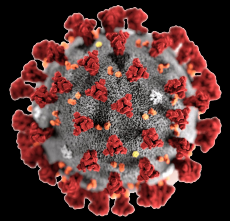
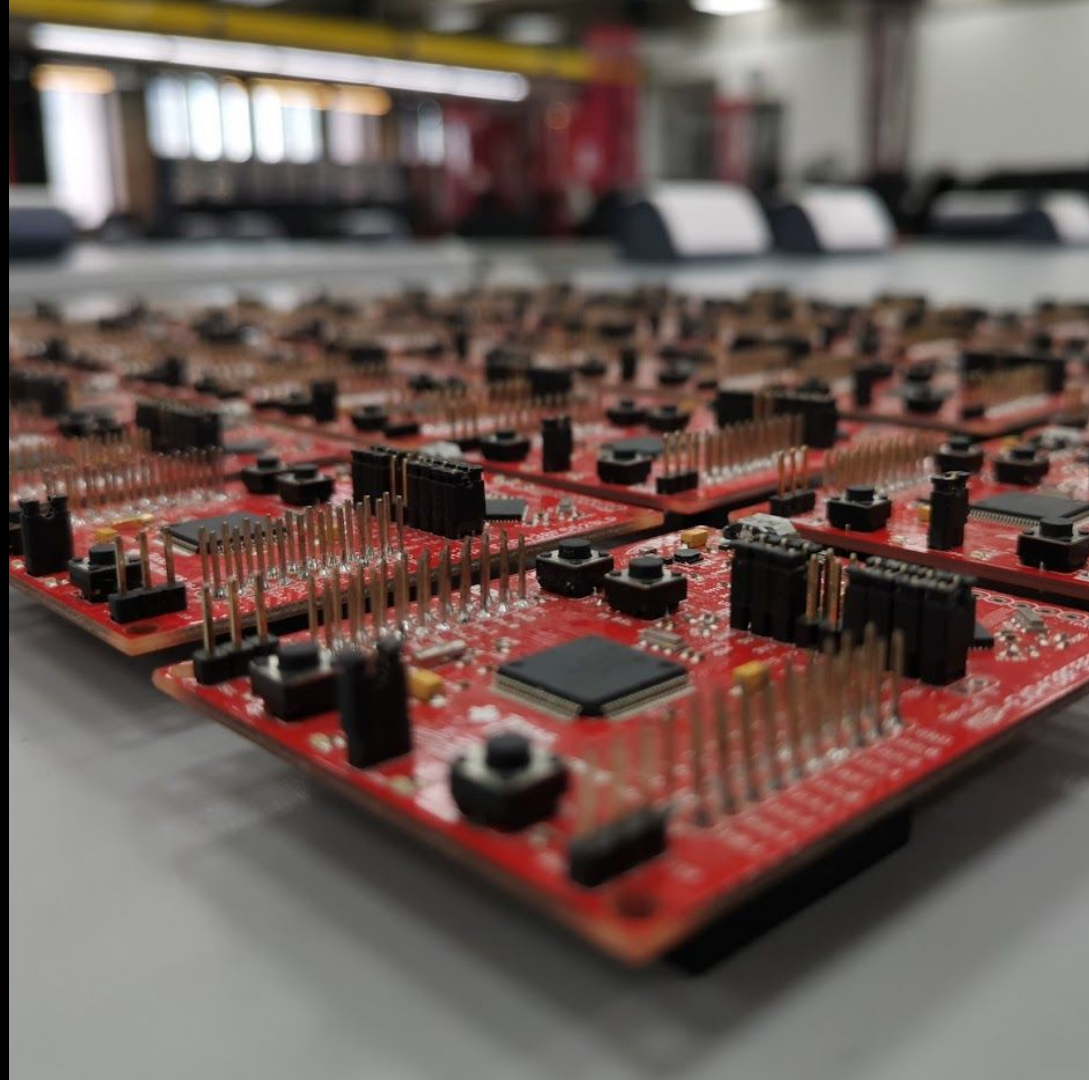
IEE2463 Sistemas Electrónicos Programables

Plan de Acción



1er semestre 2020



Aspectos Administrativos

Evaluaciones

- Laboratorios
  - Existen **5 niveles** de dificultad para los laboratorios, por lo que cada uno tendrá una ponderación equivalente a esto.
  - **Total 9 Labs**
  - La nota de cada uno estará dada por una **rúbrica de evaluación**.
  - Nota Lab calculada como:

$$L = \frac{1}{\text{Suma Ponderacion Total}} \sum_{i=1}^{N_L} \text{Nota Lab}_i \times \text{Ponderacion Lab}_i$$

- Además, se debe cumplir con:
  - Se debe lograr **al menos un 60 %** de logro en el **80 % de los laboratorios**.
    - **Significa un máximo de 2 Labs menor a 4.0**
  - Nota de laboratorio igual o superior a **4.50**

Proyecto Final

- Programación y diseño de una aplicación utilizando un microcontrolador por definir, con entradas/salidas y procesamiento local y/o remoto.
- Cambia todos los semestres
- Enunciado a publicarse la primera semana de Junio.
- Fechas y entregas por definir.
- Trabajo individual o grupal.

Recomendaciones

Normas y recomendaciones

- No tomen atajos!!!
  - Un entendimiento profundo de la arquitectura y funcionamiento de un microprocesador es preferible a una solución copiada de la web y que no saben cómo o bajo qué condiciones funciona.
- Trabajen con anticipación y sean constantes.
- Aprovechen el foro (GitHub)
  - No teman hacer preguntas.
    - Si ustedes tienen dudas, seguramente otros alumnos también las tendrán
  - Sean generosos y compartan sus soluciones para la configuración o utilización del software en diferentes plataformas.
- Normas para trabajo colaborativo
  - Está bien discutir ideas de implementación con otros alumnos.
  - **No compartan código -- habrá sanción para quienes violen esta norma!!!**

¿Qué se puede y qué no se puede hacer ?

¿Qué se puede y qué no se puede hacer ?

Laboratorios

¿Qué sucede de ahora en adelante?

- TinkerCad no es la mejor herramienta para seguir con el curso:
  - Falla en registros.
  - Inconsistencia a la hora de programación.
  - Tiempo de simulación.
- Las experiencias prácticas **no se eliminan**:
  - Serán relocalizadas temporalmente hasta que el semestre lo permita.
- Los días lunes serán utilizados para el desarrollo de clases / ayudantías con enfoque práctico.
- Ejemplos y aplicaciones en:
  - C (uso de compilador)
  - MSP430F5529
  - Atmega328P

¿Qué significa clases / ayudantía con enfoque práctico?

- No son ayudantías de los contenidos de cátedras.
- Se verá programación de microcontroladores.
- Cuidados a la hora de trabajar.
- Buscar información en datasheets.
- Planificar y estructurar una solución.
- Problemas típicos y sus soluciones.

Fechas

- 4 Mayo: Introducción al uso de microcontroladores
- 11 Mayo:
  - Uso y aplicaciones de Timers
  - Publicación Lab 04
- 18 Mayo: Uso de punteros y estructuras de datos
- 25 Mayo: Revisión Lab 04
- 1 Junio: Protocolos de comunicación, UART
- 8 Junio: Protocolos de comunicación, I2C y SPI
- 15 Junio: Recapitulación protocolos de comunicación
- 22 Junio: Recapitulación y diseño

Presentación ayudantes

Ayudantes

Camila Turrieta

Francisco Fonseca

José Gutiérrez

Fernando Huanca

Javier Diaz

Mayron Barahona

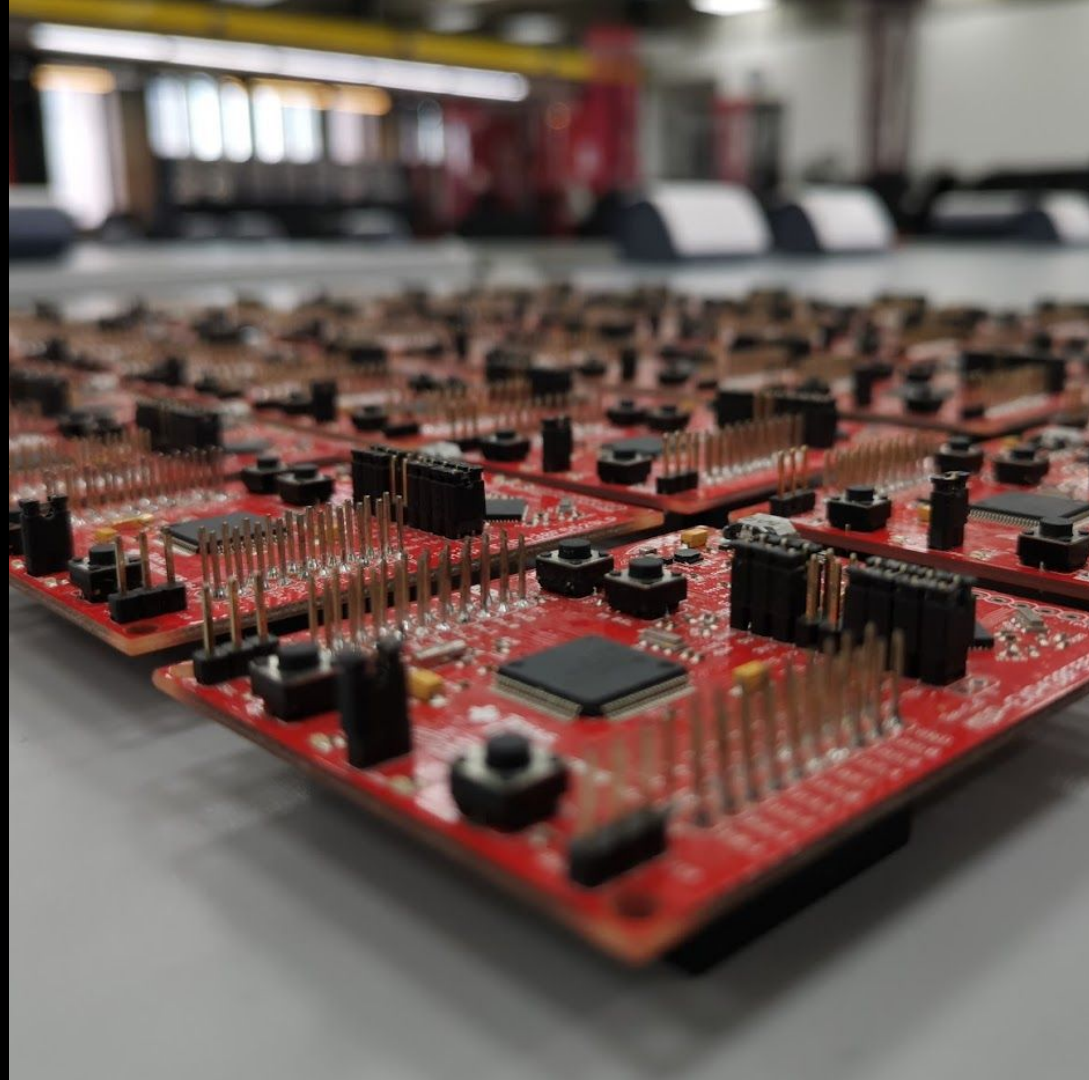
Pablo Orellana

¿Dudas?

IEE2463 Sistemas Electrónicos Programables

Introducción al uso de microcontroladores

1er semestre 2020

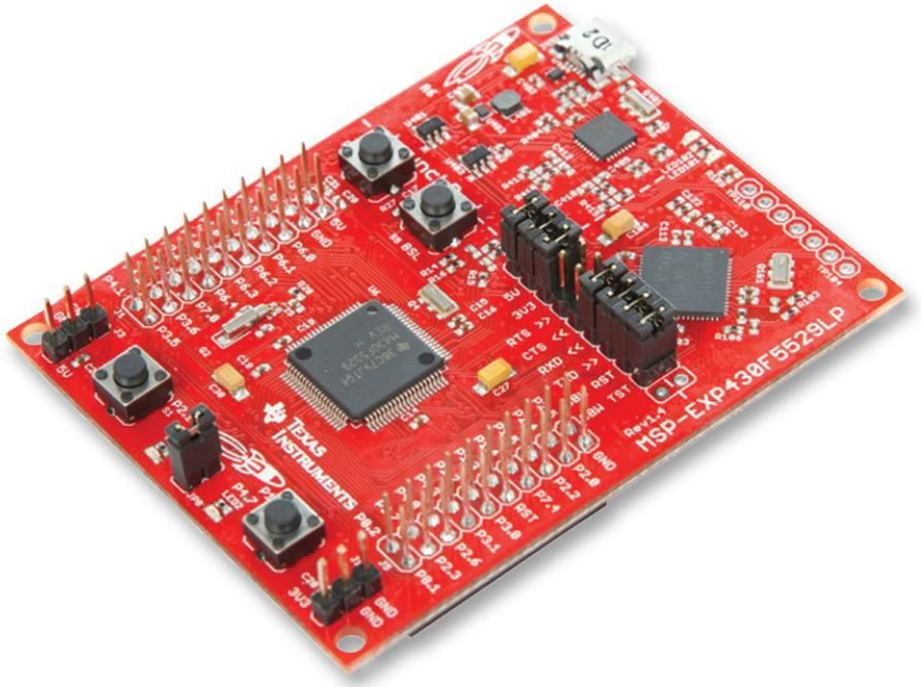


Atmega328P

Consideraciones

- Evitar programar si está conectado el puerto PD0 o PD1.
- A veces es necesario presionar reset.
- No siempre el computador lo detecta.
- Leer los warnings y errores del compilador.
- No toque Vcc y GND al mismo tiempo.
- Tomar placa siempre por los bordes.

MSP430F5529LP



Consideraciones

- A veces es necesario resetear para que funcione.
- Ojo con el modo debug y modo flash.
- No desconectar mientras se está programando.
- Cuidado con los jumpers blocks de la placa.
- No toque Vcc y GND al mismo tiempo.
- Tomar placa siempre por los bordes.

Precauciones

- Evitar tener manos mojadas.
- Cuidado con las mascotas.
- No programar en la cama.
- Idealmente descargar estática antes de tocarlos.
- Evitar juntar Vcc con GND (su PC podría sufrir las consecuencias).
- Revisar conexiones.

Aclaraciones laboratorios anteriores

LAB 03 ADC (main)

```c
/*
 * name: Ayudantia #1 ADC y PWM
 *
 * author:
 *      Camila Turrieta
 *
 * Microcontroller:
 *      ATMEGA328P
 */

#define F_CPU 16000000 //16MHz
#include <avr/io.h>
#include <util/delay.h>

void Config_ADC(void);
void Config_Timer(void);
void ADC_conversion(void);

int main(void)
{
    Config_ADC();
    Config_Timer();

    //Configuracion pines: output pwm
    DDRD   |=  (1 << DDD6);

    //Pin para debuguear
    DDRD   |=  (1 << DDD5);
    PORTD  |=  (1 << PORTD5);

    //Variables auxiliares
    int duty = 0;
    float ADC_value = 0.0;
    float aux = 0.0;

```

```c
/*
 * name: Ayudantia #1 ADC y PWM
 * Author:
 *     Camila Turrieta
 *     Felipe Sánchez
 *
 * Microcontroller:
 *      MSP430F5529
 * Copyright (c) 2012, Texas Instruments Incorporated
 *
 */

#include<msp430.h>
#include<stdlib.h>

void timer_A(void);
void config_adc(void);
int start_conversion(void);

void main (void)
{
    // WatchDog timer
    WDTCTL = WDTPW | WDTHOLD;

    config_adc();
    timer_A();

    // Debugueo
    P1DIR |= BIT3;
    P1OUT &= ~(BIT3);

    // PWM
    P1DIR |= BIT2;
    P1SEL |= BIT2;

    // ADC
    P6DIR &= ~(BIT0);
    P6SEL |= BIT0;

    // Auxiliaries
    int duty;
    int ADC;
    float ADC_value;
    float aux;

```

LAB 03 ADC (while)

```
37      while (1)
38      {
39          ADC_conversion();
40
41          // Valor del ADC a float
42          ADC_value = (float) ADC;
43
44          //Linealizacion
45          aux = ADC_value*255/671;
46
47          //Conversion del valor a numero entero
48          duty = (int) aux;
49
50          //Debuguear
51          if (duty > 255)
52          {
53              PORTD  &=  ~(1 << PORTD5);
54          }
55
56          //Valor para comparar con el timer
57          OCR0A = duty;
58
59      }
60  }
61
```

```
47
48      while(1)
49      {
50          //Get ADC value
51          ADC = start_conversion();
52
53          // Valor del ADC a float
54          ADC_value = (float) ADC;
55
56
57          // LED DEBUG
58          if (ADC_value > 1023)
59          {
60              P1OUT |= (BIT3);
61          }
62
63          //Linealizacion
64          aux = ADC_value*255/4095;
65
66          //Conversion del valor a numero entero
67          duty = (int) aux;
68
69          TA0CCR1 = duty;
70      }
71  }
72
```

LAB 03 ADC (funciones ADC)

```
66  // Configuracion del ADC de 10 bits
67  void Config_ADC(void)
68  {
69      //  Voltage de referencia interno
70      // y pin de lectura PC0 (MUX3:0 = 0)
71      ADMUX |= (1 << REFS0);
72      //Deshabilitar funcion I/0 PC0
73      DIDR0 |= (1 << ADC0D);
74      // Prescaler: 128
75      ADCSRA |= (1 << ADPS2)|(1 << ADPS1)|(1 << ADPS0);
76
77      //Modo de operacion: single conversion
78          //Enable
79          //ADCSRA |= (1 << ADEN);
80          //Start conversion
81          //ADCSRA |= (1 << ADSC);
82  }
83
84  void ADC_conversion(void)
85  {
86      //Habilitar conversion e iniciar
87      ADCSRA |= (1 << ADSC)|(1 << ADEN);
88
89      //Esperar a que la conversion este lista
90      while((ADCSRA & (1<< ADSC)));
91
92      //Deshabilitar
93      ADCSRA &= ~(1 << ADEN);
94  }
95
```

```
73  void config_adc(void)
74  {
75      // Turn on ADC
76      ADC12CTL0 = ADC12ON;
77
78      // ClK = SMLK  SHP = Use sampling timer+ Prescaler
79      ADC12CTL1 |= ADC12SSEL_3 + ADC12SHP + ADC12DIV_7;
80
81      // Reference Voltage (3.3) + PIN P6.0
82      ADC12MCTL0 |= ADC12SREF_0 + ADC12INCH_0;
83
84      // Enable conversion
85      ADC12CTL0 |= ADC12ENC;
86  }
87
88  int start_conversion(void)
89  {
90      // Star conversion
91      ADC12CTL0 |= ADC12SC;
92
93      // Wait conversion
94      while (!(ADC12IFG & BIT0));
95
96      // Return ADC value
97      return ADC12MEM0;
98  }
99
```

LAB 03 (Funciones Timer)

```
96  // Configuracion de TIMER para PWM
97  void Config_Timer(void)
98  {
99      //Modo de operacion: Fast PWM Mode
100     TCCR0A |= (1 << COM0A1)|(1 <<WGM01)|(1 <<WGM00);
101
102     //Comparar TCNT0 con OCR0A
103     TIMSK0 |= (1 << OCIE0A);
104
105     //Prescaler: 1024
106     TCCR0B |= (1 << CS02)|(1 << CS00);
107
108     //Comparador con TCNT0
109     OCR0A = 255;
110 }
111
```

```
100 void timer_A(void)
101 {
102     // TA0CTL control register:
103     // SMCLK = 1.048 MHz - Prescaler = 8 - UPMODE
104     TA0CTL |= TASSEL_2 + ID_3 + MC_1;
105
106     // Output mode PIN 1.2
107     TA0CCTL1 |= OUTMOD_7;
108
109     // Flags
110     TA0CCR0 = 255;          //Top
111     TA0CCR1 = 10;           //Flag to toggle
112 }
```

LAB 03 ADC (código incorrecto)

```
1  void ADC_conversion(void)
2  {
3      //Habilitar conversion e iniciar
4      ADCSRA |= (1 << ADSC)|(1 << ADEN);
5
6      //Esperar a que la conversion este lista
7      while((ADCSRA & (1<< ADSC))){
8          // Valor del ADC a float
9          ADC_value = (float) ADC;
10         //Linealizacion
11         aux = ADC_value*255/671;
12         //Conversion del valor a numero entero
13         duty = (int) aux;
14
15         OCR0A = duty;
16     }
17
18     //Deshabilitar
19     ADCSRA &= ~(1 << ADEN);
20 }
```

Ejemplos de programación