# Getting Started with EE4524/ED5502 Project 2, Spring 2020

Version 1.0: Initial version (minimum proof reading). V1.1 Added close Serial Program before programming.

## General Ideas

This document describes how I would go about getting started with Project 2. It’s just a suggestion, other people will go about thing differently.

A good way of doing software development is to design top-down and implement bottom up. The idea is that you can start from some working code and build on top of that. This does not mean you should just hack away at the problem: try to plan what you will do, outline the functions and data that you’ll be using and get an idea of the overall structure of your program. When you’ve done that then I recommend building up your program a bit at a time, rather than trying to get it going in one big lump.

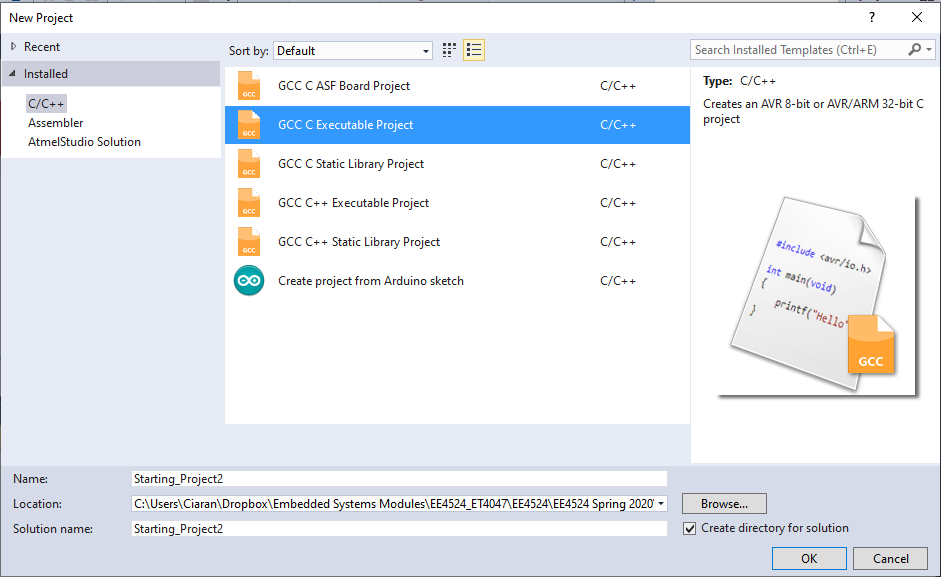
In this project, the structure of the program is going to be built around polling the serial port for a new character received. Since we are using just one character there is no advantage here to using interrupts for character received. But if we were parsing a string, we might use the receive interrupt to place characters in a buffer, and only flag the polling loop when a string was received. But for a single character this doesn’t help. You could go into a low power mode and use receive interrupts to wake up, but I haven’t asked you to do this.

## Specifics

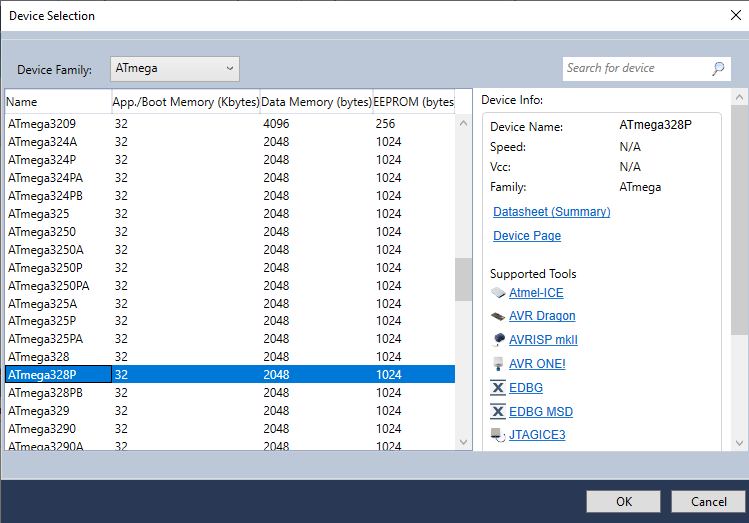
So, I design my software with Initialisation code for the on-chip peripherals I’ll be using. To begin with I just put place holders (stubs) for these functions. I’ll fill them in later. So, my program will have some initialisation functions and some variables initialisation followed by a global interrupt enable and an infinite while loop. Yours will probably be similar.

**One very important point:** if you initialise an interrupt and enable interrupts you **MUST** have an ISR (even if it’s empty) for that interrupt source.

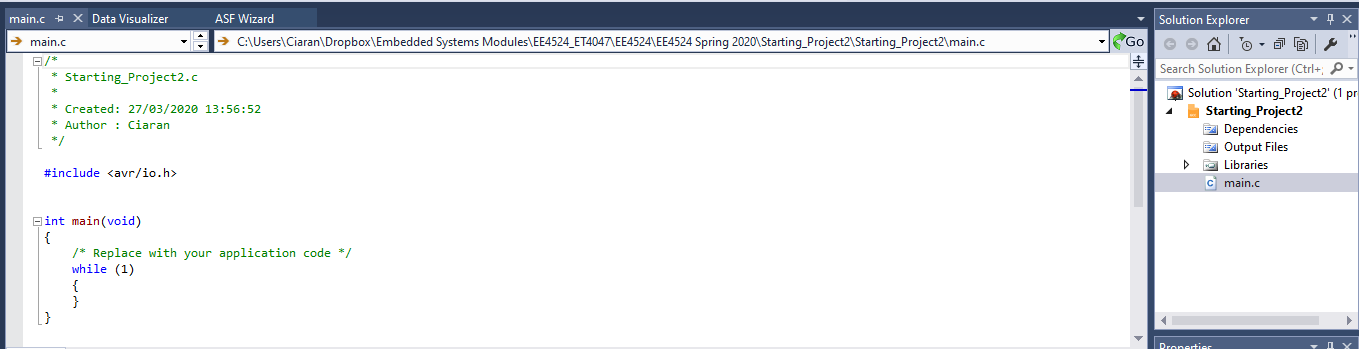
I set up a new project as we have done in the past. My project is called Starting\_Project2.



I set ATMega328P as the MCU and continue.

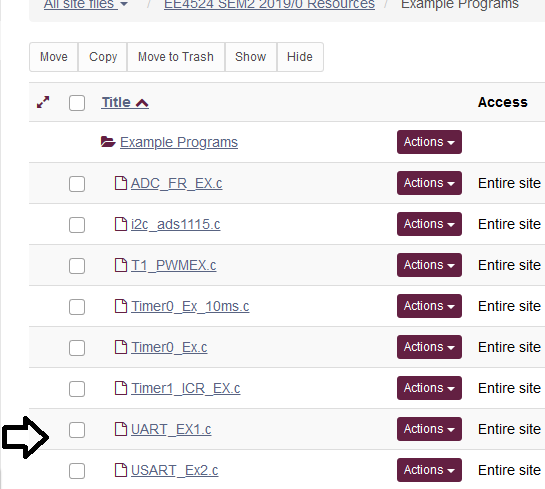


We get the usual program template from Studio 7.

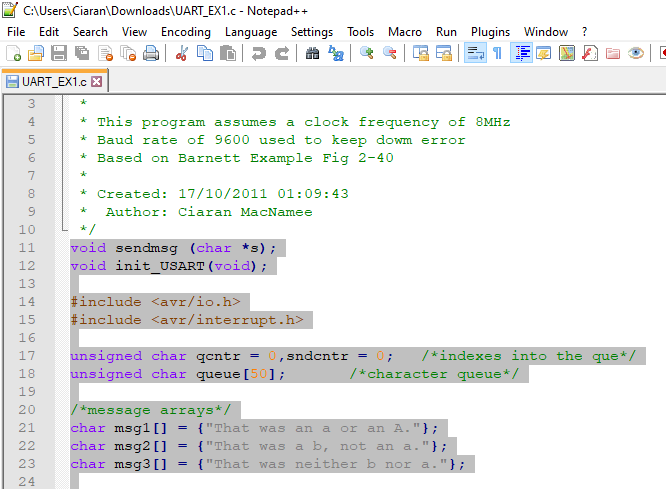


Now let’s be more specific. Project 2 uses the Serial Port. I find that if I get this part of the project up and running then I can always output some information to the serial port on my PC to help me debug.

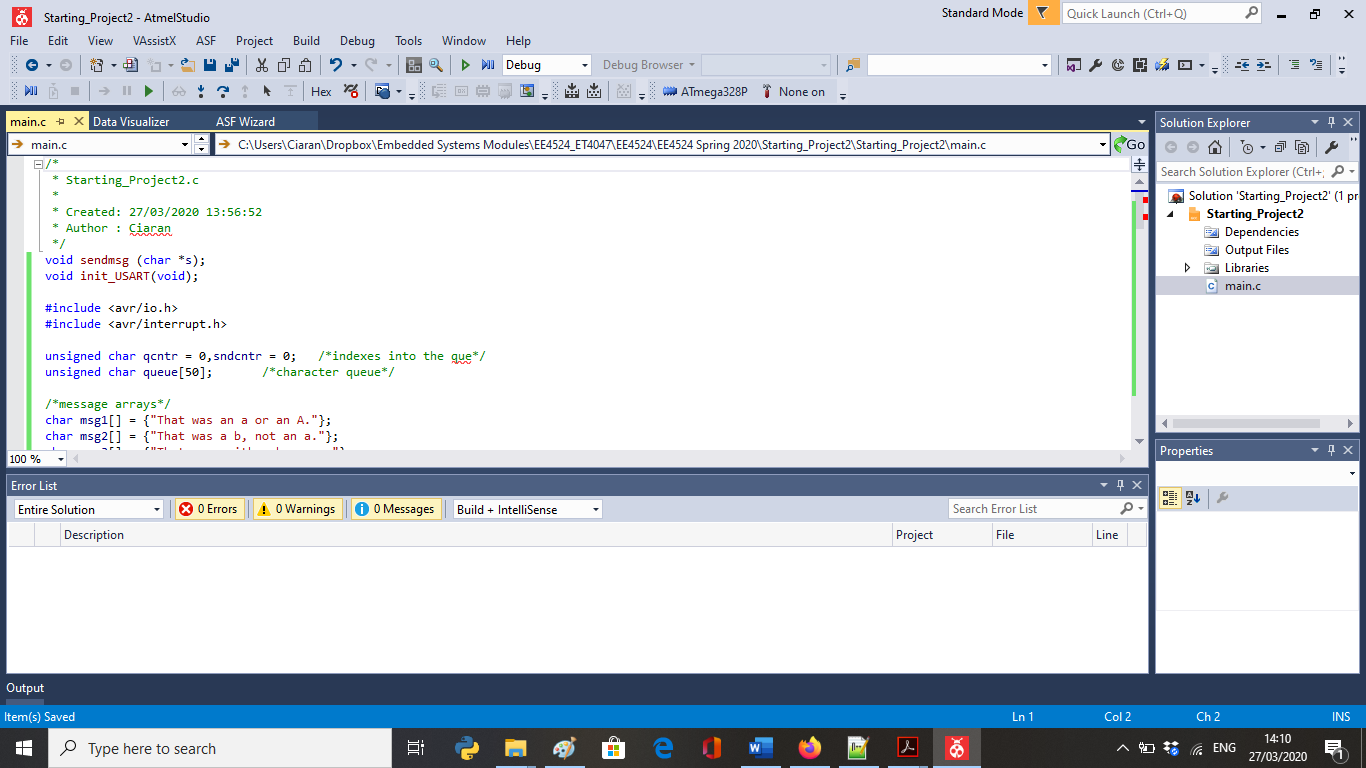
The Sulis Example programs have a program called USARTEX1.c. I’m going to copy the text from that program into my new project code.



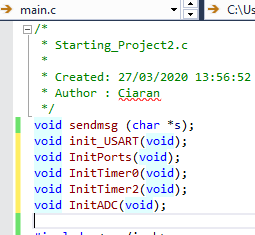
Download the file and open it in a text editor (I use Notepad++) then Select All (Ctrl-A) and Copy (Ctrl-C) to copy the text. (Actually, I didn’t copy the comment block, but used everything else). Only part of the file is shown.



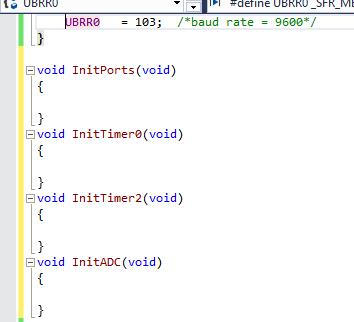
Copy this into AVR Studio 7 main.c



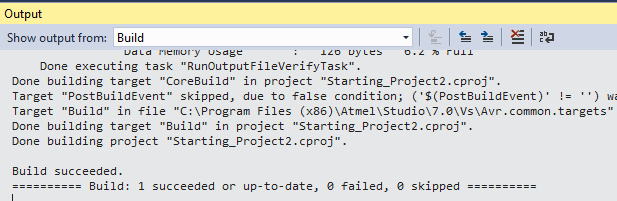
Now let’s modify this with some empty initialisation functions (to be filled in later): InitPorts(), InitTimer0(), InitTimer2(), InitADC(). InitUSART() is there already.



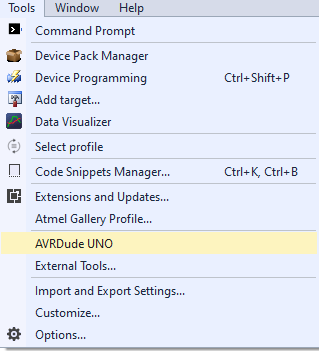
Add in the empty functions. I put them after init\_USART():



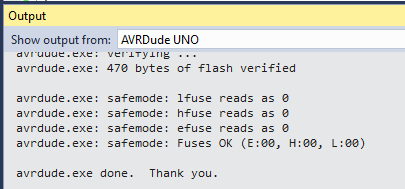
Now let’s build what we have: (F7 or use the Menu). If all has gone well, we’ll see:



Now connect my Arduino UNO R3 + Shield (even if we’re not using it yet). And use the AvrDude UNO Tool we setup for Project 1 (You may have used a different name, but the idea is the same).

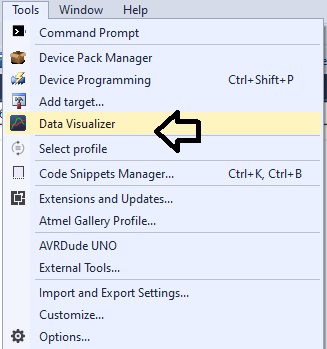


All going well I see in the Output Window:



Now let’s use this program; can we get data back from the USART serial port?

Go to Tools->Data Visualizer



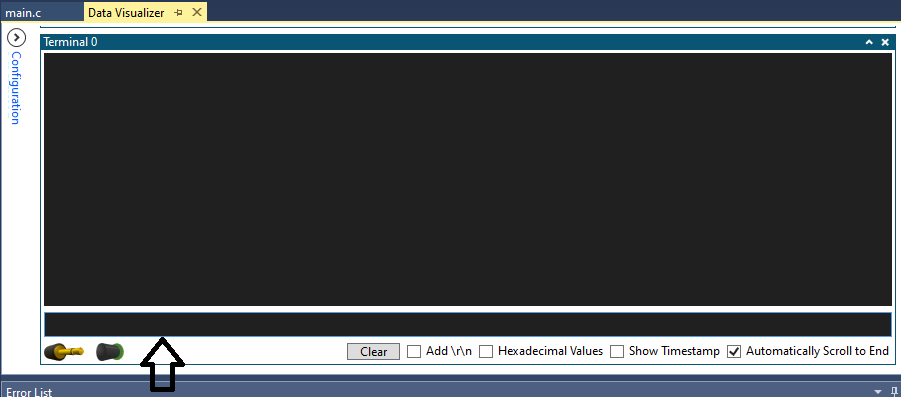
We see the following, and we want to click drop down option.



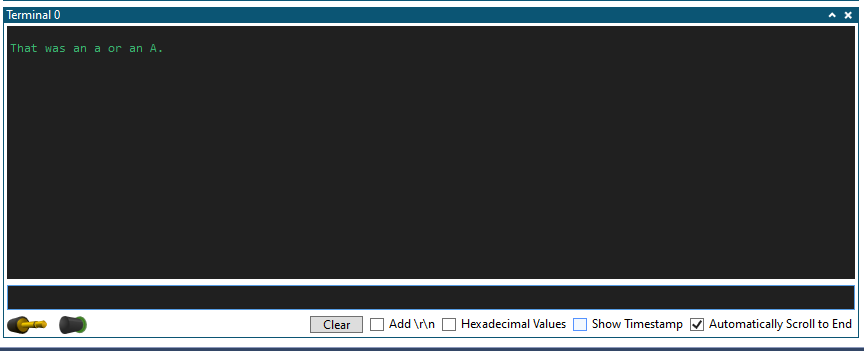
We get:



I recommend Unchecking the DTR box. **NB** Make sure Autodetect protocols is **NOT** checked. Press Connect. This opens up a Terminal.



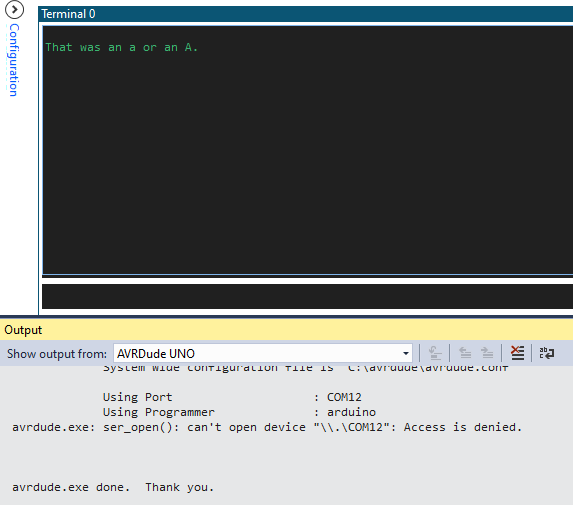
The bottom bar of the terminal is where you can enter characters. I entered ‘a’:



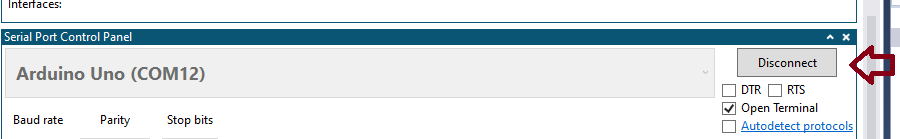
Now at least I have a starting point to build on.

## Note: Avrdude and the Serial Port

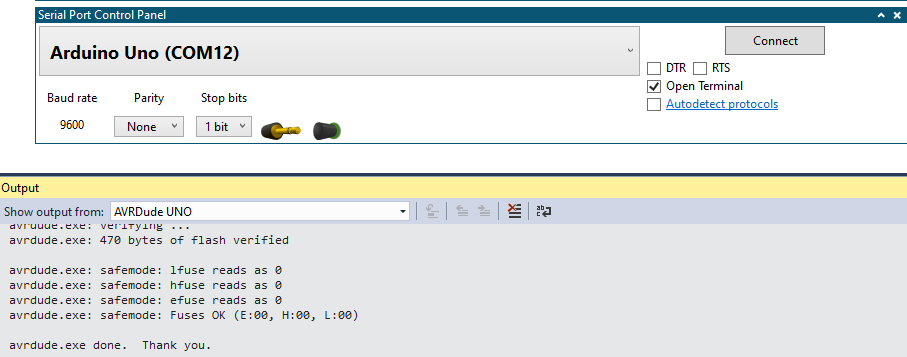
Both Avrdude and the Serial Port Terminal opened in the Data Visualizer use the Virtual Com Port (in my case COM12). If you want to load (Flash) a new version of your program to the UNO, you’ll have to Disconnect the Serial Port first. Otherwise you’ll get error messages.



Press Disconnect to release COM12 so that Avrdude can use it.



Now it can work:



## Note on Formatted Printing

The definition of message strings in USART\_EX1.c is inflexible and hard to modify. We would prefer to use formatted printing as in printf. But this requires polling the transmitted characters. So, we can use a close relative of printf: sprintf to make formatted printing easier. I’ll show this in a separate document.