

Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ingeniería Mecánica Eléctrica

Laboratorio de Electrónica

Proyecto de Electrónica 4.

Primer semestre 2015

Eddy Alexander Díaz Contreras 2002-18209

Ángel Estuardo Lorenzo Marroquín 2011-13947

Axel Giovanni Arellano García 2009-14915

**Grupo: D69**

**Levitador de Flujo de Aire**

# Introducción

El efecto de que un cuerpo se suspenda en el aire sin contacto físico se denomina levitación, la cual es el resultado de una fuerza que contrarresta el peso del cuerpo u objeto levitante.

La levitación se clasifica según los medios empleados, entre las cuales están, la levitación electrostática, magnética, neumática, acústica y óptica; estas diferentes clases de levitación permiten lograr el mismo efecto de suspender cuerpos en el aire, pero con diferentes características. A partir de lo anterior, este caso de estudio, tratará la levitación neumática; área seleccionada para el desarrollo del modelo en el dominio del tiempo.

Esta clase de levitación opera las variaciones en la presión ejercida por gases, en este caso el aire, para mantener objetos suspendidos en posición estable. Esta levitación debe garantizar los siguientes efectos sobre el objeto: Una fuerza que contrarreste el peso del cuerpo (la fuerza de gravedad que actúa sobre el objeto que levita) y para que se halle en suspensión estable, es necesaria una fuerza adicional que contrarreste cada pequeño desplazamiento del objeto en levitación.

OBJETIVOS

General

Aplicar los conocimientos de semiconductores y electrónica de potencia para crear un levitador de flujo de aire con potencia mínima de 100 Watts.

Específicos

* Utilizar semiconductores para sustituir el uso de Relays en el manejo de medias y altas potencias.
* Utilizar Trías como dispositivo de switcheo y control variable de potencia en circuitos de corriente alterna.
* Realizar el control de circuitos de alta potencia por medio de interfaces de baja potencia.
* Implementar un control inalámbrico por medio de interfaz Bluetooth y sistema operativo Android para expandir el uso del sistema

MARCO TERICO

Un Levitador de flujo de aire es un excelente ejemplo de la aplicación de los conocimientos adquiridos de electrónica 4, en donde se puede implementar algunos conocimientos de electrónica de potencia, empezando por el control de la velocidad angular de un motor de corriente alterna el cual hará el papel del ventilador de aire que contrarreste el peso del cuerpo que levite, debido a que este motor es de corriente alterna en este caso se hace evidente el uso de circuitos de electrónica de potencia para su control, las variables a controlar serán:

* Velocidad del motor.
* 3 Posiciones distintas (altura) del objeto a levitar.
* Disposición del objeto en el Levitador.

*Velocidad del motor:*

La velocidad del motor se controla por medio de un circuito de control de potencia, implementado con circuitos que controlen el Angulo de disparo de Triacs.Por otra parte se controlará cada una de las tres velocidades inalámbrica por medio de un Modulo de comunicación bluetooth, con el fin de hacer su uso más práctico.

*Posición del objeto a levitar:*

Las posiciones a controlar serán tres, las cuales corresponden a tres alturas predeterminadas del objeto a levitary estas dependerán de la velocidad seleccionada en el control de velocidad.

DIAGRAMAS

Diagrama de bloques simplificado

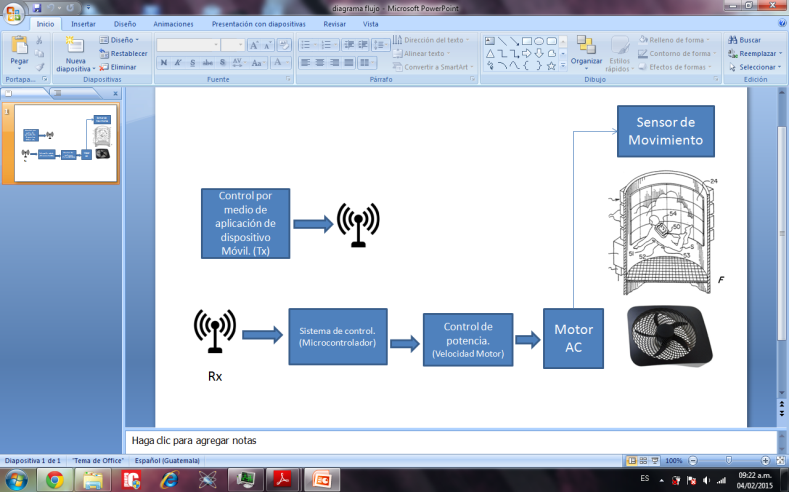


Diagrama de circuito de control de potencia

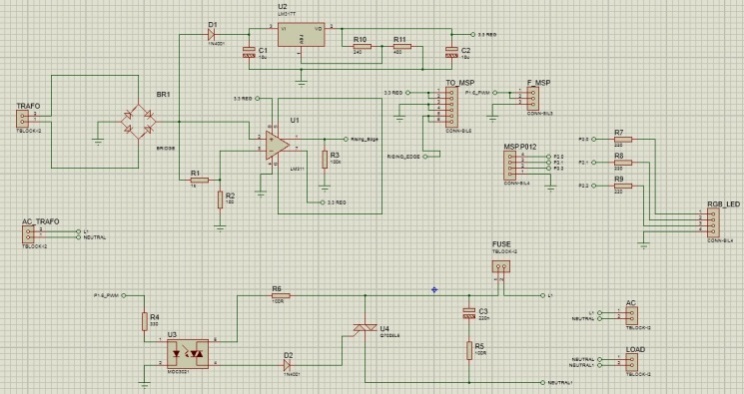
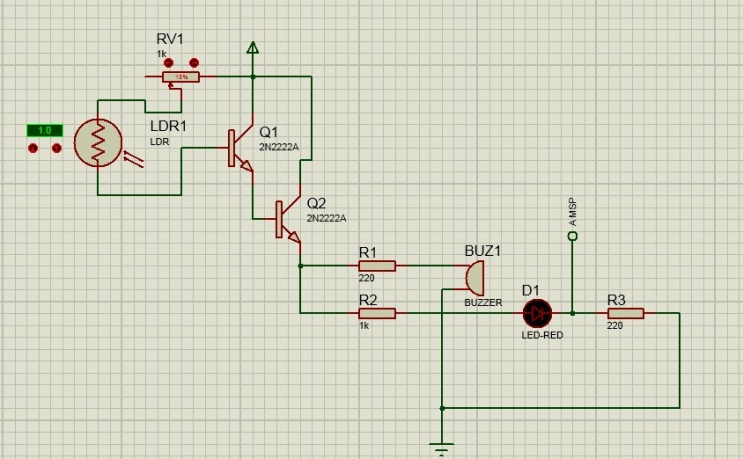


Diagrama de sensor de deteccion de objeto



Conclusiones

* Los dispositivos de potencia ofrecen grandes ventajas sobre los dispositivos electromecanicos como bajo ruido, rapido switcheo, bajo consumo, accionamiento sin chispas, por lo que son sustitutos ideales para para el manejor de altas potencias.
* El Triac permite el control de la potencia de cargas de corriente alterna, mediante el control analogico o digital del angulo de disparo del mismo. Tambien presenta funciones de inversion de giro para motores y regeneracion de energia a fuente según el angulo de disparo controlado.
* Los semiconductores de potencia ofrecen la ventaja de que pueden ser controlados desde dispositvos de baja potencia por medio de optoacopladores y diacs.
* La implementacion de sistemas de control inalambricos aumenta las posibilidades de uso y control de los sistemas de potencia de manera remota asi como amplía las posibilidades de la monitorizacion del funcionamiento de los mismos de acuerdo a las tendencias y necesidades actuales.

ANEXOS

Codigo utilizado para el control del angulo de disparo por medio de la tarjeta de desarrollo MSP430G2553 de Texas Instruments.

#include <msp430g2553.h>

//RGB PINS

unsigned short CURRENT\_RGB;

#define RED BIT0

#define GREEN BIT1

#define BLUE BIT2

//UART PINS

#define RX BIT1

#define TX BIT2

unsigned char Data\_Ready; //VARIABLE USED AS UART FLAG

unsigned char Buffer[1]; //VARIABLE USED TO STORE A RECEIVED DATA

#define RISING\_EDGE BIT4

#define PWMOut BIT6

unsigned short CURRENT\_TACCR1;

const int PWM\_PERIOD = 8000;

/\*

\* main.c

\*/

void ConfigWDT(void);

void ConfigIO(void);

void ConfigClock(void);

void ConfigTimer(void);

void Config\_USCI\_AS\_UART(void);

int main(void) {

ConfigWDT();

ConfigClock();

ConfigIO();

ConfigTimer();

Config\_USCI\_AS\_UART();

\_BIS\_SR(GIE); //Habilitamos las interrupciones generales. IMPORTANTE!!

while(1){

if (Data\_Ready == 1){

switch (Buffer[0]){

case 'H':{

CURRENT\_TACCR1 = 500; // 99% DUTY CYCLE

CURRENT\_RGB = RED;

break;}

case 'M':{

CURRENT\_TACCR1 = 4500; // MORE THAN 50% DUTY CYCLE

CURRENT\_RGB = GREEN;

break;}

case 'L':{

CURRENT\_TACCR1 = 6200; // 25% DUTY CYCLE

CURRENT\_RGB = BLUE;

break;}

case 'O':{

CURRENT\_TACCR1 = 1; // ALMOST 0% DUTY CYCLE, DEVICE OFF!

CURRENT\_RGB = 0x00;

break;}

//default:

//TACCR1 = CURRENT\_TACCR1;

} //SWITCH

Data\_Ready = 0; //DATA RECEIVED, COME BACK TO WHILE(1) AND WAIT FOR ANOTHER INTERRUPT.

}// IF

TACCR1 = CURRENT\_TACCR1;

P2OUT = CURRENT\_RGB;

} //while(1)

}

void ConfigWDT(void){

WDTCTL = WDTPW | WDTHOLD; // Stop watchdog timer

}

void ConfigIO(void){

//P2.x CONFIG

P2DIR |= (RED | GREEN | BLUE); //AS OUTPUTS

P2SEL &= ~(RED | GREEN | BLUE); //AS I/O FUNCTION

P2SEL2 &= ~(RED | GREEN | BLUE); //AS I/O FUNCTION

//BUTTON CONFIG ROUTINE

P1DIR &= ~RISING\_EDGE; //P1.4 AS INPUT

P1DIR |= PWMOut; //P1.6 AS OUTPUT

P1SEL &= ~RISING\_EDGE; //P1.4 AS I/O FUNCTION

P1SEL |= PWMOut | RX | TX; // P1.6 AS TA0.1 | P1.1 AS UCA0RXD | P1.2 AS UCA0TXD

P1SEL2 |= RX | TX; //P1.1 AS UCA0RXD | P1.2 AS UCA0TXD

P1SEL2 &= ~(RISING\_EDGE | PWMOut) ;//P1.4 AS I/O FUNCTION | P1.6 AS TA0.1

P1REN |= RISING\_EDGE; //PULL UP/DOWN RESISTOR ENABLED FOR P1.4

P1OUT &= ~(RISING\_EDGE | PWMOut); //P1.4 IS PULLED DOWN ; BIT = 0: THE PIN IS PULLED DOWN ; BIT = 1: THE PIN IS PULLED UP

//BUTTON INTERRUPT CONFIG

P1IES &= ~RISING\_EDGE; //P1.4 INTERRUPT EDGE IN A LOW TO HIGH TRANSITION

P1IFG &= ~RISING\_EDGE; //NO INTERRUPT IS PENDING

P1IE |= RISING\_EDGE; //P1.4 INTERRUPT ENABLED

}

void ConfigClock(void){

if(CALBC1\_1MHZ == 0xFF || CALDCO\_1MHZ == 0xFF){

P1OUT = BIT0;

while(1);

}

BCSCTL1 = CALBC1\_1MHZ;

DCOCTL = CALDCO\_1MHZ;

}

void ConfigTimer(void){

TACCR0 = PWM\_PERIOD; // PERIOD OF 8ms OR LESS TO CONTROL TRIAC'S GATE.

TACCR1 = 1; // 1% PWM DUTY CYCLE AS INITIAL VALUE.

TACTL |= TASSEL\_2 | ID\_0 | MC\_1 ; //SMCLK AS CLOCK SOURCE | CLOCK DIVIDER PER 1 | COUNT MOD: UP MODE

TACTL |= TAIE; //TA INTERRUPT ENABLED | NO INTERRUPT PENDING

}

void Config\_USCI\_AS\_UART(void){

UCA0CTL1 |= (UCSWRST + UCSSEL\_2); //1)SET USCWRST AND CLOCK SOURCE

//STEPS 2-3

//SELECTING BAUD RATE (SEE TABLE 15-4. COMMONLY USED BAUD RATES, SETTING, AND ERRORS, UCOS16=0) ON PAGE 424 DATASHEET

//IN THIS CASE WE USE THE CONFIGURATION AS CLOCK SOURCE 1MHZ. AND A BAUDRATE OF 9600KBPS

UCA0BR0 = 104;

UCA0BR1 = 0; //THIS IS ZERO AS A RESULT OF THE VALUE ON UCA0BR0 STILL IS A VALUE BETWEEN 0 AND 255 (8\_BITS)

//THIS VALUE MUST BE DIFFERENT IF THIS IS A NUMBER GREATER THAN 256. (16\_BIT VALUE).

//(IN A NUTSHELL THIS IS AN EXTENSION FOR A 16\_BIT VALUE, IF EXIST)

UCA0MCTL = UCBRS\_1; //THIS VALUE MUST BE SET, AS LONG AS WE HAVE A CLOCK SOURCE OF 1\_MHZ.

//4) CLEARING UCSWRST BY SOFTWARE

UCA0CTL1 &= ~(UCSWRST);

//5) ENABLE INTERRUPTS

IE2 |= UCA0RXIE; //RX INTERRUPT ENABLED

IFG2 &= ~(UCA0RXIFG); //NO INTERRUPT PENDING

} //END Config\_USCI\_AS\_UART(void)

//PORT1 INTERRUPT SERVICE ROUTINE

#pragma vector = PORT1\_VECTOR

\_\_interrupt void P1\_ISR(void){

TAR = 0; //AN INTERRUPT ON P1.x WILL LEAD TO RESET THE COUNTER REGISTER.

switch(P1IFG & RISING\_EDGE) {

case RISING\_EDGE:

P1IFG &= ~RISING\_EDGE; //INTERRUPT FLAG ON P1.4 CLEARED

TACCTL1 = OUTMOD\_0; //SET TA0.1 TO OUTPUT MODE

TACCTL1 &= ~OUT; //SET TA0.1 LOW

TACCTL1 = OUTMOD\_1; // RESET AT CCR0 = 0 AND SETS AT VALUE ON TACCR1

return;

default:

P1IFG = 0x00; //NO INTERRUPTS PENDING, BACK TO MAIN();

return;

}

}

//TIMER ISR

#pragma vector = TIMER0\_A1\_VECTOR

\_\_interrupt void TA0\_Rollover\_ISR(void){

switch(TAIV){ //TIMER\_A INTERRUPT VECTOR VALUE TAIV. FOR MORE INFO. SEE PAG 373 OF uC DATASHEET

case(0x0A): //WHEN AN OVERFLOW AT UP MODE OCCOURS EXECUTE THE CODE BELOW

TACTL &= ~TAIFG; //TAIFG CLEARED

TACCTL1 = OUTMOD\_0; //SET TA0.1 TO OUTPUT MODE

TACCTL1 &= ~OUT; //SET TA0.1 LOW

TACTL |= MC\_0; //TIMER HALTED

TAR = 0; //RESET COUNTER REGISTER

//TACCR1 = 0;

return;

default:

TACTL &= ~TAIFG; //NO INTERRUPTS PENDING, COME BACK TO MAIN();

return;

}

}

//USCIAB0RX ISR

#pragma vector = USCIAB0RX\_VECTOR

\_\_interrupt void USCI0RX\_ISR(void){

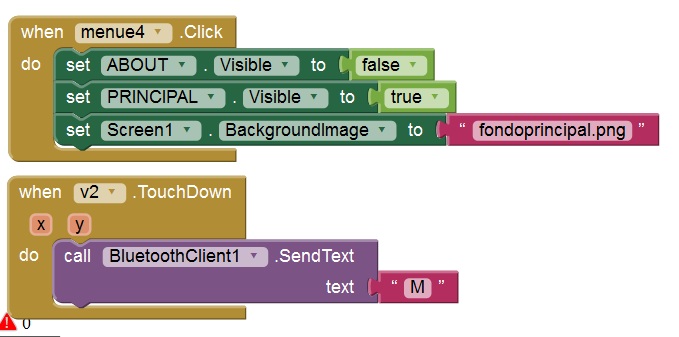
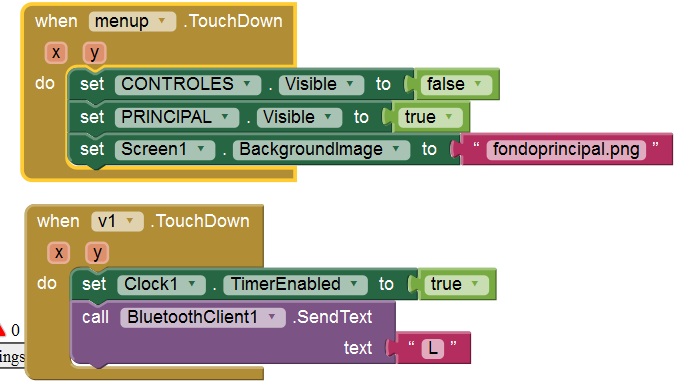
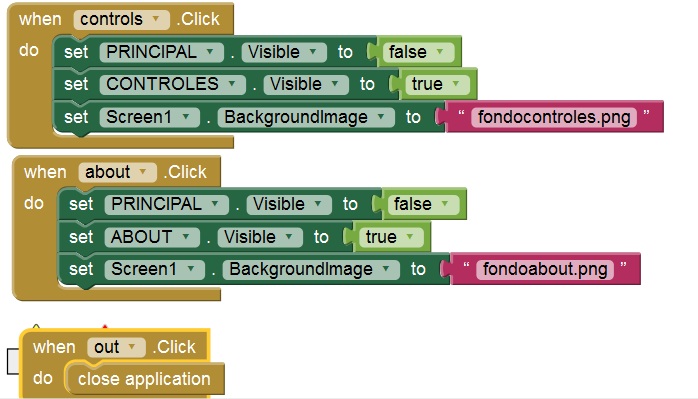
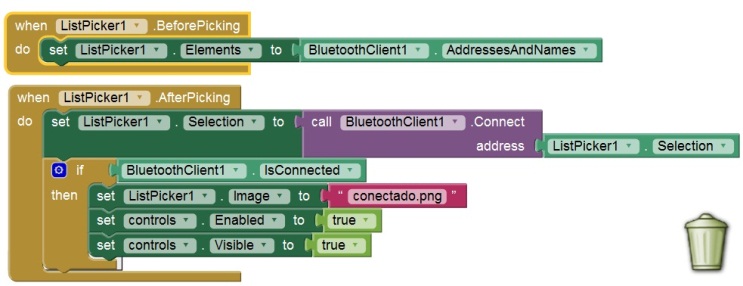
Buffer[0] = UCA0RXBUF; //STORE THE VALUE RECEIVED (EACH DATA RECEIVED WILL TRIGGER AN INTERRUPT)

Data\_Ready = 1; // THE DATA RECEIVED IS READY TO BE READ

IFG2 &= ~(UCA0RXIFG); //NO INTERRUPT IS PENDING, BACK TO MAIN();

}

**CODIGO DE APP INVENTOR DE LA APLICACIÓN PARA ANDROID**



VENTANAS DE LA APLICACIÓN PARA ANDROID



