**COMSATS University**

**Islamabad**



**Lab Report # 01**

**Real Time Embedded Systems**

**(EEE-446)**

|  |
| --- |
| **Introduction to Development Tools and Lab Software. Understanding digital I/O ports, interfacing LED and push button with Arduino Board.** |

**Submitted By:**

**Arwa Aamir (FA16-EEE-002)**

**Submitted To:**

**Dr. Ahsen Malik**

**Lab # 01**

**Introduction to Development Tools and Lab Software. Understanding digital I/O ports, interfacing LED and push button with Arduino Board.**

**Objectives**

* Learn to use software development tools such as Arduino Integrated Development Environment (IDE), and Simulator (Proteus) for Arduino Uno microcontroller board.
* Interfacing LED with Aruduino Port.
* Interfacing Push Button with and without external pull-up resistor with Arduino.

**Softwares Used**

* Arduino IDE
* Proteus (Version 8.4)

**Pre Lab**

Get the following software installers from MP/VLSI lab and install them in your laptops and lab PCs (if not installed). Also paste the Arduino simulation files for Proteus in Library folder.

**Arduino:**

[Arduino](http://arduino.cc/) is an open-source platform used for building electronics projects. It consists of both a physical programmable circuit board and an IDE (Integrated Development Environment) that runs on your computer. Most Arduino boards consist of an Atmel 8-bit AVR microcontroller with varying amounts of flash memory, pins and features. Arduino is programmed using the [Arduino](https://www.arduino.cc/en/Main/Software) [Software (IDE)](https://www.arduino.cc/en/Main/Software) which is a cross-platform application for windows, macOS and Linux. It is connected to a PC via USB cable to upload computer code to the physical board. This also provides power to the board, as indicated by a LED.

**Proteus:**

The Proteus Design Suite is a complete software solution for circuit simulation and PCB design. Proteus can simulate electric/electronic and microcontroller based circuits. It supports number of microcontrollers available in the market.

**Arduino**

Arduino is used in many educational programs around the world, particularly by designers and artists who want to easily create prototypes but do not need a deep understanding of the technical details behind their creations. Arduino is best known for its hardware, but you also need software to program that hardware. Both the hardware and the software are called “Arduino.” The combination enables you to create projects that sense and control the physical world. The software is free, open source, and cross-platform.

**Arduino Software**

Software programs, called sketches, are created on a computer using the Arduino integrated development environment (IDE). The IDE enables you to write and edit code and convert this code into instructions that Arduino hardware understands. The IDE also transfers those instructions to the Arduino board (a process called uploading).

**Arduino Hardware**

The Arduino board is where the code you write is executed. The board can only control and respond to electricity, so specific components are attached to it to enable it to interact with the real world. These components can be sensors, which convert some aspect of the physical world to electricity so that the board can sense it, or actuators, which get electricity from the board and convert it into something that changes the world. Examples of sensors include switches, accelerometers, and ultrasound distance sensors. Actuators are things like lights and LEDs, speakers, motors, and displays.

There are a variety of official boards that you can use with Arduino software but we will use the most widely used board called

Arduino UNO.

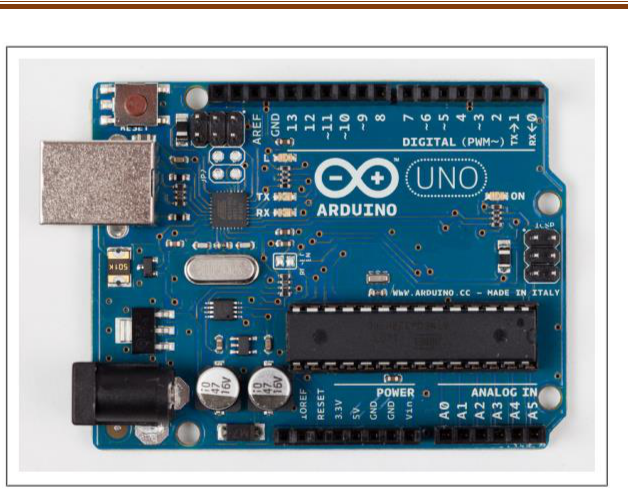
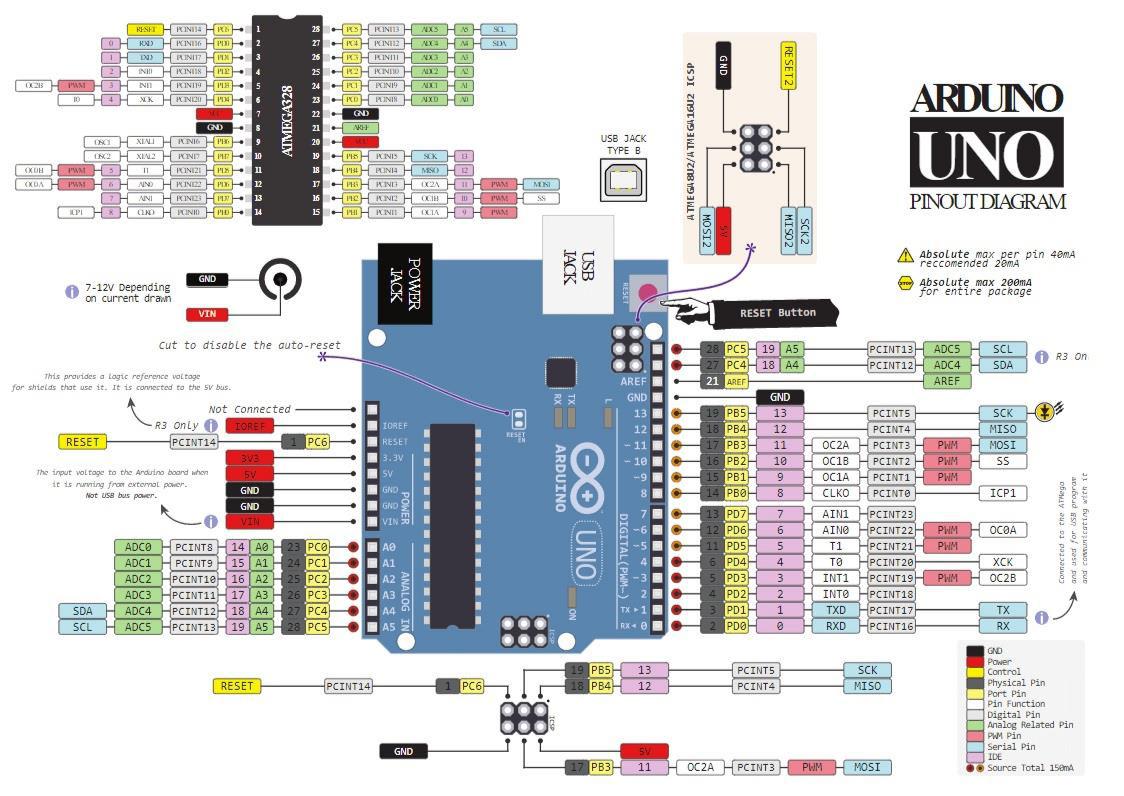
****

Figure 1.1a. Basic board: the Arduino Uno.

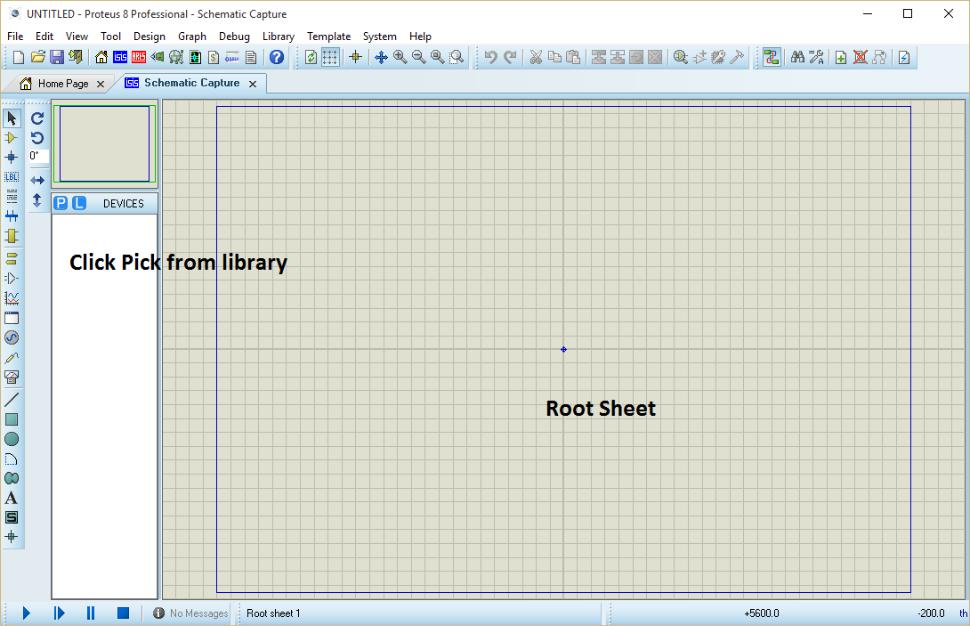
**

*Figure 1.1b. Aruduino Uno Pin Diagram*

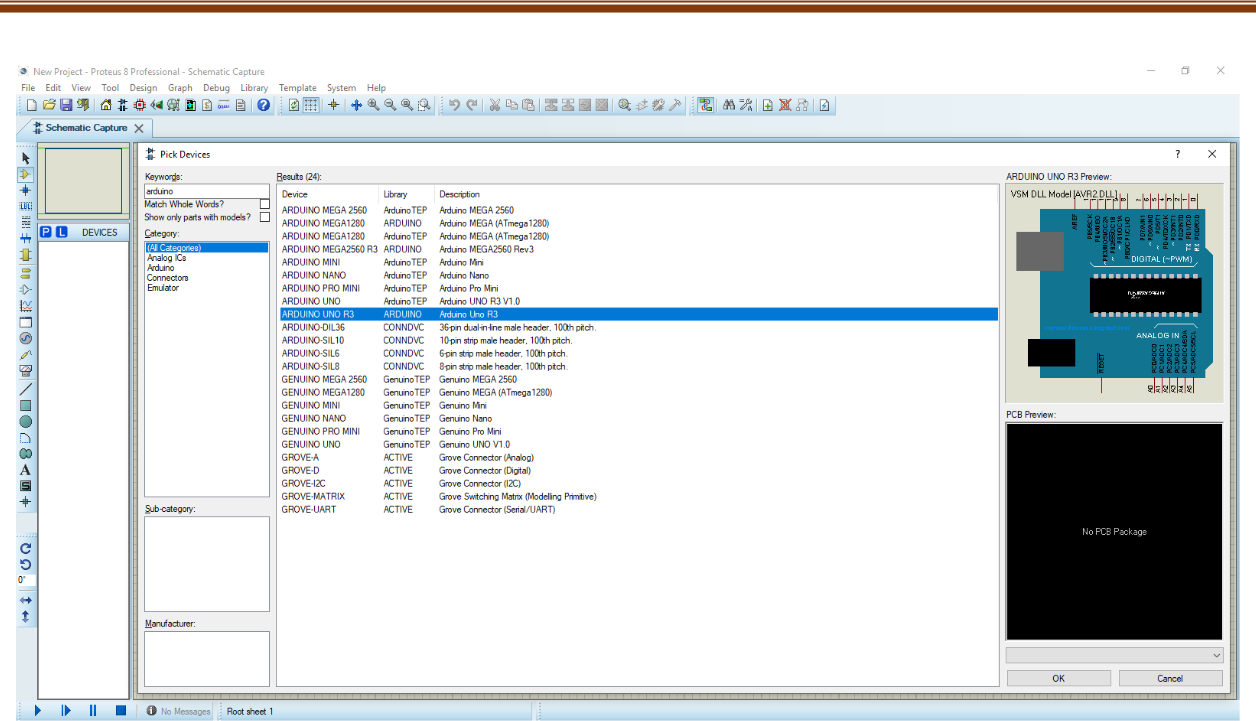
**Proteus Introductory Learning Tutorial**

To learn basics about Proteus, we will simulate a Arduino circuit in Proteus.

Launch Proteus from start menu or by desktop icon. Proteus layout shown below will appear on the screen as shown in Figure 1.2. There is a root sheet to draw circuit. The device window will initially be empty and you need to pick components from library. Proteus component library is equipped with lots of components and ICs including microcontrollers. To pick necessary components required for circuit, click on small P in device window.



*Figure 1.2: Proteus ISIS overview of window*

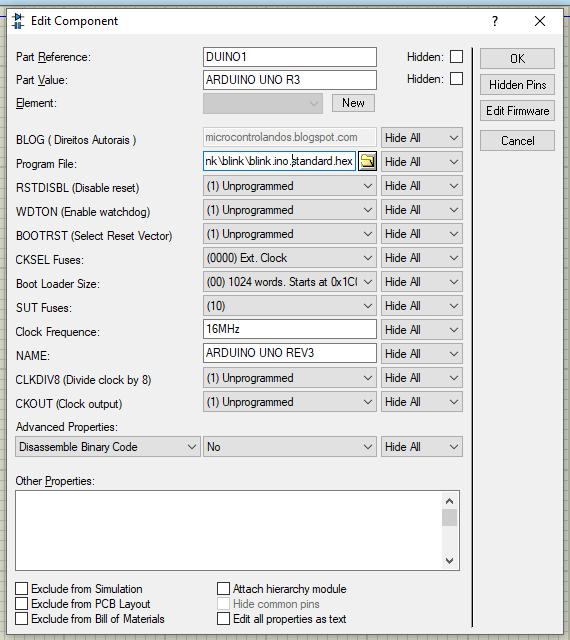
1. ****After Clicking Pick devices menu as in figure 1.3 will appear on the screen. There is a keyword prompt which can refine your search. Write Arduino in keyword prompt and result window will display components having keyword of Arduino. Double click Arduino Uno component in result window to add it to device list.

*Figure 1.3: Pick Devices for circuit diagram*

1. Following components needed for this tutorial:
   1. Arduino Microcontroller
   2. Animated LED
   3. 470/220 ohms resistor
   4. Push Button
2. To add any component on root sheet, simply select that component from device window and click anywhere in the root sheet to place that component. To draw wire connection between two pins of any components, simply move mouse cursor to the tip of the pin and click. Then click on tip of the other pin you wanted to connect.
3. On the left side of Proteus layout, different modes can be selected. To add ground terminal to the circuit, click terminal mode from buttons on left side of screen and then select ground terminal. You can change characteristic properties of any component. To change, simply double click the required component. An edit component menu will appear for that particular component where you can change the characteristic properties of that component.
4. You also need to load your required program into the microcontroller memory as shown in Figure 1.15. Program is loaded into microcontroller memory through hex file. Double click microcontroller, edit menu will have prompt for load program. Select the hex file generated

from Atmel Studio tutorial. This hex (Lab1.hex) file can be found in the Lab1 directory you created on desktop for the Atmel Studio tutorial.

1. The default frequency of Arduino Uno is 16MHz. Select .hex file from Program File menu and load your code in Arduino.



*Figure 1.4: Edit component menu*

1. Now run the simulation from the bottom of Root Sheet and show result to your lab instructor.

**In-Lab**

**In-Lab Task 1: Interface led with Arduino hardware and program and simulate it to blink.**

1. Launch Arduino IDE.
2. Write your code in Arduino IDE.
3. Use tick button to compile the program.
4. From sketch menu, click ‘use compiled binary’ to create the .hex file of program.
5. Simulate the code in Proteus.

|  |  |
| --- | --- |
| **SINK MODE** | |
| ARDUINO IDE CODE | PROTEUS SCHEMATIC |
| void setup() {  pinMode(12, OUTPUT);  }  void loop() {  digitalWrite(12, HIGH);  delay(1000);  digitalWrite(12, LOW);  delay(1000);  } |  |
| **SOURCE MODE** | |
| ARDUINO IDE CODE | PROTEUS SCHEMATIC |
| void setup() {  pinMode(12, OUTPUT);  }  void loop() {  digitalWrite(12, HIGH);  delay(1000);  digitalWrite(12, LOW);  delay(1000);  } |  |

**In-Lab Task 2: Push Button Interfacing with Arduino Uno**

Interface push button at pin 2 of Arduino Uno with external pull-up resistor. When push button is pressed the led at pin 12 in on and vice versa.

|  |  |
| --- | --- |
| **EXTERNAL PULLUP CODE** | |
| ARDUINO IDE CODE | PROTEUS SCHEMATIC |
| bool inp;  void setup() {  pinMode(2,INPUT);  pinMode(12, OUTPUT);  }  void loop() {  inp=digitalRead(2);  if (inp== HIGH)  {  digitalWrite(12,LOW);  }  else  {  digitalWrite(12,HIGH);  }  } |  |

**In-Lab Task 3: Push Button Interfacing without external pull-up with Arduino Uno**

|  |  |
| --- | --- |
| **INTERNAL PULLUP CODE** | |
| ARDUINO IDE CODE | PROTEUS SCHEMATIC |
| bool inp;  void setup() {  pinMode(2,INPUT\_PULLUP);  pinMode(12, OUTPUT);  }  void loop() {  inp=digitalRead(2);  if (inp== LOW)  {  digitalWrite(12,LOW);  }  else  {  digitalWrite(12,HIGH);  }  } |  |

**Post Lab Task 1:**

1. **Explain what is button debouncing effect.**

|  |
| --- |
| Switch bounce or contact bounce or even called as chatter is a common problem associated with mechanical switches and relays. Switch, relay contacts are made up of spring metals which are forced to contact each other by an actuator. While they collide each other there is  a possibility of rebounding for some time before they make a stable contact. As a result of this effect there will be on/off transitions generated as the contacts rapidly open and close.  In short **Switch bounce** is a non-ideal behavior which generates multiple transitions for a single user input.    This effect is not taken as important while doing power circuits or electrical circuits but creates a major problem while we deal with logic circuits. (i.e. counters).  So here we have to remove bounces. And the method to get rid of such bounces is called **Switch Debouncing.**  It can be implemented in 4 ways:   1. Hardware Debouncing 2. R-C Debouncing 3. Software Debouncing 4. Using Debouncing IC’s |

1. **How to cater button debouncing effect in our code.**

|  |
| --- |
| We can cater this affect in our code by introducing a delay of 20ms-200ms (depending upon the switch). Whenever the button is pressed and gives HIGH state, the delay is introduced, after the delay has ended, the button state is read again, if it is still HIGH then the output is HIGH, otherwise it is regarded as an interference and the output retains its LOW state. |

1. **Give hardware solution to cater button debouncing effect.**

|  |
| --- |
| Hardware debouncing can be done in two ways:   1. Using SR Latches 2. R-C Debouncing 3. **SR Latches**   This debouncing technique uses an S-R latch to avoid bounces in the circuit along with the pull up resistors. S-R circuit is most effective of all debouncing approaches    The circuit uses two cross coupled NAND gates which form an S-R latch, A SPDT (Single Pole Double Throw) switch, two pull up resistors. The resistor generates a logic ‘one’ for the gates, Switch pulls one of the inputs to ground.  If  the switch is in position as shown in figure  the output of the upper gate is ‘1’ irrespective of the input of the other gate and the one created by the bottom pull up resistor which drives the lower NAND gate to zero which in return races back to the other gate. If the switch moves back and forth between the contacts and is suspended for a while in neither region between the terminals, the latch maintains its state because ‘0’ from the bottom NAND gate is fed back. The switch may move between the contacts but the latch’s output ensures it never bangs back and thus switch is bounce free.  **Drawbacks:**  The S-R circuit is common but the bulkiness of the circuit causes it to be used rarely also SPDT switches are costlier than SPST (Single Pole Single Throw) switch.   1. **R-C Debouncing**   Another method of debouncing is to use a R-C circuit. The basic idea behind such circuit is to use a capacitor to filter out quick changes in the switch signal.    The basic R-C circuit used for debouncing is shown above. The circuit uses two Resistors, Capacitor, Schmidt trigger hex inverter (eg : 7414) , SPST switch.   * If  the switch is open, the voltage across capacitor which is initially zero now charges to Vcc through the R1 & R2. The voltage at Vin is high  hence the output of the inverting Schmitt trigger is low ( logic 0 ) * If the switch is closed the capacitor discharges to zero hence the voltage at Vin is ‘0’  and output of the inverting Schmidt trigger is high ( logic 1 )   During the bouncing condition the capacitor will stop the voltage at Vin when it reaches either Vcc or Gnd.  We can’t use the standard inverter gate here. TTL defines a zero input when the applied voltage is in between 0 and 0.8. The output becomes unpredictable in some situations. So we use a Scmitt trigger hex inverter. The output remains stable even if the inputs vary or dither also it prevents the output from switching because of its hysteresis property. |

**Post Lab Task 2:**

1. **Give demo on hardware of all the in-lab tasks and post-lab task.**

|  |
| --- |
| To be demonstrated to the examiner on hardware. |

