Lab 1: "Hola, mundo!" en microcontroladores

1. Objetivos

Este laboratorio introduce la programación en microcontroladores (MCUs o μC). Se aprenderá cómo escribir códigos en un computador y cargarlos y ejecutarlos en un microcontrolador. A su vez se sugieren las herramientas disponibles para la realización de esta actividad.

Esta experiencia estará separada en 3 partes. En primer lugar se utilizará el microcontrolador ATmega328P, posteriormente se ocupará el microcontrolador MSP430F5529 y para finalizar, se deberá realizar un programa de prueba de pines, que les ayudará a identificar posibles errores en sus tarjetas de desarrollo. De existir dichos problemas, deberán ser informadas a su ayudante corrector y notificadas en el siguiente cuestionario.

Figura 1: https://forms.gle/zTwhfcf15fPNNH6v8

2. Descripción de la actividad

Las actividades de este laboratorio constituyen principalmente un acercamiento tanto a las placas de desarrollo como a los medios mediante los cuales cargarán su código a las placas. En las actividades 1 y 2 deberán crear un programa que encienda y apague el LED integrado en la placa de desarrollo y cargarlo a las respectivas tarjetas de desarrollo. Por otro lado, la tercera actividad corresponde a un chequeo de los pines de cada placa, a modo de que ustedes puedan comprobar de modo oportuno que sus microcontroladores se encuentran operando adecuadamente. Deben subir todo el código utilizado a GitHub. Recuerden que cada laboratorio tiene su propio repositorio.

En caso de necesitarlo pueden recurrir a tutoriales en youtube donde se orienta al cómo

utilizar los distintos programas, algunos canales son: Microchip Makes [1] , pantechsolutions, [2] Code Composer, [3] Ragavesh Dhandapani, [4] , Human Hard Drive, [5] entre otros.

2.1. Task 1: ATmega328P

2.1.1. Descarga e instalación de plataforma de desarrollo

Windows: Para este caso se hará uso del programa Atmel Studio 7, el cual está disponible en la página de Microchip. Simplemente se debe descargar el web installer, ejecutar el archivo, aceptar los términos y condiciones y seleccionar la arquitectura de 8-bits (opcionalmente pueden incorporar las demás, pero para Atmega 328p esta es estrictamente necesaria).

Para comenzar, deben crear un proyecto nuevo y seleccionar la carpeta en donde lo quieren guardar, para su mayor comodidad, guardarlo en la carpeta que clonaron desde Github para este laboratorio, realizando una subcarpeta denominada AVR.
Para crear un archivo .c deben seleccionar la opción "*GCC Executable Project*" y seleccionar el microcontrolador Atmega328P. Acá deben copiar el código que está adjunto a este documento. Para subir el programa a la placa de desarrollo deben crear la solución seleccionando el modo "release" y luego presionar "Build Solution (F7)", posterior ir a **Tools** > **Device Programming**, seleccionar mEDBG como Tool, Atmega 328P como Device, ISP como Interface y presionar **Apply**, además deben leer el device signature.
Luego en la opción "Memories" seleccionar el archivo .hex creado en la carpeta **Release** que se creó en la carpeta de proyecto y presionar programar.

Nota: En algunos casos la subida del código podría fallar con ISP, esto ocurre cuando debugWIRE está habilitado y no fue finalizado antes de desconectar el microcontrolador. Para deshabilitarlo deben seguir el tutorial que se subirá a la carpeta de Github.

macOS: Para compilar en el sistema OSx se debe instalar el compilador avr-gcc, portado de Linux, mediante la instalación del gestor de paquetes HomeBrew en la terminal con el siguiente comando:

[1] Microchip Makes: https://www.youtube.com/watch?v=UMi6lg563BA&list=PLtQdQmNK_0DRhBWYZ32BEILOykXLpJ8tP
[2] Pantech Solutions: https://www.youtube.com/playlist?list=PLmdxycs9ZImfZwGqEfry9CWEcIkqmuYTK
[3] Code Composer: https://www.youtube.com/user/CodeComposerStudio/playlists
[4] Ragavesh Dhandapani: https://www.youtube.com/playlist?list=PLRqKd7sGGKMTB5egcnNuR_SJjMgzFntKj
[5] Human Hard Drive: https://www.youtube.com/playlist?list=PLA6BB228B08B03EDD

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/
    install)"
```

Una vez instalado HomeBrew procedemos a instalar complementos de Xcode para compilación integrados en el sistema operativo mediante el siguiente comando:

```
xcode-select --install
```

Finalmente procedemos a instalar el compilador de avr-gcc y su toolchain para cargar código a la tarjeta avr-dude:

```
brew tap osx-cross/avr
brew install avr-libc
brew install avrdude --with-usb
```

Si se obtiene un error en este paso entonces reemplazar los últimos dos comandos por:

```
brew install avr-gcc
brew install avrdude
```

En este punto ya puedes utilizar el editor de texto que prefieras para crear tu archivo .c. Una vez creado tu archivo procedes a establecerte en la carpeta de trabajo con el comando:

```
cd (directorio de trabajo)
```

Y procedes a la compilación con el siguiente comando donde se debe reemplazar la palabra "nombre" por el nombre de tu archivo .c a compilar.

```
avr-gcc -Wall -Wextra -Wpedantic --std=gnu99 -mmcu=atmega328p -Os -o nombre.elf nombre.c
avr-objcopy -j .text -j .data -O ihex nombre.elf nombre.hex
```

Para subir a la placa se utilizarán los siguientes comandos:

```
avrdude -vvvv -p atmega328p -F -c stk500v1 -P /dev/tty.usbmodem_puerto -b57600 -D -V -U
    flash:w:nombre.hex:i
```

El puerto deben conocerlo, esto puede averiguarse fácilmente conectando su placa AVR en algún puerto USB y mediante el comando `ls /dev` buscar el puerto correspondiente anotando los números de este y reemplazándolo en el código anterior.

Se sugiere realizar un bash con el código para evitar repetir el proceso de escribir estos comandos, pudiendo agregar la opción de recibir inputs si lo desea.

Linux: Se recomienda actualizar los paquetes que tengan previamente instalados, esto desde la terminal con:

```
    sudo apt-get update
    sudo apt-get upgrade all
```

Posteriormente descargar las librerías necesaria para el uso del ATmega328P:

```
    sudo apt-get install gcc-avr binutils-avr avr-libc
```

Pueden revisar la debida instalación colocando `avr-` en terminal y sin apretar ENTER sino que oprimiendo tab dos veces, debería aparecer una lista de funciones que pueden usar incluyendo avr-gcc. Ahora solo faltaría instalar avrdude esto haciendo:

```
    sudo apt-get install avrdude
```

Se recomienda la creación de una función para el momento de pasar sus programas a la tarjeta, esta se puede crear fácilmente buscando el archivo `/home/user/.bashrc` (ubuntu 18.04, podría tener otro nombre en un sistema operativo distinto), en donde al final del archivo pueden agregar:

```
function avrcompile () {
avr-gcc -Wall -Wextra -Wpedantic --std=gnu99 -mmcu=atmega328p -Os -o outputprogram.elf "$@"
avr-objcopy -j .text -j .data -O ihex outputprogram.elf outputprogram.hex
avrdude -vvvv -p atmega328p -F -c stk500v1 -P /dev/ttyACM0 -b57600 -D -V -U flash:w:outputprogram.hex:i
rm outputprogram.elf
rm outputprogram.hex
}
```

Podría ocurrir que el puerto USB tuviese otro nombre en su computador, por lo que habría que cambiar `/dev/ttyACM0` por el debido nombre en caso que tengan error con este.

2.1.2. "Hola, mundo!" en el microcontrolador

En la carpeta `ATmega328P` se encuentra el archivo `hello_world_atmega.c`. Este debe subirse a la placa. Luego, un LED de esta parpadeará.

2.1.3. Testeo de pines

No es raro que pines de las placas dejen de funcionar. Esto puede generar grandes pérdidas de tiempo, ya que uno piensa que su código está fallando, cuando en realidad el problema es la placa. Para evitar situaciones como ésta, se van a probar todos los pines.

Se debe subir el programa `pin_test_atmega.c` al ATmega328P. Este código encenderá todos los pines del microcontrolador. Se puede ver con un LED externo si un pin está funcionando. El circuito es extremadamente simple: se debe conectar el LED entre el pin y tierra **con un resistor de 560Ω conectado en serie**. La figura 2 muestra el *pinout* del ATmega328P Xplained Mini.

2.2. Task 2: MSP430F5529

2.2.1. Descarga e instalación de plataforma de desarrollo

Windows: En este caso se hará uso del programa Code Composer Studio 8 (o superior), el cual se encuentra disponible en la página de TexasInstruments[6]. Una vez descargado el archivo comprimido deben descomprimirlo y abrir el archivo ejecutable de setup. Al iniciar la instalación deberán aceptar los términos y condiciones y seleccionar la familia de procesadores, en este caso la opción corresponde a **"MSP430 ultra-low power MCUs"**. Cuando el programa termine de instalarse y se inicie deberán crear un nuevo proyecto y seleccionar la carpeta de destino, al igual que en el caso de Atmel Studio, se recomienda guardarlo en una subcarpeta MSP dentro de la carpeta clonada desde Github.

Cuando inicien un proyecto, deberán seleccionar el modelo del microcontrolador, el cual corresponde al **MSP430F5529**. Una vez terminen de escribir y guardar su código deberán presionar "Build" y luego "Flash". Una buena práctica para la detección de errores y depuración del código es probarlo con la opción "Debug", la cual permite monitorear el comportamiento de su programa línea a línea.

Nota: Por buena praxis se recomienda presionar el botón de reset luego de cargar el código a su placa. Esto debido a que en algunos casos el nuevo código empezará a ejecutarse en el microcontrolador después de presionar el botón.

macOS: Para proceder a la instalación del compilador y su respectivo toolchain requerimos instalar en primer lugar el gestor de paquetes Homebrew mediante el comando (si ya realizaste este paso en AVR omítelo):

```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/
    install)"
```

Luego aceptamos e instalamos el compelmento de Xcode mediante el siguiente comando (omitir si ya se ha hecho):

```
xcode-select --install
```

[6] http://www.ti.com/tool/CCSTUDIO

Este proceso requiere de varias librerías las cuales se instalarán mediante Homebrew de la siguiente forma:

```
brew tap sampsyo/mspgcc

brew install msp430-libc

brew install mspdebug

brew install git

git clone https://github.com/sampsyo/homebrew-mspgcc.git

cd homebrew-mspgcc

./addlinks.sh
```

Finalmente se debe descargar los drivers de las placas msp430 mediante el link https://github.com/loky32/msp430_OSX.git.

Movemos el archivo libmsp430.so a la ubicación `/usr/local/lib/`. Con esto concluye la instalación y ahora puedes compilar y subir a la placa msp430. Probamos el compilador mediante el siguiente comando:

```
msp430-gcc -Wall -Wextra --std=gnu99 -mmcu=msp430f5529 -Os -o nombre.elf nombre.c
msp430-objcopy --output-target=elf32-msp430 nombre.elf nombre.bin
```

Finalmente subimos el archivo compilado a la placa mediante el comando:

```
mspdebug tilib "prog nombre.bin"
```

Linux: Para la debida instalación de el compilador de terminal se requiere de la descarga directa del programa en el link: http://www.ti.com/tool/msp430-gcc-opensource, donde deben seleccionar MSP430-GCC-OPENSOURCE $->$ Get Software, y luego seleccionar el programa correspondiente a su sistema operativo, en este caso se usará

- `"msp430-gcc-full-linux-x64-installer-6.1.0.0.run"`

Una vez descargado en la terminal ejecutar:

```
chmod +x msp430-gcc-full-linux-x64-installer-6.1.0.0.run
sudo ./msp430-gcc-full-linux-x64-installer-6.1.0.0.run
```

Eso abrirá una ventana con el instalador. Seguir los pasos de instalación, el cual instalará todo en una ubicación similar a: `/home/user/ti/gcc`
Bastaría agregar al *bash profile* ubicado en normalmente en el archivo: `/home/user/.bashrc` (Ubuntu 18.04, podría tener otro nombre en un sistema operativo distinto) la siguiente linea:

- `export PATH="$PATH:/home/USER/ti/gcc/bin"`

Ya debería estar todo listo para poder pasar tus programas a la tarjeta de desarrollo, se recomienda la creación de una función agregando al archivo anterior las lineas:

```
function mspcompile () {
msp430-gcc -Wall -Wextra --std=gnu99 -mmcu=msp430f5529 -Os -o out_msp.elf "$@"
msp430-objcopy --output-target=elf32-msp430 out_msp.elf out_msp.bin
mspdebug tilib "prog out_msp.bin"
rm out_msp.bin
rm out_msp.elf
}
```

2.2.2. "Hola, Mundo!" en el microcontrolador

En la carpeta `MSP430F5529` se encuentra el archivo `hello_world_msp.c`. Este debe subirse a la placa. Luego, un LED de esta parpadeará.

2.2.3. Testeo de pines

No es raro que pines de las placas dejen de funcionar. Esto puede generar grandes pérdidas de tiempo, ya que uno piensa que su código está fallando, cuando en realidad el problema es la placa. Para evitar situaciones como esta, se van a probar todos los pines.

Se debe subir el programa `pin_test_msp.c` al MSP430F5529. Este código encenderá todos los pines del microcontrolador. Se puede ver con un LED externo si un pin está funcionando. El circuito es extremadamente simple: se debe conectar el LED entre el pin y tierra **con un resistor de 560Ω conectado en serie**. La figura 3 muestra el *pinout* del MSP430F5529 LaunchPad.

3. Subir código al repositorio de GitHub

Ya leída la guía sobre instalación y uso de la plataforma GitHub (se encuentra en el repositorio principal del curso), se procede a subir el código al repositorio asociado a su usuario registrado. A continuación se darán algunas sugerencias e instrucciones sobre el proceso de carga de códigos.

Para subir el código se debe tener la carpeta del repositorio clonada en el computador. En caso contrario la generaremos por primera vez mediante el comando:

t `git clone url_del_repositorio.git`

Cuando se realicen cambios importantes se sugiere fuertemente enviar un `push` a su repositorio con su respectivo `commit` lo más descriptivo posible sobre la modificación principal del código, de modo que no olvide cuál fue el cambio realizado.

También puede agregar una hoja de texto en formato Markdown (.md) para especificar el funcionamiento de su código u otras características que considere relevantes.

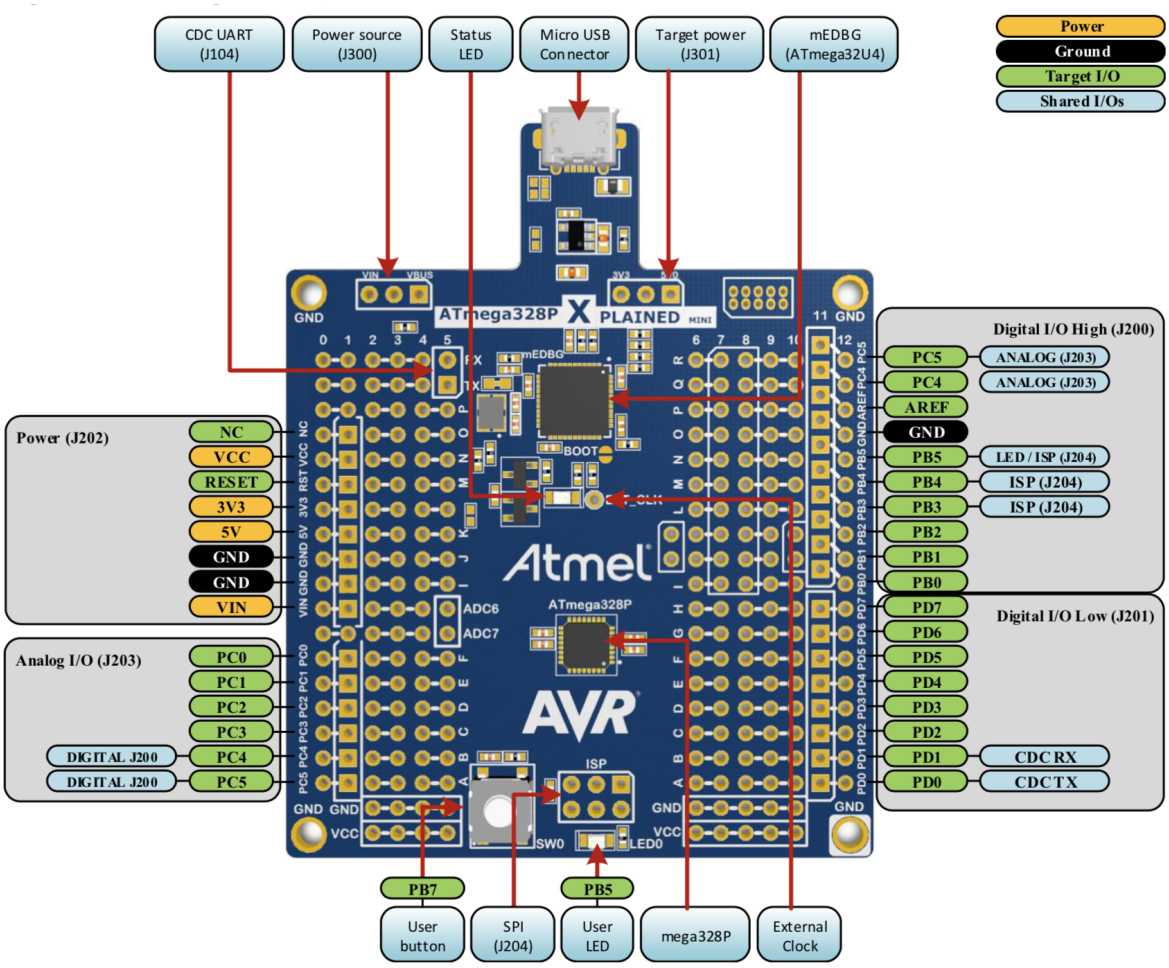

Figura 2: *Pinout* del ATmega328P Xplained Mini

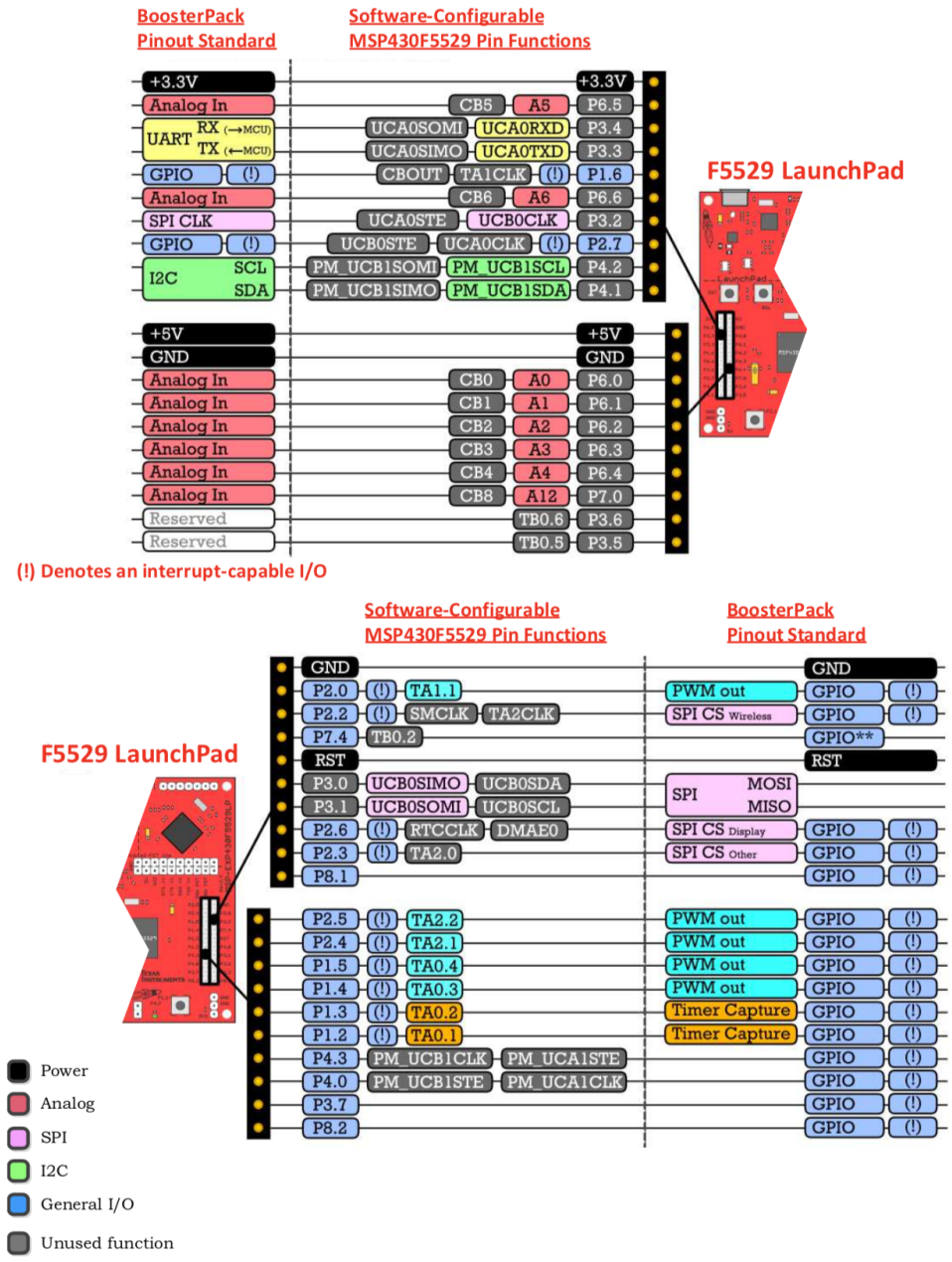

Figura 3: *Pinout* del MSP430F5529 LaunchPad