LAB #1—Intro to the lab tools

Kizito NKURIKIYEYEZU

September 26, 2021

1 LAB OBJECTIVES

- To be familiar with some the simulation software: PROTEUS VSM
- To learn how to create firmware using Microchip Studio
- How to simulate embedded system using PROTEUS and Microchip Studio
- To examine the I/O port operation using a simulator.
- Review C programming/compiling.
- Learn how to address ports/pins and set i/o configuration.
- Masking, toggling, if-then, subroutines, and looping.
- To create and debugging firmware using Microchip Studio
- To examine the I/O port operation using a simulator.
- Learn how to address ports/pins and set I/O configuration.
- Understand I/O pin drive and input levels.

2 INTRODUCTION TO MICROCHIP STUDIO

3 Introduction

This tutorial will teach you how to write, compile, and trace a simple program in Microchip Studio. Microchip Studio is an Integrated Development Environment based on Microsoft's Visual studio IDE. It is used for developing and debugging all AVR and SAM microcontroller applications. The Microchip Studio gives you an environment to write, build and debug your applications written in C/C++ or

assembly code. It also connects to the debuggers, programmers and development kits that support AVR and SAM devices.

The IDE was initial developed by Atmel corporation but since 2016, Microchip Technology acquired the Atmel Corporation and, for all practical purses, Atmel studio is deprecated and replaced by Microchip Studio. We will use this IDE for this lab while I am still reevaluating a better IDE to use[1].

4 Installing and using Microchip Studio

1. Download the newest version of Microchip Studio[2]
2. Run the downloaded program to install the Atmel Studio IDE.
3. Create a new project as shown in this Youtube video[3]
4. Enter the C code in Listing 1. It is not important to understand the code at this stage, but you can do that by reading the C comments.

```
#include <avr/io.h>
#define F_CPU 1000000UL  //XTAL frequency = 1 MHz
#include <util/delay.h>
#define DDB0 0
#define PB0  0
int main(void){
  // Set the 1st bit on PORTB (i.e. PB0 to 1) to an output
  DDRB |= (1<<DDB0);
  // Infinite loop (loop until manually stopped)
  while(1)  {
    // Toggle the 1st bit on PORT B (i.e. PB0)
    PORTB ^= (1<<PB0);
    // Wait for one second
    _delay_ms(1000);
  }
  //Note: he program should never reach this point.
  return 0;

}
```

Listing 1: Blink an LED connected on PB0 on an AVR MCU

[1] There exist many IDEs for embedded development. Prominently, PlatformIO IDE provides a new generation toolset for embedded C/C++ development and has an easy to use plugin for the incredibly popular Microsoft's Visual Studio Code. Alternatively, you can use MPLAB X IDE—albeit I find it cranky compared to atmel studio

[2] https://www.microchip.com/content/dam/mchp/documents/parked-documents/as-installer-7.0.2542-web.exe

[3] https://youtu.be/ySwev9nLkBE

5. Watch this video to understand how to use microchip studio to program and debug your code.

5 Compiling C code to HEX file

- Read this document for a step by step guide on how to program AVR microcontroller in Atmel Studio 7. The document is based on the previous version of Atmel studio, but the information has not changed much.
- Alternatively, you can watch this video to understand how to compile your code using microchip studio.

6 Hardware simulation with PROTEUS

6.1 What is Proteus?

Proteus contains everything you need to develop, test and virtually prototype your embedded system designs based around the Atmel AVR series of microcontrollers. The unique nature of schematic based microcontroller simulation with Proteus facilitates rapid, flexible and parallel development of both the system hardware and the system firmware. This design synergy allows engineers to evolve their projects more quickly, empowering them with the flexibility to make hardware or firmware changes at will and reducing the time to market.

Proteus VSM models will fundamentally work with the exact same HEX file as you would program the physical device with, binary files (i.e. AVR Hex files) produced by any assembler or compiler.

6.2 Get started to proteus

- Install proteus VSM (The detailed instructions will be given to you later)
- Select create a new Project
  - This wizard guides you through the setup of your Proteus project.
  - There is a start page in which you specify the project name and destination directory and then potentially three main screens for schematic, PCB and firmware.
  - Select create schematic by checking the box at the top of the screen and then select the default template, then press next

- The next screen asks you to create a PCP. Just don't create on for the moment. Press next
- Add the ATmega168 the schematic , by pressing P. You will then be presented with a window that allows you to add components. For example, to:
  - To add the ATmega168 microcontroller, just search for "ATmega168"
  - To add a red led, just search for "red led"
  - To add a 1k resistor, just search for "1k resistor"
  - To add a button switch, just search for button
  - The library contains many component. Just search for what you're looking for.
- Build the circuit
  - Add the ATmega168 microcontroller
  - Add a red LED and rename it to LED1
  - Connect the LED to pin PB0 of the microcontroller
  - Add a current limiting resistor, and change its value to 240 Ohms
  - Add a ground

6.3 Running the simulation in Proteus

- Watch the following Youtube videos
  - https://youtu.be/8XRIhIvnewA
  - `https://youtu.be/h13p8s_32Eg`
- Right click the ATtiny13 microcontroller and select edit properties (Figure 1). The browse for the hex file created with Atmel studio. Note: The hex file is usually located in the debug folder or in the release folder depending on how you compiled your program.
- Select the hex file, and the press OK to dismiss the property screen
- Run the simulation
- Now, press the run button. The LED should blink
- Congratulation! Your systems works as expected!

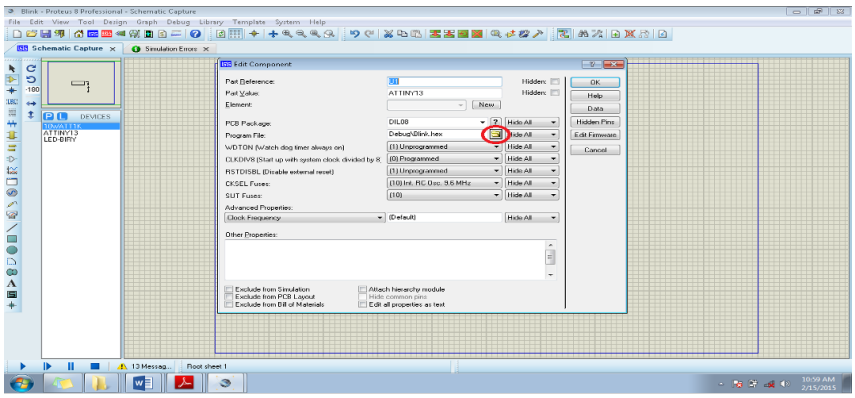


FIGURE 1. Specifying the hex file that will run on the microprocessor

7 LAB EXERCISE

7.1 Background and description

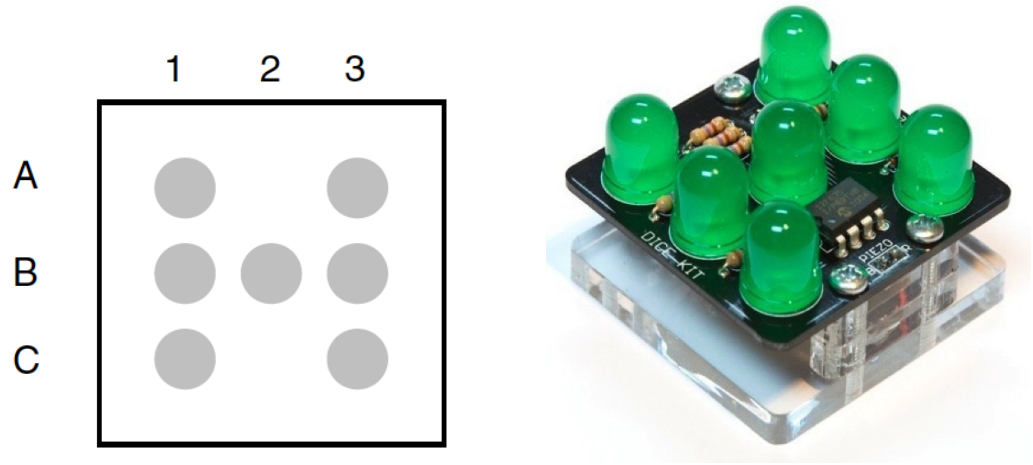
In this exercise, you will design and build an electronic dice similar to this one[4].

The decide will functions as follow:

- The device consist of ATtiny2313, seven LEDs, and two switches
- The device should have an input switch, and upon pressing the switch, a number (selected randomly by the controller) between 1 and 6 should be displayed. As shown in Figure 2, Figure 4 and Figure 3 the LEDs are organized such that when they turn ON, they indicate numbers as on a real dice.
- The device has two switches:
  - RESET_SWITCH —when the reset switch is pressed, the device turn off all LED
  - ROLL_SWITCH —when the roll switch is pressed, the display should go off momentarily before displaying a random number. This gives feedback to the user that the switch press was recognized by the processor. In the absence of this blanking feature, if the user presses a switch and the next number happens to be the same as the last one, the user may not recognize that.
- The dice should be very compact and small. Thus, use a small microcontroller. For example, the ATtiny2313 will suffice.
- You should follow the industry accepted embedded coding standard[5]

[4] https://www.spikenzielabs.com/learn/dicekit.html

- The final circuit of the device is shown in Figure 5. Calculate the values of resistors that are required and draw, test and virtually prototype the device in proteus.



(a) Dice LED arrangment (b) Real-world implementation

FIGURE 2. Output LED arrangement for the dice.

7.2 Program Pseudocode

The operation of the dice is described in Algorithm 1. At the beginning of the program PORTC pins are configured as outputs and bit 0 of PORTB (RB0) is configured as input. The program then executes in a loop continuously and increments a variable between 1 and 6. The state of the push-button switch is checked and when the switch is pressed (switch output at logic 0), the current number is sent to the LEDs.Asimple array is used to find out the LEDs to be turned ON corresponding to the dice number. You can use the AVR Libc to generate the random number. Please refer to its random function, which is described here.
NOTE: Please read carefully how this function works and figure out how you can use its output to generate a number between 0 and 6. Alternatively, you can generate pseudo random number based on how the switch is pressed. Use your imagination!

[5] https://barrgroup.com/sites/default/files/barr_c_coding_standard_2018.pdf

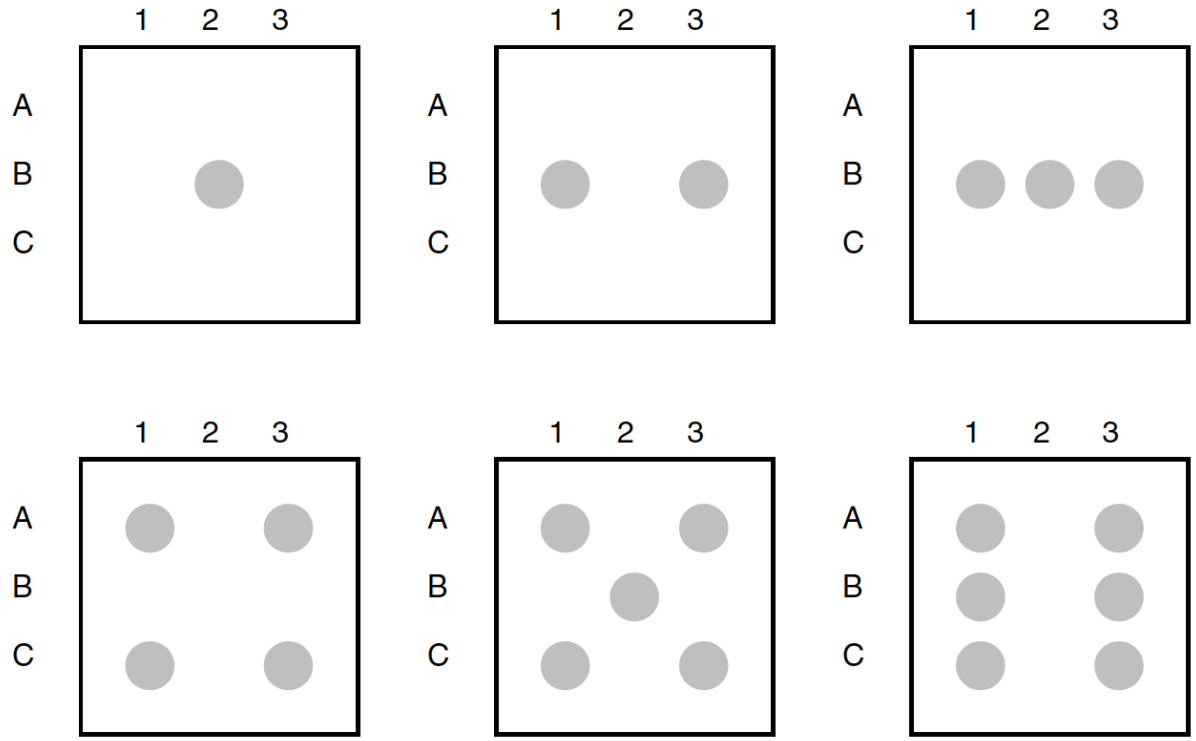


FIGURE 3. LEDs light up in this fashion for the numbers 1 to 6.

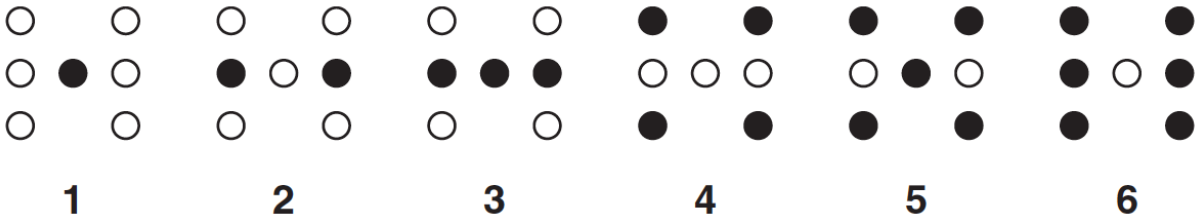


FIGURE 4. LED layout arrangement and the corresponding digits

8 LAB REPORT

The lab report should be written as if the reader was another member of the class. With your report in hand, one should be able to reproduce what you did and understand it well enough to add to it. The report shall contain the following sections

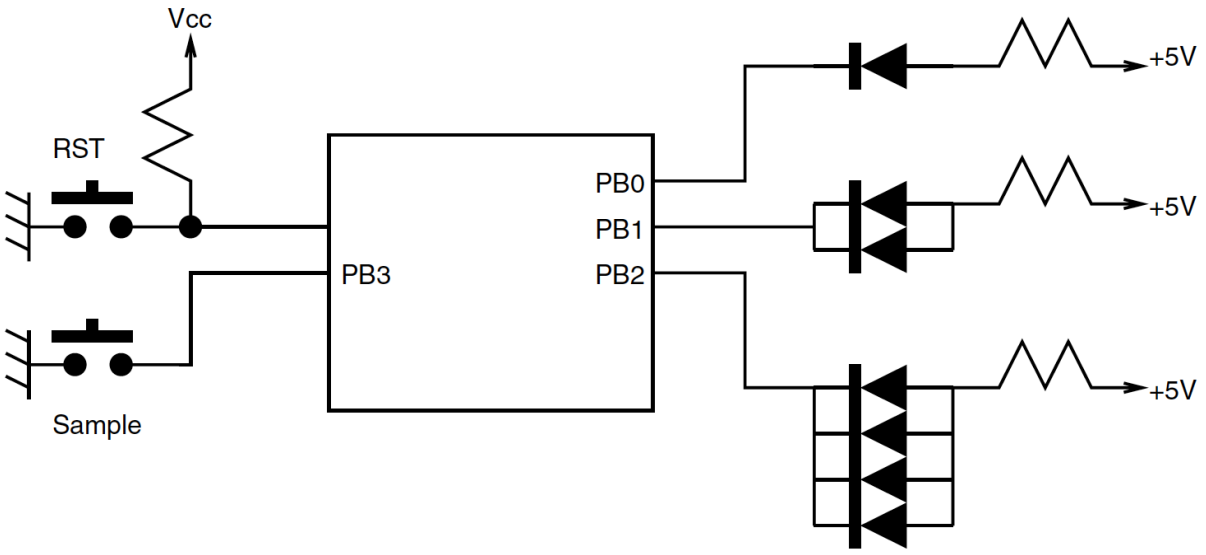


FIGURE 5. Block diagram for the electronic dice using an ATtiny2313 MCU

Algorithm 1: Program Pseudocode to control the dice

```
Input:
    • RESET_SWITCH —when the reset switch is pressed,
      the device turn off all LED
    • ROLL_SWITCH —when the roll switch is pressed, a new
      random number is displayed
Initialization
    • All pins connected to the switches are initialized as inputs
    • All pins connected to the LEDs are initialized as outputs
    • Turn off all LEDs
end
Output:  A random number between 1 and 6 displayed on the LED
Loop
    if RESET_SWITCH==PRESSED then
        Turn OFF all LEDs
    if ROLL_SWITCH==PRESSED then
        Wait for one second
        Generate a new random number
        Turn ON the LEDs corresponding to the random number
EndLoop
```

- Abstract —1 paragraph describing what you did. It should explain the purpose of the lab, results, its scope, summary of your solution and a conclusion.

- Introduction —What is the lab about project? Why is it interesting? What technical details does your reader need to understand?
- Circuit description Draw the circuit diagram. Please note that you should note use the Proteus diagram. Instead, you should manually draw a separate diagram that shows only components and pins that are being used on the MCU. Discuss how various resistors, transistors have been selected. Show any pertinent calculation when necessary.
- Code description —Start with a very high level description of your code (block diagrams, perhaps) and how it uses subsystems... Get increasingly detailed in your description until you get to the level of short sections of code that are critical to the proper operation of your project.
- Discussion and conclusion —What worked well? What didn't? If something did not work as expected, what do you think is the reason? How could your solution be improved?

9 Lab grading criteria

- FUNCTIONALITY (60%)
  - Code does not compile .... deduct 60%
  - The simulator does not run (e.g, missing the .hex file) .. .. deduct 60%
  - The device does not work as intended .... deduct 20%
  - LEDs and/or switches on the schematic not properly arranged as shown in Figure 3 ..... deduct 20%
  - If the LED's bounce when the switch is pressed .... deduct 20%
  - Any missing/non-implemented specification .... deduct 20%
- CODE QUALITY AND DOCUMENTATION(20%)
  - Lack of comments, no or improper comments at the headers of the files, misleading or useless comments .... deduct 5%
  - Poor coding style ..... deduct 5%
  - Poor code organization and modularization..... deduct 10%
  - Using magic numbers .... deduct 5%
  - Messy, Unreadable Code .... deduct 5%
  - Violates major embedded C coding standard ..... deduct 5%

  - Does not use proper naming convention .... deduct 5%
  - Does not use meaningfull variable names..... deduct 5%
  - Use God functions .... deduct 5%
- LAB REPORT 20%
  - abstract—conveys a sense of the full report concisely 2%
  - introduction—effectively presents the purposes of the lab 5%
  - circuit description—clear, concise but complete 5%
  - messy circuit diagrams ..... deduct 2%
  - lack of justification of component selection (e.g., current limiting, internal or external pull-ups or pull downs) 5%
  - software description 5%
  - Conclusion 3%
- Any cheating or copying someone else's code .... deduct 100%
- TOTAL 100%

10 Lab submission

The lab is due on September 28, 2021. The submission shall go as follows:

- You should submit the report, all your code file (c or cpp files), the proteus simulation file and the .hex file in one .zip file and submit them through the e-learning platform no later than midnight on September 28, 2021.
- Before submission, please make sure that, when the simulation file is properly linked with the .hex file. I should be able to run your simulator without compiling your code.
- Please submit your work before the deadline. I will not accept any submission through my email no matter the reasons