

(PROJECT 1): INTRODUCING THE LAB AND DEVELOPMENT ENVIRONMENT

J. Vining

ECE/CSE 474, Embedded Systems

University of Washington – Dept. of Electrical and Computer Engineering

Winter 2021

REV: 8 Jan 2020

TABLE OF CONTENTS

1.0	INTRODUCTION	5
2.0	PREREQUISITES	5
3.0	BACKGROUND	5
3.1	Engineering Process	5
3.2	Target Platform	6
3.3	Programming and Downloading Code to an Embedded System	6
3.4	Development Environment	7
4.0	PROJECT OBJECTIVES.....	8
5.0	CODEBLOCKS AND THE C LANGUAGE – WORKING WITH C IN THE PC ENVIRONMENT	8
5.1	Building a Program.....	9
5.2	Building Your Own Applications	16
5.3	Troubleshooting	17
6.0	LEARNING THE ATMEGA ENVIRONMENT AND TOOLS.....	17
6.1	Working with the Development Environments and Target Platforms: C for the ATmega Environment.....	18
6.2	First Steps: Loading Code to the ATmega	20
6.3	Building, Editing, and Running a Program	21
7.0	DELIVERABLES	23

LIST OF TABLES

No table of figures entries found.

LIST OF FIGURES

Figure 1 – CodeBlocks Download Options	9
Figure 2 – CodeBlocks Splash Screen.....	10
Figure 3 – CodeBlocks Project Type Selection.....	10
Figure 4 – CodeBlocks Console Welcome Screen	11
Figure 5 – CodeBlocks Implementation Language Selection.....	11
Figure 6 – CodeBlocks Project Directory Selection.....	12
Figure 7 – CodeBlocks IDE: Removing Files from Project	12
Figure 8 – CodeBlocks IDE: Adding Files to Project	13
Figure 9 – CodeBlocks IDE: Selecting Build Type.....	13
Figure 10 – CodeBlocks IDE: Opening C File for Editing	14
Figure 11 – CodeBlocks IDE: Setting the Compiler Installation Directory	15
Figure 12 – CodeBlocks IDE: Console Window.....	15
Figure 13 – Arduino Mega.....	17
Figure 14 – Arduino IDE Screen	18
Figure 15 – Arduino IDE: Selecting the Target	19
Figure 16 – Arduino IDE: Selecting the COM Port & Connecting to the Target	20
Figure 17 – Installing a Library in the Arduino IDE.....	22

1.0 INTRODUCTION

This project has two main purposes:

1. The first is to **introduce the C language**, the CodeBlocks C compiler, explore some of the important aspects of the C language including pointers and multiple file programs, then practice simple debugging.
2. The second is to **introduce the Arduino ATmega 2560 microcomputer**, the Arduino subset of the C language, the associated development environment, and the world of embedded applications.

Like a typical embedded system, we have a hardware piece and a software piece. This term, our major focus will be on the software piece.

For each of the projects, you are strongly encouraged to **read through the entire project specification before trying to start designing, writing code, or exploring.**

2.0 PREREQUISITES

Familiarity with data types, data structures, as well as standard program design, development, and debugging techniques. No beer or shenanigans until the project is completed. I really mean it..

3.0 BACKGROUND

3.1 Engineering Process

At this stage in the engineering process, playing with the “toys” often seems to be the most exciting part...the documentation, formal design, and so on, is, well, often seen as rather boring. Why do I have to go through all this...why can't I just go to the Internet and find something like the design and make a few modifications and be done?

In the real-world, documentation and formal design are absolutely critical parts of all phases of the engineering development process. It's also very important to recognize that the answers to most interesting real-world engineering problems originate in our brains, discovered as we use our imaginations and knowledge to creatively apply the underlying

theory and tools; they are reached through our persistent hard work and diligence. **We challenge you to explore, to think, and to make discoveries.**

Sometimes your instructor or TAs will have the answers and sometimes we won't. This platform and development environment are complex.

Regardless of the specific platform/environment used, embedded systems development requires at least the following:

- An understanding of the problem that we are trying to solve. This is the key point.
- A target platform on which to develop the application.
- A mechanism for programming the target platform.

3.2 Target Platform

Target platform is a term used loosely in industry to mean the actual piece of hardware that performs the computation and/or control – the place where our application (the software) will ultimately reside and run. For this course, the Arduino ATmega 2560 will be our target platform.

3.3 Programming and Downloading Code to an Embedded System

In this lab, and for the typical microprocessor-based application, the software will be written in the C language. On many occasions, the C++ language or, on the opposite extreme, the processor's assembly language, may be used for what are called hard real-time systems, but Java is rarely used. On a programmable logic device like an FPGA, it may be a mixture of C and Verilog.

The term to program here has two meanings:

1. The more traditional embedded sense and simply means writing software to control a given piece of hardware (in this case the target platform and any peripheral devices that may be connected to it – see above).
2. In the embedded world, programming the target also means storing the executable into memory on the target.

The characteristics of the target platform can vary greatly, and so too can the mechanism used to program the target platform.

In this course, you will develop code on your computer, compile it, and then transfer to the target platform. We call this transfer downloading to the target platform.

3.4 Development Environment

The laboratory and development environment will be a combination of the traditional PC, the CodeBlocks IDE, the Arduino Mega microcontroller and peripheral devices, and the Arduino integrated development environment (IDE).

We will work with a feature-rich single board microcomputer, the Arduino Mega, as we design and develop our project. We also have a collection of peripheral devices that we will work with over the course of the quarter.

Arduino Mega (Microchip ATmega2560) features include:

- Microcontroller ATmega2560
- Clock
 - 16 MHz
 - 4KB EEPROM
 - 8KB SRAM
- Memory
 - 256 KB Flash
 - 4KB EEPROM
 - 8KB SRAM
- I/O
 - 54 Digital I/O pins
 - 16 Analog Input
 - 4 Serial I/O
 - 4 SPI
 - 2 TWI
 - Ethernet interface
- Debug Support
- Internal Peripherals
 - 13 LED
 - 15 8-bit PWM
 - 16 Channel 10-bit A/D

In this class, we will use many of these hardware features as we design and develop the various design projects; the details regarding each of the components will be given as needed.

4.0 PROJECT OBJECTIVES

- Introduce the C language, C programs, and the PC development environment:
 - Basic C/C++ preprocessor, program structure, multiple file programs, pointers, passing and returning variables by value and by reference to and from subroutines, designing, compiling, and debugging on the target platform.
- Introduce the working environment: the embedded development environment, debugging tools, the target platform, host PC, equipment, etc.
- Learn how to confirm that the target platform is functioning properly.
- Learn a bit about the Arduino ATmega2560 processor.

Every member of your team should do each of these exercises.... not just watch. Your team can submit one final report.

5.0 CODEBLOCKS AND THE C LANGUAGE – WORKING WITH C IN THE PC ENVIRONMENT

We will start this lab project by taking the first steps in the formal engineering design and development process. We will begin with the C language and the CodeBlocks development environment on a PC.

There are two options for running CodeBlocks:

1. Remote desktop
https://vannevar.ece.uw.edu/computing/faq/labs/rdp_mac_desktop_setup.html
https://vannevar.ece.uw.edu/computing/faq/labs/remote_computing.html
2. On your own PC (Windows ONLY).. see instructions below

To download and install the open source CodeBlocks development environment (IDE) on your computer, go to the CodeBlocks home page then find and run the file `codeblocks-17.12mingw-setup.exe` from the binary release download page to bring in and install the GCC compiler and GDB debugger on your computer. Note there are several options to download the software, some

of which do not have the GCC Compiler as shown in Figure 1. The compiler path may have to be set up in the program once it's installed depending on the computer. This IDE works well and is straightforward to use.



File	Date	Download from
codeblocks-17.12-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12-setup-nonadmin.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12-nosetup.zip	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw-nosetup.zip	30 Dec 2017	FossHUB or Sourceforge.net
codeblocks-17.12mingw_fortran-setup.exe	30 Dec 2017	FossHUB or Sourceforge.net

NOTE: The codeblocks-17.12-setup.exe file includes Code::Blocks with all plugins. The codeblocks-17.12-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).

NOTE: The codeblocks-17.12mingw-setup.exe file includes *additionally* the GCC/G++ compiler and GDB debugger from **TDM-GCC** (version 5.1.0, 32 bit, SJLJ). The codeblocks-17.12mingw_fortran-setup.exe file includes *additionally* to that the GFortran compiler (**TDM-GCC**).

NOTE: The codeblocks-17.12(mingw)-nosetup.zip files are provided for convenience to users that are allergic against installers. However, it will not allow to select plugins / features to install (it includes everything) and not create any menu shortcuts. For the "installation" you are on your own.

If unsure, please use codeblocks-17.12mingw-setup.exe!

Figure 1 – CodeBlocks Download Options

We will now explore the development environment. To do so, we will begin with a known good C program, then, we will work through the first steps of the development cycle with a program that we write.

5.1 Building a Program

1. Using CodeBlocks on your PC, we will create a project then add the C file *project1a-2021.c* which you can find under *Files >> Assignments >> Lab1* on the course Canvas web page.

To start, select: *File >> New Project*

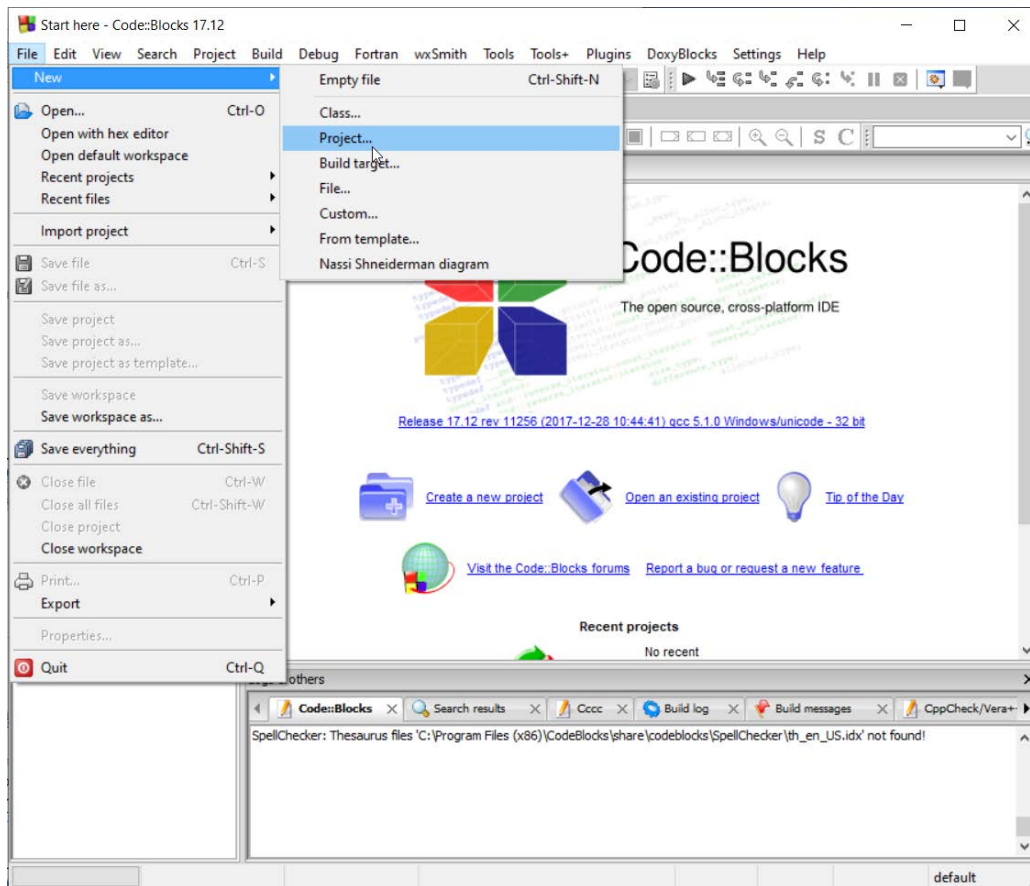


Figure 2 – CodeBlocks Splash Screen

2. In the popup window, select *Console Application* and click *Go*

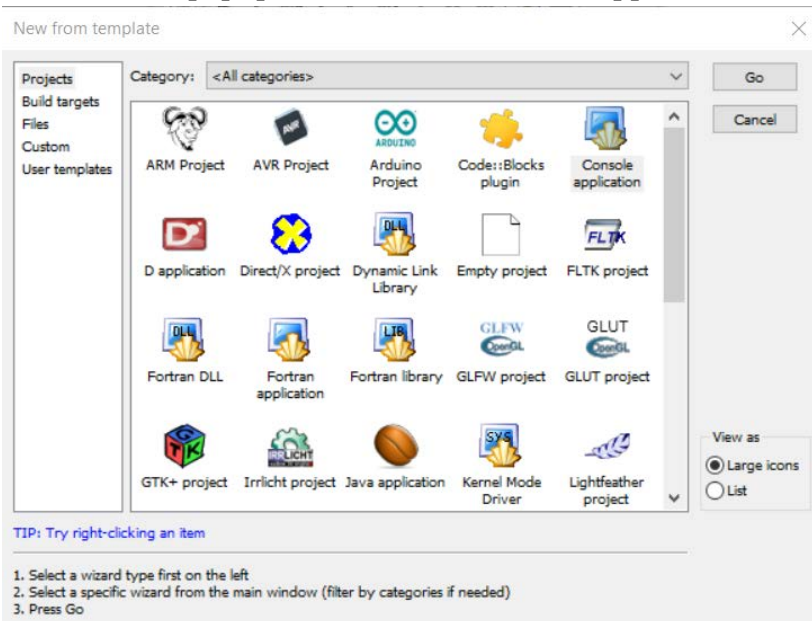


Figure 3 – CodeBlocks Project Type Selection

3. Click *Next*



Figure 4 – CodeBlocks Console Welcome Screen

4. Select *C* and click *Next*

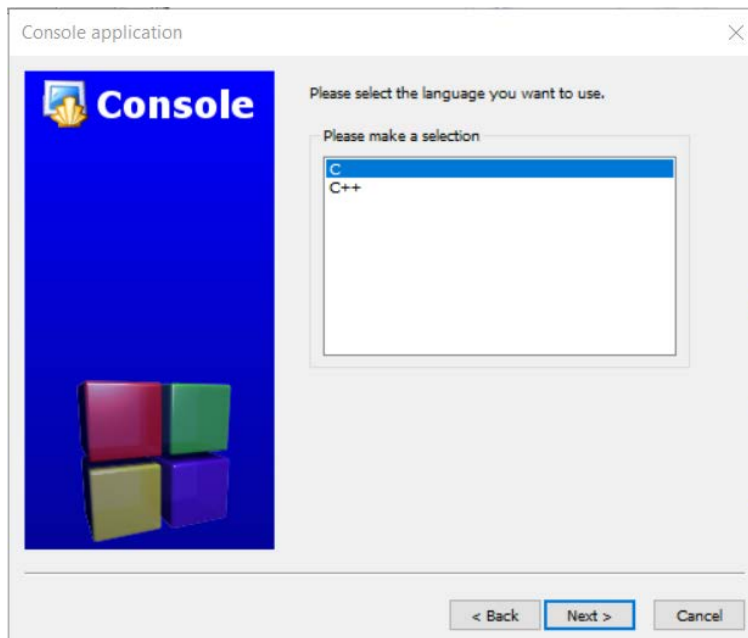


Figure 5 – CodeBlocks Implementation Language Selection

5. Give the project a *Title*, *Location*, and *Filename*. Select *Next* then *Finish*.

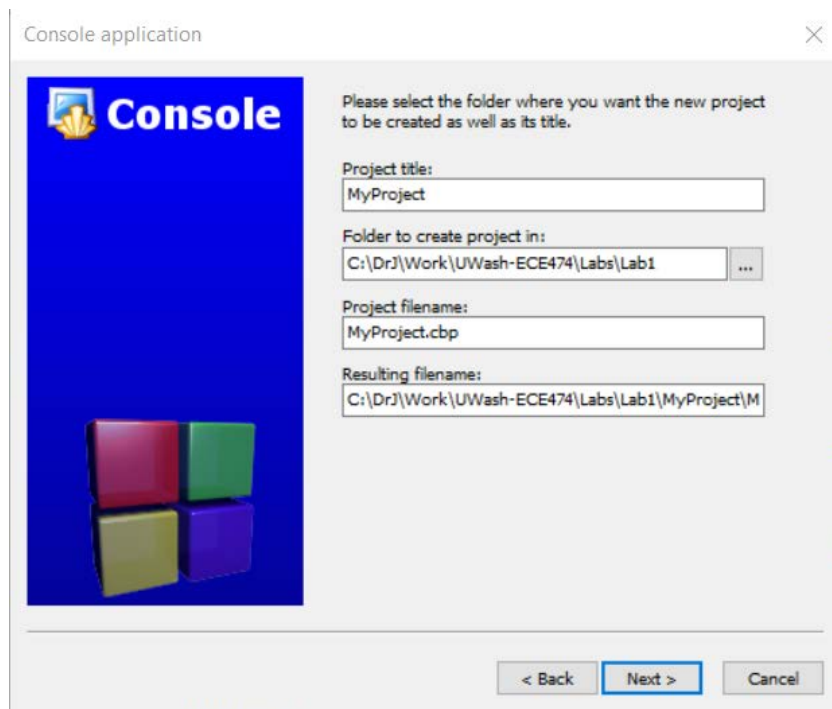


Figure 6 – CodeBlocks Project Directory Selection

6. Add the C file *project1a-2021.c* to the project directory that you just created.
7. In the left hand pane of the workspace window that pops up, right click *main.c* and select *Remove file from project*

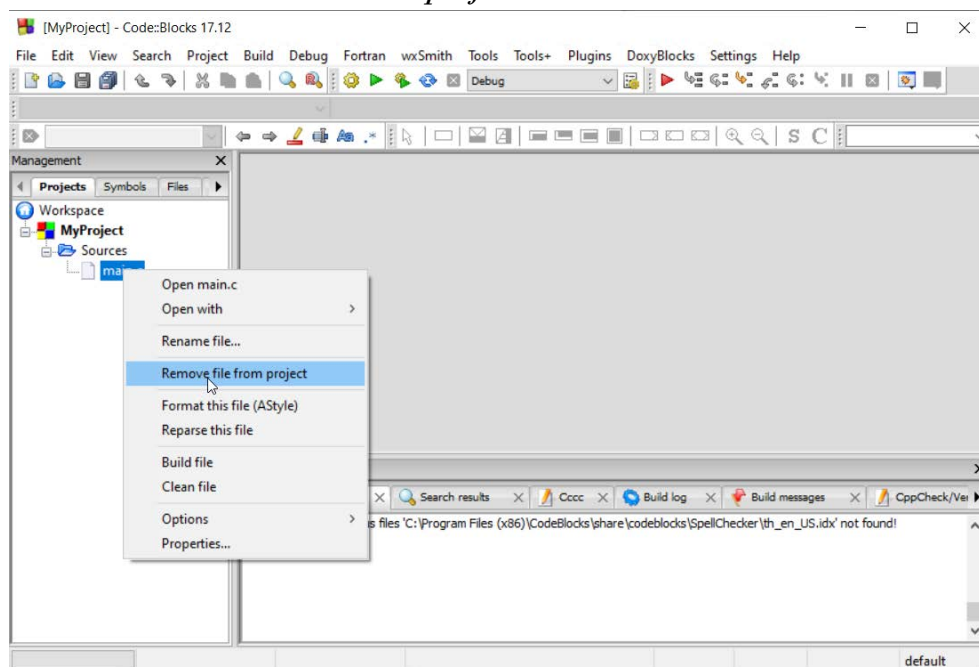


Figure 7 – CodeBlocks IDE: Removing Files from Project

8. Right click *MyProject* and select *Add Files*

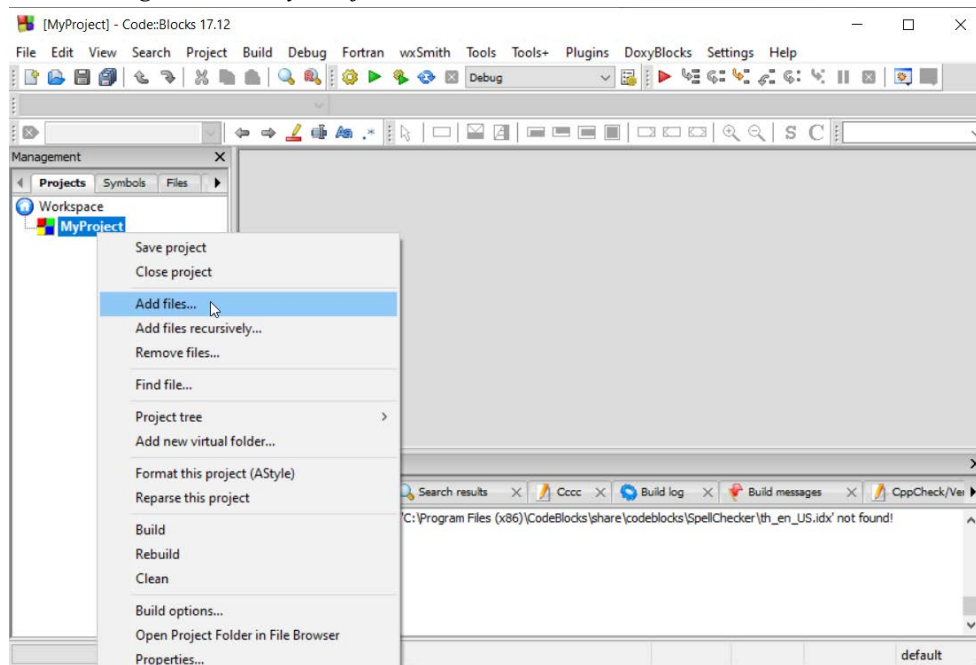


Figure 8 – CodeBlocks IDE: Adding Files to Project

9. In the popup window, browse to where you have saved *project1a-2021.c*. Select the file and select *Open* to add it to the project.

10. In the next popup window, select *Ok* for both Debug and Release build types

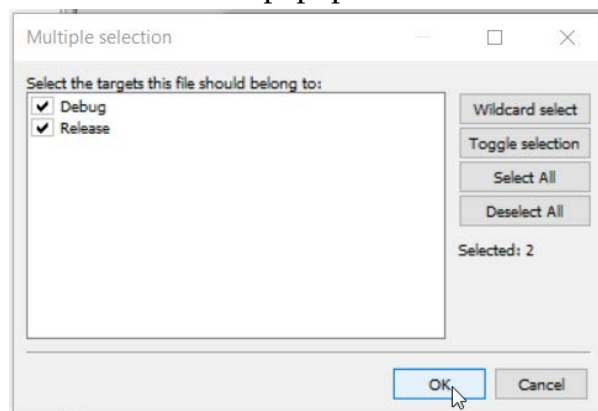


Figure 9 – CodeBlocks IDE: Selecting Build Type

11. Double click the file *project1a-2021.c* which should open in the project window

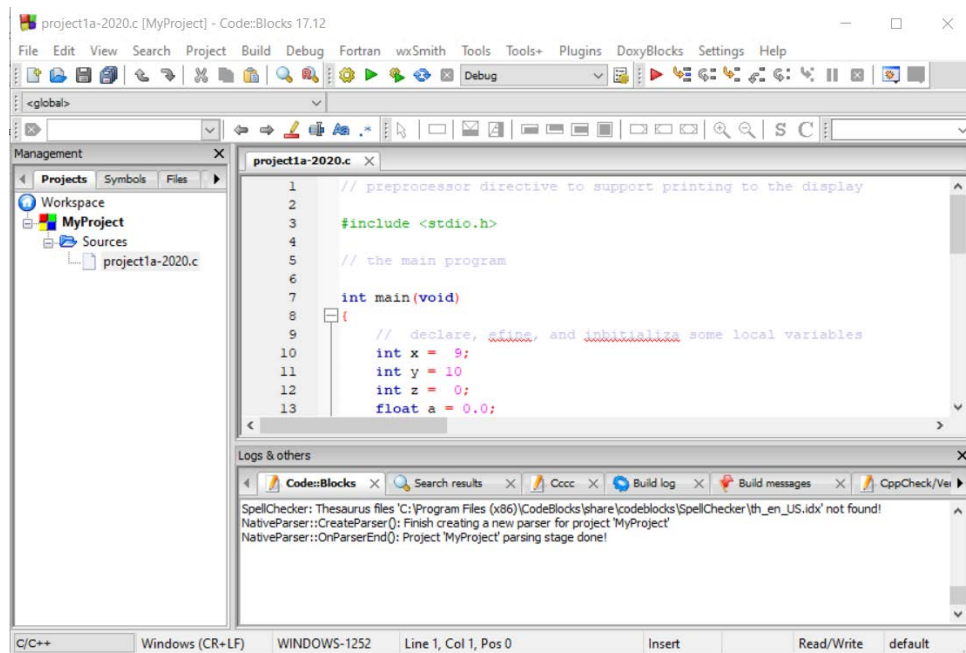


Figure 10 – CodeBlocks IDE: Opening C File for Editing

12. Now compile and execute the program: Select *Build >> Build* or press *Ctrl-F9*.

There should be no errors, if there are, then bad copying or you may need to update compiler settings.

To modify the GCC compiler path in CodeBlocks, select *Settings >> Compiler*, navigate to the *Toolchain Executables* tab and type in the installation directory, i.e. C:\Program Files (x86)\CodeBlocks\MinGW\bin.

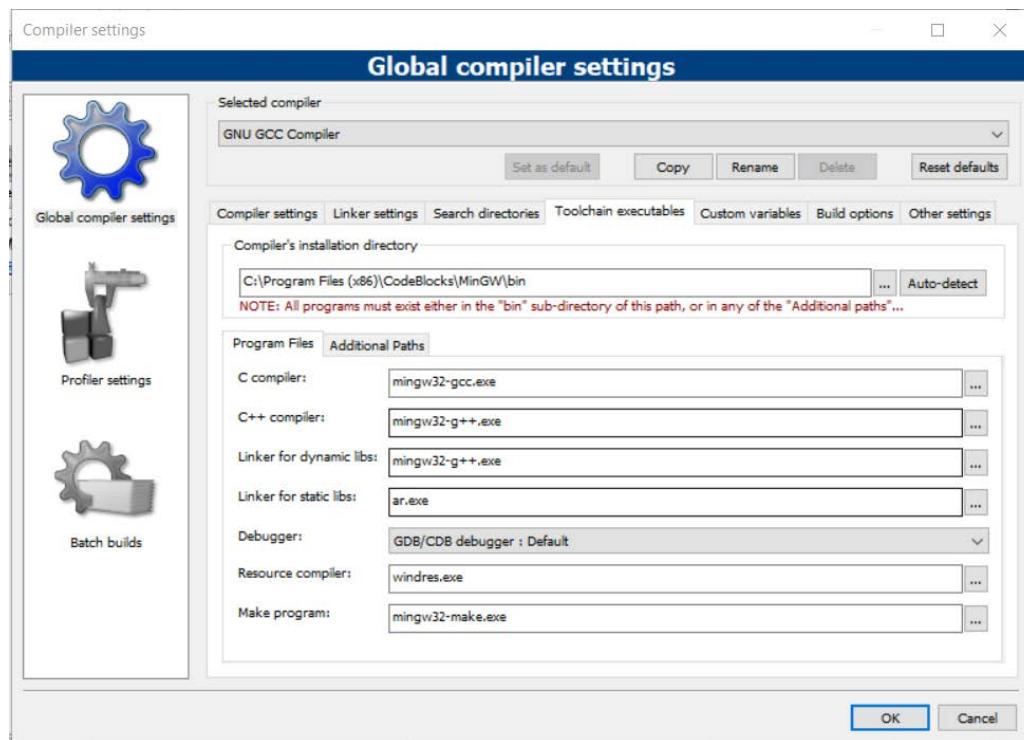


Figure 11 – CodeBlocks IDE: Setting the Compiler Installation Directory

13. To run the program, click *Build >> Run* or *Ctrl-F10*

The program should print the first two lines and halt waiting for user input. The results printed for the value of *z* may be different for your project.

Here the number 23.4 is entered and the remaining lines are then printed.

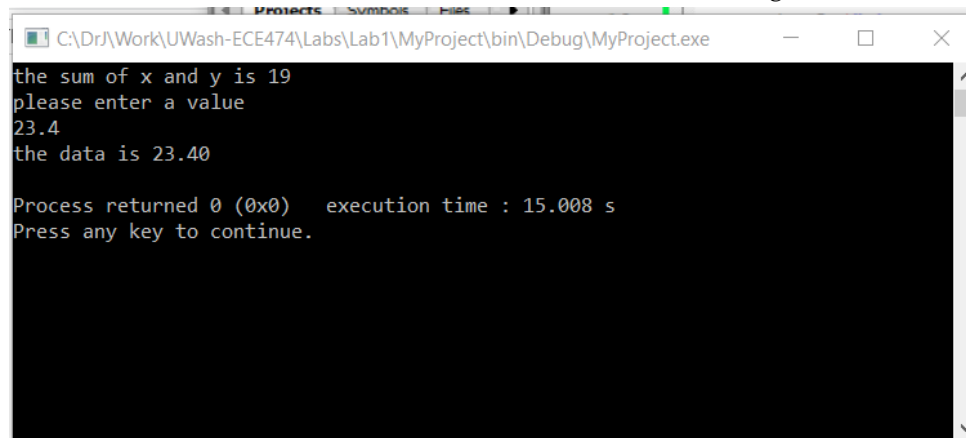


Figure 12 – CodeBlocks IDE: Console Window

5.2 Building Your Own Applications

Now that you have successfully run an existing application on the target platform, this next exercise will have you write your own program and then make a series of changes to it to gain practice in some of the techniques that we will have to use in our more complex applications.

NOTE: Create a new file for each application.

Application 1

Write a program that will display the letters: “A B C D” in the console and flash them together at approximately a one-second rate.

NOTE: There are delay functions available in standard C libraries. If you don't wish to use a library-defined delay function then you can use for loops to produce delay in a C program.

Application 2

Modify the program in Application 1 to parameterize the two *delay()* function calls in the two for loops so that they will support different user specified delays rather than the single hard coded value as they are now. Where will those values have to be specified?

Application 3

Modify the previous application (Application 2) so that each of the two respective delays are replaced by the following functions.

```
void f1Data(unsigned long delay1);  
void f2Clear(unsigned long delay2);
```

NOTE: The clear function will print " " for the letter.

Application 4

Modify the previous application program so that parameter, *delay*, is passed into the two functions in Application 3 above by reference rather than by value.

Application 5

Modify the previous application program so that the two functions in Application 3 are replaced by a single function. The function should be able to be called with the character to be displayed and the value of the delay.

Application 6

Modify the program so that the function you wrote in Application 5 is in a separate file.

Your program will now be composed of two files.

5.3 Troubleshooting

1. Find the file *project1b-2021.c* on Canvas and upload it to your project directory in the CodeBlocks environment.
2. The program will compile and execute apparently correctly, however, there is a problem with it.
3. Identify what the problem is and indicate how you found it. Correct the problem and demonstrate that you have, indeed, fixed the problem.
4. Find the file *project1c-2021.c* on Canvas and upload it to your project directory in the CodeBlocks environment. The program will compile, however, it has a problem during execution. Identify what the problem is and indicate how you found it. Correct the problem and demonstrate that you have, indeed, fixed the problem.

6.0 LEARNING THE ATMEGA ENVIRONMENT AND TOOLS

As our next step, we will now move to the ATMEGA 2560, explore the development tools, build, and run an existing simple program on the board. The pinout for the ATmega board is given in Figure 13. The USB ports used to connect to a computer are on the upper left-hand side of the board.



Figure 13 – Arduino Mega

The source code for several of the programs we will be using is in the same directory as the project assignment; the Arduino files will have a “.ino” suffix.

6.1 Working with the Development Environments and Target Platforms: C for the ATmega Environment

When we move to the Arduino ATmega 2860 microcontroller, we will work with the Arduino development environment (IDE).

To download and install the open source development environment IDE, go to the Arduino home page:

<https://www.arduino.cc/en/Main/Software>

Then find and select the appropriate installer for your environment and run the file. This will install the IDE and the GCC compiler. This IDE works well and is straightforward to use.

Our next step is to connect the ATmega to the PC: Run the Arduino IDE. You should now see the IDE workspace, see Figure 14.



Figure 14 – Arduino IDE Screen

The next step is to select the target by clicking *Tools >> Board >> Arduino/Genuino Mega or Mega 2560*, see Figure 15.

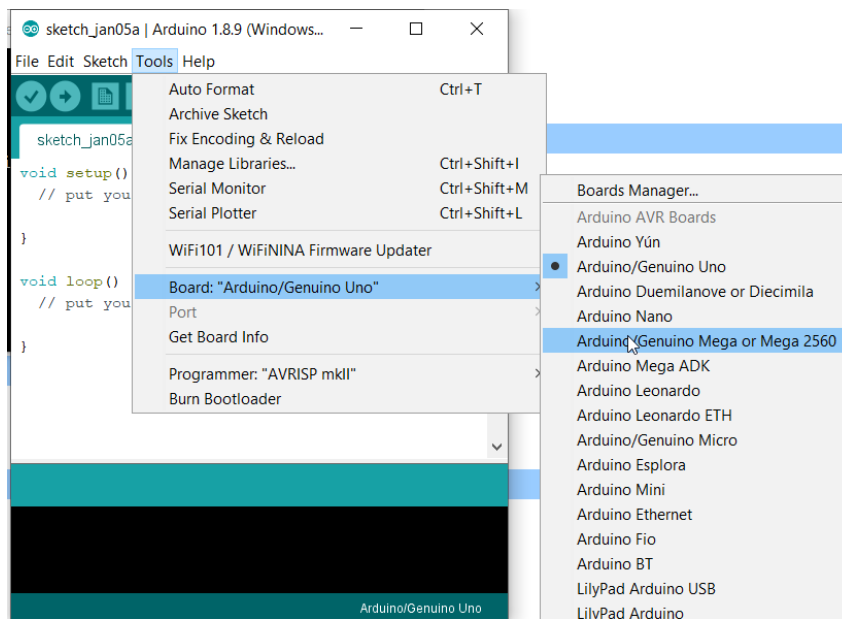


Figure 15 – Arduino IDE: Selecting the Target

When you connect your Arduino board to the PC, the Arduino IDE creates a virtual COM port (communications port). This is a connection that functions like a serial port on your PC and it is how your PC talks with the ATmega and vice versa.

Now connect to the target by selecting *Tools >> Port >> COM3 (Arduino/Genuino Mega or Mega 2560)*, see Figure 16.

NOTE: Your COM port number will probably be different. In any case, select the Mega, not COM1 or COM2 if they appear.

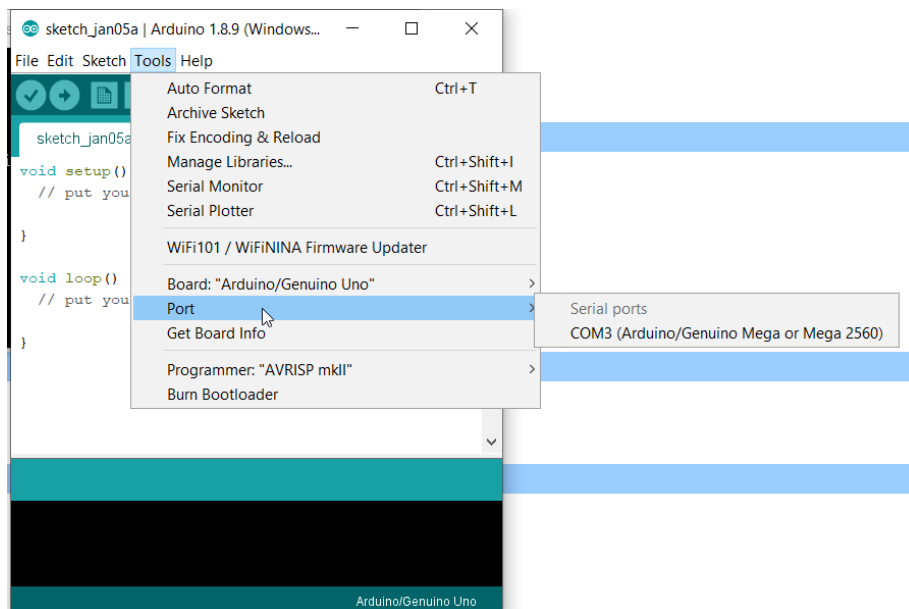


Figure 16 – Arduino IDE: Selecting the COM Port & Connecting to the Target

6.2 First Steps: Loading Code to the ATmega

Now, the ultimate task for all electrical engineers: Getting an LED to flash...this will be our first Arduino program...

STEP 1: Load Program

Select File >> Examples >> 01.Basics >> Blink

A new window will be created and opened. The window that opened contains all of the code for the *Blink* program. In Arduino jargon, such a program is called a *sketch*.

Start by identifying the major functional blocks...you should be able to identify 3. The program is well annotated, so read through and understand what each line of code does.


Don't get too excited, though. This is an example program...all your programming assignments will not be this easy nor will they be hidden here.

An important thing to notice is that, unlike the traditional C program, an Arduino sketch does not have a top-level *main()* function. Rather, it has a top-level *loop()* function (delimited by the two {} that we call curly brackets....ahhhh, that's because they are curly).

Once the *loop()* function is entered, the program will run forever. Well, not if you turn power off, of course, or jump up and down on your Arduino board. It's pretty cool, but, it ain't that cool.


Our next step is to compile the program into a form that the ATmega will understand - we're going to compile our C code.

STEP 2: Verify/Compile Code

Select  in the second toolbar to *Verify/Compile* your code. This command can also be reached under *Sketch >> Verify/Compile*.

If there is an error, the bottom pane will contain a message such as shown. If there are errors, you will have to fix them before proceeding. If there aren't, then, we are ready for the next step.

STEP 3: Uploading code to the microcontroller

Select  in the second toolbar to *Upload* code to the microcontroller. This command can also be reached under *Sketch >> Upload*.

When the upload is complete, the program will start running on the target processor – the ATmega 2560. In this case, an LED on the board will start blinking. Wahoo!!!! An engineer's initiation and success...cool.

6.3 Building, Editing, and Running a Program

Let's now modify an existing project.

1. Install the TFT display onto your ATmega board (see the Elegoo user manual in Canvas under *Pages >> Labs >> Software & Datasheets >> Elegoo 2.8 Inch Touch Screen User Manual V1.00.2019.03.19.zip*)
2. Install the libraries *Elegoo_GFX.zip*, *Elegoo_TFTLCD.zip*, *TouchScreen.zip* under *Sketch >> Add .ZIP Library* as shown in Figure 17.

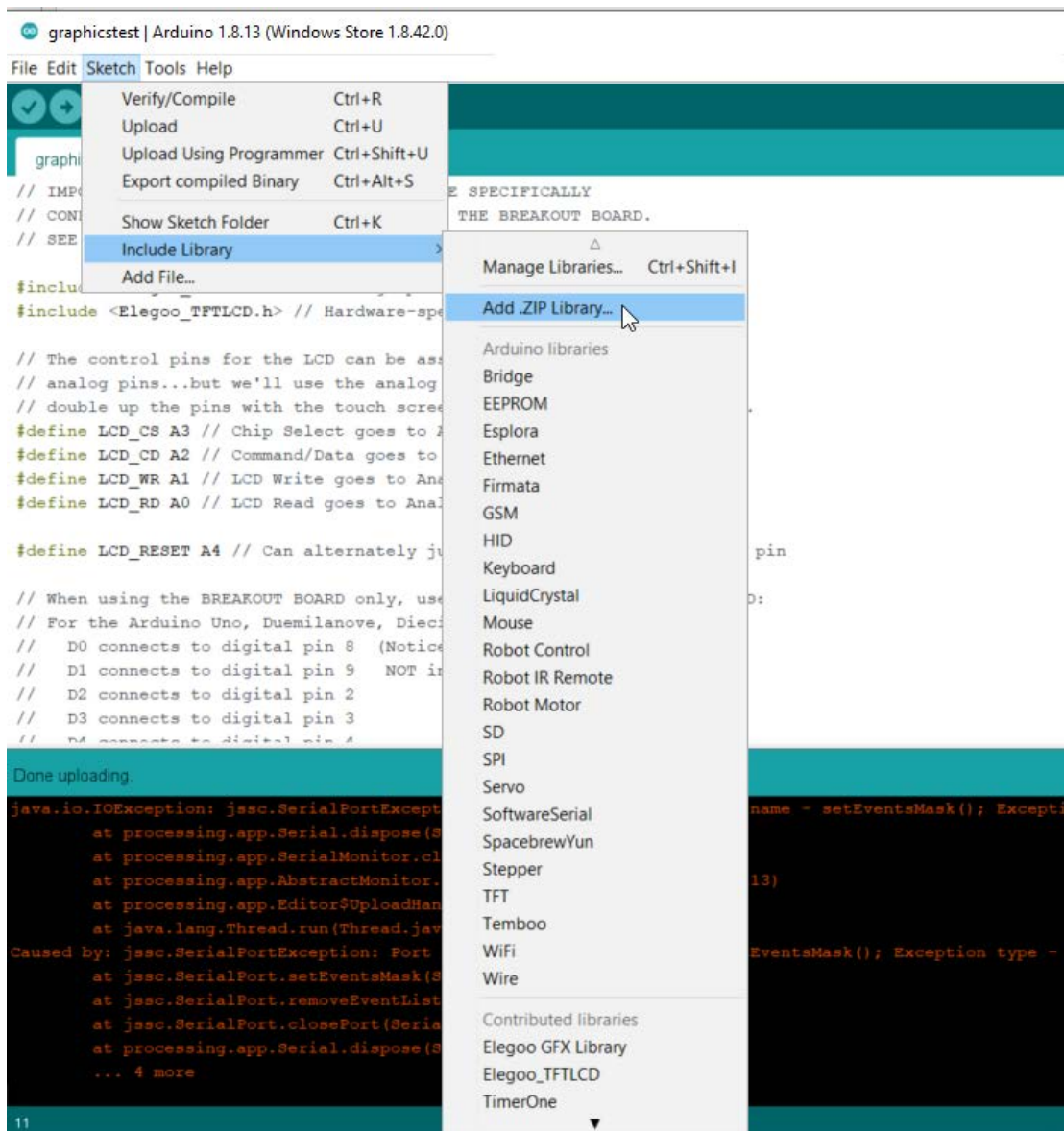


Figure 17 – Installing a Library in the Arduino IDE

3. Find the file *project1d-2021.ino* in the *Assignments >> Lab1* Files folder on Canvas. Save and open the file on the ATmega.
4. Annotate each line of the main program to identify its purpose in the program and what it does (these really are different).
5. When you are finished, save the program.
6. Compile your program and download to the ATmega.
7. Execute your program. There should be no errors – if there are, then bad copying.

The program should count in decimal according to the following pattern. Note that the following is actually 10 iterations through the program and the program's output will appear on only one line on the console display....at time t0, the value 9 is displayed....at time t1, the values 9 8 are displayed and so forth.

```
The value of i is:  
t0: 9  
t1: 9 8  
t2: 9 8 7  
t3: 9 8 7 6  
t4: 9 8 7 6 5  
t5: 9 8 7 6 5 4  
t6: 9 8 7 6 5 4 3  
t7: 9 8 7 6 5 4 3 2  
t8: 9 8 7 6 5 4 3 2 1  
t9: 9 8 7 6 5 4 3 2 1 0
```

8. Port Application 6 that you developed for CodeBlocks to the ATmega 2560. User input can come from the Arduino Serial Port or the Touch Screen.

7.0 DELIVERABLES

A project report containing:

1. The annotated source code for all CodeBlocks and the ATmega Processor applications.
2. Representative screen shots showing the results of executing the applications on the PC screen. Representative screen shots means that you need to visually document that you got the programs to compile and run on your system and that your design was able to meet the project requirements and specifications.

Please refer to the **Lab Report Rubric** for formatting. The project report will be graded according to the rubric.