Universidad del Valle de Guatemala
Digital 2 - Laboratorio
Kurt Kellner
Pablo Rene Arellano Estrada
151379

MINI-PROYECTO 2
-I2C con MPU9250-

ARCHIVOS Y LIBRERIAS .C Y .H:

PIC	ARDUINO	ESP32
PIC_MPU_6050.C	Serial_Arduino.cc	Adaruitio_07_Digital_out.cc Config.h
Configuracion_Header_File.H		Adaruitio_07_Publish.cc Config.h
MPU6050_res_define.H		
I2C_Master_File.C LCD_SPI.H I2C_Source_File.C		
USART_Header_File.H USART_Source_FileI.C		

OBSERVACIONES:

- No se pudo encontrar el manual del sensor MPU9250, solo del 6500, por lo que se usaron los registros encontrados en este manual, y en la pagina de KHALED MAGDY.
- La comunicación serial simultanea entre el PIC y el ESP32 no fue posible, por lo que se transmitieron datos del PIC al ESP32 con UART, pero para los leds se conectaron pines del ESP32 al PIC para efectos de demostración usando el regulador de voltaje.
- Adaruit no permitió enviar y recibir datos de manera razonable con los botones y los medidores, por lo que se hicieron en archivos separados y en momento separados para evitar los 30 cambios por minuto en los Feeds.

PSEUDOCODIGO-MAESTRO:

1. Se llama a las librerias correspondientes:
    - xc.h
    - Variables de ancho definido stdint.h
    - Tipos de variables, macros, entradas y salidas stdio.h
    - Libreria de comunicacion I2C
    - Libreria de comunicacion UART
    - Libreria de registros MPU
2. Genera interrupcion
    - Si bit de boton uno esta en alto
        - Enciende led 1
    - Si bit de boton uno esta bajo
        - Apaga led 1
    - Si bit de boton dos esta en alto
        - Enciende led 2
    - Si bit de boton dos esta en bajo
        - Apaga led 2
3. Configuracion
    - Buffer de 40 bytes
    - Configuracion puertos, oscilador, timer, interrupciones
    - Inicio de I2C, MPU9200, UART
4. Iniciar en direccion de Aceleracion X, y con I2C
    - Leer Byte Alto
    - Leer Byte Bajo
    - Unir Bytes
    - Repetir hasta leer todas las variables
    - Envair AKN y apagar transmision
5. Realizar conversion de variables
    - Para este caso Ax / 4;
    - Enviar buffer entero con coma
    - Enviar string a buffer de UART
6. Configurar MPU9250
    - Configurar frecuencia
        - Empezar en direccion 0XD0
        - Escribir direccion SMPLRT_DIV
        - Escribir valor a direccion
        - Detener comunicacion
    - Configurar administrador de energia
        - Empezar en direccion 0XD0
        - Escribir direccion PWR_MGMT_1
        - Escribir valor a direccion
        - Detener comunicacion
    - Configurar Giroscopio
        - Empezar en direccion 0XD0
        - Escribir direccion GYRO_CONFIG
        - Escribir valor a direccion
        - Detener comunicacion
    - Configurar Interrupciones
        - Empezar en direccion 0XD0
        - Escribir direccion INT_ENABLE
        - Escribir valor a direccion
        - Detener comunicacion

7. Empezar en direccion de variables de lectura
    - Configurar Interrupciones
        - Empezar en direccion 0XD0
        - Escribir direccion **57**
        - Realizar un stop
        - Borrar valor dummie

LIBRERIA I2C:

1. Funcion Init ()
    - Colocar bits de inicio de lectura en RC3 y RC4
    - Frecuencia a (xtal/(4*c))-1
    - SSPCON2 = 0 para no modificar nada
    - SSPIE = 1 que permite interrupcion
    - SSPIF = 0 bandera apagada
2. Funcion Start
    - Envia condicion inicial SEN = 1
    - Mientras se de la condicino inicial
    - Se prueba si comunicacion es exitosa con SSPSTATbits.S
    - Se indica la direccion al esclavo
3. Funcion Start Wait
    - Envia condicion inicial SEN = 1
    - Mientras se de la condicino inicial
    - Si el bit inicial no fue detectado regresa
    - Se Indica la direccion del esclavo para escribirle
4. Funcion Repeat Start
    - Envia condicion inicial RSEN = 1
    - Mientras se de la condicino inicial
    - Se apaga la bandera
    - Se Indica la direccion del esclavo para escribirle
5. Funcion Write
    - Se escribe dato a SSPBUF
    - Se verifica que no haya interrupciones
    - Se determina si se recibe o no AKN
6. Funcion AKN
    - ACKDT = 0
    - Se habilita ACKEN = 1 para enviar
7. Funcion NAKN
    - ACKDT = 0
    - Se habilita ACKEN = 1 para enviar
8. Funcion Read
    - Habilita recepcion con RCEN = 1
    - Espera a que el buffer este lleno
    - Leer valor con SSPBUF
    - Se envia o no el AKN
9. Funcion Stop
    - Se inicializa condicion de parade con PEN = 1
    - Mientras este la condicion de parade se apaga interrupcion
    - Falla el Stop? Se regresa a 0
10. Funcion Ready
    - Mientras se apaga bandera de interrupcion
    - SSPIF=0

LIBRERIA UART:

1. Funcion USART inicial
    - Tx para transmitir datos
    - Rx para recibir
    - SPBRG = (4MHz/(16*9600))-1 = 25
    - Se habilitan transmission con TXSTA
        - TX9 = 0 ya que es solo para 8 bits
        - TXEN = 1 para permitir transmision
        - SYNC = 0 para comunicacion Asincrona
        - BRGH = 0 para alta velocidad
    - Se habilita recepcion con RCSTA
2. Funcion Transmitir Char
    - El buffer esta vacio?
    - Se envia dato a TXREG
3. Funcion Recibir Char
    - Hay error? Con OERR
    - Apagar modulo para apagar error
    - CREN = 1 para permitir recibir
    - Se envia valor a RCREG
4. Funcion Transmitir String
    - Mientras cadena de Bytes no este vacia?
    - Se agregan bytes
5. Funcion Delay de 10ms
    - Para un valor de 1 a 150 se realiza un conteo inicial
    - Para un valor de 1 a 165 se realiza un conteo final

ESP32-PUBLICAR A ADAFRUIT:

1. Inicializar feed de para enviar con *Az_feed = io.feed("Az")
    - Declarar variables de aceleracion enteras
    - Decalrar variables tipo String de las recibidas de PIC
2. Comunicacion Serial:
    - Empezar comunicacion serial con Serial.begin(9600);
    - Esperar a que el monitor se abra
    - Se conecta a io.adafruit.com con io.connect();
    - Se espera conexion y se imprimen puntos de prueba con io.status() < AIO_CONNECTED
    - Se determina si la conexion fue exitosa
3. Loop Principal
    - Si hay caracter?
    - Se lee en entero del Rx con parseInt
    - Se realiza conversion a Sistema metrico
    - Conexion con Adaruit.com con io.run();
    - Se imprime valor leido de UART en monitor
    - Se envia valor a Adafruit.com con Az_feed->save(Az);
    - Delay para evitar mas de 30 eventos por minuto
4. Configuracion IO Key y Username
    - #define IO_USERNAME "pabloarellanoestrada"
    - #define IO_KEY   "aio_axLC941OvDyf8Zoxc7rRY8Y43zK2

ESP32-LEER BOTONES:

1. Inicializar feed de para enviar a PIC para botones
    - Led_1Feed = io.feed("Led_1");
    - Led_2Feed = io.feed("Led_2");
    - Decalrar variables tipo byte para enviar por TX
    - Declarar variables para puertos
2. Comunicacion Serial:
    - Puertos de salida con OUTPUT
    - Empezar comunicacion serial con Serial.begin(9600);
    - Esperar a que el monitor se abra
    - Se conecta a io.adafruit.com con io.connect();
    - Funciones que reciben feeds para verificar cambios en los botones
    - Se espera conexion y se imprimen puntos de prueba con io.status() < AIO_CONNECTED
    - Se determina si la conexion fue exitosa
    - Se reciben valores con get()
3. Loop Principal
    - Conexion con Adaruit.com con io.run();
4. Funcion para Cambios de Botones (dos veces)
    - Se imprime valor de Adafruit para verificar
    - Se recibe valor con data->value()
    - Si boton uno esta en alto?
        - Se imprime High
        - Se pone el Pin2 en High
        - Se imprime uno com Byte
    - Si boton uno esta en bajo?
        - Se imprime Low
        - Se pone el Pin2 en Low
        - Se imprime cero com Byte
    - Se imprime valor en monitor.
5. Configuracion IO Key y Username
    - #define IO_USERNAME "pabloarellanoestrada"
    - #define IO_KEY   "aio_axLC941OvDyf8Zoxc7rRY8Y43zK2

DIAGRAMA DE FLUJO:

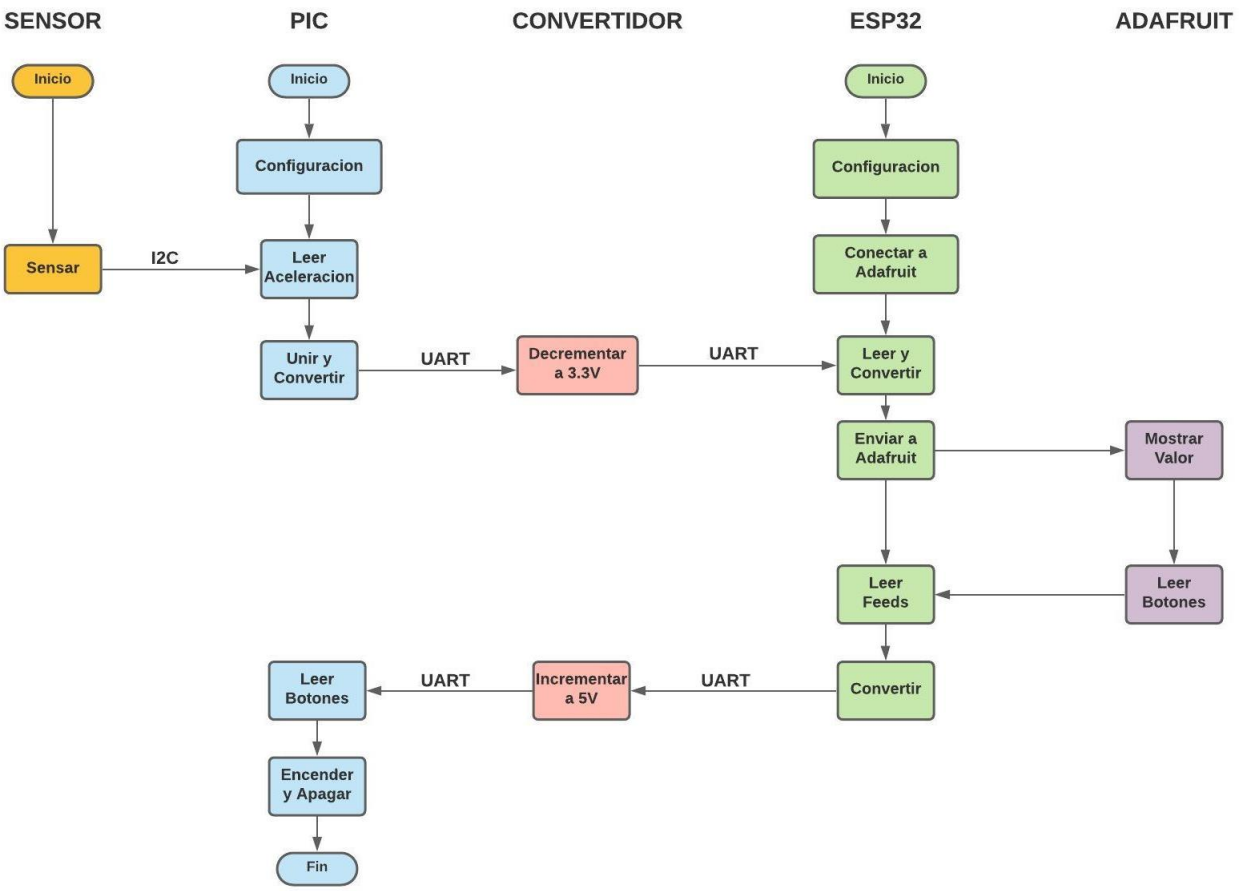

DIAGRAMA FISICO: