ECE3411 – Fall 2015

Lab 2b.

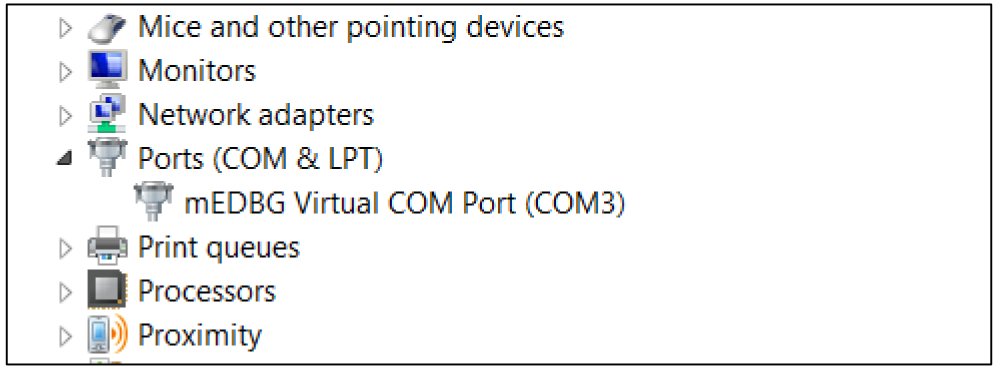
UART: Universal Asynchronous Receiver & Transmitter

Marten van Dijk, Syed Kamran Haider
Department of Electrical & Computer Engineering
University of Connecticut
Email: {vandijk, syed.haider}@engr.uconn.edu

UART Setup: COM Port Identification (1)

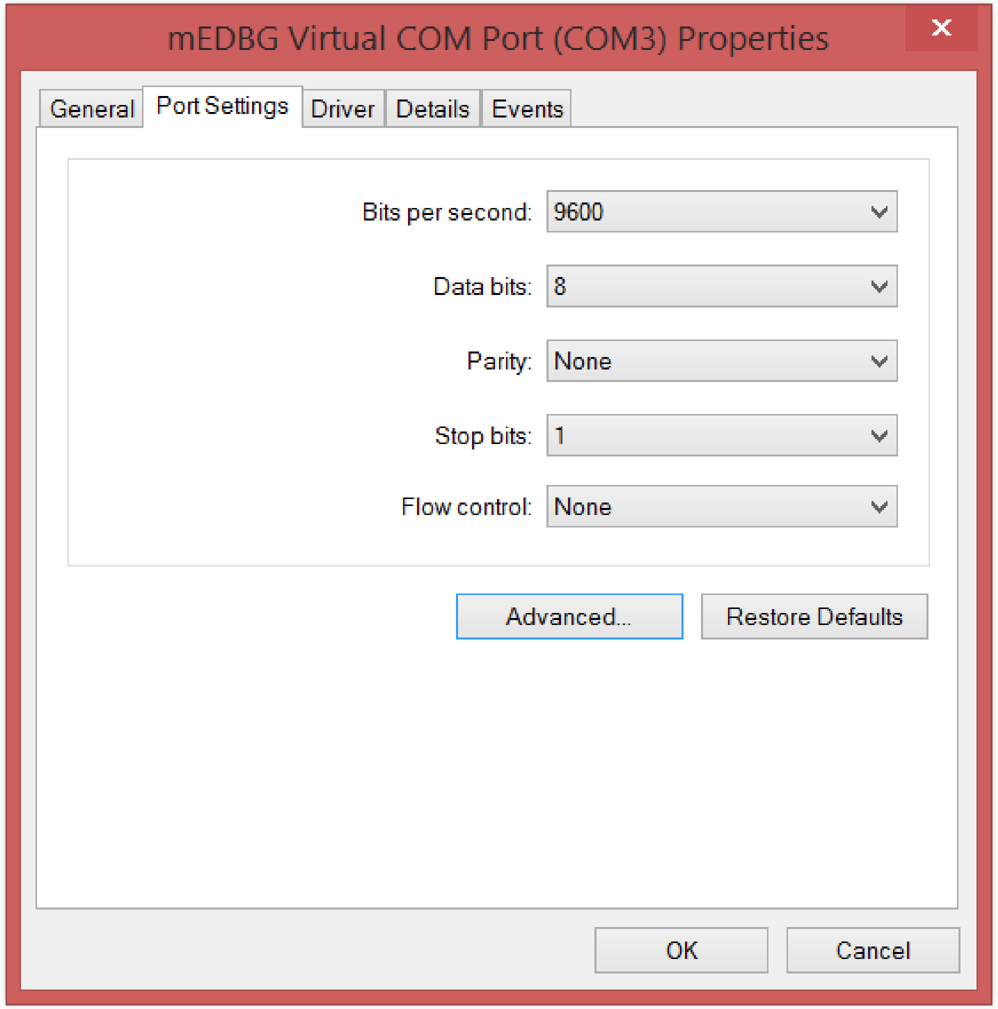
In order to setup UART communication between the Xplained mini and your PC, we first need to identify and setup the COM port used by Xplained Mini board

- Connect the Xplained Mini board to your computer via USB cable
- Go to: Control Panel → Device Manager
- Expand the Ports (COM & LPT) section as shown in the figure below.
- Note down the Port number shown against mEDBG Virtual COM Port, i.e. COM3 in the figure below.



UART Setup: COM Port Identification (1)

- Double Click to open the Properties window of mEDBG Virtual COM Port.
- Make sure the Port Settings are the same as shown below.
- If necessary, the COM Port number can be changed under Advanced tab. However, generally the default COM Port number works just fine.

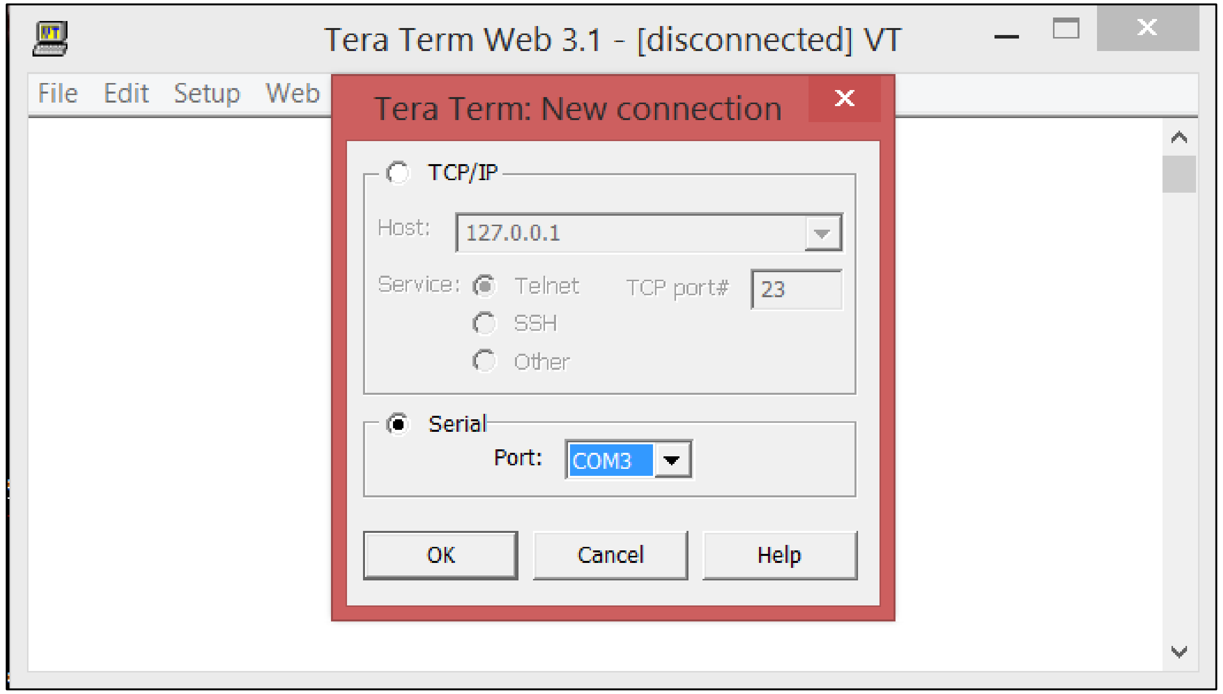

mEDBG Virtual COM Port (COM3) Properties

Bits per second: 9600
Data bits: 8
Parity: None
Stop bits: 1
Flow control: None

UART Setup: TeraTerm Pro

We will use TeraTerm Pro terminal to send/receive data to Xplained Mini over UART

1. Download ttpro313.zip file posted under Resources on Piazza
2. Unzip the file and run the application ttermpro.exe
3. In the New Connection window, select Serial and select your mEDBG COM Port number, e.g. COM3 (refer to the previous slide) and click OK.



Tera Term Web 3.1 - [disconnected] VT

Tera Term: New connection

( ) TCP/IP
Host: 127.0.0.1
Service: (●) Telnet   TCP port# 23
( ) SSH
( ) Other

(●) Serial
Port: COM3

UART Setup: Using uart.h Library

- In order to facilitate you, we provide a library file “uart.c” which defines some useful basic UART functions.
  - “uart.h” and “uart.c” can be downloaded from Piazza under Resources.

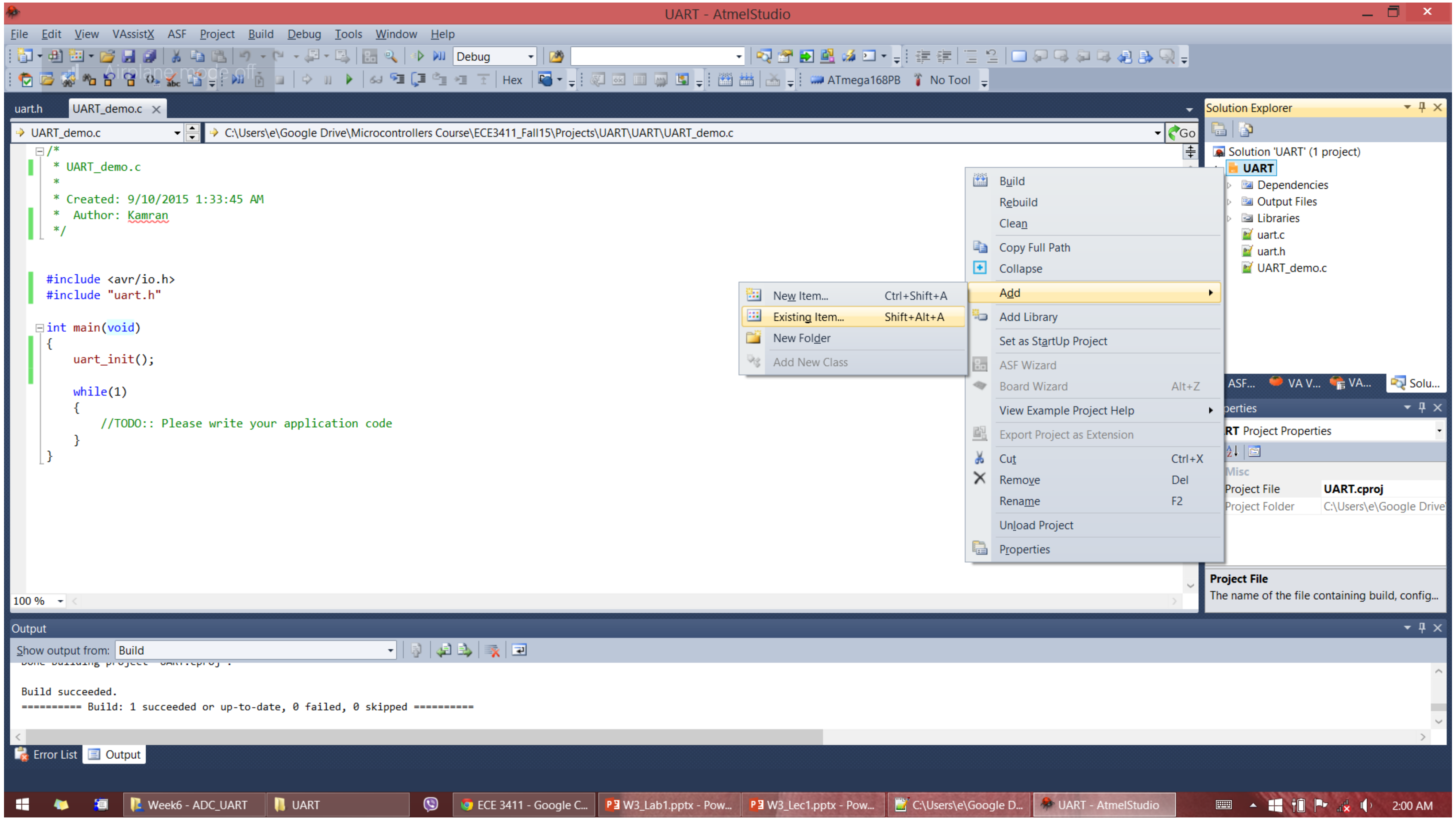
- The corresponding prototypes of the functions are declared in “uart.h” file which comes along with “uart.c” file.

- In order to use the function provided by “uart.c”, you need to:
  1. Add “uart.c” and “uart.h” files in your Atmel Studio project source files
  2. Include “uart.h” as a header file in your code, i.e. `#include "uart.h"`

Adding Header and C Files to a Project

- Often, it is more convenient to include files within your project that contain definitions and functions that you will use frequently.
- This reduces the length of your main c file and eliminates the need for copying and pasting functions you've already written in the past.
- Suppose we want to add "uart.c" and "uart.h" to a project:
  1. Create a new project in Atmel Studio.
  2. Copy the files "uart.c" and "uart.h" into the project directory.
  3. In the 'Solution Explorer' window, right click on the project's name → Add → Existing Item …
  4. Select "uart.c" and "uart.h" and click "Add".
  5. Don't forget to declare/include the header file in you code by calling #include "uart.h"
- See the next few slides for illustration

Adding Header and C Files to a Project



UART - AtmelStudio

File Edit View VAssistX ASF Project Build Debug Tools Window Help

uart.h UART_demo.c

C:\Users\e\Google Drive\Microcontrollers Course\ECE3411_Fall15\Projects\UART\UART\UART_demo.c

```
/*
 * UART_demo.c
 *
 * Created: 9/10/2015 1:33:45 AM
 *  Author: Kamran
 */


#include <avr/io.h>
#include "uart.h"

int main(void)
{
    uart_init();

    while(1)
    {
        //TODO:: Please write your application code
    }
}
```

Project File UART.cproj

Project Folder C:\Users\e\Google Drive

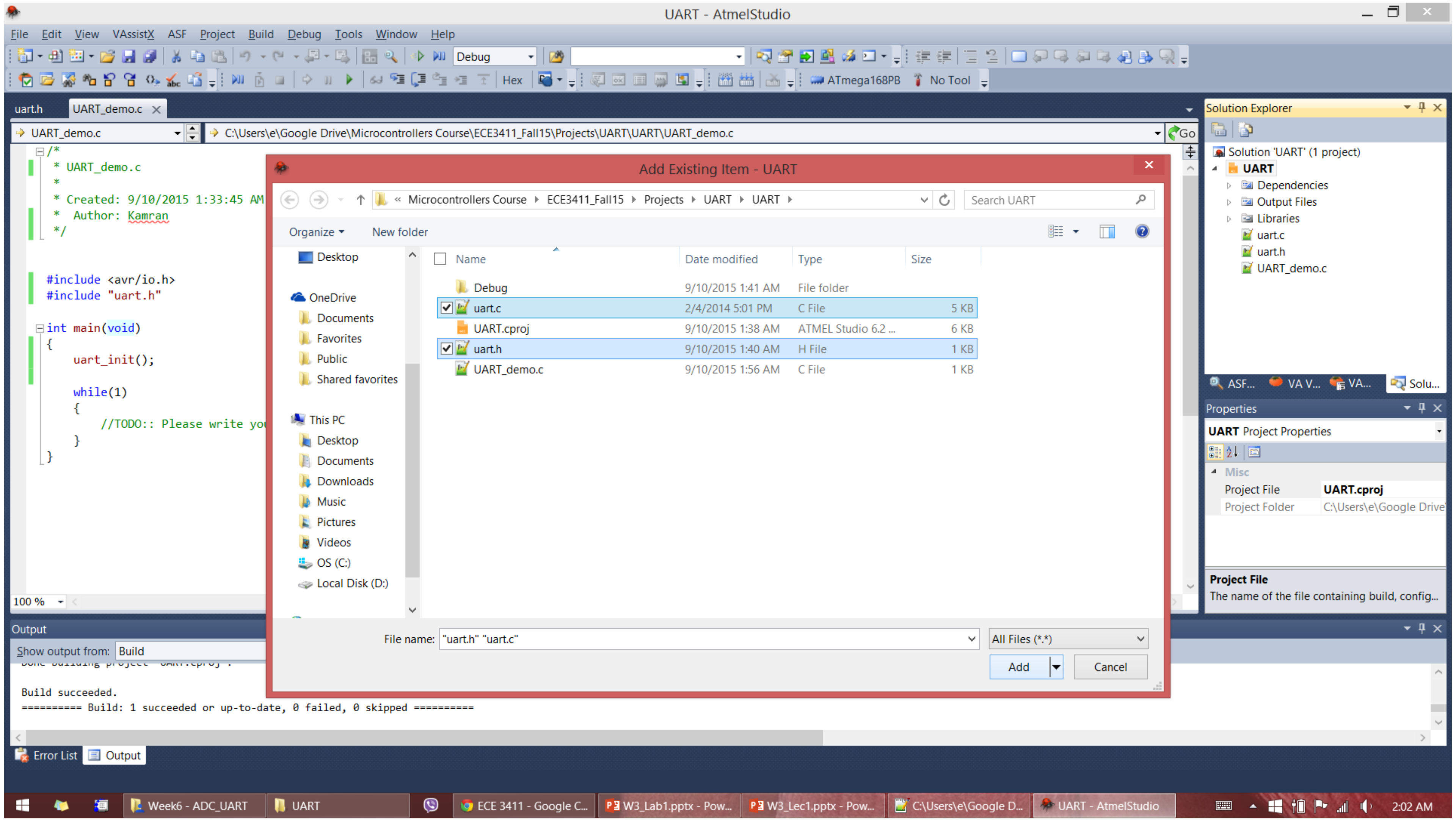
Project File
The name of the file containing build, config...

Output

Show output from: Build

```
Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Adding Header and C Files to a Project



C:\Users\e\Google Drive\Microcontrollers Course\ECE3411_Fall15\Projects\UART\UART\UART_demo.c

```
/*
 * UART_demo.c
 *
 * Created: 9/10/2015 1:33:45 AM
 *  Author: Kamran
 */


#include <avr/io.h>
#include "uart.h"

int main(void)
{
    uart_init();

    while(1)
    {
        //TODO:: Please write yo
    }
}
```

« Microcontrollers Course ▸ ECE3411_Fall15 ▸ Projects ▸ UART ▸ UART ▸

Name	Date modified	Type	Size
Debug	9/10/2015 1:41 AM	File folder	
☑ uart.c	2/4/2014 5:01 PM	C File	5 KB
UART.cproj	9/10/2015 1:38 AM	ATMEL Studio 6.2 ...	6 KB
☑ uart.h	9/10/2015 1:40 AM	H File	1 KB
UART_demo.c	9/10/2015 1:56 AM	C File	1 KB

Desktop, OneDrive, Documents, Favorites, Public, Shared favorites, This PC, Desktop, Documents, Downloads, Music, Pictures, Videos, OS (C:), Local Disk (D:)

File name: "uart.h" "uart.c"   All Files (*.*)

Solution 'UART' (1 project)
- UART
  - Dependencies
  - Output Files
  - Libraries
  - uart.c
  - uart.h
  - UART_demo.c

UART Project Properties
Misc
Project File	UART.cproj
Project Folder	C:\Users\e\Google Drive

Project File
The name of the file containing build, config...

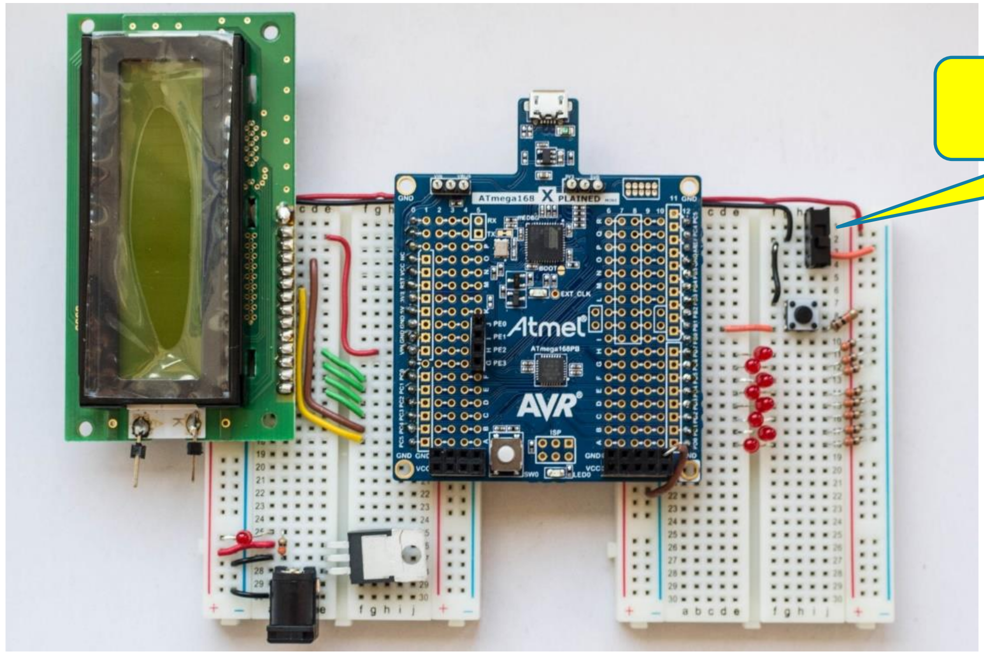
Show output from: Build

```
Build succeeded.
========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========
```

Week6 - ADC_UART | UART | ECE 3411 - Google C... | W3_Lab1.pptx - Pow... | W3_Lec1.pptx - Pow... | C:\Users\e\Google D... | UART - AtmelStudio | 2:02 AM

Running a UART Test Program (1)

- Connect the board to your computer and create a new project in Atmel Studio
- Include the “uart.h” and “uart.c” files in your project as described earlier
- Notice that PD0 and PD1 serve as UART RXD and TXD pins
  - **Hence these pins should not be used for any other purpose when using UART**
  - **Therefore make sure to turn off the switch to disconnect LEDs from Ground**



Turn Off (switch up)

Running a UART Test Program (2)

- Open TeraTerm Pro terminal and create a connection as described earlier
- Copy the test program given on the next slide, compile it and run.
- You should see a “Hello!” message on TeraTerm window
  - Notice that you need to create the TeraTerm connection before running the program to see this greeting message.
- The test program simply sends back the string which it receives from the terminal
- Try writing some small strings, and you should see it printed out on the terminal once you hit Enter key.

UART Test Program

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include "uart.h"

// File stream for UART. Used for Transmission to demonstrate the fprintf function.
FILE uart_str = FDEV_SETUP_STREAM(uart_putchar, uart_getchar, _FDEV_SETUP_RW);

char rec[50];                  // Declare a character buffer
int main(void)
{
            uart_init();                    // Initialize UART
            stdout = stdin = stderr = &uart_str; // Set File outputs to point to UART stream
            fprintf(stdout, "Hello! \n");

            while(1){
                        fscanf(stdin, "%s", rec);
                        printf("Received: \n");
                        fprintf(stdout, "%s \n", rec);

   }
}
```

Task 1: Blinking a single LED

- Blink a single LED at two different rates based on a push switch.
  - When the switch is not pressed, LED should blink at 2Hz frequency.
  - As long as the switch is pressed, LED should blink at 8Hz frequency.
- The blinking duty cycle should be 50%
  - E.g. for 2Hz frequency, the LED should be on for 1/4th of a second, then off for next 1/4th of a second and so on.
- You may use the on-board LED and push switch for this task.

Task 2: Blinking 8 LEDs one after another

Extend the Task1 with another switch which activates the blinking to loop through all 8 LEDs one after another.

- When the system starts, LED 0 is active and blinks at 2Hz.
- As long as switch 1 is pressed, the currently active LED blinks at 8Hz. Otherwise it blinks at 2Hz.
- As long as switch 2 is pressed, the currently active LED keeps shifting towards left at the frequency depending upon the position of switch 1, and starts from 0 again.
  - E.g. if LED 0 is active currently, pressing switch 2 shifts the blinking to LED 1, 2, 3, … , 7 and then again LED 0 and so on.
- When switch 2 is released, the last active LED should keep blinking without anymore shifting.

Task 3: Changing LED Mode using UART

Extend Task 1 such that the blinking frequency of the LED can be switched between 2Hz and 8Hz depending upon the string entered from the Terminal.

- The LED starts blinking at 2Hz
- After every 10 seconds, the program prints the message on terminal:
  "Do you want to change the LED mode? (Yes/No)"
- If the user enters "Yes", the LED blinking rate switches to the other frequency
  - E.g. if currently the frequency is 2Hz then it switches to 8Hz and vice versa
- If user enters "No" then the frequency stays the same.
- You may use the on-board LED for this task.