**ECE 4180 Lab 5**

**Due Date: Odd Monday, March 30th and Even Tuesday, March 31st**

**C/C++ and C# I/O application development**

**in Windows and fidgeting with Phidgets**

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Item** | **TA Checkoff** | |
| Basic C/C++ Helloworld |  | |
| Basic Phidgets LCD demo |  | |
| Phidgets LCD Time & Sensor Display |  | +2.5 EC |
| Time & Sensor on ATOM PC |  | |
| Mbed I/O using USB Serial port |  | |
| Basic C# Helloworld |  | |

Grade: \_\_\_\_\_\_\_\_\_\_\_\_\_ TA Signature/Date\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

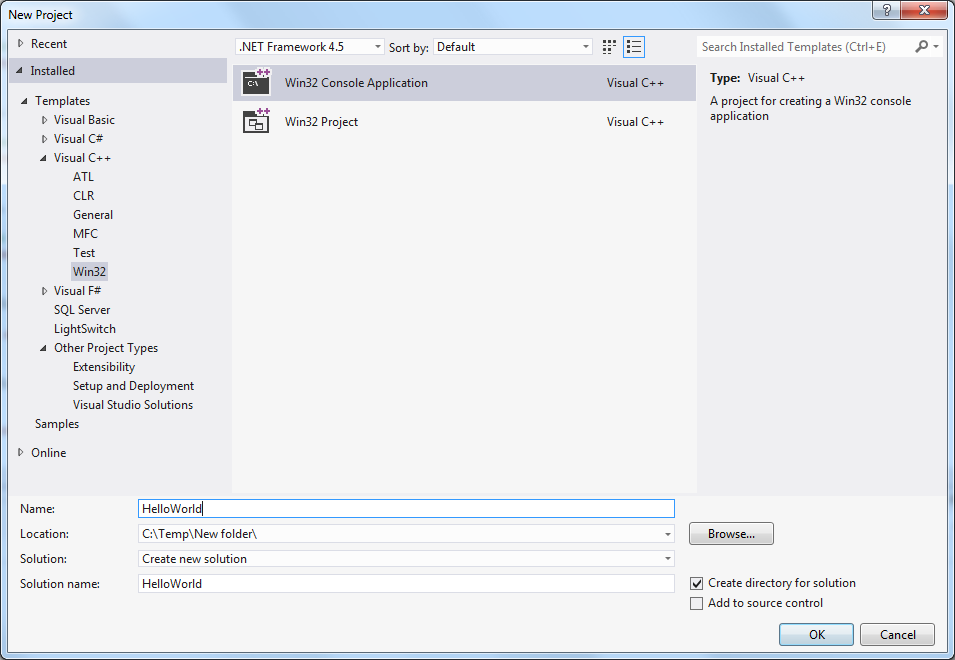
This lab has several short tutorials showing how to develop a few simple C/C++ and C# I/O applications for PC-based hardware using Visual Studio. I/O devices include USB and a serial port on a PC and various Phidgets to display and read sensor data. An example will also be ported to the small low-power ATOM embedded PCs. Next, a virtual com port will be used to transfer data to an mbed. Finally, C# will be used to setup a simple Windows GUI application.

Screen captures of the code and final output display can be used for the TA Checkoff for all parts other than the Phidgets LCD Time & Sensor Display Demos.

This lab uses Microsoft Visual Studio, which is already installed on the lab computers. You may install VS for free on your own computer through the [Microsoft DreamSpark](https://www.dreamspark.com/Product/Product.aspx?productid=93) program if you’d like.

**A short Visual Studio tutorial on C/C++ application development**

Start Visual Studio, select **File -> New Project** and then click C++ on the dialog window that opens and on the left first select **Visual C++->Win32** and then on the right **Win32 Console Application**. This sets up a project that reads and writes text to a command window (i.e., no Graphics or GUI).



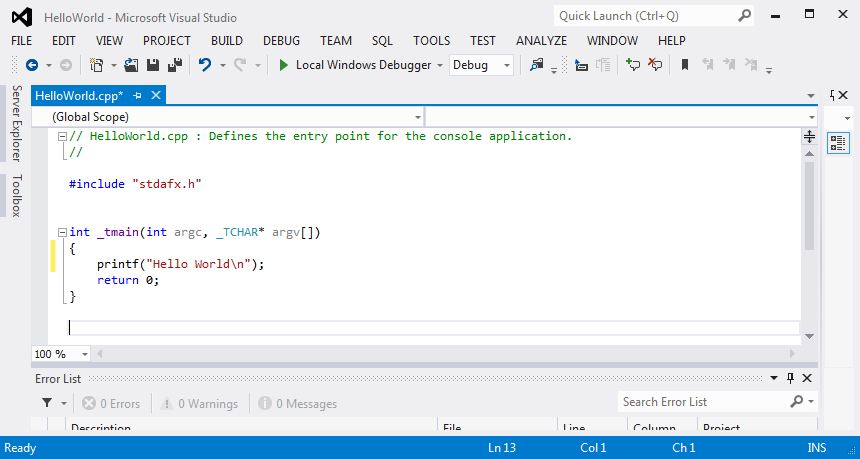
Setting up a Windows Text-only Console Application

Type **HelloWorld** for the project name, select a save location, and **click OK**. If you get a box which says “Welcome to the Win32 Application Wizard,” click **Finish** to continue.

**IMPORTANT!** Make sure you select a location which includes a *drive letter*, e.g. P:\ECE4180\Lab5 or C:\Temp\Lab5, and **NOT** a path which starts with “\\ad.gatech.edu\...”. Do not use the Desktop or “Documents” Library on the lab computers. If you save your projects to the C:\ drive of a lab computer, please copy the files to something you own (prism drive, flash drive, whatever) and delete them from the C:\ drive when you finish the lab.

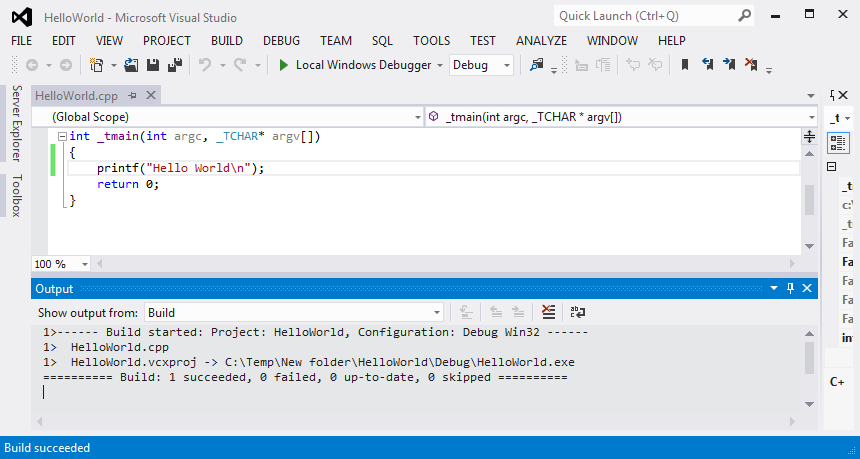
If the above is confusing, just save your projects to C:\Temp\ and delete them when you’re done.

It creates a console application that automatically includes the basic C/C++ source code for HelloWord. Type in the new **printf()** to the source code as seen below:



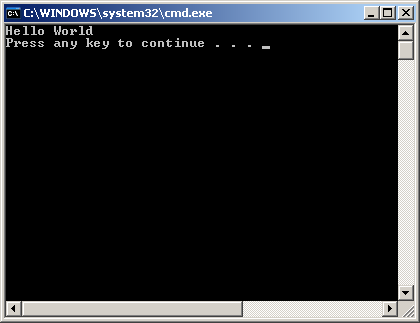
Source code for a C/C++ Hello World Console Application

Next to compile and link it, click **Build -> Build Solution.** It should compile and link (ie., build) with no errors (i.e, this is called succeeded in the output window) as seen in the output error log window at the bottom. In addition to breakpoints, there are a number of more advanced debug features that can also be used.



Compiling and linking a C/C++ Application Program

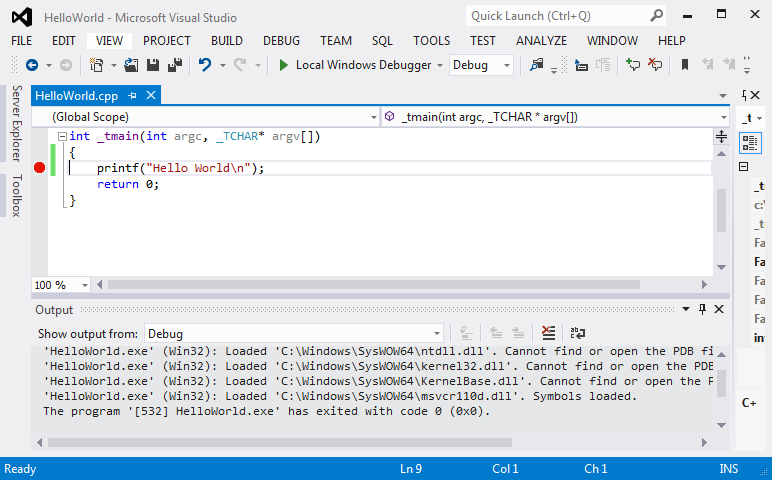
Now then program can be run and debugged. Select **Debug - > Start without debugging** to run the program. A command window with the HelloWorld message will pop up as seen below.



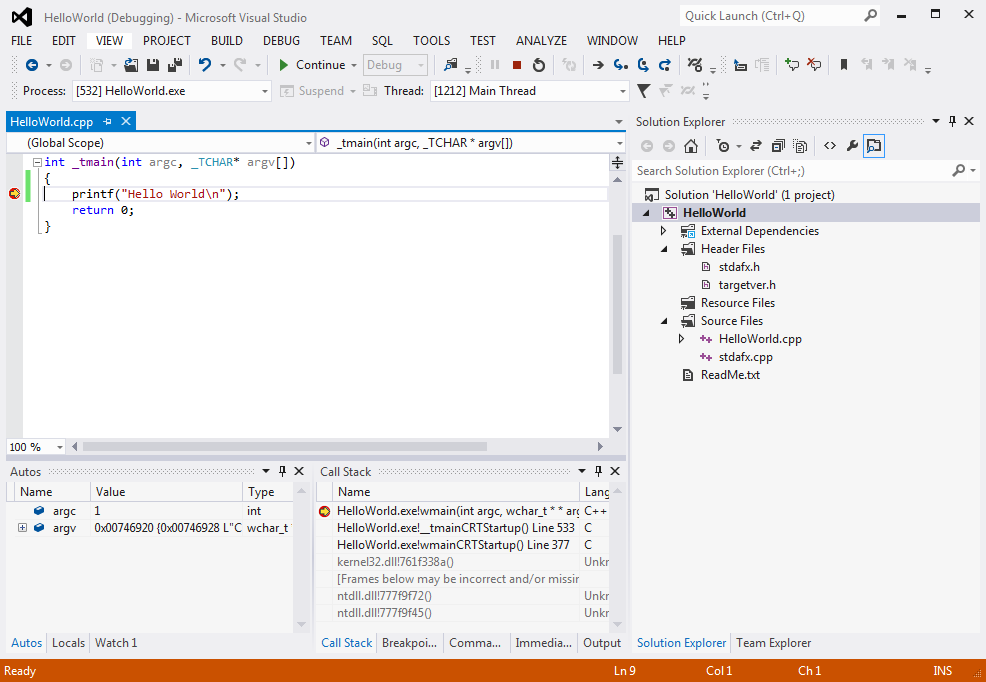
Console output window from the HelloWorld C/C++ application

Select the console window and hit any key to exit and return to the VS window.

Next back in the VS edit window set a breakpoint on a source line in the editor window by **clicking to the left of the line** with the mouse. A red dot should appear that indicates a breakpoint has been set as seen below.



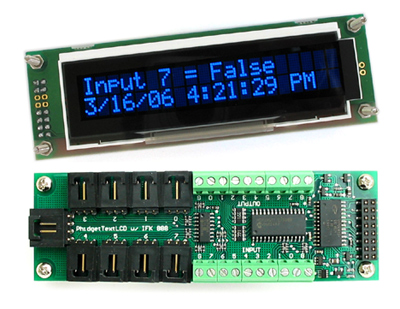
Run the program again using **Debug -> Start Debugging** and the program should run until it hits the breakpoint and stops. When it stops, note that a yellow arrow appears in the red dot showing which breakpoint was encountered.



**Lab Checkoff:** Demo your Hello World program running with the debugging breakpoint.

**Fidgeting with Phidgets**

[Phidgets](http://www.phidgets.com) offers a wide assortment of low cost USB I/O devices and sensors. They come with USB drivers for the devices and sample application code in C/C++ and C#. The [free Phidgets USB driver](http://www.phidgets.com/docs/OS_-_Windows#Quick_Downloads) must first be installed on the PC. It should already be installed on all of the 4180 lab PCs. C# examples install with the driver. Additional Phidgets [C/C++ examples](http://www.phidgets.com/downloads/examples/VCpp.zip) can be found online. The Phidgets device driver adds Phidgets C/C++ APIs that can be used in user applications. When writing new code for Phidgets, in addition to these code examples there is also an [extensive Phidgets API manual](http://www.phidgets.com/documentation/web/cdoc/index.html).

****

A Phidgets USB Text LCD is seen above and along with the Interface Kit (on back side of LCD board)

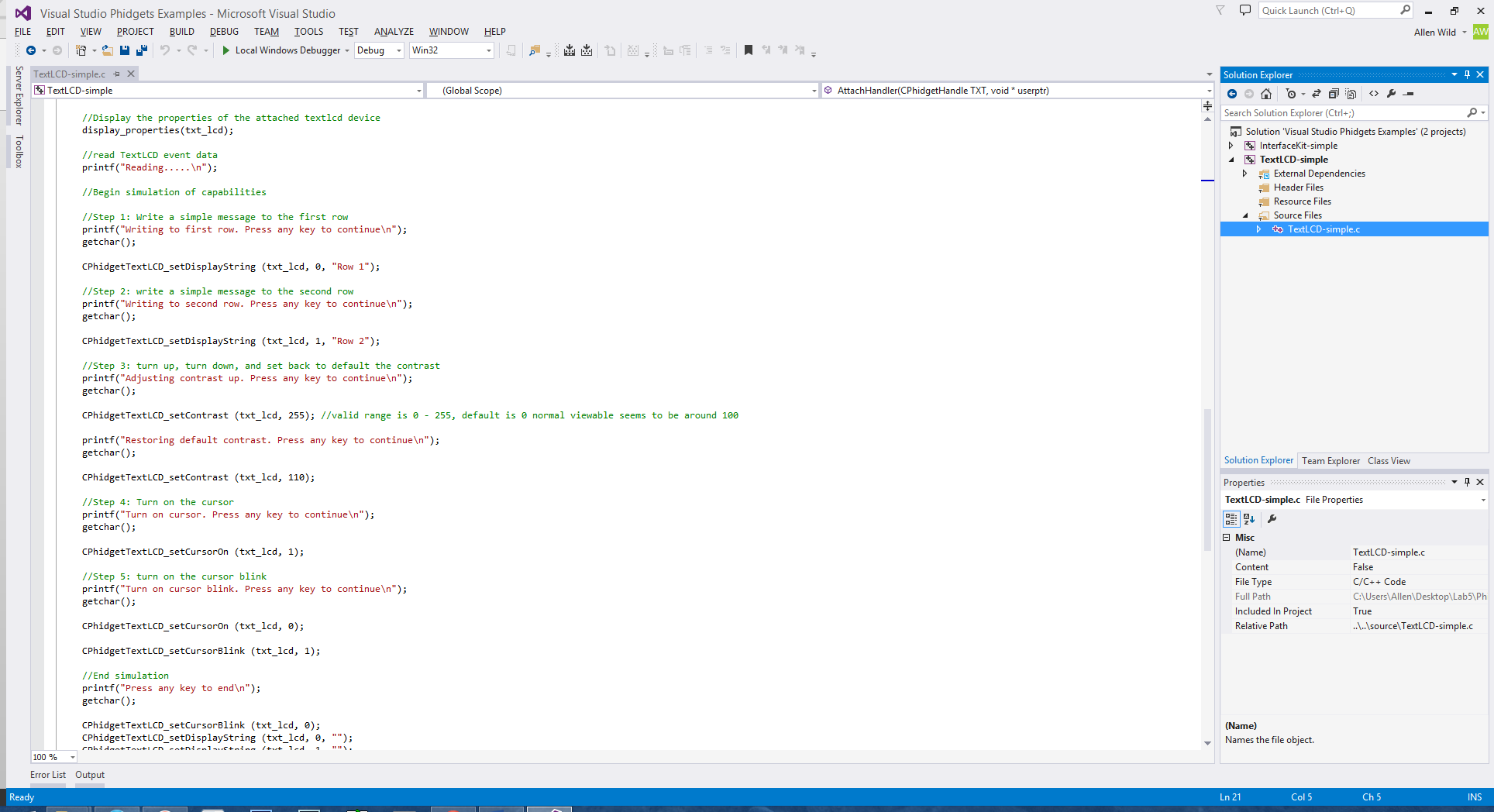
**Using Phidgets APIs for C/C++ applications**

**An introduction to Visual Studio project management**

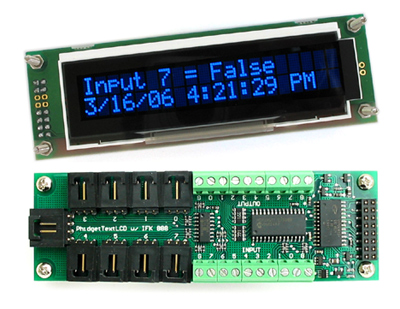
In Visual Studio, a Project is collection of source files and resources as well as configuration information to build into an application or library. A Solution is a collection of related projects which can be compiled and managed together. In the first part of this lab, you created a new solution with a single project.

**Download the Phidgets Examples Code from T-square** **Resources** and unzip it somewhere with a drive letter (i.e. not the lab computers’ desktop or documents). Double click the “Visual Studio Phidgets Examples.sln” file to open the solution.

The upper-right portion of the Visual Studio window is the Solution Explorer, it contains 2 sample projects. Double-click the TextLCD-simple project to expand its contents. Under “Source Files,” open TextLCD-simple.c and Examine the C/C++ source code for the Phidgets Text LCD module. Read through it and note the use of Phidget API calls. When writing new code for Phidgets, in addition to these code examples there is also an [extensive Phidgets API manual](http://www.phidgets.com/documentation/web/cdoc/index.html).



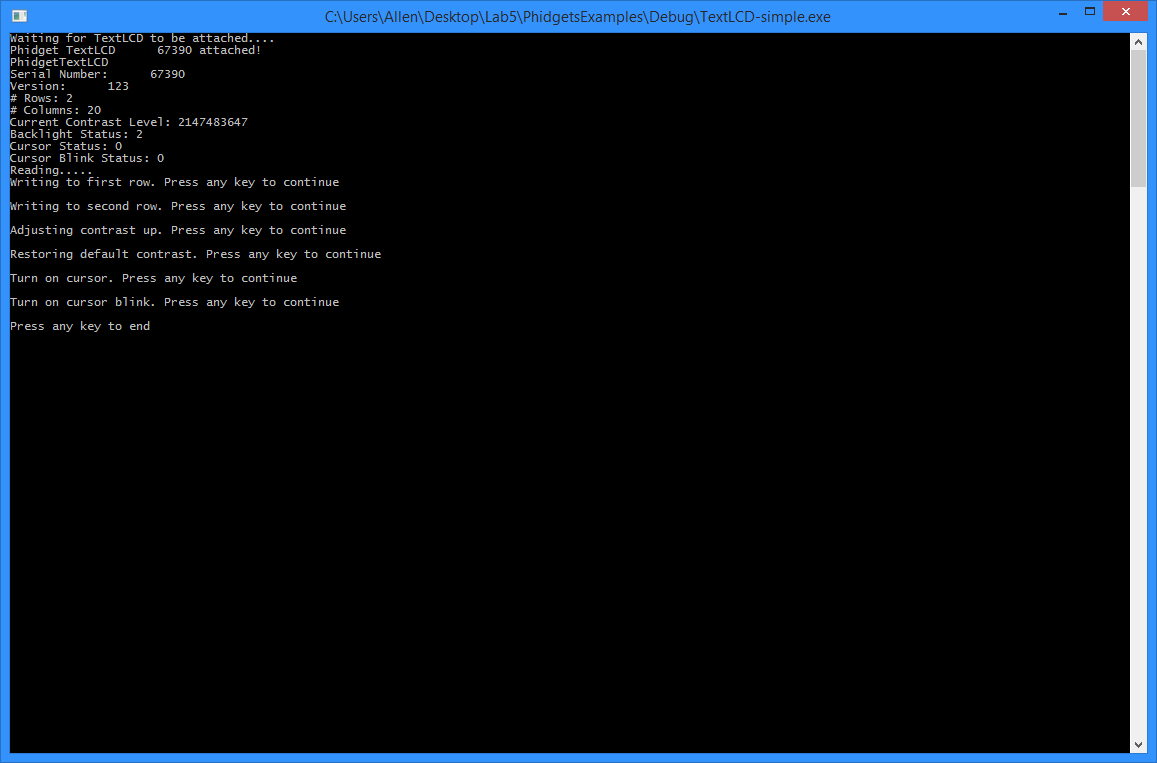
Plug in a Phidgets TextLCD module into one of the computer’s USB ports.

****

Next to compile and link it, click **Build -> Build Solution**. It should compile and link (ie., build) with no errors (i.e, this is called succeeded in the output window) as seen in the output error log window at the bottom. The Phidget’s API are defined in a library file (phidget21.lib) provided with the driver that must be linked to any Phidgets project. This should already be setup correctly in the example projects, if it is missing all Phidget’s APIs will be undefined.

If you get a bunch of “unresolved external symbol” errors, then the phidget21.lib library has not linked correctly. Right-click on the TextLCD-simple project in Solution Explorer and select **Add->Existing Item**. Navigate to the PhidgetsExamples directory and select phidget21.lib. Then retry the build and it should work.

Then run it **Debug - > Start without debugging** and watch for the console window that displays test info and pauses for user input. Hit a few characters in the console window and some text test messages should appear on the Phidgets LCD attached to the PC.



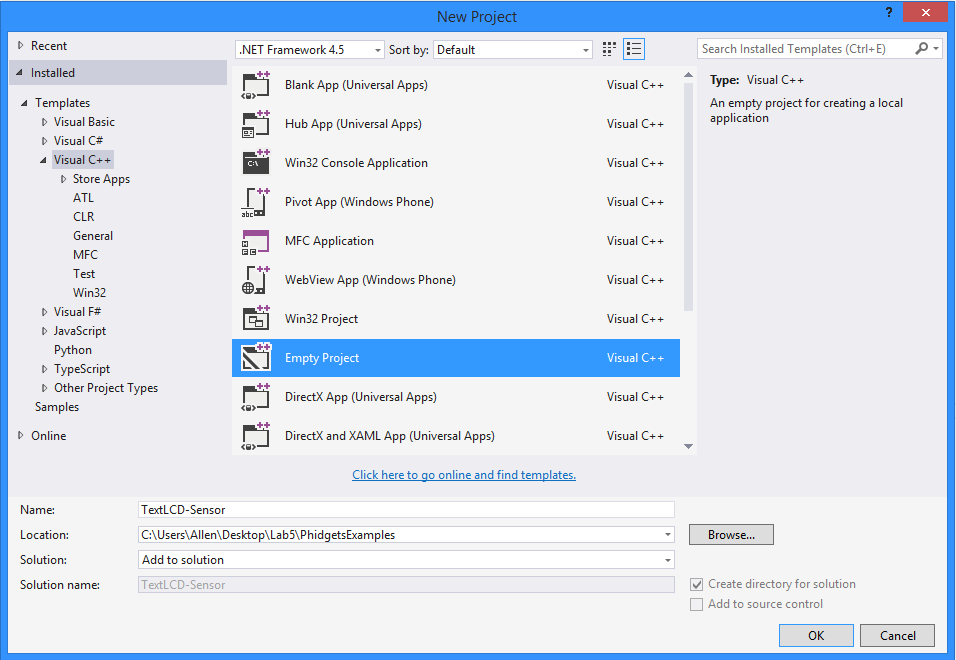
**Lab Checkoff:** Demo the TextLCD-simple project running correctly.

**Developing a custom Phidgets Application**

For this portion of the laboratory experiment, you have to develop a Phidgets application that reads a sensor (you can select any interesting one available in the lab) and display the sensor’s output on the TextLCD (scaled in a meaningful way) along with the current time. Use the search feature in Visual Studio’s online help to find an API that returns the current time (hint, look up the standard C time() and strftime() functions). Print the current time (HH:MM:SS) on line 1 of the LCD and the sensor value on line 2. You might want to select a sensor that you are likely to use again in your final design project.

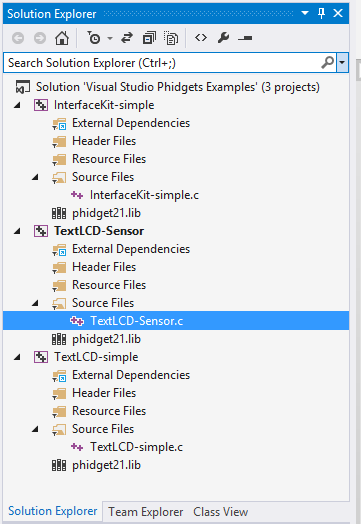
**Creating a new Project within your Solution**

1. Select **File->New->Project**. Choose the “Visual C++ -> Empty Project” project type, name it “TextLCD-Sensor,” and set “Solution” as “Add to solution.” This should change the location to the PhidgetsExamples directory from before. Press OK when you’re done.



1. Next, you need to add the relevant files to your project in order to develop a custom application. Right-click on the “Source Files” folder of your new project in Solution Explorer and select **Add->New Item**. Select a C++ file and name it “TextLCD-Sensor.c” This will create an empty source file for your project.
2. Adding the Phidgets library to your project: Right-click on your project in Solution Explorer and select **Add->Existing Item**. Navigate to the PhidgetsExamples directory and load the phidget21.lib file.
3. Now it’s time to add the PhidgetsExamples directory to the compiler’s include path so it can find the phidget21.h header file. Select **Project->TextLCD-Sensor Properties…** to open the project properties. Navigate to **Configuration Properties ‑> C/C++ -> General** on the left, and enter “..” (2 periods) for **Additional Include Directories**. “..” is a shortcut for “parent directory” which in this case is the PhidgetsExamples directory containing phidget21.h.
4. Finally, right-click your project in Solution Explorer and select **Set As StartUp Project**. This tells Visual Studio which project in the solution to launch when you click Run.
5. Copy/paste the contents of TextLCD-simple.c to TextLCD-Sensor.c, this will give you a template and much of the overhead code for your custom application.

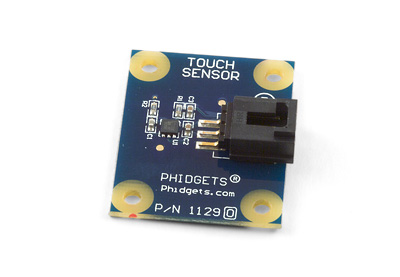
When everything is added, your Solution Explorer should look like this



Build your solution and verify that everything compiles/links correctly. Run your project and verify that it runs correctly (which at this point should be exactly like TextLCD-simple).

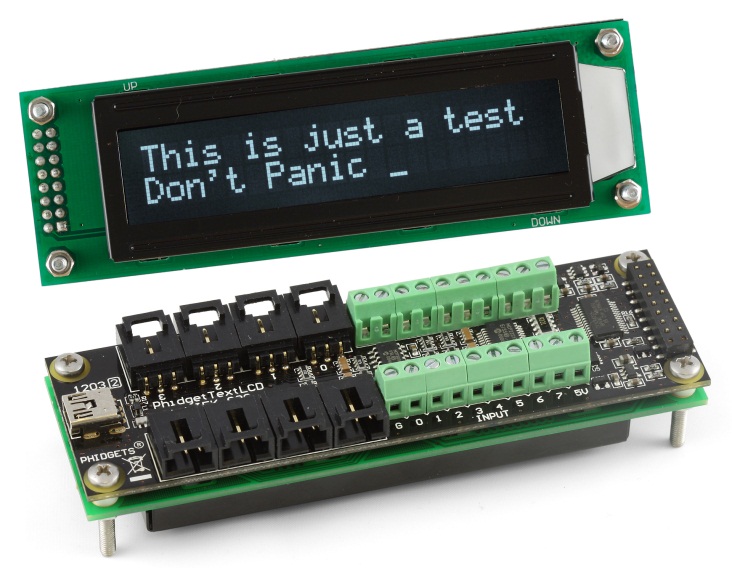
**Selecting a Sensor**

There are several options for sensors. The interface kit on the backside of the LCD can read in eight analog inputs. Analog Phidgets sensors are mounted on small boards and use a 3-pin black connector as show below. The InterfaceKit-simple has examples showing how to read analog sensors using the Phidgets APIs.

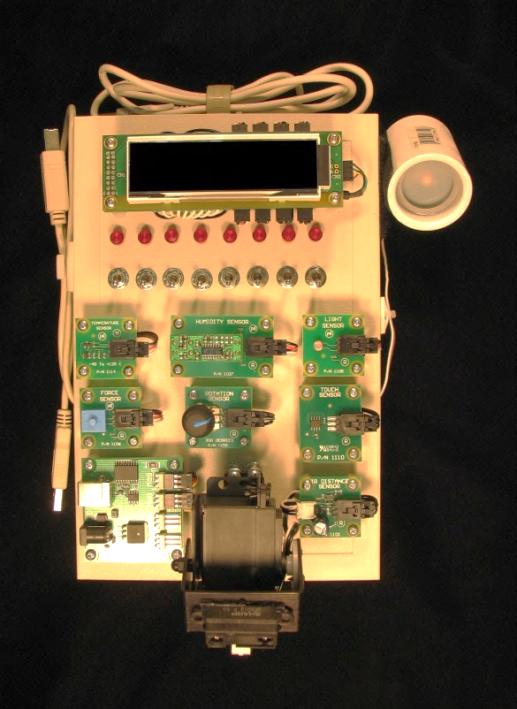


A Phidgets analog sensor board and the required 3-pin cable

The cable attaches it to one of the black connectors on the back side of the Text LCD module.



There are several boxes in the lab that have a TextLCD with several analog sensors already hooked up. If you can’t get hold of an available box there are also several LCD modules and sensors that are not mounted.

****

**Lab Box of Phidgets – A TextLCD with eight analog sensors and a servo controller**

There are eight toggle switches and eight LEDs on the Phidgets box that are already hooked up to digital I/O bits on the Phidgets LCD/Interface board. There is also a second USB cable that is needed if you use the servo controller board that controls the two servos on the pan tilt sensor mount that moves the IR distance sensor. If you use more than one servo or any servo other than servo 1, the servo board needs an AC adapter for extra power (i.e., a USB device can only take .5 AMP from the cable and a single servo’s DC motor consumes nearly that much current when moving).

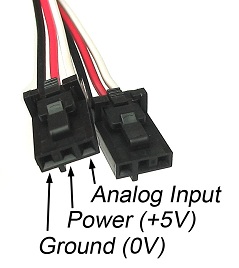
Here is a handy table for the default analog sensor hookups to the Phidgets LCD/Interface board’s A to D (ADC) on the box of Phidgets in the lab. This value is needed as a parameter in a Phidgets API call that selects one of eight analog inputs to an analog mux. The mux output then feeds the selected sensor’s analog signal into the analog to digital convertor’s input.

Table of Phidget Sensor Indices.

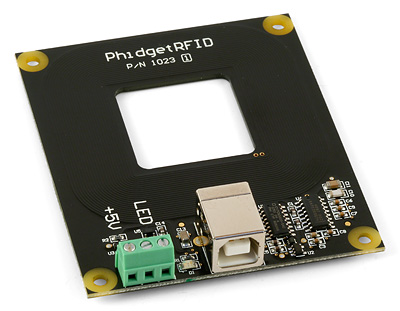
**Sensor Type Index**

|  |  |
| --- | --- |
| Humidity Sensor | 0 |
| Temperature Sensor | 1 |
| Force Sensor | 2 |
| Light Sensor | 3 |
| Touch Sensor | 4 |
| Rotation Sensor | 5 |
| IR distance Sensor | 6 |
| Motion Sensor | 7 |

If you do not use the box of Phidgets, but connect your own Phidgets analog sensor board to the LCD module use the three pins cables with the black connectors and plug them into the black sockets on the back of the LCD module.



There are also several more complex Phidgets sensors that contain their own USB interface with another USB cable. RFID, linear and circular touch sensors, relays, the servo, and DC motor controllers fall into this category. They each also have an example project showing the APIs needed for that device. A standalone USB sensor needs a bit of code based on the examples to open the device and read the sensor values. However, it is probably easier to use one of the analog sensors connected to the TextLCD/InterfaceKit for this lab.

****

**Standalone Phidgets USB RFID reader module**

**Writing your custom Phidgets program**

Included with the PhidgetsExamples solution is an InterfaceKit-simple project. Analyze the code in this example and copy the relevant initialization functions and code into your TextLCD-Sensor.c file. Make note of the SensorChangeHandler() function; this is the callback which gets executed whenever an analog sensor’s value changes. Index is the number of the sensor and Value is the value it’s changed to.

Your TextLCD-Sensor program should display the current time on the first line as HH:MM:SS, and a sensor value on the second line.

Hint: You will probably want to move the declaration of your “CPhidgetTextLCDHandle txt\_lcd” to global scope (outside of textlcd\_simple()) so that your SensorChangeHandler can update the TextLCD.

Hint 2: Visual C++ doesn’t support some normally-standard function calls. Use sprintf\_s() instead of snprintf() and localtime\_s() instead of localtime(). The IntelliSense popups will help you with the arguments to these functions (and so will Google).

**Lab Checkoff:** Demo your custom time/sensor Phidgets program.

**Extra Credit (+2.5):** Rather than a numeric sensor value, use the second line to display a “bar graph” scale of your sensor’s value using an appropriate number of ‘=’ characters to represent the current value’s fraction of full-scale.

After getting your custom Phidgets application code to run on the PC, the next step is to run the code on one of the embedded ATOM PCs. Find your project directory on the disk and the application code is located in the project’s Debug subdirectory. Copy the application code to a USB flash drive to move it to the ATOM PC. Move over the Phidgets LCD device(s) to a USB connector on the ATOM PC and demo the application running on the ATOM PC. Setup the ATOM PC so that it runs the application automatically after booting without the need for use input (use the Windows [Startup folder](http://lifehacker.com/5829375/how-to-start-a-program-automatically-when-your-computer-boots)). This is handy for headless embedded devices that might not have a monitor or keyboard.

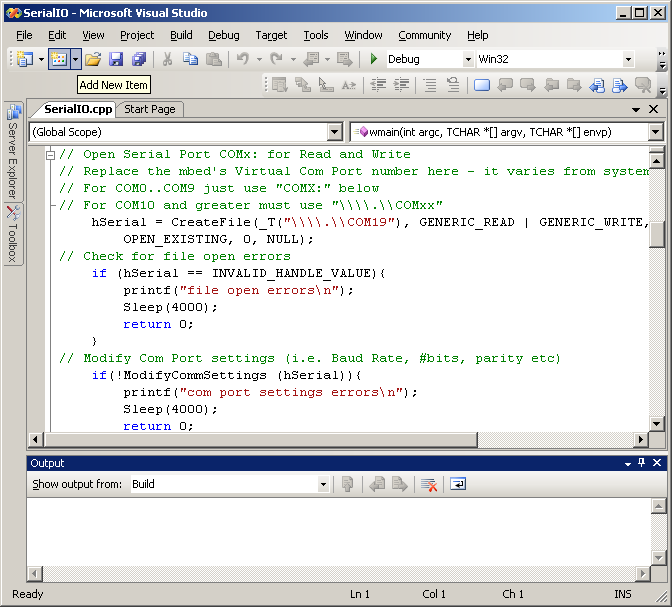
**Lab Checkoff:** Demo your custom Phidgets program auto-starting and running on one of the ATOM PCs.

**Using the serial port to communicate with I/O devices.**

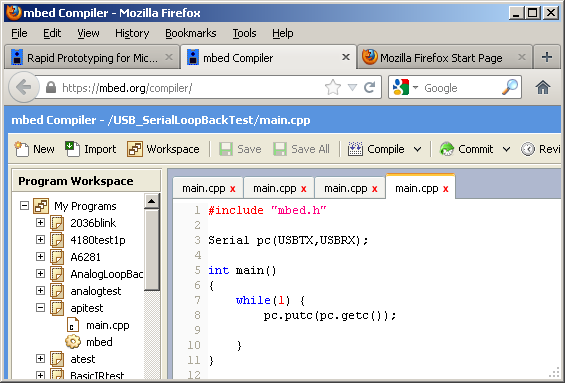
Many I/O devices interface using a serial port or COM port on the PC. Examples include GSM modems, GPS units, Magnetic card readers, and Barcode readers.

Mbed also uses a virtual com port via the USB cable, but from a software perspective it operates the same as a real COM port. Note the COM port number that the mbed chip is using on the PC by checking it with a terminal application program while the mbed is plugged into a USB port. It will be needed before the next step.

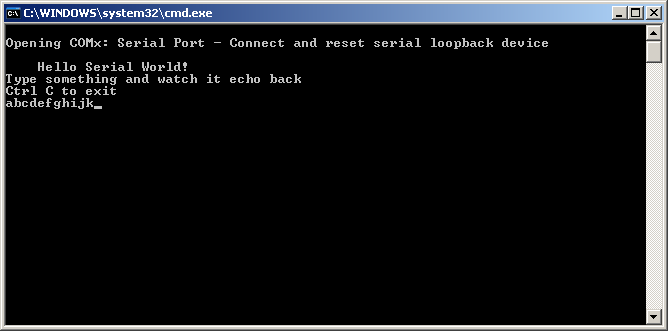
An example project for VS2005 called “Serial” has been setup that contains code to send data out on the serial port and read it back in. Download a copy of the project directory from www.ece.gatech.edu/~hamblen/489X/classmat/Labs/serial.zip and unzip it to the C:/temp directory. Open the project, and in the source code, edit the entry for the com port by typing in the number of your local mbed as seen below (with your COM number). Finally, rebuild the project.



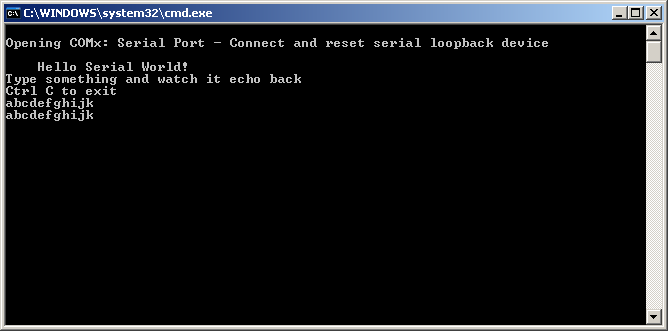
For the test program to work, the mbed should be running code that reads in serial data and sends it back (i.e. called a serial loopback). The code needed is shown below:



After building the project with the COM port change. Start the program with **Debug->Start without Debugging**. If the correct COM port change has been made and the mbed is attached, the COM port should open with errors and wait for user input. Type a few test characters in the window as seen in the next screen shot.



Hit return, and the line will be sent over USB to mbed. The mbed program sends it back (echoes) and the same line should appear in the console window.

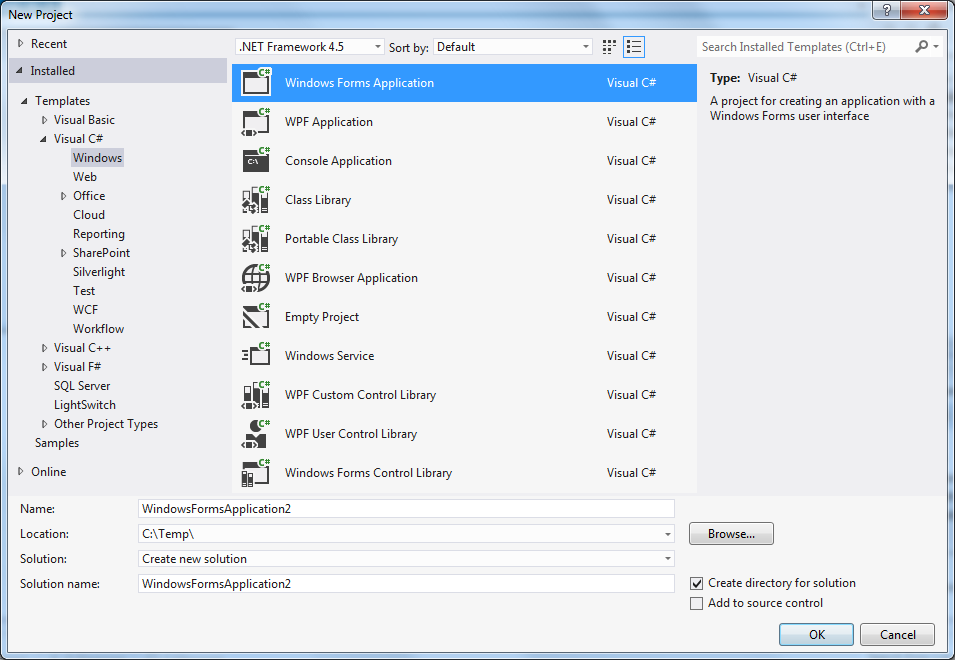


Type Ctl C and then another character to exit and close the window. This program showed how to use serial port to transfer data between a PC and mbed. It might come in handy for any serial devices in the final design project.

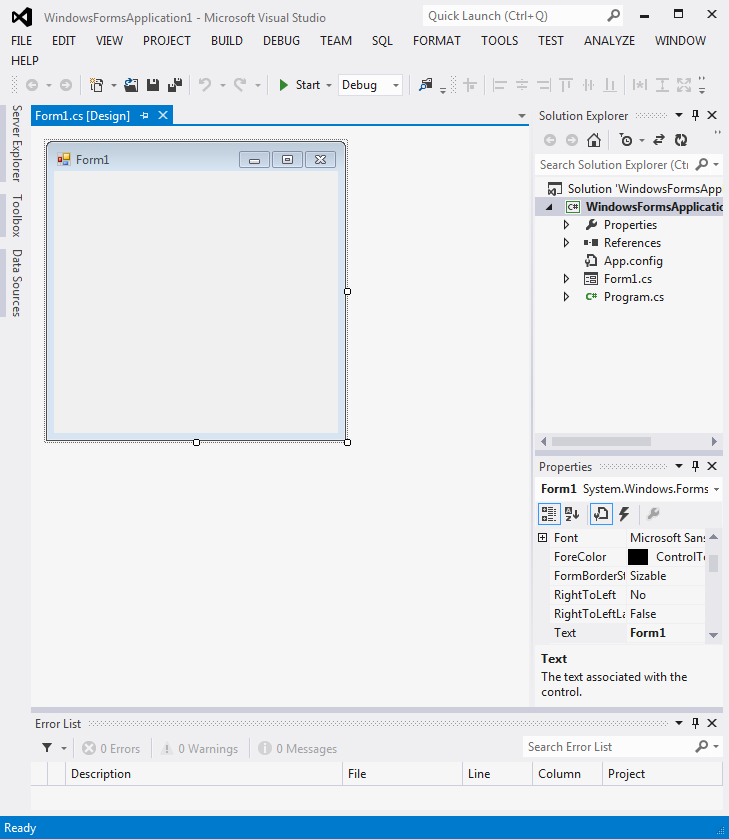
**Lab Checkoff:** Demo your COM port echo working correctly.

**Developing a C# Windows Application in Visual Studio**

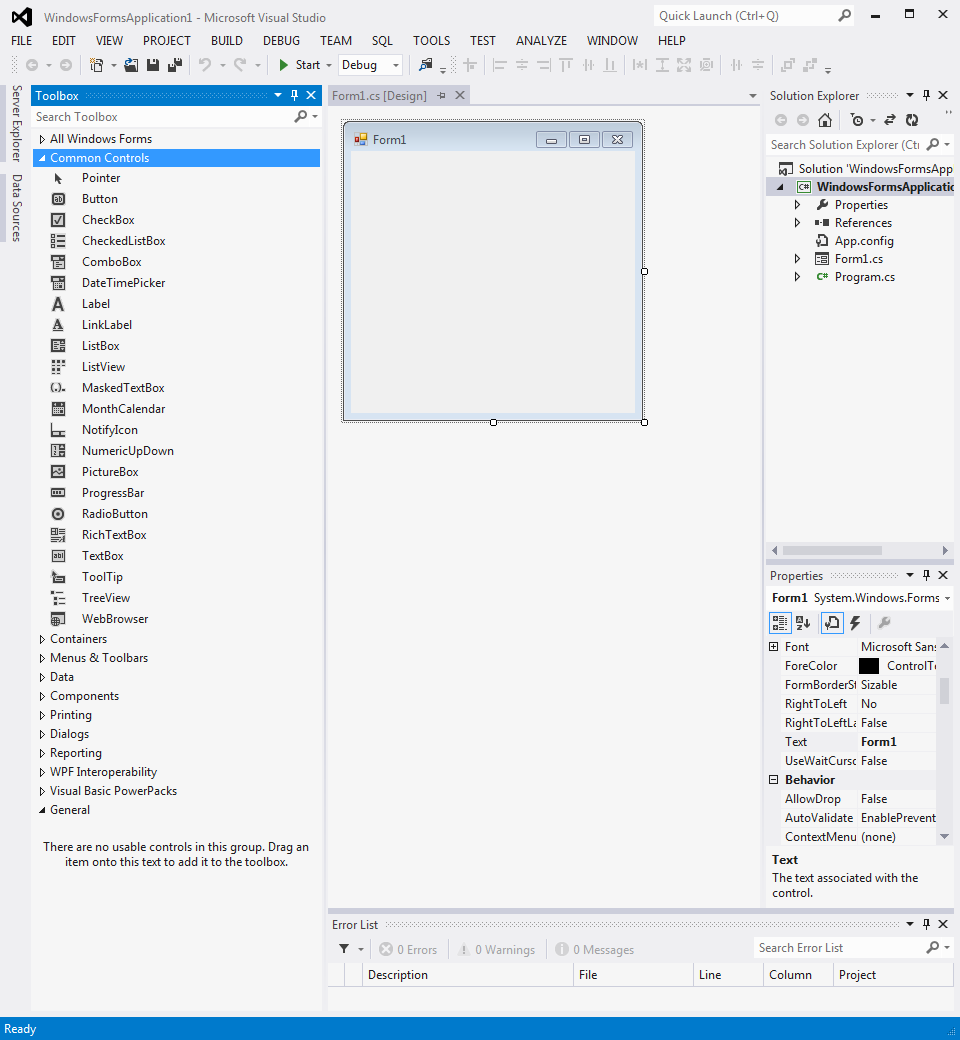
Start Visual Studio and as seen below select Visual C# Window project type (left)with a Windows Forms Application Template (right).

****

**Click OK** and a new project will be setup in the graphical forms editor with a blank window as seen in the next screen capture.

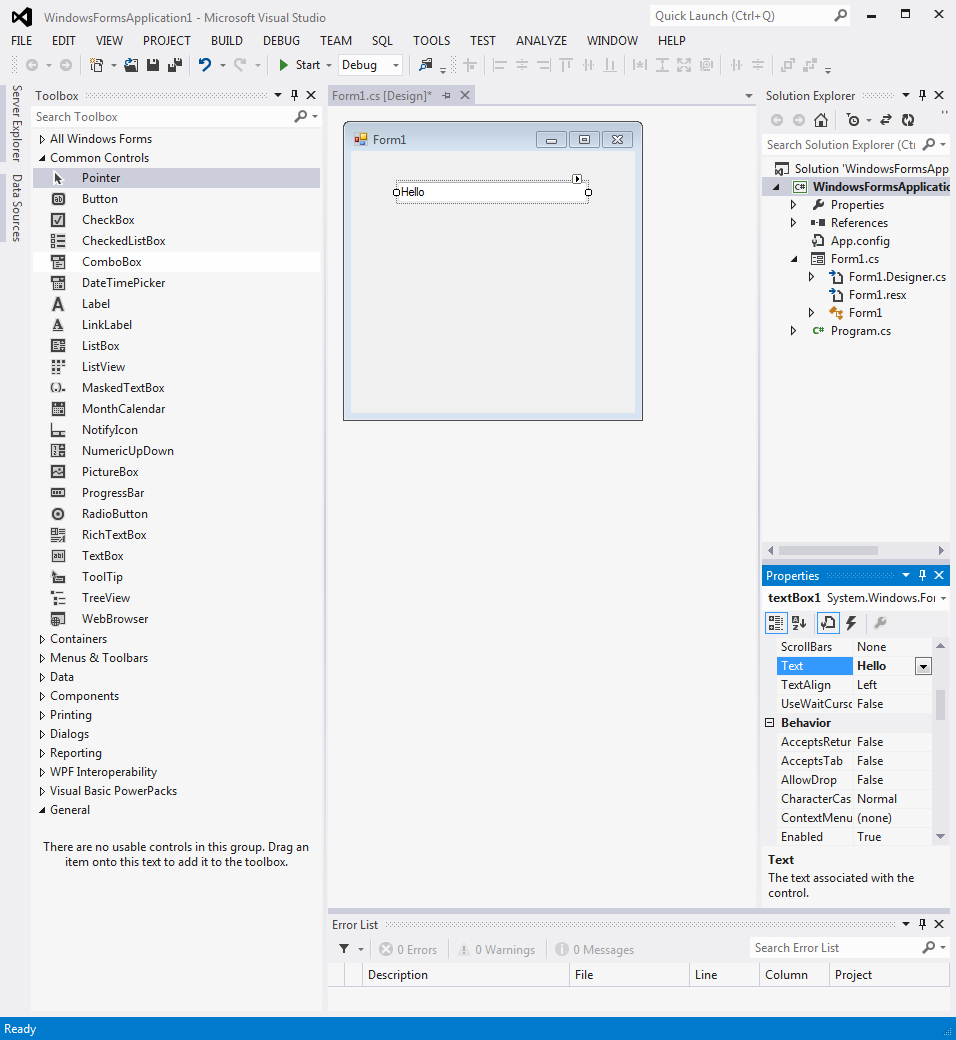
****

Select **View-> Toolbox** and a new window should appear as seen below

****

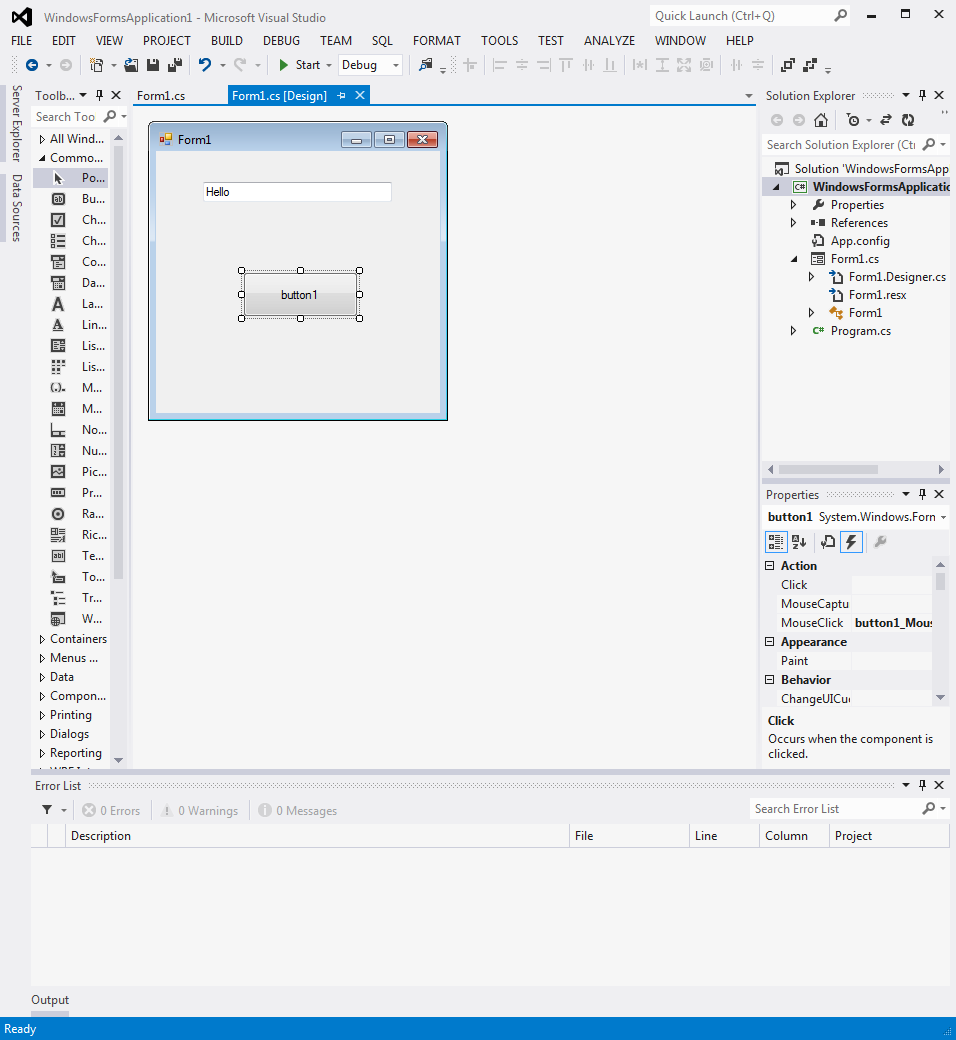
**Select Common Controls,** Scroll down and **find “textbox”. Add a textbox to the forms drawing** using the mouse**. Right click on the** textbox just drawn and select properties.Enter **“Hello”** for the Text property associated with the new textbox.

It should look like the image below.

****

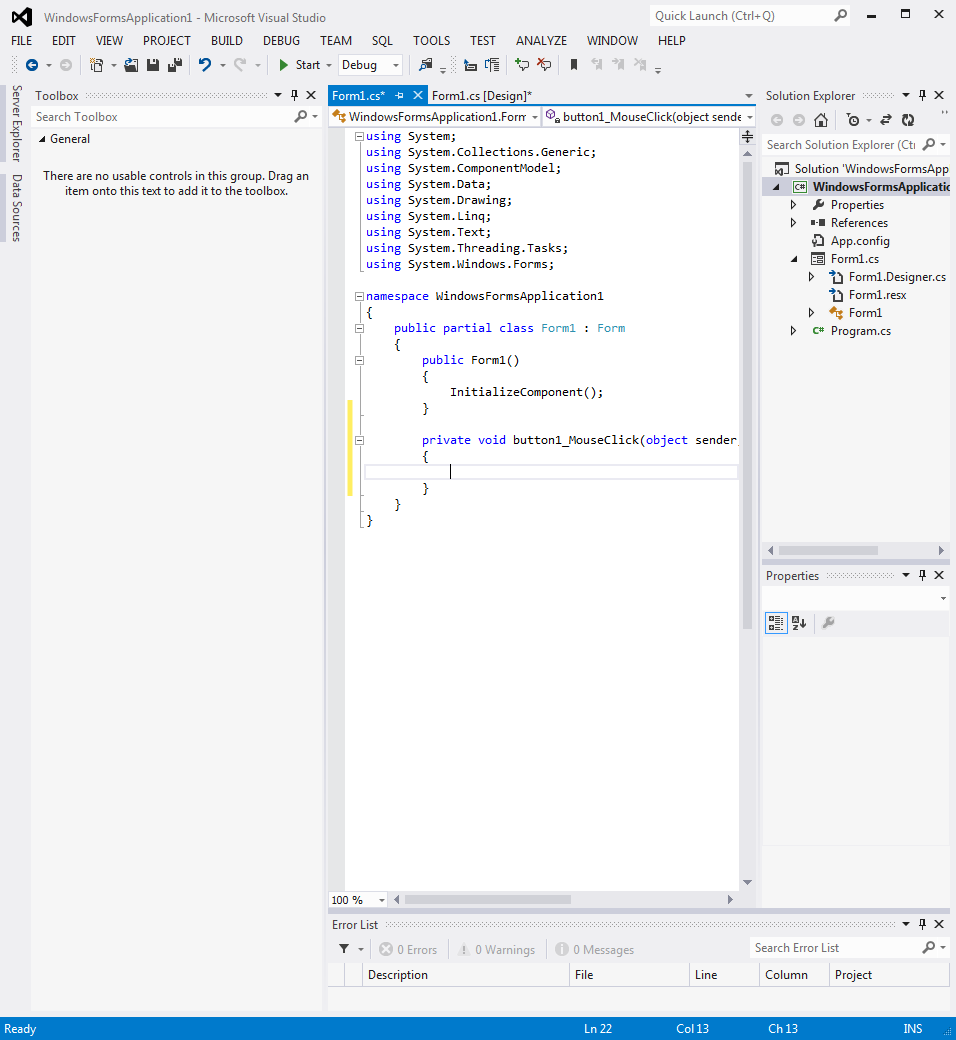
**Click on the toolbox side tab** and **select button.**

**Add a button** and it should look like the following image**.**

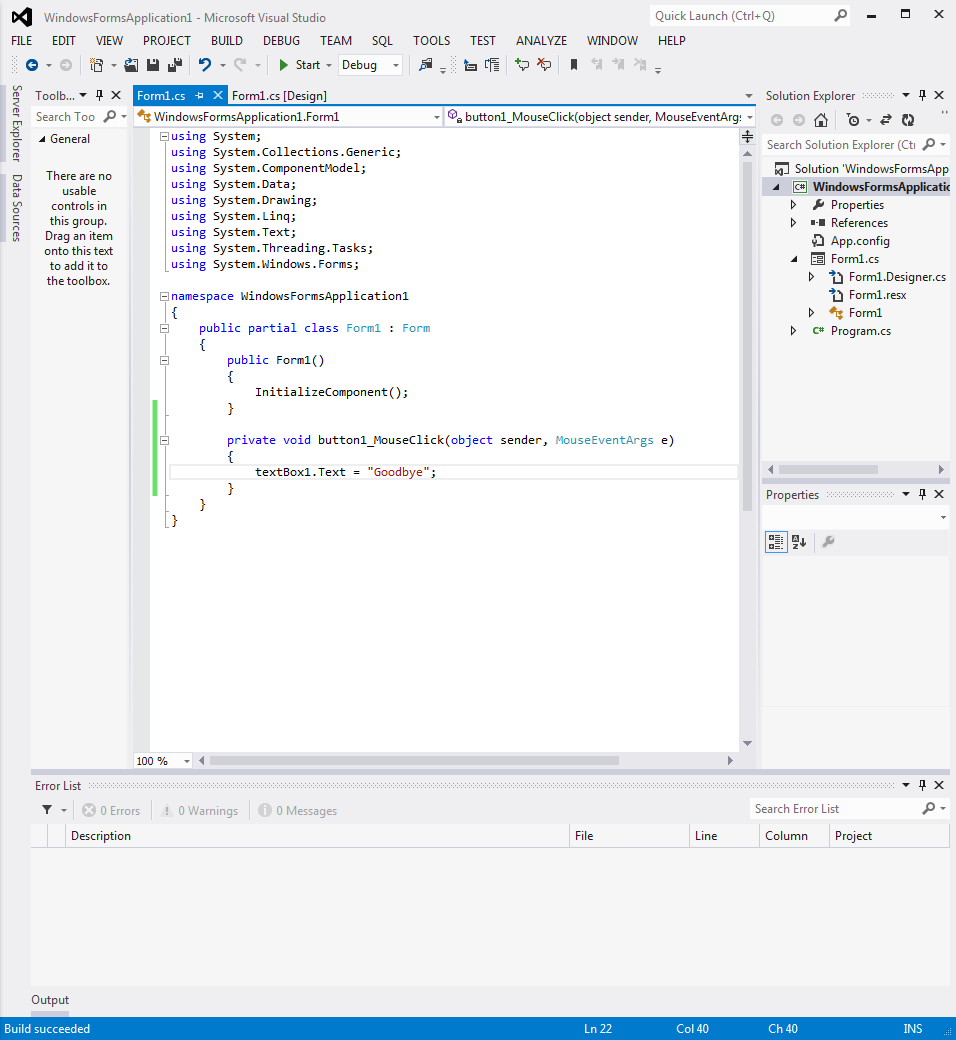
****

Next, notice the lightning bolt icon in the properties window above. **Click the lightning bolt, double click on “click”** and watch out!

The C# forms wizard automatically takes you to the code that executes when a button click occurs as seen below.

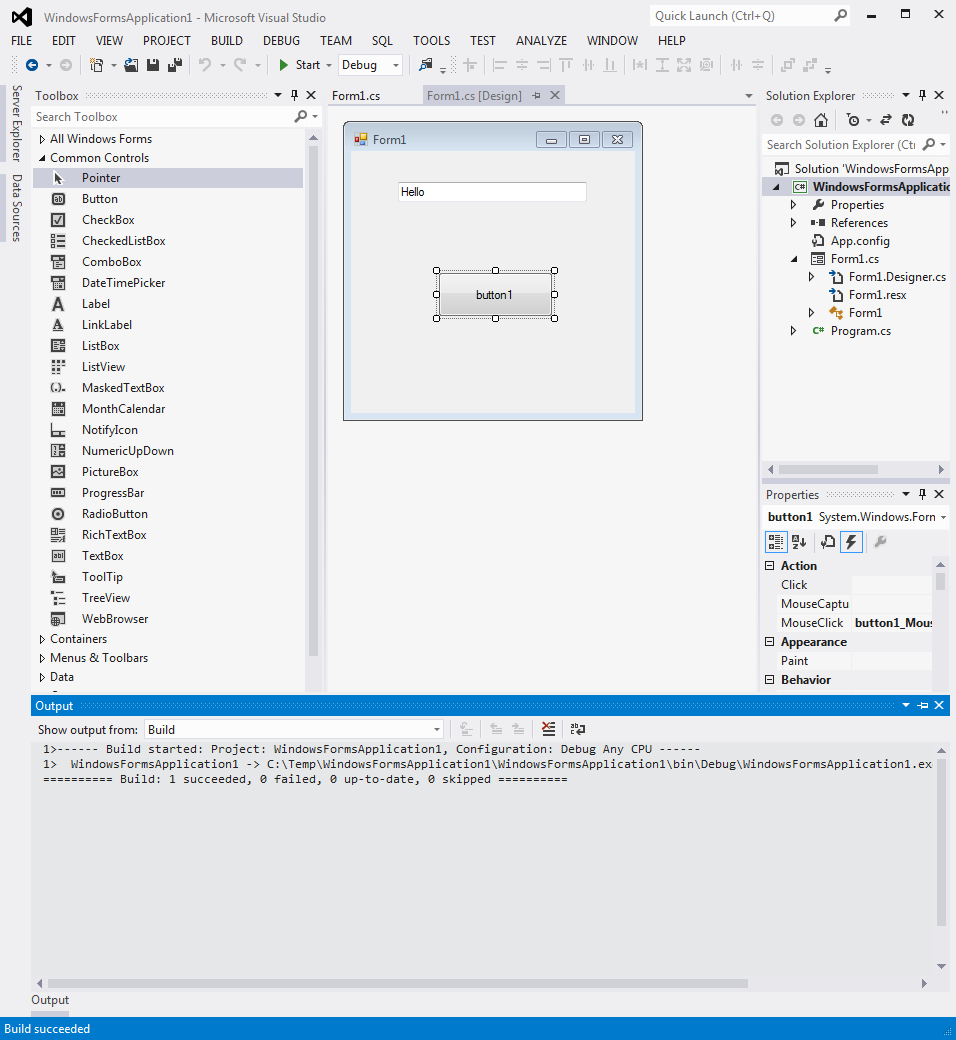
****

Type in a line of code to execute when the button is hit that changes the text in the text box as seen in the next screen capture

****

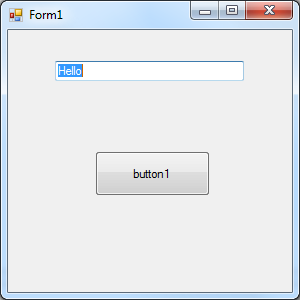
This code should change the textbox message to “Goodbye” when the button is hit. Next at the top menu. **Select Build->Build Solution.**

Your new C# windows application should build as seen below.

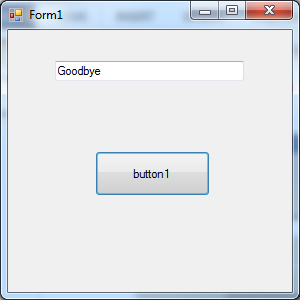
****

It is now ready to run. **Select Debug->Start without debugging.**

The window below should appear.

****

**Click the button** and you should see the text change to

****

**Click on the right corner X** to exit the application and delete the window.

**Lab Checkoff:** Demo your basic C# application

**C# I/O Example programs**

For C# application programs, C:\Program Files\Phidgets\Examples also has Phidgets C# examples for Visual Studio as seen below. The 4180 textbook also has an example showing how to open, read, and write to the serial port in C#.

