Carrera: Telecomunicaciones **Materia: Electronica Controlada** **Grupo: 3**

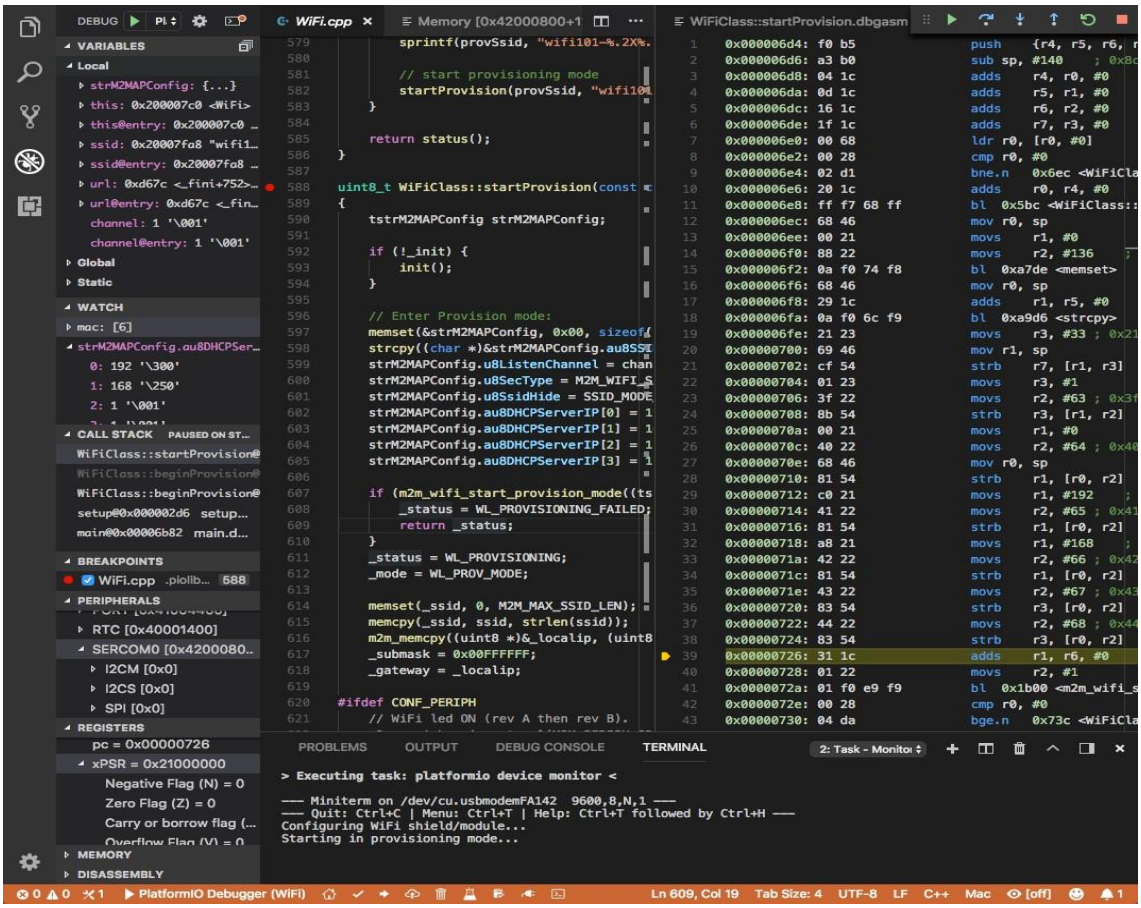
Docente: Jorge Morales – Gonzalo Vera

Alumnos: Carolina Nis – Fernando Vexenat – Rodolfo Paz – Juan Antoniazzi – Leonardo Gonzáles – Andrés Montaño

EJERCICIO 2 IDE 1)-

PlatformIO IDE para VSCode:

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C#, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity)



Contenido

Instalación:

Tenga en cuenta que no necesita instalar PlatfomaIO Core por separado si va a utilizar PlatformIO IDE para VSCode. PlatformIO Core (CLI) está integrado en PlatformIO IDE y podrá usarlo dentro de PlatformIO IDE Terminal.

Si va a utilizar Git para instalar plataformas de desarrollo ascendentes, clonar proyectos externos, instalar dependencias de biblioteca desde un repositorio, asegúrese de que el comando funcione desde una terminal del sistema. De lo contrario, instale un Cliente de git.

1.Descargue e instale el código oficial de Microsoft Visual Studio. PlatformIO IDE está construido sobre él.

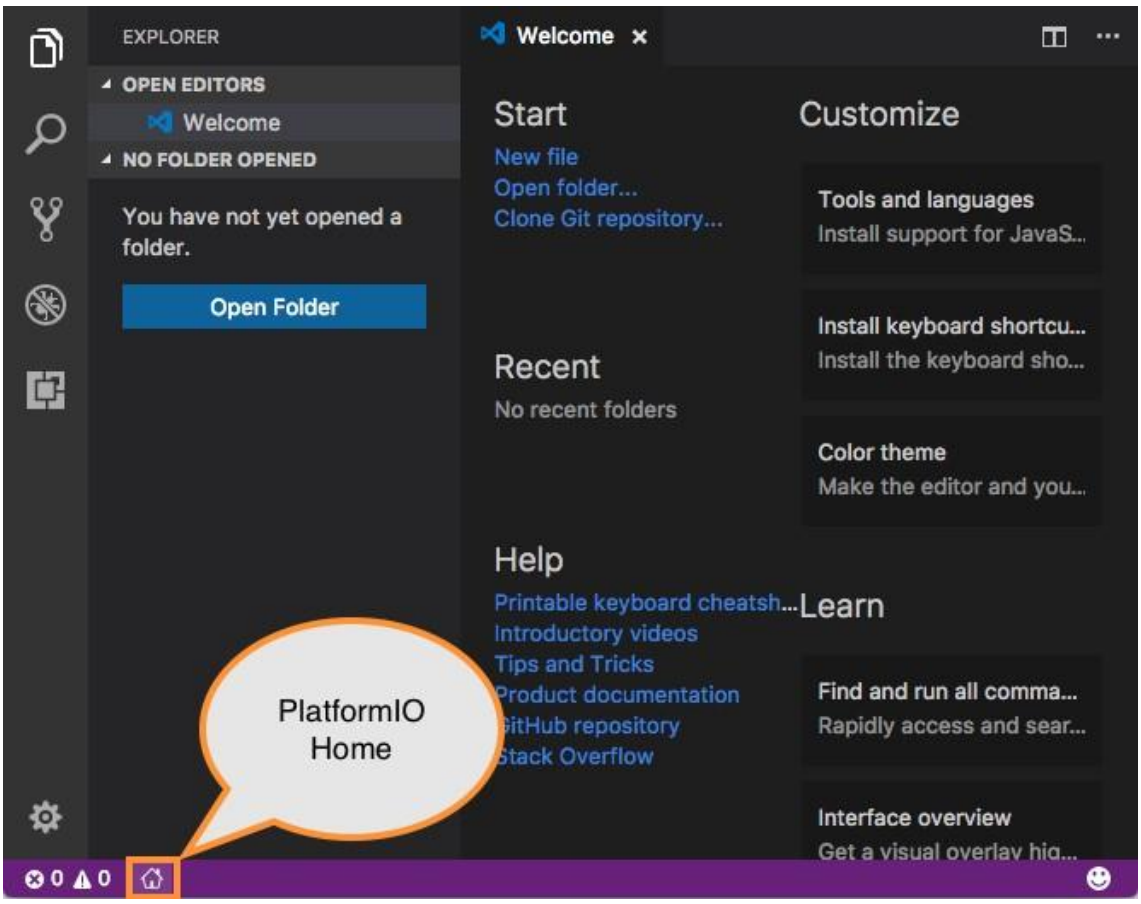
2. Abra el administrador de paquetes VSCode **3.**
Buscar la extensión oficial platformIO ide
4. Instale PlatformIO IDE.

Inicio rápido:

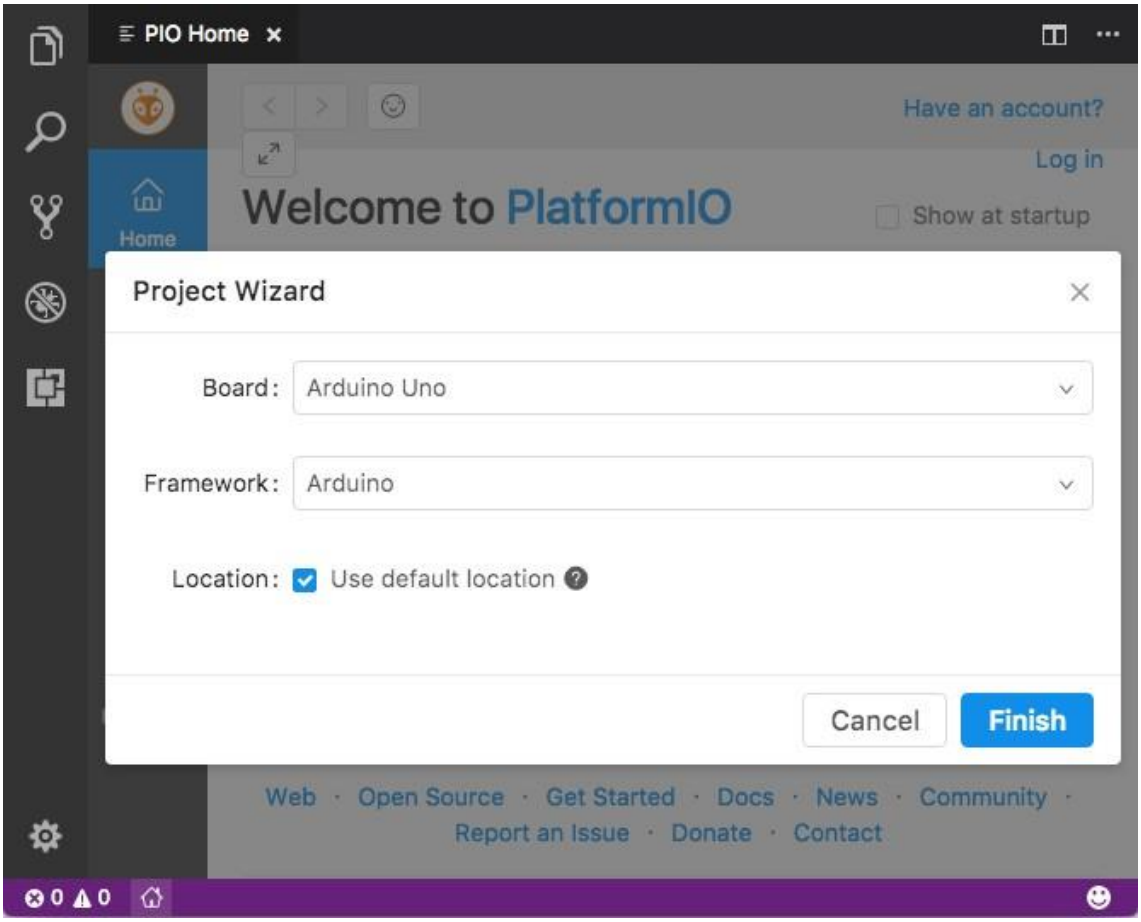
Este informe presenta los conceptos básicos del flujo de trabajo de PlatformIO IDE y le muestra un proceso de creación de un ejemplo simple de "Blink". Después de terminar, tendrá una comprensión general de cómo trabajar con proyectos en el IDE.

Configuración del proyecto.

1)- Haga clic en el botón "Inicio de PlatformIO" en la barra de herramientas de PlatformIO inferior



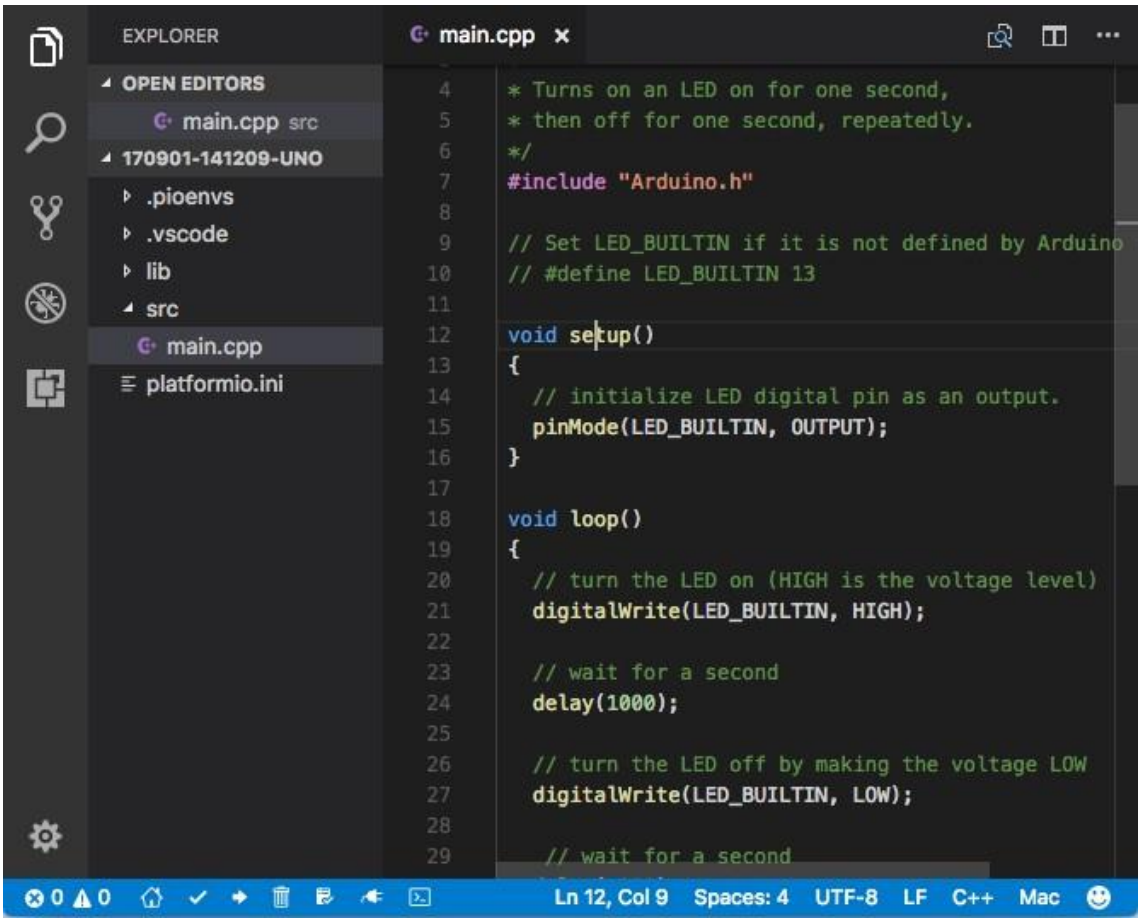
2)- Haga clic en "Nuevo proyecto", seleccione una placa y cree un nuevo proyecto PlatformIO



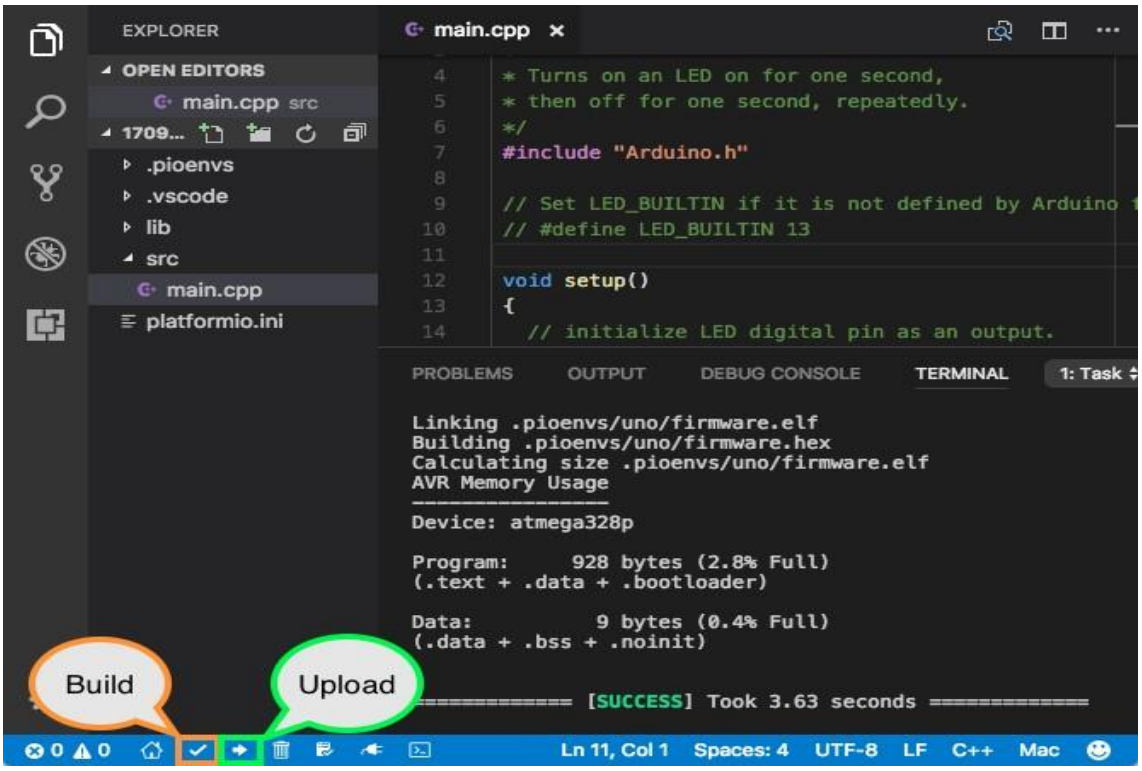
3)- Abra la carpeta main.cpp del formulario de src. archivo y reemplace su contenido con el siguiente:

Nota: El siguiente código funciona solo en combinación con placas basadas en Arduino. Siga el repositorio de ejemplos de proyectos de PlatformIO para ver otros proyectos preconfigurados.

```
/**
 * Blink
 *
 * Turns on an LED on for one second,
 * then off for one second, repeatedly.
 */
#include "Arduino.h"

// Set LED_BUILTIN if it is not defined by Arduino framework
// #define LED_BUILTIN 13

void setup()
{
  // initialize LED digital pin as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop()
{
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(LED_BUILTIN, HIGH);

  // wait for a second
  delay(1000);

  // turn the LED off by making the voltage LOW
  digitalWrite(LED_BUILTIN, LOW);

   // wait for a second
  delay(1000);
}
```



4)- Cree su proyecto con una ctrl+alt+btecla de acceso rápido (vea todas las combinaciones de teclas en la sección "Guía del usuario" a continuación) o usando el botón "Crear" en la barra de herramientas de PlatformIO



Barra de herramientas de PlatformIO

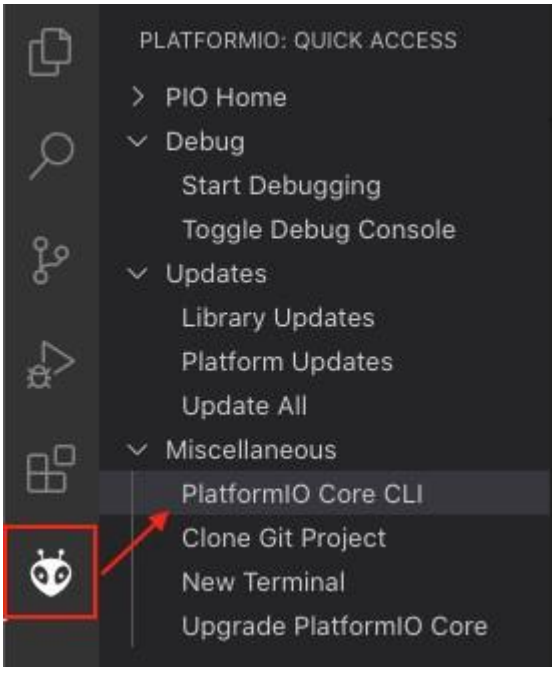
La barra de herramientas IDE de PlatformIO se encuentra en la barra de estado de VSCode (esquina izquierda) y contiene botones de acceso rápido para los comandos populares. Cada botón contiene una pista (retraso del mouse sobre él).

1. PlataformaIO Inicio
2. PlataformaIO: Construir
3. PlataformaIO: Subir
4. PlataformaIO: Limpio
5. Monitoreo de puerto serie
6. PlatformIO Core (CLI)
7. Conmutador de entorno de proyecto (si hay más de un entorno disponible). Consulte la Sección [env] de “platformio.ini” (Archivo de configuración del proyecto).

PlatformIO Core (CLI).

Hay 2 formas de acceder a PlatformIO Core (CLI) :

1. Icono de “Terminal” en la barra de herramientas de PlatformIO
2. Barra de actividad izquierda > PlatformIO (icono de hormiga) > Acceso rápido > Varios > PlatformIO Core CLI



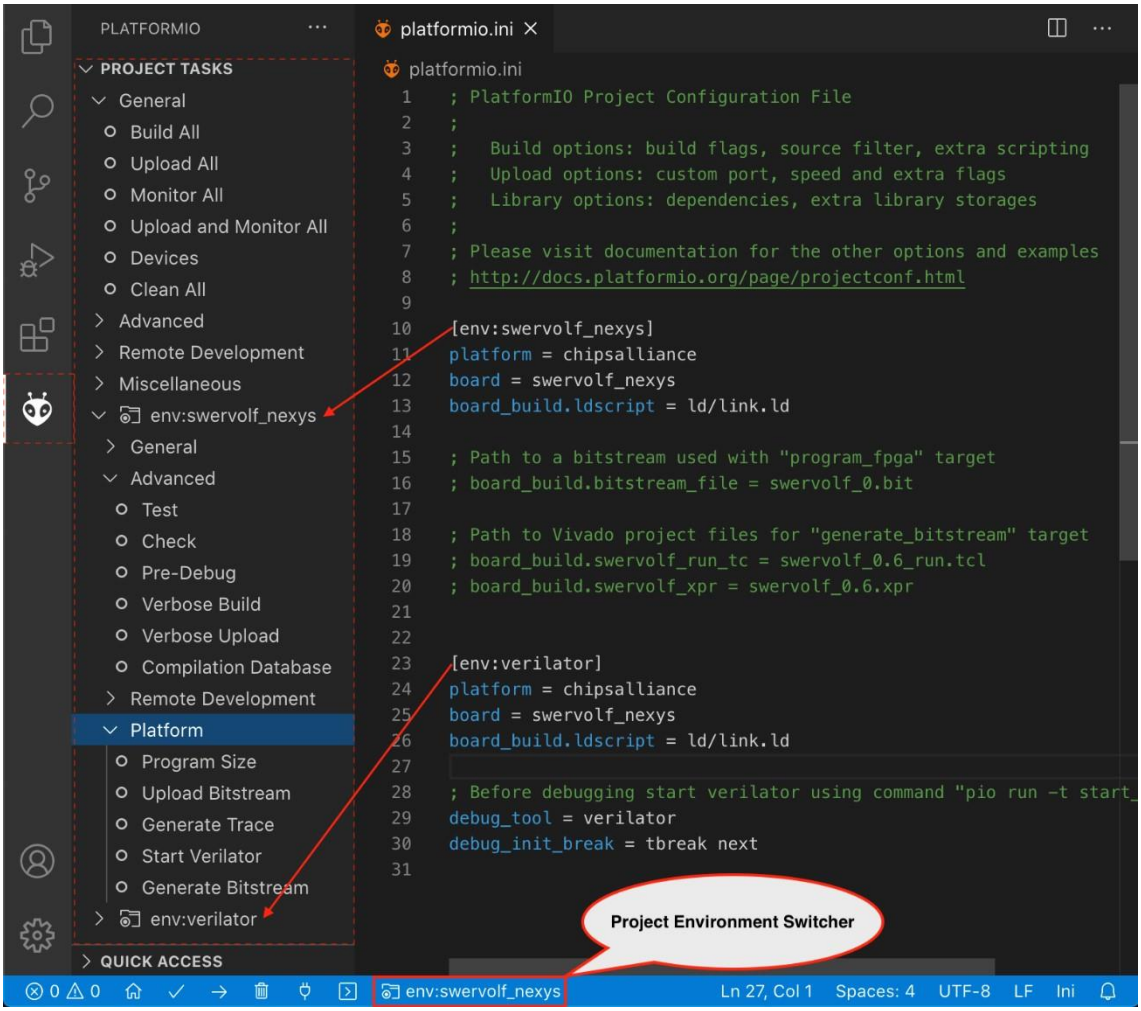
Tareas del proyecto:

Explorador de tareas.

PlatformIO proporciona acceso a la "Tarea del proyecto" donde puede controlar el proceso de construcción de los entornos declarados en "platformio.ini" (Archivo de configuración del proyecto) . El Explorador de tareas del proyecto se encuentra en la barra de actividad de VSCode, debajo del ícono de PlatformIO. También puede acceder a él a través de "VSCode Menu > Open View... > PlatformIO".

Insinuación

Tenga en cuenta que puede arrastrar/mover "Tarea del proyecto" a otra vista dentro de VSCode, como "Explorador".



Ejecutor de tareas:

PlatformIO IDE proporciona tareas integradas a través del menú (Crear, Cargar, Limpiar, Monitorear, etc.) y tareas personalizadas por entorno “platformio.ini” (Archivo de configuración del proyecto) ( ). El comportamiento predeterminado es usar paneles de terminales para la presentación, un panel dedicado a cada tarea única.Terminal > Run Task...[env:***]

El IDE de PlatformIO proporciona su propio Problems Matcher llamado $platformio. Puede usarlo más tarde si decide cambiar la configuración de la tarea base.

Puede anular las tareas existentes con sus propias opciones de presentación. Por ejemplo, configuremos PlatformIO Task Runner para usar un panel de terminal NUEVO para cada comando "Construir":

1. El elemento del menú abre una lista de tareas de VSCode para PlatformIO. En la línea, presione el icono de engranaje en el extremo derecho de la lista. Esto crea o abre el archivo con algún código de plantilla.Terminal > Run Task...PlatformIO: Build.vscode/tasks.json

2. Reemplace la plantilla tasks.jsoncon este código

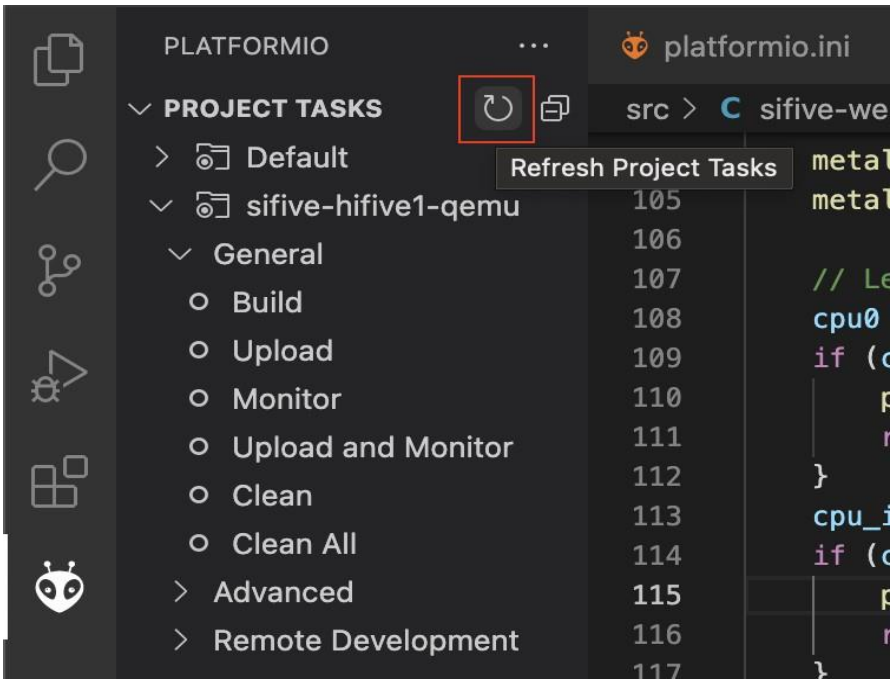
```
{
    "version": "2.0.0",
    "tasks": [
        {
            "type": "PlatformIO",
            "task": "Monitor",
            "problemMatcher": [
                "$platformio"
            ],
            "presentation": {
                "panel": "new"
            }
        }
    ]
}
```

3.

Ver más opciones en la documentación oficial de VSCode.

Tareas personalizadas:

Declare objetivos personalizados y actualice el Explorador de tareas:



<u>Espacios de trabajo multiproyecto:</u>

Puede trabajar con varias carpetas de proyectos en Visual Studio Code con espacios de trabajo multiraíz. Esto puede ser muy útil cuando está trabajando en varios proyectos relacionados al mismo tiempo. Obtenga más información en la documentación Espacios de trabajo multirraíz .

Monitoreo de puerto serie:

Puede personalizar Serial Port Monitor usando las opciones de Monitor en “platformio.ini” (Archivo de configuración del proyecto):

- monitor_puerto.
- monitor_velocidad.
- monitor_paridad.
- monitor_filtros.
- monitor_rts.
- monitor_dtr.
- monitor_eol.
- monitor_sin procesar.
- monitor_echo.

Ejemplo:

[env:esp32dev] platform

= espressif32 framework

= arduino board =

esp32dev

; Custom Serial Monitor port monitor_port

= /dev/ttyUSB1

; Custom Serial Monitor speed (baud rate) monitor_speed

= 115200

Depuración:

La depuración en VSCode funciona en combinación con la depuración. Debe tener una cuenta de PlatformIO para trabajar con ella. VSCode tiene una vista de actividad separada llamada "Depurar" (a la que se accede mediante el icono de error en la barra de herramientas de la izquierda). La depuración lo amplía con funciones e instrumentos de depuración más avanzados:

- Explorador de variables locales, globales y estáticas
- Puntos de ruptura condicionales
- Expresiones y puntos de observación
- Registros Genéricos

- Registros Periféricos
- Visor de memoria
- Desmontaje
- Soporte multihilo
- Un reinicio en caliente de una sesión de depuración activa.

Hay dos configuraciones de depuración preconfiguradas:

Depuración de PIO:

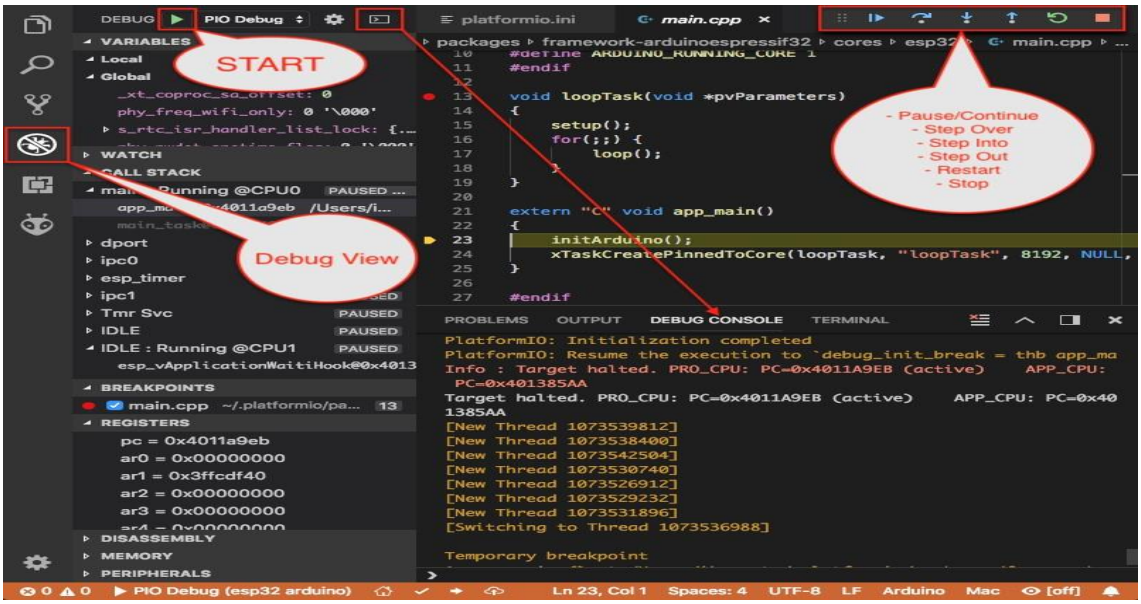
Configuración por defecto. PlatformIO ejecuta la tarea **previa a la depuración** y compila el proyecto mediante la configuración de depuración. Además, comprueba si hay cambios en el proyecto.

Depuración PIO (omitir Pre-Depuración):

PlatformIO se salta la etapa **previa a la depuración** y NO compila ni comprueba si hay cambios en el proyecto. Si realiza cambios en los archivos de origen del proyecto, no se reflejarán en las sesiones de depuración hasta que vuelva a cambiar a la configuración "PIO Debug" o ejecute manualmente la tarea "Pre-Debug".

Esta configuración es muy útil para una sesión de depuración rápida. Es súper rápido al omitir varios controles, lo que le permite controlar los cambios del proyecto manualmente.

Nota: Tenga en cuenta que la depuración utilizará el primer entorno de compilación declarado en "platformio.ini" (archivo de configuración del proyecto) si no se especifica la opción default_envs .



Formato variable:

Actualmente, VSCode no proporciona una interfaz de usuario ni una API para cambiar el formato de la variable. Consulte el problema de VSCode n.º 28025 relacionado.

Una solución temporal es establecer la base numérica predeterminada en la que el depurador muestra la salida numérica en la consola de depuración. (La consola de depuración está visible

durante las sesiones de depuración activas). Por ejemplo, para mostrar variables en formato hexadecimal, copie el siguiente código y péguelo en la "Consola de depuración": set output-radix **16.**

Los valores posibles, enumerados en base decimal, son: 8, 10, 16.

Los puntos de vigilancia:

Lea primero GDB: Establecimiento de puntos de observación.

Actualmente, VSCode no proporciona una API para cambiar el formato de valor de los puntos de observación. Puede emitir manualmente expresiones de punto de observación para mostrar el valor como tipos de puntero específicos:

- $pc, formato de entero decimal predeterminado
- *0x10012000, una dirección, formato de entero decimal predeterminado
- (void*)$pc, registro $pc, formato hexadecimal
- *(void**)0x10012000, una dirección, formato hexadecimal

Instalar comandos de shell.

Consulte Comandos de shell de instalación de PlatformIO Core.

Compatibilidad con servidor proxy:

Hay dos opciones para configurar un servidor proxy:

1. Declare las variables de entorno del sistema HTTP_PROXYy (por ejemplo, etc.) HTTPS_PROXYHTTP_PROXY=http://user:pass@10.10.1.10:3128/
2. Abra la configuración de VSCode y busque "Proxy". Configure "Http: Proxy" y **deshabilite "Http: Proxy Strict SSL".**

Atajos de teclado:

- ctrl+alt+b// cmd-shift-bConstruir ctrl-shift-bProyecto
- cmd-shift-d/ ctrl-shift-dProyecto de depuración
- ctrl+alt+uSubir firmware
- ctrl+alt+sMonitor de puerto serie abierto

Puede anular los enlaces de teclas existentes o agregar uno nuevo en VSCode. Consulte la documentación oficial de Key Bindings para Visual Studio Code .

Ajustes:

¿Cómo configurar los ajustes de VSCode? **platformIo-ide.activateOnlyOnPlatformIOProject:**

Si es verdadero, active la extensión solo cuando un proyecto basado en PlatformIO (que tiene un "platformio.ini" (archivo de configuración del proyecto)) está abierto en el espacio de trabajo. El valor predeterminado es. platformIO ide false **platformio-ide.activateProjectOnTextEditorChange:**

Activar automáticamente el proyecto dependiendo de un editor de texto abierto activo. El valor predeterminado es false. **platformio-ide.autoOpenPlatformIOIniFile:**

Abra automáticamente el archivo “platformio.ini” (Archivo de configuración del proyecto) de un proyecto cuando no haya otros editores abiertos. El valor predeterminado es true.

platformio-ide.autoCloseSerialMonitor:

Si es verdadero, cierre automáticamente el monitor del dispositivo pio antes de cargar/probar. El valor predeterminado es true. **platformio-ide.autoRebuildAutocompleteIndex:**

Si es verdadero, reconstruirá automáticamente el índice de proyectos de C/C++ cuando se cambie “platformio.ini” (archivo de configuración del proyecto) o cuando se instalen nuevas bibliotecas. El valor predeterminado es true. **platformio-ide.buildTask:**

La tarea de compilación (etiqueta) que se inicia con el botón "Generar" en la barra de herramientas y enlaces de teclas de PlatformIO. El valor predeterminado es. PlatformIO: Build

Puede crear tareas personalizadas personalizadas y asignar una de ellas a platformioide.buildTask.

platformio-ide.autoPreloadEnvTasks:

Precargue automáticamente TODAS las tareas del entorno del proyecto. El valor predeterminado es false. **platformIO-ide.customPATH:**

RUTA personalizada para el platformIOcomando. Pegue aquí el resultado del comando (Unix) / (Windows) escribiendo en la terminal de su sistema si prefiere usar una versión personalizada de PlatformIO Core (CLI). El valor predeterminado es, lo que significa que PlatformIO busca el comando en la ruta del sistema.echo $PATHecho %PATH%nullplatformio **platformio-ide.disableToolbar:**

Deshabilite la barra de herramientas de PlatformIO. El valor predeterminado es false.

platformio-ide.forceUploadAndMonitor:

Si es verdadero, el platformio-ide.uploadcomando Cargar ( ) se cambia para usar la tarea "Cargar y monitorear". El valor predeterminado es false. **platformIO-ide.reopenSerialMonitorDelay:**

Configure el tiempo en milisegundos antes de volver a abrir Serial Port Monitor. El valor predeterminado es 0, lo que significa volver a abrir al instante. **platformIO-ide.useBuiltinPython:**

Utilice un intérprete de Python 3 portátil si está disponible. El valor predeterminado es true.

platformio-ide.useBuiltinPIOCore:

Si es verdadero, utilice PlatformIO Core (CLI) integrado. El valor predeterminado es true.

platformio-ide.useDevelopmentPIOCore:

Si es verdadero, use la versión de desarrollo de PlatformIO Core (CLI) . El valor predeterminado es false.

platformio-ide.disablePIOHomeStartup:

Deshabilite mostrar PlatformIO Home al inicio. El valor predeterminado es false. **platformio-ide.pioHomeServerHttpHost:**

Host HTTP del servidor de PlatformIO Home. El valor predeterminado es 127.0.0.1, pero en el caso de entornos dockerizados 0.0.0.0. **platformio-ide.pioHomeServerHttpPort:**

Puerto HTTP del servidor principal de PlatformIO. El valor predeterminado 0asigna automáticamente un puerto libre en el rango [8010..8100]). **platformio-ide.customPyPiIndexUrl:**

URL base personalizada del índice de paquetes de Python (predeterminado https://pypi.org/simple).

Problemas conocidos:

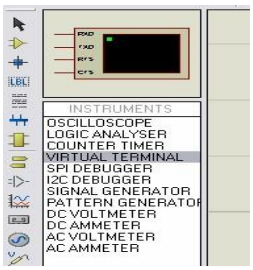
PackageManager no puede instalar la herramienta:

Este es un error conocido en el problema #61 de VSCode Terminal.

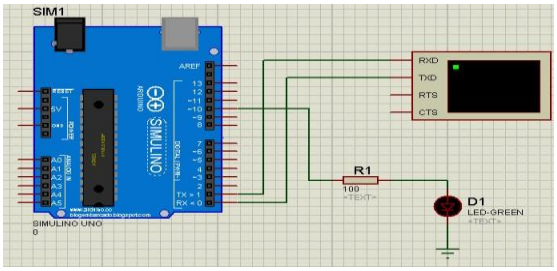
Una solución temporal es instalar paquetes usando una terminal del sistema (no la Terminal VSCode). Utilice la "Solución 3: Ejecutar desde la terminal" en Preguntas frecuentes > Administrador de paquetes > [Error 5] Acceso denegado. Luego, vuelva a usar la Terminal VSCode.

SIMULAR MONITOR SERIAL EN PROTEUS EN ARDUINO

Para simular el monitor serial en proteus debemos hacer uso del instrumento virtual terminal.



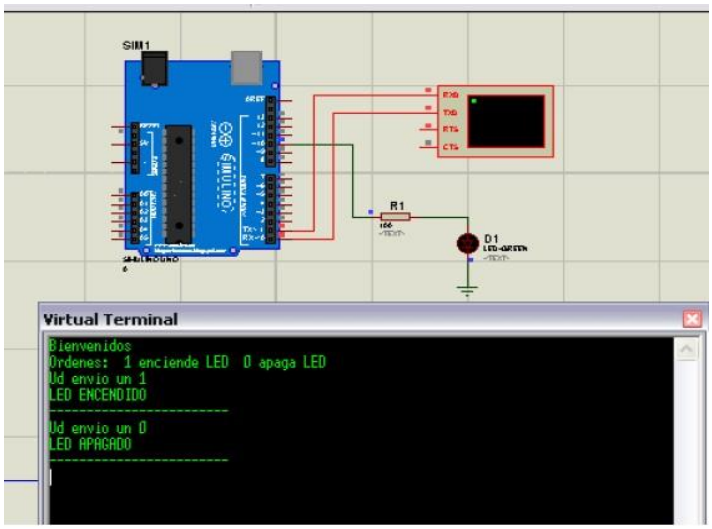
Lo incluimos en nuestro circuito Arduino muestra figura.



Notar que los TX y RX van a cruzados entre el Virtual Terminal y la placa Arduino.

```
//Programa LED_CONTROL

void setup()
{
  Serial.begin(9600);// inicializamos la comunicación serial
  pinMode(10,OUTPUT); //definimos el PIN 10 como salida

Serial.println("Bienvenidos "); //Mensaje a Monitor Serial
Serial.println("Ordenes:  1 enciende LED  0 apaga LED");//Mensaje por Monitor Serial
delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño
}

void loop()
{
  if(Serial.available())    //Si la comunicacion serial es utilizable, pregunta aqui.
  {
    char c=Serial.read(); // Se lee el monitor serial y esperando solo un caracter
                  // se almacena en una variable tipo char que llamamos c

  if(c=='1')    //pregunta por el contenido de la variable c

  {
   digitalWrite(10,HIGH); // coloca en ALTO la salida digital PIN 10
   Serial.println("Ud envio un 1"); //Mensaje a Monitor Serial
   Serial.println("LED ENCENDIDO"); //Mensaje a Monitor Serial
    Serial.println("-----------------------");
    delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño
  }
  if(c=='0')  //pregunta por el contenido de la variable c

  {
   digitalWrite(10,LOW);
   Serial.println("Ud envio un 0");  //Mensaje a Monitor Serial
   Serial.println("LED APAGADO");  //Mensaje a Monitor Serial
   Serial.println("-----------------------");  //Mensaje a Monitor Serial
   delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño

  }


 }
}
```

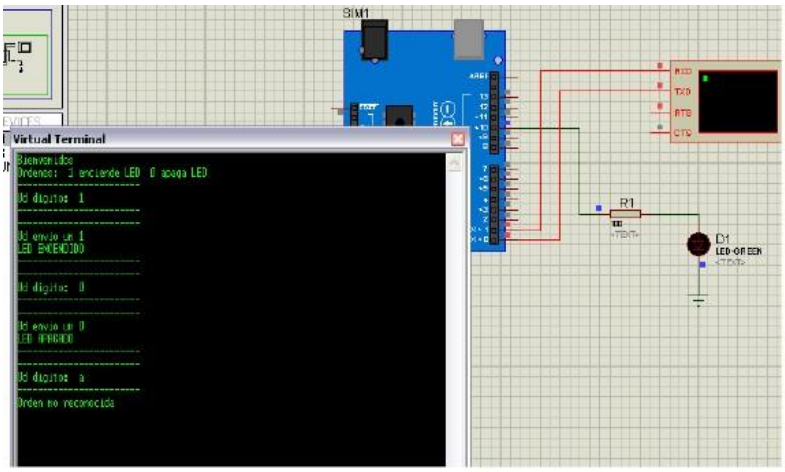
Podemos tipear las ordenes si nos posicionamos dentro de la ventana del Virtual Terminal.



Otro Ejemplo:

```
//PROGRAMA LED_CONTROL1

void setup()
{
  Serial.begin(9600);// inicializamos la comunicación serial
  pinMode(10,OUTPUT); //definimos el PIN 10 como salida

Serial.println("Bienvenidos "); //Mensaje a Monitor Serial
Serial.println("Ordenes:  1 enciende LED  0 apaga LED");//Mensaje por Monitor Serial
delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño
}

void loop()
{
  if(Serial.available())    //Si la comunicacion serial es utilizable, pregunta aqui.
  {
    char c=Serial.read(); // Se lee el monitor serial y esperando solo un caracter
                          // se almacena en una variable tipo char que llamamos c


     Serial.println("------------------------"); //Mensaje a Monitor Serial
     Serial.print("Ud digito:  "); //Mensaje a Monitor Serial

     Serial.println(c); //Mensaje a Monitor Serial

    Serial.println("------------------------"); //Mensaje a Monitor Serial


    if(c=='1')     //pregunta por el contenido de la variable c

    {
     digitalWrite(10,HIGH); // coloca en ALTO la salida digital PIN 10
      Serial.println("------------------------");  //Mensaje a Monitor Serial
     Serial.println("Ud envio un 1"); //Mensaje a Monitor Serial
     Serial.println("LED ENCENDIDO"); //Mensaje a Monitor Serial
      Serial.println("------------------------");
      delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño
    }
    if(c=='0')   //pregunta por el contenido de la variable c

    {
     digitalWrite(10,LOW);
      Serial.println("------------------------");  //Mensaje a Monitor Serial
      Serial.println("Ud envio un 0");  //Mensaje a Monitor Serial
      Serial.println("LED APAGADO");  //Mensaje a Monitor Serial
      Serial.println("------------------------");  //Mensaje a Monitor Serial
      delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño

    }

      if((c!='0')&& (c!='1')) Serial.println("Orden no reconocida");  //Mensaje a Monitor Serial


  }
}
```

Si observamos en proteus sucede lo siguiente:



En este caso enviaremos números de 3 dígitos para las órdenes.

```
//PROGRAMA LED_CONTROL2

//Lee algo en el puerto serial y lo almacena en num

int num; // Definida como variable global


void setup()
{
  Serial.begin(9600);// inicializamos la comunicación serial
  pinMode(10,OUTPUT); //definimos el PIN 10 como salida

Serial.println("Bienvenidos "); //Mensaje a Monitor Serial
Serial.println("Ordenes:  345 enciende LED  678 apaga LED");//Mensaje por Monitor Serial
delay(100);  //Los retardos son necesarios en la practica para mejorar desempeño

}

void loop()
{

 /*
   * Evaluamos el momento en el cual recibimos un caracter
   * a través del puerto serie
   */


 if(Serial.available())    //Si la comunicacion serial es utilizable, pregunta aqui.
 {


 //Delay para favorecer la lectura de caracteres

    delay(300); //Este tiempo es grande para PODER HACER SIMULACION PROTEUS
                //originalmente decia 22 para Arduino real

    //Se crea una variable que servirá como buffer
    String bufferString = "";


    /*
     * Se le indica a Arduino que mientras haya datos
     * disponibles para ser leídos en el puerto serie
     * se mantenga concatenando los caracteres en la
     * variable bufferString
     */

    while (Serial.available()>0) {
      bufferString += (char)Serial.read();
    }

    num = bufferString.toInt(); //Se transforma el buffer a un número entero
                                //Se carga lo leido en la variable num
                                 //Luego podemos preguntar sobre el valor
                                 // de dicha variable – Por ejemplo
                                // en Tachos LED su valor selecciona color


   Serial.println("-------------------------"); //Mensaje a Monitor Serial
    Serial.print("Ud digito:  "); //Mensaje a Monitor Serial

    Serial.println(num); //Mensaje a Monitor Serial

   Serial.println("-------------------------"); //Mensaje a Monitor Serial


   if(num==345)     //pregunta por el contenido de la variable num

   {
    digitalWrite(10,HIGH); // coloca en ALTO la salida digital PIN 10
     Serial.println("-------------------------");  //Mensaje a Monitor Serial
    Serial.println("Ud envio un "); //Mensaje a Monitor Serial

    Serial.println(num);


    Serial.println("LED ENCENDIDO"); //Mensaje a Monitor Serial
     Serial.println("-------------------------");
     delay(2000);  //Los retardos son necesarios en la practica para mejorar desempeño
   }
   if(num==678)   //pregunta por el contenido de la variable num

   {
    digitalWrite(10,LOW);
     Serial.println("-------------------------");  //Mensaje a Monitor Serial
    Serial.println("Ud envio un ");  //Mensaje a Monitor Serial

      Serial.println(num);
    Serial.println("LED APAGADO");  //Mensaje a Monitor Serial
    Serial.println("-------------------------");  //Mensaje a Monitor Serial
    delay(200);  //Los retardos son necesarios en la practica para mejorar desempeño

   }

    if((num!=345)&& (num!=678)) Serial.println("Orden no reconocida");  //Mensaje a Monitor Serial

}


 }
```

Si observamos en proteus