UNIVERSIDAD DE ALCALÁ ESCUELA POLITÉCNICA SUPERIOR DEPARTAMENTO DE ELECTRÓNICA		GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES	
ASIGNATURA	SISTEMAS ELECTRÓNICOS DIGITALES AVANZADOS	FECHA	MARZO 2017
APELLIDOS, NOMBRE	SOLUCIÓN	GRUPO	

PRUEBA DE EVALUACIÓN INTERMEDIA 1

Ejercicio 1

El diagrama de bloques de figura 1 muestra un sistema de control a distancia por infrarrojos de servomotor basado en el LPC1768 (Fcpu=100Mhz). El sistema consta de un módulo transmisor y un módulo receptor. El módulo transmisor (TX) dispone de un potenciómetro que permite controlar la posición del servomotor conectado al módulo receptor (RX) en un rango de 0 a 180º variando su recorrido.

El sistema de codificación de infrarrojos se muestra en el Anexo I. Tenga en cuenta que la portadora es una señal periódica cuadrada de 38kHz (zona pintada en negro de la señal). La posición del potenciómetro se codifica con 7 bits y se transmite periódicamente cada 50ms comenzando por el LSB.

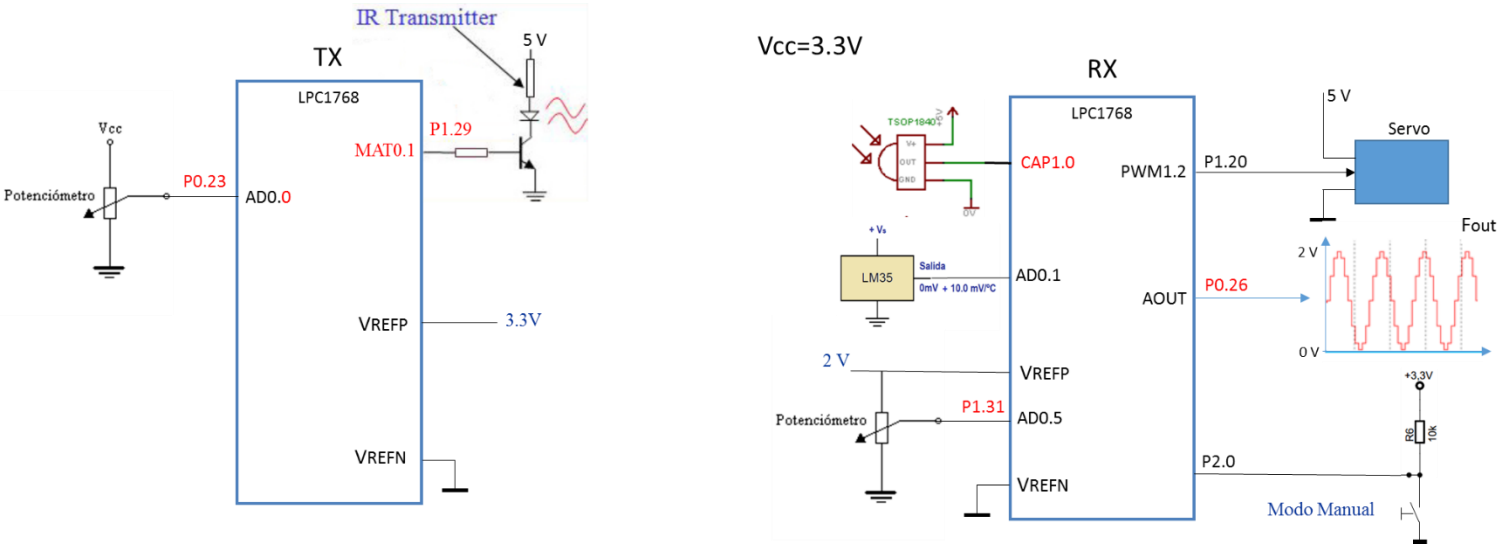


Figura 1. Diagrama de bloques del sistema

El módulo receptor (RX) se encarga de decodificar la señal infrarroja y posicionar el servo en función del código de 7 bits recibido. El potenciómetro conectado a AD0.5 (Modo manual) permite mover el servo en su recorrido (0-180º) mientras se mantiene cerrado el pulsador conectado a P2.0, a la vez que se varía la frecuencia obtenida a la salida del DAC entre 100 y 1 kHz. **En este modo no se tiene en cuenta el código IR recibido.**

Un sensor de temperatura (LM35) adosado al servo nos permite monitorizar su temperatura cada segundo.

a) Complete sobre el diagrama de la figura1, a medida que contesta a los siguientes apartados, el nombre del pin (**Pn.x**) sobre la línea de conexión y el nombre del recurso utilizado dentro del bloque que representa el LPC1768 (ej. **MAT1.0**).

Módulo Transmisor (TX)

b) Complete la función de configuración del ADC (incluyendo los comentarios) para la entrada a la que se conecta el potenciómetro y calcule **tiempo de conversión** para la **mínima frecuencia de reloj** posible del ADC. Escriba en **pseudocódigo** la función de configuración del Timer utilizado para generar una frecuencia de muestreo de **20Hz**. Considere que el ADC funciona por interrupción.

```
void init_ADC_potencimetro(void)
{
LPC_SC->PCONP|= (1<<12);              // Power ON
LPC_PINCON->PINSEL1|= (1<<14  );      // ADC input P0.23 (AD0.0)*
LPC_PINCON->PINMODE1|=(2<<14  );      // Deshabilita pullup/pulldown
LPC_ADC->ADCR= (   1<<0)|             // Canal 0 *
               (0xFF<<8)|             // Fclk=25e6/256= 97,656kHz
                 (1<<21)|            //  PDN=1
                (6<<24);             //  MAT1.0 -> frecuencia de muestreo
LPC_ADC->ADINTEN=(1<<0);             // Interrumpe canal 0 cada 50ms
NVIC_EnableIRQ(ADC_IRQn);
}
```

Tconv=65*1/97,656Khz = 0,66ms

```
void init_TIMER_1 (void)
{
Power ON Timer1
Reset on Match 0
MR0=25e6/20/2-1; // Cada 2 Match inicio de conversión (Fs=20Hz)
Start Timer
Sin interrupción
}
```

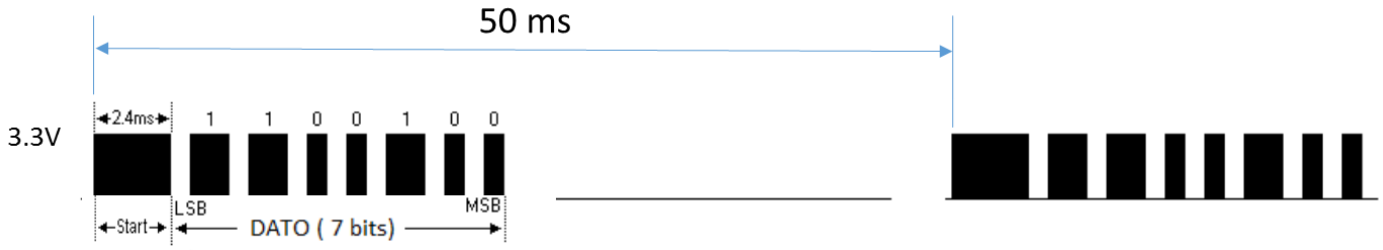
c) Complete las líneas de la función de interrupción del ADC para guardar en una variable global el código de 7 bits correspondiente a la posición del potenciómetro.

```
void ADC_IRQHandler(void)
{
   codigo_tx=(LPC_ADC->ADDR0>>9)&0x7F; // Convertimos a 7 bits
   LPC_TIM2->TCR=0x01; // Start Timer 2 para generar la señal modulada por MAT0.1
   LPC_TIM0->TCR=0x01; // Start Timer 0 para generar la portadora
}
```

d) Explique sin escribir el código la configuración del **Timer 0** que genera exclusivamente la portadora de 38kHz.

Timer 0 en modo salida por comparación (MAT0.1)
Habilitar salida MAT0.1 (P1.29) en PINSELx
Habilitamos el modo Toggle.
MR1=25e6/38e3/2 -1;
Pin MAT0.1 =0 (manualmente en EMR)
Sin interrupción!!!
Stop Timer (por defecto)

e) Escriba en código o pseudocódigo la función de interrupción del **Timer 2** que permite generar la señal modulada que excita el transistor (ver Anexo I)



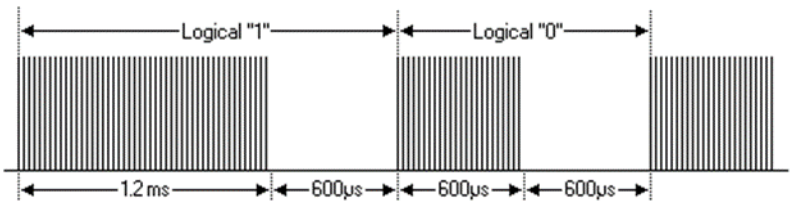
Timer 2 en modo Match (MR0)
Prescaler Timer2 =24 → Ttick= 1 microsegundo
Modificamos MR0 dinámicamente para ir generando los tiempos 4T (START) , 2T y 1T (T= 600us)

```
void Timer2_IRQHandler(void)
{
static char estado;
switch(estado){
            case 0:
            LPC_TIMER0->TCR=0x02;          // Stop Timer (Portadora OFF)
            LPC_TIMER0->EMR&=~(1<<1);      // MAT0.1 =0 (garantizamos un nivel bajo)
            LPC_TIMER2->MR0=600;           // Match= 1T
            estado=1;
            break;

            case 1:
            cuenta++;
            if(cuenta%2){                 // valor cuenta impar

                        LPC_TIMER0->TCR=0x01;                       // Portadora ON
                        if (tx_codigo&0x01) LPC_TIMER2->MR0=1200;// Match= 2T
                        else LPC_TIMER2->MR0=600;                   // Match= 1T
                        tx_codigo>>=1;                              //siguiente bit
                        }
            else {
                  LPC_TIMER0->TCR=0x02;          // Stop Timer (Portadora OFF)
                  LPC_TIMER2->EMR&=~(1<<1);      // MAT0.1 =0 (garantizamos un nivel bajo)
                  LPC_TIMER2->MR0=600;           // Match=1T
                  }

            if(cuenta==7*2)                      // Fin TX del código de 7 bits
                        {
                           LPC_TIMER0->TCR=0x02; // Stop Timer 0 (Portadora OFF)
                           LPC_TIMER2->TCR=0x02; // Stop Timer 2
                           estado=0;
                           cuenta=0;
                           LPC_TIMER2->MR0=Fpclk*2400; // Preparo MR0 para 4T (START)
                        }
            break;
            }
      }
}
```



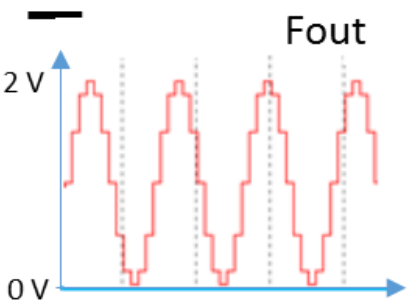
Módulo Receptor (RX)

f) Complete la función de inicialización del ADC e indique justificadamente la frecuencia de muestreo de cada canal.

```
void init_ADC(void)
{
LPC_SC->PCONP|= (1<<12);                  // Power ON
LPC_PINCON->PINSEL1 |= (1<<16    );      // P0.24 es AD0.1
LPC_PINCON->PINSEL3 |= (3<<30    );      // P1.31 es AD0.5
LPC_PINCON->PINMODE1 |= (2<<16    );     // Deshabilita pullup/pulldown
LPC_PINCON->PINMODE3 |= (2<<30    );     // Deshabilita pullup/pulldown
LPC_ADC->ADCR= (0b00100010<<0 )|          // Canales 1 y 5
                         (24<<8)|         // Fclk=25e6/25 = 1Mhz
                         (1<<21)|         // PDN=1
                         (1<<16);         // Modo BURST
LPC_ADC-ADINTEN=0;                       // Sin interrupción
}
```

Fs= 1Mhz/(65*2)= 7692Hz

g) A la vista de señal obtenida a la salida del DAC del módulo receptor, complete la función que genera las muestras de 8 bits de la función seno. Considere declarado el array *uint8_t muestras[ ]*




```
void genera_muestras_seno(void)
{
char i;
for(i=0;i<12  ;i++) muestras[i]=(uint8_t)( 127 + 127*sin(2*pi*i/12  );
}
```

NOTA: Vrefp= 2V.

12 Muestras/ciclo (ver figura) codificadas en 8 bits a partir del tamaño del array

h) Escriba la función de interrupción del **Timer 3** que saca las muestras hacia el DAC para generar la señal senoidal.

```
void TIMER3_IRQHandler(void)
{
LPC_TIM3->IR|=(1<<0);           // Clear flag MR0
LPC_ADC_DACR=muestras[i++]<<8; // DAC 8 bits
if(i==12)i=0;



}
```

i) Explique en detalle, sin escribir código, qué recurso utilizaría y como lo configuraría para reducir la carga de CPU durante la generación de la señal senoidal.

Evitaríamos la ejecución de la interrupción del Timer3 empleando el DMA en modo Linked con el propio canal (ej. Canal 0), para que al finalizar la transferencia se inicie de nuevo sin que sea necesaria la interrupción del DMA.

LPC_GPDMACH0->DMACCSrcAddr = (uint32_t) &muestras[0]; // Fuente Memoria
LPC_GPDMACH0->DMACCDestAddr = (uint32_t) &(LPC_DAC->DACR) **+ 1**; //Destino Periférico (ojo 8bits)
Transfer Size=**12**; Tamaño transferencia= **8 bits**; Incrementa Fuente; No incremento Destino
Sin interrupción (ojo, modo Linked)

Para modificar la Fout:
LPC_DAC->DACCNTVAL = (F_pclk/12/Fout) -1; // Frecuencia transf. DMA

j) Escriba el código de la función de interrupción del SYSTICK (periodo 50ms) que se encarga de leer el estado del pulsador y de modificar la posición del servo en función del potenciómetro, a la vez que modifica el valor de la frecuencia de salida del DAC, así como de almacenar en una variable global la temperatura. Considere que la señal PWM del servo ya está configurada y para un periodo de **15ms**, y que el tiempo a nivel alto varíe entre **0.8-2.4ms**, para un movimiento de su posición entre 0º y 180º.

```
void SysTick_IRQHandler(void)
{
contador++;
if((LPC_GPIO2->FIOPIN&0x01)==0){

        if(codigo|=(LPC_ADC->ADDR5>>8)&0xFF){

              LPC_PWM1->MR2=(uint32_t)(F_pclk*0.8e-3 + F_pclk*1.6e-3*codigo/255);
              LPC_PWM1->LER|=(1<<2)|(1<<0);
              Fout= 100 + 900*codigo/255;              // Fout del DAC
              LPC_TIMER3->MR0=Fpclk/Fout/12;           // 12 muestras/ciclo
              LPC_TIMER3->TC=0;
              LPC_TIMER3->TCR=0x01;                    // Start Timer
              }

}
else LPC_TIMER3->TCR=0x02;                             // Stop Timer

codigo=(LPC_ADC->ADDR5>>8)&0xFF;


//cada segundo leo la temperatura
if(contador%20==0) temperatura=((LPC_ADC->ADDR1>>4)&0xFFF)*200/4095; // grados=mV/10



}
```

ANEXO I (Codificación – Decodificación IR)