

EE4144 – Intro to Embedded Systems

Lab Exercise 1

Introduction to Arduino IDE

Debugging the Blink Program

Objective:

This exercise focuses on implementing a simple embedded program on the Atmel328P microcontroller using both sketches and ANSI C. The primary objective is to demonstrate the differences in programming languages and understanding the advantages and disadvantages of each approach. The exercise concludes with examples of how to use the Arduino IDE to help debug source code.

Setup:

1. Download the latest Arduino IDE from <http://arduino.cc/en/Main/Software>. This is the Java program used to create, write, compile, and upload the embedded software.
2. Download the latest Serial-to-USB drivers from <http://www.ftdichip.com/Drivers/VCP.htm>. These drivers allow for the RS-232 serial communications via your computer's USB port. Once installed, they create a virtual serial port, so the IDE is able to connect to the Arduino hardware via a physical USB cable. The driver converts USB communication to and from a serial port, so the IDE sends and received RS-232 data as though you had a serial port on your computer. The Arduino circuit board includes an FTDI chip that performs a similar conversion between USB communications and RS-232 serial communications for the ATmega328P microcontroller.
3. Then:
 - Mac OSX: Install the USB drivers. Then connect the development board via the USB cable.
 - Windows: Connect the development board via the USB cable. When prompted, install the USB drivers.

4. The green power LED should light up. Additionally, if you have a new board, the yellow LED should blink due to the pre-loaded application stored in the flash.
5. Run Arduino IDE:
 - Mac OSX: Copy the Arduino IDE application to the Application directory. Launch the Arduino IDE.
 - Windows: Open the Arduino folder and launch the Arduino IDE.

→(0) Demonstrate to the instructor that the Arduino IDE opens successfully and the Arduino is correctly connected to the laptop.

Running the Blink Sketch:

1. Open the example “Blink” program by selecting File-> Examples-> Digital-> Blink.
2. You should have the following source code listed in an editor pane. Note that text in between `/*` and `*/` is a comment so it is not important information regarding the program; that is, it is not actually part of the code used by the processor, and so it is omitted here. Also, all text on the same line following `//` is considered a comment and so is omitted here.


```
int ledPin = 13;
void setup ()
{
    pinMode(ledPin, OUTPUT);
}

void loop ()
{
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

Regarding this example, all programs built in the Arduino IDE include the two functions **setup()** and **loop()**. As the labels are meant to imply, the **setup()** function is run once when the program begins, so initialization

would take place here, while the **loop()** function executes as an infinite loop until the processor is reset. This example includes a global variable **ledPin**. When the program is compiled on the host computer, this particular location in the target memory is initialized with the value 13 because the Arduino circuit board's 13th I/O pin has the yellow LED labeled "L" on the board attached to it. When **setup()** is run at the start of the program, a single library call is made to make the 13th I/O pin, referenced via that named variable, as an output pin, so that the processor can drive the LED either high or low, hence turning the LED on or off. Within the **loop()** function, the LED is turned on, and then a fixed delay of 1000 milliseconds is executed before the LED is turned off, and another delay.

Notice that none of the functions presented in this example are either assembly nor are they ANSI C; instead, they are all part of the Arduino library of functions.

3. With the 'Blink' program in the editor, press "play" icon in order to compile the program (the tool refers to this as "Verify"). 
4. To upload the program to the embedded processor, first make sure you have the correct development board by checking under Tools -> Board. It is most likely set to the correct value of Arduino UNO or Nano w/ ATmega328. If you have a different board, you should adjust accordingly.
5. Next, connect to the proper serial port under Tools -> Serial Port
 - MacOSX: /dev/tty.usbserial-(something)
 - Windows: USB Serial (COMx); check device managerIn either case, the driver you installed earlier is allowing the development environment to communicate to the embedded processor with RS-232 serial communications via the USB cable.
6. Press the penultimate icon, the tool refers to this as "Upload". You should see the Arduino reset and the Rx and Tx yellow LEDs flash as the program is sent to the target flash memory. When it finishes, you should see the LED blink at one second rate. Make sure no errors appear in the lower window of the Arduino IDE.

→(1) Demonstrate the successful upload to the instructor.

7. Now that you have compiled and uploaded your first program, let's convert it from the easy-to-use-yet-non-standard library calls to easy-once-

you-know-them-and-portable ANSI C instructions. Change the ‘Blink’ program to the following.

```
void MyDelay (unsigned long mSecondsApx);

void setup ()
{
    unsigned char *portDDRB;
    portDDRB = (unsigned char *) 0x24;
    *portDDRB |= 0x20;
}

void loop ()
{
    unsigned char *portB;
    portB = (unsigned char *) 0x25;
    *portB |= 0x20;
    MyDelay(1000);
    *portB &= 0xDF;
    MyDelay (1000);
}

void MyDelay (unsigned long mSecondsApx)
{
    volatile unsigned long i;
    unsigned long endTime = 1000 * mSecondsApx;

    for (i = 0; i < endTime; i++);
}
```

→(2) Demonstrate the successful upload of the new Blink program

→(3) Explain what each line of setup() and loop() does.

→(4) Why is “|=” used in one line of loop(), but “&=” is used in the other?

→(5) Explain exactly what MyDelay() does.

Further Experiments:

1. Modify the provided example Blink sketch to cause the LED to blink in the following pattern:
 - a. Blink once
 - b. Pause for one second
 - c. Blink twice
 - d. Pause for one second
 - e. Blink 3 times
 - f. Pause for one second and then start over.

→ (6) Show the instructor your code.

2. What is the total number of bytes required by your program above. Obtained this value after you compile your program in the bottom window of the editor.
3. Now modify the ANSI C version of the Blink program to cause the LED to blink in the pattern described in 1.

→ (7) Show the instructor your code.

4. What is the total number of bytes required by your program above (in 3).
5. What can you conclude about the required bytes for each program?

Debugging:

Run-time debugging often utilizes the **Serial.print()** and **Serial.println()** functions in Arduino. Modify the Blink Arduino sketch to include the following in the **setup()** function:

```
Serial.begin(9600);
```

Now modify the **loop()** function to the following:

```
void loop()
{
    digitalWrite(led, HIGH);
    Serial.println("LED ON");
    delay(1000);
    digitalWrite(led, LOW);
    Serial.println("LED OFF");
    delay(1000);
}
```

1. Download the modified blink sketch to the Arduino. After a successful download, confirm that the LED “L” is blinking. Now, open the serial monitor in the Arduino IDE by pressing the icon on the upper right corner of the IDE.



2. Now observe the output in the serial window.

→(8) Show the instructor the serial output window.

3. Practice *compile-time* debugging by fixing all of the syntax errors in the following listing. Use the Arduino IDE compile output information to help identify the errors. Note the required changes for the program to compile successfully.

```
void MyDelay (unsigned long mSecondsApx);
```

```
void setup ()  
{
```

```
    unsigned char  *portDDRB;
```

```
    portDDRB = (unsigned char *) 0x24;
```

```
    *portDDRB |= 0x20;
```

```
}
```

```
void loop ()  
{
```

```
    unsigned char  *portB;
```

```
    portB = (unsigned char *) 0x25;
```

```
    *portB |= 0x20;
```

```
    MyDelay{1000}
```

```
    portB &= 0xDF;
```

```
    MyDelay [1000],
```

```
}
```

```

void MyDelay (unsigned long  mSecondsApx)
{
    volatile unsigned long  i;
    unsigned long endTime = 1000 * mSecondsApx;

    for (i = 0; i < endTime; i++);
}

```

→(9)Show the instructor the successful compile

4. Below are three ANSI C programs (A., B., and C.). Practice run-time debugging by fixing each so that the LED will blink correctly (once per second).

Program A.

```

void NewDelay (unsigned char mSecondsApx);

void  setup()
{
    unsigned char *portDDRB;

    portDDRB = (unsigned char *) 0x24;

    *portDDRB |= 0x20;
}

void loop ()
{
    unsigned char *portB;
    portB = (unsigned char *) 0x25;
    *portB |= 0x20;
    NewDelay (100);
    *portB %= 0xDF;
    NewDelay (100);
}

void NewDelay (unsigned char mSecondsApx)
{
    volatile unsigned char i;
    unsigned long endTime = 1000 * mSecondsApx;
    for (i = 0; i < endTime; i++);
}

```

Program B.

```
void NewDelay (unsigned long mSecondsApx);
void setup()
{
    unsigned char *portDDRB;
    portDDRB = (unsigned char *) 0x24;
    *portDDRB |= 0x20;
}
void loop ()
{
    unsigned char *portB;
    portB = (unsigned char *) 0x25;
    *portB |= 0x20;
    NewDelay (100);
    *portB &= 0xDF;
    NewDelay (100);
}
void NewDelay (unsigned long mSecondsApx)
{
    volatile unsigned long i;
    unsigned char j;
    unsigned long k;
    unsigned long endTime = 100 * mSecondsApx;

    for (i = 0; i < endTime; i++)
    {
        j = 10;
        do
        {
            j = j - i;
            k = i/j;
        } while (k>0);
    }
}
```


Program C.

```
void NewDelay (unsigned long mSecondsApx);
void setup()
{
    unsigned char *portDDRB;

    portDDRB = (unsigned char *) 0x24;

    *portDDRB |= 0x20;
}
void loop ()
{
    unsigned char *portB;

    portB = (unsigned char *) 0x25;

    *portB |= 0x20;
    NewDelay (100);
    *portB &= 0xDF;
    NewDelay (100);
}
void NewDelay (unsigned long mSecondsApx)
{
    volatile unsigned long i;
    unsigned char j = 0;
    unsigned long endTime = 100 * mSecondsApx;
    i = 0;
    while (j = 0)
    {
        i++;
        if (i = endTime)
        {
            j = 1;
        }
    }
}
```

→(10) Explain to the instructor the changes made to Program A.

→(11) Explain to the instructor the changes made to Program B.

→(12) Explain to the instructor the changes made to Program C.