Semillero de investigación en robótica ASIMOV

Equipo de electrónica

Mayo, 2021

1.Objetivo: El desarrollo de un circuito electrónico para el correcto funcionamiento de Pascal

1.1 Objetivo general: Diseñar e implementar los circuitos electrónicos necesarios para garantizar el desarrollo de Pascal mediante el uso de microcontroladores, controladores de carga, sensores, actuadores, entre otros.

1.2 Objetivos específicos

- Diseñar un circuito que permita controlar el movimiento de pascal a nivel bajo mediante motores.
- Utilizar herramientas de simulación de circuitos eléctricos para el diseño a implementar.
- Acoplar el diseño electrónico a todo el conjunto de elementos que le enviarán las señales al robot para asegurar su funcionamiento.
- Diseñar una tarjeta para la implementación del circuito.

2. Metodología

1. Selección del microcontrolador

Por medio de evaluación utilizando el software de Altium, después de las propuestas expuestas por los integrantes, se escogió el Teensy 3.5

1. Selección de complementos

Principalmente el circuito contará con:

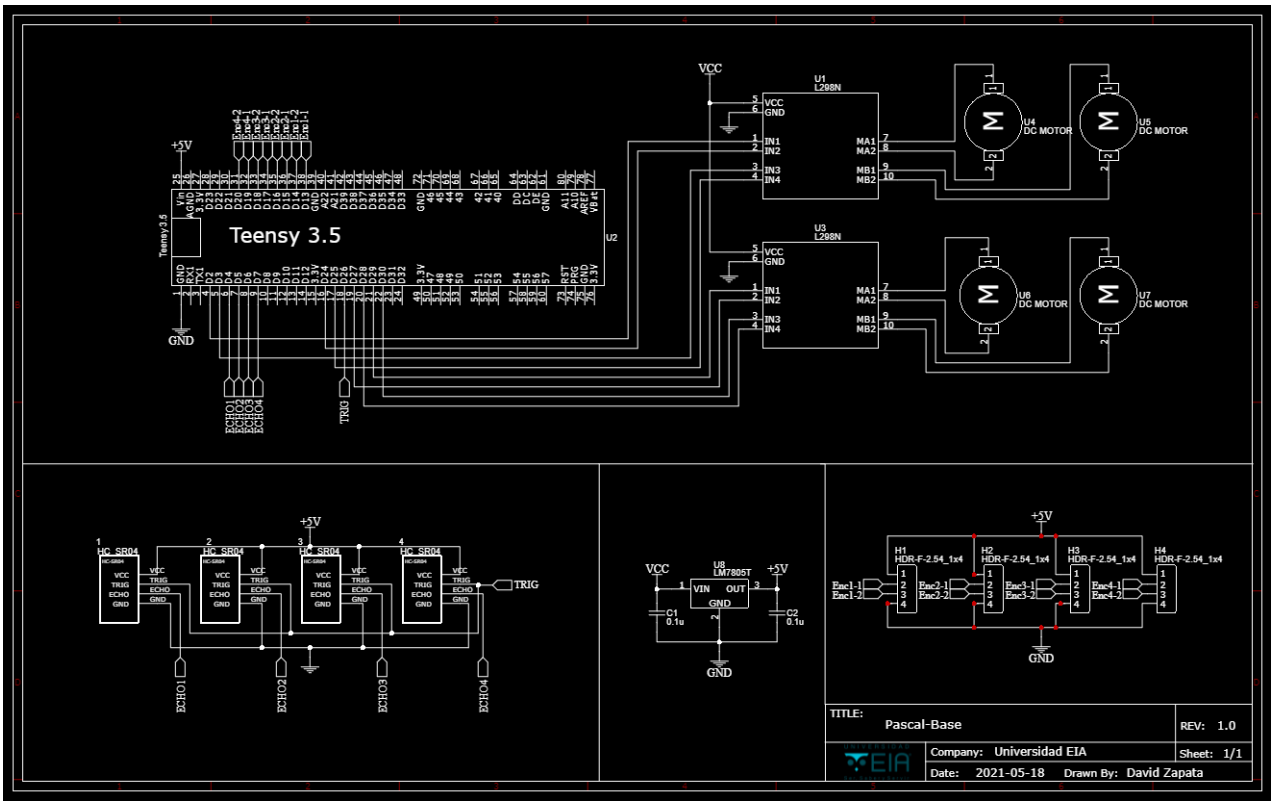
- 4 motores con encoder

- dos puente H L298N (los cuales permitirán el control de sentido de giro de los motores junto con la posibilidad de variar la velocidad de estos a través de una señal PWM)

-microcontrolador Teensy 3.5, al que se conectan los complementos resistivos para que luego de programarse, sea el encargado de enviar la señal a los motores y otros complementos.

- 4 HC SR04 sensor ultrasonido

2. Elaboración de esquemático a través de EasyEDA.

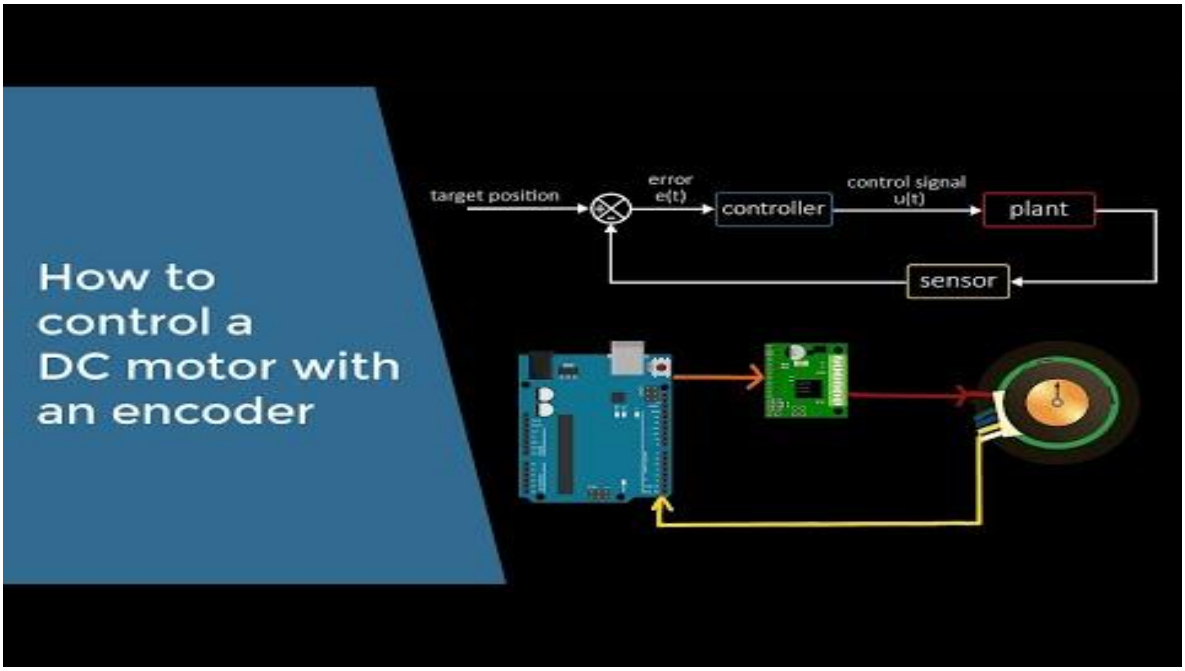


3. Simulación PID

Para realizar la simulación primero se abordó que es, cómo funciona el sistema PID y como controlar un motor DC con este. Esto con un video y una clase:

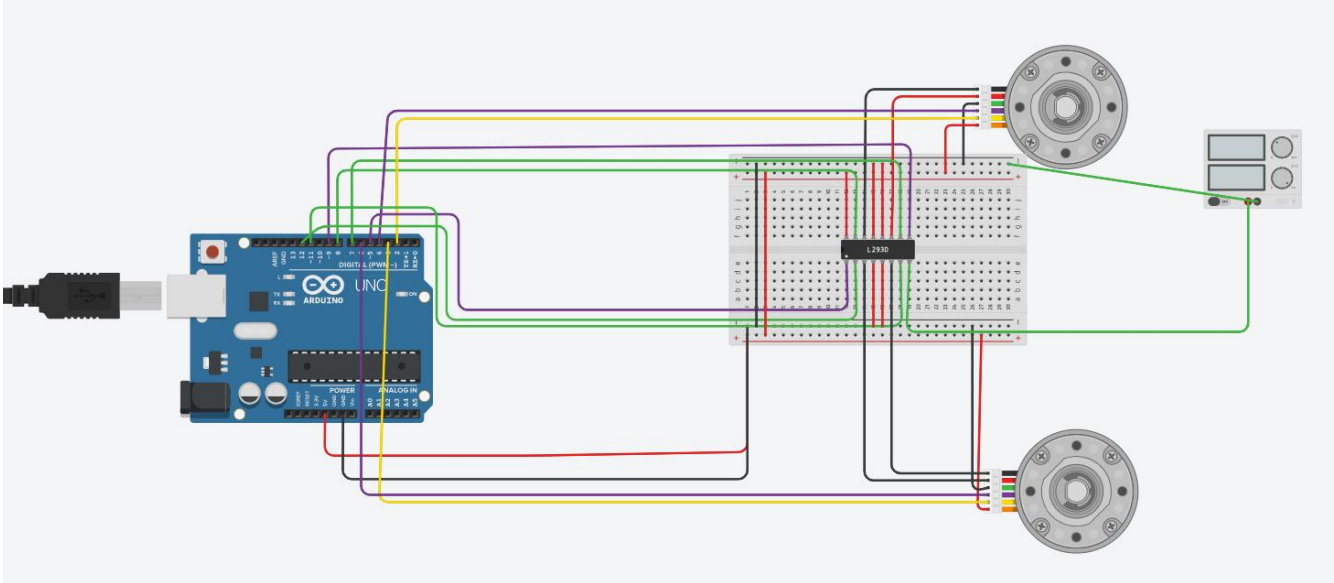
-Microsoft Stream

- How to control a DC motor with an encoder



Luego guiándonos de la explicación del segundo video se hizo una simulación y un código para controlar 2 motores usando controladores PID:

https://www.tinkercad.com/things/jYh3HoXMNgd-encoder-movimiento/editel?sharecode=Qla_xvD2TVQ2ih0_Yvh4itEGK5_ROyKgEObXbO-o1Rk



Explicación del código: (primero ver la clase y el video), para ver el código completo ir al link de la simulación.

Primero se define cada uno de los pines que vamos a usar y se les da un nombre especifico según su función.

ENCAi (encoder A del motor izquierdo) ENCBi (encoder A del motor izquierdo) PWMi (pin PWM del motor izquierdo) IN2(pin de salida del motor izq hacia el puente H) IN1(pin de salida del motor izq al puente H)

```
1  // se Definen los pines
2  #define ENCAi 2
3  #define ENCBi 4
4  #define ENCAd 3
5  #define ENCBd 6
6  #define PWMi 9
7  #define PWMd 5
8  #define IN2 8
9  #define IN1 7
10 #define IN3 12
11 #define IN4 11
12
```

Se definen algunas variables globales antes del void set up: posi (posición del motor izquierdo), posd (posición del motor derecho), prevT (tiempo en el loop anterior), eprevi (error previo en el motor izquierdo), eprevd (error previo en el motor derecho), eintegrali (variable para ir acomulando la suma de errores en el motor izquierdo), integrad (variable para ir acomulando la suma de errores en el motor derecho).

En el void set up se definen los pines de los encoders como inputs, y cada vez que el encoder A de cada motor da una lectura, se hace una interrupción para ejecutar la función de leer el encoder.

```
// variables globales
int posi = 0;
int posd=0;
long prevT = 0;
float eprevi = 0;
float eprevd = 0;

float eintegrali = 0;
float eintegrald= 0;




void setup() {
  Serial.begin(9600);
  pinMode(ENCAi,INPUT_PULLUP);
  pinMode(ENCBi,INPUT_PULLUP);
  pinMode(ENCAd,INPUT_PULLUP);
  pinMode(ENCBd,INPUT_PULLUP);
  attachInterrupt(digitalPinToInterrupt(ENCAi),readEncoderi,RISING);
  attachInterrupt(digitalPinToInterrupt(ENCAd),readEncoderd,RISING);
}
```

Se crean 3 funciones

-Función para leer el enconder izquierdo (readEncoderi): esta función se activa con cada interrupción del enconder dada por el void set up, la función mira si el Encoder B esta encendido o apagado. si el encoder B esta encendido a la misma vez que el A significa que estamos en dirección contraria a las manecillas del reloj y le suma 1 a la posición (la variable global), si el encoder B esta apagado vamos en dirección de las manecillas del reloj y le resta 1 a la posición.

-Función del encoder derecho (readEncoderd): esta función se activa con cada interrupción del enconder dada por el void set up, la función mira si el Encoder B esta encendido o apagado. si el encoder B esta encendido a la misma vez que el A significa que estamos en dirección contraria a las manecillas del reloj y le suma 1 a la posición (la variable global), si el encoder B esta apagado vamos en dirección de las manecillas del reloj y le resta 1 a la posición.

-función setmotor: se le entregan 5 parámetros dir(dirección), pwmVal (un valor entre 0 y 255 encargado de controlar la velocidad), pwm (pin que emite la señal pwm), in1(pin1 conectado al puente h), in2 (pin2 conectado al puente h)

Si la función recibe como dirección un 1 emitirá un high por el in1 y un low por el in2, depende de como este conectado el puente h esto hará que gire en un sentido, si la función recibe un -1 emitirá un low por el in1 y un high por el in2 y hará que gire en sentido contrario al anterior.

Además la función recibe el valor y el pin del PWM y los envía de forma análoga hacia el puente h.

```
}
// funcion para controlar un motor
void setMotor(int dir, int pwmVal, int pwm, int in1, int in2){
  analogWrite(pwm,pwmVal);
  if(dir == 1){
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
  }
  else if(dir == -1){
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
  }
  else{
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
  }
}
// funcion para leer encoder izquierdo
void readEncoderi(){
  int b = digitalRead(ENCBi);
  if(b > 0){
    posi++;
  }
  else{
    posi--;
  }
}
  void readEncoderd(){
  int c = digitalRead(ENCBd);
  if(c > 0){
    posd++;
  }
  else{
    posd--;
  }
 }
```

En el void loop se definen las 3 constantes que del PID, una para la parte lineal kp, una para la integral ki, y una para la derivada kd. Además se define el target o la posición a la que se busca poner el motor. Esta posicion esta dada dependiendo de las lecturas de cada encoder, por lo que hay que analizar que encoder se va a usar y cuantas lecturas da en una vuelta.

```
void loop() {
 //constantes del PID
  float kp = 1;
  float kd = 0.01;
  float ki = 0.055;



  //Motor IZQUIERDO


 // Objetivo en radianes

 float targetradi =6.28;

 float targeti =targetradi*97.1;

```

-Se Inicia el código de control PID (para el motor izquierdo) donde lo primero que se hace es usar una función del Arduino (micros())para saber el tiempo en cada loop y se almacena en CurrT.

-Con este tiempo se calcula la diferencia de tiempo en cada loop y se almacena en deltaT. Luego de saber el deltaT se reescribe el tiempo previo con el del ultimo loop para poder calcular esta diferencia cada ciclo.

```
57    //PID
58    long currT = micros();
59    float deltaT = ((float) (currT - prevT))/( 1.0e6 );
60    prevT = currT;
```

-Luego se calcula el error (ei) entre la posicion en la que esta el motor y la que queremos que llegue y imprimos este error para luego poder revisar que funcione bien el programa.

```
63
64    int ei =targeti - posi;
65    Serial.print("  ");
66
67    Serial.print(ei);
68  Serial.print(" ");
```

-Se crea una variable dedti la cual mide la pendiente de cuanto vario el error respecto a la diferencia del tiempo. (DERIVADA) (D)

```
69    float dedti = (ei-eprevi)/(deltaT);
70
```

-Se empieza a sumar en cada loop el error (ei) * el delta de tiempo (deltaT) , lo cual es la definicion de la INTEGRAL (I)

```
71    eintegrali = eintegrali + ei*deltaT;
```

-Se crea una variable u que es la suma de los PID, la respectiva constante multiplicada por la integral, la derivada, y el error (la parte lineal)

```
73
74    float u = kp*ei + kd*dedti + ki*eintegrali;
75
```

-Con esta variable u creamos la variable pwri, la cual es el valor absoluto de u. si U es un numero mayor a 255 se deja en 255, esto es para controlar luego la salida de pwm y por ende la velocidad.

```
77    float pwri = abs(u);
78    if( pwri > 255 ){
79      pwri = 255;
80    }
81
```

-Tambien se crea la variable diri (direccion izquierda) la cual mira si u es mayor o menor que 0. Que u se mayor o menor que 0 depende solamente del error, ya que si el error es mayor que 0 significa que el target indicado es positivo, osea que se quiere que el motor rote encontra de las manecillas del reloj y la variable da un guarda un 1. Si u es menor que 0 es por que el target esta a

favor de las manecillas del reloj y la variable guarda un -1. (la direccion del movimiento depende de como este conectado el puente H)

```
82
83    int diri = 1;
84    if(u<0){
85      diri = -1;
86    }
87
```

-Ya teniendo estas variables se usan como parametros para la funcion setmotor antes explicada. La cual hara rotar el motor hacia una direccion,con una velocidad deseada hasta llegar al target.

-Por ultimo se guarda el error como error previo para usarlo en el siguiente loop, y se imprime la posicion.

```
57    //PID
58    long currT = micros();
59    float deltaT = ((float) (currT - prevT))/( 1.0e6 );
60    prevT = currT;
61  Serial.print(targeti);
62
63
64    int ei =targeti - posi;
65    Serial.print("  ");
66
67    Serial.print(ei);
68  Serial.print(" ");
69    float dedti = (ei-eprevi)/(deltaT);
70
71    eintegrali = eintegrali + ei*deltaT;
72
73
74    float u = kp*ei + kd*dedti + ki*eintegrali;
75
76
77    float pwri = abs(u);
78    if( pwri > 255 ){
79      pwri = 255;
80    }
81
82
83    int diri = 1;
84    if(u<0){
85      diri = -1;
86    }
87
88
89    setMotor(diri,pwri,PWMi,IN1,IN2);
90
91
92    eprevi = ei;
93
94
95    Serial.println(posi);
96    Serial.println();
```

Monitor en serie

Para el motor derecho se usa la misma explicación que para el izquierdo solo que con diferentes nombres en las variables y diferentes pines.

```
  // DERECHO

 // Objetivo en radianes

 float targetradd =6.28;

 float targetd =targetradd*97.1;
Serial.print("d");
  Serial.print(targetd);

  int ed =targetd - posd;
  Serial.print("  ");
  Serial.print(ed);
Serial.print(" ");

  float dedtd = (ed-eprevd)/(deltaT);

  eintegrald = eintegrald + ed*deltaT;

  float o = kp*ed + kd*dedtd + ki*eintegrald;

  float pwrd = abs(o);
  if( pwrd > 255 ){
    pwrd = 255;
  }

  int dird = 1;
  if(o<0){
    dird = -1;
  }

  setMotor(dird,pwrd,PWMd,IN3,IN4);


  eprevd = ed;


  Serial.println(posd);
  Serial.println();
```

4. Objetivos por cumplir

-Programación de los componentes digitales.

-Realizar la transición del Arduino UNO en uso actualmente hacia el sistema Teensy 3.5.

-Diseño e implementación de un sistema de potencia que administre la carga de la batería a bordo de Pascal.

-Realizar el montaje físico del circuito diseñado y realizar diferentes pruebas para probar su funcionamiento y además realizar los ajustes que se requieran.