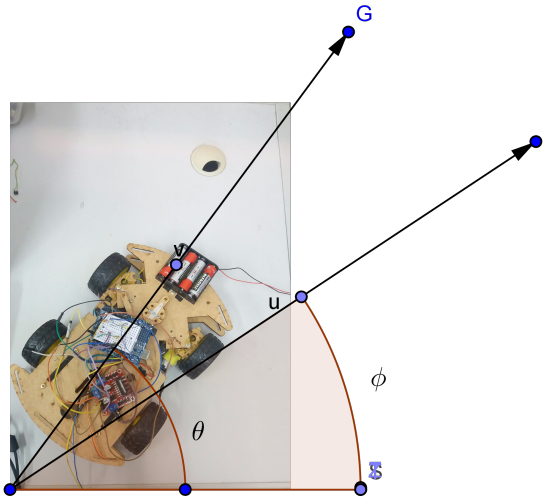


Figura 3.8: Ángulos presentes en el cálculo de la velocidad angular en el método no holonómico *(propio)* [5].

$$\begin{cases} \phi_d = Arctg\left(\frac{y_d - y}{x_d - x}\right) \end{cases} \tag{3.31}$$

con $error = \phi_d - \phi$
donde (x_d, y_d) representa las coordenadas del marcador de la parte delantera del vehículo y (x, y) representa las coordenadas traseras.

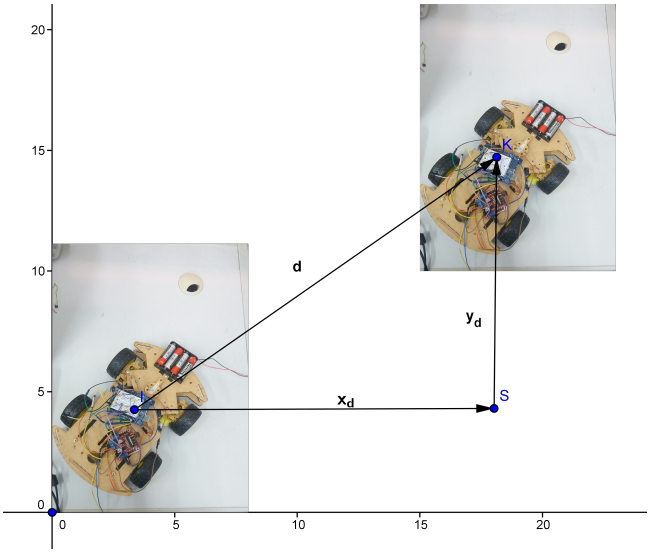


Figura 3.9: Distancia entre puntos método no holonómico. *(propio)* [5].

3.5 Interpolación en Arduíno

tenemos (n +1) puntos de datos $(x_0, y_0), (x_1, y_1)...(x_n, y_n)$ pares que representan (n + 1) puntos de la gráfica de la función y =f(x) , cuya forma explícita no se conoce, asumimos diferentes valores donde la función es continua. y el polinomio a encontrar satisface las siguientes restricciones.

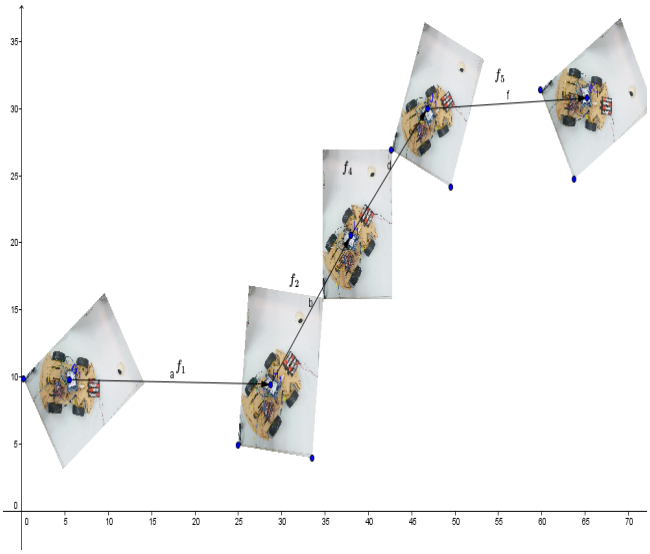


Figura 3.10: Segmentos de la ruta definida por funciones *(propio)* [5].

$$P_n(x_i) = y_i, i = 0, ...n \tag{3.32}$$

si el polinomio

$$P_n(x_i) = a_0 + a_1x + a_2x^2 + ... + a_nx^n \tag{3.33}$$

para cumplir con las restricciones, se generan (n + 1) ecuaciones y (n + 1) incógnitas

$$\begin{aligned} a_0 + a_1x_0 + a_2x_0^2 + ... + a_nx_n^n &= y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + ... + a_nx_n^n &= y_1 \\ a_0 + a_1x_2 + a_2x_2^2 + ... + a_nx_n^n &= y_0 \\ a_0 + a_1x_+a_2x_3^2 + ... + a_nx_n^n &= y_1 \\ &. \\ &. \\ &. \end{aligned} \tag{3.34}$$

en forma matricial

$$\begin{bmatrix} 1 & x_0 & x_0^2 & ...x_0^n \\ 1 & x_1 & x_1^2 & ...x_1^n \\ 1 & x_1 & x_1^2 & ...x_2^n \\ & & & ... \\ & & & ...x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ . \\ . \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ . \\ y_n \end{bmatrix} \tag{3.35}$$

programación en el software Arduino para el móvil autónomo

```
#include <math.h>
#define PI 3.14159265358979323846264338327 95
#include "InterpolationLib.h"
int N = 20;
int contadorTicks = 1;
int tam = 10;
int k = 10;
volatile unsigned muestreoActual = 0;
volatile unsigned muestreoAnterior = 0;
volatile unsigned deltaMuestreo = 0;

float error = 0;
float Kp = 70;
int PWMr = 0;
int PWMl = 0;

int PWMmax=100;
int PWMmin=60;
```

Figura 3.11

```
float Cdistancia = 0;
float x = 0;
float y = 0;
float phi = 0;

//------------------------------- Variables Posición de

float Xd=0 ;
float Yd=15 ;
float Phid= atan2(Yd-y, Xd-x);

//------------------------------- Variables del robot

float diametro = 6.8;
float longitud = 13.4;
float V = 0;
float W = 0;

//------------------------------- Variables de motor de

volatile unsigned muestreoActualInterrupcionR = 0;
volatile unsigned muestreoAnteriorInterrupcionR = 0;
double deltaMuestreoInterrupcionR = 0;

int encoderR = 3;   // pin de conexiòn del encoder dere
int llantaR = 11;      // pin de conexiòn de llanta der
```

Figura 3.12

```
double frecuenciaR = 0;
double Wr = 0;
double Vr = 0;
int CR = 0;
float vectorR[10] = {0, 0, 0, 0, 0, 0, 0, 0, 0,0};    //

float Rdistancia = 0;
int Rtick = 0;
int RtickAnt = 0;
int deltaRtick = 0;

//-----------------------------  Variables de motor Izq

volatile unsigned muestreoActualInterrupcionL = 0;
volatile unsigned muestreoAnteriorInterrupcionL = 0;
double deltaMuestreoInterrupcionL = 0;

int encoderL = 2;   // pin de conexiòn del encoder Izqui
int llantaL = 10;      // pin de conexiòn de llanta Izqu

double frecuenciaL = 0;
double Wl = 0;
double Vl = 0;
int CL = 0;
```

Figura 3.13

```
double frecuenciaL = 0;                                    // frecuencia
double Wl = 0;                                             // Velocidad a
double Vl = 0;                                             // velocidad l
int CL = 0;                                                // contador Ti
float vectorL[10] = {0, 0, 0, 0, 0, 0, 0, 0,0,0};    // vector de almac

float Ldistancia = 0;                                      // distancia r
int Ltick = 0;                                             // ticks del e
int LtickAnt = 0;                                          // ticks del e
int deltaLtick = 0;                                        // diferencia
//----------------------------  interpolacion ----------------------
const int numValues = 12;
double XdValues[12] = {  0, 1, 2,  3,  4,  5,  6,  7,  8,  8.5, 9, 10
double YdValues[12] = {  0, 1, 2,  2.5, 2,  1, -1, -2.5,-2,-1, -0.5,  (
void setup() {
  attachInterrupt(digitalPinToInterrupt(encoderR),REncoder,FALLING);
  attachInterrupt(digitalPinToInterrupt(encoderL),LEncoder,FALLING);
  Serial.begin(9600); // inicio de la comunicaciòn serial
  while (!Serial) { ; }
```

Figura 3.14

```
  Serial.begin(9600); // inicio de la comunicaciòn serial
  while (!Serial) { ; }
  for (float XdValue = 0; XdValue <= 10; XdValue += .05)
  {
      Serial.print(Interpolation::Step(XdValues, YdValues, numValues, XdValue, 0.0));
    Serial.print(',');
    Serial.print(Interpolation::Step(XdValues, YdValues, numValues, XdValue, 0.5));
    Serial.print(',');
    Serial.print(Interpolation::Step(XdValues, YdValues, numValues, XdValue, 1.0));
    Serial.print(',');
    Serial.print(Interpolation::SmoothStep(XdValues, YdValues, numValues, XdValue));
    Serial.print(',');
    Serial.print(Interpolation::Linear(XdValues, YdValues, numValues, XdValue, false));
    Serial.print(',');
    Serial.print(Interpolation::Linear(XdValues, YdValues, numValues, XdValue, true));
    Serial.print(',');
    Serial.print(Interpolation::CatmullSpline(XdValues, YdValues, numValues, XdValue));
    Serial.print(',');
    Serial.println(Interpolation::ConstrainedSpline(XdValues, YdValues, numValues, XdValue));
  }
}
```

Figura 3.15

```
void REncoder() {
      Rtick++;
      CR++;
      if (CR == contadorTicks){
          float media = 0;
//---------------------------------- ----------------------------
          for(int i=0;i < tam-1;i++){
              vectorR[i]=vectorR[i+1];
          }
          vectorR[tam-1]=deltaMuestreoInterrupcionR ;

          for(int i=0;i<tam;i++){
              media = vectorR[i]+ media;
          }
          media = media/tam;
          deltaMuestreoInterrupcionR = media;
//---------------------------------- ---------------------------- -
          frecuenciaR = (1000)/ deltaMuestreoInterrupcionR;
          muestreoAnteriorInterrupcionR = muestreoActualInterrupcionR;
          CR = 0;
      }
 }

void LEncoder() {
      Ltick++;
```

Figura 3.16

```
void LEncoder() {
      Ltick++;
      CL++;
      if (CL == contadorTicks){
          float media = 0;
//------------------------------------ ---------------------------
          for(int i=0;i < tam-1;i++){
              vectorL[i]=vectorL[i+1];
          }
          vectorL[tam-1]=deltaMuestreoInterrupcionL;

          for(int i=0;i<tam;i++){
              media = vectorL[i]+ media;
          }
          media = media/tam;
          deltaMuestreoInterrupcionL = media;
//------------------------------------ ---------------------------
          frecuenciaL = (1000)/ deltaMuestreoInterrupcionL;
          muestreoAnteriorInterrupcionL = muestreoActualInterrupcionL;
          CL = 0;
       }
 }
```

Figura 3.17

```
void loop() {
    muestreoActual = millis();
    muestreoActualInterrupcionR = millis();
    muestreoActualInterrupcionL = millis();

    deltaMuestreo =(double) muestreoActual - muestreoAnterior;
    if ( deltaMuestreo >= k)
    {
        float Phid= atan2(Yd-y, Xd-x);
        deltaMuestreoInterrupcionR = muestreoActualInterrupcionR -  muestreoAnteriorInterrupcionR;
        deltaMuestreoInterrupcionL = muestreoActualInterrupcionL -  muestreoAnteriorInterrupcionL;

        if(deltaMuestreoInterrupcionR >= 300*contadorTicks){
          frecuenciaR=0;
        }
        if(deltaMuestreoInterrupcionL >= 300*contadorTicks){
          frecuenciaL=0;
        }

        Wr = contadorTicks*((2*PI)/N)*frecuenciaR;
        Vr= Wr*(diametro/2);
        Wl = contadorTicks*((2*PI)/N)*frecuenciaL;
        Vl= Wl*(diametro/2);
        V = 70;
        error = Phid - phi;
        W = (Vr-Vl)/longitud + Kp * error;
```

Figura 3.18

```
void loop() {
    muestreoActual = millis();
    muestreoActualInterrupcionR = millis();
    muestreoActualInterrupcionL = millis();

    deltaMuestreo =(double) muestreoActual - muestreoAnterior;
    if ( deltaMuestreo >= k)
    {
        float Phid= atan2(Yd-y, Xd-x);
        deltaMuestreoInterrupcionR = muestreoActualInterrupcionR -  muestreoAnteriorInterrupcionR;
        deltaMuestreoInterrupcionL = muestreoActualInterrupcionL -  muestreoAnteriorInterrupcionL;

        if(deltaMuestreoInterrupcionR >= 300*contadorTicks){
          frecuenciaR=0;
        }
        if(deltaMuestreoInterrupcionL >= 300*contadorTicks){
          frecuenciaL=0;
        }

        Wr = contadorTicks*((2*PI)/N)*frecuenciaR;
        Vr= Wr*(diametro/2);
        Wl = contadorTicks*((2*PI)/N)*frecuenciaL;
        Vl= Wl*(diametro/2);
        V = 70;
        error = Phid - phi;
        W = (Vr-Vl)/longitud + Kp * error;
```

Figura 3.19

```
void odometria(){

   deltaRtick = Rtick - RtickAnt;
   Rdistancia = PI*diametro*(deltaRtick/(double) 20);

   deltaLtick = Ltick - LtickAnt;
   Ldistancia = PI*diametro*(deltaLtick/(double) 20);

   Cdistancia = (Rdistancia + Ldistancia)/2;
   x = x + Cdistancia*cos(phi);
   y = y + Cdistancia*sin(phi);

   phi = phi + ((Rdistancia - Ldistancia)/longitud);
   phi = atan2(sin(phi),cos(phi));

   RtickAnt = Rtick;
   LtickAnt = Ltick;
}
```

Figura 3.20

3.6 Diseño y construcción de un sistema automatizado de control de bombas de agua en un cultivo hidropónico en el entorno Arduino

El objetivo del trabajo es desarrollar un sistema automatizado de riego en torno al Arduino para cultivos hidropónicos. El sistema utiliza una placa Arduino ATmega328P a la que se conecta diferentes componentes, tales como un módulo de reloj en tiempo real, pantalla LCD I2c, dos módulos relé de un canal y resistencias eléctricas; con la

finalidad de obtener un sistema de micro controlador programable que puede activar y desactivar la bomba de agua de un invernadero en tiempos determinados de acuerdo al desarrollo del cultivo, particularmente en los cultivos de lechugas con diferentes variedades, para ello es necesario utilizar diferentes técnicas de control y realizar varias pruebas experimentales con el fin de ajustar ciertos parámetros de control, especialmente la humedad y temperatura. Este sistema fue instalado en el invernadero de la Comunidad de Mollepata en Ayacucho, dentro del convenio tria-anual "Inti, la energía que alimenta la Tierra" entre la Universidad Nacional de San Cristóbal de Huamanga (UNSCH) y el Comité Regional de Educación para el Desarrollo Internacional de Lanaudière (CREDIL–JOLIETTE) – Canadá.

La inversión en ciencia y tecnología agrícola, expresada en términos de investigación y servicios de extensión, ha demostrado tener una gran relevancia en el aumento del rendimiento de los cultivos agrícolas y la reducción de la pobreza en los países en desarrollo. Sin embargo, es importante que esta inversión esté definida según las necesidades de conocimiento expresadas por todos los actores involucrados [9].El cultivo hidropónico se basa fundamentalmente en el desarrollo de las hortalizas en 8 canales de PVC dispuestos en cada parcela, llamados canales de cultivo, donde pasa un bajo caudal de agua con una pequeña cantidad de solución nutritiva la cual fluye hacia las raíces de las lechugas manteniéndolas nutridas, hidratadas, oxigenadas y de esta manera en constante desarrollo. Cada canal de cultivo tiene agujeros, donde se colocan las plantas, estos canales están apoyados sobre caballetes de fierro y madera que tiene una ligera pendiente o desnivel que facilita la circulación de la solución nutritiva. La solución es recolectada y almacenada en un tanque de polietileno y mediante la bomba de agua que lleva la solución hasta los canales de cultivo [8].

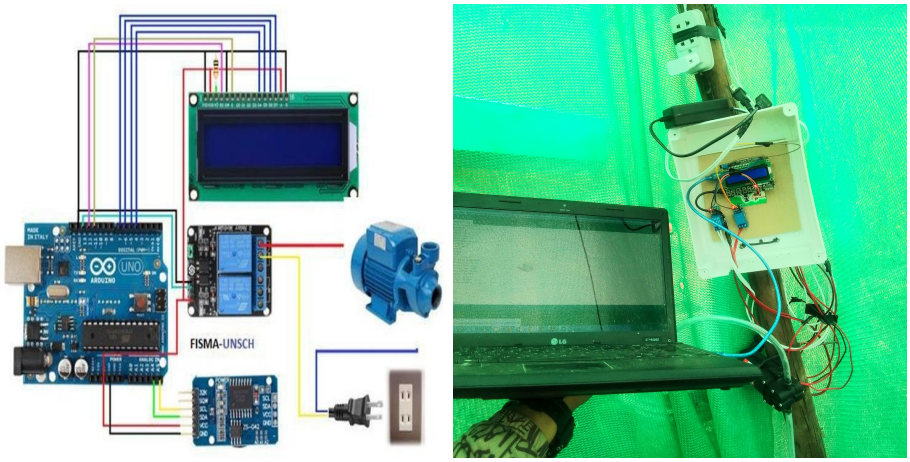


Figura 3.21: figura de la fase terminal del sistema de control automatizado y muestra las conexiones.

Cruz Díaz, y Gutiérrez Bulla en 2021 desarrollaron un prototipo tipo invernadero con condiciones de temperatura y humedad relativa controladas y un sistema de iluminación suplementaria, con el fin de garantizar condiciones ambientales posicionadas para la planta y mejorar su crecimiento y desarrollo, permitiendo la maximización del cultivo por área [10]. Luego en el mismo año Jose Luis Huayanay, and José Paulo VS Cunha desarrollo una importante investigación "Controladores de modo deslizante para Humedad del suelo modelada por ecuación Diferencial parcial parabólico no lineal" consistía en que el control de modo deslizante (SMC) se aplica para regular el contenido de agua en el suelo para riego de precisión. La dinámica de la infiltración de agua en medios porosos (suelo) se modela mediante la ecuación de Richards y se consideran dos objetivos de control: el control de la humedad promedio y el control del contenido de agua del suelo a una profundidad específica [11].

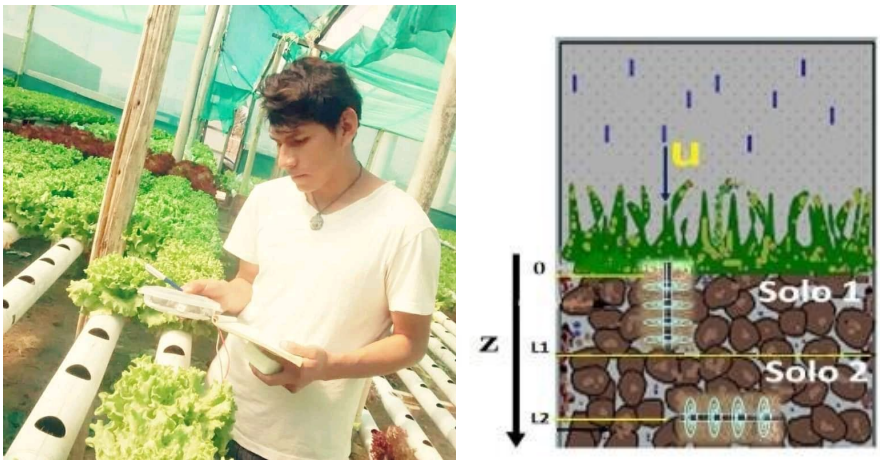


Figura 3.22

La ventaja de nuestro proyecto con respecto a los sistemas industriales, es que podemos incorporar más relés para diferentes bombas de agua en el caso de ampliar el área de cultivo y graduar los tiempos de bombeo.

Para un óptimo funcionamiento de nuestro sistema se debe proveer de constante energía eléctrica, de otra forma se originaria una desactualización del tiempo programado, este problema se debe al tipo de precisión de los módulos de reloj insertados en el sistema automatizado.

3.7 Programa para el sistema de riego automatizado

Para el funcionamiento del sistema se programó en el software Arduino, en lenguaje Java. El programa es el siguiente: #include <Wire.h>

```
#include "RTClib.h"
#include <LiquidCrystal.h>
liquidCrystal lcd (12, 11, 4,5,6, 7);
RTC_DS3231 RTC;
int hora=0;
int minutos=0;
int segundos=0;
int salida=13;
int salida2=2;
void setup () { lcd.begin (16,2);
Inicializamos el puerto serie, wire y el módulo RTC
Serial.begin (9600);
Wire.begin ();
RTC.begin ();
Si quitamos el comentario de la línea siguiente, se ajusta la hora y la fecha con la del
ordenador
RTC.adjust (Date Time ( DATE , TIME ));
PinMode (salida, OUTPUT); }
void loop () { DateTime now = RTC.now();
hora=(now.hour(),DEC);
minutos=(now.minute(),DEC);
segundos=(now.second(),DEC);
lcd.setCursor(0,0);


    lcd.print(now.day(), DEC);
lcd.print('/');
Imprimimos el mes
lcd.print(now.month(), DEC);
lcd.print('/');
Imprimimos el año
lcd.print(now.year(), DEC);
lcd.print(' '); Imprimimos la hora
lcd.setCursor(0,1);
lcd.print(now.hour(), DEC);
lcd.print(':');
Imprimimos los minutos
lcd.print(now.minute(), DEC);
lcd.print(':');
Imprimimos los segundos
```

```
lcd.print (now.second(), DEC);
lcd.setCursor (11,0);
lcd.print("FOCO");
if (now.hour()== 1 && now.minute() == 00 && now.second() == 00)
digitalWrite(salida,LOW);
lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 6 && now.minute() == 00 && now.second() ==00 )  digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 7 && now.minute() == 00 && now.second() == 00) digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 8 && now.minute() == 00 && now.second() == 00)  digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 9 && now.minute() == 00 && now.second() == 00)  digital-
Write(13,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);  else if (now.hour()==
10 && now.minute() == 00 && now.second() == 00) digitalWrite(salida,LOW); lcd.setCursor(12,1);
lcd.print("ON ");
delay(480000);
else if (now.hour()== 10 && now.minute() == 30 && now.second() == 00) digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON ");
delay(480000);
else if (now.hour()== 11 && now.minute() == 00 && now.second() == 00) digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 11 && now.minute() == 30 && now.second() == 00)  digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 12 && now.minute() == 00 && now.second() == 00) digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 12 && now.minute() == 30 && now.second() == 00) digi-
talWrite(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);  else if
(now.hour()== 13 && now.minute() == 00 && now.second() == 00) digitalWrite(salida,LOW);
lcd.setCursor(12,1);
lcd.print("ON "); delay(480000);
else if (now.hour()== 13 && now.minute() == 30 && now.second() == 00)  digi-
talWrite(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);  else if
(now.hour()== 14 && now.minute() == 00 && now.second() == 00) digitalWrite(salida,LOW);
lcd.setCursor(12,1);
lcd.print("ON "); delay(480000);
else if (now.hour()== 14 && now.minute() == 30 && now.second() == 00)
digitalWrite(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 15 && now.minute() == 00 && now.second() == 00)  digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
```

```
else if (now.hour()== 15 && now.minute() == 30 && now.second() == 00)  digital-
Write(salida,LOW);
lcd.setCursor(12,1);
lcd.print("ON "); delay(480000);
else if (now.hour()== 16 && now.minute() == 00 && now.second() == 00)  digital-
Write(salida,LOW);
lcd.setCursor(12,1); lcd.print("ON ");
delay(480000);
else if (now.hour()== 16 && now.minute() == 00 && now.second() == 00) digital-
Write(salida,LOW); lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else if (now.hour()== 16 && now.minute() == 30 && now.second() == 00)  digital-
Write(salida,LOW);
lcd.setCursor(12,1);
lcd.print("ON ");
delay(480000);
else if (now.hour()== 18 && now.minute() == 00 && now.second() == 00)  digital-
Write(salida,LOW);
lcd.setCursor(12,1);
lcd.print("ON ");
delay(480000);
else if (now.hour()== 22 && now.minute() == 00 && now.second() == 00)  digital-
Write(salida,LOW);
lcd.setCursor(12,1); lcd.print("ON "); delay(480000);
else
{digitalWrite(salida,HIGH);
lcd.setCursor(11,1); lcd.print("OFF"); }}
```

Bibliografia

[1] Trajectory Planning for a Four-Wheel-Steering Vehicle Danwei Wang ,Feng Qi,2001

[2] KEVIN STEVEN SARMIENTO GAMBOA,DESARROLLO DE PLANEACIÓN Y SEGUIMIENTO DE TRAYECTORIAS PARA UN VEHÍCULO AUTÓNOMO,2015

[3] PEARCE, J. George C. Devol, Inventor of Robot Arm, Dies at 99. In: The New York Times. https://www.nytimes.com/2011/08/16/business/george-devol-developer-of-robot-arm-dies-at-99.html: [s.n.].

[4] AYRES, M. Conheça a história dos robôs. Disponível em: UOL Notícias. Tecnologia. https://tecnologia.uol.com.br/ultnot/2007/10/01/ult4213u150.jhtm:. Acesso em: 02/11/2019.

[5] Baú da Eletrônica. Disponívem em: https://www.baudaeletronica.com.br/disco-encoder.html. Acesso em: 07/12/2019.

[6] FILIPEFLOP. Disponívem em: https://www.filipeflop.com/produto/sensor-de-velocidade-encoder/. Acesso em: 07/12/2019.

[7] NIKU, Saeed B. Introdução à Robótica: Análise, Controle e Aplicações. 2. ed. Rio de Janeiro: LTC, 2013. 382 p. ISBN 978-85-216-2237-6.

[8] Gilsanz, Juan C. Hidroponia. No. CIDAB- SB321-G5h. Instituto Nacional de Investigación Agropecuaria (Uruguay)., 2007

[9] MUNDIAL, Banco. Agricultura para el desarrollo. Informe sobre el desarrollo mundial, v. 2008, p. 19, 2008.).

[10] Cruz Díaz, Wilmer Orlando, and Juan Pablo Gutiérrez Bulla. "Diseño y construcción de un invernadero para el cultivo de fresa."2021.

[11] Villar, Jose Luis Huayanay, and José Paulo VS Cunha. "Sliding-Mode Controllers for Soil Moisture Modeled by Nonlinear Parabolic Partial Differential Equations."2022 16th International Workshop on Variable Structure Systems (VSS). IEEE, 2022.

Automatización y control en el entorno
Arduino I

Maecenas urna nisi, luctus nec lorem eu, vehicula varius eros. Nullam non quam tempus, ultrices lorem at, viverra felis. Nam eu ligula sodales, suscipit sem sed, bibendum augue.

0 001708 417846