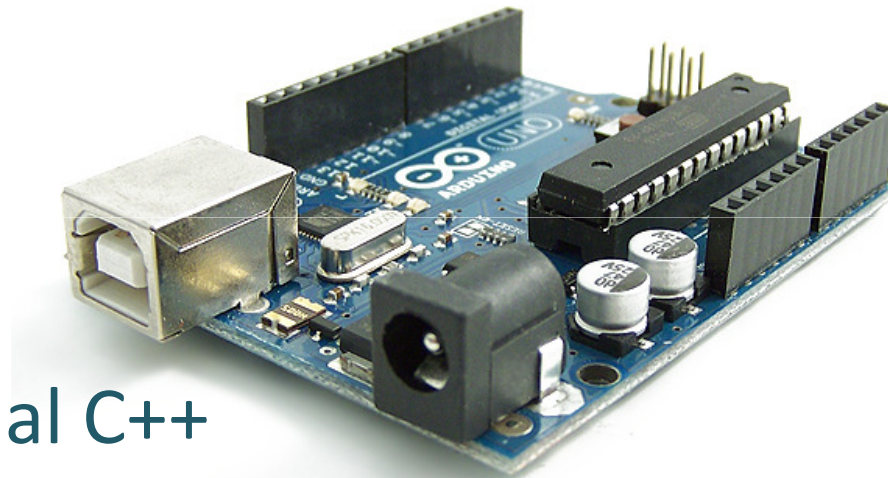




Aplicaciones Prácticas con Sistemas Arduino



Introducción a visual C++

CEP SEVILLA
IES Los Viveros
Curso 2011/2012
Ref: 11412FP23

Luis Modesto González
José Pujol Pérez
Coordinador:
Leopoldo Acal Rodríguez



Microsoft Visual Studio .NET



- Las aplicaciones se ejecutan sobre un framework(.net).
- Se compone de
 - Visual basic .net
 - Visual C++
 - Visual C#
 - Otras extensiones y complementos como SQL Server, Silverlight.....





Visual C++



- A favor:
 - Es similar al c de arduino
 - Está bien documentado y existe ayuda
 - Está bastante difundido
 - El IDE está completamente en español
 - La versión express es gratuita
- En contra:
 - Sólo para windows
 - No existen alternativas similares en otros sistemas operativos





Obtener visual c++ Express



- La versión Express es completamente funcional y gratuita.
- La versión actual es la 2010, sin embargo se recomienda la 2008 por disponer de “intelli sense”. En 2010 intellisense es una opción de terceros (de pago).
- Se instala desde:
<http://www.microsoft.com/visualstudio/en-us/products/2008-editions/express>(seleccionando idioma español)
- Si queremos la iso de visual studio , la podemos descargar desde la misma página
- Número de serie, se obtiene al registrarse: HGBMHWLDH243HF





PROGRAMACION ORIENTADA A OBJETOS



- Un objeto es una estructura de programación que contiene:
 - **Métodos:** funciones
 - **Atributos (propiedades):** variables
 - **Eventos:** Acciones que ocurren a veces y se pueden asociar a una función ó método





PROGRAMACION ORIENTADA A OBJETOS



- **Constructor (método con el mismo nombre que la clase):** Se ejecuta de forma automática al crear el objeto.
- **Destructor (método con el mismo nombre que la clase+~):** Se ejecuta de forma automática al destruir el objeto.
- **This:** Es una referencia al propio objeto
- **NameSpaces:** Indica el “espacio de nombres” donde el compilador buscará una función. Permite agregar librerías, similar a include.
- **Operador ^:** Nos permite declarar punteros. En Visual C++, los objetos se nombran con punteros.
- **Operador ->:** Nos permite seleccionar un atributo o método de un objeto
- **Gcnew:** Permite crear un objeto. Es lo mismo que new, pero facilita que visual C, destruya el objeto de forma automática si ya no se usa.

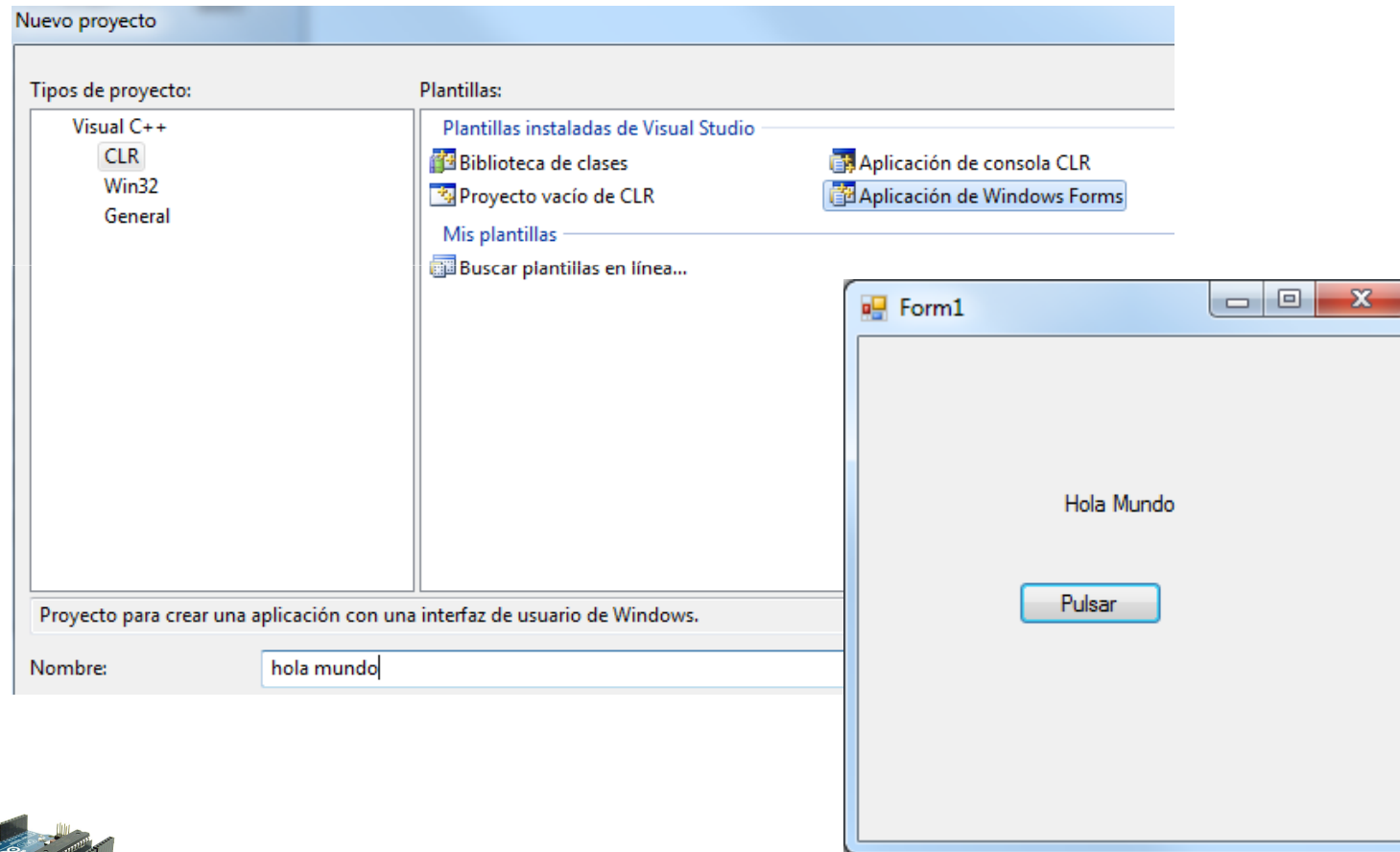




Visual C++ “hola mundo”



- Ejemplo “hola Mundo”





Arduino y visual C



- Nuevo proyecto Windows forms
- Incluimos la librería <windows.h>
- Incluimos la sentencia (sólo si nuestro programa usará threads –hilos)
CheckForIllegalCrossThreadCalls=0;

```
1 #include <windows.h>
2 #pragma once
3
4
```

```
25
26 public:
27     Form1(void)
28     {
29         CheckForIllegalCrossThreadCalls=0;
30         InitializeComponent();
31     }
32
```

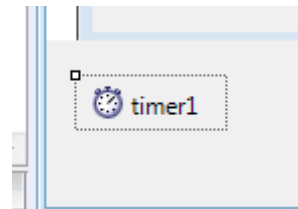




Componente Timer



- Es un componente “no visual”.
- Le indicamos la duración de la espera, tras la cual lanzará un evento Tick.
- En el evento, indicaremos las acciones a realizar.





Componente Serial Port



- Componente no visual, que simplifica el acceso al puerto serie
- En sus propiedades, ajustamos, velocidad, nombre del puerto, etc
- Dispone de métodos:
 - Open()
 - Close()
 - Read(),readByte(),readLine()...
 - Write(),writeLine()



serialPort1 System.IO.Ports.SerialPort	
Datos	
(ApplicationSettings)	
Diseño	
(Name)	serialPort1
GenerateMember	True
Modifiers	Private
Varios	
BaudRate	9600
DataBits	8
DiscardNull	False
DtrEnable	False
Handshake	None
Parity	None
ParityReplace	63
PortName	COM1
ReadBufferSize	4096
ReadTimeout	-1
ReceivedBytesThresh	1
RtsEnable	False
StopBits	One
WriteBufferSize	2048
WriteTimeout	-1



Componente Serial Port



- Dispone del evento: DataReceived, que se dispara al recibir datos.
- Cada vez que se ejecuta el evento DataReceived, lo hace en otro hilo de ejecución.
- Los datos recibidos, los obtenemos con el método read(),readByte() o readLine(), para leer caracteres, bytes o hasta el fin de línea.

```
private: System::Void serialPort1_DataReceived(System::Object^  
sender, System::IO::Ports::SerialDataReceivedEventArgs^ e) {  
    byte valor;  
    valor=this->serialPort1->ReadByte();  
    switch (valor){  
        .....  
    }
```





Componente Serial Port



- Para escribir un dato, usamos el método write().
- Write, necesita, un buffer donde estarán los datos a enviar, una posición inicial dentro del buffer, y un número de bytes a enviar.

```
array<byte>^ buffer = gcnew array<byte>(10);  
buffer[0]=52;//envia un código 51  
buffer[1]=this->trackBar1->Value;  
serialPort1->Write(buffer,0,2);
```

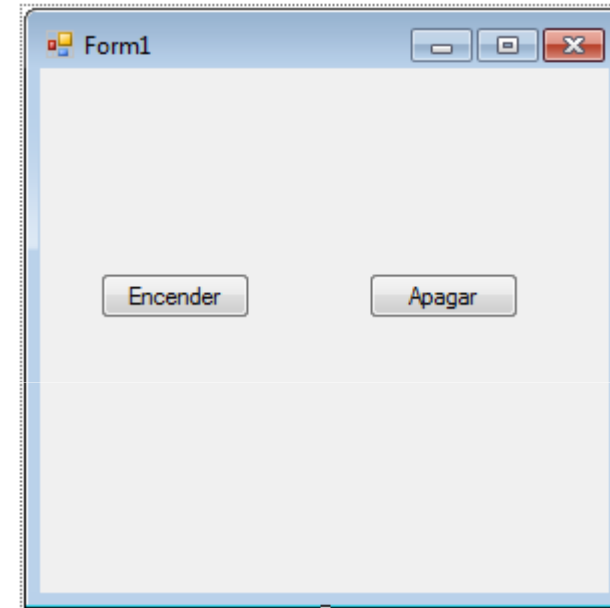




Ejemplo 1: Enciende led - Visual C



- Creamos el interface
- Codificamos las funcionalidades:
 - 50: encender led
 - 51: apagar led
- En el constructor abrimos el puerto serie
- En el destructor lo cerramos



```
array<byte>^ buffer = gcnew array<byte>(10);  
buffer[0]=50;//envia un código 50  
serialPort1->Write(buffer,0,1);
```





Ejemplo 1: Enciende led - Arduino



- Leemos un byte del puerto serie.
- Lo comparamos con los comandos configurados
- Ejecutamos las acciones asociadas

```
#define led 13
int valor=0;

void setup() {
  pinMode(led, OUTPUT);
  Serial.begin(9600);
}

void loop () {

  if(Serial.available()>0)
  {
    valor=Serial.read();
    switch(valor)
    {
      case 50: digitalWrite(led,HIGH);break;
      case 51: digitalWrite(led,LOW); break;
    }
  } delay(10);
}
```

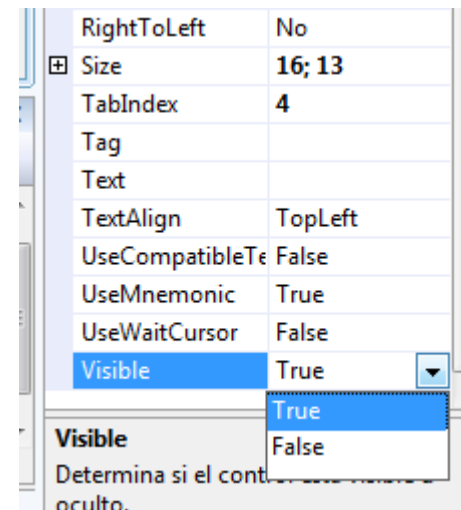
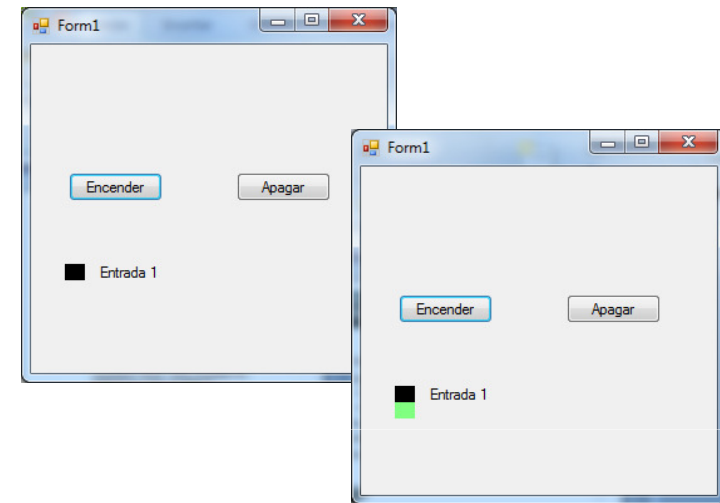




Ejemplo 2:Lectura digital - Visual C



- Añadimos un nuevo comando al ejemplo anterior, que nos permitirá obtener el estado de una entrada digital.
- Lo representamos con un cuadro negro, o verde en función que haya entrada o no.
- El cuadro lo hacemos con un label con 3 espacios, y fondo negro(BackColor)
- Para poner el fondo en verde debemos escribir:
 - `this->label2->BackColor=System::Drawing::Color::GreenYellow;break;`
- O ponemos otro label con el color personalizado y le copiamos el atributo.
 - `this->label2->BackColor=this->label3->Backcolor;`
- Para que el otro label no se vea, el ponemos el atributo visible a false





Ejemplo 2:Lectura digital - Visual C



- Para solicitar la lectura, usaremos un timer, con disparo a 5 veces por segundo
- Con el evento Tick, enviaremos un código 52 a arduino.
- Arduino debe devolver la lectura de la entrada digital correspondiente
- El evento DataReceived del serialPort, nos indicará que hay datos recibidos
- Leemos un Byte, y según su valor, efectuamos las acciones correspondientes



```
private: System::Void  
timer1_Tick(System::Object^ sender,  
System::EventArgs^ e) {
```

```
    array<byte>^ buffer = gcnew array<byte>(10);  
    buffer[0]=52;           //envia un código 52  
    serialPort1->Write(buffer,0,1);  
}
```

```
private: System::Void  
serialPort1_DataReceived(System::Object^ sender,  
System::IO::Ports::SerialDataReceivedEventArgs^ e)  
{  
    byte dato,dato1;  
    dato=this->serialPort1->ReadByte();  
    switch (dato){  
        case 52:dato1=this->serialPort1->  
>ReadByte();  
        if (dato1==0)  
            {this->label2->BackColor=  
System::Drawing::Color::Black;break;}  
        else  
            this->label2->BackColor=  
System::Drawing::Color::GreenYellow;break;  
        break;  
    }  
}
```



Ejemplo 2:Lectura digital - Arduino



- Implementamos el comando dentro de arduino.
- Primero devolvemos el mismo código para que el programa visual C, pueda identificarlo.
- Luego enviamos el estado del puerto



```
#define led 13
int valor=0,valor1=0;

void setup(){
  pinMode(led,OUTPUT);
  pinMode(5,INPUT);
  Serial.begin(9600);
}

void loop (){

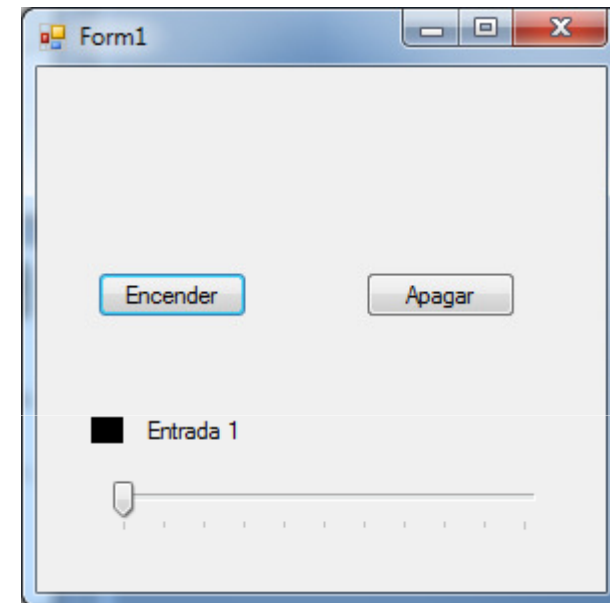
  if(Serial.available(>0)
  {
    valor=Serial.read();
    switch(valor)
    {
      case 50: digitalWrite(led,HIGH);
               break;
      case 51: digitalWrite(led,LOW);
               break;
      case 52: valor1=digitalRead(5);
               Serial.write(52);
               Serial.write(valor1);
               break;
    }
  } delay(10);
}
```



Ejemplo 3:Enviar dato – Visual C



- En este ejemplo, regularemos la intensidad de un led con un componente trackBar
- Crearemos un código 54 que enviará también el nivel de intensidad deseado
- Ajustamos los límites del trackBar a los de arduino (0-255)



Maximum	255
MaximumSize	0; 0
Minimum	0





Ejemplo 3:Enviar dato – Visual C



- El evento Scroll del trackBar, nos indicará cuando debemos enviar el comando a arduino.

```
private: System::Void trackBar1_Scroll(System::Object^
sender, System::EventArgs^ e) {
    array<byte>^ buffer = gcnew array<byte>(10);
    buffer[0]=53;//envia un código 53
    buffer[1]=trackBar1->Value;
    serialPort1->Write(buffer,0,2);
}
```





Ejemplo 3:Enviar dato – Arduino



- Implementamos el comando dentro de arduino.
- Leemos un 2º dato del puerto serie.
- Luego escribimos dicho dato en la salida analógica

```
#define led 13
int valor=0,valor1=0;
void setup(){
  pinMode(led,OUTPUT);
  pinMode(5,INPUT);
  pinMode(11,OUTPUT);
  digitalWrite(5,HIGH);
  Serial.begin(9600);
}
void loop (){
  if(Serial.available()>0)
  {
    valor=Serial.read();
    switch(valor)
    {
      case 50: digitalWrite(led,HIGH);break;
      case 51: digitalWrite(led,LOW); break;
      case 52: valor1=digitalRead(5);
                Serial.write(52);
                Serial.write(valor1);break;
      case 53: valor1=Serial.read();
                analogWrite(11,valor1);break;
    }
  }
  delay(10);
}
```

